T.C.
BALIKESIR ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
BİLGİSAYAR VE BİLİŞİM MÜHENDİSLİĞİ ANABİLİM DALI

MAKİNE ÖĞRENMESİ İLE ADRES ÇÖZÜMLEME PROTOKOLÜ SAHTECİLİĞİNİN TESPİTİ

MUSTAFA FURKAN CEYLAN

YÜKSEK LİSANS TEZİ

Jüri Üyeleri : **Prof. Dr. Selçuk KAVUT (Tez Danışmanı)**
Prof. Dr. Kemal FİDANBOYLU
Doç. Dr. Fatih AYDIN

BALIKESİR, KASIM - 2024

ETİK BEYAN

Balıkesir Üniversitesi Fen Bilimleri Enstitüsü Tez Yazım Kurallarına uygun olarak tarafımca hazırlanan "**Makine Öğrenmesi ile Adres Çözümleme Protokolü Sahteciliğinin Tespiti**" başlıklı tezde;

- Tüm bilgi ve belgeleri akademik kurallar çerçevesinde elde ettiğimi,
- Kullanılan veriler ve sonuçlarda herhangi bir değişiklik yapmadığımı,
- Tüm bilgi ve sonuçları bilimsel araştırma ve etik ilkelere uygun şekilde sunduğumu,
- Yararlandığım eserlere atıfta bulunarak kaynak gösterdiğimi,

beyan eder, aksinin ortaya çıkması durumunda her türlü yasal sonucu kabul ederim.

Mustafa Furkan CEYLAN

ÖZET

MAKİNE ÖĞRENMESİ İLE ADRES ÇÖZÜMLEME PROTOKOLÜ SAHTECİLİĞİNİN TESPİTİ
YÜKSEK LİSANS TEZİ
MUSTAFA FURKAN CEYLAN
BALIKESİR ÜNİVERSİTESİ FEN BİLİMLERİ ENSTİTÜSÜ
BİLGİSAYAR VE BİLİŞİM MÜHENDİSLİĞİ ANABİLİM DALI

(TEZ DANIŞMANI: PROF. DR. SELÇUK KAVUT)

BALIKESİR, KASIM- 2024

Adres Çözümleme Protokolü (Address Resolution Protocol - ARP), ağlardaki cihazların IP adreslerini MAC adreslerine dönüştürmek için kullanılan temel bir ağ protokolüdür. Ancak, bu protokolün doğası gereği güvenlik açısından zayıf olması, kötü niyetli kişiler tarafından kolayca manipüle edilmesine ve ARP sahteciliği (ARP spoofing) gibi saldırılara yol açabilmektedir. ARP sahteciliği, ağ güvenliği açısından büyük risk teşkil eden ve hassas verilerin çalınmasına, ağ trafiğinin yönlendirilmesine ve veri bütünlüğünün bozulmasına neden olabilen kritik bir saldırıdır. Bu tezde, ARP sahteciliğinin tespitinde çeşitli makine öğrenmesi ve derin sinir ağları (Deep Neural Networks - DNN) algoritmalarının kullanımına yönelik bir çalışma gerçekleştirilmiştir. Özel olarak, bu çalışmada, yakın zamanda Alani ve ark. tarafından önerilen DNN modelinin kullanıldığı ARP sondası (APR Probe) sistemi gerçeklenmiş ve bağımsız olarak elde ettiğimiz başarım sonuçları, bahsedilen çalışmada bulunan sonuçlarla karşılaştırılmıştır. Aynı çalışmada, DNN modelinin tasarımında Nesnelerin İnterneti Ağ Saldırısı Veri Seti (Internet of Things Network Intrution Dataset – IoT-ID) kullanıldığı belirtilmiş olmakla birlikte, çalışmamızda bu veri setinin tamamının kullanılmadığı gözlenmiş ve tamamının kullanılması sonucunda başarım metriklerinin daha düşük olduğu bulunmuştur. Ayrıca, Karar Ağacı, Rastgele Orman, K-En Yakın Komşu ve Lojistik Regresyon gibi makine öğrenmesi yöntemleri kullanılarak performansları DNN ile karşılaştırılmıştır. Bunun bir sonucu olarak, Rastgele Orman ve Karar Ağacı algoritmaları en yüksek doğruluk (sırasıyla, %95.82, %95.94) ve keskinlik (sırasıyla, %93.24, %93.77) değerlerine ulaşarak en iyi performansı göstermiştir. DNN uygulamaları ise özellikle duyarlılık açısından %99.72 ile en yüksek sonucu vermiştir.

ANAHTAR KELİMELER: ARP sahteciliği, makine öğrenmesi, derin sinir ağları, rastgele orman, karar ağacı, sızma testleri, siber güvenlik
Bilim Kod / Kodları: 92403,92432 Sayfa Sayısı : 110

ABSTRACT

ADDRESS RESOLUTION PROTOCOL SPOOFING DETECTION WITH MACHINE LEARNING
MSC THESIS
MUSTAFA FURKAN CEYLAN
BALIKESIR UNIVERSITY INSTITUTE OF SCIENCE
COMPUTER AND INFORMATION ENGINEERING

(SUPERVISOR: PROF. DR. SELÇUK KAVUT)

BALIKESİR, NOVEMBER - 2024

The Address Resolution Protocol (ARP) is a fundamental network protocol used to translate IP addresses into MAC addresses for devices on a network. However, the inherent security weaknesses of this protocol make it susceptible to manipulation by malicious actors, leading to attacks such as ARP spoofing. ARP spoofing is a critical attack that poses significant risks to network security, potentially resulting in the theft of sensitive data, redirection of network traffic, and compromise of data integrity. This thesis presents a study on the use of various machine learning and deep neural network (DNN) algorithms for detecting ARP spoofing. Specifically, this study implements the ARP Probe system, which uses a DNN model recently proposed by Alani et al. and independently compares our performance results with those found in the referenced study. Although the referenced study indicates that the Internet of Things Network Intrusion Dataset (IoT-ID) was used in the design of the DNN model, our study observed that the entire dataset was not utilized in their approach. By employing the complete dataset, our findings revealed that the performance metrics were lower when the entire dataset was used. Additionally, machine learning methods such as Decision Tree, Random Forest, K-Nearest Neighbors, and Logistic Regression were also applied and compared with DNN. As a result, Random Forest and Decision Tree algorithms achieved the highest accuracy (95.82% and 95.94%, respectively) and precision (93.24% and 93.77%, respectively), demonstrating the best performance. DNN applications, on the other hand, yielded the highest sensitivity result, with 99.72%.

KEYWORDS: ARP spoofing, machine learning, deep neural networks, random forest, decision tree, penetration testing, cybersecurity
Science Code / Codes: 92403,92432 Page Number : 110

İÇİNDEKİLER

Sayfa

ÖZET ..... **i**
ABSTRACT ..... **ii**
İÇİNDEKİLER ..... **iii**
ŞEKİL LİSTESİ ..... **v**
TABLO LİSTESİ ..... **vii**
KISALTMA LİSTESİ ..... **viii**
ÖNSÖZ ..... **x**
1. GİRİŞ **1**
1.1 Sızma Testi Nedir? 1
1.2 Sızma Testinin Pazardaki Yeri 2
1.3 Adres Çözümleme Protokolü (ARP) ve Zafiyeti 3
1.4 Tezin Katkısı 4
2. SIZMA TESTLERİ VE ARP SAHTECİLİĞİ **6**
2.1 Sızma Testi Metodolojileri 6
2.1.1 Kara Kutu Testi 6
2.1.2 Beyaz Kutu Testi 7
2.1.3 Gri Kutu Testi 7
2.2 Sızma Testi Standartları 8
2.2.1 Açık Web Uygulama Güvenlik Projesi (Open Web Application Security Project-OWASP) 8
2.2.2 Açık Kaynak Güvenlik Test Metodolojisi Kılavuzu (Open Source Security Testing Methodology Manual - OSSTMM) 9
2.2.3 Sızma Testi Yürütme Standardı (The Penetration Testing Execution Standard - PTES) 9
2.2.4 Bilgi Sistemleri Güvenlik Değerlendirme Çerçevesi (Information Systems Security Assessment Framework- ISSAF) 11
2.2.5 Ağ Güvenliği Test Kılavuzu (Guideline on Network Security Testing) 11
2.3 Sızma Testi Süreci 11
2.3.1 Test Öncesi Etkileşim 11
2.3.2 Bilgi Toplama 12
2.3.3 Tehdit Modelleme 13
2.3.4 Zafiyet Analizi 13
2.3.5 İstismar 16
2.3.6 İstismar Sonrası 17
2.3.7 Raporlama 19
2.4 İstismar Saldırıları 20
2.4.1 SQL Enjeksiyonu (SQL Injection) 20
2.4.1.1 Çerezler Aracılığıyla Enjeksiyon 20
2.4.1.2 Sunucu Değişkenleri Üzerinden Enjeksiyon 20
2.4.1.3 İkinci Dereceden Enjeksiyon 21
2.4.1.4 SQL Zafiyet Uygulamaları 22
2.4.2 Ortadaki Adam Saldırısı (Man in the middle - MITM) 25
2.4.2.1 MITM Saldırısı Örneği 27
2.4.3 Hizmet Reddi Saldırısı (Denial of Service Attack - DoS) 30

2.4.3.1 DoS Saldırısı Örneği 31
2.4.4 Şifre Saldırısı (Password Attack) 32
2.4.4.1 Şifre Saldırısı Örneği 33
2.4.5 Siteler Arası Betik Çalıştırma Saldırısı (Cross-site scripting - XSS) 35
2.4.5.1 XSS Saldırı Örneği 36
2.5 ARP Sahteciliği ve İlgili Çalışmalar 37
3. MATERYAL VE YÖNTEM 50
3.1 Veri Seti ve Ön İşleme 50
3.1.1 Veri Seti 50
3.1.2 Öznitelik Seçimi 52
3.1.3 Veri Ön İşleme 52
3.2 Makine Öğrenmesi Modelleri 52
3.2.1 Derin Sinir Ağları (Deep Neural Networks - DNN) 53
3.2.1.1 Aktivasyon Fonksiyonları 54
3.2.1.2 Optimizasyon Algoritması 56
3.2.1.3 Kayıp Fonksiyonu 57
3.2.2 Karar Ağacı (Decision Tree) 57
3.2.3 Lojistik Regresyon (Logistic Regression) 58
3.2.4 Rastgele Orman (Random Forest) 59
3.2.5 K-En Yakın Komşu Algoritması (K-Nearest Neighbors - KNN) 59
3.3 Model Doğrulama Stratejisi 60
3.3.1 Çapraz Doğrulama (Cross-Validation) 60
3.3.2 Doğrulama Metrikleri 61
4. ARP SAHTECİLİĞİ TESPİT UYGULAMASI 65
4.1 Veri Ön İşleme 65
4.1.1 Alani ve ark. (ARP-PROBE) Çalışması Veri Bilgileri ile Karşılaştırma 71
4.2 Derin Sinir Ağı Uygulaması 73
4.2.1 Derin Sinir Ağı (DNN) Modeli 73
4.2.2 Çapraz Doğrulama (Cross-Validation) 76
4.3 Geleneksel Makine Öğrenmesi Yöntemlerinin Uygulanması 77
5. BULGULAR 80
5.1 Derin Sinir Ağları Yönteminin Deney Sonuçları 80
5.1.1 Deney 1 – 25 Epoch 80
5.1.2 Deney 2 – 50 Epoch 85
5.1.3 Deney 3 – 100 Epoch 88
5.1.4 Deney 4 – 200 Epoch 92
5.2 Geleneksel Makine Öğrenmesi Yöntemlerinin Deney Sonuçları 95
5.2.1 Karar Ağacı 95
5.2.2 Rastgele Orman 97
5.2.3 K-En Yakın Komşu 99
5.2.4 Lojistik Regresyon 101
6. TARTIŞMA VE SONUÇ 103
6.1 DNN Yönteminin Değerlendirilmesi 103
6.2 Geleneksel ML Yöntemlerinin Değerlendirilmesi 103
6.3 Sonuç 104
7. KAYNAKLAR 107
ÖZGEÇMİŞ 110

ŞEKİL LİSTESİ

Sayfa

Şekil 1.1: Siber güvenlik pazar hacmi 2028 tahmini (milyar dolar). 3
Şekil 2.1: Sızma testleri arasındaki farklar. 6
Şekil 2.2: Sızma testi süreç akış şeması. 10
Şekil 2.3: Şifre güncelleme sorgusu. 21
Şekil 2.4: Enjeksiyon sorgusu. 21
Şekil 2.5: Mutillidae test sayfası. 22
Şekil 2.6: Hata mesajı. 23
Şekil 2.7: Proxy yöntemi. 23
Şekil 2.8: Manipüle edilecek olan istek. 24
Şekil 2.9: Manipülasyon işlemi. 24
Şekil 2.10: Manipülasyon sonrası başarılı istek. 25
Şekil 2.11: Ettercap açılışı. 27
Şekil 2.12: Tarama başlatma. 28
Şekil 2.13: Ağ listesinden hedef cihazların seçimi. 28
Şekil 2.14: ARP sahteciliğinin başlatılması. 29
Şekil 2.15: ARP sahteciliği ile trafiğin dinlenmesi. 30
Şekil 2.16: DoS saldırısı sırasında hedef web sayfasının tepkisi. 32
Şekil 2.17: Şifre saldırısı istek detayı. 34
Şekil 2.18: Şifre saldırısı yanıt detayı. 34
Şekil 2.19: Doğru kullanıcı bilgileri ile giriş. 35
Şekil 2.20: Arama alanına JavaScript kodunun enjekte edilmesi. 36
Şekil 2.21: Başarılı XSS saldırısı çıktısı. 37
Şekil 2.22: ARP istek yanıt yapısı. 38
Şekil 2.23: ARP paketi. 38
Şekil 2.24: Örnek ARP isteği. 42
Şekil 2.25: Örnek ARP yanıtı. 42
Şekil 2.26: İstek öncesi ARP tablosu (192.168.0.1). 43
Şekil 2.27: Cevap sonrası ARP tablosu (192.168.0.1). 43
Şekil 2.28: İstek sonrası ARP tablosu (192.168.0.23). 43
Şekil 2.29: Arp sahteciliği saldırı senaryosu. 45
Şekil 2.30: İkinci ARP saldırı senaryosu. 46
Şekil 3.1: Saldırı tespit sistemi akış diyagramı. 53
Şekil 3.2: DNN Mimarisi. 53
Şekil 3.3: Relu aktivasyon fonksiyonu grafiksel gösterimi. 55
Şekil 3.4: Sigmoid aktivasyon fonksiyon grafiksel gösterimi. 56
Şekil 3.5: ARP sahteciliğine özel bir karar ağacı yapısı 58
Şekil 3.6: Lojistik regresyon grafiksek gösterim. 58
Şekil 3.7: KNN sınıflandırma örneği. 60
Şekil 3.8: Mükemmel sınıflandırılmış ROC ve AUC grafiği. 64
Şekil 4.1: DNN model mimarisi. 74
Şekil 5.1: Ölçüm metrikleri grafiği – Deney 1. 81
Şekil 5.2: Model doğruluk ve kayıp grafiği – Deney 1. 81
Şekil 5.3: Karışıklık matrisi – Deney 1. 82
Şekil 5.4: ROC eğrisi grafiği – Deney 1. 83
Şekil 5.5: Ölçüm metrikleri grafiği – Deney 2. 85
Şekil 5.6: Model doğruluk ve kayıp grafiği – Deney 2. 86

Şekil 5.7: Karışıklık matrisi – Deney 2. 87
Şekil 5.8: ROC eğrisi grafiği – Deney 2. 87
Şekil 5.9: Ölçüm metrikleri grafiği – Deney 3. 89
Şekil 5.10: Model doğruluk ve kayıp grafiği – Deney 3. 89
Şekil 5.11: Karışıklık matrisi – Deney 3. 90
Şekil 5.12: ROC eğrisi grafiği – Deney 3. 91
Şekil 5.13: Ölçüm metrikleri grafiği – Deney 4. 92
Şekil 5.14: Model doğruluk ve kayıp grafiği – Deney 4. 93
Şekil 5.15: Karışıklık matrisi – Deney 4. 93
Şekil 5.16: ROC eğrisi grafiği – Deney 4. 94
Şekil 5.17: Ölçüm metrikleri grafiği – Karar Ağacı. 95
Şekil 5.18: ROC eğrisi grafiği – Karar Ağacı. 96
Şekil 5.19: Karışıklık matrisi – Karar Ağacı. 96
Şekil 5.20: Ölçüm metrikleri grafiği – Rastgele Orman. 97
Şekil 5.21: ROC eğrisi grafiği – Rastgele Orman. 98
Şekil 5.22: Karışıklık matrisi – Rastgele Orman. 98
Şekil 5.23: Ölçüm metrikleri grafiği – KNN. 99
Şekil 5.24: ROC eğrisi grafiği – KNN. 100
Şekil 5.25: Karışıklık matrisi – KNN. 100
Şekil 5.26: Ölçüm metrikleri grafiği – Lojistik Regresyon. 101
Şekil 5.27: ROC eğrisi grafiği – Lojistik Regresyon. 102
Şekil 5.28: Karışıklık matrisi – Lojistik Regresyon. 102

TABLO LİSTESİ

Sayfa

Tablo 2.1: İlk on donanım türleri. 40
Tablo 2.2: İlk on protokol türü. 41
Tablo 3.1: Veri seti paket bilgileri. 51
Tablo 3.2: Veri setinde kullanılan dosya bilgileri. 51
Tablo 3.3: Karışıklık matrisi. 63
Tablo 4.1: Uygulama ortamı. 65
Tablo 4.2: Seçilen özellikler. 66
Tablo 4.3: Alani ve ark. makalesi ve çalışmamız arasındaki veri seti karşılaştırması. 72
Tablo 5.1: Çapraz doğrulama sonuçları – Deney 1. 84
Tablo 5.2: Çapraz doğrulama sonuçları – Deney 2. 88
Tablo 5.3: Çapraz doğrulama sonuçları – Deney 3. 91
Tablo 5.4: Çapraz doğrulama sonuçları – Deney 4. 94
Tablo 6.1: DNN uygulama sonuçları 103
Tablo 6.2: Geleneksel makine öğrenmesi yöntemlerinin sonuçları 104

KISALTMA LİSTESİ

ARCNET: Attached Resource Computer Network
ARP: Address Resolution Protocol
AUC: Area Under the ROC Curve
CNN: Convolutional Neural Networks
CPU: Central Processing Unit
CSV: Comma-Separated Values
CVE: Common Vulnerabilities and Exposures
DAI: Dynamic ARP Inspection
DDoS: Distributed Denial of Service
DNN: Deep Neural Networks
DNS: Domain Name System
DOM: Document Object Model
FPR: False Positive Rate
FQDN: Fully Qualified Domain Name
GAN: Generative Adversarial Networks
HTTP: Hypertext Transfer Protocol
ICMP: Internet Control Message Protocol
IDS: Intrusion Detection Systems
IoT-ID: Internet of Things Network Intrution Dataset
IP: Internet Protocol
IPS: Intrusion Prevention System
IPX: Internetwork Packet Exchange
ISECOM: The Institute for Security and Open Methodologies
ISSAF: Information Systems Security Assessment Framework
KNN: K-Nearest Neighbors
LAN: Local Area Networks
MAC: Media Access Control
MITM: Man-in-the-Middle Attack
ML: Machine Learning
Nadam: Nesterov-accelerated Adaptive Moment Estimation
NIST: National Institute of Standards and Technology
OSSTMM: Open Source Security Testing Methodology Manual
OSVDB: Open Sourced Vulnerability Database
OWASP: Open Web Application Security Project
PCAP: Packet Capture
PIN: Personal Identification Numbers
PPPoE: Point-to-Point Protocol over Ethernet
PTES: The Penetration Testing Execution Standard
ReLU: Rectified Linear Unit
ROC: Receiver Operating Characteristic Curve
S-ARP: Secure Address Resolution Protocol
SATAN: Security Administrator Tool for Analyzing Networks
SQL: Structured Query Language
SSL: Secure Sockets Layer
TCP: Transmission Control Protocol
TPR: True Positive Rate
TLS: Transport Layer Security
UDP: User Datagram Protocol

VM: Virtual Machine
WAF: Web Application Firewall
XSS: Cross-site scripting

ÖNSÖZ

Bu tez çalışmam boyunca bana her aşamada destek olan, bilgi ve tecrübelerini esirgemeyen değerli danışmanım Prof. Dr. Selçuk Kavut'a en içten teşekkürlerimi sunarım. Aynı zamanda, Balıkesir Üniversitesi Bilgisayar Mühendisliği Bölümü'ndeki tüm değerli öğretim üyelerine, bilgi birikimlerini bizlere aktardıkları ve her daim yol gösterici oldukları için teşekkür ederim. Bu süreçte yanımda olan, manevi desteklerini esirgemeyen değerli aileme ve sevdiklerime de sonsuz teşekkürlerimi sunarım. Tezimin bu noktaya gelmesinde katkıda bulunan herkese minnettarım.

Balıkesir, 2024 **Mustafa Furkan CEYLAN**

1. GİRİŞ

1.1 Sızma Testi Nedir?

Günümüzde teknolojinin gelişmesiyle, web ve mobil uygulamalarının insan hayatının odağına yerleşmesi, internetin nesnelere entegre edilmesi ve bilginin hızla yayılması beraberinde güvenlik risklerini de getirmiştir. Geliştirilen ilgili ürünlerin özenli şekilde standartlara uygun olarak üretilmemesinden dolayı güvenlik zafiyetleri oluşmaktadır. Bu zafiyetlerden faydalanmak isteyen saldırganlar, amaçlarına hizmet eden çeşitli araçlar kullanarak elde ettikleri bilgiyi çıkarları doğrultusunda kullanmaktadırlar. Zafiyetlerin önceden tespit edilmesi ve olası saldırılardan önce gerekli tedbirlerin alınması, karşılaşılabilecek maddi ve manevi zararların büyük çoğunluğunun önlenmesinde etkin rol oynamaktadır. İşte bu işlemlerden biri olan sızma testi, siber güvenlik pazarında önemli bir paya sahiptir.

Sızma testi, sistemin güvenliğini değerlendirmek için mevcut güvenlik açıklarını sürekli olarak tespit etmeye çalışır. Bu yaklaşım, saldırganların yöntemlerini simüle ederken saldırganlar gibi zafiyetten yararlanıp bilgileri yok etmeyi amaçlamaz, aksine mevcut sistemin güvenliğini arttırmayı hedeflemektedir. Ürün yöneticilerine sistem hakkında zafiyetlerin önem dereceleri ve yol açabilecekleri risk durumlarını ayrıntılı olarak ele alan yönetici raporu hazırlanarak sunulur. Ürün geliştiricilerine ise, sistem zafiyetlerinin teknik detayları ve bu zafiyetlerin ortadan kaldırılması için alınması gereken tedbir tavsiyelerinden oluşan bir rapor sunulur. Böylece saldırganlardan önce zafiyetler kapatılarak finansal ve kurumsal zararlardan korunmuş olur.

Sızma testi bazı araçlar kullanılarak manuel veya farklı seviyelerde otomatize edilmiş olarak uygulanabilir. Otonom test sistemleri ile karşılaştırıldığında, manuel sızma testi için yetkin bir takıma ve daha fazla zamana ihtiyaç olduğundan dolayı maliyeti daha fazladır. Manuel olarak yapılan zafiyet taramalarında atlama ya da hata yapma olasılığı yüksektir. Yarı otonom sızma testi zafiyet tarama araçları ile otomatik taramaların yapılabildiği ancak sızma testi uzmanı tarafından saldırıların yönlendirildiği ve bazı taramaların testten çıkarılabildiği bir yöntemdir. Otonom sızma testi ise, tamamıyla sızma testini ve tespit edilen zafiyetlerin raporlanma işlemini baştan sona otomatik olarak gerçekleştiren bir yöntemdir. İnsan hatasını, analiz ve bilgi toplama süreçlerinde işlem zamanını azalttığından dolayı maliyeti düşürmektedir.

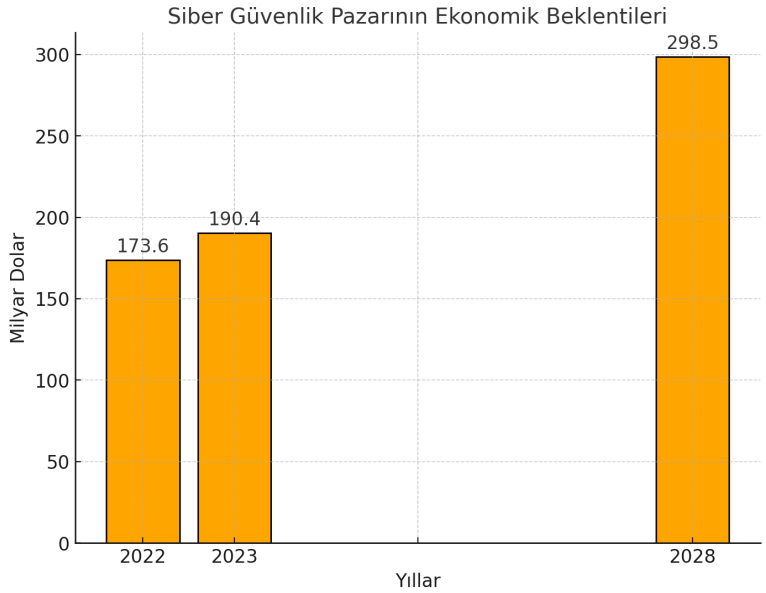
Sızma testi bilgisayar sistemlerinin güvenlik seviyelerini tespit eden en eski tekniklerden birisidir. 1970 yılının başlarında Savunma Bakanlığı tarafından bu teknik kullanılarak bilgisayar sistemlerindeki güvenlik açıklarını ortaya çıkarmak ve daha güvenli sistemlerin geliştirilmesine katkıda bulunmak istenilmiştir [1].

İlk ağ tarayıcısı olarak gösterilen SATAN (Security Administrator Tool for Analyzing Networks), Amerika Adalet Bakanlığı tarafından ulusal güvenliğin tehdit edildiği gerekçesiyle tasarımcılarına yaptığı baskılardan sonra 1995 yazında popüler oldu. SATAN 100'den fazla bilinen zafiyetleri ağda tarayarak sistemin güvenlik seviyesini belirleyebilen bir program olarak tasarlanmıştır [2]. Daha sonrasında bu programın yerini Nmap ve Nessus gibi araçlar aldı.

1.2 Sızma Testinin Pazardaki Yeri

Giderek yaygınlaşan internet tabanlı uygulamalar, web ve mobil uygulama kullanımının ve bu uygulamaların kritiklik düzeyinin artması doğrultusunda siber saldırı tehditlerini de artırmaktadır. Özellikle Covid-19 pandemisiyle şirketlerin ofislerini kapatıp tamamen evden çalışma modeline geçmesi ve tüm hizmetlerde ciddi anlamda dijitalleşmeye gidilmesi, siber saldırı tehditlerine farklı fırsatlar tanımıştır. Buna karşı önlem almak isteyen bazı ülkelerin zorunlu sızma testi uygulamaları, şirketlerin ürün güveni ve kalitesinin artırılmasını istemesi nedenleriyle, sızma testi pazarının gün geçtikçe büyüdüğü görülmektedir.

Şekil 1.1 de görüldüğü üzere sızma testi pazar hacmi, 2022'de 1.4 milyar dolar iken 2027'de 2.7 milyar dolara yükseleceği öngörülmektedir. Coğrafik olarak bakıldığında, dünyada pazarın büyük bir bölümünü Kuzey Amerika oluşturmaktadır [3]. Sızma testinin de içinde bulunduğu siber güvenlik pazar hacmi 2022 ve 2023 yıllarında sırasıyla 173.6 ve 190.4 milyar dolarken, 2028'de %9.4 lük büyümeyle 298.5 milyar dolara çıkması beklenmektedir [4].



Şekil 1.1: Siber güvenlik pazar hacmi 2028 tahmini (milyar dolar).

1.3 Adres Çözümleme Protokolü (ARP) ve Zafiyeti

Adres Çözümleme Protokolü (Address Resolution Protocol- ARP), bir ağ üzerinde bulunan cihazın İnternet Protokolü (Internet Protocol - IP) adresini Medya Erişim Kontrolü (Media Access Control - MAC) adresine dönüştürmeyi sağlayan temel bir ağ protokolüdür [5]. IPv4 tabanlı ağlarda cihazların birbirleriyle doğrudan iletişim kurabilmesi için IP adreslerinin MAC adresleri ile eşleştirilmesi gerekir. Bu eşleştirme, cihazlar arasında sorunsuz bir ağ iletişimi sağlar. Ancak ARP, bu işlevi yerine getirirken güvenlik açısından ciddi zafiyetler barındırmaktadır.

ARP zafiyetleri, bu protokolün çalışma prensibinden kaynaklanır. ARP, ağda bir cihaza yönelik ARP isteği gönderildiğinde, herhangi bir doğrulama yapmadan gelen ARP yanıtını kabul eder. Bu durum, saldırganın sahte ARP yanıtları göndererek cihazların ARP tablolarını manipüle etmesine olanak tanır. ARP sahteciliği (spoofing) olarak bilinen bu saldırıda, saldırgan, ağdaki cihazların ARP tablolarına yanlış MAC adreslerini ekler ve böylece iki cihaz arasındaki veri trafiğini ele geçirerek sızma saldırılarından biri olan Ortadaki Adam (Man-in-the-Middle - MITM) saldırısını gerçekleştirir. Bu güvenlik zafiyeti, ağ üzerinde hassas bilgilerin çalınmasına veya veri akışının bozulmasına neden olabilir [6].

ARP sahteciliği saldırıları, günümüz ağ güvenliği açısından ciddi tehditler oluşturmaktadır. Saldırganlar, ağdaki veri akışını manipüle ederek şifreleri, oturum bilgilerini ve kişisel verileri ele geçirebilir. Bu nedenlerle, ARP sahteciliği saldırılarının tespiti ve bu tür

saldırılara karşı önlemlerin alınması kritik öneme sahiptir. Bu tezde, aşağıda sunulduğu gibi ARP sahteciliğinin tespit edilmesinde makine öğrenmesi ve derin öğrenme yöntemi çalışılmıştır.

1.4 Tezin Katkısı

A. **Tüm Veri Setinin (IoT-ID) Kullanımı**

Çalışmamızda, yakın zamanda Alani ve ark. [7] tarafından önerilen ARP sondası (APR Probe) sistemi gerçekleştirilmiş ve bunun bir sonucu olarak, bahsedilen sistemde kullanılan DNN modelinin tasarımında Nesnelerin İnterneti Ağ Saldırısı Veri Setinin (Internet of Things Network Intrusion Dataset – IoT-ID) [8] kısmi olarak ele alındığı ve yalnızca belirli veri segmentleri üzerinde işlem yapıldığı gözlenmiştir. Diğer bir ifadeyle, [7] makalesinde kullanılan IoT-ID, belirli kısıtlamalar çerçevesinde işlenmiş ve veri setindeki örneklerin tamamı model eğitimine dahil edilmemiştir. Bu çalışmada ise, tüm veri seti kullanılarak model eğitimi gerçekleştirilmiş ve sonuçların daha geniş bir veri tabanı üzerinden değerlendirilmesi sağlanmıştır. Bu sayede, verideki tüm varyasyonlar ve olası senaryolar dikkate alınarak daha gerçekçi ve kapsamlı bir sonuç elde edilmiştir. Tüm veri setinin kullanılması, modelin genelleme yeteneğini artırmış ve özellikle pozitif ve negatif sınıfların daha dengeli bir şekilde temsil edilmesi sağlanmıştır.

B. **Farklı Modellerin Uygulanması**

Çalışmamızın diğer önemli bir katkısı, ARP sahteciliği tespiti için kullandığımız modellerin çeşitliliği ve bu modellerin başarımlarının karşılaştırılmasıdır. Bu tezde ele alınan çalışmada [7] tek bir model olan DNN kullanılmıştır ve literatürde bulunan birçok çalışmada [9],[10], [11] ARP sahteciliği tespiti için sınırlı sayıda yöntem kullanılması ile elde edilen sonuçlar paylaşılmaktadır. Ancak, bu tezde tek bir yönteme bağlı kalınmadan çeşitli geleneksel makine öğrenmesi yöntemleri (Karar Ağacı, Rastgele Orman, K-En Yakın Komşu (KNN), Lojistik Regresyon) ve özel olarak [7]çalışmasında kullanılan DNN modeli uygulanarak, bu yöntemlerin ARP sahteciliği tespiti üzerindeki etkileri karşılaştırmalı olarak incelenmiştir. Bu farklı yöntemlerin uygulanması, her birinin güçlü ve zayıf yönlerini görme olanağını sağlamıştır. Özellikle Rastgele Orman ve Karar Ağacı gibi geleneksel yöntemlerin, ARP sahteciliği tespitinde yüksek doğruluk ve keskinlik sunduğu görülmüştür. Ayrıca, DNN modelinin uygulanması sonucunda, bu modelin ele aldığımız makine öğrenmesi yöntemleri ile karşılaştırıldığında pozitif sınıfları tespit etmede daha başarılı olduğu gösterilmiştir.

C. **Sonuçların Geliştirilmesi**

Bu tezde, ARP sonda sisteminde [8] uyguladığımız geleneksel makine öğrenmesi yöntemleri ile ARP sahteciliğinin tespitinde doğruluk, keskinlik ve F1 skoru metrikleri DNN modeline göre iyileştirilmiştir. Bu yöntemlerin veri kümesi üzerinde çapraz doğrulama ile test edilerek, sonuçların daha güvenilir olmasını sağlanmıştır. Ayrıca, makine öğrenmesi yöntemleri kullanılarak elde edilen sonuçların detaylı bir şekilde çalışılması ve bu sonuçların DNN modeli ile karşılaştırılması, ARP sahteciliği tespitinde hangi yöntemlerin daha etkili olduğunu anlamamıza yardımcı olmuştur. Karar Ağacı, Rastgele Orman gibi geleneksel yöntemler ile DNN modeli karşılaştırıldığında, geleneksel yöntemlerin daha düşük hesaplama maliyetine rağmen yüksek performans sunduğu gözlenmiştir. Bununla birlikte, DNN modelinin saldırıları doğru bir şekilde sınıflandırmada etkili bir performans sergilediği gösterilmiştir.

D. **Performans Değerlendirmeleri**

Bu tez kapsamında, kullanılan tüm yöntemlerin başarım değerlendirmeleri, doğruluk, keskinlik, duyarlılık ve F1 skoru gibi başarım metriklerinin yanı sıra, Alıcı İşletim Karakteristiği Eğrisi (Receiver Operating Characteristic Curve - ROC) ve ROC Eğrisi Altındaki Alan (Area Under the ROC Curve - AUC) değerleri kullanılarak geniş bir perspektifte yapılmıştır. Böylelikle, modelin genelleme yeteneği konusunda kapsamlı bir anlayış sunulmaktadır.

Sonuç olarak, bu çalışmada kullanılan hem derin sinir ağları hem de geleneksel makine öğrenmesi yöntemleri farklı metriklerle test edilmiş ve en uygun modeller ortaya konmuştur. Ayrıca, tüm veri setinin kullanıldığı farklı makine öğrenmesi yöntemlerinin uygulanması sonucu elde ettiğimiz metrik değerleri, başarım artışı sağlayarak nispeten yeni bir bakış açısı sağlamıştır. Bu kapsamda, çalışmamız ARP sahteciliği tespitinde önemli katkılar sağlamıştır.

2. SIZMA TESTLERİ VE ARP SAHTECİLİĞİ

2.1 Sızma Testi Metodolojileri

Sızma testleri, yazılım test yöntemleri arasında önemli bir yer işgal eder ve sistematik test süreçlerinin kritik bir parçasıdır. Bu testler, çeşitli saldırı senaryoları tasarlanarak gerçekleştirilir ve yazılım veya sistemdeki güvenlik zafiyetlerini belirlemeyi amaçlar. Şekil 2.1'de görüldüğü üzere üç farklı türü bulunur: kara kutu, beyaz kutu ve gri kutu testleri. Kara kutu testi saldırganın hiçbir bilgiye sahip olmadığı bir senaryoyu varsayarken, beyaz kutu testi sistemin iç işleyişinin tam olarak bilindiği bir senaryoyu varsayarak gerçekleştirilir. Gri kutu testi ise sınırlı bilgiye sahip bir saldırganın bakış açısıyla gerçekleştirilir. Bu farklı yaklaşımlar, farklı türde zayıf noktaların tespit edilmesine ve güvenlik açıklarının giderilmesine yardımcı olur [12], [13]. Bu sayede yazılımlar daha güvenli hale getirilir ve potansiyel saldırılara karşı daha dirençli hale gelir.

Şekil 2.1: Sızma testleri arasındaki farklar.

2.1.1 Kara Kutu Testi

Bu test türü, bilgi gereksiniminin düşük olduğu bir yaklaşımla icra edilir ve genellikle son kullanıcı bakış açısıyla değerlendirilir. Ürün ya da şirket ortamının dışından yapılan bir test yöntemidir. Yazılım geliştirme döngüsünün erken aşamalarından itibaren uygulanabilir olmaları test yapanların ürün hakkında belirli bir bilgi birikimi gerektirmemesi açısından avantaj sağlar, ancak zaman bakımından maliyeti fazladır. Test yapanlar ile yazılım

geliştiricileri genellikle bağımsız çalıştıklarından, bu testler geliştiricilerin gözünden kaçan hataların tespit edilmesine yardımcı olur. Aynı zamanda alınan güvenlik tedbirlerinin ne kadar etkili olduğunu da ölçer. Dolayısıyla, kara kutu yöntemi ile gerçekleştirilen testler, yazılımın güvenilirliğini artırmak için önemli bir araç olarak kabul edilir.

2.1.2 Beyaz Kutu Testi

Beyaz kutu testi, kara kutu testiyle birlikte yazılım test süreçlerinde önemli bir rol oynar ve birbirlerini tamamlayıcı olarak kullanırlar. Beyaz kutu testinde saldırgan, ürünün ağ topolojisi, portları, sistem bilgileri, servisler dahil olmak üzere hedefler hakkında detaylı bilgiye hatta kaynak koda sahiptir. Dolayısıyla, test edilmek istenen unsurlar daha özel olarak belirlenebilir ve testler daha derinlemesine tasarlanabilir. Beyaz kutu testinde, tasarım ve planlamanın detaylarına hâkim olunduğu için, test senaryoları ve stratejileri daha doğrudan ve özelleştirilebilir bir şekilde oluşturulabilmektedir. Kara kutu testinde gereken bilgi toplama adımına ihtiyaç duyulmadığından dolayı bu test yöntemi zaman açısından daha az maliyetlidir. Bu test türü genellikle şirket içinde gerçekleştirilir ve kara kutu testi ile kullanıldığında, yazılımın farklı yönlerini kapsamlı bir şekilde test etmeye yardımcı olur. Bu nedenle, beyaz kutu testi, yazılımın güvenilirliğini ve kalitesini artırmak için önemli bir araç olarak kabul edilir.

2.1.3 Gri Kutu Testi

Gri kutu testi, beyaz kutu ve kara kutu testlerinin birleşiminden oluşan bir test tekniğidir. Bu test tekniği, test uzmanının sistem hakkında sınırlı bir bilgiye sahip olduğu ve dolayısıyla hem beyaz kutu testi gibi tüm kaynak koda hâkim olmadığı, hem de kara kutu testi gibi tamamen dışarıdan olmadığı bir denge sağlar. Gri kutu testinde, test senaryolarının tasarlanmasında beyaz kutu testi yöntemleri, bu senaryoların uygulanmasında ise kara kutu testi yöntemleri uygulanır. Diğer bir ifadeyle, yazılımın iç yapısını kısmen anlamak ve kodu incelemek için beyaz kutu testi yaklaşımını kullanırken, aynı zamanda sistemin davranışlarını ve kullanıcı deneyimini gözlemlemek için kara kutu testi yaklaşımını da benimser. Bu sayede, test uzmanları hem yazılımın iç işleyişine dair bilgi sahibi olabilirler, hem de saldırgan perspektifinden sistemi değerlendirerek saldırılarda bulunur. Gri kutu testi hem yazılımın iç zayıflıklarını hem de dışarıdan bir saldırgan gibi testler yaparak dış zayıflıklarını tespit etmede etkili bir yöntemdir. Bu nedenle, yazılımın güvenilirliğini artırmak ve hataları daha etkili bir şekilde tespit etmek için önemli bir araç olarak kabul edilir.

2.2 Sızma Testi Standartları

Sızma testi, genellikle uzman ekipler tarafından yürütülen ve yüksek düzeyde uzmanlık gerektiren bir uygulamadır. Günden güne yeni güvenlik açıklarının keşfedilmesi ve saldırıların artması, bu alanda ortak bir anlayışın ve standartların oluşturulmasını zorunlu kılmaktadır. Bireylerin kendi keşfettikleri güvenlik açıklarını paylaşması ve açık kaynaklı platformların oluşturulması, güvenlik sektörü için en uygun yöntemlerden biridir. Bu nedenle, sızma testinin işleyişi, metodolojisi ve adımları için uygulanabilir açık kaynaklı bazı standartlar mevcuttur.

