

LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

E-TİCARET SİTELERİ İÇİN GÜVENLİK AÇIĞI TESPİTİNE İLİŞKİN SİSTEM TASARIMI

Kenan İSLAMOĞLU

YÜKSEK LİSANS TEZİ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI
İSTANBUL – 2025

T.C. İSTANBUL TİCARET ÜNİVERSİTESİ

LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

E-TİCARET SİTELERİ İÇİN GÜVENLİK AÇIĞI TESPİTİNE İLİŞKİN SİSTEM TASARIMI

Kenan İSLAMOĞLU

Danışman
Dr. Öğr. Üyesi Fatma Nur AKI

YÜKSEK LİSANS TEZİ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI
İSTANBUL – 2025

KABUL VE ONAY SAYFASI

Kenan İSLAMOĞLU tarafından hazırlanan **"E Ticaret Siteleri İçin Güvenlik Açığı Tespitine İlişkin Sistem Tasarımı"** adlı tez çalışması 02/09/2025 tarihinde aşağıdaki jüri üyeleri önünde başarı ile savunularak, İstanbul Ticaret Üniversitesi Fen Bilimleri Enstitüsü **Bilgisayar Mühendisliği Anabilim Dalı**'nda **Siber Güvenlik** Tezli **YÜKSEK LİSANS TEZİ** olarak kabul edilmiştir.

Danışman	**Dr. Öğr. Üyesi Fatma Nur AKI** İstanbul Ticaret Üniversitesi	.....
Jüri Üyesi	**Unvanı Adı SOYADI** İstanbul Ticaret Üniversitesi	.....
Jüri Üyesi	**Unvanı Adı SOYADI** İstanbul Ticaret Üniversitesi	.....
Jüri Üyesi	**Unvanı Adı SOYADI** İstanbul Ticaret Üniversitesi	.....
Jüri Üyesi	**Unvanı Adı SOYADI** İstanbul Ticaret Üniversitesi	.....

Onay Tarihi : **(Enstitü Tarafından Doldurulacaktır.)**

Doç. Dr. Başak ERDEM ÇANKAYA
Enstitü Müdürü

AKADEMİK VE ETİK KURALLARA

UYGUNLUK BEYANI

İstanbul Ticaret Üniversitesi, Fen Bilimleri Enstitüsü, tez yazım kurallarına uygun olarak hazırladığım bu tez çalışmasında,

- Tez içindeki bütün bilgi ve belgeleri akademik kurallar çerçevesinde elde ettiğimi,
- Görsel, işitsel ve yazılı tüm bilgi ve sonuçları bilimsel ahlak kurallarına uygun olarak sunduğumu,
- Başkalarının eserlerinden yararlanılması durumunda ilgili eserlere bilimsel normlara uygun olarak atıfta bulunduğumu,
- Atıfta bulunduğum eserlerin tümünü kaynak olarak gösterdiğimi,
- Kullanılan verilerde herhangi bir tahrifat yapmadığımı,
- ve bu tezin herhangi bir bölümünü bu üniversitede veya başka bir üniversitede başka bir tez çalışması olarak sunmadığımı

beyan ederim.

Tarih

İmza

Kenan İSLAMOĞLU

İÇİNDEKİLER

Sayfa

İÇİNDEKİLER i
ÖZET iiv
ABSTRACT vi
TEŞEKKÜR viii
ŞEKİLLER iix
ÇİZELGELER x
SİMGELER VE KISALTMALAR xi
1.GİRİŞ 1
2.LİTERATÜR ÖZETİ 3
2.1 E-Ticaret Mimarisinin Yapısı 3
2.2 Güvenlik İhlalleri ve Etkileri 8
2.3 Mevcut Güvenlik Önlemleri 10
2.3.1 SSL/TLS Protokolleri 10
2.3.2 CAPTCHA Sistemleri 12
2.3.3 İki Faktörlü Kimlik Doğrulama (2FA) 13
2.3.4 Güvenli Yazılım Geliştirme Teknikleri 14
2.3.5 Diğer Yaygın Güvenlik Uygulamaları: Güvenlik Duvarı, Sızma Testi, Veri Şifreleme 15
3.GÜVENLİK AÇIKLARI: TÜRLERİ VE ANALİZİ 17
3.1.1 Dinamik analiz 27
3.1.2 Statik analiz 28
3.1.3 Sızma testleri (Pentest) 29
3.1.4 Zafiyet tarayıcıları 30
4.GÜVENLİK AÇIĞI TESPİTİ 34
4.1 Mimari Yapısı 35
4.2 Açık Tespit Süreci 37
4.2.1 Araç Seçim Kriterleri ve Gerekçeler 38
4.3 Otomatikleştirme Yaklaşımları 39
4.3.1 Python Script'leri ile Otomasyon 39
4.3.2 API Tabanlı Entegrasyonlar 40
4.3.3 Otomatik Raporlama Sistemleri 40
4.3.4 Yararları ve Sınırlılıkları 40
4.4 Yasal ve Etik Boyutlar 42
5.RAPORLAMA SİSTEMİ 45

5.1 Raporlama Formatı ve İçeriği 45
5.1.1 Teknik Raporlama Bileşenleri 45
5.1.2 Açıklık Türüne Göre Rapor Şablonları 47
5.2 Önceliklendirme ve Risk Skorlaması 49
5.3 Otomatik Raporlama Araçları 52
5.4 Açık Özeti 53
5.5 Etki Seviyesi 54
5.6 Önerilen Çözüm 56
5.6.1 Kod Tabanlı Zafiyetler için Güvenli Geliştirme Yaklaşımları 56
5.6.2 Kimlik Doğrulama ve Oturum Güvenliği İçin Yapısal Önlemler 57
5.6.3 Güncellenmeyen Bileşenler ve Yapılandırma Hataları için Süreçsel Çözümler 57
5.6.4 Sürekli İzleme, Alarm ve Denetim Süreçleri 58
6.ARAŞTIRMA BULGULARI: UYGULAMA VE VAKA ÇALIŞMASI 59
6.1 Test Ortamının Kurulması 59
6.1.1 Hedef Sistem Alt Yapı Tanımı 59
6.1.2 Gerekli Yazılım ve Araçların Kurulumu 60
6.1.3 Ağ ve Sertifika Yapılandırması 61
6.2 Sistem Tasarımının Uygulanması 63
6.2.1 Bilgi Toplama 64
6.2.1.1 Port ve Servis Taraması (Nmap) 65
6.2.1.2 Açık Kaynak İstihbaratı (Shodan vb.) 66
6.2.2 Zafiyet Tarama 68
6.2.2.1 SSL/TLS Konfigürasyon Analizi (nmap ssl-enum-ciphers) 68
6.2.2.2 HTTP Güvenlik Başlıkları Analizi (Nikto) 70
6.2.2.3 OWASP ZAP ile Web Uygulaması Taraması 71
6.2.2.4 Burp Suite ile Zafiyet Analizi 75
6.2.3 Açık Doğrulama ve Yanlış Pozitiflerin Eliminasyonu 77
6.3 Bulguların Raporlanması 78
6.3.1 Teknik Rapor (CVSS Skorları ile) 78
6.3.2 Yönetici Özeti Raporu 80
7.DEĞERLENDİRME VE SONUÇLAR 83
7.1 Sistem Tasarımının Güvenilirliği ve Etkililiği 83
7.2 Zayıf Noktalar ve Geliştirilebilecek Alanlar 84
7.3 E-Ticaret Siteleri İçin Öneriler 86
7.3.1 Periyodik Testler 86
7.3.2 Açık Yönetim Süreçlerinin Kurumsallaştırılması 88

7.3.3 Eğitim ve Farkındalık Faaliyetleri 90
SONUÇLAR VE TARTIŞMA 93
KAYNAKLAR 96
ÖZGEÇMİŞ 99

ÖZET

Yüksek Lisans Tezi

E TİCARET SİTELERİNDE GÜVENLİK AÇIĞI TESPİTİ VE RAPORLAMA MODELİ

Kenan İSLAMOĞLU

İstanbul Ticaret Üniversitesi
Lisansüstü Eğitim Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı

Danışman: Dr. Öğretim Üyesi Fatma Nur AKI

2025, 99 sayfa

Günümüzde e-ticaret platformları, kullanıcı verileri ve finansal işlemleri bünyesinde barındırmaları nedeniyle siber saldırıların öncelikli hedefleri arasında yer almaktadır. Bu durum, işletmelerin veri güvenliğini sağlamada etkili yöntemlere duyduğu ihtiyacı her geçen gün artırmaktadır. Mevcut literatürde güvenlik açıklarının tespitine yönelik pek çok çalışma bulunsa da, çoğu yaklaşım ya yalnızca belirli bir açık türüne odaklanmakta ya da yalnızca manuel veya otomatik analiz yöntemlerinden birini kullanmaktadır. Bu tez, söz konusu sınırlılıkları aşmak amacıyla, manuel ve otomatik analiz tekniklerini birleştiren, çok aşamalı ve ölçeklenebilir bir Güvenlik Açığı Tespitine İlişkin Sistem Tasarımı önermektedir.

Önerilen tasarım, hedef belirleme, bilgi toplama, zafiyet taraması, açık doğrulama ve raporlama olmak üzere beş ana aşamadan oluşmaktadır. İlk aşamada, test kapsamı ve hedef platformlar belirlenmiş; ikinci aşamada, pasif ve aktif bilgi toplama yöntemleri ile sistem altyapısına ilişkin veriler elde edilmiştir. Üçüncü aşamada, OWASP ZAP, Nikto, Nmap ve Burp Suite gibi yaygın kullanılan güvenlik araçları ile zafiyet taraması gerçekleştirilmiş; dördüncü aşamada ise tespit edilen açıklar manuel test teknikleri ile doğrulanmıştır. Bu doğrulama süreci, yanlış pozitifleri en aza indirmeyi ve gerçek anlamda sömürülebilir (exploitable) açıklıkları önceliklendirmeyi sağlamıştır.

Tespit edilen her bir açık, Common Vulnerability Scoring System (CVSS) v3.1 metodolojisine göre skorlanmış; etki düzeyleri, istismar zorluk dereceleri ve potansiyel zararları değerlendirilmiştir. Bulgular, incelenen e-ticaret sitelerinin önemli bir kısmında SQL Injection (SQLi), Cross-Site Scripting (XSS), güvenlik yapılandırma hataları, kimlik doğrulama zafiyetleri, eksik güvenlik başlıkları ve eski TLS protokol desteği gibi kritik seviyede açıklıkların bulunduğunu ortaya koymuştur.

Modelin uygulama süreci sonunda hazırlanan raporlar iki farklı formatta sunulmuştur:

1. Teknik Rapor: Güvenlik ekiplerinin doğrudan müdahale edebileceği, ayrıntılı teknik açıklamalar ve çözüm önerileri içeren rapor.
2. Yönetici Özeti: Karar vericilerin teknik detaylara boğulmadan risk seviyelerini ve öncelikli aksiyonları görebileceği sadeleştirilmiş özet rapor.

Çalışma, yalnızca açıklıkların tespitine değil, aynı zamanda önceliklendirilmesine ve çözüm önerilerinin uygulanabilirliğine odaklanmaktadır. Bu yönüyle önerilen tasarım, sınırlı kaynaklara sahip KOBİ'lerden büyük ölçekli e-ticaret platformlarına kadar geniş bir yelpazede uygulanabilir niteliktedir. Sonuç olarak, önerilen tasarımın, e-ticaret sistemlerinde güvenlik farkındalığını artırdığı, müdahale sürelerini kısalttığı ve siber tehditlere karşı proaktif bir savunma sağladığı görülmüştür.

Anahtar Kelimeler: CVSS, E-ticaret, Güvenlik açığı tespiti, OWASP, Sistem tasarımı, Siber güvenlik

ABSTRACT

M.Sc. Thesis

SECURITY VULNERABILITY DETECTION AND REPORTING MODEL IN E-COMMERCE SITES

Kenan İSLAMOĞLU

İstanbul Commerce University
Institute of Graduate School
Department of Computer Engineering

Supervisor: Assistant Professor Dr. Fatma Nur AKI

2025, 99 pages

Today, e-commerce platforms are among the primary targets of cyberattacks due to hosting user data and financial transactions. This situation increases the need for businesses to adopt effective methods for ensuring data security. Although numerous studies in the existing literature focus on vulnerability detection, most approaches either target a specific type of vulnerability or rely solely on manual or automated analysis methods. This thesis proposes a Security Vulnerability Detection System Design that overcomes these limitations by integrating both manual and automated analysis techniques within a multi-stage and scalable framework.

The proposed model consists of five main stages: target identification, information gathering, vulnerability scanning, vulnerability verification, and reporting. In the first stage, the test scope and target platforms were defined; in the second stage, both passive and active information gathering methods were employed to obtain data on the system infrastructure. In the third stage, vulnerability scanning was performed using widely adopted security tools such as OWASP ZAP, Nikto, Nmap, and Burp Suite; in the fourth stage, the detected vulnerabilities were verified through manual testing techniques. This verification process minimized false positives and ensured the prioritization of truly exploitable vulnerabilities.

Each identified vulnerability was scored according to the Common Vulnerability Scoring System (CVSS) v3.1 methodology, evaluating their impact levels, exploitation difficulty, and potential damage. The findings revealed that a significant proportion of the analyzed e-commerce sites contained critical vulnerabilities such as SQL Injection (SQLi), Cross-Site Scripting (XSS), security misconfigurations, authentication flaws, missing security headers, and outdated TLS protocol support.

At the end of the implementation process, the reports generated were presented in two formats:

1. Technical Report: Detailed technical documentation including comprehensive explanations and solution recommendations for security teams to act upon immediately.
2. Executive Summary**:** A simplified overview enabling decision-makers to assess risk levels and prioritize actions without delving into technical complexity.

This study focuses not only on the detection of vulnerabilities but also on their prioritization and the applicability of proposed solutions. In this respect, the developed design is applicable to a wide range of contexts, from small and medium-sized enterprises (SMEs) with limited resources to large-scale e-commerce platforms. As a result, the proposed design has been shown to enhance security awareness, shorten response times, and provide a proactive defense against cyber threats in e-commerce systems.

Keywords: CVSS, Cybersecurity, E-commerce, OWASP, System design, Vulnerability detection

TEŞEKKÜR

Tez çalışmamın her aşamasında, akademik birikimi, rehberliği ve değerli katkılarıyla bana yol gösteren, araştırma sürecinde karşılaştığım tüm zorlukları bilimsel bir bakış açısıyla aşmamda destek olan çok kıymetli Danışman Hocam Dr. Öğretim Üyesi Fatma Nur AKI'ya en içten teşekkürlerimi sunarım. Hocamın titiz yaklaşımı, yapıcı eleştirileri ve yönlendirmeleri, çalışmanın bilimsel niteliğini güçlendirmiştir.

Ayrıca bu sürecin en başından itibaren yanımda olan, destek ve anlayışlarını esirgemeyen aileme şükran borçluyum. Tez sürecindeki yoğunluğumu sabırla karşılayan ve desteğini esirgemeyen sevgili eşim Kübra İSLAMOĞLU'na; yaşamıma neşe, ilham ve anlam katan çocuklarım Aslı BERİL ve Selim ALP'e sevgi, saygı ve minnetle teşekkür ederim. Onların varlığı ve desteği, bu çalışmayı tamamlamamdaki en büyük güç kaynağı olmuştur.

Kenan İSLAMOĞLU
İSTANBUL, 2025

ŞEKİLLER

Sayfa

Şekil 2.1. E–Ticaret Sistemlerinin Mimarisindeki Katmanlar 3
Şekil 2.2. SSL Çalışma Döngüsü .....…11
Şekil 3.1. E-Ticaret Sitelerinde En Sık Görülen Güvenlik Açıkları 26
Şekil 3.2. Sızma testi süreci ..... ….…….....30
Şekil 3.3. Zafiyet Tarayıcılarının İşlevi ve Çalışma Prensibi 31
Şekil 4.1. Güvenlik açığı tespitine ilişkin sistem tasarımının akış şeması..... 35
Şekil 6.1. Hyper-V Manager – Attacker ve Target VM'ler 60
Şekil 6.2. nopCommerce ana sayfası (HTTPS) 60
Şekil 6.3. PowerShell ile Sertifika Oluşturması 62
Şekil 6.4. Sertifikayı Güvenilir Kök Sertifika Deposuna Ekleme 62
Şekil 6.5. Kestrel'in Sertifikayı Kullanacak Şekilde Yapılandırılması 63
Şekil 6.6. Modelin Mimari Yapısı 64
Şekil 6.7. Tespit İçin Kullanılan Komut 65
Şekil 6.8. Ekran Görüntüsü: Nmap – Port ve Servis Taraması Sonuçları …….. 66
Şekil 6.9. Kullanılan Komut 68
Şekil 6.10. Ekran Görüntüsü: SSL/TLS Tarama Çıktısı……..…….69
Şekil 6.11. HTTP Güvenlik Başlıkları Analizi İçin Kullanılan Komut 70
Şekil 6.12. Ekran Görüntüsü: Nikto HTTP Güvenlik Başlıkları Analizi…..... 71
Şekil 6.13. Hedef Belirlemek İçin Kullanılan HTTPS Servisi .....…72
Şekil 6.14. OWASP ZAP ile Web Uygulaması Taraması Sonuç Ekranı 74
Şekil 6.15. Burp Suite ile kullanıcı kayıt formunda SQL Injection dene.mesi …….76
Şekil 6.16. Yanlış Pozitif SQL Injection Tespitinin Manuel Doğrulama Süreci (Burp Suite Repeater) ..... …….……78

ÇİZELGELER

Sayfa

Çizelge 4.1. Otomatik Açık Tespit Yöntemlerinin Yararları ve Sınırlılıkları....42
Çizelge 5.1. Örnek Yönetici Raporu....46
Çizelge 5.2. CVSS Hesaplama Kriterleri....50
Çizelge 5.3. CVSS Skorlarının Sınıflandırılması....51
Çizelge 5.4. Açıklık Etkisine ve Olasılığına Göre Risk Matrisi....51
Çizelge 5.5. E-Ticaret Sistemleri İçin Örnek Önceliklendirme Senaryoları....51
Çizelge 5.6. Örnek Açıklık Özeti....54
Çizelge 5.7. Etki Seviyesini Gösteren Örnek Tablo....55
Çizelge 6.1. Hedef Sistem Alt Yapıları....59
Çizelge 6.2. Tarama Sonuçları....65
Çizelge 6.3. Örnek Shodan Çıktısı (Eğitim Amaçlı)....67
Çizelge 6.4. Elde Edilen Bulgular....68
Çizelge 6.5. Örnek Tespit Edilen Bulgular....73
Çizelge 6.6. Güvenlik testleri sonucunda tespit edilen Bulgular....79
Çizelge 6.7. Tespit edilen güvenlik açıklarının CVSS skorlarına ve risk derecelerine göre sınıflandırılması....82
Çizelge 6.8. CVSS skorları ve açık türlerinin yüzdesel dağılımı....82
Çizelge 6.9. Yönetici Odaklı Rapor Örneği....82
Çizelge 6.10. Teknik rapor ve yönetici Raporları Arasındaki Fark....82
Çizelge 7.1. Geliştirilebilecek Alanların Önceliklendirilmesi….....85
Çizelge 7.2. Uygulanabilir Öneri Tablosu…….....87
Çizelge 7.3. Açık Sorumluluk Takip Tablosu..…….....89

SİMGELER VE KISALTMALAR

API Application Programming Interface – Uygulama Programlama Arayüzü
CIA Confidentiality, Integrity, Availability – Gizlilik, Bütünlük, Erişilebilirlik
CMS Content Management System – İçerik Yönetim Sistemi
CSS Cascading Style Sheets – Basamaklı Stil Şablonları
CVSS Common Vulnerability Scoring System – Ortak Güvenlik Açığı Puanlama Sistemi
DNS. Domain Name System – Alan Adı Sistemi
HTML HyperText Markup Language – Köprü Metin İşaretleme Dili
HTTP HyperText Transfer Protocol – Kö.prü Metin Transfer Protokolü
HTTPS HyperText Transfer Protocol Secure – Güvenli Köprü Metin Transfer Protokolü
ID Identifier – Tanımlayıcı
IP Internet Protocol – İnternet Protokolü
ISO. International Organization for Standardization – Uluslararası Standartlar Organizasyonu
JSON. JavaScript Object Notation – JavaScript Nesne Gösterimi
OWASP Open Web Application Security Project – Açık Web Uygulama Güvenliği Projesi
PCI. Payment Card Industry – Ödeme Kartı Endüstrisi
PDF Portable Document Format – Taşınabilir Belge Formatı
PHP. Hypertext Preprocessor – PHP Programlama Dili
REST. Representational State Transfer – Temsili Durum Transferi
SDK. Software Development Kit – Yazılım Geliştirme Kiti
SHA. Secure Hash Algorithm – Güvenli Karma Algoritması
SIEM. Security Information and Event Management – Güvenlik Bilgi ve Olay Yönetimi
SQL Structured Query Language – Yapılandırılmış Sorgu Dili
SSL Secure Sockets Layer – Güvenli Yuva Katmanı
TCP. Transmission Control Protocol – İletim Kontrol Protokolü
TLS Transport Layer Security – Taşıma Katmanı Güvenliği
UI User Interface – Kullanıcı Arayüzü
URL Uniform Resource Locator – Tekdüzen Kaynak Konumlayıcı
VM Virtual Machine – Sanal Makine

WAF. Web Application Firewall – Web Uygulama Güvenlik Duvarı

XML. Extensible Markup Language – Genişletilebilir İşaretleme Dili

XSS Cross-Site Scripting – Siteler Arası Komut Dosyası Çalıştırma

ZAP. Zed Attack Proxy – OWASP Güvenlik Test Aracı

1. GİRİŞ

1.1 Tezin Amacı ve Önemi

Günümüzde dijitalleşmenin hız kazanmasıyla birlikte e-ticaret, bireyler ve kurumlar için vazgeçilmez bir ticaret yöntemi hâline gelmiştir. Ancak bu hızlı dönüşüm, beraberinde ciddi siber güvenlik tehditlerini de getirmiştir. Özellikle e-ticaret siteleri, barındırdıkları müşteri bilgileri, ödeme verileri ve ticari sırlar nedeniyle siber saldırıların başlıca hedefleri arasında yer almaktadır.

Bu çalışmanın temel amacı, e-ticaret sitelerinde en yaygın biçimde karşılaşılan güvenlik açıklarını belirlemek, bu açıkların sistematik bir biçimde nasıl tespit edilebileceğini ve etkin şekilde nasıl raporlanabileceğini ortaya koymaktır. Aynı zamanda, bu açıklara karşı geliştirilebilecek önleyici yöntemlerin değerlendirilmesi de tezin hedefleri arasındadır.

Tezin önemi, siber saldırıların doğrudan ekonomik ve itibari kayıplara neden olabildiği dijital ticaret ortamında, güvenlik açığı tespiti ve raporlamasının sistematik bir modele oturtulması gerekliliğinden kaynaklanmaktadır. Bu kapsamda geliştirilecek öneri ve yaklaşımlar hem yazılım geliştiriciler hem de e-ticaret altyapısını yöneten kurumlar için önemli bir rehber niteliği taşıyacaktır. Ayrıca, kullanıcı verilerinin korunmasına yönelik farkındalığın artmasına katkı sağlayarak, güvenli bir dijital ticaret ortamının inşasına destek olması beklenmektedir.

1.2 Problem Tanımı

E-ticaret platformlarının dijital dünyada kazandığı önemle birlikte, bu sistemlere yönelik siber tehditlerin sayısı ve çeşitliliği de artış göstermektedir. Gerek yazılımsal mimarideki zafiyetler gerekse kullanıcı davranışlarına bağlı olarak ortaya çıkan güvenlik açıkları hem bireysel kullanıcılar hem de firmalar açısından ciddi riskler barındırmaktadır. Kullanıcı bilgilerinin ele geçirilmesi, kimlik hırsızlığı, dolandırıcılık, sahte ödeme işlemleri ve sistemlerin hizmet dışı bırakılması gibi tehditler, e-ticaret ortamında güvenin sarsılmasına yol açmaktadır.

Firmalar açısından bakıldığında ise bu güvenlik açıkları; maddi kayıpların yanı sıra itibar zedelenmesi, müşteri sadakatinde azalma ve yasal yaptırımlarla karşı karşıya

kalma gibi sonuçlar doğurabilmektedir. Güvenlik zaaflarının zamanında tespit edilememesi, saldırıların daha geniş çaplı sonuçlara yol açmasına neden olmaktadır.

Bu bağlamda, e-ticaret sitelerinde yaygın biçimde karşılaşılan güvenlik açıklarının türlerinin belirlenmesi ve bu açıkların kullanıcılar ve firmalar üzerindeki olumsuz etkilerinin analiz edilmesi, çalışmanın temel problemini oluşturmaktadır. Söz konusu açıklara karşı etkili önlemler geliştirilebilmesi için öncelikle bu tehditlerin yapısal olarak tanımlanması ve sistematik biçimde raporlanması gerekmektedir.

1.3 Tezin Kapsamı ve Sınırlamaları

Bu tez çalışması, e-ticaret sitelerinde en sık karşılaşılan güvenlik açıklarının teknik, yapısal ve uygulamaya yönelik yönlerini ele almayı amaçlamıştır. Özellikle XSS (Cross-Site Scripting) ve SQL Injection gibi yaygın güvenlik açıkları üzerinden yapılan analizlerle, bu açıklara karşı geliştirilebilecek önleme yöntemleri değerlendirilmiştir.

Çalışmada test ortamı, ASP.NET MVC tabanlı nopCommerce altyapısı kullanılarak oluşturulmuştur. Uygulama, Visual Studio üzerinde Kestrel web sunucusu ile geliştirici modunda çalıştırılmıştır. Bu yöntem, platformun kaynak koduna doğrudan erişim ve özelleştirilmiş test senaryolarının kolayca entegre edilmesini sağlamıştır.

Tez çalışması, yalnızca teknik açıkların tespitiyle sınırlı kalmayıp, aynı zamanda güvenlik zafiyetlerinin yasal ve etik boyutlarını da ele alınmıştır. Bu kapsamda, güvenlik testi süreçlerinde uyulması gereken etik kurallar, yasal sorumluluklar ve veri gizliliği ilkeleri gözetilmiştir.

2. LİTERATÜR ÖZETİ

2.1 E-Ticaret Mimarisinin Yapısı

E-ticaret sistemleri, dijitalleşmenin yaygınlaşmasıyla birlikte modern ekonomilerin temel yapı taşlarından biri hâline gelmiştir. Bu sistemler; tüketicilerin, firmaların ve kurumların ürün veya hizmetlere çevrimiçi erişimini mümkün kılan çok katmanlı bir teknoloji altyapısına sahiptir. Ancak bu sistemlerin sağlıklı ve güvenli çalışabilmesi, mimari yapının bileşenlerinin doğru anlaşılması ve bu bileşenlere yönelik güvenlik önlemlerinin bütüncül bir yaklaşımla değerlendirilmesi ile mümkündür .

E-ticaret mimarisi genel anlamda dört temel katmandan oluşur: web sunucuları**,** veritabanları**,** ödeme sistemleri **ve** kullanıcı arayüzleri. Bu katmanların her biri, sistemin fonksiyonelliğini sağlarken aynı zamanda potansiyel siber tehditlere de açık hâle gelmektedir.Aşağıda e-ticaret sistemlerinin katmanlarını gösteren grafiği ve detaylı açıklamaları yer almaktadır.



Şekil 2.1: E – Ticaret Sistemlerinin Mimarisindeki Katmanlar

- **Web Sunucusu Katmanı**, istemciden gelen HTTP/HTTPS isteklerini işleyen, kullanıcı arayüzü ile sistemin çekirdek yapısı arasında köprü görevi gören birimdir. Apache, Nginx gibi açık kaynak kodlu sunucu yazılımları sıklıkla tercih edilmektedir. Sunucu taraflı uygulamalarda yapılan kodlama hataları ya da eksik yapılandırmalar, saldırganlar tarafından manipüle edilerek DDoS, Directory Traversal ya da Remote Code Execution gibi saldırılara zemin hazırlayabilmektedir (OWASP, 2023). Bu bağlamda,

sunucu yazılımının güncel tutulması, log takibinin yapılması ve güvenlik duvarlarının (WAF) etkin şekilde yapılandırılması kritik önem taşımaktadır.

Web sunucusu, istemcilerden (genellikle tarayıcılar aracılığıyla gelen) gelen HTTP/HTTPS isteklerini alarak bunlara uygun yanıtları dönen temel bileşendir. Kullanıcıların e-ticaret sitesine eriştiği ilk temas noktasıdır. Apache, Nginx, LiteSpeed ve Microsoft IIS gibi çeşitli sunucu yazılımları, bu katmanın yapı taşlarını oluşturur. Web sunucuları, HTML belgeleri, stil dosyaları (CSS), komut dosyaları (JavaScript) ve medya içerikleri gibi web kaynaklarını sunar. Bununla birlikte, dinamik içeriklerin oluşturulması için sunucu tarafı betik dillerine (PHP, Python, Node.js, vb.) destek sunması da beklenir. Web sunucusunun performansı, kullanıcı deneyimi açısından doğrudan etkileyici bir unsurdur. Yüksek trafiğe sahip e-ticaret platformlarında bu katmanın optimizasyonu, sayfa yükleme sürelerini düşürmek ve kesintisiz hizmet sağlamak açısından kritik öneme sahiptir.

Güvenlik açısından bakıldığında, web sunucusu katmanı sıklıkla siber saldırıların hedefi hâline gelir. DDoS (Distributed Denial of Service) saldırıları, dizin taramaları, komut enjeksiyonu ve yetkisiz erişim girişimleri bu katmanda yaygındır. Bu nedenle, güvenlik duvarı (WAF – Web Application Firewall), SSL/TLS protokollerinin uygulanması, dizin erişimlerinin sınırlandırılması ve oturum yönetimi gibi güvenlik önlemleri bu katmanda mutlaka uygulanmalıdır.

- **Veritabanı Katmanı**, e-ticaretin operasyonel yükünü taşıyan ve sistemin en hassas bilgilerini saklayan yapıdır. Müşteri bilgileri, sipariş geçmişi, ürün verileri ve ödeme kayıtları gibi hayati bilgiler burada tutulur. SQL Injection gibi klasik saldırılar, hâlen en yaygın güvenlik tehditleri arasında yer almakta olup, girdi doğrulama ve sorgu filtreleme sistemlerinin yetersizliği ciddi veri ihlallerine neden olabilmektedir. Veritabanı erişim yetkilerinin minimum düzeyde tanımlanması, verilerin şifrelenerek saklanması ve düzenli yedekleme politikalarının uygulanması bu katmanın güvenliğini sağlamak açısından zorunludur (NIST SP 800-53, 2022).

  Veritabanı katmanı, e-ticaret sisteminin bilgi yönetim merkezidir. Ürün bilgileri, kullanıcı hesapları, sipariş geçmişi, stok durumu, kampanyalar ve müşteri geri bildirimleri gibi tüm veriler bu katmanda depolanır ve

yönetilir. Genellikle ilişkisel veritabanı sistemleri (MySQL, PostgreSQL, Microsoft SQL Server) ya da büyük veri ihtiyacına göre NoSQL çözümleri (MongoDB, Cassandra) tercih edilir. Veritabanı tasarımında verinin bütünlüğü, tutarlılığı ve erişilebilirliği temel önceliklerdir. Özellikle e-ticaret sistemlerinde stok yönetimi ve kullanıcı işlemleri gibi yüksek frekanslı veri alışverişi, veritabanı performansını doğrudan etkiler. Bu nedenle, veri indeksleme, normalizasyon ve yedekleme stratejileri sistemin sürekliliği açısından hayati öneme sahiptir.

Siber güvenlik açısından veritabanı katmanı, ciddi tehditlerle karşı karşıyadır. Özellikle SQL Injection saldırıları, veritabanına doğrudan erişimi mümkün kılan kritik açıklardandır. Saldırganlar, kullanıcıdan gelen girdi verilerinin doğrulanmaması durumunda, zararlı SQL komutları ile veritabanı üzerinde yetkisiz işlemler gerçekleştirebilir. Bu tür tehditlerin önlenebilmesi için kullanıcı girişlerinin sunucu tarafında doğrulanması, parametrik sorgular kullanılması ve veritabanı kullanıcılarına minimum yetki prensibiyle erişim izni verilmesi gerekmektedir. Ayrıca, hassas verilerin şifrelenmesi ve belirli aralıklarla zafiyet taraması yapılması da önerilen uygulamalar arasındadır.