2.2.1 Açık Web Uygulama Güvenlik Projesi (Open Web Application Security Project-OWASP)

Açık Web Uygulama Güvenlik Projesi, kâr amacı gütmeyen ve yazılım güvenliğinin artması için çalışan bir platformdur. Bu platform sektördeki çoklu fikirleri birleştirmek, ortak fayda için çokça karşılaşılan güvenlik hatalarını ve buna karşı standartlarını paylaşmak için 2001'de kurulmuştur. Kurulduğu günden beri açık kaynaklı birçok proje gerçekleştirmiştir ve bu projelerden birisi de OWAPS Top Ten'dir. OWAPS, bazı yıllarda güvenlik sektörü için zafiyetin kritiklik ve kullanılabilirlik parametrelerine göre bir ilk 10 zafiyet listesi yapar. Geliştiriciler ve test uzmanları son çıkan zafiyetleri OWAPS sayesinde takip ederek, bu zafiyetleri ve alınması gereken tedbirleri kendi projelerine uyarlamaktadırlar. Bu tez yazılırken en son çıkan ilk 10 zafiyet listesi aşağıdaki gibidir [14]:

OWAPS İlk On Zafiyet - 2021

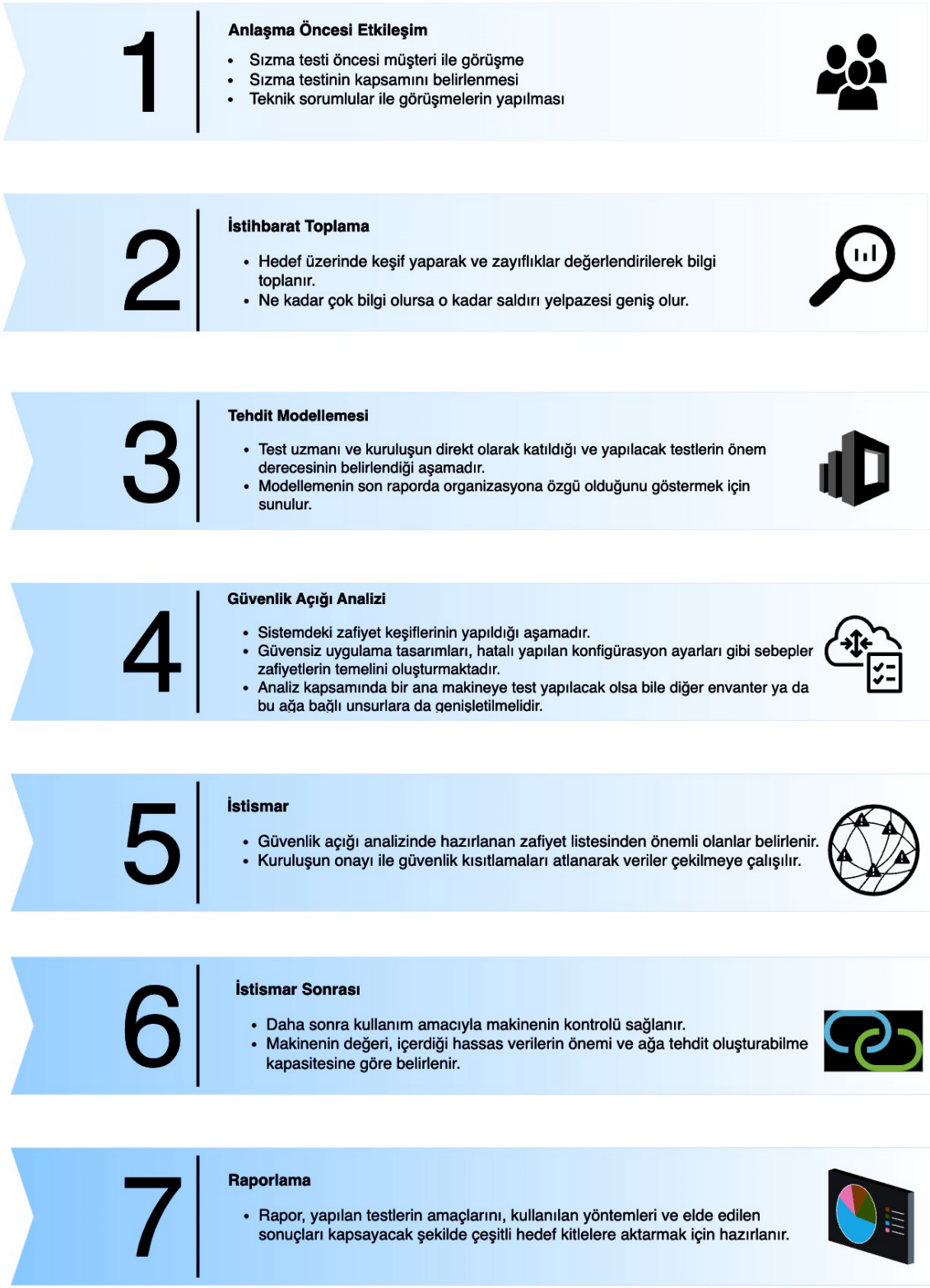
I. Bozuk Erişim Kontrolü (Broken Access Control)
II. Şifreleme Hataları (Cryptographic Failures)
III. Enjeksiyon (Injection)
IV. Güvensiz Tasarım (Insecure Design)
V. Güvenlik Yanlış Yapılandırma (Security Misconfiguration)
VI. Savunmasız ve Eski Bileşenler (Vulnerable and Outdated Components)
VII. Tanımlama ve Kimlik Doğrulama (Hataları Identification and Authentication Failures)
VIII. Yazılım ve Veri Bütünlüğü Hataları (Software and Data Integrity Failures)
IX. Güvenlik Loglama ve İzleme (Security Logging and Monitoring)
X. Sunucu Taraflı İstek Sahteciliği (Server Side Request Forgery)

2.2.2 Açık Kaynak Güvenlik Test Metodolojisi Kılavuzu (Open Source Security Testing Methodology Manual - OSSTMM)

ISECOM (The Institute for Security and Open Methodologies) tarafından ilk olarak 2000 yılında ortaya çıkarılan bu proje, çeşitli disiplinlerden ve dış meslek gruplarından uzmanların oluşturduğu açık bir topluluk tarafından sürdürülmektedir. Ticari ve siyasi etkilerden bağımsız olarak finanse edilen ISECOM projeleri, ortaklıklar, abonelikler, sertifikalar ve lisanslara dayalı araştırmalarla desteklenmektedir [15]. Teknolojik ortamların giderek karmaşıklaşması, uzaktan operasyonlar, sanallaştırma ve bulut bilişim gibi gelişmelere paralel olarak, OSSTMM buna uyum sağlamıştır. Metodolojinin günümüzdeki üçüncü versiyonu, insan, fiziksel, kablosuz, telekomünikasyon ve veri ağları da dahil olmak üzere geniş bir kanal yelpazesini kapsamaktadır. Bu esneklik, OSSTMM'nin bulut bilişiminden mobil iletişim ağlarına kadar çeşitli altyapıları değerlendirmek için ideal olmasını sağlamakta ve birden fazla kanalda karşılaşılan karmaşık güvenlik sorunlarını ele alınmasında yardımcı olmaktadır. İnsan faktörünün sistemde ciddi bir güvenlik zafiyeti yaratabileceğine değinen ve sosyal mühendisliğe karşı yöntemler öneren ilk standart olarak bilinmektedir [16].

2.2.3 Sızma Testi Yürütme Standardı (The Penetration Testing Execution Standard - PTES)

PTES, bilgi teknolojileri uzmanları tarafından geliştirilen kapsamlı bir sızma testi metodolojisidir. Şekil 2.2'den görüleceği üzere sızma testini baştan sona 7 ana başlık altında detaylı bir rehber olarak sunar ve ayrıca uzmanların deneyimlerini de içerir. PTES, bilgi teknolojileri sektöründe en yaygın ve tanınmış standartlardan biridir [17].



Şekil 2.2: Sızma testi süreç akış şeması.

2.2.4 Bilgi Sistemleri Güvenlik Değerlendirme Çerçevesi (Information Systems Security Assessment Framework- ISSAF)

ISSAF, Açık Sistem Bilgi Güvenliği Grubu (Open Information Systems Security Group-OISSG) tarafından desteklenen ancak günümüzde aktif olarak geliştirilmeyen bir yönergeler kılavuzudur. Temel amacı, ürün geliştirme süreçlerinde ve sızma testi faaliyetlerinde kullanılabilecek bir metodoloji ve rehber oluşturmaktır. Hali hazırda olan sızma testi standartlarının, güvenlik metotlarının nasıl ve neden uygulanması gerektiğine değinmedikleri öne sürülerek böyle bir standart ortaya çıkartılmıştır. Diğer standartlara göre daha kapsamlı olduğu ve tüm sorulara cevap verildiği savunulmaktadır [18].

2.2.5 Ağ Güvenliği Test Kılavuzu (Guideline on Network Security Testing)

Bu belge, ABD'nin Ulusal Standartlar ve Teknoloji Enstitüsü (National Institute of Standards and Technology - NIST) tarafından ağ güvenliği zafiyetleri ve politika uygulamaları konusunda rehberlik sağlaması amacıyla NIST 800-42 özel yayınını oluşturmuştur. İnternetin evrimi ile ürün geliştirme süreçleri hız kazanmış, ancak sistem yönetimi giderek zorlaşmıştır. Ayrıca, saldırı çeşitliliği ve etkinliği artmıştır. Saldırganlar, saldırıları manuel veya otomatik olarak başlattıklarında, bütün sistemin güvenliğini aynı anda sağlamak zorlaşmaktadır. Bu sebeplerden dolayı bahsedilen test kılavuzu, internete bağlı sistemler ve ağlarla ilgili güvenlik sorunlarını ele almak ve çözüm üretmek üzere hazırlanmıştır. Kılavuzda önerilen test teknikleri, standart sistem ve ağ yönetimi uygulamalarına entegre edildiğinde, olayların önlenmesi ve bilinmeyen zafiyetlerin keşfi açısından maliyet ve etki bakımından faydalı olacaktır [19].

2.3 Sızma Testi Süreci

Bu bölümde, sızma testlerinde genellikle kullanılan ve güvenlik endüstrisinde kabul gören Şekil 2.2'te gösterildiği gibi test süreci ve alt başlıkları incelenmiştir [18], [20], [21], [22]

2.3.1 Test Öncesi Etkileşim

Sızma testlerinin icrası öncesinde, müşteri ile yapılan toplantılar esas alınarak testin amaçları, kapsamı, süresi, metodolojisi, zaman planlaması, maliyeti ve ücretlendirme gibi hususlar ele alınır. Bu aşamada, test sürecinde karşılıklı iletişim ve koordinasyon gerektirecek kişilerin kimlik bilgilerini içeren bir liste oluşturulur. Testin yasal çerçevesi ve uygunluğu için gerekli olan tüm belgeler ve sözleşmeler hazırlanır ve imzalanır. Tüm bu hazırlıklar tamamlandıktan sonra, test uzmanları, belirtilen yetkilendirme ve izinler doğrultusunda sızma testlerine başlamak üzere harekete geçerler. Bu şekilde, test süreci

başarılı bir şekilde yürütülmüş ve aynı zamanda taraflar arasındaki güven ve iş birliği pekiştirilmiş olur.

2.3.2 Bilgi Toplama

Bilgi toplama süreci, sızma testinin temel bir unsurudur ve testin etkinliğini doğrudan etkiler. Bilginin çeşitliliği ve niteliği, yapılan testlerde keşfedilebilecek zafiyetlerin kapsamını belirleyen kritik faktörlerdir. Bu bilgiler arasında fiziksel bilgiler (sunucu odasının yeri, geçiş kontrol sistemleri, donanım özellikleri vb.), bireysel bilgiler (çalışan ismi, ünvanı, mail adresi ve sosyal mühendislikte kullanılmak üzere potansiyel hassas kişisel bilgiler vb.), mantıksal ilişkiler (ağ topolojisi, sunucu – istemci ilişkisi ve veri akış diyagramı vb.), organizasyon yapısı (organizasyonun yönetim yapısı, departmanlar ve çalışanları, anahtar kişiler vb.), kullanılan cihazlar (bilgisayar, mobil telefonlar, ağ ekipmanları ve diğer donanımlar vb.), gibi çeşitli türleri bulunmaktadır. Test uzmanı belirlenen hedefe üç farklı perspektiften yaklaşır.

I. Pasif Keşif: Hedef tarafından asla tespit edilmemek gibi bir amaçla yapılan testlerde pasif keşif kullanılır. Pasif keşif esnasında, hedefe internet trafiği üzerinden ne kendi cihazımızdan ne de anonim bir sunucudan herhangi bir istek gönderilmemektedir. Elde edilen bilgiler, doğal olarak arşivlenmiş veya önceden kaydedilmiş kaynaklardan olacaktır. Ancak, bu bilgiler güncel olmayabilir ve yanıltıcı olabilir.

II. Yarı Pasif Keşif: Hedefin dikkatini üzerimize çekmeden, çeşitli kaynaklardan taramalar gerçekleştirilir. Bu kaynaklar, önceden yayınlanmış web sayfaları ve bunların barındığı sunucular, haberler, blog yazıları gibi çeşitli platformlar olabilir. Ağ düzeyinde aktif port taramaları ya da gizli içerik aramaları gibi müdahaleler yapılmaz. Bunun yerine, normal internet trafiği gibi görünen hareketlerle hedefin profil bilgileri toplanmaya çalışılır. Bu yaklaşım, saldırı sonrasında hedefin geriye dönüp trafik akışına baktığında kesin bir yargıya varamamasını sağlamayı amaçlar.

III. Aktif Keşif: Bu aşama, tespit edilme riski yüksek olmasına rağmen, ağ haritalaması ve güvenlik taraması gibi süreçlerle daha derinlemesine bilgi edinme amacıyla gerçekleştirilir. Ağ haritalaması sürecinde, tam port taramasıyla ağda hangi portların açık olduğu belirlenir ve bu sayede potansiyel güvenlik açıkları tespit edilir. Güvenlik taraması ise ağdaki sistemlerin ve yazılımların güvenlik zafiyetlerini

değerlendirmek için yapılan bir adımdır. Ayrıca, yayınlanmamış dizinler ve dosyaların taranmasıyla, hedef ağdaki gizli veya korunan bilgilere erişim sağlanmaya çalışılır. Bununla birlikte, sunucuların aktif olarak taranmasıyla, ağdaki hedef sistemlerin ve sunucuların güvenlik düzeyi hakkında ayrıntılı bilgi elde edilir. Bu adımlar, test uzmanlarına, hedef ağdaki güvenlik açıklarını ve potansiyel riskleri belirleme konusunda önemli bir perspektif sunar.

Elde edilen bilgiler ileride açık bir arka kapı bırakmak için ya da farklı test yaklaşımlarında kullanmak amacıyla önemlidir.

2.3.3 Tehdit Modelleme

Önceden toplanan bilgilerin ışığında, en uygun ve etkili test tekniklerini kapsayan bir yol haritası oluşturulur ve modelleme süreci başlatılır. Bu modelleme süreci, iş varlıklarını, iş süreçlerini, potansiyel tehdit oluşturabilecek toplulukları ve bu toplulukların yeteneklerini detaylı bir şekilde ele alır. Sistem, her bir açıdan olası tehditleri belirlemeye çalışır ve bu tehditler, şirket yönetimiyle yapılan görüşmeler neticesinde belirlenen risk düzeylerine göre önceliklendirilir. Öncelikli tehditlerin belirlenmesinin ardından, test uzmanları bu tehditlere karşı etkili bir şekilde mücadele etmek için sızma testi senaryoları geliştirirler. Sızma testi ekibi, testin saldırı stratejisini belirler, kullanılacak araçları seçer ve bir plan oluşturur. Bu planlama süreci, her bir senaryonun ve testin detaylarının titizlikle incelenmesini içerir. Potansiyel zafiyetler, risk seviyelerine, etki derecelerine ve önem düzeylerine göre sıralanır ve detaylı bir şekilde değerlendirilir. Ayrıca, olası risklerin ve etkilerinin belirlenmesi için kapsamlı bir değerlendirme yapılır. Böylece zafiyet analizinin yönünü belirlenir.
Bu kapsamlı yaklaşım, testin etkinliğini artırır ve şirketin güvenlik önlemlerini daha sağlam bir temele oturtmasına yardımcı olur.

2.3.4 Zafiyet Analizi

Sistem ve uygulamalardaki zafiyetlerin tespit edilmesi, siber güvenlik alanında kritik öneme sahip bir süreç olan zafiyet analizinin temelini oluşturur. Bu analiz, bir organizasyonun bilgi teknolojisi altyapısında potansiyel güvenlik açıklarını belirlemek ve bu açıkların etkilerini anlamak için önemlidir.

Zafiyet analizinde, sızma testi ekibi, testin derinliğini ve kapsamını özenle belirler. Derinlik, her bir sistem veya uygulamanın kimlik doğrulama, erişim kontrolleri ve güvenlik önlemleri gibi kritik bileşenlerini detaylı bir şekilde incelemeyi içerir. Bu inceleme, sistemin savunma mekanizmalarının etkinliğini değerlendirmeyi amaçlar. Kapsam ise, testin hangi sistemlerin ve uygulamaların dahil edileceğini ve hangi ağ segmentlerinin test edileceğini belirler. Bu aşamada, organizasyonun iş ihtiyaçları, kritik varlıkları ve risk toleransı gibi faktörler dikkate alınır. Kapsamlı bir zafiyet analizi, organizasyonun tüm dijital varlıklarını koruma stratejilerini geliştirmesine ve siber saldırılara karşı daha hazırlıklı olmasına yardımcı olur.

Zafiyet analizi sürecinde, aktif testler, pasif gözlemlemeler, araştırmalar ve doğrulama adımları gibi farklı yöntemler kullanılır. Bu adımların tamamlanmasıyla, organizasyonun güvenlik zayıflıkları ve risk alanları hakkında kapsamlı bir anlayış elde edilir. Uzman bir test ekibi, deneyim ve bilgi birikimi sayesinde bazen daha önce bilinmeyen (zero-day) zafiyetleri bile tespit edebilir ve organizasyonun savunma stratejilerini güçlendirmesine yardımcı olabilir.

Aktif testler, uygulama arayüzü gibi üst seviyeden veya düşük seviyeli bir ağ bileşeniyle etkileşime geçebilir. Bu etkileşimler, manuel ve otomatize yöntemlerle gerçekleştirilebilir. Örneğin, aktif port taraması gibi işlemler, manuel olarak oldukça zaman alabilir; ancak, otomasyon ile yapılarak zaman tasarrufu sağlanır ve test uzmanının diğer alanlara daha fazla odaklanabilmesi mümkün hale gelir [23]. Otomatize port, servis veya direkt zafiyet tarayıcı araçlar mevcuttur. Bu araçların kullanılması zaman tasarrufu sağlayabilir, ancak otomatize sistemlerde hata payı bulunmaktadır. Bu nedenle, tespit edilen zafiyetler, açık portlar veya servisler manuel olarak da doğrulanmalıdır. Bu doğrulama süreci, zafiyet analizinin daha güvenilir verilere dayanmasını sağlar ve saldırı vektörlerinin belirlenmesinde aktif bir rol oynar.

Pasif gözlemlerde, şirket içinden elde edilen dosyaların metadata analizi yapılabilir. Dosyanın oluşturan kişi, oluşturma tarihi, son kaydetme tarihi ve kullanıcının unvanı gibi bilgiler metadatayı oluşturur. Başka bir pasif veri elde etme yöntemi ise ağ trafiğinin dinlenmesidir. Bazen ağ paketlerin de veri sızıntıları oluşabilir ve bu sızıntılar dışarıya açık ve güvensiz bir şekilde yayılabilir. Bu veriler yeterince toplandığında, önemli bilgiler elde edilebilir.

Doğrulama aşamasında, sızma testi uzmanları genellikle aynı ana bilgisayarda birden fazla araç kullanarak elde ettikleri tekrarlanan belirli zafiyetlerin mikro sorunlarına odaklanır. Ancak bu durum, test çıktısındaki istatistiksel sonuçları yanıltıcı bir şekilde etkileyebilir ve yanlış bir risk profili oluşturabilir. Belirli korelasyon, belirli bir tanımlanabilir soruna ilişkin bilgileri (zafiyet kimliği, CVE[1], OSVDB[2], satıcı dizinleme numaraları vb.) mikro faktörlerle (makine adı, IP, FQDN[3], MAC Adresi vb.) gruplandırır. Örneğin, aynı zafiyeti CVE numarasına göre gruplandırarak, farklı araçlardan gelen bulguları birleştirebiliriz. Böylece zafiyetler tekrara uğramaz ve farklı araçlardan faydalanarak daha doğru sonuçlar elde edilir.
Araştırma aşamasında ise tespit edilen zafiyetlerin kaynağı araştırılır ve bir açık kaynak yazılım veya aracından kaynaklanıp kaynaklanmadığı incelenir. Ayrıca, zafiyetin ne kadar kolay ulaşılabilir ve kullanılabilir olduğu değerlendirilir. Genellikle, bulunan zafiyetler bir sürecin açığı olabileceğinden dolayı hızla çözüme kavuşturulabilir.

[1] CVE, kısaca "Common Vulnerabilities and Exposures" (Sık Görülen Güvenlik Açıkları ve Maruziyetler) anlamına gelir. CVE, bilgisayar sistemlerinde bulunan güvenlik açıklarını ve zayıflıkları tanımlamak için kullanılan bir standart numaralandırma sistemidir.

[2]OSVDB, "Open Sourced Vulnerability Database" (Açık Kaynaklı Güvenlik Açıkları Veritabanı) anlamına gelir. OSVDB, bilgisayar güvenliği alanında yaygın olarak kullanılan bir veritabanıdır ve bilgisayar sistemlerindeki güvenlik açıklarını, zafiyetleri ve maruziyetleri kaydetmek ve takip etmek için tasarlanmıştır. CVE ve OSVDB bilgisayar güvenliği alanında kullanılan iki farklı güvenlik açıkları veritabanıdır. İşlevleri ve kullanımları bakımından farklılıklar bulunmaktadır. OSVDB, bağımsız bir proje olup, kullanıcılar tarafından sağlanan verilere dayanarak güvenlik açıkları hakkında bilgi sağlar. Ancak, OSVDB projesi 2016 yılında sonlandırılmıştır.

[3]FQDN, "Fully Qualified Domain Name" (Tam Tanımlı Alan Adı) kısaltmasıdır. Bir FQDN, bir ağ üzerindeki bir cihazın veya hizmetin tam ve benzersiz adını ifade eder. Bir FQDN, genellikle üç bileşenden oluşur: host adı, alt alan ad(lar)ı ve üst seviye etki alanı adı. Örneğin, "mail.example.com" bir FQDN'dir.

2.3.5 İstismar

İstismar aşaması, sızma testi sürecinin en kritik ve zorlu adımlarından biridir. Bu aşamada, sızma testi ekibi, hedef sistemde tespit edilen güvenlik açıklarını kullanarak yetkisiz erişim elde etmeye veya sistemin normal işleyişini aksatmaya çalışır. Sızma testi ekibinin amacı, gerçek dünyadaki saldırganlar gibi davranarak, hedef sistemin güvenlik önlemlerini aşmak ve hedef sistem üzerinde kontrol sağlamaktır.

İstismar süreci, yalnızca teknik bilgi ve becerileri değil, aynı zamanda hedef sistemi analiz etme, uygun saldırı tekniklerini seçme ve saldırıları dikkatlice planlama yeteneğini de gerektirir. Bu nedenle, bu aşama genellikle diğer aşamalardan daha fazla zaman alır ve titizlikle yürütülmesi gerekir. İstismar aşaması sırasında, sızma testi ekibi aşağıda belirtilen çeşitli gerçek dünya saldırı tekniklerini kullanır [20]:

- SQL Enjeksiyonu (SQL Injection)
- Ortadaki Adam Saldırısı (Man-in-the-middle (MITM))
- Hizmet Reddi Saldırısı (Denial of Service Attack- DoS)
- Şifre Saldırısı (Password Attack)
- Siteler Arası Betik Çalıştırma Saldırısı (Cross-site scripting - XSS)

Bu saldırı türleri, Bölüm 2.4'te detaylı şekilde incelenecektir.

İstismar aşamasında, saldırı türlerinden birini veya birkaçını hedef sistem üzerinde denemek, mevcut güvenlik açıklarının ne kadar ciddi olduğunu ortaya koymak açısından önemlidir. Ancak bu aşamada, saldırıların dikkatli bir şekilde yürütülmesi gerekmektedir; çünkü hedef sistem üzerinde gerçekleştirilen herhangi bir saldırı, sistemin çökmesine veya işlevselliğinin bozulmasına neden olabilir. Sızma testi ekipleri, sistemin güvenlik mekanizmalarını atlatmaya çalışırken genellikle şu tür engellerle karşılaşırlar:

1. **Güvenlik Yazılımları**: Çoğu sistem anti-virüs yazılımı, saldırı tespit sistemleri (Intrusion Detection Systems - IDS), saldırı önleme sistemleri (Intrusion Prevention System - IPS) ve web güvenlik duvarları (WAF) ile korunur. Sızma testi ekipleri, bu yazılımları aşarak saldırılarını gerçekleştirmek zorundadır.
2. **Şifreleme ve Kod Karartma Teknikleri**: Modern sistemler veri güvenliğini sağlamak için şifreleme (encryption) ve kod karartma (obfuscation) tekniklerini

kullanır. Bu yöntemler, sızma testi ekibinin şifrelenmiş verileri çözmesini ya da karartılmış kodları analiz etmesini zorlaştırır.

3. **Gelişmiş Güvenlik Mekanizmaları**: Sistemlerde kullanılan beyaz liste (*whitelisting*) ve kara liste (*blacklisting*) stratejileri, saldırıların engellenmesi için önemli bir güvenlik önlemidir. Beyaz listeleme, yalnızca belirlenen güvenli kaynaklardan gelen işlemlere izin verirken, kara listeleme bilinen zararlı kaynakları engeller. Sızma testi ekibi, bu güvenlik önlemlerini aşmak için daha karmaşık saldırı teknikleri geliştirmek zorundadır.
4. **Saldırı Tespit ve Önleme Sistemleri**: IDS ve IPS sistemleri, ağ trafiğini analiz ederek şüpheli davranışları tespit eder ve saldırıları engeller. Bu tür sistemlerin etkin olduğu ortamlarda sızma testi ekipleri, saldırılarını bu sistemlerden gizlemek zorundadır.
5. **Ağ Segmentasyonu ve İleri Düzey Erişim Denetimleri**: Kurumlar genellikle ağlarını segmentlere ayırır ve her segment için ayrı erişim denetimleri uygularlar. Sızma testi ekipleri, bu önlemleri aşarak farklı ağlara sızmayı hedefler.
6. **Gerçek Zamanlı İzleme ve Alarmlar**: Şüpheli faaliyetlerin tespit edilmesi durumunda yöneticilere anında bildirim gönderen gerçek zamanlı izleme ve alarm sistemleri kullanılır. Sızma testi ekipleri, bu alarmları devre dışı bırakmaya ya da farklı bir saldırı stratejisi geliştirmeye çalışır.

İstismar aşamasında bulunan zafiyetlerin kritiklik düzeyi, sızma testinin genellikle en önemli sonuçlarından biridir; çünkü bu aşama, sistemin güvenlik seviyesinin ne derece zayıf olduğunu açık bir şekilde ortaya koyar. İstismar aşaması tamamlandığında, sızma testi ekibi ele geçirdiği sistemde kalıcı bir yer edinme aşamasına geçer. Bu süreçte, daha önce ele geçirilen sistemde, saldırıların devamlılığını sağlamak için çeşitli yöntemler kullanılır. Ayrıca, sistemde daha derinlere inmek ve ek hedefler keşfetmek amacıyla yatay hareket (lateral movement) stratejileri devreye sokulabilir.

2.3.6 İstismar Sonrası

Sızma testinin istismar sonrası aşamasında, ele geçirilen bir makine üzerindeki kontrolü sürdürmek amaçlanır. Bu aşamada, sızma testi ekibi ağ arayüzleri, yönlendirme tabloları, Alan Adı Sistemi (Domain Name System - DNS) ayarları ve vekil sunucuları (proxy servers) gibi bileşenleri analiz ederek ek hedefler belirler ve uzun vadeli erişim için arka kapı

programları yükler. Gerektiğinde, test sonrası sistemlerde izleri silmek için temizlik işlemi de yapılır.

Uzun Vadeli Erişim Sağlama

1. *Arka Kapı Yerleştirme:* Arka kapılar, sistem yeniden başlatılsa ya da oturum kapatılsa bile saldırganın sisteme erişimini sürdürmesini sağlayan yazılım ya da yöntemlerdir. Bu, bir hizmeti manipüle etmek ya da zararlı bir yazılım yerleştirmek suretiyle yapılabilir. Arka kapılar, saldırganın ileriki aşamalarda sisteme daha hızlı ve fark edilmeden erişmesini sağlar.
2. *Yetki Yükseltme:* Sızma testi sırasında elde edilen yetkiler her zaman sistemdeki en yüksek yetkiler olmayabilir. İstismar sonrası aşamada saldırgan ya da sızma testi ekibi, yönetici (root) yetkilerini ele geçirmeye çalışır. Bu yetkiler, sistem üzerinde tam kontrol sağlamanın yanı sıra daha fazla hedefe ulaşmada da kullanılabilir.
3. *Bilgi Çıkışı:* Ele geçirilen sistemde ya da ağda önemli veriler bulunabilir. Bu aşamada, kritik verilere erişilip bu verilerin dışarıya sızdırılması test edilir. Bu veriler, kimlik bilgileri, müşteri bilgileri, ticari sırlar veya başka hassas bilgiler olabilir. Bilgi çıkışı, genellikle saldırganların nihai hedeflerinden biridir ve ciddi veri ihlallerine neden olabilir.
4. *Yatay Hareket Kabiliyeti:* İstismar sonrası süreçte, saldırganlar mevcut ele geçirdikleri sistemleri kullanarak başka sistemlere erişmeye çalışır. Yatay hareket, aynı ağ içinde ya da başka bir ağ segmentine geçmek anlamına gelir. Bu süreç, test ekibinin ağın ne kadar derinine nüfuz edebileceğini ve hangi diğer cihazları hedef alabileceğini anlamak için önemlidir. Bu, birden fazla sistemi kontrol altına almayı ve en kritik verilere ulaşmayı sağlayabilir.

İzlerin Temizlenmesi

Gerçek dünya saldırılarında, saldırganlar genellikle sistemdeki izlerini temizleyerek kendilerini gizlerler. Bu süreç, sızma testi sırasında da simüle edilebilir. Log dosyalarının silinmesi veya log manipülasyonu, sistemde yapılan işlemler hakkında iz bırakmamak için yaygın kullanılan yöntemlerdir. Ayrıca, saldırı sırasında kullanılan araçlar ve arka kapılar da kaldırılır ya da tespit edilmesi zor olacak şekilde gizlenir. Bu adım, saldırganın sistemde uzun süre fark edilmeden kalabilmesi için kritik öneme sahiptir. Ancak, sızma testleri tamamlandıktan sonra test edilen sistemlerin normal çalışma düzenine dönmesi için bu izlerin tamamen temizlenmesi gerekmektedir.

1. *Log Manipülasyonu:* Test sırasında kullanılan araçların ve yapılan işlemlerin sistemde iz bırakmaması için olay günlüklerinde değişiklik yapılabilir. Bu da saldırının tespit edilmesini zorlaştırır.
2. *Zararlı Yazılım (Malware) veya Kod Bloğu (Script) Kaldırma:* Sızma testi sırasında kullanılan zararlı yazılımlar veya komut dosyaları testin sonunda kaldırılmalıdır. Bu, sistemin normal çalışma durumuna geri döndüğünden emin olmak için kritik bir adımdır.
3. *Ağ Trafiği İzleme ve Gizleme:* Saldırının tespit edilmesini önlemek amacıyla ağ trafiğinde yapılan değişiklikler ve gizli bağlantılar da temizlenmelidir.

2.3.7 Raporlama

Raporlama aşaması, sızma testinin en önemli çıktılarından biridir ve gerçekleştirilen testlerin sonuçlarının hem teknik ekip hem de yönetim seviyesine uygun biçimde sunulmasını amaçlar. Yapılan tüm test ve analizler tamamlandıktan sonra, tespit edilen zafiyetlerin detaylandırıldığı kapsamlı bir teknik rapor hazırlanır. Teknik rapor, sızma testi süresince kullanılan yöntemleri, tespit edilen güvenlik açıklarının ciddiyetini ve her bir zafiyetin ortaya çıkarılmasında uygulanan teknikleri ayrıntılı bir şekilde ele alır. Ayrıca, sektörde yaygın olarak kullanılan diğer test metodolojilerine referanslar da bu rapor kapsamında sunulur. Bu şekilde, teknik ekip, hangi yöntemlerin ve araçların kullanıldığını net bir şekilde anlayarak, bulguların güvenilirliği ve geçerliliği hakkında daha kapsamlı bir değerlendirme yapabilir.

Yönetim seviyesine sunulan rapor ise, daha stratejik ve özetleyici bir yaklaşımla hazırlanır. Bu rapor, belirlenen zafiyetlerin risk seviyeleri ve bu zafiyetlerin kuruma veya sistemin işleyişine doğurabileceği potansiyel etkiler üzerine kapsamlı bir analiz sunar. Yönetim düzeyinde hazırlanan rapor, genellikle teknik detaylardan arındırılarak, güvenlik açıklarının iş sürekliliği, itibar kaybı ve mali zararlar gibi kritik konular üzerindeki etkilerine odaklanır. Bu değerlendirme, üst düzey yöneticilere, kurumun mevcut siber güvenlik durumunu anlamaları ve alınması gereken önlemler hakkında bilinçli kararlar verebilmeleri için gerekli olan bilgileri sağlar.

Her iki raporda da, tespit edilen güvenlik açıklarına yönelik düzeltici ve önleyici öneriler sunulur. Teknik raporda, her bir zafiyetin nasıl giderilebileceği üzerine ayrıntılı teknik çözümler ve en iyi uygulama önerileri yer alırken, yönetim seviyesindeki rapor, bu güvenlik

açıklarının kapatılmasının sağlayacağı işlevsel ve operasyonel faydaları vurgular. Özellikle, zafiyet yönetimine yönelik öneriler, siber güvenlik politikalarının iyileştirilmesine ve sistemlerin genel güvenlik duruşunun güçlendirilmesine yönelik somut adımlar içerir.
Bu raporlama süreci, sızma testinin bütünsel bir çerçevede değerlendirilebilmesi açısından kritik bir öneme sahiptir. Raporlar hem teknik hem de yönetim seviyesinde, kurumun güvenlik açıklarını kapatma yönündeki eylem planlarını şekillendirmekle kalmaz, aynı zamanda kurumun uzun vadeli siber güvenlik stratejilerine katkı sağlar. Sonuç olarak, bu sistematik ve hedefe yönelik raporlama yaklaşımı, kurumun genel siber güvenlik duruşunu güçlendirmek için gerekli olan stratejik kararların alınmasını kolaylaştırır ve potansiyel tehditlere karşı daha dirençli bir savunma mekanizması oluşturulmasına zemin hazırlar.

2.4 İstismar Saldırıları

Bu bölümde sistem de bulunan zafiyetlere nasıl saldırı testlerin yapıldığını göreceğiz. Sızma testi süreçlerinde bahsedilen istismar tiplerinin bazılarından bahsedilecektir.

2.4.1 SQL Enjeksiyonu (SQL Injection)

Saldırgan, SQL enjeksiyonu gerçekleştirmek için, normal bir SQL komutunun anlamını değiştirmek amacıyla kodun içine anahtar kelimeler veya operatörler yerleştirdiğinde, bir SQL saldırısı gerçekleşmiş olur. Kötü niyetli SQL ifadeleri, güvenlik açığı bulunan bir uygulamada, farklı giriş mekanizmaları aracılığıyla uygulamaya dahil edilebilir [24] .

2.4.1.1 Çerezler Aracılığıyla Enjeksiyon

SQL enjeksiyonu, çerezler aracılığıyla da gerçekleştirilebilir. Çerezler, istemci tarafında yerel bir dosyada saklanır ve kullanıcının oturum durumunu koruyarak web uygulamalarında kullanılır. Kötü niyetli bir saldırgan, çerezlere zararlı kodlar enjekte edebilir. Eğer web uygulaması, kullanıcı oturumlarını çerezler aracılığıyla izliyorsa, saldırgan bu çerezlere zararlı SQL kodlarını gömebilir ve böylece sunucuya gönderilen SQL sorgularını manipüle edebilir.

2.4.1.2 Sunucu Değişkenleri Üzerinden Enjeksiyon

Sunucu değişkenleri, Hipermetin Transfer Protokolü (Hypertext Transfer Protocol-HTTP), ağ başlıkları ve çevresel değişkenler (bir kullanıcının oturum bilgileri, geçerli dizin yolu vb.) gibi bir dizi veriyi kapsar. Web uygulamaları, bu sunucu değişkenlerini kullanıcı etkinliğini izlemek ve tarama desenlerini analiz etmek gibi çeşitli amaçlar için kullanır. Ancak, bu

değişkenler uygun temizlenmeden bir veri tabanında saklanırsa, bu bir SQL enjeksiyonu zafiyeti yaratabilir. Kötü niyetli aktörler, HTTP ve ağ başlıklarında saklanan değerleri manipüle ederek, SQL enjeksiyon saldırılarını doğrudan başlıklara yerleştirebilirler. Sonuç olarak, veri tabanı sunucu değişkenlerini kaydetmeye çalıştığında, başlıklardaki enjekte edilmiş SQL kodu çalıştırılabilir ve potansiyel güvenlik ihlallerine yol açabilir.

2.4.1.3 İkinci Dereceden Enjeksiyon

İkinci dereceden enjeksiyonlar, saldırganların bir sistem veya veri tabanına kötü niyetli girdiler ekleyerek, bu girdiler daha sonra kullanıldığında dolaylı olarak bir SQL enjeksiyon saldırısını tetiklemesidir. Bu tür saldırının amacı, tipik bir (yani birinci dereceden) enjeksiyon saldırısından önemli ölçüde farklılık gösterir. İkinci dereceden enjeksiyonlar, kötü niyetli girişin veri tabanına ilk gönderildiği anda saldırının gerçekleşmesini sağlamaya odaklanmaz. Bunun yerine, saldırganlar, girdinin daha sonradan hangi amaçla kullanılacağını bilerek ve saldırılarını bu kullanım sırasında gerçekleşecek şekilde planlarlar.

Şekil 2.3'te görülen bir şifre güncelleme SQL sorgusu üzerinde nasıl bir enjeksiyon örneği yapılabileceğini aşağıda verilmektedir.

```
queryString="UPDATE users SET password='" + newPassword + "' WHERE
    userName='" + userName + "' AND password='" + oldPassword + "'"
```

Şekil 2.3: Şifre güncelleme sorgusu.

Burada newpasword yerine istediğimiz değeri (ör:*newpwd*) , oldpassword yerinede rastgele bir (ör: *oldpwd*) değer ve username yerine "`admin' --`" yazılırsa SQL sorgusu Şekil 2.4'deki haline dönüşür.

```
UPDATE users SET password='newpwd'
WHERE userName= 'admin'--' AND password='oldpwd'
```

Şekil 2.4: Enjeksiyon sorgusu.

"--" operatöründen sonraki alanlar yorum satırı haline gelir ve şifre kısmı etkisiz hale gelir.

Bu SQL çalıştığı zaman admin kullanıcısının istediğimiz ve SQL sorgusuna yazdığımız şifresiyle girebiliyor olacağız.

2.4.1.4 SQL Zafiyet Uygulamaları

Zafiyet uygulaması, OWASP Mutillidae II sanal sunucu üzerinden yapılmıştır. http://127.0.0.1:8080/mutillidae/ sayfasına gidilir. Şekil 2.5'te sisteme kullanıcı bilgileri ile giriş yapılırken kullanılan arayüz üzerinde veri girişi yapılacak kısımlara manipülatif operatör veya ifadeler yazılarak SQL sorgusu üzerinden sızma yapılmaktadır.

Şekil 2.5: Mutillidae test sayfası.

Güvenlik Seviyesi 0

1.Yöntem

```
Select * from accounts where username ='admin' and password='$PASSWORD'
```

Password kutusuna enjeksiyon edilecek metin : "`Aaa' or 1=1 #`"

Sonuç: Giriş yapıldı.

2.Yöntem

```
Select * from accounts where username ='$USERNAME' and password='$PASSWORD'
```

Name kutusuna enjeksiyon edilecek metin: "`admin' #`"

Sonuç: Giriş yapıldı.

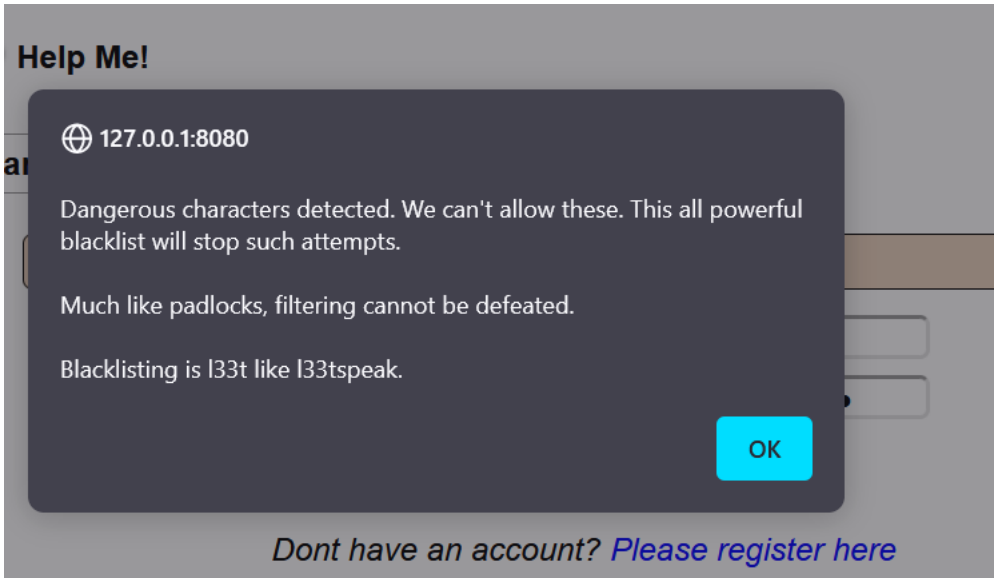
Bu iki yöntem ile giriş işlemindeki zafiyet kullanılarak giriş başarıyla yapılmaktadır.

Güvenlik Seviyesi 1

```
Select * from accounts where username ='admin' and password='$PASSWORD'
```

Password kutusuna enjeksiyon edilecek metin: "`Aaa' or 1=1 #`"

Sonuç: Aynı işlem burada yapılırken filtreye takıldığından dolayı Şekil 2.6'te görüldüğü üzere engellenmektedir.

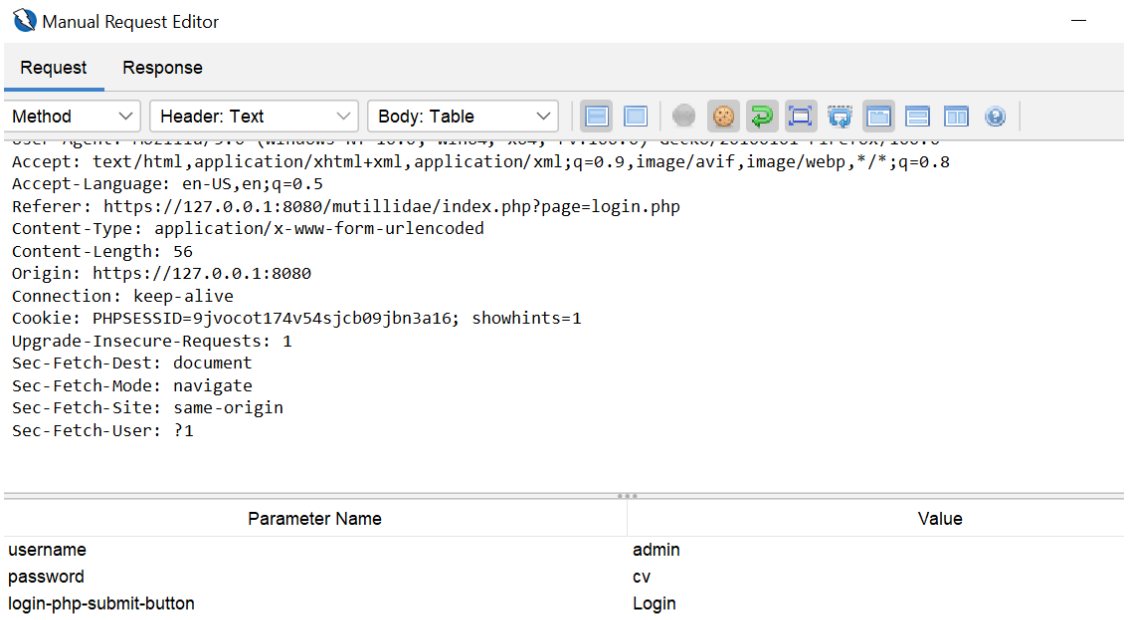


Şekil 2.6: Hata mesajı.

Bu güvenlik seviyesinde http isteğini manipüle etmek için bir Proxy aracına ihtiyaç duyulur. Owaps Zap Proxy, burada devreye girer. Web zafiyetlerini otomatik olarak tespit etmeyi sağlayan açık kaynak kodlu ve ücretsiz bir web güvenlik tarayıcısıdır. Şekil 2.7'te görüldüğü üzere tarayıcı ile sunucu arasına girerek http isteğinin manipüle edebilmektedir [25]. Öncelikle yanlış şifre ile bile olsa kullanıcı adı ve şifre kısmı doldurulur ve istek gönderilir.