- **Ödeme Sistemleri Katmanı**, e-ticaretin finansal boyutunu oluşturan ve güvenliğin en üst düzeyde sağlanması gereken yapıdır. Bu katman genellikle üçüncü taraf ödeme sağlayıcıları (PayU, Iyzico, PayPal gibi) ile entegre biçimde çalışır. Ödeme sistemlerinin, PCI DSS (Payment Card Industry Data Security Standard) gibi uluslararası standartlara uygun olarak yapılandırılması gerekmektedir. Bu standartlar, kart verilerinin şifrelenmesi, işlem doğrulama süreçlerinin çift faktörlü yapılması ve loglama işlemlerinin bütünlüğünün sağlanmasını zorunlu kılmaktadır (PCI DSS, 2023).

  E-ticaretin dijital doğasına rağmen kullanıcılar ile satıcılar arasında kurulan ticari ilişkinin temelinde ödeme işlemleri yer almaktadır. Ödeme sistemleri katmanı, bu işlemlerin hızlı, güvenli ve şeffaf bir şekilde gerçekleştirilmesini sağlar. Kredi kartı, banka havalesi, dijital cüzdanlar (örneğin Apple Pay, Google Pay, PayPal) ve kripto paralar gibi çeşitli ödeme yöntemleri bu katmanda yönetilir. Ödeme sistemlerinin güvenliği, PCI DSS (Payment Card Industry Data Security Standard) gibi uluslararası standartlarla belirlenmiştir. Bu standartlar, kart bilgilerinin şifrelenerek

saklanmasını, kimlik doğrulamanın çok adımlı yapılmasını ve işlem geçmişinin detaylı biçimde log'lanmasını zorunlu kılar. Ayrıca üçüncü taraf ödeme sağlayıcılarla entegrasyon yapılırken, veri aktarım protokollerinin şifrelenmiş olması ve karşılıklı kimlik doğrulama mekanizmalarının kullanılması gereklidir.

Bu katmanda yaygın olarak karşılaşılan tehditler arasında sahte ödeme siteleri (phishing), kart numarası kopyalama (carding), ödeme adımlarının manipülasyonu ve ödeme sonrası kimlik bilgilerine ulaşılması gibi riskler yer almaktadır. E-ticaret sitesi geliştiricilerinin, bu katmanda gelişmiş dolandırıcılık tespit sistemleri, işlem doğrulama API'leri ve zaman aşımı politikaları uygulamaları önerilmektedir. Ayrıca, 3D Secure sistemlerinin aktif hâle getirilmesi, ödeme işlemleri sırasında ekstra güvenlik katmanı sağlamaktadır.

- **Kullanıcı Arayüzü Katmanı**, e-ticaret sitelerinin görünür yüzüdür ve doğrudan kullanıcı etkileşiminin gerçekleştiği bölümdür. Kullanıcı arayüzleri, yalnızca görsel estetikle değil aynı zamanda güvenlik protokolleriyle de doğrudan ilişkilidir. Kullanıcılardan gelen veri girişlerinin filtrelenmemesi, XSS (Cross-Site Scripting) saldırılarına zemin hazırlayabilir. Bu nedenle, istemci tarafı ve sunucu tarafı girdi doğrulama sistemlerinin çift yönlü uygulanması, bu tür açıkların önlenmesinde kritik rol oynamaktadır (Akı & Yılmaz, 2021).

  Kullanıcı arayüzü (UI – User Interface) katmanı, e-ticaret sisteminin son kullanıcıyla doğrudan etkileşime geçtiği görsel ve işlevsel yüzeydir. Bu katman, kullanıcıların ürünleri görüntülemesi, sepet oluşturması, ödeme işlemlerini gerçekleştirmesi, yorum yapması ve kişisel hesap bilgilerine erişmesi gibi işlemleri gerçekleştirdiği temel etkileşim alanıdır. Web tabanlı e-ticaret sistemlerinde HTML, CSS ve JavaScript gibi teknolojilerle oluşturulan bu katman, mobil platformlarda ise native ya da hibrit uygulamalarla desteklenmektedir. UI tasarımında kullanılabilirlik, erişilebilirlik ve kullanıcı deneyimi (UX) kadar, güvenlik de ön planda tutulmalıdır. Kullanıcıların sistemle olan etkileşimini yöneten bu katmanda yapılan küçük bir hata, kritik güvenlik açıklarına neden olabilmektedir. Örneğin, istemci tarafında yapılan form doğrulamaları sunucu tarafında

yinelenmezse, XSS (Cross-Site Scripting) gibi saldırılar için zemin hazırlanmış olur. Bu tür saldırılarla kullanıcıya ait oturum bilgileri ele geçirilebilir, yönlendirme yapılabilir ya da kötü amaçlı komutlar çalıştırılabilir.

Kullanıcı arayüzü katmanında güvenlik açıklarının önlenmesi için temel ilkelerden biri "input validation" yani kullanıcıdan alınan tüm girdilerin güvenlik açısından filtrelenmesidir. Özellikle arama kutuları, yorum alanları, giriş ve kayıt formları gibi veri girişi yapılan bileşenler hem istemci hem de sunucu tarafında kontrol edilmelidir. Ayrıca, URL yönlendirmeleri, buton işlevleri ve JavaScript komutlarının kötüye kullanımını engellemek için güvenli kodlama standartlarına uyulmalıdır.

Görsel arayüz tasarımının kullanıcı güven algısına doğrudan etki ettiği de unutulmamalıdır. Güvenilir bir görsel yapı, örneğin adres çubuğunda görülen "https://" uzantısı, kurum logosu, güvenlik sertifikalarının görünür hâlde sunulması gibi unsurlar, kullanıcıların sisteme olan güvenini artırır. Özellikle sahte e-ticaret sitelerine karşı kullanıcıyı bilinçlendiren bilgilendirme mesajları ve yardım sekmeleri de güvenlik bilincinin oluşmasına katkı sağlar. Ayrıca UI/UX tasarım sürecinde, kullanıcıların güvenlik uygulamalarını kolayca anlayıp uygulayabileceği bir yapı oluşturulmalıdır. Aksi hâlde, örneğin güçlü parola talebi veya iki faktörlü kimlik doğrulama sistemlerinin karmaşık tasarlanması, kullanıcıyı sistem dışına itebilir ve güvenlik uygulamalarının devre dışı kalmasına neden olabilir. Bu nedenle kullanıcı arayüzü tasarımı ile güvenlik uygulamaları arasında denge kurulması zorunludur. Bu bağlamda; kullanıcı arayüzü katmanı yalnızca bir görsel tasarım alanı değil, aynı zamanda e-ticaret sisteminin güvenlik stratejilerinin ilk uygulandığı ve kullanıcı davranışlarının yönlendirildiği katmandır. Bu katmanın tasarımında hem teknik güvenlik önlemleri hem de psikolojik güven algısı birlikte değerlendirilmelidir. Böylece kullanıcı deneyimi ile güvenli kullanım arasında sağlıklı bir denge kurulabilir.

Sonuç olarak, e-ticaret sistemlerinin güvenliği; sistemin her katmanında uygulanan teknik önlemler, standart uyumları ve kullanıcı etkileşimlerinin bütüncül bir güvenlik politikası çerçevesinde değerlendirilmesiyle sağlanabilir. E-ticaret mimarisinin tüm

bileşenlerinin entegre biçimde yapılandırılması ve her bir katmanın potansiyel tehditler doğrultusunda düzenli olarak test edilmesi, sürdürülebilir bir siber güvenlik ortamı oluşturmanın temel taşlarını oluşturmaktadır.

2.2 Güvenlik İhlalleri ve Etkileri

E-ticaret sistemlerinde yaşanan güvenlik ihlalleri, dijital ekonomi içerisinde yalnızca teknolojik bir tehdit değil, aynı zamanda ekonomik, sosyal ve hukuki bir sorun alanı olarak karşımıza çıkmaktadır. Günümüzde çevrim içi alışverişin yaygınlaşması, kullanıcı bilgileri ve finansal verilerin dijital ortamlarda işlenmesini zorunlu hâle getirmiştir. Bu durum, e-ticaret sitelerini siber saldırganlar açısından cazip hedefler hâline getirmekte; özellikle veri sızıntısı, kimlik hırsızlığı ve ödeme dolandırıcılığı gibi güvenlik ihlalleri ciddi sonuçlar doğurmaktadır. Bu bağlamda, ihlallerin türlerinin tanımlanması ve etkilerinin değerlendirilmesi, güvenli e-ticaret altyapılarının geliştirilmesi için temel bir gereklilik hâline gelmiştir.

Veri sızıntısı, e-ticaret sistemlerinde en sık karşılaşılan ve en yıkıcı sonuçlara neden olabilen güvenlik açıklarından biridir. Kullanıcıların ad, soyad, adres, e-posta, telefon numarası gibi kişisel bilgilerinin yanı sıra, kredi kartı bilgileri, şifreler ve alışveriş geçmişi gibi hassas verilerin yetkisiz kişilerce ele geçirilmesi, veri sızıntısının tipik bir örneğidir. Bu tür bir ihlal, yalnızca bireylerin kişisel mahremiyetini tehdit etmekle kalmaz; aynı zamanda şirketlerin yasal sorumluluk altına girmelerine, idari para cezalarına ve marka imajının zedelenmesine de yol açar. Avrupa Birliği'nin GDPR (General Data Protection Regulation) gibi düzenlemeleri, veri güvenliği ihlallerine karşı sıkı denetimler ve ağır yaptırımlar getirmiştir. Dolayısıyla, veri sızıntısı yalnızca teknik bir sorun değil, aynı zamanda kurumsal riskin önemli bir bileşeni olarak değerlendirilmelidir.

Kimlik hırsızlığı, veri sızıntısından türeyen ancak daha geniş ölçekli etkileri bulunan bir başka güvenlik problemidir. Saldırganlar, ele geçirdikleri kullanıcı bilgilerini kullanarak sahte kimlikler oluşturmakta, bu kimliklerle yasa dışı işlemler gerçekleştirmekte ya da kullanıcının mevcut hesabına erişerek sistem içi manipülasyonlar yapabilmektedir. Kimlik hırsızlığına bağlı dolandırıcılık olayları, yalnızca bireylerin ekonomik zarar görmesine neden olmakla kalmaz; aynı zamanda

dijital güvenliğe olan genel güvenin azalmasına da sebep olmaktadır. Çevrim içi kimlik doğrulama sistemlerinin zayıf kurgulanmasının, saldırganların sistemlere kullanıcı gibi erişmesini kolaylaştırdığını ve bu nedenle çok faktörlü kimlik doğrulama günümüzde yaygın olarak zorunlu bir güvenlik katmanı olarak kabul edilmektedir.

Ödeme dolandırıcılığı ise e-ticaret sistemlerine yönelik saldırıların doğrudan finansal zarara neden olan yönüdür. Bu tür ihlaller, sahte kredi kartı kullanımı, ödeme adımlarının manipüle edilmesi, üçüncü taraf ödeme sağlayıcıların ele geçirilmesi gibi yöntemlerle gerçekleştirilmektedir. Özellikle sahte ödeme siteleri veya 'phishing' (oltalama) yöntemiyle kullanıcıların ödeme bilgilerini ele geçirme girişimleri yaygındır. PCI DSS standartları, ödeme sistemlerinin güvenliğini sağlamak adına kart verilerinin şifrelenmesi, saklanmaması ve yalnızca doğrulanabilir işlemlerle sınırlı kalması gerektiğini belirtmektedir (PCI DSS, 2023). Tüm bu önlemlere rağmen ödeme güvenliğinin sağlanması, sistematik izleme, anormallik tespiti ve kullanıcı davranış analizlerinin düzenli olarak yapılmasını gerektirmektedir.

Güvenlik ihlallerinin etkileri sadece teknik ya da finansal düzeyde kalmamaktadır. Psikolojik etkiler, kullanıcı davranışlarında değişim, pazar dinamiklerinde daralma ve regülasyon baskısının artması gibi sonuçlar da doğurabilmektedir. Kullanıcılar, daha önce güvenlik açığı yaşanan bir e-ticaret sitesine tekrar dönmemekte, bu da müşteri sadakatinde azalma anlamına gelmektedir. Aynı zamanda firmalar, güvenlik ihlalleri sonrası denetim süreçlerine maruz kalmakta, bu durum da operasyonel maliyetlerin artmasına neden olmaktadır. Yüksek maliyetli tazminatlar, hukuki süreçler ve medya yansımaları ise kurumların uzun vadeli stratejik hedeflerini olumsuz etkileyebilmektedir. Dolayısıyla, güvenlik ihlalleri yalnızca bir bilgi güvenliği meselesi değil, aynı zamanda kurumsal sürdürülebilirlik sorununa dönüşmektedir.

Sonuç olarak, e-ticaret sistemlerinde güvenlik ihlalleri çok boyutlu ve disiplinler arası bir sorun alanı olarak ele alınmalıdır. Veri sızıntısı, kimlik hırsızlığı ve ödeme dolandırıcılığı gibi temel tehdit türleri, yalnızca teknik önlemlerle değil; aynı zamanda hukuki, yönetsel ve kullanıcı farkındalığına dayalı önlemlerle de kontrol altına alınmalıdır. Literatürde yer alan güncel çalışmalar, hibrit güvenlik stratejilerinin (örneğin davranışsal analiz + teknik filtreleme) bu tehditlerle mücadelede daha etkili sonuçlar verdiğini göstermektedir (Zhou ve ark., 2022). Bu

bağlamda, güvenlik ihlallerine karşı geliştirilecek önleme modellerinin sadece sistem merkezli değil, bütünsel bir güvenlik perspektifiyle tasarlanması gerekmektedir.

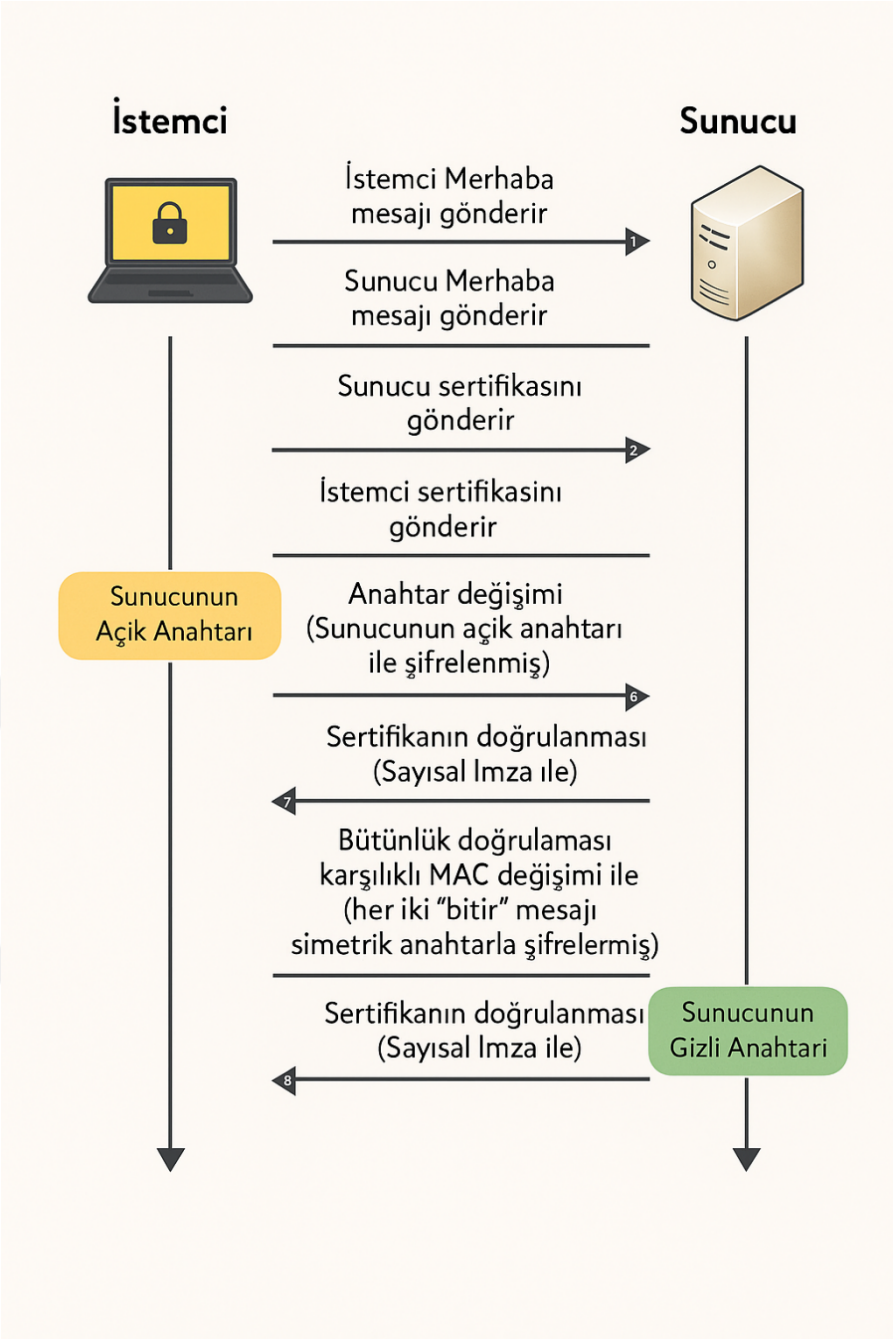
2.3 Mevcut Güvenlik Önlemleri

E-ticaret sistemlerinin dijital dünyada kazandığı öneme paralel olarak, bu platformlara yönelik siber tehditlerin sayısı ve çeşitliliği de artmıştır. Kullanıcı bilgileri, ödeme verileri ve ticari sırlar gibi hassas verilerin barındırıldığı bu sistemler, saldırganlar için oldukça değerli hedefler hâline gelmiştir. Bu nedenle, e-ticaret altyapılarında kullanılan güvenlik önlemleri, sadece teknik açıdan değil; aynı zamanda yönetsel, hukuki ve kullanıcı odaklı yaklaşımları da kapsayan çok boyutlu stratejiler içermelidir. Bu bölümde, günümüzde yaygın olarak kullanılan başlıca güvenlik önlemleri olan **SSL/TLS protokolleri**, **CAPTCHA sistemleri**, **iki faktörlü kimlik doğrulama (2FA)**, **güvenli yazılım geliştirme teknikleri**, **güvenlik duvarları**, **sızma testleri** ve **veri şifreleme politikaları** detaylı biçimde ele alınacaktır. Bu önlemler, bir yandan sistemin bütünlüğünü ve sürekliliğini sağlarken, diğer yandan kullanıcıların dijital güvenlik algısını da doğrudan etkilemektedir.

2.3.1 SSL/TLS Protokolleri

SSL (Secure Sockets Layer) ve daha gelişmiş sürümü olan TLS (Transport Layer Security), istemci ile sunucu arasında gerçekleştirilen veri iletişiminin şifrelenmesini sağlayan güvenlik protokolleridir. Özellikle kullanıcı girişi, ödeme işlemleri ve kişisel veri formlarının yer aldığı sayfalarda bu protokollerin kullanımı hayati önem taşır. SSL/TLS kullanımı sayesinde, iletilen veriler üçüncü şahıslar tarafından okunamaz, değiştirilemez veya taklit edilemez.

E-ticaret sitelerinde SSL sertifikası olmayan bir sayfanın kullanıcı gözünde güvensiz görünmesi, kullanıcı kaybına da neden olabilir. Tarayıcıların “güvenli değil” uyarıları, özellikle ödeme işlemlerinde kullanıcıların işlemi yarıda kesmesine yol açmaktadır. Bu nedenle tüm e-ticaret altyapıları, TLS 1.2 veya tercihen TLS 1.3 sürümünü desteklemeli, sertifikalarını güvenilir otoritelerden almalı ve düzenli olarak güncellemelidir.



Şekil 2.2. SSL Çalışma Döngüsü

SSL/TLS kullanımının kendisi tek başına yeterli değildir. Doğru konfigürasyon yapılmayan bir TLS bağlantısı, downgrade saldırılarına veya zayıf şifreleme algoritmalarına açık hâle gelebilir. Bu yüzden, yalnızca sertifika sağlamak değil, güvenli konfigürasyonlarla çalışmak da sistem yöneticilerinin sorumluluğundadır.

2.3.2 CAPTCHA Sistemleri

CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart), insan kullanıcıları otomatik bot yazılımlarından ayırt etmek amacıyla geliştirilen bir güvenlik mekanizmasıdır. Özellikle e-ticaret platformlarında kayıt formları, giriş sayfaları, ödeme adımları ve yorum alanlarında kullanılan CAPTCHA sistemleri; brute-force saldırılar, spam içerik üretimi ve otomatik komut yürütme girişimlerine karşı etkili bir savunma sağlar. Klasik CAPTCHA sistemlerinde kullanıcıdan, bozulmuş metin karakterlerini tanımlaması, basit matematiksel işlemleri çözmesi veya belirli görselleri seçmesi beklenir. Bu testler sayesinde yalnızca gerçek insanlar tarafından tamamlanabilecek görevler sunulur ve sistemin kötüye kullanılması büyük ölçüde engellenmiş olur. Ancak zamanla bu tür CAPTCHA'lar, saldırganlar tarafından geliştirilen yapay zeka destekli botlar ile aşılabilir hâle gelmiştir. Bu durum, güvenlik uzmanlarını daha gelişmiş doğrulama mekanizmaları geliştirmeye yönlendirmiştir. Google tarafından geliştirilen reCAPTCHA sistemleri, bu evrimin en bilinen örneklerinden biridir. Özellikle reCAPTCHA v3, kullanıcılara herhangi bir test sunmaksızın arka planda etkileşimlerini analiz ederek bir risk puanı oluşturur. Bu sayede kullanıcı deneyimini kesintiye uğratmadan güvenlik sağlanmış olur. Ancak bu sistemlerin de kullanıcı gizliliği üzerindeki etkileri tartışılmaktadır; çünkü sistem, kullanıcının davranışsal verilerini izleyerek karar vermektedir.

CAPTCHA teknolojilerinin başarısı, yalnızca algoritmanın doğruluğuna değil; aynı zamanda sistemin nasıl entegre edildiğine ve güncel tehditlere karşı ne kadar adapte edilebildiğine bağlıdır. E-ticaret platformları, CAPTCHA sistemlerini diğer güvenlik önlemleriyle birlikte düşünmeli ve zamanla kullanıcıları yormayacak, ancak güvenlik düzeyini koruyacak çözümleri tercih etmelidir. Bu bağlamda görsel ve mantıksal testlerin çeşitlendirilmesi, kullanıcı deneyimi ile güvenlik arasında dengeli bir yapı kurulmasını sağlar. Sonuç olarak, CAPTCHA sistemleri, otomasyon temelli tehditleri engellemede etkili bir araçtır. Ancak günümüzün gelişmiş bot teknolojileri karşısında sürekli güncellenmeleri ve diğer doğrulama sistemleriyle desteklenmeleri gerekmektedir.

2.3.3 İki Faktörlü Kimlik Doğrulama (2FA)

İki Faktörlü Kimlik Doğrulama (2FA), kullanıcı kimliğini doğrulamak için yalnızca bir şifreyle yetinmeyen, ikinci bir doğrulama adımı ekleyen güvenlik yöntemidir. E-ticaret platformları açısından kritik öneme sahip olan bu yöntem, özellikle hesap güvenliği, ödeme işlemleri ve kişisel bilgilerin değiştirilmesi gibi hassas işlemlerde ek bir koruma katmanı sağlar.

Geleneksel kimlik doğrulama sistemleri, yalnızca kullanıcı adı ve parola kombinasyonuna dayanır. Ancak parolaların ele geçirilmesi, tahmin edilmesi ya da sosyal mühendislik yoluyla ifşa edilmesi sık karşılaşılan tehditler arasındadır. Bu nedenle 2FA, parolanın yanı sıra kullanıcının yalnızca kendisinde bulunan veya sadece kendisinin bildiği bir unsurla kimliğini tekrar doğrulamasını ister.

2FA yöntemleri genellikle üç temel doğrulama kategorisinden biriyle çalışır:

- Kullanıcının bildiği bir şey (örneğin şifre, PIN),
- Kullanıcının sahip olduğu bir şey (örneğin cep telefonuna gelen SMS kodu, mobil onay uygulaması, güvenlik anahtarı),
- Kullanıcının kendisiyle ilgili bir şey (örneğin parmak izi, yüz tanıma, iris taraması gibi biyometrik veriler).

E-ticaret sistemlerinde en yaygın kullanılan 2FA yöntemleri arasında, SMS yoluyla gönderilen tek kullanımlık doğrulama kodları (OTP), mobil uygulama (örneğin Google Authenticator, Microsoft Authenticator) ile oluşturulan zaman tabanlı kodlar ve e-posta ile gönderilen doğrulama bağlantıları yer almaktadır. Daha ileri düzeyde güvenlik için donanımsal güvenlik anahtarları veya parmak izi doğrulaması da tercih edilebilir. 2FA sistemlerinin uygulanması, kullanıcı güvenliğini artırmakla birlikte, kullanıcı deneyimi açısından bazı zorlukları da beraberinde getirebilir. Kodun ulaşmaması, doğrulama sürecinin karmaşık algılanması veya internet bağlantısı gerektirmesi gibi durumlar, kullanıcıların sistemden uzaklaşmasına neden olabilir. Bu nedenle kullanıcı dostu, anlaşılır ve hızlı çalışabilecek 2FA entegrasyonları tercih edilmelidir. Sonuç olarak, 2FA e-ticaret sistemlerinde hesap güvenliğini sağlamak için etkili ve artık neredeyse zorunlu bir önlemdir. Parola güvenliğinin yetersiz kaldığı günümüzde, 2FA yalnızca ek bir katman değil, temel bir ihtiyaç hâline gelmiştir. Özellikle yönetici hesaplarında ve ödeme sistemlerinde bu doğrulamanın

aktif hâle getirilmesi, sistemin genel güvenlik seviyesini ciddi ölçüde yükseltmektedir.

2.3.4 Güvenli Yazılım Geliştirme Teknikleri

E-ticaret sistemlerinin güvenliği yalnızca dışarıdan uygulanan savunma mekanizmalarıyla değil, yazılımın tasarım ve geliştirme aşamalarında alınan önlemlerle de sağlanmalıdır. Bu bağlamda, güvenli yazılım geliştirme teknikleri, sistemin en baştan itibaren güvenlik prensiplerine uygun şekilde kurgulanmasını hedefler. "Security by Design" (tasarımda güvenlik) ilkesi doğrultusunda geliştirilen yazılımlar, daha az açık içerir ve dış tehditlere karşı daha dayanıklıdır. Güvenli yazılım geliştirme süreci, yazılım yaşam döngüsünün (SDLC – Software Development Life Cycle) tüm aşamalarına entegre edilmelidir. Bu aşamalar; ihtiyaç analizi, sistem tasarımı, kodlama, test, dağıtım ve bakım olarak sıralanır. Her aşamada potansiyel güvenlik risklerinin analiz edilmesi ve gerekli önlemlerin alınması gerekmektedir.

Kodlama aşamasında geliştiriciler, güvenli kodlama standartlarına uymalıdır. Örneğin kullanıcıdan alınan veriler doğrudan veritabanına gönderilmeden önce doğrulanmalı ve filtrelenmelidir. Aksi takdirde SQL injection, XSS veya CSRF gibi açıklara zemin hazırlanabilir. OWASP (Open Web Application Security Project) tarafından yayımlanan **OWASP Top 10** listesi, geliştiricilere en yaygın güvenlik açıkları konusunda yol gösterir. Bu liste, e-ticaret sistemleri gibi yüksek riskli uygulamalar için mutlaka dikkate alınmalıdır.

Güvenli yazılım geliştirme sürecinde kullanılan bazı önemli teknikler şunlardır:

- **Tehdit modelleme**: Sistemdeki olası saldırı yüzeylerinin tanımlanması ve bu alanlara yönelik önlemlerin planlanması,
- **Kod inceleme (code review)**: Başka bir geliştirici veya otomatik araç tarafından yazılım kodunun gözden geçirilmesi,
- **Statik ve dinamik analiz araçları**: Kodun çalıştırılmadan önce (statik) ve çalışırken (dinamik) test edilmesi,
- **Birim testleri ve güvenlik testleri**: Yazılımın modül bazında ve tüm sistemde güvenlik açısından sınanması.

Ayrıca, üçüncü taraf kütüphanelerin ve açık kaynak bileşenlerin kullanımı sırasında bu bileşenlerin güncel ve güvenli olduğundan emin olunmalıdır. Aksi takdirde, dış bileşenlerdeki açıklardan tüm sistem etkilenebilir. Yazılım tedarik zinciri güvenliği de bu nedenle önem kazanmıştır. Sonuç olarak, güvenli yazılım geliştirme teknikleri, e-ticaret sistemlerinin sadece bugün için değil, uzun vadede de korunmasını sağlayan temel bir yaklaşımdır. Saldırılar gerçekleştikten sonra değil, daha sistem inşa edilirken önlem almak; maddi kayıpları, itibar zedelenmesini ve hukuki sorumlulukları büyük ölçüde engelleyecektir.

2.3.5 Diğer Yaygın Güvenlik Uygulamaları: Güvenlik Duvarı, Sızma Testi, Veri Şifreleme

E-ticaret sistemlerinde güvenliği sağlamak için temel protokollerin yanı sıra, çeşitli tamamlayıcı uygulamalar da büyük önem taşımaktadır. Güvenlik duvarları, sızma testleri ve veri şifreleme yöntemleri bu tür tamamlayıcı uygulamaların başında gelir. Bu tekniklerin her biri, sistemin farklı zayıf noktalarını hedef alarak genel güvenlik seviyesini yükseltmeyi amaçlar.

Güvenlik duvarları (firewall)**,** iç ve dış ağ trafiğini kontrol eden, belirli kurallar çerçevesinde şüpheli hareketleri engelleyen bir sistemdir. E-ticaret platformlarında genellikle web uygulama güvenlik duvarı (WAF) kullanılır. WAF'lar, özellikle HTTP trafiğini izler ve SQL injection, XSS gibi uygulama katmanı saldırılarını tespit edip önlemeye çalışır. Ayrıca, gelen verinin filtrelenmesi, oturum yönetimi ve IP kısıtlamaları gibi ek güvenlik önlemleri de bu sistemler aracılığıyla uygulanabilir.

Sızma testleri (penetrasyon testleri) ise bir sistemin dışarıdan gelen tehditlere ne kadar açık olduğunu anlamak amacıyla yapılan kontrollü saldırı simülasyonlarıdır. Bu testler, manuel olarak uzmanlar tarafından ya da otomatik araçlar yardımıyla gerçekleştirilebilir. Sızma testleri sonucunda, mevcut zafiyetler detaylı bir rapor hâlinde sunulur ve bu açıkların nasıl kapatılması gerektiği önerilir. E-ticaret siteleri gibi kritik altyapılar, belirli aralıklarla bu testlere tabi tutulmalı ve elde edilen bulgular ışığında sistem güncellenmelidir. Ayrıca bazı ülkelerde büyük çaplı dijital hizmet sağlayıcılarının periyodik sızma testi yaptırmaları yasal bir zorunluluktur.

Veri şifreleme ise hem kullanıcı verilerinin hem de işlem bilgilerinin korunması açısından temel bir güvenlik uygulamasıdır. Şifreleme, verinin yalnızca yetkili kişiler

tarafından okunabilir olmasını sağlar. E-ticaret sistemlerinde en çok karşılaşılan uygulamalar arasında AES (Advanced Encryption Standard**)** gibi simetrik şifreleme yöntemleri **ve** RSA gibi asimetrik şifreleme teknikleri yer alır. Özellikle kredi kartı bilgileri, parola verileri ve kişisel kimlik bilgileri gibi hassas içerikler, hem veri tabanında hem de veri aktarımı sırasında mutlaka şifrelenmelidir.

Bunlara ek olarak, log kayıtlarının bütünlüğünün sağlanması, sistem izleme araçlarının kullanılması ve olay müdahale planlarının oluşturulması da güvenliğin sürdürülebilirliği açısından kritik öneme sahiptir. Güvenlik sadece kurulumla sınırlı bir işlem değil, sürekli izleme, test ve iyileştirme gerektiren dinamik bir süreçtir.