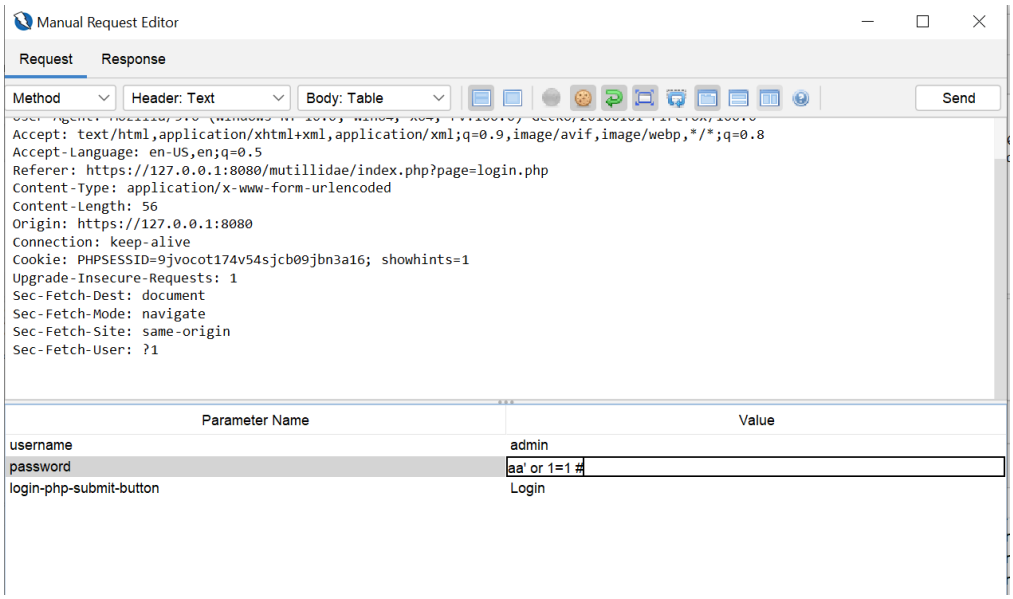
Şekil 2.7: Proxy yöntemi.

İstek Şekil 2.8'te olduğu gibi http istek yönetiminden açılır ve istediğimiz gibi manipüle edilebilir.



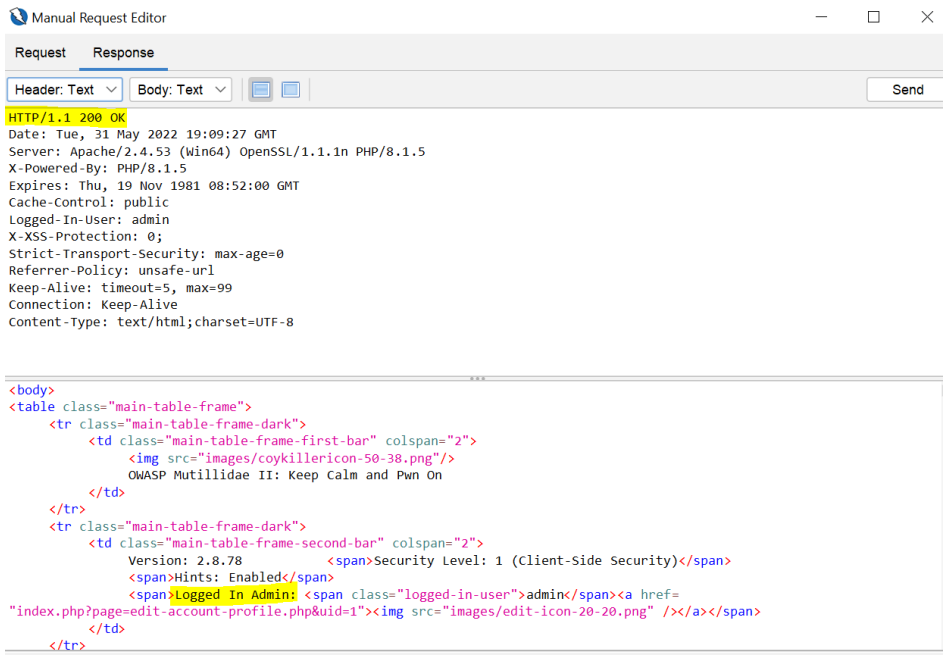
Şekil 2.8: Manipüle edilecek olan istek.

İstek Şekil 2.9'te görüldüğü üzere bu adımda manipüle edilerek, tekrar sunucuya gönderilir.



Şekil 2.9: Manipülasyon işlemi.

Proxy sayesinde filtreden geçen istek yakalanarak manipüle edilir, Şekil 2.10'te görüldüğü üzere tekrar sunucuya istek gönderildiğin de sızma başarıyla gerçekleşmiş olur. Ekran görüntüsünde http isteğinin başarılı (200) koduyla cevap döndüğünü ve sayfa kodu incelendiğinde "*logged in Admin*" görülmektedir.



Şekil 2.10: Manipülasyon sonrası başarılı istek.

2.4.2 Ortadaki Adam Saldırısı (Man in the middle - MITM)

Ortadaki adam saldırısı, iki taraf arasındaki iletişim trafiğine sızarak, tarafların farkında olmadan veri akışını dinlemeye, manipüle etmeye veya yönlendirmeye dayanan ciddi bir siber saldırı türüdür. Bu saldırı türü, özellikle güvenli olmayan ağ bağlantılarında veya zayıf şifreleme protokollerinin kullanıldığı ortamlarda gerçekleştirilebilir. MITM saldırıları, saldırganın aktif olarak veri iletişimini izlemesine, değiştirmesine veya kimlik bilgilerini çalmasına olanak tanır. Bu durum, özellikle kişisel bilgiler, banka verileri veya hassas şirket bilgileri gibi kritik verilerin tehlikeye girmesine neden olabilir.

Bir MITM saldırısında, saldırgan iki cihaz ya da sunucu arasında bir köprü gibi davranır. Bu sayede, iki taraf arasındaki veri trafiği saldırganın üzerinden geçer. Normalde iki cihaz arasında gerçekleşmesi gereken iletişim, saldırganın araya girmesi ile onun kontrolüne geçer. Böylece saldırgan, hem trafiği izleyebilir hem de trafiği manipüle ederek yanıltıcı bilgi gönderebilir. Bu tür bir saldırı, ağ güvenliğinin zayıf olduğu yerlerde, özellikle de açık

Wi-Fi ağlarında, kolayca gerçekleştirilebilir. Saldırgan, ortadaki adam pozisyonunda, kullanıcıya ait bilgileri çalabilir, sahte kimlik doğrulama bilgileri yaratabilir ya da kullanıcıları zararlı sitelere yönlendirebilir.

MITM saldırıları genellikle şu aşamalardan oluşur:

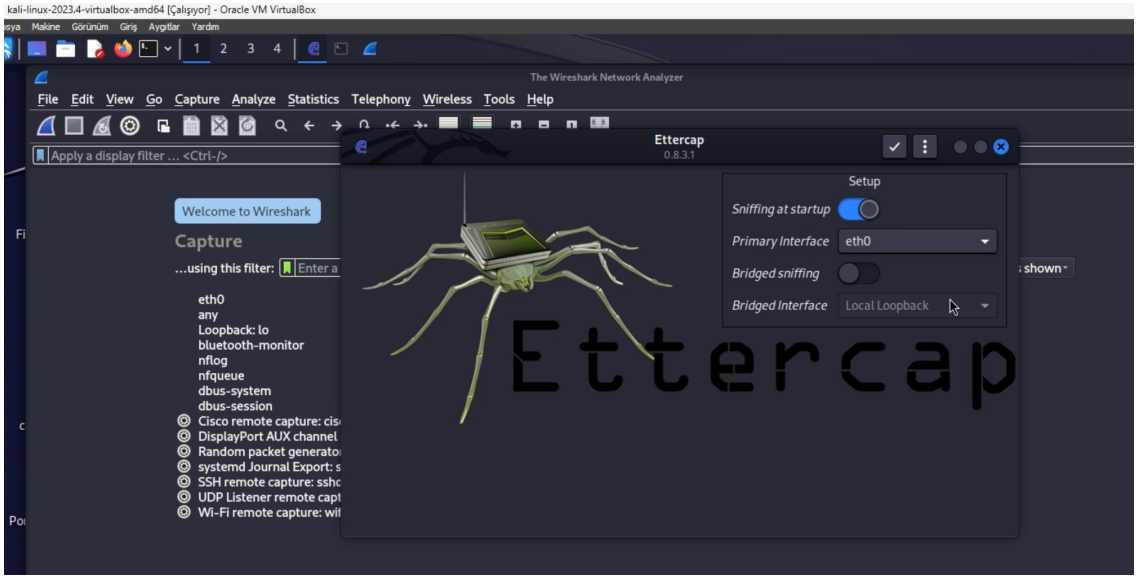
- *Yakalama* (*Intercepting*): Saldırgan, iki taraf arasındaki iletişimi kesmek veya yönlendirmek için bir yöntem bulur. Bu, genellikle ARP sahteciliği, DNS zehirlemesi veya Güvenli Soket Katmanı (Secure Sockets Layer - SSL) / Taşıma Katmanı Güvenliği (Transport Layer Security - TLS) zafiyetleri üzerinden yapılır. Örneğin, ARP sahteciliği ile saldırgan, yerel ağdaki cihazlara kendi MAC adresini yönlendirir ve böylece iki taraf arasındaki trafiği kendine çeker.
- *Dinleme* (*Eavesdropping*): Saldırgan, iki cihaz arasındaki trafiği başarılı bir şekilde üzerine aldıktan sonra, bu trafiği izleyebilir. Şifrelenmemiş veriler saldırgan tarafından kolayca okunabilir ve hassas bilgiler elde edilebilir. Şifreli trafiğin ise SSL/TLS protokollerindeki güvenlik açıkları kullanılarak çözülmesi mümkündür.
- *Manipülasyon* (*Manipulation*): Trafiğin saldırganın eline geçmesiyle birlikte, saldırgan iletişimdeki veri paketlerini değiştirebilir, sahte veri gönderebilir veya iletişim akışını bozar. Örneğin, kullanıcının girdiği banka bilgilerinin değiştirilebilmesi ya da bir kullanıcıyı kimlik avı sitesine yönlendirme gibi durumlar bu aşamada gerçekleşir.

MITM saldırıları, farklı yöntemler kullanılarak gerçekleştirilebilir ancak en yaygın olarak ARP sahteciliği yapılarak gerçekleştirilir [6]. Bu yöntem, ARP protokolünün zayıf güvenliği ve saldırının basit bir şekilde uygulanabilmesi nedeniyle tercih edilmektedir. ARP, MAC adresi ile IP adresi arasında bir eşleme yaparak çalışır ve iki tür mesaj kullanır: istek ve yanıt. Bu iletişim, kaynak ve hedef olmak üzere iki cihaz arasında gerçekleşir. ARP'de herhangi bir güvenlik önlemi bulunmadığı için, sahte bir ARP yanıtı bir cihaz tarafından alınabilir ve istekte bulunulmamış olsa bile bu yanıttaki bilgi cihazın belleğine kaydedilebilir. Bu durum, MITM saldırılarının kolayca gerçekleştirilmesine zemin hazırlamaktadır. ARP protokolü ve zafiyeti hakkında Bölüm 2.5'de daha detaylı olarak bahsedilmiştir.

2.4.2.1 MITM Saldırısı Örneği

Uygulamada bir iOS cihazın modem olarak davrandığı ve Windows işletim sistemine sahip iki bilgisayarın bağlı olduğu bir ağ ortamında ARP sahteciliği kullanılarak bir MITM saldırısı gerçekleştirilmiştir.

Bu saldırının gerçekleştirilmesi kullanılan araç Ettercap, Linux tabanlı sistemlerde yaygın olarak kullanılan bir ağ analiz ve saldırı aracıdır. Şekil 2.11'de görüldüğü gibi açılır.

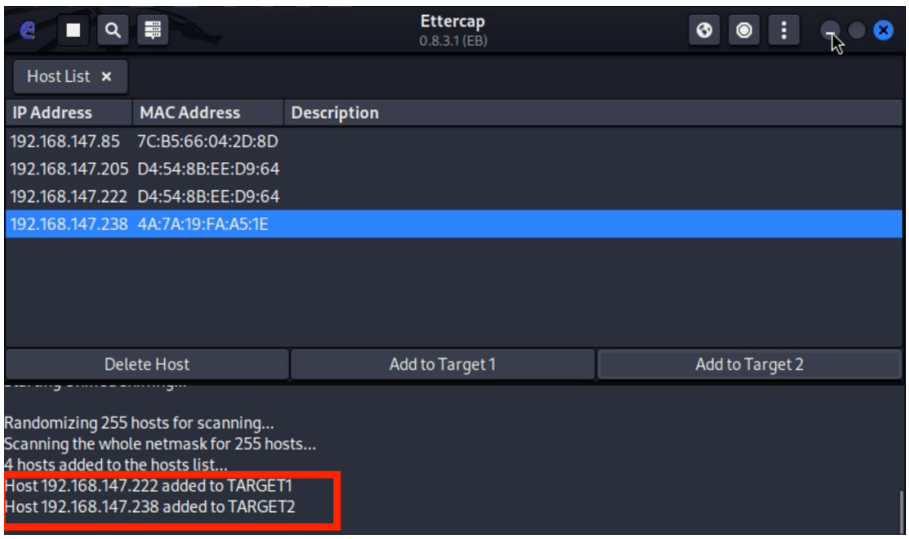


Şekil 2.11: Ettercap açılışı.

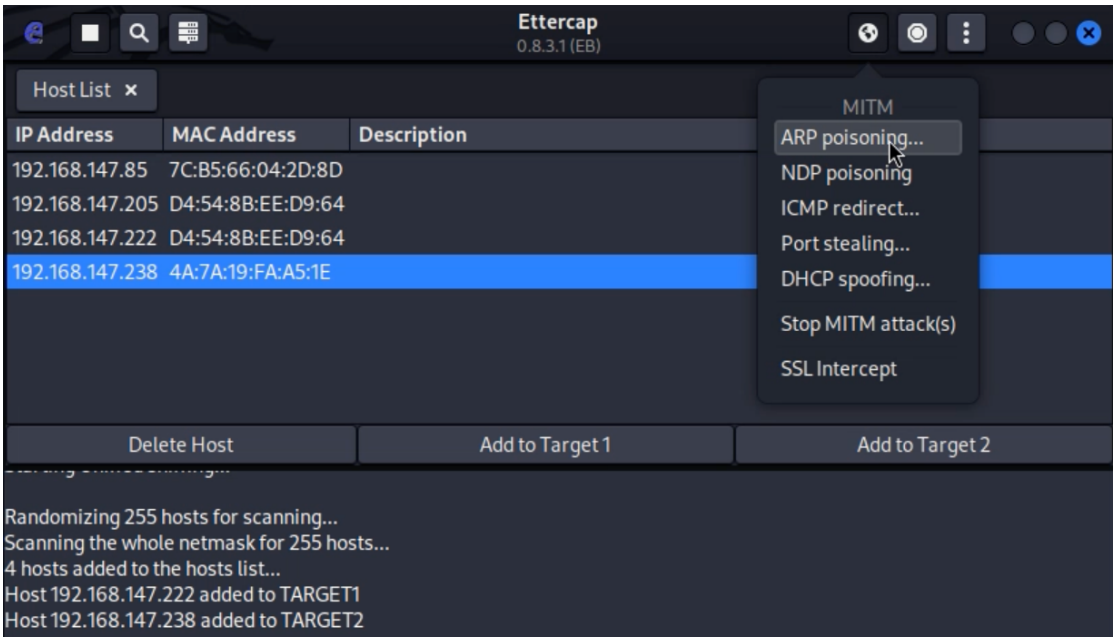
Ettercap başlatıldığında, ilk olarak hangi ağ arayüzünün kullanılacağına karar verilmesi gerekmektedir. Genellikle, kullanılan ağ arayüzü "*eth0*" veya "*wlan0*" olacaktır. Çıkan pencereden uygun ağ arayüzünü seçilir ve onaylanır (örneğin, Şekil 2.11'de görüldüğü gibi *eth0* seçilir). Böylece tarama (unified sniffing) başlatılır. MITM saldırısının başarılı olması için hedef cihazları belirlemeniz gerekmektedir. Ettercap, ağdaki tüm cihazları tarayabilir ve bunları IP adresleriyle birlikte listeler. Ettercap arayüzünde menüden Şekil 2.12'de olduğu gibi arama sembolüne tıklanarak tarama başlatılır ve listeyi yanındaki butona tıklayarak görüntüleyebiliriz.

Şekil 2.12: Tarama başlatma.

Şekil 2.13'te görüldüğü üzere saldırı düzenlenmek istenen cihazlar belirlenir. Örneğin, modem (router) ve kurban cihaz seçilebilir. Ardından ARP sahteciliği kullanılarak MITM saldırısını başlatılacaktır. Bu saldırı, iki cihaz (örneğin, bir kullanıcı cihazı ve modem) arasındaki trafiğin saldırgan üzerinden geçirilmesini sağlar. Şekil 2.14'de görüldüğü gibi menüde yer alan MITM > ARP Poisoning seçeneği izlenerek, "*Sniff remote connections*" seçeneği işaretlenip onaylandığında, iki cihaz arasındaki tüm trafiğin dinlenmesi sağlanmaktadır. Bu işlem, ağ üzerinde gerçekleşen veri iletişimini izlemeye olanak tanımaktadır.



Şekil 2.13: Ağ listesinden hedef cihazların seçimi.



Şekil 2.14: ARP sahteciliğinin başlatılması.

MITM saldırısı ile iki cihaz arasındaki trafik başarılı bir şekilde yönlendirilip kontrol altına alındıktan sonra, ağdaki veri trafiğinin detaylı olarak izlenmesi ve analiz edilmesi amacıyla Wireshark kullanılmıştır. Wireshark, ağ üzerindeki veri paketlerini yakalayarak bu paketlerin içeriğini ayrıntılı şekilde inceleme imkânı sunan güçlü bir ağ analiz aracıdır. Saldırı esnasında Wireshark aracılığıyla ağ trafiği dinlenmiş ve iki cihaz arasında gerçekleşen tüm veri paketleri detaylı olarak incelenmiştir. Özellikle ARP sahteciliği işlemi sonucunda, iki cihaz arasındaki veri alışverişi Wireshark yardımıyla yakalanmış ve analiz edilmiştir.

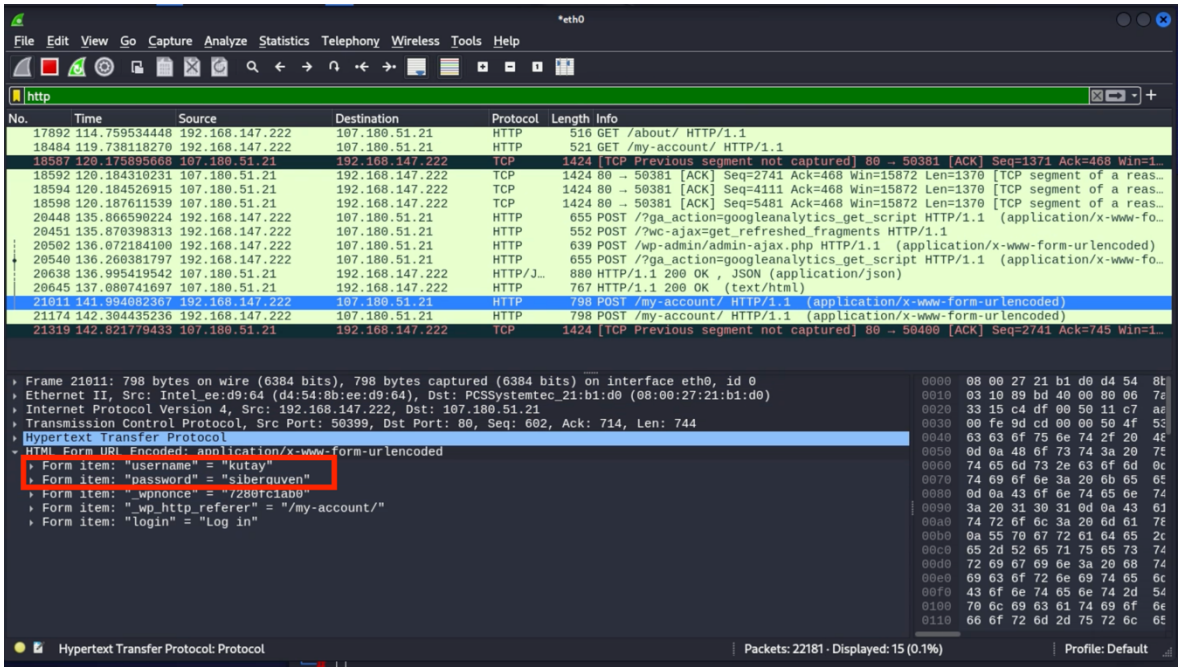
MITM saldırısı yapıldığı sırada mağdur aşağıdaki işlemleri yapmıştır:

- Bir web sayfasına girmiş.
- Kullanıcı adı ve şifresini doğru bir şekilde girmiş.
- Oturumunu açmıştır.

Saldırganın eşzamanlı hamleleri aşağıdaki gibidir:

- MITM saldırısı başarıyla yapılır.
- Herhangi bir ağ izleme aracı açılır (Örneğin Wireshark).
- Saldırganın trafiğini takip etmek için ilgili protokol filtresi uygulanır.
- Gelen ve giden istekler detaylıca incelenerek trafik dinlenir.

Saldırgan, Şekil 2.15'de görüldüğü gibi kullanıcının kullanıcı adı ve şifresini yazıp gönderdiği "HTTP POST" isteğini yakalayarak kullanıcının bilgilerini ele geçirmiştir.



Şekil 2.15: ARP sahteciliği ile trafiğin dinlenmesi.

Bu uygulamadan da anlaşılacağı üzere, kolayca gerçekleştirilebilen ARP sahteciliği saldırıları, bireyler ve kurumlar açısından ciddi güvenlik riskleri taşımakta ve önemli zararlara yol açabilmektedir. Bu tez kapsamında, ağ güvenliğini güçlendirmek amacıyla ARP sahteciliğine karşı etkin bir tespit sistemi geliştirilmesine yönelik çalışılmıştır.

2.4.3 Hizmet Reddi Saldırısı (Denial of Service Attack - DoS)

Hizmet Reddi (Denial of Service - DDoS) veya birden fazla makine/bilgisayar kullanan DoS saldırısı türü olan Dağıtılmış Hizmet Reddi (Distributed Denial of Service - DDoS) saldırıları, web sunucuları için önemli bir tehdit oluşturmaktadır. Birçok ele geçirilmiş sistem, hedefe aşırı trafik göndererek normal hizmet işleyişini kesintiye uğratır[26]. Bu saldırılar genellikle uygulama katmanında HTTP istekleriyle gerçekleştirilir, bu da kötü niyetli trafiğin normal kullanıcı taleplerini taklit etmesi nedeniyle tespit edilmesini zorlaştırır. (D)DoS saldırısının amacı, bant genişliği, Merkezi İşlem Birimi (Central Processing Unit - CPU) ve bellek gibi sunucu kaynaklarını aşırı yükleyerek gerçek kullanıcıların hizmete erişmesini engellemektir. Bu bölümde tek bir makine tarafından tek bir hedefe hizmet reddi (DoS) saldırı örneği çalışması yapılacaktır.

2.4.3.1 DoS Saldırısı Örneği

Yapılan uygulamada, hedef sistemin DoS saldırısına maruz bırakılarak hizmet kesintisine uğratılması için HPing3 aracı kullanılmıştır. İlk aşamada, hedef sistemin domain adresine ping gönderilerek IP adresi elde edilmiştir. Bu işlem, terminale aşağıdaki komutun girilmesiyle gerçekleştirilmiştir:

```
ping hedefdomain.com
```

Bu komut ile hedef domeyne İnternet Kontrol Mesajlaşma Protokolü Yankı İsteği (Internet Control Message Protocol (ICMP) echo request) gönderilmiş ve dönüş yanıtlarından hedef IP adresi elde edilmiştir. Elde edilen IP adresi, sonraki aşamalarda DoS saldırısında kullanılmak üzere kaydedilmiştir.

DoS saldırısının bir parçası olarak İletim Kontrol Protokolü Senkron (Transmission Control Protocol Synchronous – TCP SYN) saldırısı gerçekleştirilmiştir. Bu saldırı türü, hedef sistemin TCP bağlantılarını aşırı yükleyerek hizmetlerin kesintiye uğratılmasını amaçlamaktadır. HPing3 aracı kullanılarak sahte TCP SYN paketleri hedefe gönderilmiş ve bu şekilde sistemin kaynaklarının tüketilmesi sağlanmıştır. Saldırının başlatılması için aşağıdaki komut kullanılmıştır:

```
sudo hping3 -S --flood -V -p 80 hedefIP
```

Bu komutta:

- `-S` bayrağı, TCP SYN paketlerinin gönderilmesini sağlamaktadır,
- `--flood` parametresi, paketlerin sürekli ve hızlı bir şekilde gönderilmesine olanak tanımaktadır,
- `-V` seçeneği, saldırı sürecindeki bilgilerin ayrıntılı olarak gösterilmesini sağlamaktadır,
- `-p 80` parametresi, hedefin 80 numaralı portuna saldırı yapılacağını belirtmektedir; bu port, genellikle web sunucuları tarafından HTTP hizmetleri için kullanılır,
- `hedefIP` ise, ping komutuyla elde edilen IP adresini temsil etmektedir.

Saldırı esnasında hedefe sürekli TCP SYN paketleri gönderilmiş ve bu işlem sonucunda hedefin TCP bağlantıları yoğun bir şekilde aşırı yüklenmiştir. Bu yüklenme sonucunda, Şekil

2.16’te görüldüğü gibi hedef web sayfasının erişilemez/tepkisiz hale geldiği gözlemlenmiştir. Normal kullanıcılar, bu saldırı esnasında hizmetlere erişimde zorluk yaşamış ve web sayfası yüklenememiştir.

Şekil 2.16: DoS saldırısı sırasında hedef web sayfasının tepkisi.

2.4.4 Şifre Saldırısı (Password Attack)

Kaba kuvvet (brute force) şifre saldırısı, bir siber güvenlik saldırısı olup saldırganın şifrelenmiş verilere yetkisiz erişim sağlamak amacıyla her karakter veya değer kombinasyonunu, genellikle sıralı bir şekilde, doğru kombinasyon bulunana kadar sistematik olarak denemesini ifade eder. Doğru şifreyi tahmin edebilmek için tüm olası kombinasyonların denendiği kaba kuvvet saldırısının uzun ve karmaşık şifreler için hesaplama maliyetini artıracağı açıktır.

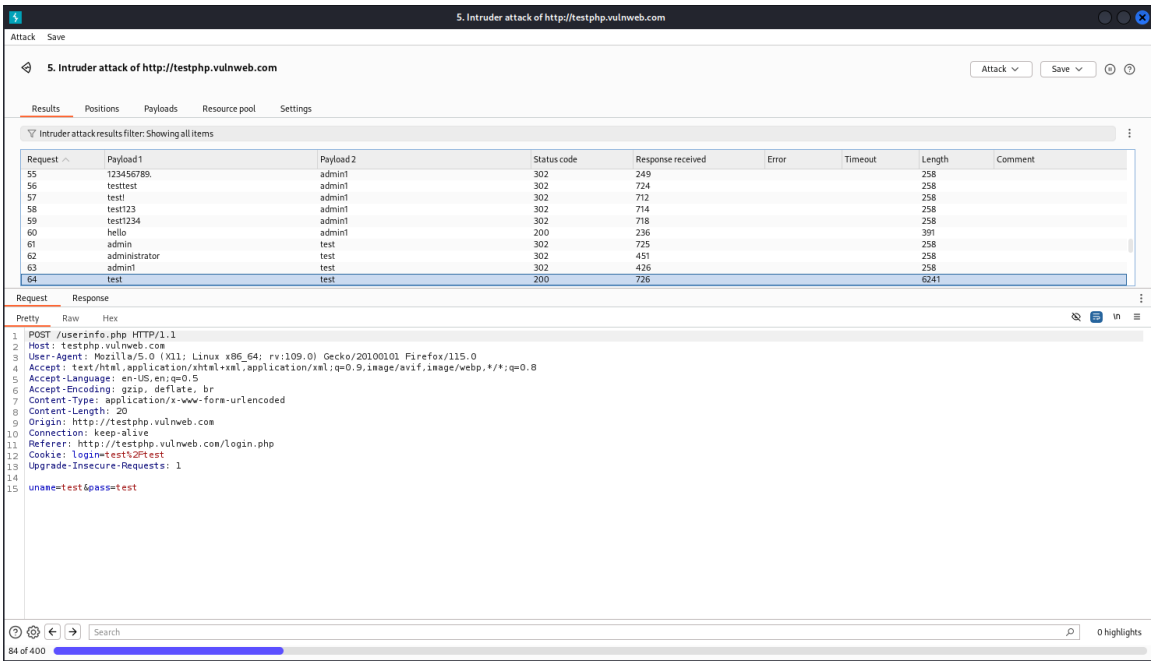
Öte yandan, sözlük saldırısı (dictionary attack) örneğin bir sözlükteki kelimelere veya yaygın şifre listelerine dayanır. Bu saldırı, çoğu kullanıcının yaygın veya tahmin edilebilir şifreler kullandığı varsayımına dayandığından, kaba kuvvet saldırısına kıyasla oldukça hızlıdır [27]. Yapılan testler, işlemci gücü açısından düşük ila orta seviye tek bir GPU'nun bile birkaç gün içinde şifrelerin %95'inden fazlasını kırabildiğini, daha özel bir sistemin ise en güçlü %0,5'lik dilim dışındaki tüm şifreleri çözebildiğini göstermiştir [27].

2.4.4.1 Şifre Saldırısı Örneği

Bu bölümde, kaba kuvvet yöntemi kullanılarak bir şifre saldırısının nasıl gerçekleştirildiği anlatılacaktır. Görseller ile süreç detaylandırılarak saldırının işleyişi ve sonuçları açıklanacaktır. Doğru giriş bilgilerini elde etmek amacıyla, birçok olası şifre ve kullanıcı adı kombinasyonunun sistematik bir şekilde denendiği bir kaba kuvvet saldırısı gerçekleştirilecektir. Bu saldırı yöntemi, özellikle zayıf şifreleme sistemlerine karşı oldukça etkilidir.

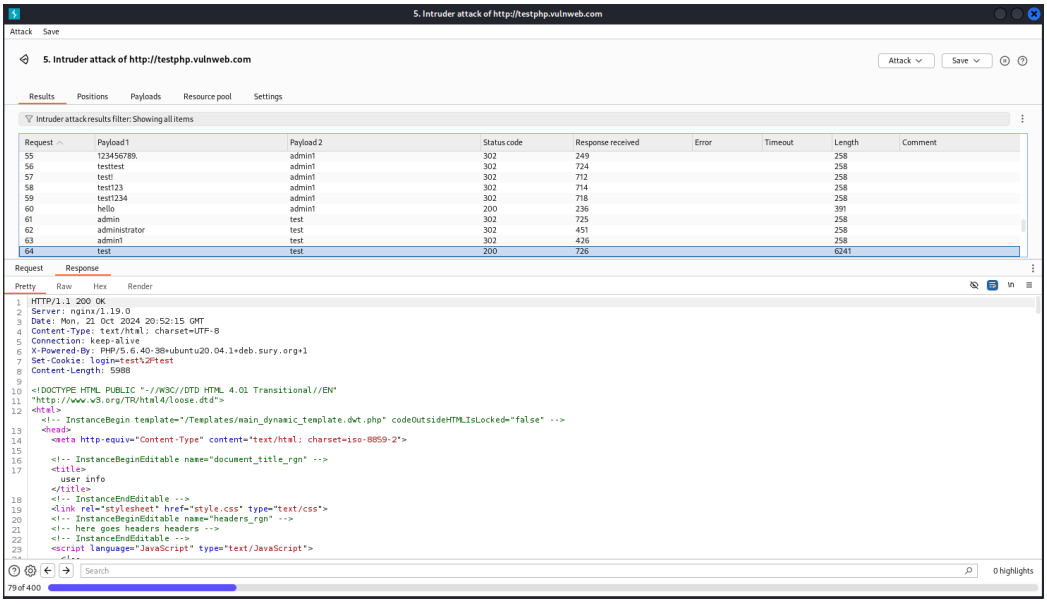
Bu saldırı senaryosunda, Burp Suite'in *Intruder* modülü kullanılarak hedef sisteme ardışık istekler gönderilmektedir. *Intruder* modülü, saldırganın belirli bir parametreyi (örneğin, kullanıcı adı ve şifre alanlarını) belirleyip bu parametre üzerinde çok sayıda deneme yapmasına olanak tanır. Burp Suite, farklı kullanıcı adı ve şifre kombinasyonlarını otomatik olarak deneyerek, bir başarı durumu elde edene kadar sürekli istek göndermektedir. Her deneme için sunucudan alınan yanıtlar kaydedilmekte ve bu yanıtlar üzerinden saldırının başarı durumu değerlendirilmektedir.

Şekil 2.17 ve Şekil 2.18 görsellerinde, Burp Suite kullanılarak gerçekleştirilen kaba kuvvet saldırısının sırasıyla istek ve yanıt bilgileri gözlemlenmektedir. Bu adımda, farklı kullanıcı adı ve şifre kombinasyonlarını kullanılarak hedef sistemin giriş formuna ardışık istekler gönderilmektedir. İsteklerin her birinde farklı "kullanıcı adı" ve "şifre" değerleri denenmektedir (örneğin, admin/test, test/test123 gibi). Gönderilen isteklerin yanıtları incelendiğinde, hedef sunucu her kombinasyona karşı bir yanıt kodu döndürmektedir. Yanıtların 302 ve 200 durum kodları dikkate alınmalıdır. 302 durumu, yönlendirme anlamına gelir ve genellikle hatalı giriş bilgilerine karşı sistemin giriş sayfasına geri yönlendirdiği anlamına gelir. 200 durum kodu ise, başarılı bir girişin gerçekleştiğini ve oturumun açıldığını gösterir.



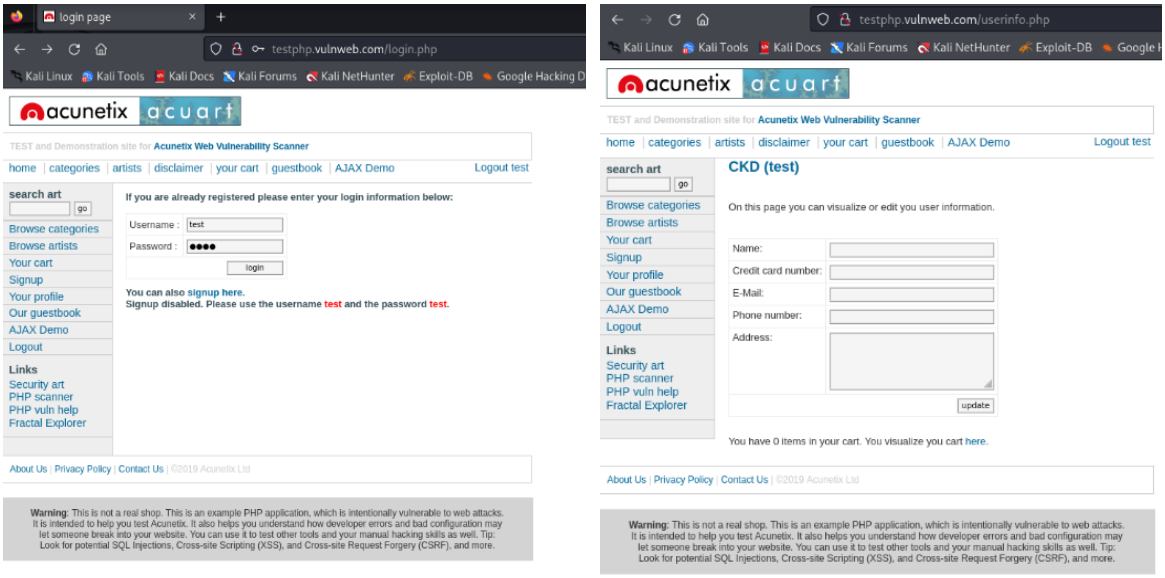
Şekil 2.17: Şifre saldırısı istek detayı.

Aşağıdaki görseldeki istek ve yanıt analizine göre, denenen kombinasyonlardan biri (test/test) başarıya ulaşmış ve hedef sistemden 200 OK yanıtı alınmıştır. Bu durum, doğru kullanıcı adı ve şifre kombinasyonunun bulunduğu anlamına gelir.



Şekil 2.18: Şifre saldırısı yanıt detayı.

Şekil 2.19'de soldaki resimde, Şekil 2.18'de görülen 200 OK yanıtı ile doğrulanan kullanıcı adı ve şifre kombinasyonu (test/test) kullanılarak giriş yapılmıştır. Bu noktada, saldırgan, doğru giriş bilgilerini bularak hedef sistemde oturum açmıştır. Bu işlem, saldırının başarıya ulaştığını göstermektedir. Kullanıcı adı ve şifre girildikten sonra, Şekil 2.19'da sağdaki resimde oturum açıldığına dair bir sayfa görüntülenmektedir. Bu sayfa, başarılı girişin kanıtıdır ve kullanıcının sisteme giriş yaptığını göstermektedir. Giriş ekranında kullanıcı bilgileri ve diğer oturum detayları yer almaktadır.



Şekil 2.19: Doğru kullanıcı bilgileri ile giriş.

Saldırganın sistemde oturum açtıktan sonra kullanıcı profili bilgilerine eriştiği gözlemlenmektedir. Kullanıcıya ait e-posta adresi, adres bilgileri ve diğer kişisel bilgilerin yer aldığı bu sayfa, saldırının başarılı bir şekilde sonuçlandığını ve hedef sistemdeki kullanıcı verilerine ulaşıldığını göstermektedir. Bu aşamada, saldırganın hedef sistemdeki hassas bilgilere erişimi olduğu ve bunları görüntüleyebildiği anlaşılmaktadır. Kaba kuvvet saldırısı sonucunda sistemin zayıf şifre koruma mekanizması aşılmış ve yetkisiz erişim sağlanmıştır.

2.4.5 Siteler Arası Betik Çalıştırma Saldırısı (Cross-site scripting - XSS)

Siteler arası betik çalıştırma saldırısı (Cross-Site Scripting - XSS), saldırganın bir web uygulamasına kötü niyetli komut dosyaları enjekte ettiği bir tür kod enjeksiyonu saldırısıdır. Bu komut dosyaları, genellikle kurbanın haberi olmadan, kurbanın tarayıcısında çalıştırılır

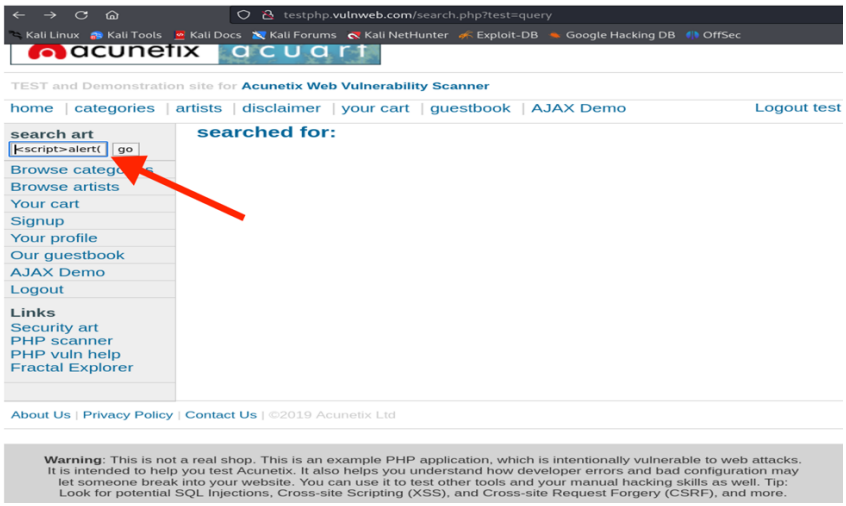
ve çerezlerin çalınması, kullanıcı oturumlarının ele geçirilmesi, kullanıcıların kötü amaçlı sitelere yönlendirilmesi veya kötü amaçlı yazılım yayılması gibi çeşitli zararlı sonuçlara yol açar [28]. XSS saldırıları, web uygulamalarındaki yetersiz girdi doğrulama açıklarını istismar eder ve saldırganın içeriği manipüle etmesine olanak tanır. XSS saldırılarının üç ana türü vardır: Yansıtmalı (kalıcı olmayan), depolanmış (kalıcı) ve belge nesne modeli (Document Object Model - DOM) tabanlı XSS. Yansıtmalı XSS, kötü niyetli komut dosyalarının bir web sunucusu üzerinden yansıtılmasıyla gerçekleşirken, depolanmış XSS, saldırganın sunucuya enjekte ettiği ve web sayfasına erişen kullanıcılar tarafından çalıştırılan kodu içerir. DOM tabanlı XSS ise istemci tarafı kodundaki güvenlik açıklarından kaynaklanır. Bu saldırılar, güçlü girdi doğrulaması, içerik güvenliği politikalarının uygulanması ve XSS tespit araçları kullanılması ile hafifletilebilir.

2.4.5.1 XSS Saldırı Örneği

Burada XSS saldırısının gerçekleştirilme süreci anlatılmaktadır. XSS, web uygulamalarında güvenlik açıklarından faydalanarak kullanıcı tarayıcılarında zararlı komutlar çalıştırmayı amaçlayan bir saldırı türüdür. Bu tür saldırılar, genellikle yetersiz girdi doğrulama mekanizmalarıyla savunmasız hale gelmiş web sitelerinde yapılır ve saldırganların zararlı kodu hedef siteye enjekte etmesine olanak tanır.

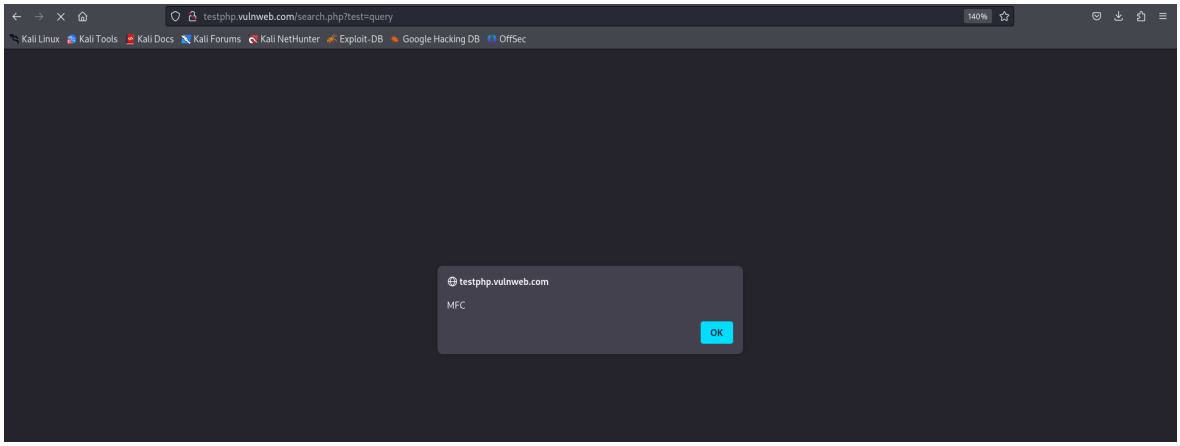
Şekil 2.20'de sunulan görselde, kalıcı olmayan (*reflected*) bir XSS saldırısının başlangıç adımı görülmektedir. Burada, bir web uygulamasının arama veya form alanına, JavaScript kodu enjekte edilmiştir. Enjekte edilen kod aşağıdaki gibidir:

```
<script>alert("MFC")</script>
```



Şekil 2.20: Arama alanına JavaScript kodunun enjekte edilmesi.

Bu kod, web uygulamasının girdi alanına gönderildiğinde, tarayıcıya bu komutun çalıştırılmasını emreder. Web uygulamasının bu girdiyi doğrulamadan doğrudan kabul etmesi ve tarayıcıya iletmesi, güvenlik açığının varlığını gösterir. Özellikle, sunucu tarafında girdi doğrulama mekanizmaları yetersiz olduğu zaman, bu tür saldırılar başarılı olur. Şekil 2.21'de sunulan görselde, gönderilen XSS kodunun tarayıcıda başarılı bir şekilde çalıştığı gözlemlenmektedir. Tarayıcı, bu enjekte edilen JavaScript kodunu algılar ve komutu çalıştırarak bir uyarı penceresi (popup) oluşturur. Bu durumda, ekranda "MFC" içeren bir uyarı mesajı görüntülenir. Bu işlem, saldırganın tarayıcı üzerinde zararlı kod çalıştırabileceğini ve potansiyel olarak daha ciddi saldırılar gerçekleştirebileceğini kanıtlamaktadır.



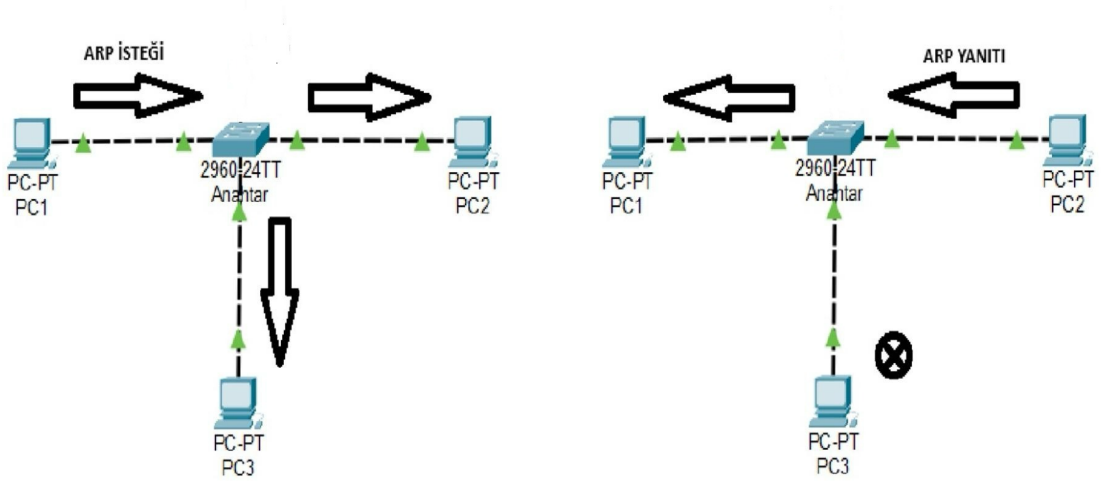
Şekil 2.21: Başarılı XSS saldırısı çıktısı.

2.5 ARP Sahteciliği ve İlgili Çalışmalar

Temel bir ağ protokolü olan Adres Çözümleme Protokolü (ARP – Address Resolution Protocol), bir IP adresini fiziksel bir MAC adresine dönüştürür. Ethernet gibi bir yerel ağda birçok cihazın birbirleriyle iletişim kurmasına izin verir. Bir IP adresine karşılık gelen MAC (Media Access Control) adresini bulmak için ARP, ağa istek paketleri göndererek ilgili cihaza bu adresi vermesini ister. Ağ cihazlarının verileri doğru hedeflere yönlendirebilmesi için bu süreç gereklidir. RFC 826 tarafından tanımlanan ARP, IP adreslerinin 48-bit Ethernet adreslerine sürekli olarak dönüştürülmesini sağlar [5].

Bağlantı katmanı adresleri (MAC adresleri) ile ağ katmanı adresleri (örneğin İnternet IP adresleri) arasında bir çeviri yapılması gerekmektedir. Bu çeviri işlemi Adres Çözümleme Protokolü'nün (ARP) [RFC 826] sorumluluğundadır [29].

Bir ARP istek paketi Şekil 2.23'te görüldüğü gibi göndericinin MAC ve IP adreslerini ve hedef IP adresini içerir ve bu paket yerel ağ segmentindeki tüm cihazlara yayınlanır. ARP isteğini alan ve IP adresi eşleşen bir cihaz, MAC adresini sağlayarak istekte bulunan cihaza ARP yanıtı verir ve böylece adres çözümleme sürecini tamamlar.



Şekil 2.22: ARP istek yanıt yapısı.

Örneğin Şekil 2.22'de görselleştirildiği gibi, PC1 İstemcisi PC2 İstemcisi ile iletişim kurmak istediğinde, PC1 İstemcisi bir canlı yayın ARP isteği gönderebilir. PC2 İstemcisi, kendi IP adresini bu yayında gördüğünde isteğin kendisine yapıldığını anlar ve MAC adresini içeren bir ARP yanıtı ile doğrudan PC1 İstemcisine yanıt verir.

Donanım Türü (2 bayt)		Protokol Türü (2 bayt)
Donanım Adresi Uzunluğu (1 bayt)	Protokol Adresi Uzunluğu (1 bayt)	İşlem Kodu (2 bayt) 1: İstek, 2: Yanıt
Gönderici MAC Adresi (6 bayt)		
Gönderici Protokol Adresi (4 bayt)		
Hedef MAC Adresi (6 bayt)		
Hedef Protokol Adresi (4 bayt)		

Şekil 2.23: ARP paketi.

Bu durumda, ARP isteğine yanıt olarak oluşan iletişimin altında yatan yapı, ARP paketinin bileşenlerine dayanır. Şekil 2.23'te görselleştirildiği gibi, ARP paket yapısı; donanım türü, protokol türü, donanım adresi uzunluğu, protokol adresi uzunluğu, işlem kodu, gönderenin donanım adresi, gönderenin protokol adresi, hedefin donanım adresi ve hedefin protokol adresi gibi veri alanlarını içerir. Bu alanlar, ağ katmanı adreslerinin veri bağlantı katmanı adreslerine doğru bir şekilde dönüştürülmesi için gerekli tüm bilgileri sağlamaktadır [30].

- **Donanım Türü:** ARP, çeşitli ağ donanım türlerinde çalışabilen çok yönlü bir protokoldür ve her donanım türü, kendine özgü bir donanım türü kodu ile temsil edilir. Bu donanım türlerinin derinlemesine anlaşılması, etkili ARP sahteciliği tespit mekanizmalarının geliştirilmesi açısından kritik bir öneme sahiptir. Örneğin, Ethernet ve IEEE 802 ağları gibi yaygın olarak kullanılan donanım türleri, geniş kullanım alanları ve yapılandırma esneklikleri nedeniyle ARP sahteciliği saldırıları için sıkça hedef alınan ağlardır. Aşağıda verilen Tablo 2.1'de ARP sahteciliği de en sık hedef alınan ilk 10 donanım türü listelenmiştir. ARP sahtecilik tespit stratejileri, Tablo 2.1'de sunulduğu gibi farklı ağ ortamlarının özgül zafiyetlerine ve karakteristiklerine uygun olarak uyarlanmasına olanak tanır. Böylece, ağ güvenliği daha kapsamlı ve etkili bir şekilde güçlendirilebilir [5], [31], [32].

- **Donanım Adres Uzunluğu:** MAC adresinin uzunluğu. Ethernet için bu değer 6'dır.
- **İşlem Kodu:** Bu alan, ARP paketinin bir istek mi yoksa yanıt mı olduğunu belirtir. 1 değeri ARP isteği, 2 değeri ARP yanıtı anlamındadır.
- **Gönderenin IP Adresi:** ARP paketini gönderen cihazın IP adresini içerir.
- **Gönderenin MAC Adresi:** ARP paketini gönderen cihazın MAC adresini içerir.
- **Hedef IP Adresi:** ARP paketinin hedef cihazın IP adresini içerir. Bu alan, ARP isteğinde hangi IP adresi için MAC adresi aranıyorsa o IP'yi gösterir.
- **Hedef MAC Adresi:** Hedef cihazın MAC adresini içerir. ARP isteği yapıldığında bu alan bilinmediği için genellikle 0 olarak doldurulur. ARP yanıtında ise hedefin gerçek MAC adresi burada bulunur.

Tablo 2.1: İlk on donanım türleri.

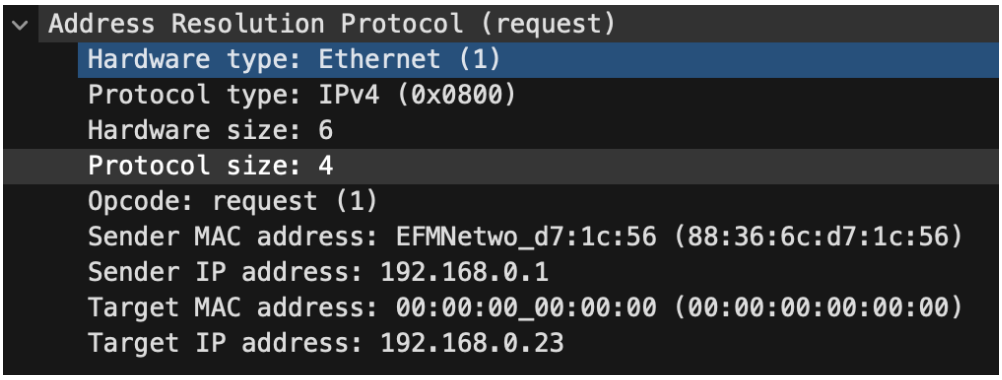
Donanım Türü	Kod	Açıklama
Ethernet (10Mb)	1	Standart Ethernet, ağlarda yaygın olarak kullanılan ve ARP için en yaygın ortam olan Ethernet türüdür.
Deneysel Ethernet (3Mb)	2	Eski ve deneysel bir Ethernet türü olup, küçük yerel ağlar için uygundur
Amatör Radyo AX.25	3	Temel ağ teknolojisi paket radyolardan oluşmaktadır. Ağda hata kontrolü ile birlikte paketlerin sıralı iletimini sağlar.
Proteon ProNET Token Ring	4	ProNET, Proteon, Inc. tarafından geliştirilen bir ağ teknolojisidir. Yerel alan ağları (LAN'lar) için Token Ring yöntemini kullanılmıştır.
Chaos	5	Chaos ağı, 1970'lerde Massachusetts Teknoloji Enstitüsü Yapay Zeka Laboratuvarı için geliştirilmiş ve bilgisayarlar arası iletişim amacıyla kullanılmıştır.
IEEE 802 Ağları	6	Ethernet (802.3) ve Wi-Fi (802.11) içerir; yaygın olarak kullanılır, yüksek ARP sahtekarlığı riski taşır.
Bağlı Kaynak Bilgisayar Ağı (Attached Resource Computer Network - ARCNET)	7	ARCNET, 1976 yılında John Murphy tarafından tasarlanan ve Ethernet'ten önce yaygın olarak kullanılan bir yerel ağ teknolojisiydi.
Hyperchannel	8	Hyperchannel, 1970'lerde kısa mesafelerde ana bilgisayarları bağlamak için tasarlanan yüksek hızlı bir ağ teknolojisiydi.
Lanstar	9	Lanstar, 1980'lerde geliştirilen ve kaynak paylaşımına odaklanan bir LAN teknolojisiydi. Günümüzde yerini daha modern ağ standartlarına bırakmıştır.
Autonet Kısa Adres	10	Autonet, 1990'larda tasarlanan otomatik yapılandırma ve hata kurtarma özelliklerine sahip bir LAN teknolojisiydi.

- **Protokol Türü:** Aşağıda verilen Tablo 2.2, ARP tarafından desteklenen ve her biri benzersiz bir onaltılık kodla tanımlanan ilk on protokol türünü listelemektedir. ARP bağlamında en yaygın kullanılan protokol, mevcut internet iletişiminin belkemiğini oluşturan IPv4'tür (0x0800). ARP, IPv4 ağlarında IP adreslerini MAC adreslerine eşlemek için gereklidir ve IPv4 girdileri, tipik ARP önbelleğinde bulunur. ARP protokolü (0x0806) de ARP önbelleğinde bulunur, çünkü cihazlar ilk kez iletişim kurduğunda veya ARP önbelleği girdisi süresi dolduğunda adres çözümlemesi yapmak için ARP paketlerini kullanır [5], [31], [32].
- **Protokol Adres Uzunluğu:** Protokol adresinin uzunluğu. IPv4 için bu değer 4'tür.

Tablo 2.2: İlk on protokol türü.

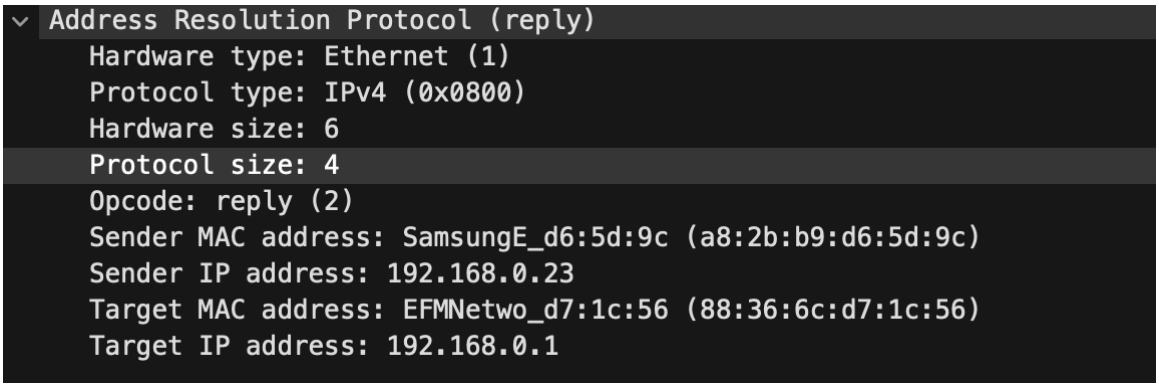
Protokol Türü	Kod (On Altılık)	Açıklama
IPv4	0x0800	IPv 4; veri iletişimi için en yaygın kullanılan protokol.
ARP	0x0806	ARP; IP adreslerini MAC adresleriyle çözümlemek için kullanılır.
Wake-on-LAN	0x0842	Düşük güç durumundaki bilgisayarları uzaktan uyandırmak için kullanılan protokol.
AppleTalk (Ethertalk)	0x809B	AppleTalk, Apple bilgisayarlarının LAN üzerinden iletişim kurmasına izin vermek için tasarlanmıştır. EtherTalk, bu protokolün Ethernet üzerinde uygulanmasıyla, Apple cihazlarının Ethernet tabanlı ağlarda iletişim kurmasını sağlar.
AppleTalk ARP	0x80F3	AppleTalk ARP, AppleTalk ağlarındaki cihazların donanım adreslerini çözümleyerek iletişim kurmalarını sağlayan bir protokoldür.
Virtual LAN etiketli çerçeve (IEEE 802.1Q)	0x8100	IEEE 802.1Q, Ethernet çerçevelerine VLAN bilgisi ekleyerek farklı VLAN'ların aynı fiziksel ağda taşınmasını sağlar. Bu, trafiği ayırma, önceliklendirme ve güvenli hale getirme imkânı sunar.
Novell Ağlararası Paket Değişimi (Internetwork Packet Exchange-IPX)	0x8137	IPX, 1980'ler ve 1990'larda Novell NetWare ağlarında hızlı paket yönlendirmesi için kullanılan bir ağ katmanı protokolüdür.
IPv6	0x86DD	IPv6; IPv4'ün daha geniş adres alanı sunan türüdür.
MPLS (Multiprotocol Label Switching) Tekli Yayın	0x8847	Ethernet çerçevesinde tek bir kaynaktan tek bir hedefe gönderilen tekli yayın trafiğini belirten bir protokoldür.
Noktadan Noktaya Protokol Üzerinden Ethernet Keşif Aşaması (Point-to-Point Protocol over Ethernet - PPPoE Discovery Stage)	0x8863	PPPoE Keşif Aşaması, PPPoE protokolünün cihazın internet sağlayıcısındaki sunucu ile bağlantı kurmasını sağlayan ilk aşamasıdır. Bu aşamada cihaz, sunucuyu keşfeder ve oturum başlatmak için gerekli bilgileri alır. Bu aşama tamamlandıktan sonra veri iletimi için bağlantı kurulur.

Şekil 2.24‘te görülen ARP isteği, 192.168.0.1 IP adresine sahip bir cihazın, 192.168.0.23 IP adresine sahip cihazın MAC adresini öğrenmek amacıyla ağdaki tüm cihazlara yayın (broadcast) olarak gönderdiği bir mesajdır. Ethernet protokolü (donanım türü 1) ve IPv4 protokolü (0x0800) kullanılarak oluşturulan bu ARP isteğinde, gönderici cihazın MAC adresi 88:36:6c:d7:1c:56 ve IP adresi 192.168.0.1 olarak belirtilmiştir. Hedef cihazın MAC adresi henüz bilinmediğinden 00:00:00:00:00:00 olarak gösterilir; bu durum, ARP isteklerinin hedef cihazın MAC adresini öğrenmek için ağdaki tüm cihazlara yayın olarak gönderildiğini belirtir. Bu nedenle, ARP isteğine yanıt verecek olan cihaz, kendisine ait IP adresiyle eşleşen bir cihaz olduğunda, kendi MAC adresini içeren bir ARP yanıtı ile geri dönecektir.



Şekil 2.24: Örnek ARP isteği.

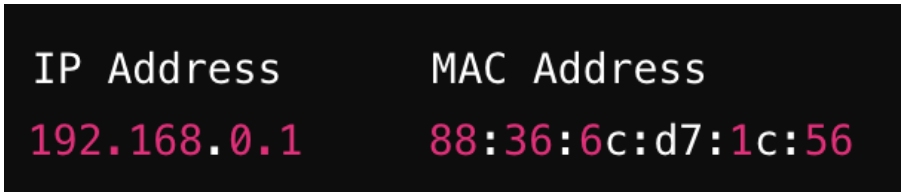
Şekil 2.25'de görülen ARP yanıtı, önceki ARP isteğine yanıt olarak, 192.168.0.23 IP adresine sahip cihaz tarafından gönderilmiştir. Yanıt, gönderici cihazın MAC adresini (a8:2b:b9:d6:5d:9c) ve IP adresini (192.168.0.23) içerir, hedef olarak ise isteği gönderen cihazın MAC adresi (88:36:6c:d7:1c:56) ve IP adresi (192.168.0.1) belirtilmiştir. Bu yanıt ile 192.168.0.23 IP adresine sahip cihaz, 192.168.0.1 IP adresine sahip cihazın ARP isteğini doğrular ve kendi MAC adresini bildirir. Bu sayede, ağdaki iletişim düzgün bir şekilde sağlanabilir, çünkü 192.168.0.1 IP adresine sahip cihaz, artık 192.168.0.23 IP adresine sahip cihazın MAC adresini öğrenmiş ve ARP tablosunu güncellemiştir. Bu işlem, aynı ağ içindeki cihazların IP adreslerini MAC adresleriyle eşleştirmesine ve veri iletimini doğru adrese yönlendirmesine olanak tanır.



Şekil 2.25: Örnek ARP yanıtı.

ARP tablosu değişiklikleri için bir örnek olarak Şekil 2.26'te, ilk ARP isteği gönderilmeden önce 192.168.0.1 makinesinin ARP tablosu gösterilmektedir. Bu tabloda,

henüz hedef makinenin (192.168.0.23) MAC adresi bilinmediği için ilgili kayıt bulunmamaktadır.



Şekil 2.26: İstek öncesi ARP tablosu (192.168.0.1).

İlk istek gönderildikten sonra, 192.168.0.23 makinesi ARP yanıtını verdiğinde, 192.168.0.1 makinesi kendi ARP tablosunu Şekil 2.27'da görüldüğü gibi günceller. Bu güncelleme ile hedef IP adresi (192.168.0.23) ile bu IP'ye karşılık gelen MAC adresi (a8:2b:b9:d6:5d:9c) tabloya eklenir.

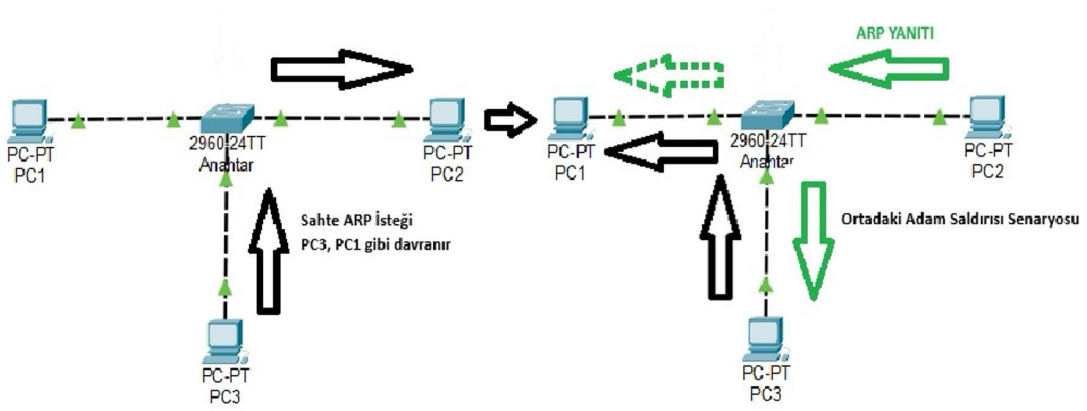
Şekil 2.27: Cevap sonrası ARP tablosu (192.168.0.1).

192.168.0.1 makinesinden gelen ARP isteğini alan 192.168.0.23 makinesi, bu isteği aldıktan hemen sonra ARP tablosunu günceller. Bu güncelleme sonucunda, 192.168.0.23 makinesinin ARP tablosunda 192.168.0.1 IP adresi ve buna karşılık gelen MAC adresi (88:36:6c:d7:1c:56) kaydedilir. Güncellenmiş tablo, Şekil 2.28'de gösterilmektedir.

Şekil 2.28: İstek sonrası ARP tablosu (192.168.0.23).

ARP protokolü, ağdaki cihazların IP adreslerini MAC adresleriyle eşleştirmesi için kullanılan basit bir protokoldür ve ARP istek/yanıt paketleri arasında herhangi bir kimlik doğrulama mekanizması bulunmamaktadır. Yukarıda verilen örneklerde görüldüğü üzere, 192.168.0.1 IP adresine sahip bir cihaz, 192.168.0.23 IP adresine sahip cihazın MAC adresini öğrenmek için bir ARP isteği gönderir. 192.168.0.23 IP adresine sahip cihaz bu isteğe bir ARP yanıtı ile karşılık verir, ancak bu yanıtın doğruluğunu kontrol eden bir mekanizma yoktur. Bu durum, ARP protokolünü kötüye kullanmak isteyen saldırganlar için bir zafiyet oluşturur; saldırgan, sahte ARP yanıt paketleri göndererek hedef cihazların ARP tablolarını manipüle edebilir ve böylece ARP sahtecilik (spoofing) saldırıları gerçekleştirebilir. ARP sahtecilik saldırıları, saldırganın ağdaki veri trafiğini izlemesine (dinleme), meşru cihazlar arasındaki trafiği kesmesine veya yönlendirmesine ortadaki adam ve ağ hizmetlerini kesintiye uğratarak hizmet reddi saldırılarına neden olmasına olanak tanır. Bu sebeple, ARP sahtecilik saldırıları ağ güvenliği açısından ciddi bir tehdit oluşturur ve güvenlik önlemleri alınmadan bu tür saldırıların önüne geçilemez [33].

MITM saldırıları, ARP sahtecilik yöntemini kullanarak saldırganın iki cihaz arasındaki iletişimi ele geçirmesine ve yönlendirmesine dayanan bir saldırı tipidir. Bu saldırıda, saldırgan hem kaynak hem de hedef cihazlara sahte ARP istekleri/yanıtları göndererek her iki cihazın da iletişim kurarken kendisinin MAC adresini kullanmasını sağlar. Böylece, ağ üzerindeki veri paketleri, gönderen ve alıcı arasında iletilmeden önce saldırgan üzerinden geçer [6]. Bu durum, saldırgana veri trafiğini dinleme (sniffing), değiştirme ve hatta tamamen durdurma imkânı tanır. MITM saldırıları, özellikle güvenli olmayan ağlar üzerinde büyük bir tehdit oluşturarak gizlilik, bütünlük ve erişilebilirlik açısından ciddi güvenlik açıkları yaratır. Giriş bilgileri, Kişisel Kimlik Numaraları (Personal Identification Numbers - PIN'ler) ve finansal veriler gibi hassas bilgiler tehlikeye girebilir. Bazı ağ protokollerinde şifrelemenin olmaması, saldırganların iletişim akışlarını okumasını, değiştirmesini veya kötü amaçlı içerik enjekte etmesini kolaylaştırarak bu zafiyeti daha da tehlikeli hale getirir. Bu nedenle, ağlarda ARP protokolüne yönelik zafiyetlerin giderilmesi ve güvenlik önlemlerinin artırılması kritik önem taşır.

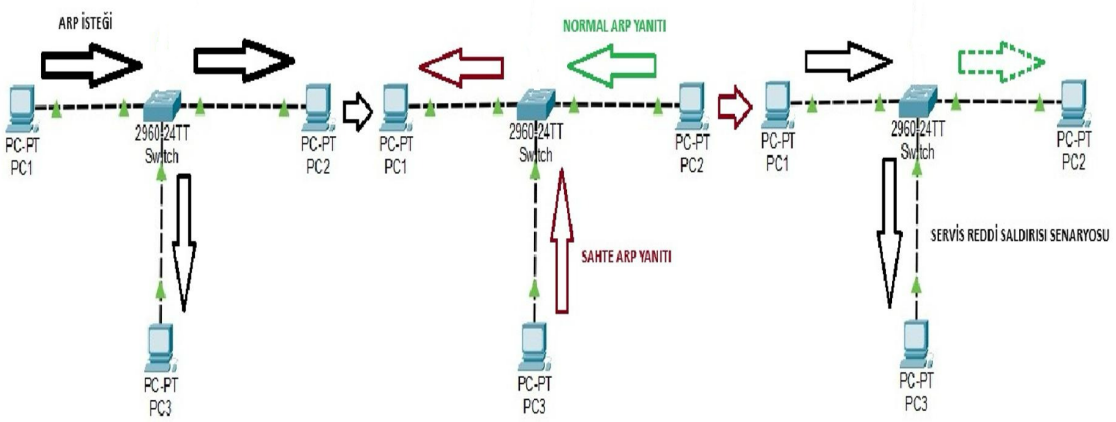


Şekil 2.29: Arp sahteciliği saldırı senaryosu.

Şekil 2.29'de bir ortadaki adam saldırısının dinamiklerini sergileyen tipik bir ARP aldatmaca senaryosu gösterilmektedir. Bu senaryoda, kötü niyetli bir sanal makine (VM – Virtual Machine) olan PC3, aynı ağ içindeki başka bir ana bilgisayar olan PC2'yi kandırmak için meşru bir VM olan PC1 olarak kendini gösterir. İlk olarak, PC3, PC1 olduğunu iddia eden sahte bir ARP isteği oluşturur. Bu ARP isteği, ağda yayınlanan sahte bir IP-MAC adres çifti içerir. PC2, bu isteği aldıktan sonra ARP önbelleğini yanlış eşlemeyle güncelleyerek PC1'in IP adresini PC3'ün MAC adresiyle ilişkilendirir. Sonuç olarak, PC1'e yönelik herhangi bir veri PC3'e yönlendirilebilir.

Saldırının sonraki aşamasında, PC3 kendini etkili bir şekilde PC1 ve PC2 arasında konumlandırarak iletişim akışını keser. Bu kesinti, PC3'ün veri akışını izleyebilmesine, değiştirebilmesine veya bozabilmesine olanak tanır, böylece başarılı bir MITM saldırısı gerçekleştirilir. Şekil 2.29, PC3'ün ARP aldatmacasını kullanarak PC1 ve PC2 arasındaki iletişim kanalını nasıl ele geçirdiğini gösteren iki yönlü veri akışını görselleştirmektedir. Bu şekil, ağ güvenliğinde ARP protokollerine ilişkin kritik zafiyetleri vurgulamakta ve ARP aldatmacasının karmaşık MITM saldırılarını ne kadar kolaylaştırabileceğini göstermektedir. Kötü niyetli bir varlığın meşru bir ana bilgisayarı taklit edebilme ve ARP tablolarını manipüle edebilme yeteneği, bu tür açıkları tespit etmek ve önlemek için güçlü ağ güvenlik önlemlerinin gerekliliğini ortaya koymaktadır.

Diğer bir saldırı türü olan DoS, ağ güvenliği açısından ciddi bir tehdit oluşturarak, meşru kullanıcıların ağ hizmetlerine erişimini kesintiye uğratmayı hedefler. DoS saldırılarının Yerel Alan Ağı (LAN) üzerinde gerçekleştirilme yöntemlerinden biri, ARP sahteciliğini kullanmaktır. ARP sahteciliği kullanılarak yapılan bir DoS saldırısında, saldırgan, yanlış IP-MAC adres eşlemelerini içeren ARP paketleriyle ağı doldurur. Saldırgan hem kaynak hem de hedef cihazlara sahte ARP istekleri/yanıtları göndererek, bu cihazların saldırganın MAC adresini kendi meşru IP adresleriyle ilişkilendirmelerini sağlar. Bu durum, kurban için hedeflenen tüm ağ trafiğinin saldırgana yönlendirilmesiyle sonuçlanır. MITM saldırılarının aksine, burada saldırgan trafiği kesip tekrar iletmek yerine, genellikle paketleri düşürerek cihazlar arasındaki iletişimi tamamen durdurur. ARP sahteciliği kullanılarak yapılan DoS saldırılarının etkisi ciddi olabilir. Bu tür saldırılar, ağ hizmetlerini tamamen aksatarak uzun süreli kesintiye ve verimlilik kaybına yol açabilir. Finansal kurumlar, sağlık sistemleri ve devlet ağları gibi sürekli iletişim ve veri alışverişinin kritik olduğu ortamlarda, bu tür saldırılar son derece zararlı olabilir. Ayrıca, ARP protokolündeki temel zayıflıkların kolayca istismar edilmesi ve bu saldırıların düşük düzeyde bir uzmanlıkla gerçekleştirilebilmesi, ARP sahteciliği kullanan DoS saldırılarının tespit edilmesini zorlaştırır. ARP sahteciliği ile yapılan DoS saldırılarına karşı korunmak için ağ yöneticilerinin güçlü güvenlik önlemleri alması gerekmektedir. Sonuç olarak, ARP sahteciliği kullanılarak yapılan DoS saldırıları, ARP protokolündeki doğuştan gelen zayıflıklar nedeniyle ağ güvenliği açısından önemli bir zafiyet oluşturmaktadır. Bu tür saldırıları önlemek, proaktif ağ yönetimi, güvenlik protokollerinin uygulanması ve potansiyel tehditleri gerçek zamanlı olarak algılamak ve yanıt vermek için sürekli izlemeyi gerektirir [11].



Şekil 2.30: İkinci ARP saldırı senaryosu.

Şekil 2.30, bir DoS sonucuna yol açan önemli bir ARP sahteciliği saldırısını vurgulamaktadır. Bu saldırı, PC1'in PC2'nin MAC adresini çözmek için yayınladığı bir ARP isteğiyle başlar. Kötü niyetli sanal makine PC3, PC2'nin IP adresini kendi MAC adresiyle ilişkilendirerek PC1'e sahte bir ARP yanıtı ile cevap verir. Sonuç olarak, PC1'in PC2 için gönderdiği trafik PC3 tarafından ele geçirilir ve bu da hizmetin kesintiye uğramasına neden olur.

Saldırganın ARP sahteciliği yapabilme yeteneği ve DoS saldırısını tetikleme yeteneği, ARP'nin sahip olduğu ciddi güvenlik tehditlerini göstermektedir. Bu durum, ağ güvenliği uygulamalarının bu tür saldırılara karşı korumasını, ağ hizmetlerinin bütünlüğünü ve kullanılabilirliğini sağlaması gerekliliğini gösterir. ARP sahteciliği saldırıları, ciddi ölçüde bilgi hırsızlığı ve manipülasyon potansiyeli taşıdığından, bu tehditlere karşı önleyici tedbirler üzerine yoğun araştırmalar yapılmaktadır. Bununla birlikte, ARP sahteciliği saldırısının gerçekleştiğini belirlemek için kullanılan temel yaklaşım şu şekildedir: Ağ trafiği izlenirken, ARP tablosunda aynı MAC adresiyle ilişkilendirilmiş birden fazla IP adresinin görülmesi, ARP saldırısı gerçekleştirildiğinin açık bir belirtisidir.

Şekil 2.29 ve Şekil 2.30'de sunulan görsellerden, ARP sahteciliği algılama araçları, statik ARP girişleri ve güvenli iletişim protokolleri gibi güvenlik önlemlerinin uygulanmasının gerekli olduğu görülebilir [34]. Bu önlemler, bu tür saldırıların riskini azaltmaya yöneliktir ve ağ hizmetlerinin bütünlüğünü ve kullanılabilirliğini sağlar. ARP saldırılarını önlemek için çeşitli donanımsal ve yazılımsal çözümler geliştirilmektedir ve bu alandaki araştırmalar devam etmektedir.

2002 yılında Gouda ve Huang'un gerçekleştirdiği çalışmada [10], ARP sahtecilik saldırılarını önlemek için Güvenli Adres Çözümleme Protokolü (S-ARP) geliştirilmiştir. Bu protokol, geçerli IP-MAC adres eşlemesini koruyarak tekrarlanan saldırıları önlemek ve veri bütünlüğünü sağlamak için ortak anahtar ve rastgele sayılar gibi kriptografik teknikler kullanan güvenli bir sunucu içerir. S-ARP, yalnızca doğrulanmış cihazların bir ağda iletişim kurmasına izin vererek ARP sahteciliği saldırılarını azaltır. Ancak, bu protokolün büyük ve dinamik ağlar için ölçeklenebilirliği sınırlıdır [10].

Sharma ve ark. [9] ağ yöneticilerinin IP-MAC eşleşmelerini izleyerek ARP sahteciliği saldırılarını tespit etmelerini sağlayan komut satırı tabanlı bir yöntem sunmuştur. Önerilen sistem, ARP paketlerini tutarlılık açısından kontrol eder ve bunları önceden tanımlanmış bir geçerli eşleşme veri tabanıyla karşılaştırır. Herhangi bir tutarsızlık tespit edilirse, sistem bir uyarı tetikler ve ağ yöneticilerinin hemen harekete geçmesini sağlar. Bu yöntem, küçük ve orta ölçekli ağlar için uygundur ancak daha büyük ağlarda sınırlamaları vardır.

Gao ve Xia [11] çalışmasında, yerel alan ağlarında ARP sahteciliğini tespit etmek için ICMP tabanlı bir algoritma sunularak, genellikle ölçeklenebilirlik ve performans sınırlamalarıyla karşılaşan statik ARP yapılandırmaları ve kriptografik yaklaşımlar gibi geleneksel yöntemlerin geliştirilmesi amaçlanmıştır. Önerilen yöntemleri, gelen ARP paketlerine dayalı ICMP yankı istekleri oluşturmayı ve sahte IP-MAC çiftlerini tanımlamak için yanıtları analiz etmeyi içerir. Önceki tekniklerden farklı olarak, bu yaklaşım ağ trafik yoğunluğunu etkilemez ve şüpheli ana bilgisayarları seçici olarak araştırarak gecikmeyi en aza indirir. Çalışmada, ICMP algoritması ARP sahtekarlığı tespiti için özellikle dinamik ağ ortamları için uygun, verimli ve ölçeklenebilir bir çözüm olarak sunulmaktadır.

Kumar ve ark. [35], SDN ortamlarında ARP sahteciliği saldırılarını tespit etmek için derin öğrenme tabanlı bir model geliştirmiştir. Konvolüsyonel Sinir Ağları (Convolutional Neural Networks - CNN) kullanan bu model, kötü niyetli trafiği %94,13 doğrulukla tespit edebilmekte ve gerçek zamanlı koruma sağlamaktadır.

Shah ve ark. [36] SDN'de bulunan Dinamik ARP Denetimi (Dynamic ARP Inspection - DAI) ortamlarında ARP önbellek zehirlenmesi saldırılarının tespiti ve azaltılmasına yönelik çeşitli yöntemleri ele almaktadır. ARP sahteciliğinin ciddi bir güvenlik tehdidi oluşturduğu SDN için mevcut yaklaşımları üç ana yöntem altında sınıflandırmışlardır: Trafik modeli tabanlı çözümler, IP-MAC eşlemesi doğrulaması ve akış grafiği tabanlı yaklaşımlar. Her bir yöntem, SDN'nin merkezi kontrol avantajından yararlanarak, sahtekarlık girişimlerini gerçek zamanlı olarak izlemek, tespit etmek ve önlemek üzere uyarlanmıştır.

Kanimozhi ve ark. [37], Nesnelerin İnterneti (Internet of Things - IoT) ağlarında ARP sahteciliği tespitini geliştirmek için derin öğrenme modellerinin kullanımını incelemiştir. CNN ve Üretici Çekişmeli Ağlar (Generative Adversarial Networks - GAN) modelleri, IoT-23 veri kümesi kullanılarak yapılan testlerde yüksek doğruluk göstermiştir; ancak, gerçek

zamanlı saldırı tespiti için işlem süresi ve hesaplama maliyetleriyle ilgili zorlukların olabileceğini belirtilmiştir.

Mvah ve ark. [38] çalışmasında, SDN'de ARP sahtekarlığına karşı oyun teorisine dayalı yenilikçi bir savunma yaklaşımı sunmaktadır. Geleneksel ve makine öğrenimi tabanlı yöntemler, büyük ölçekli ağlarda yüksek gecikme ve kaynak kullanımı sorunları yaşarken, yazarlar saldırgan ve savunucu arasındaki etkileşimi bir oyun olarak modellemiş ve Nash dengesi stratejileriyle ARP doğrulama sürecini optimize etmişlerdir. Simülasyon sonuçları, bu yöntemin daha düşük gecikme ve bellek kullanımı sağladığını göstererek, büyük ağlar için ölçeklenebilir ve etkili bir güvenlik çözümü sunduğunu ortaya koymaktadır.

Alani ve ark. [7] çalışmasında, ARP sahteciliğinin tespitine yönelik geliştirdikleri çalışmada, IoT ağlarındaki güvenlik tehditlerini incelemiş ve DNN (Derin Sinir Ağı) kullanarak yüksek performanslı bir tespit sistemi önermişlerdir. Çalışmada kullanılan Nesnelerin İnterneti Ağ Saldırı Veri Seti [Internet of Things Network Intrution Dataset – (IoT-ID)], ARP sahteciliği ve iyi huylu trafik örnekleri içermekte olup, veri dengesizliğini gidermek amacıyla random oversampling yöntemi uygulanmıştır. Eğitim ve test işlemleri sonucunda, %99.9 doğruluk, %99.9 keskinlik (precision), %99.9 duyarlılık (recall) ve %99.9 F1 skoru gibi oldukça yüksek başarı oranları elde edilmiştir.

ARP sahteciliği, sızma saldırılarından biri olduğundan, ağ güvenliğini değerlendirmek amacıyla sızma testleri yapılmaktadır. Bu testler, bir sistemin savunma mekanizmalarının ne kadar etkili olduğunu anlamak için saldırgan bakış açısıyla gerçekleştirilir. Ağ trafiği detaylı bir şekilde analiz edilerek, mevcut güvenlik önlemlerinin etkinliği incelenir. Sızma testleri sonucunda, sistemdeki zayıf noktalar tespit edilerek olası saldırı girişimlerine karşı önlemler alınır. Bu testlerde, XArp ve ARPWatch [39], [40], gibi araçlar kullanılarak ARP sahteciliği tespitleri yapılır ve ağın veya güvenlik sistemlerinin bu tür saldırılara karşı tepkisi değerlendirilir. ARP sahteciliğini tespit etmek amacıyla bazı ağlarda Wireshark gibi yazılım araçları kullanılmaktadır. Bu araçlar, kötü niyetli IP-MAC eşleşmelerini belirlemeye yönelik işlevler sunsa da bu eşleşmelerin tespiti yöneticinin inisiyatifine bağlıdır ve bu durum yüksek oranda yanlış pozitif sonuçlara neden olabilir [41].

3. MATERYAL VE YÖNTEM

Bu bölümde, çalışmamızda kullanılan veri seti, uygulanan veri ön işleme adımları, derin sinir ağları yapısı, kullanılan diğer algoritmalar ve değerlendirme metrikleri detaylı olarak açıklanmıştır. Modelin eğitimi, doğrulama stratejileri ve doğruluk metriklerinin hesaplanma yöntemleri de sunulmuştur.

3.1 Veri Seti ve Ön İşleme

Bu bölümde materyal olarak kullanılan ver seti ve ön işleme adımlarından bahsedilmiştir.

3.1.1 Veri Seti

Bu çalışmada IoT-ID [8] olarak adlandırılan ve nesnelerin interneti (IoT) ortamında oluşturulan çeşitli ağ saldırılarına ait paketlerden oluşan bir veri kümesi kullanılmıştır. Veri seti, akademik amaçlar doğrultusunda simüle edilmiş saldırılar içerir. Kullanılan IoT cihazları, SKT NUGU (NU 100) ve EZVIZ Wi-Fi Kamera (C2C Mini O Plus 1080P) olmak üzere iki tipik akıllı ev cihazlarıdır. Bu cihazlar, dizüstü bilgisayarlar ve akıllı telefonlar gibi diğer cihazlarla birlikte aynı kablosuz ağa bağlanmaktadır. Veri seti ise farklı zaman dilimlerinde yakalanmış 42 ham ağ paketi (pcap) dosyasından oluşmaktadır. Paketler, kablosuz ağ adaptörünün monitor mode modunda çalıştırılmasıyla yakalanmıştır. Saldırı türlerinin çoğu, Nmap gibi araçlar vasıtasıyla simüle edilen saldırıların paketlerinden oluşmaktadır. Ancak Mirai Botnet saldırı kategorisinde, saldırı paketleri bir dizüstü bilgisayarda üretilmiş ve ardından bu paketler, IoT cihazından geliyormuş gibi manipüle edilmiştir.

Çalışmada kullanılan veri seti, normal trafik, ARP sahteciliği ve çeşitli siber saldırılara ait ağ paketlerinden oluşmaktadır. Tablo 3.1'de, her kategori ve alt kategoriye ait toplam paket sayıları sunulmuştur[8]. Bu çalışmada, ARP sahteciliği saldırılarına odaklanılmış olup, veri setinde MITM – ARP sahteciliği kategorisinde yer alan dosyalar kullanılmıştır. Tablo 3.2'de, çalışmada kullanılan pcap dosyalarının adları, boyutları, içerdiği toplam paket sayısı ve saldırı paket sayıları detaylı olarak gösterilmektedir[8].

Tablo 3.1: Veri seti paket bilgileri.

Kategori	Alt Kategori	Paket Sayısı
Normal	Normal	1,756,276
Scanning	Host Discovery	2,454
Scanning	Port Scanning	20,939
Scanning	OS/Version Detection	1,817
MITM	ARP Spoofing	101,885
DoS	SYN Flooding	64,646
Mirai Botnet	Host Discovery	673
Mirai Botnet	Telnet Bruteforce	1,924
Mirai Botnet	UDP Flooding	949,284
Mirai Botnet	ACK Flooding	75,632
Mirai Botnet	HTTP Flooding	10,464

Tablo 3.2: Veri setinde kullanılan dosya bilgileri.

No.	Dosya Adı	Dosya Boyutu (KB)	Kategori	Alt Kategori	Toplam Paket Sayısı	Saldırı Paketi Sayısı
1	benign-dec.pcap	117,937	Normal	Normal	137,396	-
2	mitm-arpspoofing-1-dec.pcap	27,045	MITM	ARP Sahteciliği	65,768	34,855
3	mitm-arpspoofing-2-dec.pcap	12,152	MITM	ARP Sahteciliği	33,121	13,134
4	mitm-arpspoofing-3-dec.pcap	12,812	MITM	ARP Sahteciliği	34,043	15,144
5	mitm-arpspoofing-4-dec.pcap	18,813	MITM	ARP Sahteciliği	19,914	13,211
6	mitm-arpspoofing-5-dec.pcap	18,284	MITM	ARP Sahteciliği	20,314	9,743
7	mitm-arpspoofing-6-dec.pcap	19,416	MITM	ARP Sahteciliği	21,024	15,798
Toplam					**331,580**	**101,885**

Tablo 3.2'de sunulduğu üzere, çalışmamızda ARP sahteciliği tespiti üzerine odaklanıldığından, IoT-ID veri setinden normal trafik ve ARP sahteciliği saldırısı içeren dosyalar kullanılmıştır. Çalışmada kullanılan ham veri, toplamda 331.580 paketten oluşmakta olup, bu paketlerin 101.885'i saldırı, 229.695'i ise normal trafik olarak sınıflandırılmıştır. Bu sayılar, IoT-ID veri seti sahipleri tarafından sağlanan filtreler kullanılarak belirlenmiş ve ağ trafiğini analiz etmek için yaygın olarak kullanılan bir araç olan Wireshark uygulaması aracılığıyla doğrulanmıştır.