3. GÜVENLİK AÇIKLARI: TÜRLERİ VE ANALİZİ

E-ticaret platformları, sahip oldukları çok katmanlı yapılar ve geniş kullanıcı etkileşimi nedeniyle siber saldırıların en yaygın hedeflerinden biridir. Her bir bileşeni, hem sistem içi hem de dışa açık olarak çalıştığı için çeşitli güvenlik açıkları barındırma riski taşır. Bu bölümde, e-ticaret sistemlerinde karşılaşılan başlıca güvenlik açıkları detaylı biçimde incelenemiş olup ; uluslararası ölçekte kabul gören OWASP Top 10 listesi temel alınarak açıklamalar ve e-ticaret özelinde en sık görülen zafiyet türleri ile bunların tespit yöntemleri değerlendirilmiştir.

3.1 OWASP Top 10 Açıkları

OWASP (Open Web Application Security Project), dünya genelinde web uygulamalarının güvenliğine ilişkin en yaygın ve güncel tehditleri sistematik bir şekilde sınıflandırarak, yazılım geliştiricilere ve sistem yöneticilerine rehberlik eden bir organizasyondur. “OWASP Top 10” listesi, web uygulamalarında en sık karşılaşılan güvenlik açıklarını belirtmekte ve bu açıkların oluş nedenleri, etkileri ve önleme yöntemleriyle birlikte düzenli olarak güncellenmektedir. Bu liste, e-ticaret sistemlerinin güvenliğinde dikkate alınması gereken temel referanslardan biridir. Aşağıda, OWASP’in yayımladığı son liste esas alınarak her bir açıklık, e-ticaret bağlamında detaylı biçimde ele alınmaktadır.

Broken Access Control (Yetki Kontrolü İhlalleri)

E-ticaret platformlarında kullanıcı rollerine göre içerik ve işlemlere sınırlı erişim sağlanması kritik bir güvenlik unsurudur. Ancak bazı durumlarda, geliştiricilerin rol bazlı kontrolleri eksik tanımlaması ya da atlaması, sıradan kullanıcıların yönetici seviyesindeki işlemleri gerçekleştirmesine imkân tanıyabilir. Örneğin, sipariş geçmişini yalnızca ilgili kullanıcı görebilecekken, ID manipülasyonuyla başkasına ait kayıtların da görüntülenebilmesi erişim yetkisi ihlallerinin yaygın bir örneğidir.

Cryptographic Failures (Kriptografik Hatalar)

Bu açıklık, verilerin gizliliği, bütünlüğü ve güvenli iletimi açısından kritik rol oynayan kriptografik yapıların yetersiz veya yanlış uygulanmasından kaynaklanır. Şifreleme algoritmaları, dijital ticaretin temel güvenlik katmanlarından biridir. Ancak bu yapıların yanlış uygulanması ya da eskimiş tekniklerle kurulması, hassas bilgilerin

risk altında kalmasına neden olur. Örneğin, kullanıcı şifrelerinin SHA-1 gibi kırılması kolay algoritmalarla tutulması veya kart bilgileri aktarılırken HTTPS yerine HTTP kullanılması, potansiyel veri sızıntılarına yol açabilir. Modern sistemlerde TLS 1.3 ve AES gibi güncel standartlara uyum sağlanmalıdır.

Injection (Enjeksiyon Saldırıları)

En yaygın ve en tehlikeli açık türlerinden biri olan enjeksiyon saldırıları, saldırganların uygulamaya beklenmeyen veri türlerini (örneğin SQL komutları) enjekte etmesiyle oluşur. Özellikle kullanıcıdan gelen verilerin doğrudan sorgulara aktarılması, enjeksiyon riskini doğurur. Bir e-ticaret sitesinde arama kutusuna 'DROP TABLE users;–' gibi bir ifade girilmesi ve bunun doğrudan sorguda işlenmesi, sistemin tamamını felç edebilir. Bu gibi durumların önlenmesi için giriş doğrulama, input sanitizasyonu ve parametrik sorgular temel savunma hatlarını oluşturmalıdır.

Insecure Design (Güvensiz Tasarım)

Sistemlerin mimari düzeyde güvenlik prensipleri gözetilmeden tasarlanması bu açıklık kapsamında değerlendirilir. Örneğin, ödeme işlemlerinde işlem doğrulaması yapılmadan yalnızca istemci tarafındaki bilgilerin güvenilmesi, tasarımsal bir güvenlik zafiyetidir. E-ticaret yazılımlarında "Security by Design" yaklaşımı benimsenmeli ve her modül daha geliştirilmeden önce tehdit modellemeleri yapılmalıdır. Sadece uygulama aşamasındaki hatalar değil, sistemin ilk tasarım adımlarında yapılan güvenlik ihmalleri de büyük tehditlere kapı aralar. Örneğin; yalnızca istemci tarafında ödeme doğrulaması yapılması, manipülasyona açık sistemlerin temelini oluşturur. Bu nedenle güvenli mimari modellemesi, e-ticaret sistemlerinin planlama aşamasında ele alınmalı ve tehdit modellemesi ile desteklenmelidir.

Security Misconfiguration (Güvenlik Yapılandırma Hataları)

Sistem bileşenlerinin varsayılan ayarlarla bırakılması, gereksiz servislerin açık tutulması, hata mesajlarının ayrıntılı bilgi içermesi ve dosya/dizin izinlerinin yanlış yapılandırılması gibi durumlar bu kategoride yer alır. E-ticaret sistemlerinde, özellikle açık kaynaklı platformlarda, kurulum sonrası yapılandırmalar detaylı olarak gözden geçirilmeli, gereksiz servisler kapatılmalı ve sadece gerekli erişim izinleri tanımlanmalıdır. Varsayılan şifrelerin değiştirilmemesi, hata sayfalarında sistem içeriğinin açık edilmesi veya erişim izinlerinin yanlış atanması; çoğu zaman geliştiricinin gözünden kaçan ama saldırganın kolayca tespit edebileceği açıklardır. Özellikle yaygın platformlarda (Magento, WooCommerce gibi), yetersiz yapılandırma sonucunda arka kapılar oluşabilir.

Vulnerable and Outdated Components (Zayıf ve Güncellenmemiş Bileşenler)

E-ticaret sistemlerinde sıklıkla kullanılan açık kaynaklı modüller, şablonlar ve üçüncü taraf API bileşenleri zamanla güncelliğini yitirebilir. Geliştiriciler bu bileşenleri sistemlerine entegre ettikten sonra çoğu zaman güncelleme kontrolünü ihmal eder. Oysa bu bileşenlerde keşfedilen açıklıklar, saldırganlar tarafından kolayca istismar edilebilir. Örneğin, bir kampanya eklentisinin eski sürümünde bulunan XSS açığı nedeniyle tüm kullanıcı oturumları tehlikeye girebilir. Bu tür tehditlerden korunmak için sürüm takibi yapılmalı ve güvenlik yamaları zaman kaybetmeden uygulanmalıdır. Kütüphaneler, çerçeveler ve eklentiler gibi dış bileşenlerin eski sürümleri, bilinen güvenlik açıklarını içerebilir. E-ticaret sitelerinde sıklıkla kullanılan ödeme modülleri, kampanya eklentileri ya da kullanıcı arayüz bileşenleri güncel değilse sistem saldırılara açık hâle gelir. Bu nedenle tüm üçüncü taraf bileşenlerin düzenli olarak güncellenmesi ve yalnızca güvenilir kaynaklardan temin edilmesi gereklidir.

Identification and Authentication Failures (Kimlik Doğrulama Zafiyetleri)

E-ticaret sistemlerinde kullanıcı güvenliği büyük oranda kimlik doğrulama süreçlerine bağlıdır. Zayıf parola politikaları, varsayılan kullanıcı adı/parola kombinasyonlarının korunması ve oturum yönetimi eksiklikleri, hesapların kolayca ele geçirilmesine neden olabilir. Ayrıca, şifre sıfırlama süreçlerinde yeterli doğrulama yapılmaması, hesap ele geçirme vakalarının temelini oluşturur. Güvenli kimlik doğrulama; güçlü parola kurallarının yanı sıra iki faktörlü kimlik doğrulama

ve oturum izleme ile desteklenmelidir. Böylece kullanıcı hesapları sadece korunmakla kalmaz, olası ihlallerde tespit de kolaylaşır.

Zayıf parola politikaları, çok faktörlü kimlik doğrulamanın kullanılmaması, oturum yönetimi hataları ve kimlik doğrulama token'larının kötü yönetilmesi bu zafiyetin kapsamına girer. E-ticaret uygulamalarında kullanıcı kimliklerinin doğru ve güvenli şekilde doğrulanmaması, hesap ele geçirme saldırılarına yol açabilir. Bu nedenle güçlü parola kuralları uygulanmalı, oturum süresi sınırlandırılmalı ve 2FA gibi ek güvenlik önlemleri entegre edilmelidir.

Software and Data Integrity Failures (Yazılım ve Veri Bütünlüğü İhlalleri)

Yazılım güncellemelerinin doğrulanmadan sisteme entegre edilmesi, kod imzalarının atlanması ya da kritik bileşenlerin dış kaynaklardan kontrolsüz biçimde indirilmesi, e-ticaret sistemlerinin bütünlüğünü riske atar. Saldırganlar bu açıklardan yararlanarak zararlı kodları sisteme enjekte edebilir ya da güncelleme süreci sırasında araya girerek istenmeyen işlemler gerçekleştirebilir. Bu nedenle sistemde çalışan tüm bileşenlerin güvenilir kaynaklardan edinilmesi, SHA256 gibi hash doğrulamaları ile kimliklerinin kontrol edilmesi gereklidir.

Yazılım güncellemelerinin doğrulanmadan yapılması, imzasız dosyaların çalıştırılması veya içerik bütünlüğünü doğrulayan mekanizmaların eksikliği gibi durumlar sistemde ciddi güvenlik açıklarına neden olur. E-ticaret platformlarında, özellikle ödeme sağlayıcılarıyla yapılan entegrasyonlarda veri bütünlüğü kritik öneme sahiptir. Bu tür bileşenler için kod imzalama ve kontrol mekanizmaları kullanılmalıdır.

Security Logging and Monitoring Failures (Güvenlik Kayıtları ve İzleme Eksiklikleri)

Sistem olaylarının kaydedilmemesi ya da düzenli izlenmemesi, saldırıların erken tespitini imkânsız hâle getirir. Birçok e-ticaret sitesinde başarısız giriş denemeleri, veri sızıntısı belirtileri ya da şüpheli trafik hareketleri sistem günlüklerinde yer alsa da bu veriler çoğu zaman analiz edilmez. Gerçek zamanlı log analizi, SIEM (Security Information and Event Management) sistemleri ile desteklenmeli ve olağandışı davranışlar anında uyarı üretecek şekilde yapılandırılmalıdır. Böylece saldırı gerçekleşmeden önce önlem almak mümkün hâle gelir.

Sistemlerde yaşanan olayların kayıt altına alınmaması, saldırı girişimlerinin tespit edilememesine neden olur. E-ticaret sitelerinde kullanıcı oturumları, ödeme hataları, başarısız giriş denemeleri ve sistem hataları mutlaka log'lanmalı ve bu log'lar güvenli ortamlarda saklanmalıdır. Ayrıca olay yönetim sistemleri ile bu log'lar düzenli olarak izlenmeli ve olağan dışı aktiviteler hızlıca tespit edilmelidir.

Server-Side Request Forgery (SSRF)

SSRF, sistemin iç ağındaki kaynaklara dışarıdan erişim sağlanmasına neden olan bir saldırı türüdür. Özellikle bir URL üzerinden dosya yükleme, dış kaynaklardan veri çekme gibi işlevlerin denetlenmeden çalıştırıldığı durumlarda ortaya çıkar. Örneğin bir ürün görselini URL ile yükleyen sistem, saldırganın bu alanı iç IP'lere yönlendirmesine ve veritabanı sunucusu gibi iç hizmetlere ulaşmasına neden olabilir. Bu tür riskler, dış isteklere sıkı kontrol uygulanarak, whitelisting yapılması ve proxy filtreleri kullanılması ile azaltılabilir.

SSRF saldırıları, sistemin başka bir iç kaynağa yönlendirilmesini ve bu kaynaklar üzerinden yetkisiz işlemler yapılmasını sağlar. E-ticaret sistemlerinde örneğin bir görselin URL'yle yüklenmesi gibi işlemler sırasında saldırgan, sistemin iç IP'sine erişmeye çalışabilir. Bu tür saldırıların önüne geçmek için gelen isteklerin doğrulanması, kaynak IP kontrolü ve proxy sunucular üzerinden filtreleme yapılması gereklidir.

Bu OWASP Top 10 açıkları, e-ticaret sistemlerinde sık karşılaşılan ve çoğu zaman kritik zararlara yol açabilen zafiyet türleridir. Her bir açıklığın erken tespiti ve sistematik olarak giderilmesi, hem kullanıcı güvenliğini sağlamak hem de sistemin sürdürülebilirliğini artırmak adına hayati öneme sahiptir.

3.2 E-Ticaret Sitelerinde En Sık Görülen Açıklar

Günümüzde dijital ticaretin yaygınlaşması, e-ticaret platformlarını hem ekonomik hem de teknolojik açıdan kritik altyapılar hâline getirmiştir. Kullanıcı bilgilerinden ödeme sistemlerine, ürün yönetiminden sipariş süreçlerine kadar çok sayıda bileşen barındıran bu sistemler, aynı zamanda siber saldırıların da öncelikli hedefleri arasında yer almaktadır. OWASP Top 10 listesi genel web uygulamaları için

evrensel riskleri ortaya koysa da, e-ticarete özgü dinamikler belirli açıklıkların daha sık ve daha etkili görülmesine neden olmaktadır.

Bu bölümde, e-ticaret platformlarında en yaygın olarak tespit edilen güvenlik açıkları; teknik arka planları, oluşma nedenleri, etkileri ve önleme yöntemleriyle birlikte ayrıntılı biçimde analiz edilecektir.

1. SQL Injection (SQL Enjeksiyonu)

E-ticaret sitelerinde veritabanı etkileşimi neredeyse her işlemin temelinde yer alır. Ürün filtreleme, kullanıcı girişi, ödeme doğrulama, sipariş sorgulama gibi fonksiyonlar veritabanına doğrudan bağlıdır. Ancak bu süreçlerde gerekli veri doğrulama işlemlerinin yapılmaması, sistemin SQL Injection saldırılarına açık hâle gelmesine neden olur.

Bu saldırı türünde, saldırgan giriş alanlarına (örneğin arama kutusu veya giriş formu) SQL komutları enjekte ederek arka plandaki sorguların yapısını bozar. Örneğin admin' OR '1'='1 gibi bir giriş ifadesiyle sistemde oturum açılabilir. Eğer sistem bu girdiyi filtrelemeden doğrudan SQL sorgusuna yerleştirirse, veritabanındaki tüm bilgilere erişim sağlanabilir.

Etkileri:

- ✓ Tüm kullanıcı bilgilerinin sızdırılması,
- ✓ Ödeme verilerinin manipülasyonu,
- ✓ Ürün ve sipariş verilerinin silinmesi veya değiştirilmesi,
- ✓ Tam veritabanı kontrolü.

Korunma yöntemleri:

- ✓ Parametrik sorgular (prepared statements) kullanmak,
- ✓ ORM (Object-Relational Mapping) yapılarından faydalanmak,
- ✓ Girdi doğrulama ve beyaz listeleme uygulamak.

2. XSS (Cross-Site Scripting) Açıkları

XSS saldırıları, e-ticaret sistemlerinde en çok “yorum sistemleri, arama kutuları, canlı destek panelleri ve kullanıcı profilleri” gibi bileşenlerde karşımıza çıkar. Saldırgan, kötü amaçlı JavaScript kodlarını bu alanlara yerleştirerek diğer kullanıcıların tarayıcılarında bu kodların çalışmasını sağlar. Bu kodlar sayesinde kullanıcıdan veri çalınabilir, sayfa içeriği değiştirilebilir ya da kullanıcı başka bir siteye yönlendirilebilir.

Örnek bir senaryo: Saldırgan, ürün yorumuna şu şekilde bir kod ekleyebilir:

<script>document.location='http://saldirgan.com?cookie='+document.cookie</script>

Kullanıcı bu yorumu görüntülediğinde, oturum çerezi saldırgana gönderilir.

Etkileri:

- ✓ Oturum ele geçirme,
- ✓ Kimlik hırsızlığı,
- ✓ Görsel aldatmacalarla ödeme yönlendirmesi,
- ✓ Güven kaybı ve müşteri kaybı.

Önleme yöntemleri:

- ✓ Girdi verilerinin hem istemci hem de sunucu tarafında filtrelenmesi,
- ✓ HTML etiketlerinin özel karakterlere dönüştürülmesi (escaping),
- ✓ Content Security Policy (CSP) uygulaması ile zararlı komutların engellenmesi.