3.1.2 Öznitelik Seçimi

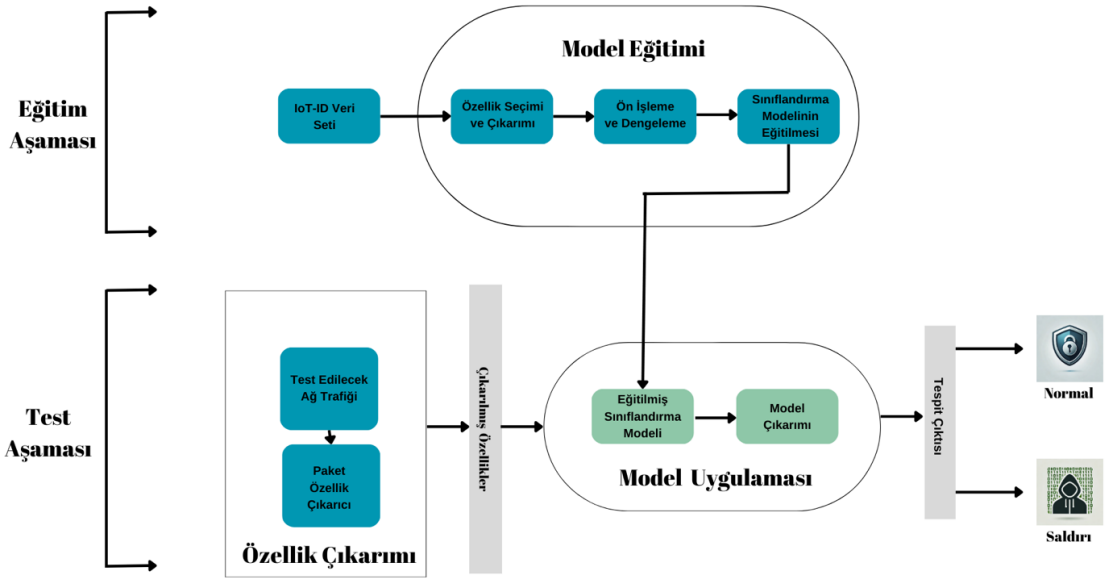
Makine öğreniminde öznitelik seçimi, bir modelin doğruluğunu ve performansını artırmak amacıyla verilen bir veri kümesinden en iyi özniteliklerin seçilmesi sürecidir. Öznitelik seçimi, yalnızca bilgilendirici değişkenleri seçerek verinin boyutunu azaltır, aşırı öğrenmeyi (overfitting) azaltır ve modelin yorumlanmasını iyileştirir. Özellikle amaç için gereksiz ve yüksek boyutlu öznitelikler modelde fazla yük oluşumuna neden olur. Hangi yaklaşım dikkate alınırsa alınsın, etkin özellik seçiminin genel sonucu, bir modelin genelleme yeteneğinin ve verimliliğini artırmasıdır.

3.1.3 Veri Ön İşleme

Makine öğrenimi ve veri bilimi süreçlerinde en önemli adımlardan biri de veri ön işleme aşamasıdır. Bu adım, ham veriyi temizlemeyi, biçimlendirmeyi ve dönüştürmeyi amaçlayarak modelin performansını doğrudan etkilemeyi hedefler. Veri ön işleme süreci; eksik verilerin doldurulması, aykırı değerlerin tespit edilip yönetilmesi ve verilerin normalleştirilmesi veya standardize edilmesi gibi adımları içerir. Bu sayede, ham veri makine öğrenimi algoritmalarıyla daha uyumlu hale getirilir. Bu işlem, modelin doğruluğunu, kararlılığını ve genel performansını artırmayı amaçlar. Verilerin doğru işlenmesi kritik bir rol oynar, çünkü yanlış işlenmiş veriler modelin performansını olumsuz etkileyebilir.

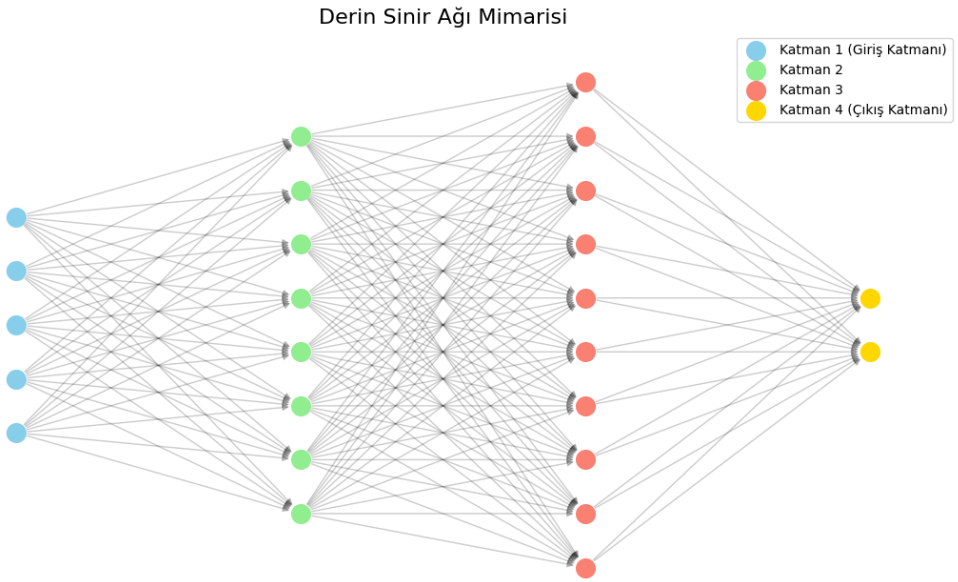
3.2 Makine Öğrenmesi Modelleri

Bu çalışmada, Alani ve ark. [7] çalışması ile karşılaştırıldığında, farklı makine öğrenmesi algoritmaları kullanılarak ARP sahteciliği tespiti gerçekleştirilmiştir. Özel olarak, [7] çalışmasında kullanılan (ve Bölüm 4.2.1'de sunulan) derin sinir ağı modeline ek olarak, çalışmamızda (Bölüm 4.3'te sunulan) çeşitli klasik makine öğrenmesi modelleri de kullanılmış ve karşılaştırmalı analizleri yapılmıştır. Her bir model, veri kümesinin saldırı ve normal trafik örneklerini sınıflandırma başarısını değerlendirmek amacıyla kullanılmıştır. Saldırı tespit sistemi uygulamasının genel akış diyagramı Şekil 3.1'te sunulmuştur.



Şekil 3.1: Saldırı tespit sistemi akış diyagramı.

3.2.1 Derin Sinir Ağları (Deep Neural Networks - DNN)



Şekil 3.2: DNN Mimarisi.

Derin sinir ağları, Şekil 3.2'de görüldüğü gibi giriş katmanı, iç katmanlar ve çıkış katmanından oluşan ve her katmandaki nöronların birbiri ile bağlantısının olduğu bir yapıya sahiptir. DNN, görüntü tanımadan doğal dil işleme kadar birçok uygulamada modern makine öğreniminin temel taşlarından biri haline gelmiştir. Geleneksel makine öğrenimi tekniklerinden farklı olarak, hiyerarşik bir yapıda birden fazla katmanlı doğrusal olmayan

dönüşümler kullanarak ham veriden otomatik olarak özellikler öğrenebilme yeteneğine sahiptir. Bu karmaşık ilişkileri modelleyebilme yeteneği, derin sinir ağlarının nesne tespiti, konuşma tanıma ve ilaç keşfi gibi görevlerde diğer makine öğrenimi yöntemlerini geride bırakmasını sağlamıştır. Derin sinir ağlarının başarısı, öncelikle büyük miktardaki parametrelerin verimli bir şekilde ayarlanmasını mümkün kılan geri yayılım (backpropagation) algoritmasının kullanımına dayanır. Geri yayılım, derin sinir ağlarının öğrenmesini sağlayan temel yöntemdir. Bu algoritma, ağın çıkışında oluşan hatayı hesaplayarak, bu hatayı azaltmak amacıyla ağın ağırlıklarını günceller. Bu süreç, ileri besleme yoluyla sinir ağından geçen girdinin, her katmanda işlenip bir sonuç üretmesinden sonra başlar. Sonuç, gerçek değerle karşılaştırılır ve hatalar hesaplanır. Ardından bu hatalar, geri yayılım adı verilen işlemle ağın her katmanına geriye doğru aktarılır ve her katmandaki ağırlıklar bu hatalara göre optimize edilerek ağ öğrenme sürecinde iyileştirilir [42].

DNN, [7] çalışmasında önerilen temel sınıflandırma algoritmasıdır. Modelin mimarisi ve eğitim süreci (bu çalışmada ele alınan tüm veri seti IoT-ID [36], farklı epoch sayıları ve geleneksel makine öğrenme yöntemleri ile) Bölüm 4 uygulama kısmında detaylandırılmıştır. DNN modeli, çok katmanlı yapısıyla karmaşık örüntüleri öğrenmekte etkili olup, ARP sahteciliği tespitinde güçlü bir sınıflandırma aracı olarak kullanılmıştır.

3.2.1.1 Aktivasyon Fonksiyonları

Aktivasyon fonksiyonları, her bir nöronun aldığı girdilere karşılık ne tür bir çıktı üreteceğini belirleyen matematiksel fonksiyonlardır. Bu fonksiyonlar, doğrusal olmayan ilişkilere sahip veriler üzerinde derin sinir ağlarının öğrenme kapasitesini artırır.

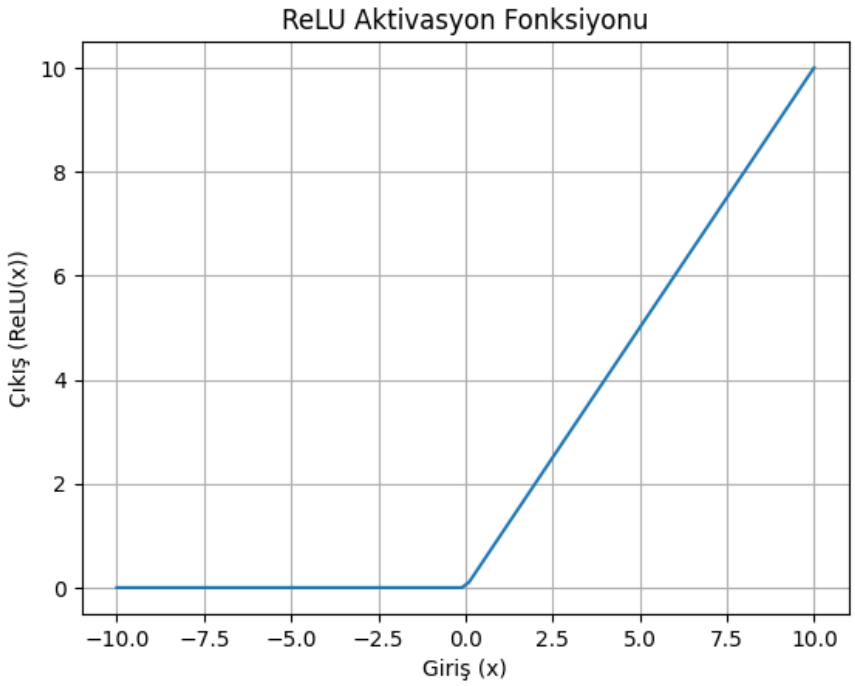
a) ReLU (Rectified Linear Unit - Düzeltilmiş Doğrusal Birim)

ReLU**,** sinir ağlarında yaygın olarak kullanılan ve özellikle derin öğrenme modellerinde performansı artıran bir aktivasyon fonksiyonudur. ReLU, girdi pozitif olduğunda girdi değerini, negatif olduğunda ise sıfırı döndüren (3.1) de sunulduğu gibi basit bir matematiksel işleyişe sahiptir. Şekil 3.3'de fonksiyonun grafiksel çıktısı görülmektedir.

$$\boldsymbol{f(x) = max(0, x)} \tag{3.1}$$

ReLU, büyük giriş değerlerinde doygunluğa ulaşmadığı için derin ağlarda kaybolan gradyan (vanishing gradient) sorununu büyük ölçüde engeller. Kaybolan gradyan sorunu, özellikle

sigmoid veya tanh gibi fonksiyonlar kullanıldığında, gradyanların çok küçük değerlere inmesiyle ağın öğrenme kapasitesinin azalmasına yol açar. Ancak ReLU, bu sorunu ortadan kaldırarak derin ağlarda gradyanların daha verimli bir şekilde yayılmasına olanak tanır ve bu sayede hesaplamalar daha hızlı gerçekleşir [43].

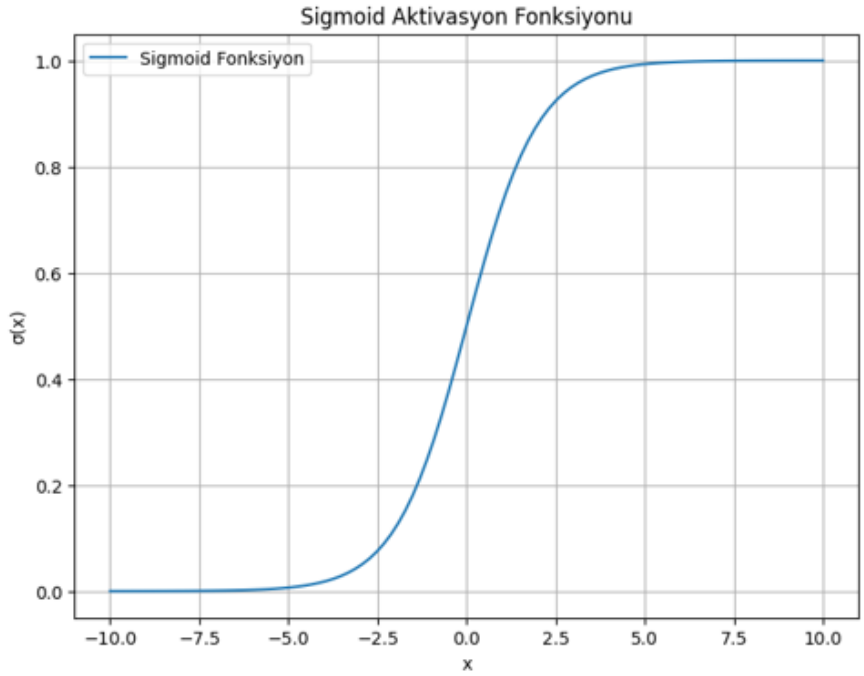


Şekil 3.3: Relu aktivasyon fonksiyonu grafiksel gösterimi.

b) Sigmoid

Sigmoid aktivasyon fonksiyonu, nöronal aktiviteyi modellemede doğrusal olmayan davranışları karakterize etmek amacıyla kullanılan önemli aktivasyon fonksiyonlarından birisidir. Şekil 3.4'te sunulan bu aktivasyon kodu aşağıdaki (3.2) eşitlikle verilmektedir:

$$\boldsymbol{\sigma(x) = \frac{1}{1 + e^{-x}}} \tag{3.2}$$



Şekil 3.4: Sigmoid aktivasyon fonksiyon grafiksel gösterimi.

Sigmoid fonksiyonu çıktısını 0 ile 1 arasında verir, bu da ikili sınıflandırma gibi görevler için uygundur. Sürekli ve türevlenebilir yapısı, öğrenme sürecinde geri yayılım algoritması için sürekli bir gradyan sağlayarak öğrenmeyi kolaylaştırır. Ancak, sigmoid fonksiyonu çok büyük veya çok küçük giriş değerlerinde kaybolan gradyan problemi yaşar, bu da derin ağlarda etkili öğrenmeyi engelleyebilir. Bu sınırlamalara rağmen, belirli sinir ağı mimarilerinde hâlâ önemini korumaktadır [44]. Çıkış katmanında kullanılan sigmoid fonksiyonu, modelin iki sınıf arasındaki olasılığı hesaplamasına olanak tanır.

3.2.1.2 Optimizasyon Algoritması

Optimizasyon algoritmaları, makine öğrenmesi ve derin öğrenme modellerinde parametrelerin en uygun değerlere ulaşmasını sağlayarak modelin doğruluğunu ve etkinliğini artırır. Parametre güncellemelerinde gradyanların hesaplanmasını içeren bu algoritmalar, özellikle büyük veri setleriyle yapılan eğitimlerde önemli rol oynar. Alani ve ark. tarafından DNN modelinde kullanılan Nadam (Nesterov Hızlandırmalı Uyarlanabilir Momentum Tahmini — Nesterov-accelerated Adaptive Moment Estimation) algoritması, eğitim sürecinde daha hızlı ve etkin yakınsama sağlamayı amaçlayan bir optimizasyon yöntemidir. Nadam, Adam ve Nesterov momentumunun avantajlarını birleştiren bir optimizasyon algoritması türüdür. Bu algoritma, esas olarak Adam algoritması üzerinde

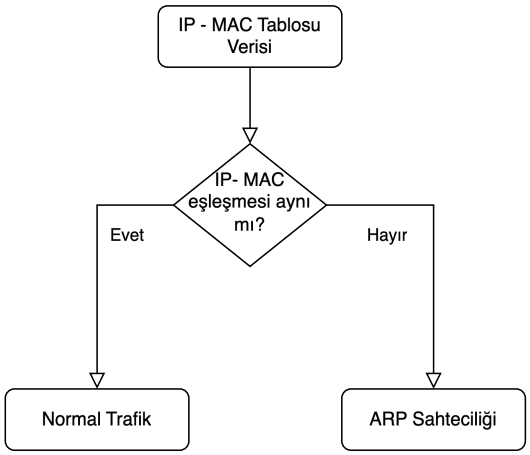
çalışır ve daha iyi yakınsama ve doğruluk elde etmek için Nesterov momentumunu ekleyerek, derin sinir ağlarının eğitim sürecinde daha iyi başarım sağlanır. Adam, gradyanların birinci ve ikinci momentlerine dayalı olarak her bir parametre için öğrenme oranlarını uyarlarken, Nadam prensipte, Nesterov momentumu aracılığıyla gelecekteki gradyanların yönünü tahmin edebilme yeteneğini ekler [45]. Nadam, yakınsama hızını artırma ve model performansını iyileştirme yeteneği sayesinde derin öğrenme görevlerinde kullanılmaktadır.

3.2.1.3 Kayıp Fonksiyonu

Kayıp fonksiyonu, bir modelin tahmin ettiği çıktının gerçek değerden ne kadar saptığını gösterir ve genellikle makine öğrenimi ile derin öğrenme modellerinde başarım ölçümü için kullanılır. Kayıp fonksiyonu, optimizasyon sürecinin temel bir bileşenidir; modelin eğitim sürecinde hataların nasıl en aza indirileceğini belirler. Farklı problemler için, kaybın türüne bağlı olarak farklı kayıp fonksiyonları uygulanır. İkili sınıflandırma problemlerinde en yaygın kullanılan kayıp fonksiyonlarından biri Binary Cross-Entropy'dir (İkili Çapraz Entropi). Bu kayıp fonksiyonu, modelin iki sınıf arasında karar vermesi gereken durumlarda özellikle kullanışlıdır. Gerçek etiketlerle tahmin edilen olasılık dağılımları arasındaki farkı ölçerek, modelin performansını değerlendirir. Lojistik regresyon ve sinir ağları gibi modellerde yaygın olarak kullanılan bu kayıp fonksiyonu, modelin olasılıkları ne kadar iyi tahmin ettiğine göre başarımını ölçer.

3.2.2 Karar Ağacı (Decision Tree)

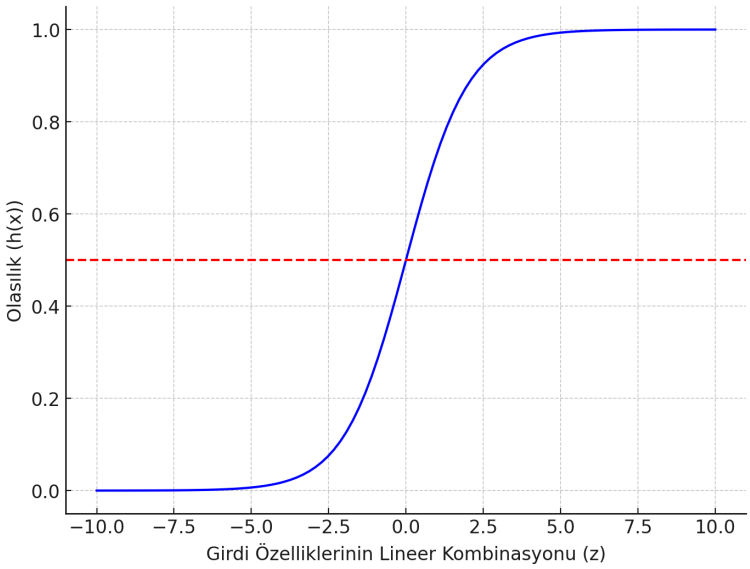
Karar ağacı algoritması hem sınıflandırma hem de regresyon görevlerinde uygulanabilen, yaygın olarak kullanılan bir denetimli öğrenme yöntemidir. Bu algoritma, Şekil 3.5'te basit bir örneği sunulduğu gibi, verileri giriş özelliklerinin değerlerine bağlı olarak tekrar tekrar böler ve bir ağaç benzeri yapı oluşturur. İç düğümler kararları, dallar bu kararların sonuçlarını ve yaprak düğümleri nihai tahmin edilen sınıf veya değeri temsil eder. Karar ağaçları, insan düşünme sürecine benzerliği nedeniyle basitliği ve anlaşılabilirliği ile oldukça popülerdir. Basit olmalarına rağmen, veri içindeki bazı karmaşık ilişkileri yakalayabilirler. Bu sebeple, karar ağaçları tıp teşhisi ve görüntü sınıflandırmasından finansal analizlere kadar geniş bir alanda, sağladıkları yorumlanabilirlik ve etkinlik ile en popüler ve başarılı araçlardan biri haline gelmiştir [46].



Şekil 3.5: ARP sahteciliğine özel bir karar ağacı yapısı.

3.2.3 Lojistik Regresyon (Logistic Regression)

Lojistik regresyon, makine öğreniminde en yaygın kullanılan istatistiksel modellerden biridir ve özellikle ikili sınıflandırma görevlerinde kullanılır. Bu modelin amacı, evet/hayır, doğru/yanlış gibi ikili sınıflandırma problemleri için bir çıktı tahmin etmektir. Doğrusal regresyonun aksine, sürekli bir aralıktaki değerleri tahmin etmek yerine, lojistik regresyon, bir girdinin belirli bir sınıfa ait olma olasılığını tahmin eder. Bu tahmin, bir doğrusal denklemin çıktısına lojistik fonksiyon, yani (eşitlik (3.2) ile verilen) sigmoid fonksiyonu Şekil 3.6'da görüldüğü gibi tahmin edilen olasılıklar 0 ile 1 arasında sınırlı tutulur.



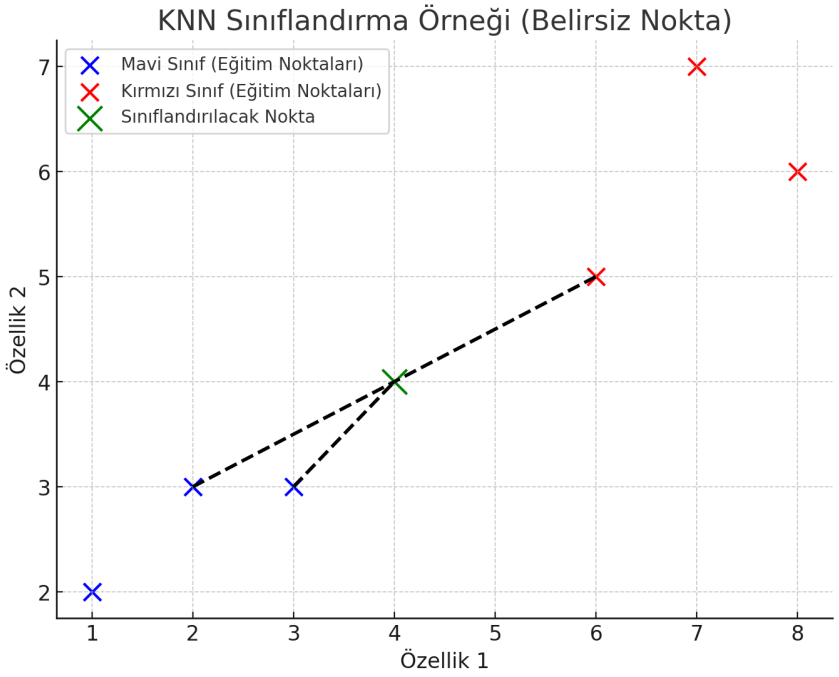
Şekil 3.6: Lojistik (sigmoid) fonksiyonu.

3.2.4 Rastgele Orman (Random Forest)

Rastgele Orman algoritması, sınıflandırma ve regresyon problemleri için kullanılan bir topluluk (*ensemble*) öğrenme yöntemidir. Bu algoritma, verilerin ve özelliklerin rastgele alt kümelerinden birden fazla karar ağacı oluşturarak, makine öğrenimi algoritmalarının kararlılığını ve doğruluğunu artırmak için tasarlanmış ve *bagging* (*bootstrap aggregating*) olarak isimlendirilen bir makine öğrenimi topluluğu meta-algoritması ile çalışır [47]. Her adımda, ağaçların inşa sürecinde özelliklerin rastgele bir alt kümesi üzerinden en iyi bölünmelerin seçilmesi, ağaçlar arasındaki korelasyonu azaltarak modelin sınıflandırma yeteneğini artırır. Bu yapı, aşırı öğrenme problemini önemli ölçüde azaltır ve modelin doğruluğunu artırır [47]. Tahmin aşamasında, regresyon problemlerinde tüm ağaçlardan elde edilen sonuçların ortalaması alınırken, sınıflandırma problemlerinde çoğunluk oyu prensibi uygulanır. Rastgele Orman algoritması, özellikle tıp, finans ve görüntü işleme gibi geniş uygulama alanlarında kullanılmaktadır.

3.2.5 K-En Yakın Komşu Algoritması (K-Nearest Neighbors - KNN)

KNN, sınıflandırma ve regresyon görevleri için basit bir şekilde kullanılabilen makine öğrenme tekniklerinden biridir. Bu algoritma, genellikle Öklid mesafesi gibi seçilen bir mesafe metriği kullanarak yeni bir girdiye en yakın olan *k* veri noktasını tanımlar. KNN, sınıflandırma problemlerinde girdiyi komşuları arasındaki en sık görülen sınıfa atarken, regresyon problemlerinde ise *k* en yakın komşunun değerlerinin ortalamasını alarak tahmin yapar. Şekil 3.7'de bir örneği sunulan KNN probleminde, sınıfı belirsiz olan ve yeşil renkle gösterilen bir X elemanı vardır. En yakın 3 komşusuna bakıldığında 2 adet mavi renkli elemana daha yakın olduğu ve bu mavi komşuların sayısının çoğunlukta olmasından dolayı yeşil X elemanının mavi X elemanlarının sınıfına ait olduğuna karar verilir. KNN'nin temel avantajı, basitliği ve esnekliğidir. Ancak, her yeni tahmin için mesafe hesaplaması yapılması gerektiğinden, büyük veri setlerine uygulandığında bu yöntem hesaplama açısından maliyetli hale gelebilir[48]. Ayrıca, veri setinde gereksiz veya gürültülü özellikler varsa, KNN'nin başarımı düşebilir. Buna rağmen, KNN çoklu sınıf sınıflandırması, doğrusal ilişkiye sahip olmayan ve düşük boyutlu veri setleri gibi durumlarda etkin bir şekilde çalışır. Bu nedenlerle, örüntü tanıma ve veri madenciliği gibi çeşitli alanlarda uygulanabilmektedir.



Şekil 3.7: KNN sınıflandırma örneği.

3.3 Model Doğrulama Stratejisi

Model doğrulama stratejisi, geliştirilen makine öğrenmesi modelinin başarımını ve sınıflandırma yeteneğini güvenilir bir şekilde değerlendirmek için kullanılan yöntem ve tekniklerden oluşur. Model doğrulama, bir modelin sadece eğitim verisi üzerinde değil, farklı veri setlerinde de iyi performans göstermesini sağlamayı amaçlar. Bu bağlamda, ARP sahteciliği tespitine yönelik geliştirilen başta derin sinir ağı (DNN) modeli ve diğer makine öğrenmesi yöntemlerinin başarısını doğru bir şekilde ölçmek için çeşitli doğrulama teknikleri ve metrikleri kullanılmıştır.

3.3.1 Çapraz Doğrulama (Cross-Validation)

Katmanlı çapraz doğrulama (stratified k-fold cross-validation), makine öğrenmesi modellerinin başarımını değerlendirmek için yaygın olarak kullanılan bir yöntemdir. Özellikle katmanlı rastgele örnekleme (stratified random sampling) ile kullanıldığında, katmalardaki sınıflar arasında bulunan dengesizliği gidermeyi amaçlar [49]. Bu yöntemde, veri seti belirlenen k sayısına (genellikle 10) göre eşit büyüklükteki parçalara (katmanlara) bölünür. Her bir katman, doğrulama için seçilirken, geri kalan $k-1$ katman modelin eğitimi için kullanılır. Bu işlemin k kez tekrarlanmasıyla, her bir katman doğrulama seti olarak kullanılır. Katmanlı rastgele örnekleme, yukarda bahsedildiği gibi, her bir doğrulama setinde sınıf oranlarının eğitim setindeki oranlarla aynı olmasını sağlar. Örneğin, eğitim

setinde iki sınıfın dağılımı dengesizse, her bir katman bu dengesizliği yansıtacak şekilde örneklenir. Bu yöntem, modelin eğitim ve doğrulama verileri arasında aşırı uyum göstermesini engelleyerek modelin sınıflandırma yeteneğini artırır. Aynı zamanda, sınırlı veri setlerinde bile daha güvenilir sonuçlar elde edilmesini sağlar. Katmanlı çapraz doğrulama, özellikle dengesiz veri setlerinde modelin her iki sınıftaki performansını dengeli bir şekilde ölçmeye imkân tanıdığı için önemli bir doğrulama stratejisidir.

3.3.2 Doğrulama Metrikleri

Modelin doğruluğunu ve başarımını ölçmek için çeşitli metrikler kullanılır. Bu metrikler, modelin nasıl tahminlerde bulunduğu ve tahminlerinin ne kadar isabetli olduğu hakkında bilgi verir. Aşağıda en yaygın kullanılan doğrulama metrikleri açıklanmıştır [50]:

a) Doğruluk (Accuracy)

Doğruluk, eşitlik (3.3)'te görüldüğü üzere modelin doğru tahmin ettiği örneklerin toplam tahminler içindeki oranını gösterir. Tüm tahminlerin ne kadarının doğru olduğunu ölçen bu metrik, özellikle sınıflar arasında dengeli bir dağılım olduğunda modelin genel başarımını değerlendirmek için kullanılır. Doğruluk, dengesiz veri setlerinde (örneğin, çok az sayıda saldırı örneği olan bir veri seti) yanıltıcı olabilir. Bu nedenle, diğer metriklerle birlikte değerlendirilmesi önemlidir.

$$\text{Doğruluk} = \frac{\text{Doğru Pozitifler} + \text{Doğru Negatifler}}{\text{Tüm Tahminler}} \tag{3.3}$$

b) Keskinlik (Precision)

Keskinlik, denklem (3.4)'ten görüldüğü üzere, modelin yaptığı pozitif sınıf tahminlerinin ne kadarının doğru olduğunu ölçer. Yani, modelin saldırı tespit ettiğinde bu tahminlerin ne kadarının gerçekten saldırı olduğunu gösterir. Keskinlik, yanlış pozitiflerin önemli olduğu senaryolarda kullanışlıdır, çünkü modelin yanlış alarm verme eğilimini azaltmayı hedefler.

$$\text{Keskinlik} = \frac{\text{Doğru Pozitifler}}{\text{Doğru Pozitifler} + \text{Yanlış Pozitifler}} \tag{3.4}$$

c) Duyarlılık (Recall)

Duyarlılık, eşitlik (3.5)'te görüldüğü üzere modelin gerçek pozitifleri ne kadar doğru tespit ettiğini gösterir. Duyarlılık, saldırı tespitinde saldırıların gözden kaçmasını engellemek adına önemli bir ölçüttür. Duyarlılığı yüksek bir model, mümkün olduğunca az saldırıyı gözden kaçırmayı amaçlar.

$$\text{Duyarlılık} = \frac{\text{Doğru Pozitifler}}{\text{Doğru Pozitifler} + \text{Yanlış Negatifler}} \tag{3.5}$$

d) F1 Skoru

Eşitlik (3.6)'dan görüldüğü üzere, keskinlik ve duyarlılık metriklerinin harmonik ortalaması olan F1 skoru, modelin genel başarımını gösteren önemli bir metriktir ve keskinlik ile duyarlılık arasında sağlanan dengeyi ölçer. Bu denge, her iki ölçütün birlikte değerlendirilmesi sayesinde sağlanır. Özellikle dengesiz veri setlerinde, F1 skoru modelin hem yanlış pozitifler hem de yanlış negatifler üzerindeki başarımını genel biçimde değerlendirmeyi mümkün kılar. Diğer bir ifadeyle, F1 skoru hem hatalı pozitif tespitlerin hem de kaçırılan pozitif örneklerin etkisini dikkate alarak, modelin genel başarı düzeyini daha bütüncül bir yaklaşımla ölçer. Bu nedenle, tek başına doğruluk metriğinin yanıltıcı olabileceği senaryolarda, modelin farklı hata türlerine karşı nasıl bir performans sergilediğini göstermek açısından güvenilir bir ölçüm sağlar.

$$\text{F1} = 2 \times \frac{\text{Keskinlik} \times \text{Duyarlılık}}{\text{Keskinlik} + \text{Duyarlılık}} \tag{3.6}$$

e) Ortalama (Mean)

Ortalama, eşitlik (3.7)'de görüldüğü üzere bir veri setindeki tüm değerlerin toplamının, veri setindeki değer sayısına bölünmesiyle elde edilen metriktir. Basit bir ifadeyle, ortalama, veri setinin merkezini temsil eder ve verilerin genel eğilimini gösterir. Ortalama, genellikle modelin farklı doğrulama aşamalarında elde ettiği metriklerin aritmetik ortalamasını ifade eder. Örneğin, çapraz doğrulama sırasında her katmanda elde edilen doğruluk, keskinlik ya da F1 skoru gibi değerlerin ortalaması alınarak modelin genel başarımı hesaplanır.

$$Ortalama(\mu) = \frac{\sum_{i=1}^{n} x_i}{n} \tag{3.7}$$

$$\mu: Ortalama, \qquad n: Verilerin\ toplam\ sayısı, \qquad x_i: i.veri$$

f) Standart Sapma (Standard Deviation)

Standart sapma, eşitlik (3.8) de görüldüğü üzere veri setindeki değerlerin ortalamadan ne kadar saptığını, yani veri setinin yayılımını veya dağınıklığını ölçen bir değerdir. Standart sapma, modelin başarısındaki değişkenliği ölçmek için kullanılır. Düşük bir standart sapma, modelin doğrulama setleri üzerinde tutarlı bir performans sergilediğini gösterirken, yüksek bir standart sapma modelin doğrulama setlerine göre değişken sonuçlar verdiğini işaret eder.

$$Standart\ Sapma = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \mu)^2}{n}} \tag{3.8}$$

$$\mu: Ortalama, \qquad n: Verilerin\ toplam\ sayısı, \qquad x_i: i.veri$$

g) Karışıklık Matrisi (Confusion Matrix)

Karışıklık matrisi, makine öğrenmesi modellerinin başarımını değerlendirmek amacıyla kullanılan önemli bir araçtır. Bu matris, modelin yaptığı tahminler ile gerçek sonuçlar arasındaki ilişkiyi dört temel kategori üzerinden değerlendirir: Modelin pozitif sınıfı doğru tahmin ettiği durumlar (gerçek pozitifler), negatif sınıfı doğru tahmin ettiği durumlar (gerçek negatifler), pozitif sınıfı yanlış tahmin ettiği durumlar (yanlış pozitifler) ve negatif sınıfı yanlış tahmin ettiği durumlar (yanlış negatifler). Karışıklık matrisi, modelin hangi tür hatalar yaptığını ve bu hataların hangi sınıflarda yoğunlaştığını görmemizi sağlar. Bu sayede, yalnızca doğruluk gibi ölçütler yerine, modelin yanlış pozitif ve yanlış negatif tahminlerindeki başarısızlıklarını daha detaylı incelemek mümkündür. Karışıklık matrisi, Tablo 3.3'te gösterildiği gibi modelin iyileştirilebilecek alanlarını belirlemekte genel bir bakış açısı sağlamaktadır. Modelin hangi sınıflarda daha iyi veya kötü performans sergilediğini anlamada oldukça etkili bir değerlendirme aracıdır.

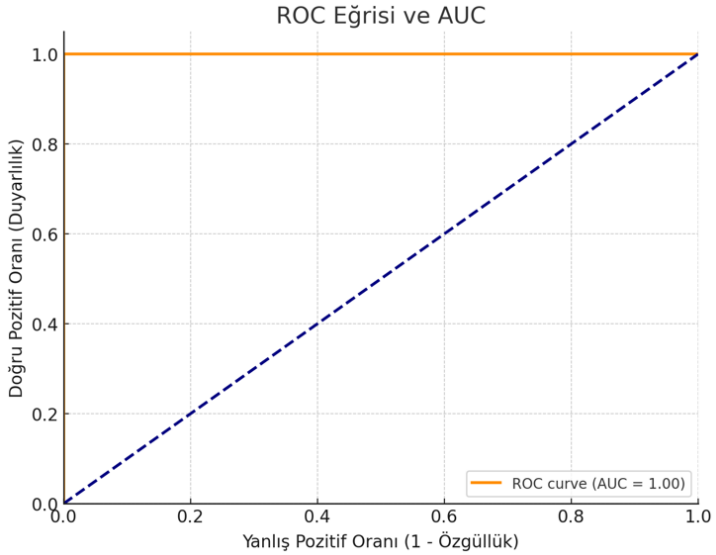
Tablo 3.3: Karışıklık matrisi.

Gerçek / Tahmin	Pozitif (1)	Negatif (0)
Pozitif (1)	Doğru Pozitif (TP)	Yanlış Negatif (FN)
Negatif (0)	Yanlış Pozitif (FP)	Doğru Negatif (TN)

h) ROC Eğrisi ve AUC Değeri

Alıcı İşletim Karakteristiği (ROC) eğrisi, ikili bir sınıflandırma modelinin çeşitli eşik değerleri için elde edilen Doğru Pozitif Oranı (TPR) ile Yanlış Pozitif Oranı (FPR) arasındaki ilişkiyi grafikte gösterir [51]. Bu grafik, modelin duyarlılığı (pozitif sınıflara ait örnekleri doğru tanımlama yeteneği) ile modelin bir örneği doğru pozitif olarak kabul etme

esnekliği arasındaki dengeyi bulmada görsel bir yardımcıdır. İdeal bir sınıflandırıcı Şekil 3.8'de görüldüğü gibi TPR'nin çok yüksek ve FPR'nin sınırlı olduğu sol üst köşeye dokunur. AUC (ROC Eğrisi Altındaki Alan), modelin genel başarımını özetleyen sayısal bir değer sağlar; bu değer 0.5 ile (rastgele tahmin yapıldığında) 1.0 (ideal sınıflandırma) arasında değişir. AUC, modelin rastgele seçilen bir pozitif örneği, bir negatif örnekten daha yüksek sıralama olasılığını ifade eder ve tüm eşik değerleri genelinde genel bir değerlendirme sunar. ROC ve AUC, sınıf dağılımına duyarlı olmadan model başarımına ilişkin bilgilendirici bir öngörü sağlar; bu durum, özellikle dengesiz veri kümeleri ile çalışırken son derece değerlidir.



Şekil 3.8: İdeal ROC ve AUC grafiği.

4. ARP SAHTECİLİĞİ TESPİT UYGULAMASI

Bu çalışmada, [7] çalışmasında önerilen ARP-PROBE sistemi IoT-ID [8] veri setinin kullanıldığı derin sinir ağları modeli ile gerçeklenmiş ve bağımsız olarak elde ettiğimiz sonuçlar bahsedilen çalışmada sunulan sonuçlar ile karşılaştırılmıştır. Kullanılan veri seti üzerinde yapılan incelemeler sonucunda [7] makalesindeki verilerle bizim işlediğimiz veriler arasında bazı farklılıklar tespit edilmiştir. Özel olarak, bazı paketlerin çıkarıldığı ve toplam veri sayısının azaltıldığı gözlemlenmiştir. Buna karşın, çalışmamızda tüm veri seti kullanılarak analizler gerçekleştirilmiştir. Bu kapsamda, ham veri setindeki tüm iyi huylu ve kötü huylu paketler korunarak, DNN modeline uygulanmıştır. Ayrıca, kullanılan DNN modeli sadece 25 epoch ile eğitilmişken, çalışmamızda daha kapsamlı bir analiz yapabilmek amacıyla DNN modelini farklı epoch sayıları (25, 50, 100 ve 200) ile eğiterek sonuçları karşılaştırdık. Böylece modelin farklı eğitim süreçlerinde nasıl bir performans gösterdiğini analiz etme imkânı ortaya çıkmıştır. Ayrıca, bu çalışmada yalnızca DNN modeli ile sınırlı kalınmamış, aynı veri seti üzerinde 4 farklı geleneksel makine öğrenme algoritması (Karar Ağaçları, Lojistik Regresyon, K-En Yakın Komşu ve Rastgele Orman) da uygulanmıştır. Bu sayede hem derin öğrenme hem de geleneksel makine öğrenme yöntemlerinin ARP sahteciliğini tespit etme başarımı kapsamlı bir şekilde değerlendirilmiştir.

Tablo 4.1'de verilen uygulama ortamında işlemler yapılmıştır. Ayrıca veri ön işleme adımlarından başlayarak model eğitimi ve değerlendirme sürecine kadar kullanılan metodoloji ayrıntılı bir şekilde bu bölümde açıklanmaktadır.

Tablo 4.1: Uygulama ortamı.

Bileşen	**Detaylar**
Çalışma Ortamı	Google Colab
Programlama Dili	Python
Python Versiyonu	3.08.2016
Kullanılan Kütüphaneler	TensorFlow 2.12.0, NumPy 1.22.4, Pandas 1.4.4, Scikit-learn 1.0.2, Matplotlib 3.5.2, Seaborn 0.11.2
GPU Kullanımı	Evet (Google Colab ile sağlanan Tesla T4 GPU)
İşletim Sistemi	Google Colab üzerinden sağlanan sanal ortam

4.1 Veri Ön İşleme

Çalışmamızda kullanılan veri setinde ilgili dosyalar Tablo 3.1'de gösterildiği gibi sadece normal ve ARP sahteciliği saldırı trafiği içeren dosyalar ayrıldıktan sonra, ARP-PROBE [7]

makalesinde belirtildiği gibi saldırı ve normal paketler arasındaki özellikler tanımlanmıştır. Söz konusu makalede 21 özellik seçilmiş olup, bu özellikler Tablo 4.2'te sunulmuştur.

Tablo 4.2: Seçilen özellikler.

Özellik İsmi	**Açıklaması**
frame.len	Paket uzunluğu (byte)
ip.proto	Protokol numarası
ip.len	IP paket uzunluğu (byte)
ip.ttl	IP paketindeki zaman aşımı (Time to Live) alanı
ip.flags	IP başlık içindeki bayraklar
ip.hdr_len	IP başlık uzunluğu
tcp.flags.syn	TCP başlığındaki SYN bayrağı
tcp.flags.ack	TCP başlığındaki ACK bayrağı
tcp.flags.reset	TCP başlığındaki RESET bayrağı
tcp.window_size	TCP başlığındaki pencere boyutu
tcp.checksum.status	TCP başlık denetim toplamı durumu
tcp.dstport	TCP hedef port numarası
tcp.srcport	TCP kaynak portu numarası
tcp.flags	Genel TCP bayrakları
tcp.len	TCP segment uzunluğu
tcp.time_delta	TCP segmentleri arasındaki zaman farkı
tcp.urgent_pointer	TCP acil işaretçi alanı
udp.srcport	Kullanıcı Datagram Protokolü (User Datagram Protocol - UDP) kaynak port numarası
udp.dstport	UDP hedef port numarası
icmp	ICMP protokolünü belirleyen bayrak
arp	ARP protokolünü belirleyen bayrak

Seçilen özelliklerin veri paketlerinden filtrelenmesi ve ayrıştırılması işlemi Tshark ile gerçekleştirilmiştir. Tshark, Wireshark'ın komut satırı versiyonudur ve ağ trafiği verilerinden spesifik özellikleri çıkarmaya olanak tanır.

Tshark ile Özelliklerin Filtrelenmesi ve CSV Dosyasının Oluşturulması

Ağ trafiği verilerini analiz etmek için kullanılan Tshark aracı, ağ paketlerini `pcap` formatından daha işlenebilir bir format olan Virgülle Ayrılmış Değerler (Comma-Separated Values - CSV) dönüştürmek için kullanılmıştır. Aşağıdaki kod bloğu, bir Paket Yakalama (Packet Capture- PCAP) dosyasından belirlenen özelliklerin filtrelenerek bir CSV (`all_packets.csv`) dosyasına yazdırılması sürecini göstermektedir.

```
tshark -r "/Users/furkanceylan/Downloads/iot_intrusion_dataset/Tsharp
Deneme/capture.pcap" -T fields \
-e frame.number \
-e frame.len \
-e ip.proto \
-e ip.len \
-e ip.ttl \
-e ip.flags \
-e ip.hdr_len \
-e arp.opcode \
-e tcp.flags.syn \
-e tcp.flags.ack \
-e tcp.flags.reset \
-e tcp.window_size \
-e icmp.type \
-e tcp.checksum.status \
-e tcp.dstport \
-e tcp.srcport \
-e tcp.flags \
-e tcp.len \
-e tcp.time_delta \
-e tcp.urgent_pointer \
-e udp.srcport \
-e udp.dstport \
-E header=y -E separator=, >
"/Users/furkanceylan/Downloads/iot_intrusion_dataset/Tsharp
Deneme/all_packets.csv"
```

Tshark ile Saldırı Trafiğinin Filtrelenmesi ve CSV Dosyasına Yazdırılması

Ağ trafiği analizi yaparken saldırı trafiğinin doğru bir şekilde ayrıştırılması, etkin bir analiz ve tespit süreci için önemlidir. Bu adımda, Tshark aracı kullanılarak ARP sahteciliği saldırılarının bulunduğu ağ paketleri, belirlenen filtreler yardımıyla ham veri setinden ayrılmış ve sadece ilgili trafik çıkarılmıştır.

Her dosya için Kang ve ark. tarafından IoT-ID [8] veri seti ile verilen filtreler kullanılarak saldırı paketleri veriden ayrılmıştır.

Örneğin mitm-arpspoofing-1-dec.pcap dosyası için saldırı paketlerinin Wireshark filtresi:

```
eth.addr == f0:18:98:5e:ff:9f and (((ip.src == 192.168.0.16 and ip.dst ==
192.168.0.13) or (ip.src == 192.168.0.13 and ip.dst == 192.168.0.16)) and
!icmp and tcp) or (arp.src.hw_mac == f0:18:98:5e:ff:9f and (arp.dst.hw_mac
== bc:1c:81:4b:ae:ba or arp.dst.hw_mac == 48:4b:aa:2c:d8:f9))
```

Aşağıda verilen komut, saldırı paketlerinin filtrelenmesi ve CSV (`attack_packets.csv`) formatına dönüştürülmesini sağlamaktadır:

```
tshark -r "/Users/furkanceylan/Downloads/iot_intrusion_dataset/Tsharp
Deneme/capture.pcap" \
-Y 'eth.addr == f0:18:98:5e:ff:9f and (((ip.src == 192.168.0.16 and
ip.dst == 192.168.0.13) or (ip.src == 192.168.0.13 and ip.dst ==
192.168.0.16)) and !icmp and tcp) or (arp.src.hw_mac == f0:18:98:5e:ff:9f
and (arp.dst.hw_mac == bc:1c:81:4b:ae:ba or arp.dst.hw_mac ==
48:4b:aa:2c:d8:f9))' \
-T fields \
-e frame.number \
-e frame.len \
-e ip.proto \
-e ip.len \
-e ip.ttl \
-e ip.flags \
-e ip.hdr_len \
-e arp.opcode \
-e tcp.flags.syn \
-e tcp.flags.ack \
-e tcp.flags.reset \
-e tcp.window_size \
-e icmp.type \
-e tcp.checksum.status \
-e tcp.dstport \
-e tcp.srcport \
-e tcp.flags \
-e tcp.len \
-e tcp.time_delta \
-e tcp.urgent_pointer \
-e udp.srcport \
-e udp.dstport \
-E header=y -E separator=, \
> "/Users/furkanceylan/Downloads/iot_intrusion_dataset/Tsharp
Deneme/attack_packets.csv"
```

Bu çalışmada, her bir `pcap` dosyasından Tablo 4.2'te sunulan özellikler seçilerek tek bir CSV dosyası halinde tüm paketlerin bulunduğu `all_packets.csv` dosyası oluşturulmuştur. Ardından, saldırı trafiğini ayrıştırmak amacıyla belirli filtreler uygulanarak sadece saldırı paketlerini içeren `attack_packets.csv` dosyaları hazırlanmıştır. Bu işlem, ağ trafiği içerisinde saldırı paketlerinin etkin bir şekilde belirlenebilmesi için gerçekleştirilmiştir. Tüm paketleri içeren `all_packets.csv` dosyasına, saldırı olup olmadığını göstermek amacıyla *attack_flag* adı verilen yeni bir sütun eklenmiştir. Bu sütunun başlangıç değeri olarak tüm satırlara 0 atanmıştır, yani varsayılan olarak tüm paketlerin normal trafik olduğu kabul edilmiştir. Daha sonra, saldırı paketlerinin bulunduğu `attack_packets.csv` dosyasındaki *frame.number* ile tüm paketlerin bulunduğu `all_packets.csv` dosyasındaki *frame.number* bilgileri karşılaştırılmış ve eşleşen satırlarda *attack_flag* sütununa 1 değeri eklenmiştir. Böylece, oluşturulan son veri setinde her bir paketin saldırı veya normal trafik olup olmadığı

net bir şekilde işaretlenmiştir. Aşağıda kod bloğu verilen ön işleme adımı, veri setinin daha sonra gerçekleştirilecek analiz ve modelleme süreçlerinde saldırı tespitini daha kolay ve etkili bir şekilde gerçekleştirilmesine olanak tanımaktadır.

```
# all packets dosyasına 'attack_flag' stununun eklenmesi, her değeri
başta 0 (normal trafik) olarak atanır.
all_packets['attack_flag'] = 0

# Eşleşen frame numaraları için 'attack_flag' değerini 1'e
ayarlayarak saldırı paketleri işaretlenir.
all_packets.loc[all_packets['frame.number'].isin(attack_packets['fram
e.number']), 'attack_flag'] = 1
```

Saldırı ya da normal trafik olarak işaretlenen veriye sonraki (verinin işlenmesini bozan durumların düzeltilmesi) ön işleme adımları uygulanır. İlk olarak, *tcp.flags* ve *ip.flags* sütunlarındaki veriler onaltılıdan onluya çevrilerek, verinin sayısal analiz için uygun hale getirilmesi sağlanmıştır. Ardından, ICMP ve ARP protokollerinin doğru sınıflandırılabilmesi amacıyla, ilgili protokol bilgileri kullanılarak yeni ICMP ve ARP sütunları eklenmiştir. Bu sütunlar karşıt olacak şekilde düzenlenmiş, böylece bir paketin hem ICMP hem de ARP olarak sınıflandırılmaması sağlanmıştır. Ayrıca, protokole özgü özelliklerin doğru modellenmesi amacıyla, UDP paketlerinde TCP'ye ait özellikler ve TCP paketlerinde UDP'ye ait özellikler -1 ile işaretlenmiştir. Bu sayede, her protokole ait olmayan gereksiz bilgilerin modelleri olumsuz etkilemesi engellenmiştir. *tcp.flags* bayrakların da, *true* 1 ile *false* 0 ile değiştirilerek daha basit ve doğru bir analiz yapılması amaçlanmıştır. Son olarak, eksik veriler -1 ile doldurularak, eksikliklerin analiz sürecinde problemlere yol açması önlenmiştir. Ardından veriler 0 ile 1 arasında ölçeklenerek tekrar yazılmıştır. Ön işleme adımları, veri setinin tutarlı, eksiksiz ve analiz için hazır hale getirilmesini sağlamıştır.

Bu çalışmada, altı farklı saldırı trafiğini ve bir normal trafiği (Tablo 3.2'den görüldüğü gibi) içeren pcap dosyaları üzerinde veri ön işleme adımları uygulanmıştır. Bu dosyalar, daha kolay işlenebilmesi amacıyla tek bir CSV dosyasında birleştirilmiştir. Her bir dosyanın satır ve sütun sayılarının, orijinal veri setindeki yapıyla tutarlı olduğu doğrulanmıştır. Aşağıdaki kod çıktısı, işlenen her bir dosyanın satır (paket) sayısını ve en sonunda toplam sayısını yazdırmaktadır. Tüm işlem kolaylığı için tüm dosyalar tek bir CSV dosyasında birleştirilmiştir. Aşağıda verilen log çıktısında da görüldüğü üzere, bu birleşim sonucu

oluşturulan dosya, 21 özellik ve eklenen *attack_flag* sütunundan oluşmakta olup, toplamda 331,580 satır içerdiği doğrulanmıştır.

```
Number of rows in benignFinal.csv: 137396
Processing file: /content/drive/MyDrive/ARP Spoofing DL
Aplication/allCsvFiles/arpFile3Final.csv
Number of rows in arpFile3Final.csv: 34043
Processing file: /content/drive/MyDrive/ARP Spoofing DL
Aplication/allCsvFiles/arpFile4Final.csv
Number of rows in arpFile4Final.csv: 19914
Processing file: /content/drive/MyDrive/ARP Spoofing DL
Aplication/allCsvFiles/arpFile5Final.csv
Number of rows in arpFile5Final.csv: 20314
Processing file: /content/drive/MyDrive/ARP Spoofing DL
Aplication/allCsvFiles/arpFile6Final.csv
Number of rows in arpFile6Final.csv: 21024
Processing file: /content/drive/MyDrive/ARP Spoofing DL
Aplication/allCsvFiles/arpFile1Final.csv
Number of rows in arpFile1Final.csv: 65768
Processing file: /content/drive/MyDrive/ARP Spoofing DL
Aplication/allCsvFiles/arpFile2Final.csv
Number of rows in arpFile2Final.csv: 33121

Total number of rows in combined DataFrame: 331580

Number of rows: 331580
Number of columns: 22
```

Veri, makine öğrenmesi modellerine sunulmadan önce, olası sınıf dengesizliklerine karşı analiz edilmiştir. Saldırı ve normal trafik örneklerinin sayıları arasında bir dengesizlik olup olmadığını belirlemek için, daha önce `attack_flag` sütunuyla işaretlenen veriler analiz edilmiştir. Veri setinde `attack_flag` değeri 0 olan (normal trafik) 229.695 paket, `attack_flag` değeri 1 olan (saldırı trafiği) 101.885 paket bulunmaktadır.

Veride `attack_flag` sütunu sorgusu yapıldıktan sonra elde edilen kod çıktısı:

```
attack_flag
0    229695 (normal paket sayısı)
1    101885 (saldırı paket sayısı)
```

Bu durum, veri setinde dengesizlik olduğunu göstermektedir. Sınıf dengesizliğini gidermek ve makine öğrenmesi modellerinin saldırı tespiti yaparken daha etkili olmasını sağlamak amacıyla, Rastgele Örnekleyici (Random Over Sampler) yöntemi kullanılmıştır. Örnek çoğaltıcı metot, sınıf dengesizliğini çözmek için azınlık sınıfına (bu çalışmada saldırı örnekleri) ait verilerin çoğaltılması işlemidir. Bu yöntemle, saldırı örneklerinin sayısı normal

trafik örneklerine eşit olacak şekilde artırılmıştır. Sonuç olarak, veri setinde 229.695 saldırı örneği ve 229.695 normal trafik örneği olmak üzere toplamda 459.390 örnek elde edilmiştir. Bu işlem, veri setinde sınıflar arası dengenin sağlanmasına ve makine öğrenmesi modellerinin saldırı tespit performansının iyileştirilmesine katkıda bulunmuştur.

Örnek çoğaltma işlemi sonrasında `attack_flag` sütunu sorgusu yapıldıktan sonra elde edilen kod çıktısı:

```
attack_flag
0    229695 (normal paket sayısı)
1    229695 (saldırı paket sayısı)
```

4.1.1 Alani ve ark. (ARP-PROBE) Çalışması Veri Bilgileri ile Karşılaştırma

Alani ve ark. tarafından yayımlanan [7] çalışmada, aynı IoT-ID [8] veri seti kullanılmasına rağmen işlenen paket sayılarında farklılıklar gözlemlenmiştir. Bu durum, hem ham veri setindeki paket sayıları hem de ön işleme adımlarından sonra elde edilen veriler açısından belirgin bir farklılık yaratmıştır. Aşağıda, çalışmamız ile Alani ve ark. çalışması arasındaki veri farklılıklarını detaylı bir şekilde karşılaştıran bir analiz sunulmaktadır.

Ham Paket Sayısı:

- [7] çalışmasında, ARP sahteciliği ve iyi huylu (*benign*) trafik içeren dosyalardan toplamda 279,105 paket olduğu belirtilmiştir.
- Çalışmamızda ise ham veri setinde 331,580 paket olduğu hesaplanmıştır. Bu sayı, makaledeki ham veri setinden daha fazladır. Bu fark, makalenin belirli dosyaları hariç tutmuş olabileceğini düşündürmektedir.

İyi Huylu ve Kötü Huylu Paket Sayısı (Ham):

- Bahsedilen makalede, ham veri setinde 214,459 iyi huylu paket ve 64,646 kötü huylu paket olduğunu belirtilmektedir.
- Bizim bulgularımızda ise 229,695 iyi huylu paket ve 101,885 kötü huylu paket bulunmaktadır. Bu da özellikle kötü huylu paket sayısında önemli bir farkın olduğunu göstermektedir. Bu fark, veri setinde belirli saldırı dosyalarının çıkarılmış olabileceğine işaret edebilir.

Ön İşleme Sonrası Paket Sayısı:

- Bahsedilen çalışmada, rastgele örnekleme yöntemi ile sınıf dengesizliği giderilerek 214,459 normal ve 214,459 saldırı paketi elde edilmiştir. Toplamda 428,918 paket olduğu belirtilmiştir.

- Bizim çalışmamızda ise rastgele örnekleme kullanılarak 229,695 normal ve 229,695 saldırı paket olmak üzere toplamda 459,390 paket bulunmaktadır. Ham veri setindeki farklar, bu aşamada da kendini göstermektedir.

Sonuç:

- [7] çalışmasında, ham veri setinden bazı saldırı dosyalarının çıkarılmış olabileceği düşünülmektedir. Bu çıkarım, ham veri setindeki paket sayısındaki fark ile desteklenmektedir.
- Ayrıca, kötü huylu paket sayısındaki fark özellikle dikkat çekicidir: Alani ve ark. makalesinde 64,646 olan iyi huylu paket sayısı, bizim çalışmamızda 101,885 olarak hesaplanmıştır. Bu, veri setinin bazı bölümlerinin çıkarılmış veya veri setinde eksiltme yapılmış olabileceğini göstermektedir.
- Son olarak, her iki çalışmada da sınıf dengesizliği rastgele örnekleme yöntemi ile giderilmiş olsa da, ham verideki farklar nedeniyle özellikle kötü huylu paket sayısında ciddi bir değişiklik gözlemlenmektedir.

Bu farklılıklar deney sonuçlarına da yansımıştır ve bu karşılaştırma DNN uygulamarında 5.1.1 bölümünde (Deney 1 - 25 Epoch) yapılmıştır. Veri seti sayılarındaki farkların genel karşılaştırması Tablo 4.3'de sunulmuştur.

Tablo 4.3: Alani ve ark. makalesi ve çalışmamız arasındaki veri seti karşılaştırması.

Veri Özelliği	**Alani ve ark. çalışması**	**Iot-ID Veriseti Ham**	**Çalışmamız**
Toplam Paket Sayısı (Ham)	279,105	331,58	331,58
Normal Paket Sayısı (Ham)	214,459	229,695	229,695
Saldırı Paket Sayısı (Ham)	64,646	101,885	101,885
Ön İşleme Sonrası Paket Sayısı	428,918	x	459,390
Ön İşleme Sonrası (Normal)	214,459	x	229,695
Ön İşleme Sonrası (Saldırı)	214,459	x	229,695

4.2 Derin Sinir Ağı Uygulaması

ARP sahteciliğinin tespiti için [7] çalışmasında tasarlanan derin öğrenme tabanlı bir model ile yapılan çalışmanın yeniden denenmesi üzerine odaklanılmıştır. Burada, IoT-ID veri kümesi kullanılarak bahsedilen çalışmada önerilen DNN modeli ile aynı yapı tekrar edilmiş, ancak veri setinin tamamı kullanılarak daha kapsamlı bir uygulama gerçekleştirilmiştir. Veri kümesi üzerinde çeşitli ön işleme adımları gerçekleştirilmiş ve elde edilen veriler DNN modeli ile eğitilmiştir. Modelin performansı, doğruluk, keskinlik, duyarlılık ve F1 skoru gibi değerlendirme metrikleri kullanılarak ölçülmüştür edilmiştir

Veri Ayrımı: Veri kümesi, özellikler ve etiketler olmak üzere ikiye ayrılmıştır. Özellikler (x), ağ trafiğine ait çeşitli nitelikleri (paket uzunluğu, TCP bayrakları, IP bayrakları vb.) içerirken, etiketler (y), trafiğin ARP sahteciliği saldırısı olup olmadığını (attack_flag) belirtmektedir.

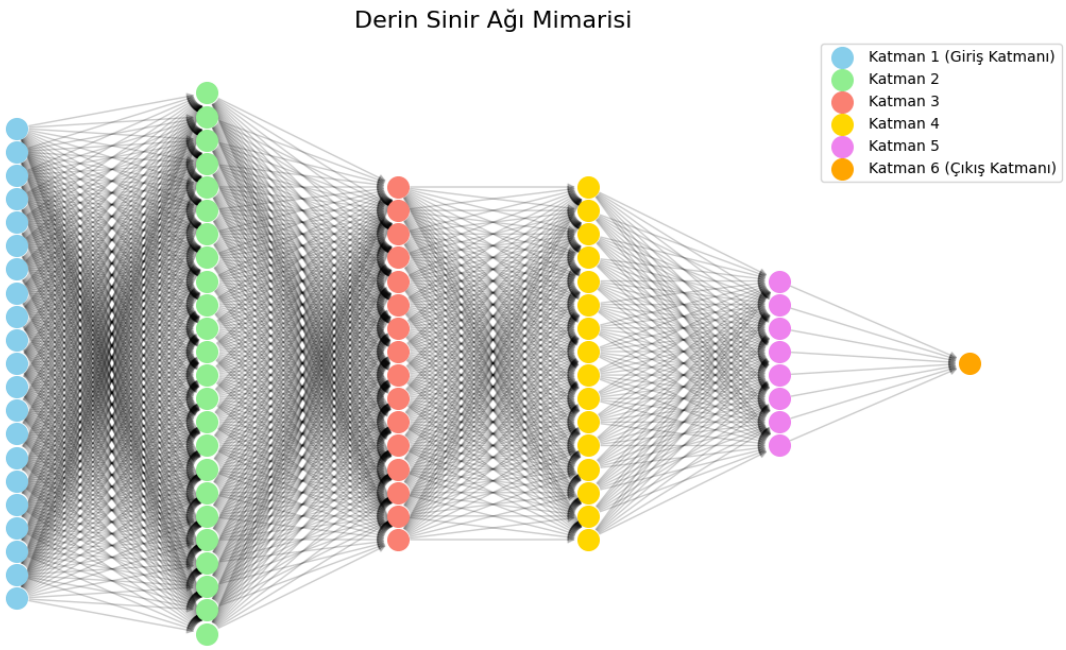
```
# Separate features and labels
X = df.drop(columns=['attack_flag'])
y = df['attack_flag']
```

Eğitim ve Test Seti: Veriler, sınıfların (normal ve saldırı) dengeli bir şekilde dağıtıldığından emin olmak amacıyla katmanlı ayrım (*stratified split*) yöntemi kullanılarak %75 eğitim ve %25 test olacak şekilde ayrılmıştır. Bu işlem, `train_test_split` fonksiyonu ile gerçekleştirilmiş ve `stratify=y` parametresi kullanılarak her iki sınıftan da dengeli örneklem alınmıştır.

```
# Stratified split: 75% eğitim, 25% test
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.25, stratify=y, random_state=42)
#stratify=y her sınıftan örnek alındığından emin olur
```

4.2.1 Derin Sinir Ağı (DNN) Modeli

Şekil 4.1'de gösterilen derin sinir ağı modeli [7] ARP sahteciliği tespiti için optimize edilmiş çok katmanlı bir yapıya sahiptir. Çeşitli katmanlardan oluşan DNN, giriş katmanı verileri işleyip, çıkış katmanında ARP sahteciliği saldırısını sınıflandırmak üzere eğitilmiştir.



Şekil 4.1: DNN model mimarisi.

Model Mimarisi:

- Giriş katmanı, eğitim veri kümesinin özellik sayısı olan 21 nörondan oluşmaktadır ve bu sayı, `input_dim = X_train.shape[1]` ifadesi kullanılarak veri kümesinin özellik sayısına göre dinamik olarak belirlenmektedir.
- Ardından, 24 nöron içeren bir gizli katman, 16 nöron içeren iki gizli katman ve 8 nöron içeren bir gizli katman yerleştirilmiştir. Bu katmanlarda aktivasyon fonksiyonu olarak ReLU kullanılmıştır.
- Son olarak, ARP sahtecilik tespitini gerçekleştiren bir nöronlu çıkış katmanı kullanılmıştır. Çıkış katmanında ikili sınıflandırma için *sigmoid* aktivasyon fonksiyonu tercih edilmiştir.

Aşağıda model mimarisinin tasarlandığı kod bloğu sunulmuştur.

```
# Model mimarisinin tanımı
def create_model():
    model = Sequential()
    model.add(Dense(24, input_dim=X_train.shape[1],
activation='relu'))
    model.add(Dense(16, activation='relu'))
    model.add(Dense(16, activation='relu'))
    model.add(Dense(8, activation='relu'))
```

```
    model.add(Dense(1, activation='sigmoid'))
    model.compile(optimizer=Nadam(), loss=BinaryCrossentropy(),
metrics=['accuracy'])
    return model
```

Modelin eğitimi sırasında kullanılan optimizatör, *Nadam* algoritmasıdır. Bu optimizatör, sinir ağlarının hızla ve verimli bir şekilde eğitilmesini sağlar. Modelin kayıp fonksiyonu olarak ikili çapraz entropi (*binary crossentropy*) seçilmiştir. Bu fonksiyonun kullanımı, ikili sınıflandırma görevlerinde standart bir yöntemdir ve modelin yanlış tahminleri en aza indirgemek amacıyla kullanılmıştır. Modelin performansı, eğitim sırasında doğruluk metriği kullanılarak izlenmiş ve model doğrulama veri kümesi ile de test edilmiştir.

Model Eğitimi ve Değerlendirme

Modelin eğitim sürecinde, veri kümesinin %75'lik kısmı eğitim seti olarak kullanılarak gerçekleştirilmiştir. Eğitim setindeki örnekler, belirli bir büyüklüğe sahip gruplara (batch'lere) ayrılır. Eğitim sırasında, her bir deneyde grup büyüklüğe 1024 olarak belirlenmiş ve modelin genel performansını değerlendirmek amacıyla dört farklı epoch değeri (25, 50, 100 ve 200 epoch) kullanılarak deneyler yapılmıştır. Eğitim sırasında modelin doğrulama performansı, test veri seti (%25 lik kısım) üzerinde değerlendirilmiştir. Modelin her epoch sonundaki kayıp ve doğruluk değerleri takip edilmiş ve grafiksel olarak 5.1 bulgular bölümünde sunulmuştur.

Eğitim tamamlandıktan sonra elde edilen modelin, test veri kümesi üzerinde değerlendirilmesiyle, doğruluk, keskinlik, duyarlılık ve F1 skoru metrikleri ile modelin performansı ölçülmüştür. Sonuçlar, aşağıdaki kod bloğunda detaylandırıldığı gibi `accuracy_score`, `precision_score`, `recall_score` ve `f1_score` fonksiyonları kullanılarak hesaplanmıştır.

```
final_model = create_model()
history = final_model.fit(X_train, y_train, epochs=25,
batch_size=1024, verbose=1, validation_data=(X_test, y_test))
```

Sonuçların değerlendirilmesi için modelin performansı çeşitli metrikler ve grafikler kullanılarak incelenmiştir. Aşağıdaki verilen kod bloğunda görüleceği gibi doğruluk, keskinlik, duyarlılık ve F1 skoru gibi temel performans ölçütleri hesaplanmış, eğitim ve doğrulama setlerine ait doğruluk ve kayıp değerleri epoch sayısına bağlı olarak izlenmiştir. Bu işlemler, modelin sınıflandırma yeteneğini ve eğitimi sırasında aşırı öğrenme yaşayıp

yaşamadığını görmek amacıyla yapılmıştır. Karışıklık matrisi, modelin doğru ve yanlış sınıflandırmalarını görselleştirerek, pozitif ve negatif sınıflandırmalarda modelin performansını analiz etmeye olanak tanımaktadır. ROC eğrisi ve AUC değeri ise modelin doğru pozitif ve yanlış pozitif oranlarının nasıl dağıldığını incelemek için kullanılmaktadır. Bu değerlendirmeler, modelin çeşitli açılardan performansını anlamak için gerçekleştirilmiştir ve her bir deney için bulgular bölümünde sunulmuştur.

```
# Son modelin test verisinde değerlendirilmesi.
test_loss, test_accuracy = final_model.evaluate(X_test, y_test,
verbose=0)
print(f"Test Accuracy: {test_accuracy:.4f}")

# Model tahmininin alınması (karışıklık matrisi için)
y_pred = (final_model.predict(X_test) > 0.5).astype("int32") #Binary
predictions
y_pred_proba = final_model.predict(X_test)  # Probabilities for ROC

# Karışıklık matrisinin hesaplanması
conf_matrix = confusion_matrix(y_test, y_pred)

# Karışıklık matrisinin çizilmesi.
plot_confusion_matrix_turkish(conf_matrix)

#Accuracy, precision, recall, and F1 score
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
```

4.2.2 Çapraz Doğrulama (Cross-Validation)

Bu deneyde, uygulanan ARP sahteciği tespit modelinin performansını değerlendirmek için 10 katmanlı çapraz doğrulama yöntemi gerçekleştirilmiş ve bu yöntem DNN'de uygulanan her epoch (25, 50, 100 ve 200) seviyesinde tekrarlanmıştır. Çapraz doğrulama, modelin performansını tek bir eğitim ve test ayrımına bağlı kalmadan değerlendiren, genellemeye dayalı bir yöntemdir. Bu deneyde aşağıdaki kod ile gerçeklenen 10 katmanlı çapraz doğrulama yöntemi modelin istikrarını, güvenilirliğini ve genel etkinliğini ölçmek için kullanılmıştır.

```
# 10-fold cross-validation
kfold = StratifiedKFold(n_splits=10, shuffle=True, random_state=42)
```

Modelin her bir katman üzerinde eğitilip değerlendirilmesinden sonra, sonuçlar kaydedilmiş ve her bir katman için metrikler ayrı ayrı yazdırılmıştır. Bu süreç, tüm katmanlar tamamlanana kadar tekrar edilmiştir. Çapraz doğrulama süreci tamamlandıktan sonra, her bir katman için hesaplanan doğruluk, keskinlik, duyarlılık ve F1 skorlarının ortalaması ve standart sapması hesaplanmıştır. Bu özet istatistikler, modelin genel performansını ve kararlılığını ortaya koymuştur ayrıca modelin ortalama performansını ve farklı veri alt kümeleri üzerindeki kararlılığını ortaya koymaktadır.

4.3 Geleneksel Makine Öğrenmesi Yöntemlerinin Uygulanması

Bu çalışmada, ARP sahteciliği tespitine yönelik dört farklı geleneksel makine öğrenmesi algoritması kullanılmıştır. Karar Ağacı, Lojistik Regresyon, Rastgele Orman ve K-En Yakın Komşu algoritmaları, veri seti üzerinde eğitim ve test edilerek değerlendirilmiştir. Bu bölümde, kullanılan yöntemlerin teknik uygulama süreçleri ele alınmaktadır.

Verilerin Hazırlanması ve Ölçeklendirilmesi

Veri seti, ARP sahteciliği tespiti amacıyla uygulanan makine öğrenmesi algoritmalarında kullanılmak üzere hazırlanmıştır. Aşağıdaki kod bloğunda sunulduğu gibi scaled_final_data.csv dosyası kullanılarak veri seti yüklenmiş ve özellikler (bağımsız değişkenler) ile etiketler (bağımlı değişken) olarak iki ana bileşene ayrılmıştır. Bağımsız değişkenler, ARP sahteciliğini tespit etmek için gereken özellikleri temsil ederken, bağımlı değişken "`attack_flag`" etiketi sahtecilik saldırılarını temsil etmektedir.

```
# Verilerin yüklenmesi
directory = '/content/drive/MyDrive/ARP Spoofing DL
Aplication/allCsvFiles'
data_path = os.path.join(directory, 'scaled_final_data.csv')
df = pd.read_csv(data_path)
# Etiketlerin ve özelliklerin ayrılması
X = df.drop(columns=['attack_flag'])
y = df['attack_flag']
```

Eğitim ve Test Setlerinin Ayrılması

Veri seti, model performansını değerlendirebilmek için aşağıdaki kodda görüldüğü üzere %75 eğitim ve %25 test olarak ayrılmıştır. Bu işlem sırasında, veri setindeki sınıf dengesizliklerini korumak amacıyla katmanlı örnekleme (stratified sampling) uygulanmıştır. Katmanlı örnekleme, her sınıfın orantılı olarak eğitim ve test setlerine dağıtılmasını sağlar.

```
# Eğitim ve test setlerine ayırma
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y,
test_size=0.25, stratify=y, random_state=42)
```

Karar Ağacı

Karar ağacı, sınıflar arasındaki ayrımı dallar ve yapraklar halinde yapılandırarak gerçekleştirir. Bu model, özellikler arasındaki ilişkilere dayalı olarak karar kuralları oluşturur ve sınıflandırma işlemi yapar. Aşağıdaki kod bloğunda karar ağacı modelinin uygulaması sunulmuştur.

```
# Karar Ağacı modeli
model = DecisionTreeClassifier()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
```

Rastgele Orman

Rastgele orman, birçok karar ağacından oluşan bir topluluk modelidir. Her bir ağaç, veri setinin rastgele bir alt kümesi üzerinde eğitilir ve bu ağaçların tahminleri birleştirilerek nihai sınıflandırma yapılır. Aşağıdaki kod bloğunda rastgele orman modelinin uygulaması sunulmuştur.

```
# Rastgele Orman modeli
model = RandomForestClassifier(n_estimators=100)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
```

K-En Yakın Komşu

KNN algoritması, her bir veri noktasını varsayılan olarak k = 5 en yakın komşusuna göre sınıflandırır. KNN için kullanılan *sklearn.neighbors.KNeighborsClassifier* kütüphanesi, k değerini otomatik olarak 5 olarak atar ve KNN, örnekler arasındaki mesafeyi kullanarak sınıflandırma işlemini gerçekleştirir. Aşağıdaki kod bloğunda KNN modelinin uygulaması sunulmuştur.

```
# K-En Yakın Komşu modeli
model = KNeighborsClassifier()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
```

Lojistik Regresyon

Lojistik regresyon, ikili sınıflandırma problemlerinde kullanılan bir modeldir. Bu model, her örneği bir sınıfa tahsis eder. Aşağıdaki kod bloğunda lojistik regresyon modelinin uygulaması sunulmuştur.

```
# Lojistik Regresyon modeli
model = LogisticRegression()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
```

Performans Değerlendirme Süreci

Her modelin performansı, aşağıdaki kod bloğunda görüldüğü gibi doğruluk, keskinlik, duyarlılık, F1 skoru ve AUC gibi metriklerle değerlendirilmiştir. Performans ölçütleri, her bir modelin tahmin yeteneğini kapsamlı bir şekilde değerlendirmek için kullanılmıştır. Ayrıca, modellerin performansını görselleştirmek amacıyla karışıklık matrisleri oluşturulmuştur:

```
# Performans ölçütlerinin hesaplanması
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
auc = roc_auc_score(y_test, y_pred_proba) if hasattr(model,
"predict_proba") else "N/A"
# Karışıklık matrisi
conf_matrix = confusion_matrix(y_test, y_pred
```

5. BULGULAR

Bu bölümde, Bölüm 4'te uygulanmış olan yöntemlerin çıktıları paylaşılmıştır.

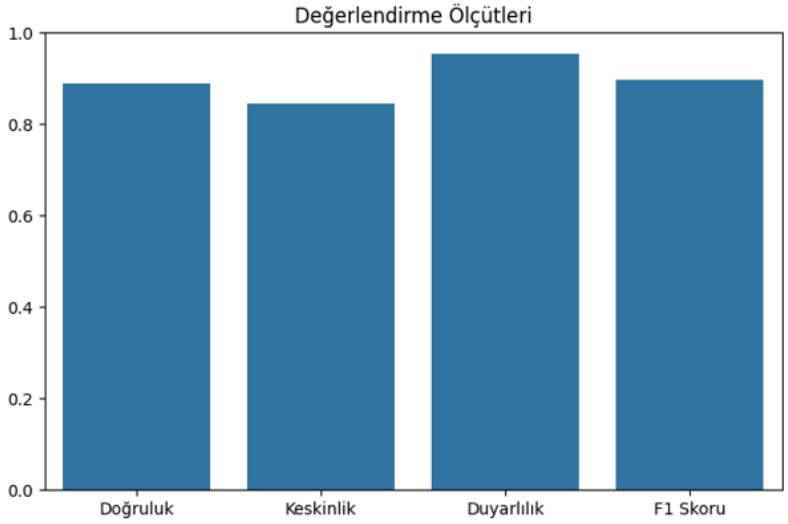
5.1 Derin Sinir Ağları Yönteminin Deney Sonuçları

Burada, DNN yöntemi ile yapılan deneylerin sonuçları sunulmaktadır. Yürütülen her bir deney, modelin farklı epoch sayıları kullanılarak eğitilmesi ile gerçekleştirilmiştir. Epoch, eğitim verisinin modelde kaç kere uygulanacağını ifade eden bir terimdir. Daha fazla epoch, modelin verilerden öğrenme süresini uzatırken, daha düşük epoch sayısı modelin daha hızlı ancak yüzeysel öğrenmesine neden olabilir. Aşağıda, bu çalışmada kullanılan dört farklı epoch sayısının (25, 50, 100 ve 200 epoch) modelin performansı üzerindeki etkilerine değinilecektir.

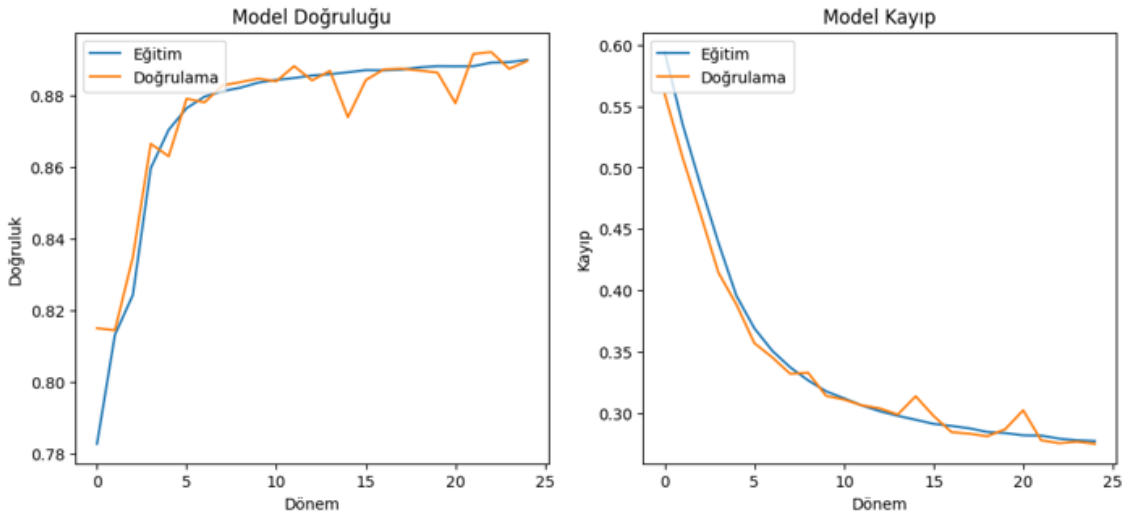
5.1.1 Deney 1 – 25 Epoch

Bu deney, Alani ve ark. (ARP-PROBE) [7] çalışmasındaki model yapısı ile aynı olduğundan karşılaştırma açısından kritik bir öneme sahiptir. Alani ve ark. çalışmasında olduğu gibi, bu deneyde de model 25 epoch boyunca eğitilmiştir. Alani ve ark. tarafından yapılan çalışmayla aynı model ve epoch sayısının kullanılması, bu çalışmada elde edilen sonuçların kapsamlı olarak karşılaştırılmasını sağlayacaktır. Deneyin sonunda bahsedilen karşılaştırma yapılmıştır.

Eğitim sonrası test verisinin uygulanmasıyla, aşağıda verilen Şekil 5.1'de grafiksel olarak gösterilen başarım metrikleri elde edilmiştir: doğruluk %88,96, keskinlik %84.51, duyarlılık %95.40 ve F1 skoru %89.63. Bu sonuçlar, modelin ilk 25 epoch boyunca eğitim sürecindeki başarısını yansıtmaktadır. Model, oldukça yüksek bir duyarlılık oranına sahip olup, pozitif sınıfları doğru tahmin etme konusunda başarılıdır. Keskinlik oranı biraz daha düşük olmakla birlikte, yanlış pozitif oranı kabul edilebilir seviyededir. Keskinlik ve duyarlılık arasında denge sağlayan bir metrik olan F1 skoru ise modelin genel performansının kabul edilebilir seviyede olduğunu göstermektedir.



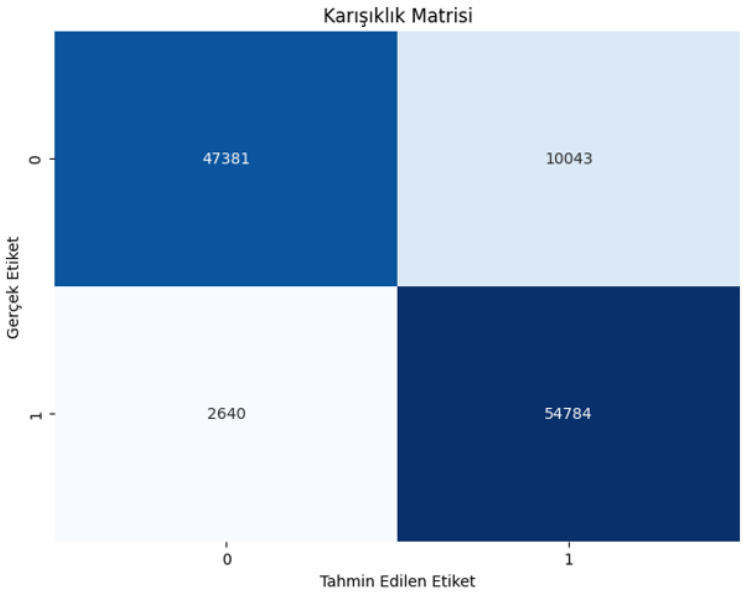
Şekil 5.1: Ölçüm metrikleri grafiği – Deney 1.



Şekil 5.2: Model doğruluk ve kayıp grafiği – Deney 1.

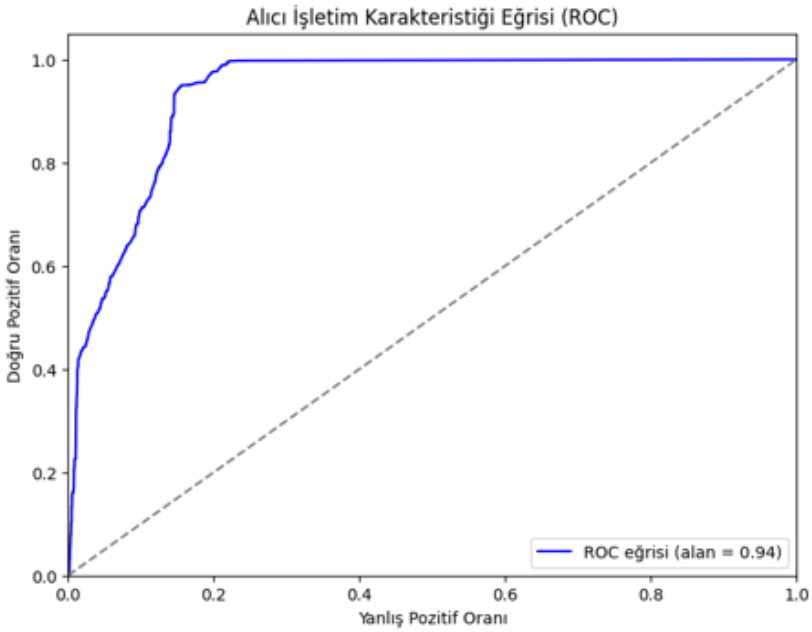
Şekil 5.2'de sol tarafta yer alan doğruluk grafiği, eğitim ve doğrulama verileri ile modelin doğruluğunu göstermektedir. Grafikte, modelin doğruluk oranının hızla arttığı ve 20. epoch sonrası neredeyse sabit kaldığı görülmektedir. Hem eğitim hem de doğrulama verisi doğruluklarının birbirine yakın olması, modelin aşırı öğrenme (overfitting) yapmadığını ve genelleme kabiliyetinin iyi olduğunu göstermektedir.

Şekil 5.2'de sağdaki kayıp grafiği ise modelin eğitim ve doğrulama verisindeki hata oranının giderek azaldığını göstermektedir. Hem eğitim hem de doğrulama verileri için kayıplar hızlı bir şekilde düşmekte ve 20 epoch sonrası sabit bir noktaya yaklaşmaktadır.



Şekil 5.3: Karışıklık matrisi – Deney 1.

Şekil 5.3'te görülen karışıklık matrisi, modelin gerçek etiketlerle tahmin edilen etiketler arasındaki ilişkiyi göstermektedir. Model, 47381 doğru negatif ve 54784 doğru pozitif tahmin yapmıştır. Yanlış negatif sayısı 2640, yanlış pozitif sayısı ise 10043 olmuştur. Bu durum, modelin daha fazla pozitif örneği doğru tahmin ettiğini ancak negatif sınıfta bir miktar hata yaptığını göstermektedir. Yanlış pozitif oranı nispeten yüksek görünse de genel başarımının kabul edilebilir seviyede olduğu düşünülebilir.