3. Kimlik Doğrulama Zayıflıkları

E-ticaret sistemlerinde kullanıcı hesapları, alışveriş deneyiminin merkezinde yer alır. Ancak bu hesaplara erişim, genellikle yalnızca kullanıcı adı ve parola kombinasyonuyla sağlanır. Zayıf parola politikaları, 2FA kullanımının olmaması ve oturum yönetimi eksiklikleri, saldırganların kimlik doğrulama mekanizmalarını kolayca aşmasına neden olur.

Sıkça görülen hatalar:

- ✓ Kullanıcıya ait varsayılan parolaların değiştirilmemesi,
- ✓ Parola sıfırlama bağlantılarının süreli olmaması,
- ✓ Oturum çerezlerinin HTTPS olmadan iletilmesi.

Önleme yöntemleri:

- ✓ Güçlü parola kuralları uygulamak (en az 12 karakter, özel karakter zorunluluğu),
- ✓ 2FA (iki faktörlü kimlik doğrulama) entegrasyonu,
- ✓ Oturum süresi sınırlandırması ve oturum sonlandırma mekanizmaları.

4. Güvenlik Yapılandırma Hataları (Security Misconfiguration)

E-ticaret sistemlerinde sıkça karşılaşılan bir diğer açık türü, güvenlik yapılandırmalarının eksik ya da hatalı yapılmasından kaynaklanmaktadır. Özellikle sunucu, veritabanı, uygulama çerçevesi ve API gibi bileşenlerin varsayılan ayarlarla bırakılması, saldırganların bu zayıf noktaları kolaylıkla istismar etmesine neden olmaktadır.

Yaygın yapılandırma hataları:

- Varsayılan yönetici kullanıcı adı ve parolalarının değiştirilmemesi,
- Hata mesajlarının sistem yapısı hakkında fazla bilgi vermesi,
- Kullanılmayan portların açık bırakılması,
- Dosya ve dizin izinlerinin hatalı tanımlanması,
- Web sunucusunda gereksiz modüllerin etkin hâlde tutulması.

E-ticaret örneği: Bir ödeme eklentisinin yapılandırma dosyasında yönetici bilgilerinin açık olarak bulunması veya yönetim panelinin .bak uzantılı yedeğinin erişime açık olması, saldırganlar için doğrudan bir fırsat yaratır.

Önleme yöntemleri:

- Tüm yapılandırmaların kurulum sonrası detaylı şekilde gözden geçirilmesi,
- Gereksiz servislerin ve modüllerin devre dışı bırakılması,

- Erişim kontrollerinin sıkılaştırılması,
- Saldırı yüzeyinin daraltılması amacıyla sadece ihtiyaç duyulan bileşenlerin etkinleştirilmesi,
- Otomatik yapılandırma denetleyicileri ve hardening script'leri ile sistemlerin periyodik olarak test edilmesi.

5. Güncellenmemiş Bileşenler ve Bağımlılıklar (Vulnerable and Outdated Components)

Birçok e-ticaret platformu; açık kaynaklı içerik yönetim sistemleri (CMS), modüller, eklentiler ve üçüncü taraf API'ler kullanmaktadır. Bu bileşenler zamanla yeni tehditler karşısında savunmasız kalabilir. Ancak çoğu işletme, bu bileşenleri düzenli olarak güncellemediği için ciddi açıklara maruz kalır.

Örnek senaryo: WordPress tabanlı bir e-ticaret sitesinde, 3 yıl önce geliştirilen ancak hiç güncellenmemiş bir kargo takip modülü kullanılmaktadır. Bu modülde bulunan XSS açığı nedeniyle tüm kullanıcı oturumları tehlikeye girer.

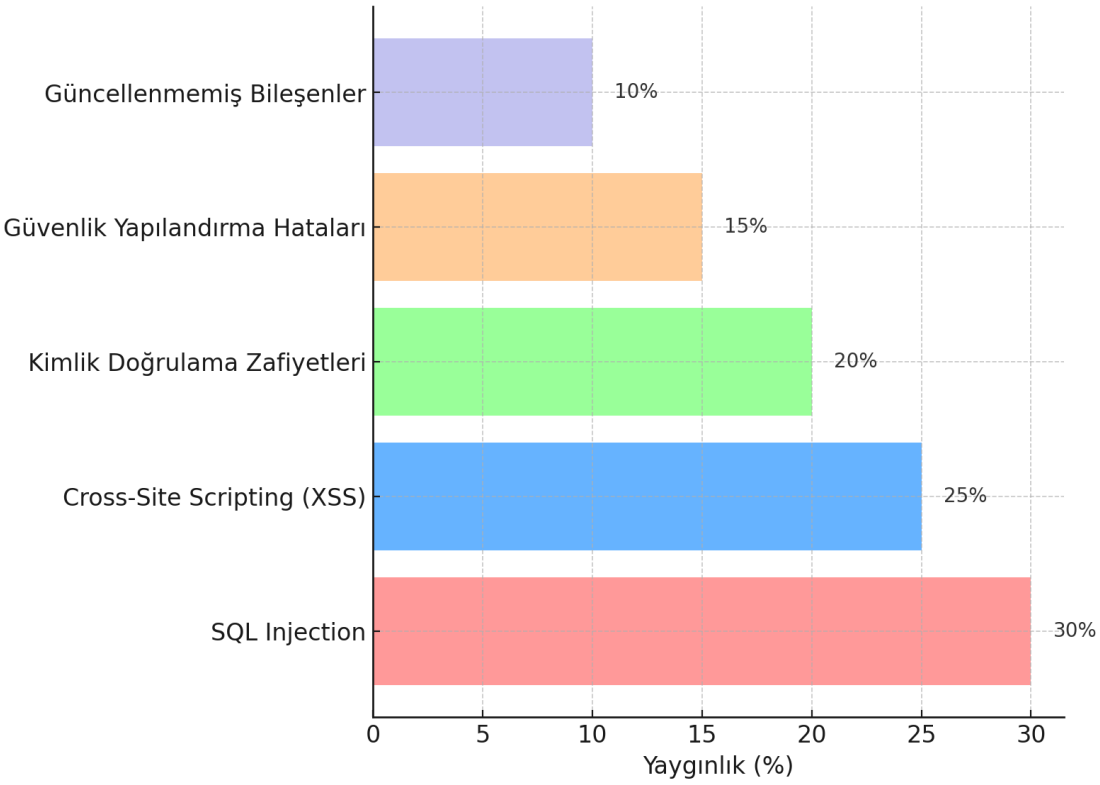
Etkileri:

- Bilinen açıklara sahip kütüphanelerin kullanılması nedeniyle sistemin hedef alınması,
- Güvenlik açıklarının internette yayımlanmış olması nedeniyle saldırganlar için kolay hedef hâline gelinmesi,
- Uyumlu olmayan güncellemelerin sistemde kararsızlığa yol açması.

Önleme yöntemleri:

- Kullanılan tüm kütüphanelerin, eklentilerin ve çerçevelerin versiyonlarının düzenli olarak takip edilmesi,
- Güvenlik yaması yayımlandığında hızlıca uygulanması,
- Sürüm takibi sağlayan araçlarla otomatik uyarı sistemlerinin kurulması (örneğin: Dependabot, Snyk),
- Gereksiz bileşenlerin sistemden tamamen kaldırılması.

Yapılan analizler, e-ticaret sistemlerinde en sık karşılaşılan güvenlik açıklarının başında SQL Injection**,** Cross-Site Scripting (XSS) **ve** kimlik doğrulama zafiyetleri**nin** geldiğini göstermektedir. Grafik 3.1'de de görüldüğü üzere, bu açıklıklar tüm vakaların %75'inden fazlasını oluşturarak sistemlerin güvenlik mimarisindeki zayıf noktaları işaret etmektedir.



Şekil 3.1: E-Ticaret Sitelerinde En Sık Görülen Güvenlik Açıkları

Güvenlik yapılandırma hataları ile güncellenmemiş bileşenlerin neden olduğu riskler ise, sıklıkla göz ardı edilmesine rağmen ciddi sistemsel tehditler barındırmaktadır. Bu bulgular, e-ticaret platformlarının yalnızca dış tehditlere karşı değil, aynı zamanda içerideki yapısal eksikliklere karşı da sistematik olarak denetlenmesi ve güçlendirilmesi gerektiğini ortaya koymaktadır.

3.3 Açıkların Tespit Edilme Yöntemleri

Güvenlik açıklarının tespit edilmesi, siber güvenlik sürecinin en kritik aşamalarından biridir. E-ticaret sistemleri gibi çok katmanlı ve yüksek trafik alan yapılar için, zafiyetlerin erken dönemde saptanması; veri kaybı, itibar zedelenmesi ve finansal zararların önüne geçilmesi açısından hayati önem taşır. Bu nedenle açıkların tespiti için sistematik analiz yöntemleri ve otomatik araçlarla desteklenmiş test süreçleri uygulanmaktadır.

E-ticaret uygulamalarında güvenlik açıklarını tespit etmenin dört temel yolu öne çıkmaktadır: **dinamik analiz**, **statik analiz**, **sızma testleri** ve **zafiyet tarama araçları**. Bu yöntemlerin her biri farklı yaklaşım, derinlik ve hedef sistem katmanlarına odaklanır. Bu bölümde her biri detaylı biçimde açıklanmıştır.

3.1.1 Dinamik analiz

Dinamik analiz, uygulamanın çalıştığı ortamda test edilmesini ve açıkların sistem davranışları gözlemlenerek tespit edilmesini amaçlayan bir yöntemdir. Bu analiz sırasında yazılımın içeriği değil, çalışma zamanı davranışı incelenir. Genellikle test sunucularında veya canlı sistem üzerinde denetimli erişimle gerçekleştirilir.

Nasıl çalışır? : Dinamik analizde uygulamaya gerçek kullanıcı davranışları taklit edilerek istekler gönderilir. Giriş formları, sorgu alanları, URL parametreleri ve oturum yönetimi gibi bileşenlere yönelik saldırı senaryoları denenir. Yanıtlar izlenir, beklenmeyen çıktılar ve hata mesajları analiz edilir. Bu süreçte otomatik test araçları ya da manuel test teknikleri kullanılabilir.

Avantajları:

- Gerçek çalışma koşullarında test yapılır,
- Kod erişimi gerekmeden uygulanabilir,
- Kritik kullanıcı senaryolarında doğrudan sonuç verir,
- Oturum bazlı açıklar (örneğin CSRF, XSS) başarıyla tespit edilebilir.

Dezavantajları:

- Kod içeriği bilinmediği için derinlemesine analiz sınırlıdır,
- Yavaş çalışabilir ve bazı hatalar gözden kaçabilir,
- Canlı sistemde yapılırsa hizmet kesintisi riski taşır.

E-ticaret bağlamında uygulama örneği**:** Bir e-ticaret sitesinde kullanıcı kayıt formuna otomatik olarak SQL Injection ve XSS payload'ları gönderilir. Yanıtta hata mesajı ya da HTML içine yerleştirilmiş kötü niyetli kod çalışıyorsa, bu bir açıklık olarak rapor edilir.

Araç önerileri:

- OWASP ZAP (dinamik analiz için ücretsiz ve güçlü bir platformdur),
- Netsparker (özellikle e-ticaret sistemlerine uygun ticari bir çözümdür),
- Acunetix, Burp Suite Professional gibi ticari analiz araçları da dinamik tarama yapabilir.

3.1.2 Statik analiz

Statik analiz, yazılımın çalıştırılmadan önce incelenmesini temel alan bir güvenlik denetim yöntemidir. Bu analiz yöntemi, genellikle kaynak kodu, ikili dosyalar veya bytecode üzerinde gerçekleştirilir. Statik analiz araçları sayesinde, yazılımın mantıksal hataları, güvenlik açıkları, kodlama standartlarına aykırılıklar ve potansiyel tehlikeler tespit edilebilir. Bu yöntem, geliştirmenin erken aşamalarında uygulandığında, daha az maliyetle daha fazla güvenlik sağlama potansiyeline sahiptir. Özellikle büyük e-ticaret sistemlerinde, uygulamaya entegre edilen açık kaynak bileşenlerin güvenliğinin doğrulanması ve kod kalitesinin sürdürülebilirliği açısından statik analiz oldukça önemlidir.

Statik Analiz Süreci

- Kod Tabanının Belirlenmesi: Analiz edilecek proje veya bileşen seçilir.
- Analiz Aracının Konfigürasyonu: Kullanılacak statik analiz aracı (örneğin SonarQube, Checkmarx, Fortify) kurulur ve proje ile entegre edilir.
- Kod Tarama: Araç, belirlenen kurallar doğrultusunda kodu tarar ve olası açıkları raporlar.

- Raporlama ve Düzeltme: Bulunan güvenlik açıkları, geliştiriciler tarafından değerlendirilir ve gerekirse düzeltilir.

Avantajları

- Erken aşamada tespit: Kod çalıştırılmadan önce hatalar bulunabilir.
- Otomatik ve sürekli denetim: CI/CD süreçlerine entegre edilebilir.
- Kaynak kod görünürlüğü: Kodun tüm detaylarına erişilerek derinlemesine analiz yapılabilir.

Dezavantajları

- Yanlış pozitif üretme riski: Gerçekte tehdit oluşturmayan ama riskli gibi görünen bulgular ortaya çıkabilir.
- Dinamik davranışları analiz edemez: Uygulama çalışırken ortaya çıkan açıklar bu yöntemle tespit edilemez.

E-Ticaret Sitelerinde Kullanımı ise; özellikle ödeme modülleri, kullanıcı doğrulama sistemleri ve API entegrasyonları gibi hassas bileşenlerin analizinde statik analiz kritik rol oynar. Örneğin, bir e-ticaret platformunun PHP ile yazılmış ödeme altyapısında SQL Injection benzeri açıkların statik analiz aracı ile erkenden tespiti, hem veri ihlallerini engeller hem de kullanıcı güvenini artırır.

3.1.3 Sızma testleri (Pentest)

Sızma testleri, bir sistemin, uygulamanın veya ağın güvenliğini değerlendirmek amacıyla yapılan kontrollü saldırı senaryolarıdır. Gerçek bir saldırganın davranışlarını taklit eden bu testler sayesinde, mevcut güvenlik açıkları önceden belirlenebilir ve giderilebilir. E-ticaret siteleri gibi kullanıcı verisi, ödeme bilgileri ve kimlik doğrulama süreçleri içeren sistemlerde sızma testleri, saldırıya açık noktaların tespitinde kritik rol oynar.



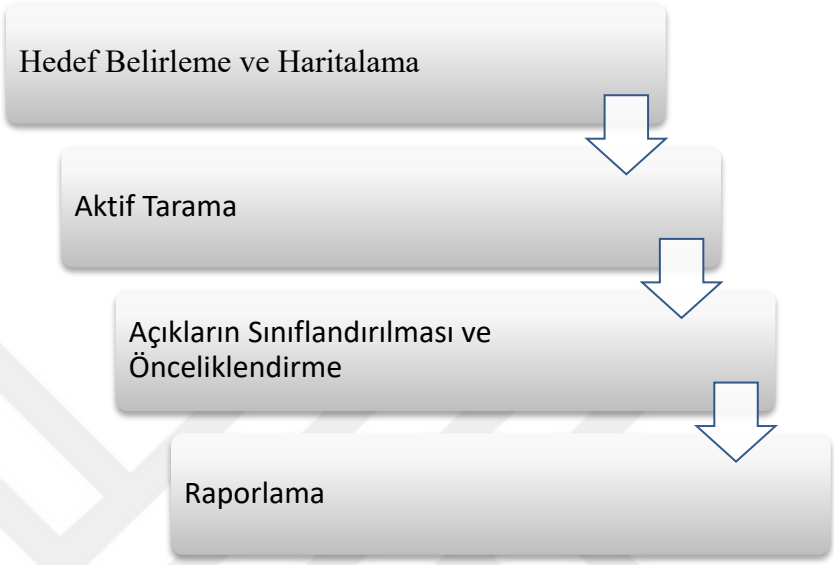
Şekil 3.2. Sızma Testi Süreci

Yukarıdaki Şekil 1, genel bir sızma testi sürecini özetlemektedir. Süreç, hedeflerin belirlenmesiyle başlar ve bilgi toplama, açıklık analizi, istismar ve raporlama adımlarıyla devam eder. Özellikle bu testler, sadece yazılımın değil, konfigürasyonların, üçüncü parti entegrasyonların ve ağ güvenliğinin de denetlenmesini sağlar.