Şekil 5.4: ROC eğrisi grafiği – Deney 1.

Farklı eşik değerlerinde duyarlılık (doğru pozitif oranı) ve yanlış pozitif oranı arasındaki ilişkiyi veren Şekil 5.4'teki ROC eğrisi, modelin karakteristiğini göstermektedir. Eğrinin altında kalan alan (AUC) 0.94 olarak hesaplanmıştır, bu da modelin genel sınıflandırma başarımı açısından kabul edilebilir olduğunu, diğer bir ifadeyle pozitif ve negatif sınıfları ayırmada başarılı olduğunu göstermektedir.

Tablo 5.1'de verilen doğrulama sonuçları, 25 epoch boyunca yapılan model eğitiminin 10 katmanlı çapraz doğrulama ile değerlendirilmesi sonucunda elde edilmiştir. Bu deneyde, 25 epoch boyunca yapılan eğitim sonucunda modelin ortalama doğruluk oranı %88.55 olarak elde edilmiştir, bu da modelin sınıflandırma yeteneğinin yüksek olduğunu göstermektedir. Keskinlik ortalaması %81.93 olup, yanlış pozitifleri azaltmada makul bir başarı gösterse de, standart sapmanın biraz daha yüksek olması (0.0157), bazı çapraz doğrulama katmanlarında keskinlik oranının dalgalandığını işaret etmektedir. Buna karşılık, duyarlılık metriği ortalama %99.01 ile en yüksek performansı sergilemiş ve modelin pozitif sınıfları tanıma başarısının tutarlı olduğunu göstermiştir. F1 skoru ortalama %89.64 ile modelin dengeli bir başarım sunabildiğini, hem keskinlik hem de duyarlılık arasında iyi bir denge kurabildiğini göstermektedir. Standart sapmaların genellikle düşük olması, modelin performansındaki istikrarı ve çapraz doğrulama katmanları arasında tutarlı sonuçlar verdiğini ortaya koymaktadır.

Tablo 5.1: Çapraz doğrulama sonuçları – Deney 1.

Dosya	Doğruluk	Keskinlik	Duyarlılık	F1 Skoru
1	0.880646	0.813437	0.987853	0.892201
2	0.891225	0.824264	0.994471	0.901403
3	0.881430	0.810476	0.995690	0.893586
4	0.886719	0.817017	0.996648	0.897937
5	0.895339	0.857205	0.948713	0.900641
6	0.885326	0.817172	0.992773	0.896454
7	0.886981	0.819197	0.993165	0.897831
8	0.893750	0.825869	0.997910	0.903775
9	0.887830	0.817696	0.998215	0.898983
10	0.865713	0.790343	0.995516	0.881144
Ortalama	**0.885496**	**0.819267**	**0.990095**	**0.896395**
Standart Sapma	**0.007990**	**0.015709**	**0.014085**	**0.006056**

Bu çalışmada, Alani ve ark. [7] tarafından kullanılan model mimarisi ve hiper parametreler korunarak 25 epoch boyunca model eğitimi gerçekleştirilmiştir. Ancak, veri seti üzerinde yapılan işlemler ve veri boyutlarındaki farklılıklar nedeniyle elde ettiğimiz sonuçlar, Alani ve ark. çalışmasında raporlanan sonuçlarla kıyaslandığında daha düşük başarım göstermiştir. Bahsedilen çalışmada, ARP sahteciliği analizinde kullanılan IoT-ID [8] veri setinden sadece ARP sahteciliği dosyalarını ayırırken, belirsiz bir şekilde veri seti üzerinde dosya eksiltme veya filtreleme işlemleri gerçekleştirdiği görülmüştür. Bu işlemler sonucunda, veri seti boyutu azaltılarak daha az saldırı ve iyi huylu trafik kullanılmıştır. Ayrıca, (bu çalışmada da yapıldığı gibi) veri setinde yer alan saldırı ve normal trafik örnekleri arasındaki dengesizlikler rastgele örnekleme (random oversampling) yöntemi ile düzeltilmiştir. Bu ön işleme adımları ve veri setindeki eksiltme işlemlerinin, modelin doğruluk (%99.9), keskinlik (%99.9), duyarlılık (%99.9) ve F1 skoru (%99.9) gibi başarım metriklerinde yüksek sonuçlar elde edilmesine katkıda bulunmuş olabileceği düşünülmektedir.

Buna karşın, çalışmamızda, tüm ARP sahteciliği verileri eksiksiz olarak kullanılmış ve veri seti üzerinde herhangi bir eksiltme yapılmamıştır. Bu nedenle, modelin 25 epoch boyunca eğitimi sonucunda doğruluk %88.96, keskinlik %84.51, duyarlılık %95.40 ve F1 skoru %89.63 olarak elde edilmiştir. Bu sonuçlar, tüm ARP sahteciliği verilerinin kullanılması ve

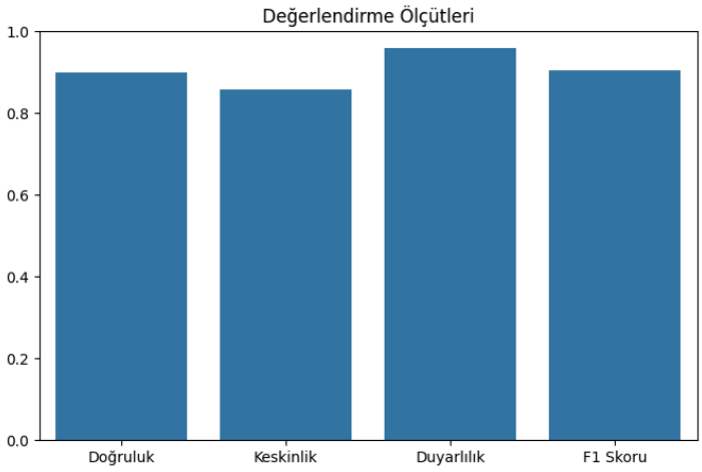
veri setinin eksiltme yapılmaksızın değerlendirilmesi sonucunda, gerçek veri dağılımına dayalı olarak daha makul ve kapsamlı bir performans ölçümü sunmaktadır.

Sonuç olarak, Alani ve ark. tarafından gerçekleştirilen çalışmada veri seti üzerinde yapılan eksiltme işlemlerinin, modelin yüksek performans metrikleri elde etmesine yol açmış olabileceği düşünülmektedir. Bu tezde ise tüm veriler kullanılarak modelin performansı değerlendirilmiş ve gerçek veri dağılımı üzerinden daha dengeli sonuçlar elde edilmiştir.

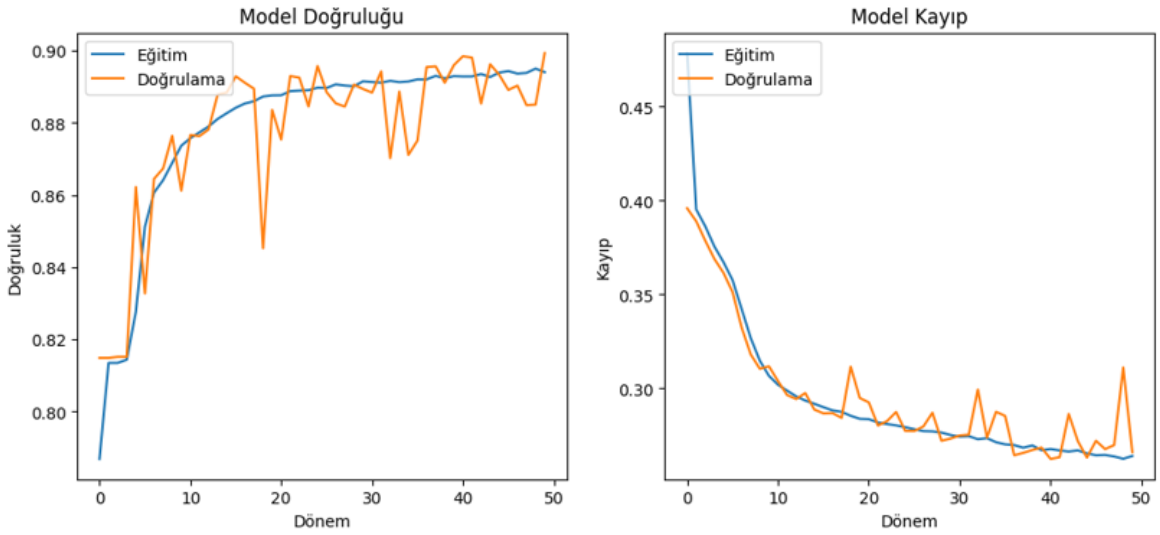
5.1.2 Deney 2 – 50 Epoch

Bu deneyde, model 50 epoch boyunca eğitilmiş ve sonuçlar aşağıdaki gibi elde edilmiştir: doğruluk %89.93, keskinlik %85.70, duyarlılık %95.86 ve F1 skoru %90.50. Model, özellikle duyarlılık açısından oldukça yüksek bir başarım değeri elde etmiş ve pozitif sınıfları doğru bir şekilde tanımlamada başarılı olmuştur. Keskinlik değeri ise %85.70 ile kabul edilebilir düzeydedir ve modelin yanlış pozitifleri nispeten kontrol edebildiği düşünülebilir. F1 skoru %90.50 ile modelin dengeli bir performans gösterdiğini ortaya koymaktadır.

Şekil 5.5'de görsel olarak sunulan bu değerlendirme ölçütleri, modelin doğruluk, keskinlik, duyarlılık ve F1 skoru metrikleri arasındaki dengeyi ortaya koymaktadır. Model, duyarlılıkta en yüksek performansı sergilemiş ve pozitif örneklerin büyük çoğunluğunu doğru bir şekilde sınıflandırmıştır. Keskinlik ve F1 skoru da %85'in üzerinde olup, genel olarak modelin dengeyi sağladığı ve hem pozitif hem de negatif sınıfları başarılı bir şekilde ayırt ettiği görülebilir.



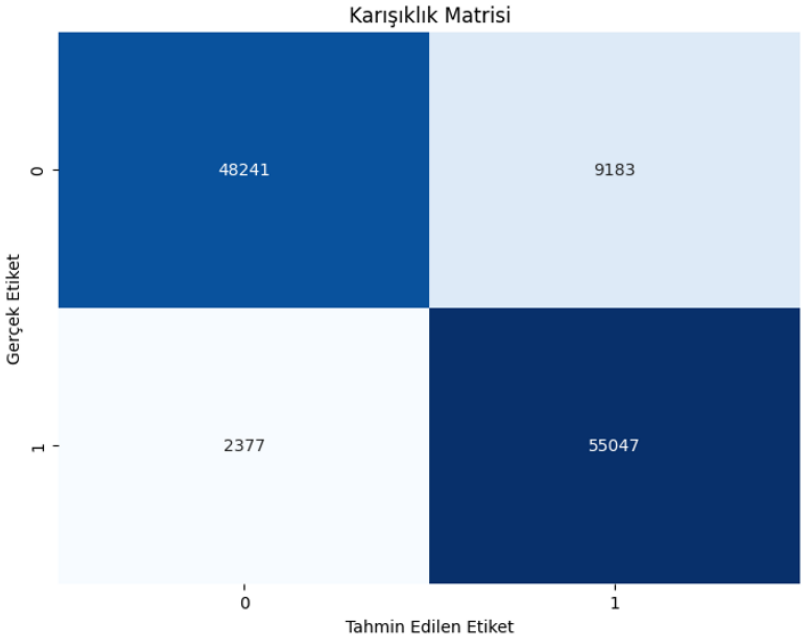
Şekil 5.5: Ölçüm metrikleri grafiği – Deney 2.



Şekil 5.6: Model doğruluk ve kayıp grafiği – Deney 2.

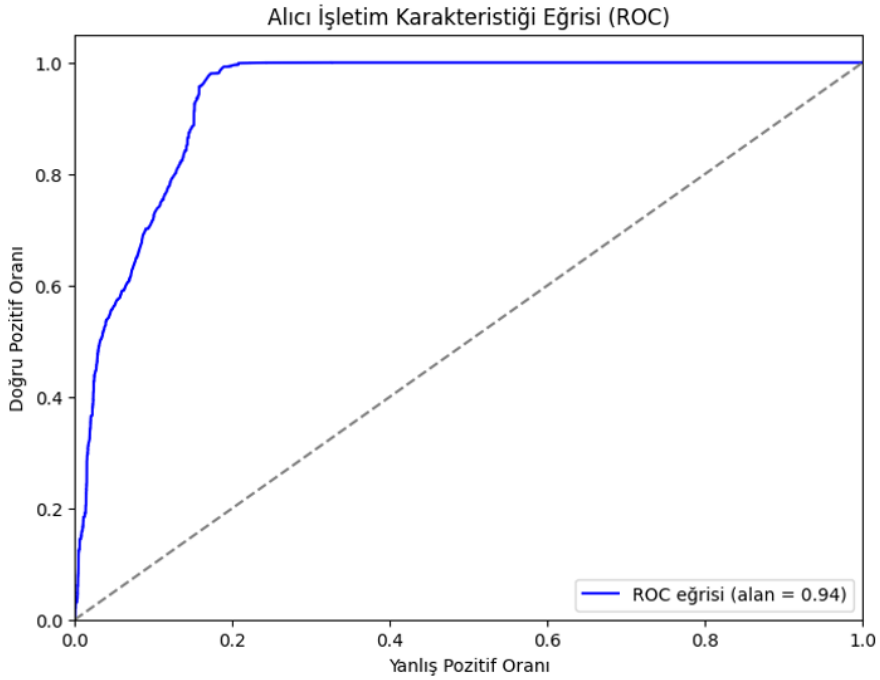
Modelin doğruluk ve kayıp grafikleri, Şekil 5.6'te gösterildiği gibi, eğitim ve doğrulama setleri üzerindeki doğruluk ve kayıp eğilimlerini yansıtmaktadır. Doğruluk grafiği, modelin doğruluk oranlarının eğitim verileri üzerinde %90 seviyelerine ulaştığını, doğrulama verilerinde ise benzer bir doğruluk oranının korunduğunu göstermektedir. Kayıp grafiği ise modelin hem eğitim hem de doğrulama verilerindeki kayıpların zamanla düştüğünü, özellikle 30. epoch sonrası nispeten dengeli bir eğilim gösterdiğini işaret etmektedir. Bu durum, modelin daha fazla epoch ile aşırı öğrenme yapmadan daha iyi öğrenmeye devam ettiğini göstermektedir.

Karışıklık matrisi ise Şekil 5.7'de gösterildiği gibi gerçek ve model tarafından tahmin edilen etiketler arasındaki ilişkiyi özetlemektedir. Model, 48241 doğru negatif, 55047 doğru pozitif tahmin yaparken, yanlış negatif sayısı 2377, yanlış pozitif sayısı ise 9183 olarak kaydedilmiştir. Bu sonuçlar, 25 epoch ile elde ettiğimiz sonuçlarla karşılaştırıldığında, modelin yanlış pozitiflerin sayısını azaltmayı başardığını ve pozitif sınıfları daha iyi ayırabildiğini göstermektedir. Ayrıca, modelin doğru tahmin oranlarını artırarak daha güçlü bir sınıflandırma yeteneği kazandığı görülmektedir.



Şekil 5.7: Karışıklık matrisi – Deney 2.

Şekil 5.8'de verilen ROC eğrisinden, modelin duyarlılık ve yanlış pozitif oranı arasındaki ilişki görülmektedir. Eğrinin altında kalan alan (AUC) 0.94 olarak hesaplanmış olup, bu değer modelin sınıflandırma yeteneğinin yüksek olduğunu ve pozitif sınıfları negatiflerden ayırmada oldukça başarılı olduğunu göstermektedir.



Şekil 5.8: ROC eğrisi grafiği – Deney 2.

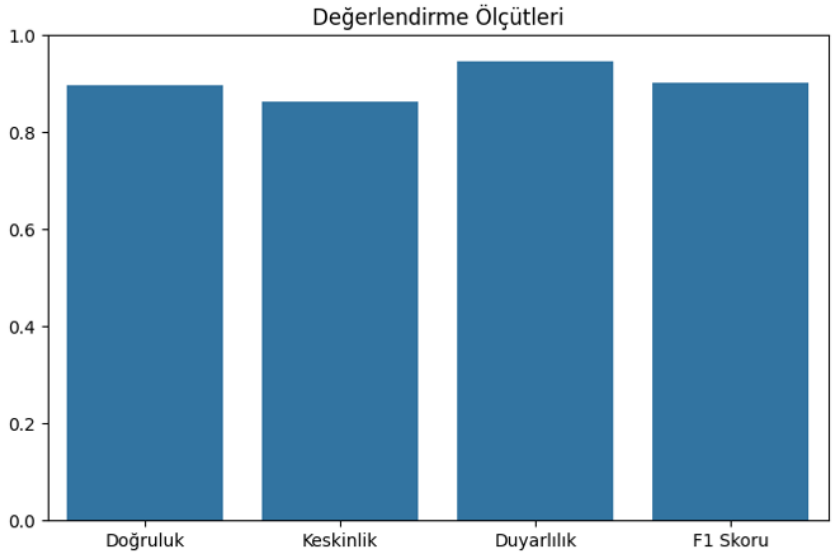
Tablo 5.2: Çapraz doğrulama sonuçları – Deney 2.

Dosya	Doğruluk	Keskinlik	Duyarlılık	F1 Skoru
1	0.895296	0.853329	0.954678	0.901163
2	0.904373	0.840694	0.997823	0.912544
3	0.880668	0.808554	0.997518	0.893151
4	0.897473	0.855054	0.957203	0.903250
5	0.886654	0.838298	0.958117	0.894212
6	0.885174	0.813530	0.999434	0.896951
7	0.894469	0.829755	0.992599	0.903901
8	0.899279	0.833740	0.997475	0.908287
9	0.893076	0.827090	0.993949	0.902875
10	0.896232	0.832694	0.991728	0.905279
Ortalama	**0.893269**	**0.833274**	**0.984053**	**0.902161**
Standart Sapma	**0.006779**	**0.014174**	**0.018094**	**0.005765**

Tablo 5.2'de sunulan deney 2'nin 10 katmanlı çapraz doğrulama sonuçlarına göre, modelin ortalama doğruluk oranı %89.33 olup, doğruluk değerleri %88.07 ile %90.44 arasında değişmiştir. Bu, modelin tutarlı bir performans sergilediğini göstermektedir. Keskinlik ortalaması %83.33 olarak kaydedilmiş ve %80.85 ile %85.50 arasında değişim göstermiştir. Bu durum, bazı katmanlarda yanlış pozitiflerin daha fazla olmasından kaynaklanmaktadır. Duyarlılık, %98.41 ortalama ile modelin en güçlü başarımını elde eden metrik olmuştur ve pozitif sınıfların doğru tespit edilmesinde başarılı sonuçlar vermiştir. F1 skoru ortalaması ise %90.22 olup, modelin hem duyarlılık hem de keskinlik arasında dengeli bir performans sunduğunu göstermektedir. Standart sapmaların düşük olması, modelin çapraz doğrulama süreci boyunca tutarlı ve dengeli sonuçlar verdiğini ortaya koymaktadır.

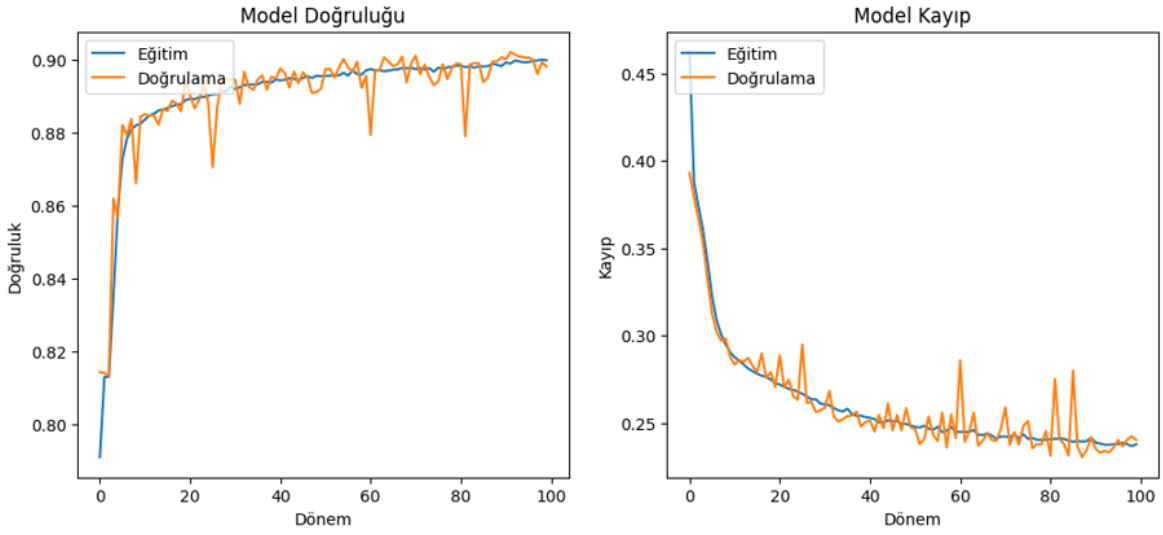
5.1.3 Deney 3 – 100 Epoch

Deney 3'te, model 100 epoch boyunca eğitilmiş ve genel başarım metrikleri aşağıdaki gibi hesaplanmıştır. Modelin doğruluğu %89.81 ile yüksek bir sınıflandırma başarısı göstermekte, keskinliği ise %86.33 ile doğru pozitifleri ayırt etmede etkili olduğunu göstermektedir. %94.59 olarak bulunan duyarlılık, modelin pozitif sınıfları başarılı bir şekilde tanımladığını, %90.27 olarak bulunan F1 skoru ise,umodelin dengeli bir başarım sunduğunu göstermektedir.



Şekil 5.9: Ölçüm metrikleri grafiği – Deney 3.

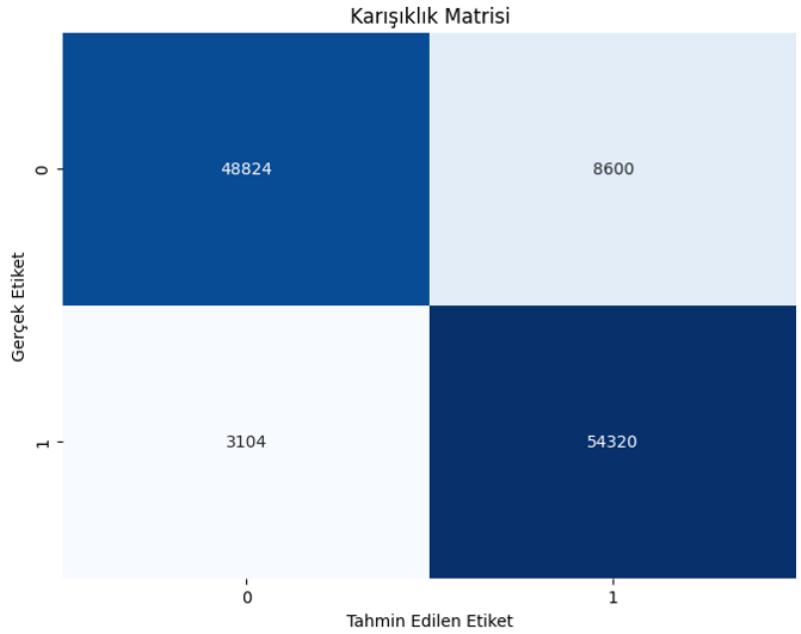
Şekil 5.9'de görüldüğü üzere, modelin doğruluk, keskinlik, duyarlılık ve F1 skoru değerleri arasında, duyarlılık en yüksek değer olarak öne çıkmaktadır. Bu durum, modelin pozitif sınıfları doğru tespit etmede oldukça başarılı olduğunu gösterirken, keskinlik oranının biraz daha düşük olması, yanlış pozitif sınıflandırmaların varlığını işaret etmektedir. Ancak F1 skoru, keskinlik ve duyarlılık arasında dengeli bir performansı yansıtmaktadır.



Şekil 5.10: Model doğruluk ve kayıp grafiği – Deney 3.

Şekil 5.10'da görüldüğü üzere, modelin doğruluk grafiği, eğitim ve doğrulama setlerinde %90 seviyelerine ulaşmış ve model genel olarak tutarlı bir doğruluk performansı sergilemiştir. Doğrulama setinde gözlemlenen küçük dalgalanmalar, modelin sınıflandırma

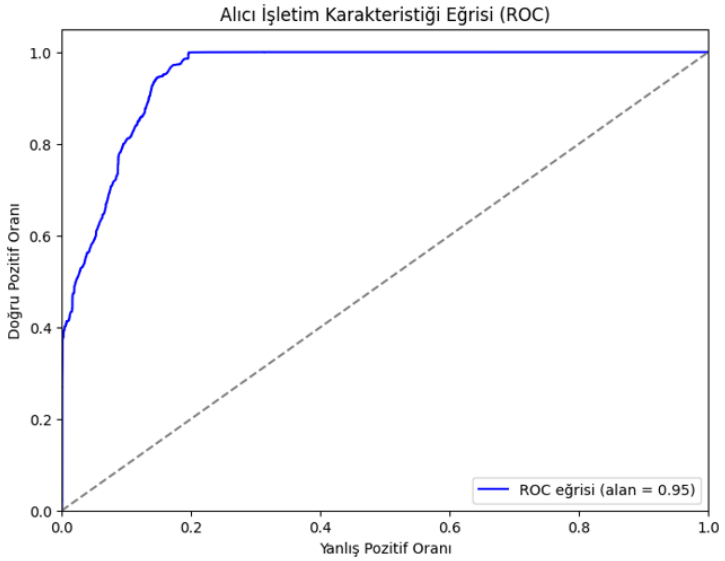
yeteneğini yansıtmakta ve aşırı öğrenme yapmadığını göstermektedir. Aynı şekilde, kayıp grafiğinde de hem eğitim hem de doğrulama verilerinde kayıpların düzenli olarak azaldığı görülmektedir. Bu durum, modelin 100 epoch boyunca dengeli bir şekilde öğrenmeye devam ettiğini ve eğitim sürecinde herhangi bir aşırı öğrenme belirtisi göstermediğini ortaya koymaktadır.



Şekil 5.11: Karışıklık matrisi – Deney 3.

Şekil 5.11'de görüldüğü üzere, karışıklık matrisi, modelin doğru ve yanlış sınıflandırmalarını özetlemektedir. Model, 48824 doğru negatif ve 54320 doğru pozitif tahmin yaparken, yanlış negatif sayısı 3104, yanlış pozitif sayısı ise 8600 olarak kaydedilmiştir. Bu sonuçlar, modelin pozitif sınıfları doğru tanımada oldukça başarılı olduğunu, ancak negatif sınıfları ayırt etmede bir miktar hata yaptığını göstermektedir.

Şekil 5.12'de görüldüğü üzere, ROC eğrisi, modelin duyarlılık ve yanlış pozitif oranı arasındaki ilişkiyi açıkça göstermektedir. Eğrinin altında kalan alan (AUC) 0.95 olarak hesaplanmış olup, bu değer modelin pozitif ve negatif sınıfları ayırt etmede yüksek bir başarıya sahip olduğunu ortaya koymaktadır. AUC'nin yüksek olması, modelin genel sınıflandırma performansının güçlü olduğunu vurgulamaktadır.



Şekil 5.12: ROC eğrisi grafiği – Deney 3.

Tablo 5.3'de sunulan çapraz doğrulama sonuçlarına göre, modelin ortalama doğruluk oranı %89.48, keskinlik %83.54, duyarlılık %98.38 ve F1 skoru %90.33 olarak hesaplanmıştır. Standart sapmaların düşük olması, modelin her bir çapraz doğrulama katmanında tutarlı bir performans sergilediğini göstermektedir. Özellikle duyarlılık metriğinin oldukça yüksek olmasından, modelin pozitif sınıfları doğru tespit etmede etkili olduğu görülmektedir.

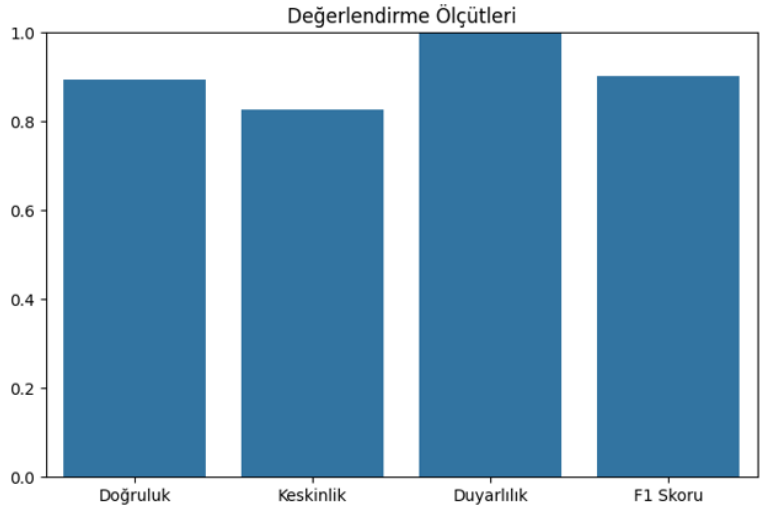
Tablo 5.3: Çapraz doğrulama sonuçları – Deney 3.

Dosya	**Doğruluk**	**Keskinlik**	**Duyarlılık**	**F1 Skoru**
1	0.893598	0.825983	0.997301	0.903594
2	0.898431	0.834350	0.994253	0.907310
3	0.878012	0.845270	0.925421	0.883531
4	0.892510	0.826850	0.992947	0.902318
5	0.897103	0.833102	0.993165	0.906119
6	0.895949	0.834056	0.988594	0.904773
7	0.897777	0.834615	0.992164	0.906596
8	0.902000	0.860253	0.959948	0.907370
9	0.897495	0.831115	0.997736	0.906835
10	0.895122	0.828745	0.996082	0.904741
Ortalama	**0.894800**	**0.835434**	**0.983761**	**0.903319**
Standart Sapma	**0.006135**	**0.009738**	**0.022086**	**0.006783**

5.1.4 Deney 4 – 200 Epoch

Deney 4'te, model 200 epoch boyunca eğitilmiş ve genel performans metrikleri aşağıdaki gibi elde edilmiştir. Modelin doğruluğu %89.32 ile oldukça iyi bir sınıflandırma yeteneği göstermekte, buna karşın keskinliği %82.55 ile doğru pozitifleri ayırt etmede belirli bir başarı sergilemesine rağmen, yanlış pozitiflerin sayısını azaltmak için daha fazla iyileştirmeye ihtiyaç duyulabileceği düşünülebilir. Duyarlılık %99.72 olup, modelin pozitif sınıfları çok yüksek bir doğrulukla tanımladığını ortaya koymaktadır. F1 skoru ise %90.33 olarak hesaplanmış olup, bu sonuçlar modelin dengeli ve genel olarak iyi bir başarıma sahip olduğunu göstermektedir.

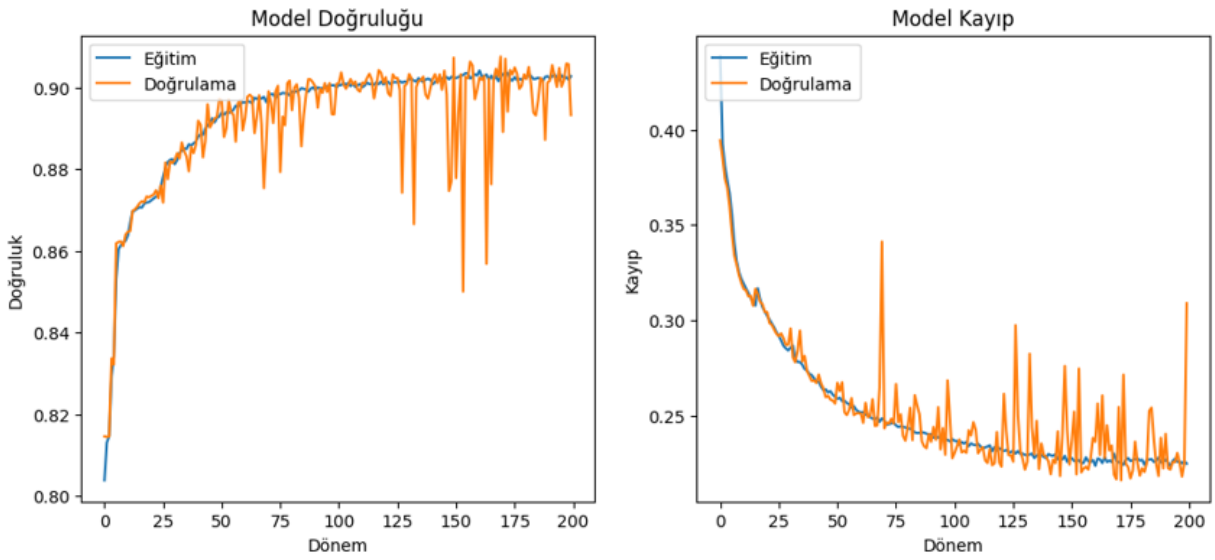
Şekil 5.13'de görüldüğü üzere, modelin doğruluk, keskinlik, duyarlılık ve F1 skoru arasında duyarlılık en yüksek değeri almıştır. Bu durum, modelin pozitif sınıfları doğru tespit etme konusunda oldukça başarılı olduğunu gösterirken, keskinlik oranı daha düşük seviyede kalarak nispeten yüksek yanlış pozitiflerin varlığını işaret etmektedir. F1 skoru, modelin dengeli bir başarım sergilediğini ve hem duyarlılık hem de keskinlik arasında iyi bir denge kurduğunu ortaya koymaktadır.



Şekil 5.13: Ölçüm metrikleri grafiği – Deney 4.

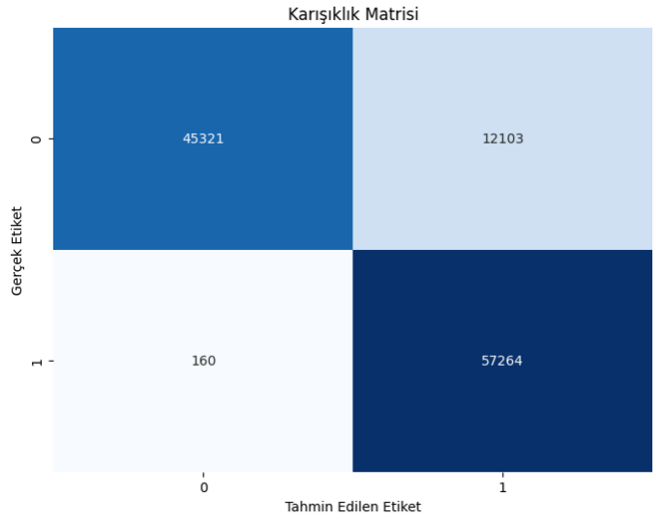
Şekil 5.14'de görüldüğü üzere, modelin doğruluk grafiği, önceki epoch sayılarında bulunan sonuçlara benzer olarak, eğitim ve doğrulama setlerinde %90 seviyelerine ulaşmış olup, doğrulama doğruluğunda küçük dalgalanmalar gözlemlenmektedir. Bununla birlikte, bu dalgalanmalar modelin sınıflandırma yeteneğinin arttığını ve aşırı öğrenme yapmadığını işaret etmektedir. Kayıp grafiğinde ise, eğitim ve doğrulama setlerinde kayıpların giderek

azaldığı, ancak doğrulama setinde daha fazla dalgalanma olduğu görülmektedir. Bu ise, farklı epoch sayıları ile gerçekleşen öğrenme sürecinde elde edilen modelin, zaman zaman doğrulama verisi üzerindeki başarısının değişkenlik gösterdiğini işaret eder.



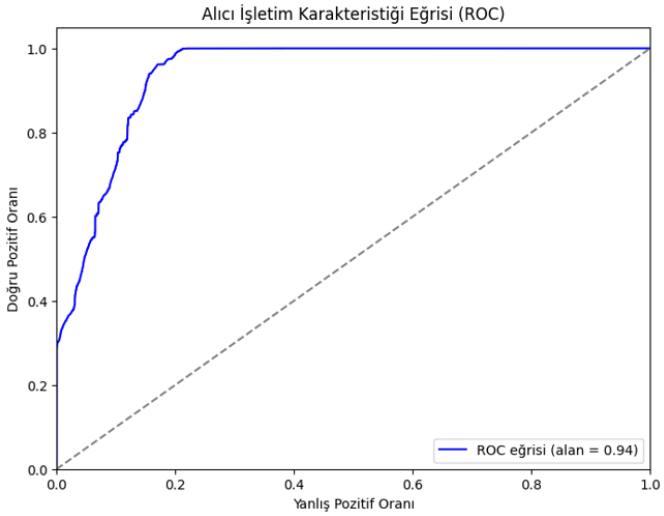
Şekil 5.14: Model doğruluk ve kayıp grafiği – Deney 4.

Şekil 5.15'de görüldüğü üzere, karışıklık matrisi, modelin doğru ve yanlış sınıflandırmalarını özetlemektedir. Model, 45321 doğru negatif ve 57264 doğru pozitif tahmin yaparken, yanlış negatif sayısı 160, yanlış pozitif sayısı ise 12103 olarak kaydedilmiştir. Bu sonuçlar, modelin pozitif sınıfları başarılı bir şekilde ayırt ettiğini, ancak negatif sınıflarda nispeten yüksek sayıda yanlış sınıflandırma yaptığını göstermektedir.



Şekil 5.15: Karışıklık matrisi – Deney 4.

Şekil 5.16'de görüldüğü üzere, ROC eğrisi modelin duyarlılık ve yanlış pozitif oranı arasındaki ilişkiyi açıkça göstermektedir. Eğrinin altında kalan alan (AUC) 0.94 olarak hesaplanmış olup, bu değer modelin sınıflandırma performansının oldukça iyi olduğunu ve pozitif sınıfları negatiflerden ayırmada yüksek başarı sağladığını göstermektedir.



Şekil 5.16: ROC eğrisi grafiği – Deney 4.

Tablo 5.4'de verilen çapraz doğrulama sonuçlarına göre, modelin ortalama doğruluk oranı %90.04, keskinlik %84.57, duyarlılık %98.02 ve F1 skoru %90.78 olarak hesaplanmıştır. Standart sapmaların düşük olması, modelin her bir çapraz doğrulama katmanında tutarlı başarım gösterdiğini ortaya koymaktadır. Özellikle duyarlılık metriği, modelin pozitif sınıfları doğru tespit etmede başarılı olduğunu göstermektedir.

Tablo 5.4: Çapraz doğrulama sonuçları – Deney 4.

Dosya	**Doğruluk**	**Keskinlik**	**Duyarlılık**	**F1 Skoru**
1	0.898866	0.859768	0.953198	0.904076
2	0.902327	0.839318	0.995167	0.910623
3	0.898714	0.839185	0.986460	0.906882
4	0.897777	0.831935	0.996952	0.906999
5	0.894491	0.826381	0.998825	0.904457
6	0.902980	0.840475	0.994776	0.911139
7	0.903372	0.839607	0.997257	0.911667
8	0.901587	0.858386	0.961863	0.907183
9	0.906311	0.868369	0.957815	0.910901
10	0.897821	0.854268	0.959295	0.903740
Ortalama	**0.900424**	**0.845769**	**0.980161**	**0.907767**
Standart Sapma	**0.003304**	**0.012870**	**0.018433**	**0.002951**

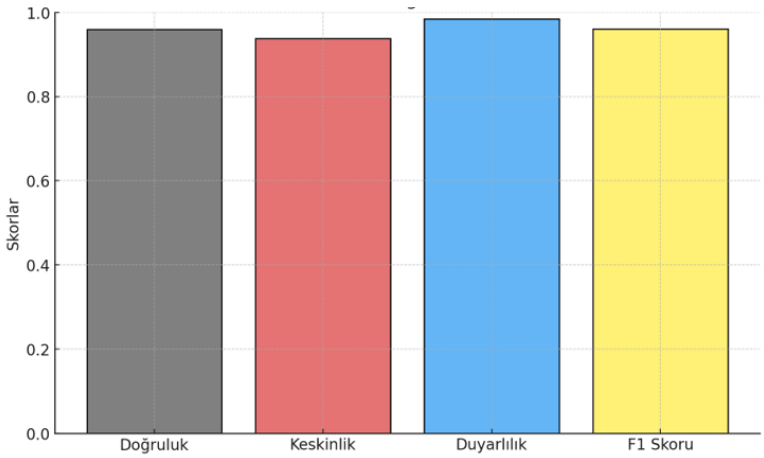
5.2 Geleneksel Makine Öğrenmesi Yöntemlerinin Deney Sonuçları

Bu bölümde, çeşitli geleneksel makine öğrenmesi yöntemlerinin ARP sahteciliği tespitine yönelik başarımları değerlendirilecektir. DNN tabanlı yaklaşımların yanı sıra, karar ağacı, rastgele orman, KNN ve lojistik regresyon gibi geleneksel yöntemler de bu çalışmada test edilmiştir. Her yöntemin doğruluk, keskinlik, duyarlılık ve F1 skoru gibi temel metriklere göre başarımı analiz edilecektir. Bu bölümdeki deneyler, yöntemler arası karşılaştırma yaparak hangi yaklaşımın ARP sahteciliği tespitinde daha başarılı olduğunu ortaya koymayı amaçlamaktadır.

5.2.1 Karar Ağacı

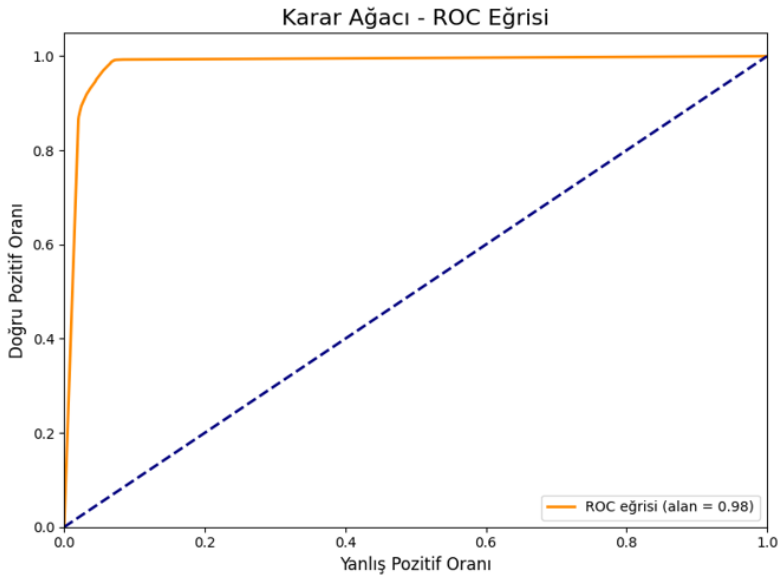
Karar Ağacı yönteminin başarım değerlendirme metriklerine göre model, hem doğruluk hem de diğer metrikler açısından yüksek bir başarı sergilemiştir. Doğruluk oranı %95.94 ile oldukça yüksek olup, modelin sınıflandırma yeteneğini göstermektedir. Keskinlik %93.77 olup, modelin yanlış pozitif oranını düşük seviyelerde tuttuğunu ve pozitif sınıfları iyi derecede ayırt edebildiğini göstermektedir. Duyarlılık ise %98.41 olarak hesaplanmış olup, modelin pozitif sınıfları yüksek oranda doğru tespit edebildiğini göstermektedir. F1 Skoru ise %96.03 ile keskinlik ve duyarlılık arasında oldukça başarılı bir denge elde edilebildiğini göstermiştir.

Şekil 5.17'de, başarım metrikleri olan doğruluk, keskinlik, duyarlılık ve F1 skoru değerleri grafiksel olarak görülmektedir. Modelin duyarlılık ve F1 skoru oldukça yüksek olup, pozitif sınıfları başarılı şekilde tespit edebildiği ve genel başarısının dengeli olduğu açıkça görülmektedir.