3.1.4 Zafiyet tarayıcıları

Günümüzde e-ticaret sistemlerinin güvenliğini sağlamak, sadece kullanıcı bilgilerinin korunması açısından değil, aynı zamanda yasal sorumlulukların yerine getirilmesi ve marka itibarının sürdürülebilirliği bakımından da kritik bir gerekliliktir. Bu bağlamda, zafiyet tarayıcıları (vulnerability scanners), sistemlerdeki güvenlik açıklarını otomatik olarak tespit eden, değerlendiren ve raporlayan araçlar olarak öne çıkar. Özellikle web tabanlı e-ticaret uygulamalarında yaygın olarak kullanılan bu araçlar, hem manuel test süreçlerini hızlandırır hem de insan hatasını en aza indirir.

Zafiyet tarayıcıları, belirli algoritmalara ve güncel güvenlik veritabanlarına dayanarak sistemleri analiz eder. Bu analiz, sistemin ağ mimarisi, web servisleri, veri tabanları ve uygulama katmanları üzerinde yapılabilir. Tarayıcılar genellikle şu adımları izler:



Şekil 3.3. Zafiyet Tarayıcılarının İşlevi ve Çalışma Prensibi

- Hedef Belirleme ve Haritalama: Tarayıcı, analiz edilecek uygulama veya sistem üzerinde tarama alanlarını belirler. IP adresleri, URL yapıları ve port bilgileri bu aşamada tanımlanır.
- Aktif Tarama: Sistem veya uygulama üzerinde aktif bir analiz gerçekleştirilir. Bu taramalar sırasında açık portlar, hizmetler, sürüm bilgileri, kullanılan teknolojiler ve potansiyel riskli bileşenler ortaya çıkarılır.
- Açıkların Sınıflandırılması ve Önceliklendirme: Tespit edilen zafiyetler; düşük, orta ve yüksek risk düzeylerine göre sınıflandırılır. Bu sınıflama, sistem yöneticilerinin önceliklendirme yapmasını kolaylaştırır.
- Raporlama: Tespit edilen açıkların türü, etki düzeyi, olası istismar senaryoları ve önerilen çözüm yolları detaylı bir rapor olarak sunulur.

Popüler Zafiyet Tarayıcıları

- **OWASP ZAP (Zed Attack Proxy)**

  OWASP tarafından geliştirilen ZAP, açık kaynaklı ve özellikle web uygulamaları üzerinde etkili olan bir zafiyet tarayıcıdır. Proxy olarak çalışan bu araç, kullanıcının tarayıcı ile hedef uygulama arasındaki trafiği analiz ederek potansiyel zafiyetleri ortaya çıkarır. Enjekte edilebilir alanlar, güvenli olmayan yönlendirmeler, şifreleme zayıflıkları ve oturum yönetimi hataları gibi pek çok açı tespit edilebilir.

- **Burp Suite**

  PortSwigger tarafından geliştirilen Burp Suite, hem ücretsiz hem de profesyonel sürümleriyle oldukça geniş bir kullanıcı kitlesine hitap etmektedir. Proxy, tarayıcı entegrasyonu, intruder (yük bindirme), repeater (manüel test) ve scanner gibi modülleri sayesinde detaylı bir güvenlik değerlendirmesi sunar. Özellikle SQL injection, XSS, CSRF gibi OWASP Top 10 kategorilerinde başarılı analizler yapabilir. Ayrıca eklenti desteği sayesinde yetenekleri geliştirilebilir.

- **Nikto**

  Nikto, web sunucularının açıklarını test etmek üzere geliştirilmiş, komut satırı tabanlı bir tarayıcıdır. Apache, Nginx, IIS gibi popüler web sunucularını hedef alarak 6000'den fazla potansiyel problemi kontrol edebilir. Versiyon açıklıkları, dizin listelemeleri, varsayılan dosyalar, yanlış yapılandırmalar ve SSL güvenlik açıkları gibi unsurlar Nikto tarafından başarıyla tespit edilebilir. Hızlı ve kapsamlı taramalar yapabilmesiyle dikkat çeker.

Zafiyet Tarayıcılarının E-Ticaret Sitelerinde Kullanımı ise; **E**-ticaret siteleri, dinamik içerik üretimi, kullanıcı oturum yönetimi, ödeme sistemleri ve çok sayıda üçüncü parti servis entegrasyonu barındırır. Bu yapı, potansiyel güvenlik açıklarının sayısını ve çeşitliliğini artırmaktadır.

Zafiyet tarayıcılarının bu tür yapılarda sağladığı başlıca avantajlar şunlardır:

- ✓ Hızlı Güvenlik Değerlendirmesi: Geniş sistemlerin manuel olarak test edilmesi zaman alırken, tarayıcılar hızlı sonuçlar sunar.
- ✓ Düzenli Güvenlik Taraması: E-ticaret altyapısında sürekli güncellenen yazılım bileşenleri nedeniyle periyodik taramalar yapılabilir.
- ✓ Raporlarla Uyum Süreçlerini Destekleme: GDPR, KVKK gibi veri koruma düzenlemelerine uyum açısından raporlar kayıt altına alınabilir.
- ✓ CI/CD Entegrasyonu: Modern zafiyet tarayıcıları, yazılım geliştirme süreçlerine entegre edilerek her sürümde güvenlik kontrolü sağlar.

Her ne kadar zafiyet tarayıcıları güçlü araçlar olsa da, bazı sınırlamaları da vardır: Yanlış Pozitif ve Negatifler: Tarayıcılar zaman zaman yanlış pozitif (gerçekte olmayan ama var gibi gösterilen) ya da yanlış negatif (var olan ama tespit edilemeyen) sonuçlar üretebilir. Manuel Doğrulama Gerekliliği: Kritik açıklar çoğu zaman manuel testlerle doğrulanmalıdır. Karmaşık Yapılarla Uyum: Dinamik içerikler, çok faktörlü oturumlar ve JavaScript yoğun sistemlerde bazı araçlar eksik taramalar yapabilir.

4. GÜVENLİK AÇIĞI TESPİTİ

E-ticaret sistemlerinin karmaşık ve dinamik yapısı, siber güvenlik tehditlerine karşı çok katmanlı bir savunma yaklaşımını zorunlu kılmaktadır. Bu doğrultuda, sistematik bir güvenlik açığı tespit modeli geliştirmek, açıklara karşı erken müdahale imkânı sunmakta ve saldırı riskini önemli ölçüde azaltmaktadır. Bu bölümde, e-ticaret altyapılarında kullanılmak üzere geliştirilen bir güvenlik açığı tespit modeli detaylı biçimde sunulmakta; modelin bileşenleri, işleyiş yapısı ve uygulanabilirlik kriterleri açıklanmaktadır.

Sİstem, iki ana tespit yaklaşımını bir arada kullanacak şekilde tasarlanmıştır: manuel analiz ve otomatik analiz.

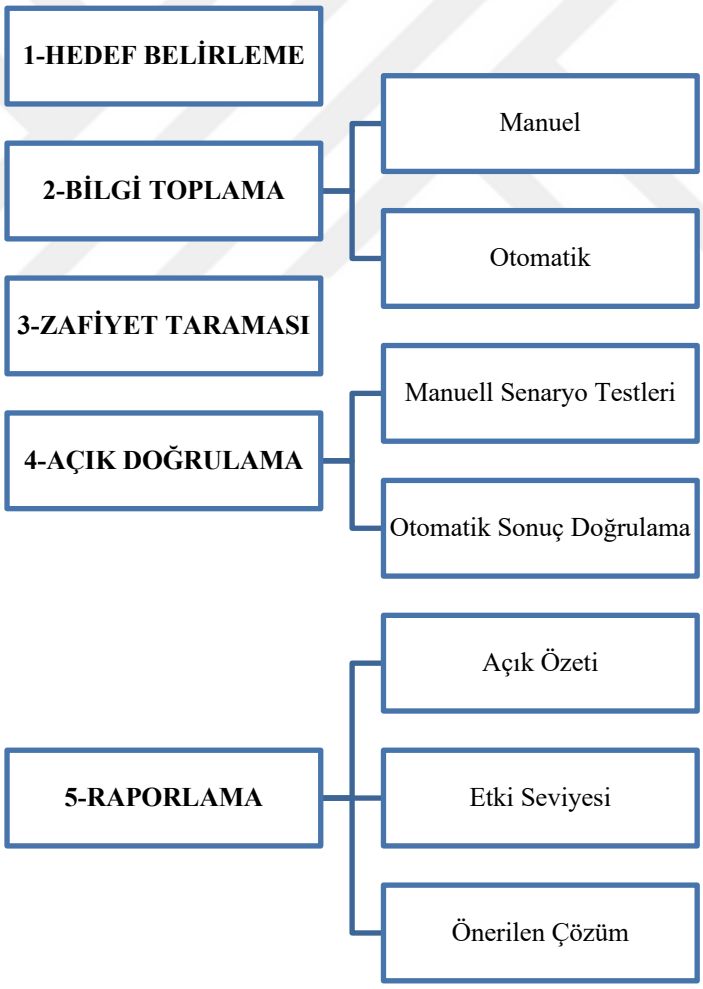
- Manuel analiz, uzman bir güvenlik testçisinin bilgi ve deneyimine dayanarak, sistem üzerinde kontrollü ve senaryoya dayalı testler gerçekleştirmesini ifade eder. Bu tür analizlerde genellikle kullanıcı girdileri, oturum yönetimi veya erişim kontrol mekanizmaları doğrudan elle test edilir. Örneğin, bilinçli olarak sistemde SQL Injection denemesi yapmak ve çıkan tepkilere göre sistemin tehlike seviyesini yorumlamak gibi.
- Otomatik analiz ise, belirli algoritmalara sahip yazılımlar aracılığıyla sistemin geniş çaplı olarak taranmasını sağlar. Bu taramalar sayesinde, büyük ölçekte açıklıkların sistematik olarak taranması ve ön değerlendirme yapılması mümkün olur. Araçlar; belirli açıklara karşı imza tabanlı veya davranışsal kontroller yaparak, sonuçları raporlar. OWASP ZAP, Nikto, Burp Suite gibi araçlar bu kapsamda yaygın olarak kullanılır.

Geliştirilen tasarımda bu iki yöntem birbirini tamamlayıcı şekilde kullanılmıştır: Otomatik araçlarla yapılan ilk taramanın ardından, manuel doğrulamalarla gerçek tehditler ayrıştırılmış ve yanıltıcı sonuçlar elenmiştir. Bu sayede hem geniş kapsamlı tarama hem de derinlemesine doğrulama mümkün kılınarak, tespitlerin doğruluk oranı artırılmıştır.

Tasarımın mimari yapısı, süreç adımları, otomasyon desteği ve yasal/etik uygunluk ilkeleri sırasıyla aşağıdaki alt bölümlerde detaylandırılacaktır.

4.1 Sistem Tasarımının Mimari Yapısı

Geliştirilen güvenlik açığı tespit modeli, modüler ve katmanlı bir yapı üzerinde kurgulanmıştır. Bu yapı, sistematik analiz süreçlerinin hem manuel hem de otomatik yöntemlerle bütünleştirilmesini sağlayarak geniş kapsamlı, ölçeklenebilir ve uyarlanabilir bir çözüm sunmayı hedeflemektedir. Modelin temel bileşenleri; hedef tanımlama**,** bilgi toplama**,** zafiyet tarama**,** açık doğrulama **ve** raporlama olmak üzere beş temel modülden oluşmaktadır. Her bir modül, sürecin ayrı bir aşamasını temsil etmekte ve genel işleyişin tutarlılığı açısından birbirine veri aktaran şekilde yapılandırılmıştır. Şekil 4.1'de modelin mimari yapısı grafiksel olarak sunulmuş ve modüller arası veri akışı görsel biçimde temsil edilmiştir. Bu yapı sayesinde tespit modeli hem teorik olarak anlaşılır hem de pratikte uygulanabilir olmuştur. Aşağıda, bu modüllerin işlevsel yapısı ve aralarındaki veri akışı detaylı biçimde açıklanmıştır:



Şekil 4.1. E-ticaret sitelerinde güvenlik açığı tespitine ilişkin sistem tasarımının akış şeması. Süreç, hedef belirlemeden raporlamaya uzanan beş adımdan oluşur.

- **Hedef Belirleme - Web Arayüzü**

Bu katman, güvenlik testlerini başlatacak olan kullanıcıya ait giriş noktasıdır. Test parametreleri, hedef bilgileri ve kullanılacak analiz yöntemleri bu panel üzerinden seçilmektedir. Kullanıcı dostu bir arayüz ile test süreci yönlendirilebilir, test sonuçları da buradan izlenebilir.

- **Hedef Belirleme Modülü**

Bu modül, analiz yapılacak sistemin teknik parametrelerini tanımlamakla sorumludur. IP adresi, alan adı (domain), port bilgileri ve sistem altyapısı gibi bilgiler bu aşamada alınır. Modül ayrıca hedef sistemin teknolojik özelliklerini sınıflandırarak sonraki aşamalarda kullanılacak analiz senaryolarını belirler.

- **Bilgi Toplama Modülü**

Bu aşamada, hedef sistem hakkında olabildiğince geniş kapsamlı bilgi toplanır. Nmap, Whois, BuiltWith, Shodan gibi araçlar kullanılarak, sistemin altyapısı, çalışan servisleri, açık portları ve kullanılan teknolojiler tespit edilir. Bu bilgiler, sonraki aşamalardaki zafiyet taramalarının kapsamını belirlemede kritik rol oynar.

- **Zafiyet Tarayıcı Katmanı**

Modelin bu bileşeni, tespit sürecinde otomatik analiz araçlarının devreye alındığı katmandır. OWASP ZAP, Nikto ve Burp Suite gibi tarayıcıların yanı sıra Nikto aracı da .NET ortamında çalışabilecek sistemler üzerinde bilinen güvenlik açıkları taranır. Bu taramalar hem hızlı ön değerlendirme sağlar hem de insan hatasını azaltır. Tarayıcı çıktıları, açıklıkların seviyesine ve türüne göre sınıflandırılarak bir sonraki adıma iletilir.

- **Açık Doğrulama Motoru**

Otomatik tarama sonuçlarının doğrulanması amacıyla devreye giren bu modül, potansiyel zafiyetlerin gerçekten sistemde sömürülebilir olup olmadığını kontrol eder. Bazı açıklıklar yalnızca belirli koşullarda aktif hâle geldiğinden, manuel testler bu süreçte tamamlayıcı rol oynar. Aynı zamanda bu modül, yanlış pozitif sonuçları eleyerek analizlerin doğruluğunu artırır.

- **Raporlama Sistemi**

Tüm analiz sürecinin sonunda, tespit edilen açıklara dair teknik ve yönetsel içerikli raporlar üretilir. Raporlar PDF, HTML veya JSON formatında hazırlanabilir. Açıkların türü, etki düzeyi, istismar senaryosu ve çözüm önerileri bu raporlarda detaylandırılır. Ayrıca bu raporlar, ilgili yöneticilere iletilerek sistem üzerinde düzeltme adımlarının planlanmasına yardımcı olur. Raporlama modelini anlatacağımız 5. bölümde bu konu detaylıca açıklanacaktır.

4.2 Açık Tespit Süreci

Güvenlik açığı tespit süreci, bir e-ticaret sisteminde potansiyel zafiyetlerin sistematik olarak saptanmasını ve bunların doğruluğunun kontrol edilmesini kapsar. Etkin bir tespit süreci hem manuel hem de otomatik analiz tekniklerinin birlikte uygulanmasını gerektirir (Anderson, 2020). Bu yaklaşım sayesinde hem geniş çaplı bir tarama gerçekleştirilir hem de elde edilen sonuçların geçerliliği manuel doğrulamalarla güçlendirilir. Süreç dört temel aşamada ilerler:

- **Hedef Belirleme:** Tespit süreci, test edilecek sistemin kapsamının net olarak belirlenmesiyle başlar. Bu kapsamda alan adları, IP blokları, port aralıkları, kullanılacak tarayıcı türleri ve dış servis entegrasyonları analiz edilir. Özellikle kapsam dışı tutulması gereken alanların belirlenmesi, etik ve hukuki sınırların çizilmesi açısından kritik öneme sahiptir (Solove, 2007). Ayrıca, bu aşama tarama araçlarının yapılandırılması ve saldırı yüzeyinin belirlenmesi için de temel teşkil eder.