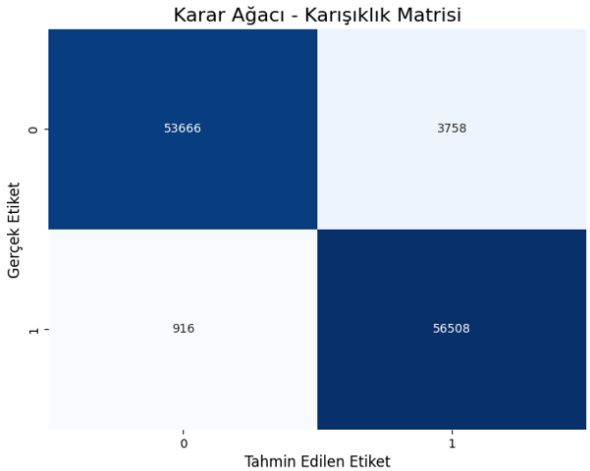
Şekil 5.17: Ölçüm metrikleri grafiği – Karar Ağacı.

Şekil 5.18'de görüldüğü üzere, ROC eğrisi, modelin duyarlılık ve yanlış pozitif oranı arasındaki ilişkiyi gösterir. ROC eğrisinin altında kalan alan (AUC) 0.98 olup, bu değer modelin oldukça yüksek sınıflandırma başarısı sergilediğini ve pozitif sınıfları negatiflerden ayırmada başarılı olduğunu göstermektedir.



Şekil 5.18: ROC eğrisi grafiği – Karar Ağacı.

Karışıklık matrisinde Şekil 5.19'de görüldüğü gibi 53683 doğru negatif ve 56510 doğru pozitif sınıflandırma yapılmıştır, 3741 yanlış pozitif ve 914 yanlış negatif tahmin bulunmaktadır. Bu sonuçlar, modelin yanlış negatif oranının çok düşük olduğunu ve doğru pozitifleri yüksek başarıyla tespit edebildiğini göstermektedir.

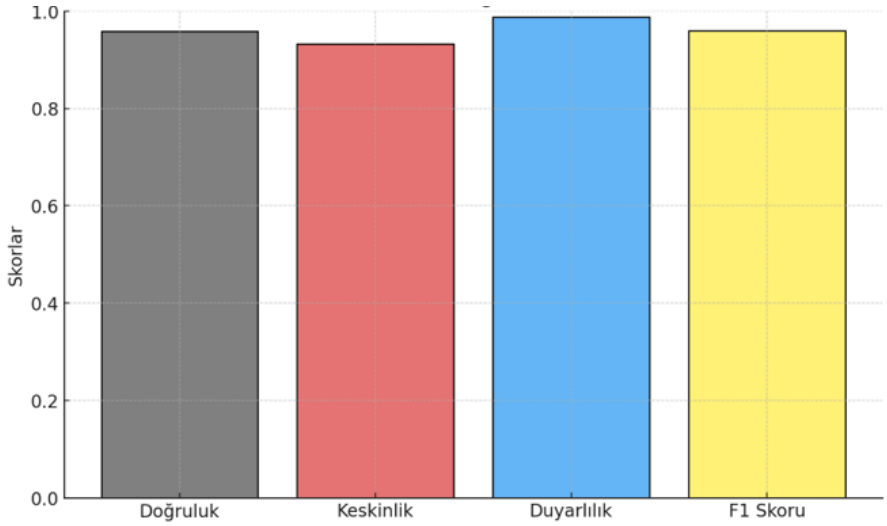


Şekil 5.19: Karışıklık matrisi – Karar Ağacı.

5.2.2 Rastgele Orman

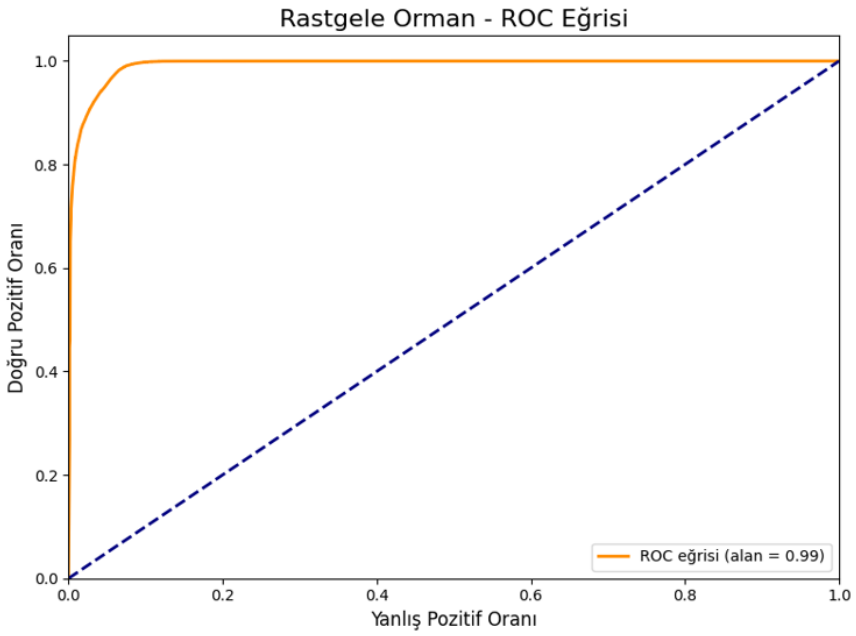
Rastgele Orman yöntemi, yüksek doğruluk oranıyla (%95.82) sınıflandırma yeteneğini başarılı bir şekilde ortaya koymaktadır. Keskinlik metriği %93.24 olarak ölçülmüş ve modelin yanlış pozitifleri iyi bir şekilde kontrol altında tutabildiği görülmüştür. Duyarlılık metriği ise %98.81 ile modelin pozitif sınıfları doğru tespit etme oranının oldukça yüksek olduğunu göstermektedir. F1 Skoru %95.95 olup, modelin keskinlik ve duyarlılık arasında başarılı bir denge sağladığını ortaya koymaktadır.

Şekil 5.20'de, modelin doğruluk, keskinlik, duyarlılık ve F1 skoru değerleri grafiksel olarak görülmektedir. Duyarlılık ve F1 skoru en yüksek değerlere sahiptir. Bu, modelin pozitif sınıfları başarılı şekilde tespit edebildiğini ve genel başarısının dengeli olduğunu göstermektedir.



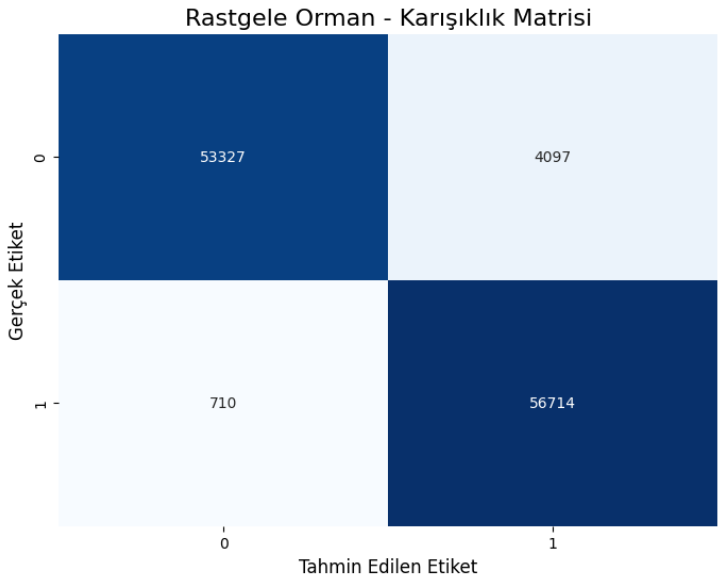
Şekil 5.20: Ölçüm metrikleri grafiği – Rastgele Orman.

Şekil 5.21'de verilen ROC eğrisinden, modelin duyarlılık ve yanlış pozitif oranı arasındaki ilişki görülmektedir. ROC eğrisi altında kalan alan (AUC) 0.99 olup, bu sonuç modelin sınıflandırma başarısının oldukça yüksek olduğunu ve pozitif sınıfları negatiflerden ayırmadaki yeteneğinin başarılı olduğunu göstermektedir.



Şekil 5.21: ROC eğrisi grafiği – Rastgele Orman.

Şekil 5.22'de görüldüğü üzere, karışıklık matrisi, modelin doğru ve yanlış sınıflandırmalarını özetlemektedir. 53327 doğru negatif ve 56714 doğru pozitif sınıflandırma yapılmış, 4097 yanlış pozitif ve 710 yanlış negatif tahmin bulunmaktadır. Bu sonuçlar, modelin pozitif sınıfları yüksek doğrulukla tespit ettiğini ve aynı zamanda negatif sınıfları da büyük oranda doğru tahmin ettiğini göstermektedir.

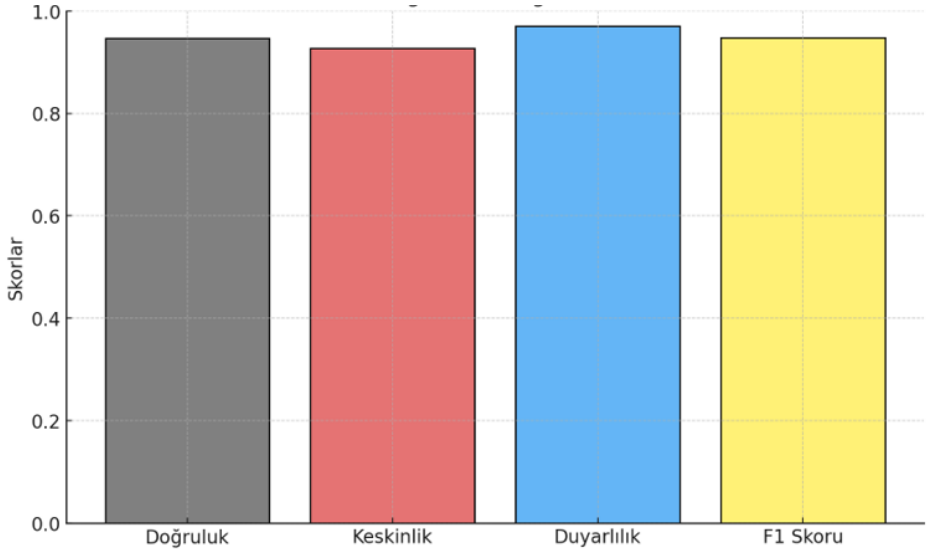


Şekil 5.22: Karışıklık matrisi – Rastgele Orman.

5.2.3 K-En Yakın Komşu

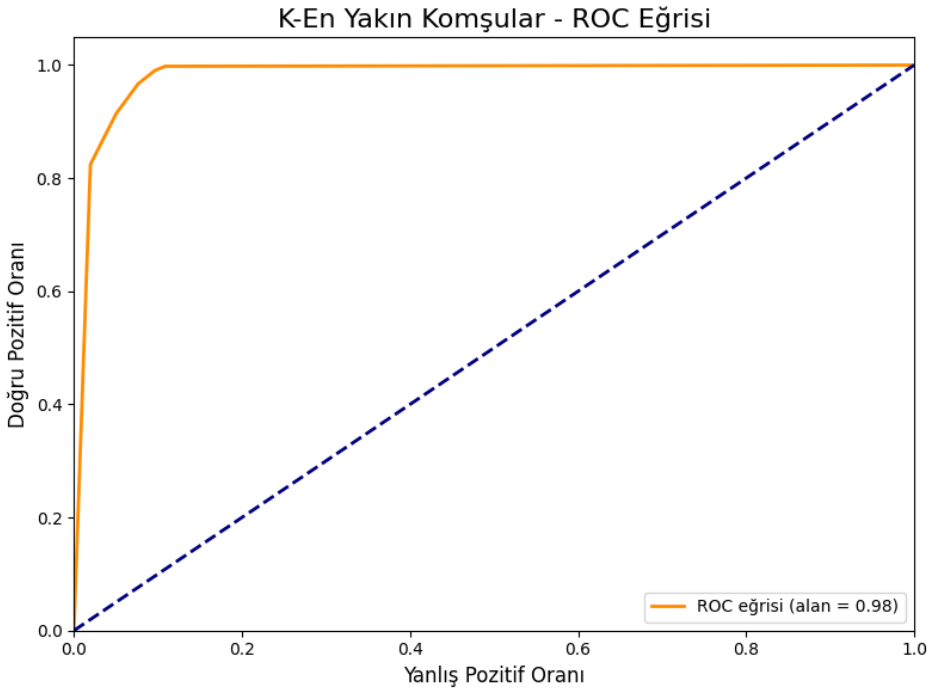
K-En Yakın Komşu yöntemi, %94.70 olarak bulunan doğruluk oranı ile başarılı bir performans sergilemiştir. Keskinlik metriği %92.70 olarak ölçülmüş ve modelin yanlış pozitif oranını nispeten düşük seviyelerde tuttuğu gözlemlenmiştir. Duyarlılık metriği ise %97.06 olup, modelin pozitif sınıfları tespit etme yeteneğinin yüksek olduğunu göstermektedir. %94.83 oranına sahip olarak bulunan F1 Skoru ise, modelin keskinlik ve duyarlılık arasında başarılı bir denge sağlayabildiğini ortaya koymaktadır.

Şekil 5.23'de, KNN yönteminin doğruluk, keskinlik, duyarlılık ve F1 skoru değerleri grafiksel olarak görülmektedir. En yüksek değerlere sahip olarak görülen duyarlılık ve F1 skoru, modelin pozitif sınıfları başarılı şekilde doğru tespit edebildiğini ve dengeli bir sınıflandırma başarımı sunduğunu ortaya koymaktadır.



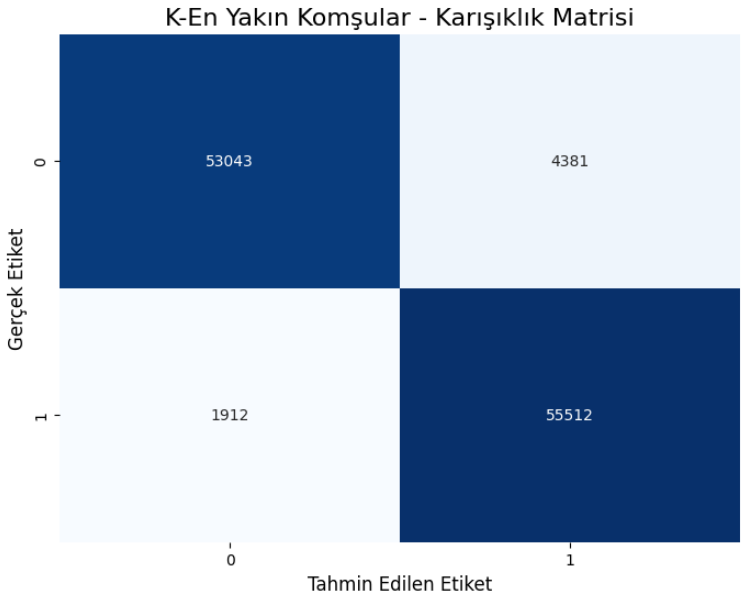
Şekil 5.23: Ölçüm metrikleri grafiği – KNN.

Şekil 5.24'de verilen ROC eğrisi, modelin duyarlılık ve yanlış pozitif oranı arasındaki ilişkiyi göstermektedir. 0.98 olarak hesaplanan AUC değeri, modelin oldukça başarılı bir sınıflandırma yeteneğine sahip olduğunu göstermektedir.



Şekil 5.24: ROC eğrisi grafiği – KNN.

Modelin doğru ve yanlış sınıflandırmalarını özetleyen Şekil 5.25'de verilen karışıklık matrisinden, 53043 doğru negatif ve 55512 doğru pozitif sınıflandırma yapıldığı, buna karşın 4381 yanlış pozitif ve 1912 yanlış negatif tahmin yapıldığı görülmektedir. Bu sonuçlar, modelin pozitif sınıfları yüksek doğrulukla tespit ettiğini ve genel olarak doğru bir sınıflandırma yapabildiğini göstermektedir.

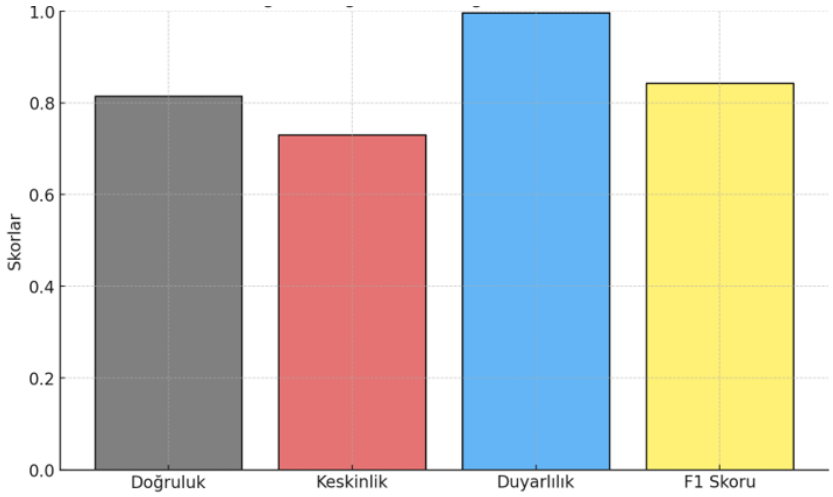


Şekil 5.25: Karışıklık matrisi – KNN.

5.2.4 Lojistik Regresyon

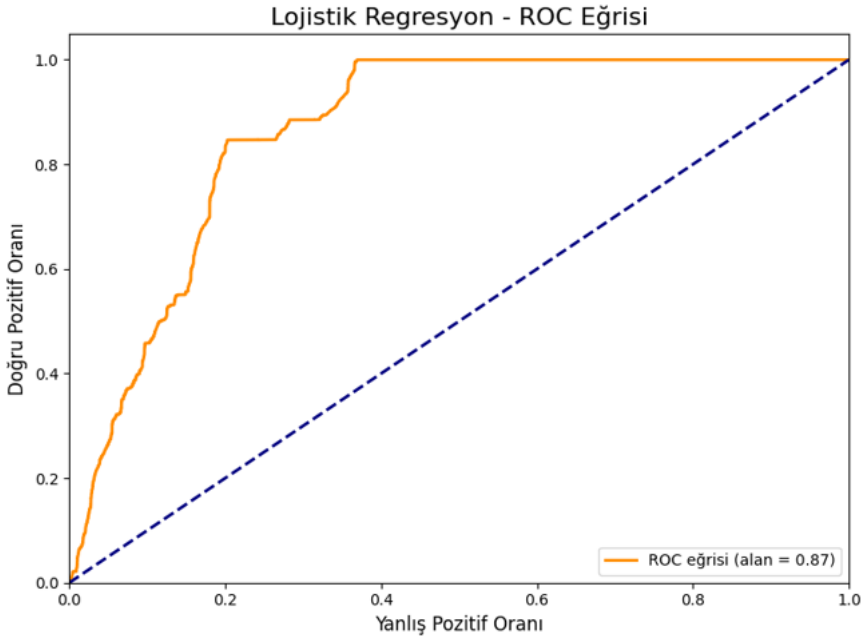
Lojistik Regresyon, diğer yöntemlerle karşılaştırıldığında biraz daha düşük doğruluk oranına (%81.42) sahip olarak bulunmuştur. Keskinlik metriği %73.03 olup, modelin yanlış pozitif oranının düşük bir seviyede bulunamayabileceğini ve böylelikle maliyet artışına sebebiyet verebileceğini göstermektedir. Ancak duyarlılık metriği %99.64 ile oldukça yüksek bir oran olarak elde edilmiş olup, modelin pozitif sınıfları tespit etme yeteneğinin başarılı olduğunu ortaya koymaktadır. %84.28 olarak hesaplan F1 skoru ise, keskinlik ve duyarlılık arasında makul bir denge kurulabildiğini göstermektedir.

Şekil 5.26'de, modelin doğruluk, keskinlik, duyarlılık ve F1 skoru değerleri görsel olarak sunulmuştur. En yüksek değere sahip olan duyarlılık, modelin pozitif sınıfları başarıyla tanımladığını, nispeten düşük kalan keskinlik ise modelin yanlış pozitifleri ayırt etme konusunda daha fazla iyileştirmeye ihtiyaç duyduğunu göstermektedir.



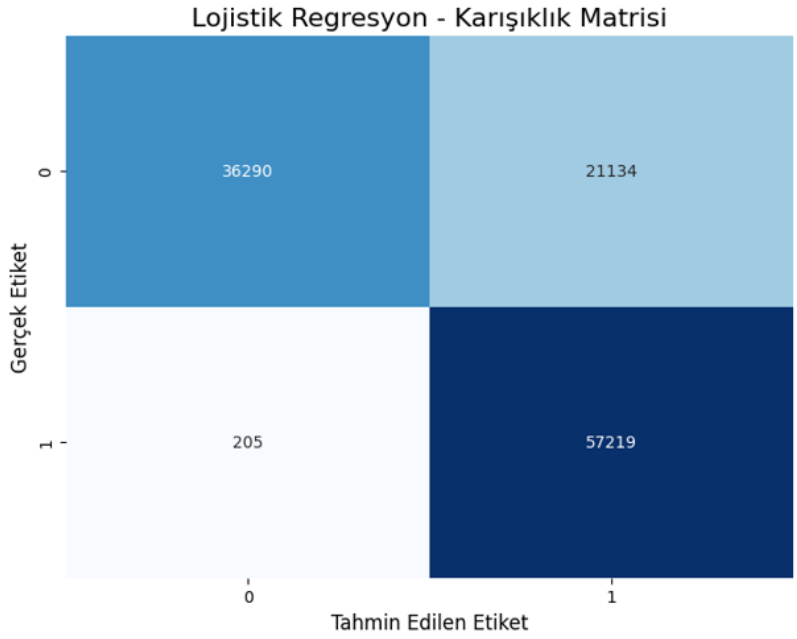
Şekil 5.26: Ölçüm metrikleri grafiği – Lojistik Regresyon.

Şekil 5.27'de görülen ROC eğrisinin altında kalan alan (AUC) %86.52 olarak hesaplanmış olup, modelin pozitif ve negatif sınıfları ayırmada orta düzeyde başarılı olduğu düşünülebilir.



Şekil 5.27: ROC eğrisi grafiği – Lojistik Regresyon.

Şekil 5.28'de verilen karışıklık matrisinden, 36290 doğru negatif ve 57219 doğru pozitif sınıflandırma yapıldığı, 21134 yanlış pozitif ve 205 yanlış negatif tahminde bulunulduğu görülmektedir. Bu sonuçlar, modelin pozitif sınıfları doğru tespit etmede oldukça başarılı olduğunu, buna karşın ancak yanlış pozitiflerin nispeten yüksek olduğunu göstermektedir.



Şekil 5.28: Karışıklık matrisi – Lojistik Regresyon.

6. TARTIŞMA VE SONUÇ

Bu çalışmada, ARP sahteciliği tespiti için hem DNN hem de geleneksel ML algoritmaları kullanılmış ve bu yöntemlerin başarımları karşılaştırılmıştır. Elde edilen sonuçlar, her bir yöntemin güçlü ve zayıf yönlerini ortaya koymuş, ayrıca ARP sahteciliği gibi siber güvenlik tehditlerine karşı hangi yöntemlerin daha etkili olduğu belirlenmiştir.

6.1 DNN Yönteminin Değerlendirilmesi

DNN yöntemi Tablo 6.1'de görüldüğü gibi dört farklı epoch sayısı (25, 50, 100 ve 200 epoch) ile test edilmiştir. Tablo 6.1'de bulunan sonuçlardan, 50 epoch sonrası bu modelin doğruluk ve F1 skoru gibi başarım metriklerinin kararlı hale geldiği ve bu değerlerin daha fazla epoch sayısıyla anlamlı bir şekilde iyileşmediği gözlemlenmiştir. 50 epoch ile elde edilen sonuçlar, en dengeli performansı göstermiştir. Modelin doğruluk ve F1 skoru bakımından en yüksek değerleri (sırasıyla, %89.93 ve %90.50) 50 epoch ile elde ettiği görülmektedir. Daha uzun epoch sayıları (100 ve 200 epoch) ile başarımda küçük iyileşmeler görülmüş olsa da, bu iyileşmeler hesaplama maliyetleri göz önüne alındığında tercih edilebilir değildir. Ayrıca, 200 epoch ile keskinlikte düşüş, diğer bir ifadeyle modelin yanlış pozitif oranının arttığı gözlemlenmiştir.

Tablo 6.1: DNN uygulama sonuçları

Epoch Sayısı	Doğruluk (%)	Keskinlik (%)	Duyarlılık (%)	F1 Skoru (%)	AUC
25 Epoch	88.96	84.51	95.40	89.63	94.00
50 Epoch	89.93	85.70	95.86	90.50	94.00
100 Epoch	89.81	86.33	94.59	90.27	95.00
200 Epoch	89.32	82.55	99.72	90.33	94.00

6.2 Geleneksel ML Yöntemlerinin Değerlendirilmesi

Geleneksel makine öğrenmesi algoritmalarından, Tablo 6.2'te görüldüğü üzere, Karar Ağacı, Rastgele Orman ve KNN yöntemlerinin, ARP sahteciliği tespitinde başarım metrikleri açısından genel olarak başarılı oldukları görülmektedir. Özellikle, Rastgele Orman algoritmasının hem doğruluk hem de AUC metrikleri yüksek değerlere sahiptir. Rastgele Orman algoritması, %95.82 doğruluk oranı ve %99.17 AUC değeri ile sınıflandırma başarımı açısından diğer yöntemlere kıyasla etkili bir sonuç sunmaktadır. Karar Ağacı da %95.94 doğruluk oranı ve %98.25 AUC değeriyle çok yakın sonuçlar vermiştir. KNN algoritması ise %94.70 doğruluk ile diğer iki algoritmanın biraz gerisinde kalmış, ancak yine de yüksek sayılabilecek bir değer sunmuştur. Diğer yandan, Lojistik Regresyon yöntemi özellikle duyarlılık metriğinde (%99.64) yüksek bir başarıma sahip

olmasına rağmen, doğruluk (%81.42) ve keskinlik (%73.03) metrikleri bakımından daha düşük başarımlara sahip olduğu görülmektedir. Bu nedenle, lojistik regresyonun ARP sahteciliği tespitinde daha az etkili bir yöntem olduğu söylenebilir.

Tablo 6.2: Geleneksel makine öğrenmesi yöntemlerinin sonuçları

Yöntem	Doğruluk (%)	Keskinlik (%)	Duyarlılık (%)	F1 Skoru (%)	AUC
Karar Ağacı	95.94	93.77	98.41	96.03	98.25
Rastgele Orman	95.82	93.24	98.81	95.95	99.17
K-En Yakın Komşu	94.70	92.70	97.06	94.83	98.18
Lojistik Regresyon	81.42	73.03	99.64	84.28	86.52

Alani ve ark. çalışması ile karşılaştırma:

Alani ve ark. [7] çalışmasında, ARP sondası sisteminde kullanılan IoT-ID veri setinde bazı eksiltmeler yapıldığı ve bu çalışmada (tüm IoT-ID veri seti kullanılarak) bulunan doğruluk oranından daha yüksek bir doğruluk oranı elde edildiği görülmüştür. Alani ve ark. [7] çalışmasında %95'in üzerinde elde edilen doğruluk metriği değeri kısmi veri kullanımına dayalı olabilirken, çalışmanın tüm veri seti kullanılarak uygulandığında daha düşük değerler elde etmesinin, modelin genelleme kabiliyeti ve güvenilirliğini zayıflattığını göstermektedir. Buna karşın, bizim çalışmamızda veri setinin tamamı kullanılarak ve çapraz doğrulama yöntemleriyle daha gerçekçi ve güvenilir sonuçlar elde edilmiştir. Örneğin, bu tezde gerçekleştirilen Rastgele Orman algoritması testinde %95.82 doğruluk oranı elde edilmiştir. Ancak daha kapsamlı bir değerlendirme için sadece bu modelle yetinmeyip farklı makine öğrenmesi yöntemleri denenmiştir. Sonuç olarak, Karar Ağacı algoritması ile %95.94, KNN ile %94.70, ve DNN (50 epoch) ile %89.93 doğruluk oranlarına ulaşılmıştır. Bu sonuçlar, ARP sahteciliği tespiti için en etkili algoritmaların belirlenmesinde daha kapsamlı bir değerlendirme olanağı sunmaktadır.

6.3 Sonuç

Bu çalışma, ARP sahteciliği tespiti için DNN ve geleneksel makine öğrenmesi algoritmalarını kapsamlı bir şekilde karşılaştırarak, ağ güvenliği açısından en etkili yöntemleri belirlemeyi amaçlamıştır. Çalışma kapsamında, IoT-ID veri seti tam olarak kullanılarak, ARP sahteciliği gibi kritik bir ağ güvenliği açığının tespitinde hangi yöntemlerin daha başarılı olduğu değerlendirilmiştir. Derin öğrenme ve makine öğrenmesi yöntemleri kullanılarak elde edilen sonuçlar, her bir yöntemin güçlü ve zayıf yönlerini ortaya koymuş, özellikle Rastgele Orman ve Karar Ağacı algoritmalarının ARP sahteciliği

tespitinde en yüksek başarıyı sağladığı görülmüştür. Rastgele Orman algoritması, yüksek doğruluk (%95.82) ve AUC (%99.17) değerleriyle en başarılı sonuçları sunarken, Karar Ağacı algoritması da benzer şekilde yüksek başarım göstermiştir. Bununla birlikte, DNN yöntemi, duyarlılık açısından güçlü sonuçlar (%99.72) elde etmesine rağmen, doğruluk ve keskinlik açısından geleneksel makine öğrenmesi yöntemlerinin gerisinde kalmıştır. Bu çalışma sadece derin öğrenme yöntemine odaklanmamış, aynı zamanda Karar Ağacı, Rastgele Orman, KNN ve Lojistik Regresyon gibi geleneksel makine öğrenmesi algoritmalarıyla karşılaştırmalar yapılmıştır. Her bir yöntemin ARP sahteciliği tespitindeki başarımı değerlendirilmiş ve farklı algoritmaların belirli bir veri setinde nasıl farklı sonuçlar verdiği gözlemlenmiştir. Çalışmamızda ele alınan bu yaklaşım, ARP sahteciliği gibi karakteristik güvenlik tehditlerine karşı hangi yöntemlerin daha etkili olabileceğini anlamamıza katkı sağlamıştır.

Bu çalışmada daha karmaşık veya hibrid modellerin incelenmemiş olup, birden fazla makine öğrenmesi algoritmasının birleştirilmesiyle oluşturulan topluluk (ensemble) yöntemlerine ya da daha gelişmiş derin öğrenme mimarilerine (RNN veya LSTM gibi) odaklanılmamıştır. Ayrıca, kullanılan veri setinin genişletilmesi veya farklı ağ güvenliği tehditlerinin de ele alınması, çalışma kapsamına dahil edilmemiştir. Bu alanlar, gelecekteki araştırmalar için önemli bir potansiyel taşımaktadır.

Gelecek çalışmalar için öneriler arasında, bu çalışmada kullanılan algoritmaların daha gelişmiş versiyonlarının veya hibrid modellerin incelenmesi yer alabilir. Örneğin, topluluk yöntemleri (ensemble) gibi güçlü modellerin kullanılmasıyla doğruluk oranlarının daha da artırılabileceği öngörülmektedir. Ayrıca, veri ön işleme tekniklerinin geliştirilmesiyle (gürültü azaltma, veri artırma gibi) model başarımlarının iyileştirilmesi mümkün olabilir. Bu çalışmada veri seti tam olarak kullanılmış olsa da farklı ve daha büyük veri setleriyle testler yapılması, sonuçların genelleme yeteneğini artırabilir ve daha güvenilir başarım sonuçları elde edilebilir.

Sonuç olarak, bu tez çalışması, ARP sahteciliği tespiti için kullanılan çeşitli makine öğrenmesi ve derin öğrenme yönteminin başarımlarını kapsamlı bir şekilde incelemiş ve bu güvenlik tehditlerine karşı en etkili algoritmaları belirlemiştir. Rastgele Orman ve Karar Ağacı algoritmalarının ARP sahteciliğini tespit etmede en etkili yöntemler olduğu

görülmüştür. Gelecekte, daha karmaşık modeller ve farklı veri setleri ile yapılacak çalışmaların bu sonuçları daha da geliştirebileceği düşünülmektedir.

7. KAYNAKLAR

[1] K. de. Leeuw ve J. A. Bergstra, "The history of information security : a comprehensive handbook", ss. 601-602, 2007.

[2] O. R. Tabibian, "Internet Scanner Finds Security Holes", *PC Magazine*, Nis. 1996, Erişim: 16 Mart 2024. [Çevrimiçi]. Erişim adresi: https://books.google.com/books?id=LYp7r6OrMdIC&pg=PA536

[3] "Penetration Testing Market Size, Share | Industry Report". Erişim: 16 Mart 2024. [Çevrimiçi]. Erişim adresi: https://www.marketsandmarkets.com/Market-Reports/penetration-testing-market-13422019.html?gad_source=1&gclid=Cj0KCQiA84CvBhCaARIsAMkAvkJ0a5wwTKj3pyowkRuuwEeVTZ9MCxqrSaworf-C3szQ73y3PIxz7aIaAqx8EALw_wcB

[4] "Cyber Security Market Size & Forecast, [Latest]". Erişim: 16 Mart 2024. [Çevrimiçi]. Erişim adresi: https://www.marketsandmarkets.com/Market-Reports/cyber-security-market-505.html

[5] D. C. Plummer, "An Ethernet Address Resolution Protocol", 1982.

[6] E. Ylli ve J. Fejzaj, "Man in the Middle: Attack and Protection", 2021.

[7] M. M. Alani, A. I. Awad ve E. Barka, "ARP-PROBE: An ARP spoofing detector for Internet of Things networks using explainable deep learning", *Internet of Things (Netherlands)*, c. 23, Eki. 2023, doi: 10.1016/j.iot.2023.100861.

[8] Hyunjae Kang, Dong Hyun Ahn, Gyung Min Lee, Jeong Do Yoo, Kyung Ho Park, ve Huy Kang Kim, "IoT network intrusion dataset | IEEE DataPort". Erişim: 26 Eylül 2024. [Çevrimiçi]. Erişim adresi: https://ieee-dataport.org/open-access/iot-network-intrusion-dataset

[9] Sharma Divya, Khan Oves ve Manchanda Nidhi, "2014 International Conference on Computing for Sustainable Global Development (INDIACom) : 05th to 07th March, 2014, Bharati Vidyapeeth's lnstitute of Computers Applications and Management (BVICAM)", Institute of Electrical and Electronics Engineers, 2014, s. 578.

[10] M. G. Gouda ve C.-T. Huang, "A secure address resolution protocol", 2003. [Çevrimiçi]. Erişim adresi: www.elsevier.com/locate/comnet

[11] G. Jinhua ve X. Kejian, "ARP spoofing detection algorithm using ICMP protocol", *2013 International Conference on Computer Communication and Informatics, ICCCI 2013*, 2013, doi: 10.1109/ICCCI.2013.6466290.

[12] M. E. Khan ve F. Khan, "A Comparative Study of White Box, Black Box and Grey Box Testing Techniques", 2012. [Çevrimiçi]. Erişim adresi: www.ijacsa.thesai.org

[13] K. Reed, T. Murnane, T. Associates ve C. V. Australia, "On the Effectiveness of Mutation Analysis as a Black Box Testing Technique. On the Effectiveness of Mutation Analysis as a Black Box Testing Technique", 2001. [Çevrimiçi]. Erişim adresi: https://www.researchgate.net/publication/220784514

[14] "OWASP Top Ten | OWASP Foundation". Erişim: 22 Mart 2024. [Çevrimiçi]. Erişim adresi: https://owasp.org/www-project-top-ten/

[15] P. Herzog, "OSSTMM 3–The open source security testing methodology manual", *Institute for Security and Open Methodologies (ISECOM)*, s. 15, 2010.

[16] M. Prandini ve M. Ramilli, "Towards a practical and effective security testing methodology", *Proc IEEE Symp Comput Commun*, ss. 320-325, 2010, doi: 10.1109/ISCC.2010.5546813.

[17] F. Abu-Dabaseh ve E. Alshammari, "AUTOMATED PENETRATION TESTING: AN OVERVIEW", ss. 121-129, 2018, doi: 10.5121/csit.2018.80610.

[18] "Information Systems Security Assessment Framework (ISSAF) draft 0.2", 2005.

[19] K. Scarfone, M. Souppaya, A. Cody ve A. Orebaugh, “Guideline on Network Security Testing”, Eki. 2003, doi: 10.6028/NIST.SP.800-42.
[20] G. Chu, “Automation of Penetration Testing”, 2021.
[21] S. Shah, · B M Mehtre, S. Shah ve B. M. Mehtre, “An overview of vulnerability assessment and penetration testing techniques”, *J Comput Virol Hack Tech*, c. 11, ss. 27-49, 2015, doi: 10.1007/s11416-014-0231-x.
[22] “The Penetration Testing Execution Standard”. Erişim: 22 Mart 2024. [Çevrimiçi]. Erişim adresi: http://www.pentest-standard.org/index.php/Main_Page
[23] E. A. Altulaihan, A. Alismail ve M. Frikha, “A Survey on Web Application Penetration Testing”, *Electronics (Switzerland)*, c. 12, sy 5, Mar. 2023, doi: 10.3390/ELECTRONICS12051229.
[24] W. G. J. Halfond, J. Viegas, ve A. Orso, “A Classification of SQL Injection Attacks and Countermeasures”, 2006.
[25] “ZAP ”. Erişim: 10 Mayıs 2024. [Çevrimiçi]. Erişim adresi: https://www.zaproxy.org/getting-started/#overview
[26] G. A. Jaafar, S. M. Abdullah ve S. Ismail, “Review of Recent Detection Methods for HTTP DDoS Attack”, 2019, *Hindawi Limited*. doi: 10.1155/2019/1283472.
[27] L. Bosnjak, J. Sres ve B. Brumen, “Brute-force and dictionary attack on hashed real-world passwords”, *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics, MIPRO 2018 - Proceedings*, ss. 1161-1166, Haz. 2018, doi: 10.23919/MIPRO.2018.8400211.
[28] U. Sarmah, D. K. Bhattacharyya ve J. K. Kalita, “A survey of detection methods for XSS attacks”, 15 Eylül 2018, *Academic Press*. doi: 10.1016/j.jnca.2018.06.004.
[29] J. F. Kurose *vd.*, *Computer Networking A Top-Down Approach Seventh Edition*. 2017. Erişim: 02 Temmuz 2024. [Çevrimiçi]. Erişim adresi: www.pearsoned.com/permissions/.
[30] B. Prabadevi ve N. Jeyanthi, “Security solution for ARP cache poisoning attacks in large data centre networks”, *Cybernetics and Information Technologies*, c. 17, sy 4, ss. 69-86, 2017, doi: 10.1515/CAIT-2017-0042.
[31] “Address Resolution Protocol (ARP) Parameters”. Erişim: 14 Eylül 2024. [Çevrimiçi]. Erişim adresi: https://www.iana.org/assignments/arp-parameters/arp-parameters.xhtml
[32] LAN/MAN Standards Committee of the IEEE Computer Society, “IEEE Standard for Ethernet”, *IEEE Standard for Ethernet*, 2018.
[33] Y. Jeong, H. Kim ve H. J. Jo, “ASD: ARP Spoofing Detector Using OpenWrt”, *Security and Communication Networks*, c. 2022, 2022, doi: 10.1155/2022/2196998.
[34] F. Mvah, V. Kengne Tchendji, C. Tayou Djamegni, A. H. Anwar, D. K. Tosh ve C. Kamhoua, “GaTeBaSep: game theory-based security protocol against ARP spoofing attacks in software-defined networks”, *Int J Inf Secur*, c. 23, sy 1, ss. 373-387, Şub. 2024, doi: 10.1007/s10207-023-00749-0.
[35] M. A. Kumar *vd.*, “Detection and mitigation of few control plane attacks in software defined network environments using deep learning algorithm”, *Concurr Comput*, 2024, doi: 10.1002/cpe.8256.
[36] Z. Shah ve S. Cosgrove, “Mitigating arp cache poisoning attack in software-defined networking (sdn): A survey”, *Electronics (Switzerland)*, c. 8, sy 10, Eki. 2019, doi: 10.3390/electronics8101095.
[37] V. Kanimozhi ve T. P. Jacob, “The Top Ten Artificial Intelligence-Deep Neural Networks for IoT Intrusion Detection System”, *Wirel Pers Commun*, c. 129, sy 2, ss. 1451-1470, Mar. 2023, doi: 10.1007/s11277-023-10198-6.

[38] F. Mvah, V. Kengne Tchendji, C. Tayou Djamegni, A. H. Anwar, D. K. Tosh ve C. Kamhoua, "Countering ARP spoofing attacks in software-defined networks using a game-theoretic approach", *Comput Secur*, c. 139, Nis. 2024, doi: 10.1016/j.cose.2023.103696.
[39] S. Hijazi, M. S. Obaidat ve S. Obaidat, "Address resolution protocol spoofing attacks and security approaches: A survey", 2018, doi: 10.1002/spy2.49.
[40] C. Venkata Guru Rao ve D. Guru Rao, "Various Solutions for Address Resolution Protocol Spoofing Attacks", 2013. [Çevrimiçi]. Erişim adresi: www.ijsrp.org
[41] S. M. Morsy ve D. Nashat, "D-ARP: An Efficient Scheme to Detect and Prevent ARP Spoofing", *IEEE Access*, c. 10, ss. 49142-49153, 2022, doi: 10.1109/ACCESS.2022.3172329.
[42] Y. Lecun, Y. Bengio ve G. Hinton, "Deep learning", 27 Mayıs 2015, *Nature Publishing Group*. doi: 10.1038/nature14539.
[43] V. Nair ve G. E. Hinton, "Rectified Linear Units Improve Restricted Boltzmann Machines".
[44] M. Roodschild, J. Gotay Sardiñas ve A. Will, "A new approach for the vanishing gradient problem on sigmoid activation", *Progress in Artificial Intelligence*, c. 9, sy 4, ss. 351-360, Ara. 2020, doi: 10.1007/s13748-020-00218-y.
[45] T. Dozat, "Workshop track-ICLR 2016 INCORPORATING NESTEROV MOMENTUM INTO ADAM".
[46] B. Charbuty ve A. Abdulazeez, "Classification Based on Decision Tree Algorithm for Machine Learning", *Journal of Applied Science and Technology Trends*, c. 2, sy 01, ss. 20-28, Mar. 2021, doi: 10.38094/jastt20165.
[47] L. Breiman, "Random Forests", 2001.
[48] K. Taunk, S. De, S. Verma ve A. Swetapadma, "A brief review of nearest neighbor algorithm for learning and classification", *2019 International Conference on Intelligent Computing and Control Systems, ICCS 2019*, ss. 1255-1260, May. 2019, doi: 10.1109/ICCS45141.2019.9065747.
[49] D. Berrar, "Cross-validation 1".
[50] H. Dalianis, "Evaluation Metrics and Evaluation", *Clinical Text Mining*, ss. 45-53, 2018, doi: 10.1007/978-3-319-78503-5_6.
[51] T. Fawcett, "An introduction to ROC analysis", *Pattern Recognit Lett*, c. 27, sy 8, ss. 861-874, Haz. 2006, doi: 10.1016/j.patrec.2005.10.010.

ÖZGEÇMİŞ

Kişisel Bilgiler

Adı Soyadı : Mustafa Furkan CEYLAN
Doğum tarihi ve yeri :
e-posta :

Öğrenim Bilgileri

Derece	**Okul/Program**	**Yıl**
Y. Lisans	Balıkesir Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar ve Bilişim Mühendisliği	2024
Lisans	Uludağ Üniversitesi / Mühendislik Fakültesi / Elektrik-Elektronik Mühendisliği Bölümü	2020
Lise	Ankara Uluğbey Anadolu Lisesi	2015