- **Bilgi Toplama:** Bu aşama, sistem hakkında pasif ve aktif yöntemlerle bilgi edinmeyi içerir. DNS kayıtları, açık portlar, çalışan servisler, web uygulaması yapısı ve kullanılan teknolojiler gibi veriler toplanır. Nmap, Shodan, BuiltWith gibi araçlar aracılığıyla toplanan bu bilgiler, sonraki adımlarda uygulanacak testlerin senaryosunu şekillendirir (OWASP, 2023). Etkili bilgi toplama, zafiyet taramasının başarısını doğrudan etkileyen bir faktördür.
- **Zafiyet Taraması:** Toplanan bilgilere dayanarak sistemde bilinen güvenlik açıklarının olup olmadığı çeşitli tarama araçlarıyla araştırılır. OWASP ZAP, Nikto, Burp Suite gibi araçlar bu aşamada yaygın biçimde kullanılır

(Symantec, 2021). Tarama işlemleri sırasında SQL Injection, Cross-Site Scripting (XSS), açık portlar veya yapılandırma hataları gibi OWASP Top 10 listesinde yer alan tehditlere karşı kontroller yapılır. Bu taramalar geniş çaplı, hızlı ve sistematik bir ön değerlendirme sağlar.

- **Açık Doğrulama:** Tarama araçlarının sunduğu sonuçlar her zaman kesin değildir. Bu nedenle olası açıklıkların manuel yöntemlerle test edilmesi ve istismar edilip edilemeyeceklerinin belirlenmesi gerekir. Açık doğrulama süreci; sahte pozitiflerin (false positive) elenmesini sağlar ve yalnızca gerçek, istismar edilebilir açıklıkların raporlanmasına imkân tanır. Bu yaklaşım, yalnızca teknik doğruluk değil, aynı zamanda sistem yöneticilerine sunulan raporların güvenilirliği açısından da büyük önem taşır (ENISA, 2023).

4.2.1 Araç Seçim Kriterleri ve Gerekçeler

Bu çalışmada güvenlik açığı tespit sürecinde kullanılan araçların seçiminde aşağıdaki teknik ve akademik kriterler dikkate alınmıştır:

Kapsam ve Tespit Edilen Açık Türleri

- OWASP ZAP, hem pasif hem aktif tarama yeteneklerine sahip olup OWASP Top 10 kapsamında tanımlanan tüm kritik açık türlerini tespit edebilme kapasitesi nedeniyle tercih edilmiştir.
- Nikto, HTTP güvenlik başlıkları ve web sunucusu konfigürasyon hataları gibi alanlarda derinlemesine analiz yapabilmesi nedeniyle eklenmiştir.
- Nmap, ağ seviyesi port ve servis keşfi ile birlikte SSL/TLS yapılandırma analizlerinde güvenilir bir standart araç olduğu için seçilmiştir.
- Burp Suite, özellikle manuel doğrulama ve özel saldırı senaryoları (ör. SQL Injection, XSS) testlerinde sunduğu gelişmiş modüller nedeniyle kullanılmıştır.

Güncellik ve Topluluk Desteği

- Tüm araçlar aktif olarak geliştirilen ve güvenlik topluluğunda yaygın kabul görmüş projelerdir.

- OWASP ZAP ve Nmap açık kaynak, Nikto açık kaynak, Burp Suite ise akademik lisans desteği ile erişilebilir durumdadır.

Literatürde Kullanım Sıklığı

- Son beş yılda yapılan benzer çalışmalarda bu araçların yaygın şekilde kullanıldığı tespit edilmiştir (örn. [OWASP, 2023], [Zhou ve ark., 2022], [PCI DSS, 2023]).
- Böylece sonuçların literatürle karşılaştırılabilir olması sağlanmıştır.

Entegrasyon Kolaylığı

- Araçların API desteği (OWASP ZAP, Nmap) ve script ile otomasyon imkânları (Python, Bash) test ortamına hızlı entegrasyon sağlamıştır.

Bu kriterler doğrultusunda seçilen araçlar, hem kapsamlı zafiyet tespiti hem de raporlama sürecinde birbirini tamamlayan nitelikte olduğundan modelin bütünsel güvenlik değerlendirmesi yapabilmesi mümkün olmuştur.

4.3 Otomatikleştirme Yaklaşımları

Güvenlik açıklarının etkin şekilde tespit edilmesi için yalnızca manuel yöntemlere güvenmek yeterli değildir. Özellikle büyük ölçekli e-ticaret platformlarında, test süreçlerinin hızlandırılması ve insan hatalarının azaltılması amacıyla otomatikleştirme yaklaşımlarına ihtiyaç duyulmaktadır. Bu kapsamda Python script'leri, Application Programming Interface (API) entegrasyonları ve otomatik raporlama sistemleri gibi araçlar yaygın biçimde kullanılmaktadır (Anderson, 2020).

4.3.1 Python Script'leri ile Otomasyon

Python, esnek sözdizimi ve zengin kütüphane desteği sayesinde güvenlik testlerinde en çok tercih edilen programlama dillerinden biridir. Özellikle requests, socket, scapy ve selenium gibi modüller sayesinde özel test senaryoları geliştirilebilir. Örneğin, e-ticaret sitesindeki giriş formlarına sistematik SQL Injection payload'ları gönderen bir Python script'i, potansiyel açıklıkların hızlıca test edilmesini sağlar. Ayrıca Selenium destekli otomasyonlar ile kullanıcı davranışı simüle edilerek XSS gibi açıklıklar tespit edilebilir (OWASP, 2023).

4.3.2 API Tabanlı Entegrasyonlar

Modern güvenlik tarayıcıları (örneğin Burp Suite Enterprise, Nessus, Rapid7 Nexpose) API desteği sunarak, bu araçların dış sistemlerle entegre şekilde çalışmasını mümkün kılar. API entegrasyonları sayesinde, belirli periyotlarla otomatik tarama başlatılabilir, bulgular alınabilir ve doğrudan bir güvenlik yönetim sistemine (örneğin SIEM) aktarılabilir (ENISA, 2023). Bu yapı aynı zamanda CI/CD (Continuous Integration/Continuous Deployment) süreçlerine entegre edilerek, her kod güncellemesi sonrası otomatik güvenlik kontrolü yapılmasına olanak tanır (NIST, 2022).

4.3.3 Otomatik Raporlama Sistemleri

Tespit edilen güvenlik açıklıklarının anlaşılır biçimde raporlanması, teknik ekiplerin hızlı müdahalesi için büyük önem taşır. Python veya JavaScript tabanlı araçlar ile elde edilen bulgular, PDF veya HTML formatında düzenli olarak üretilebilir. Bazı gelişmiş sistemlerde, açıklıkların CVSS (Common Vulnerability Scoring System) puanları da otomatik olarak hesaplanır ve rapor içerisine dahil edilir. Raporlama süreci aynı zamanda sistem yöneticilerine e-posta bildirimleri göndererek aksiyon süresini kısaltabilir (Symantec, 2021).

4.3.4 Yararları ve Sınırlılıkları

Otomatikleştirme, zaman tasarrufu, tekrarlanabilirlik ve kapsamlı analiz gibi önemli avantajlar sunar. Ancak, her otomasyonun manuel kontrol ile desteklenmesi gerekir. Çünkü bazı güvenlik açıklıkları yalnızca kullanıcı davranışı bağlamında ortaya çıkabilir ve dinamik içeriklerle çalışan sistemlerde otomatik araçlar sınırlı kalabilir (Chaffey & Ellis-Chadwick, 2019).

Otomatikleştirilmiş güvenlik açıkları tespit sistemleri, özellikle büyük ve karmaşık yapıya sahip e-ticaret platformlarında süreç verimliliğini artırma, zaman tasarrufu sağlama ve insan hatasını azaltma yönüyle önemli avantajlar sunar. Ancak bu sistemlerin bazı yapısal sınırlamaları da bulunmaktadır. Bu nedenle, otomasyon süreçlerinin her zaman manuel analizle desteklenmesi gerektiği unutulmamalıdır (NIST, 2022).

Yararları

- Hız ve Verimlilik: Otomasyon sistemleri binlerce satırlık kodu veya yüzlerce endpoint'i insan müdahalesi olmadan kısa sürede tarayabilir. Bu, özellikle periyodik test gerektiren sistemler için büyük zaman tasarrufu sağlar (ENISA, 2023).
- Tekrarlanabilirlik: Aynı güvenlik testi senaryosu birçok kez aynı doğrulukla çalıştırılabilir. Bu, test sonuçlarının karşılaştırmalı olarak analiz edilmesini kolaylaştırır.
- Geniş Kapsamlılık: Statik analiz, dinamik analiz ve zafiyet tarayıcıları gibi araçlar, sistemin birçok bileşenini aynı anda değerlendirme imkânı tanır (OWASP, 2023).
- Raporlama Entegrasyonu: CVSS puanları ile desteklenmiş teknik raporlar, otomatik olarak yönetici panellerine entegre edilebilir. Böylece müdahale süreleri kısaltılır (Symantec, 2021).

Sınırlılıkları

- Yanlış Pozitif ve Negatifler: Bazı otomatik tarama araçları, sistemde var olmayan açıklıkları "varmış" gibi (false positive) ya da mevcut açıklıkları "yokmuş" gibi (false negative) gösterebilir. Bu durum manuel doğrulama ihtiyacını artırır (Chaffey & Ellis-Chadwick, 2019).
- Karmaşık Girdilerle Zorluk: CAPTCHA, çok faktörlü kimlik doğrulama (2FA), oturum-tabanlı erişim gibi etkileşimli güvenlik mekanizmaları otomatik araçların tespit yeteneğini sınırlar.
- İnsan Yorumuna Olan İhtiyaç: Özellikle davranışsal zafiyetler, iş mantığı hataları veya kullanıcıya özel senaryolar gibi açıklar, yalnızca insan gözlemiyle tespit edilebilir (Anderson, 2020).
- Saldırıya Açık Otomasyon Sistemleri: Özellikle internete açık bırakılmış API tabanlı güvenlik tarayıcılarının kendileri de hedef hâline gelebilir. Bu nedenle yapılandırma dikkatli yapılmalıdır.

Çizelge 4.1. Otomatik Açık Tespit Yöntemlerinin Yararları ve Sınırlılıkları

Kriter	Yararları	Sınırlılıkları
Zaman Verimliliği	Büyük sistemlerde hızlı sonuç üretir	Bazı karmaşık durumları atlayabilir
Doğruluk ve Tekrar Üretilebilirlik	Aynı testler defalarca güvenilir şekilde uygulanabilir	False positive/negative oranı yüksektir
Geniş Kapsamlılık	Birden fazla sistem katmanını aynı anda tarayabilir	Oturum-tabanlı ya da CAPTCHA korumalı sistemlerde sınırlı kalır
Raporlama ve Entegrasyon	CVSS destekli otomatik raporlar oluşturabilir	Raporlar yorumsuz ve bağlamdan kopuk olabilir
Maliyet Avantajı	Uzun vadede test maliyetini düşürür	Başlangıçta kurulum ve entegrasyon için uzmanlık gerekebilir
Etkinlik	Kritik zafiyetleri hızlıca görünür kılar	Karmaşık saldırı senaryolarını analiz edemez

4.4 Yasal ve Etik Boyutlar

Güvenlik açıklarını tespit etmek amacıyla yapılan her türlü tarama ve analiz işlemi, yalnızca teknik yeterlilikle değil, aynı zamanda yasal ve etik ilkelerle de uyumlu olmalıdır. Özellikle e-ticaret sistemleri gibi kişisel veri işleyen platformlarda gerçekleştirilen sızma testleri ve otomatik taramalar hem yerel hem de uluslararası düzenlemeler çerçevesinde değerlendirilmelidir. Aksi hâlde, yapılan işlemler kötü niyetli saldırılarla benzer sonuçlara yol açabilir ve ciddi hukuki yaptırımlar doğurabilir (Solove, 2007; NIST, 2022).

İzinsiz Tarama ve Sızma Riskleri

Bir sistem üzerinde izinsiz olarak güvenlik açığı taraması yapmak, Türkiye Cumhuriyeti Türk Ceza Kanunu (TCK) 243. ve 244. maddelerine göre "bilişim sistemine yetkisiz erişim" ve "veri bozma" suçları kapsamına girebilir. Bu bağlamda, bir e-ticaret platformu üzerinde herhangi bir güvenlik testinin yapılabilmesi için mutlaka ilgili kurumun **yazılı izni** alınmalıdır. Bu izin, etik sızma testi protokollerine uygun olarak yapılandırılmalı; testin kapsamı, süresi ve raporlama biçimi önceden belirlenmelidir (ENISA, 2023).

Uluslararası düzeyde ise benzer durumlar *Computer Fraud and Abuse Act (CFAA)* gibi yasalarla düzenlenmiştir. Bu nedenle testlerin "white box" veya "gray box" yöntemleri ile, kurum bilgisi dâhilinde gerçekleştirilmesi esastır.

Kişisel Verilerin Korunması (KVKK ve GDPR)

E-ticaret sistemlerinde yapılan güvenlik analizlerinde elde edilen veriler, doğrudan veya dolaylı olarak kişisel veri niteliği taşıyabilir. Türkiye'de yürürlükte olan **6698 sayılı Kişisel Verilerin Korunması Kanunu (KVKK)**, bu tür verilerin yalnızca açık rıza veya yasal istisna kapsamında işlenebileceğini düzenlemektedir. Test ortamlarında elde edilen IP adresleri, kullanıcı oturum bilgileri veya form girdileri gibi veriler de bu kapsama girmektedir (KVKK, 2016).

Benzer şekilde Avrupa Birliği'nin **Genel Veri Koruma Tüzüğü (GDPR)** de kişisel verilerin izinsiz toplanması, saklanması ve analiz edilmesine karşı ağır yaptırımlar öngörmektedir. Bu bağlamda yapılacak her türlü test, gerçek kullanıcı verilerinden arındırılmış simülasyon ortamlarında gerçekleştirilmelidir. Aksi takdirde hem etik dışı bir yaklaşım sergilenmiş olur hem de yasal sorumluluk doğar (Chaffey & Ellis-Chadwick, 2019).

Etik İlkeler ve Profesyonel Sorumluluk

Güvenlik testi gerçekleştiren kişi veya ekipler, yalnızca teknik yeterlilikle değil; **mesleki etik** ilkelerle de donanmış olmalıdır. Bu çerçevede aşağıdaki kurallar esas alınmalıdır:

- Sisteme zarar vermeyecek şekilde kontrollü test yapılması,
- Kullanıcı deneyimini bozacak manipülasyonlardan kaçınılması,
- Gizli bilgilerin asla üçüncü taraflarla paylaşılmaması,
- Test sonrası tüm erişimlerin sistemden silinmesi,
- Tespit edilen açıklıkların yalnızca yetkili birimlerle paylaşılması (responsible disclosure).

Bu ilkeler, etik hackerlık (ethical hacking) prensipleri çerçevesinde, "önce zarar verme" (*primum non nocere*) anlayışıyla uygulanmalıdır (Anderson, 2020).

5. RAPORLAMA SİSTEMİ

5.1 Raporlama Formatı ve İçeriği

E-ticaret sistemlerinde güvenlik açıklarının yalnızca tespit edilmesi değil, aynı zamanda etkili bir biçimde raporlanması da büyük önem taşır. Raporlama süreci hem teknik ekiplerin müdahalesini yönlendirir hem de üst yönetimin farkındalığını artırır. Geliştirilen modelde raporlama; açıklığın türü, etki derecesi, istismar senaryoları ve çözüm önerileri gibi başlıkları içerecek biçimde yapılandırılmıştır.

Güvenlik açığı raporları hem teknik hem de yönetim düzeyine hitap edecek şekilde iki katmanlı yapıdadır. Teknik kısım, güvenlik uzmanları ve geliştiriciler için detaylı açıklama ve kod örnekleri içerirken; yönetici özeti, etki seviyesi, çözüm maliyeti ve iş sürekliliği üzerindeki etkileri özetler.

Raporlar genellikle şu başlıklardan oluşur:

- Açığın kısa tanımı
- Tespit edildiği sistem bileşeni
- Etki alanı (yerel, sistemsel, tüm kullanıcılar vb.)
- Risk seviyesi (CVSS skoru ya da düşük/orta/yüksek)
- İstismar senaryosu (PoC – Proof of Concept)
- Alınması gereken önlemler (best practices ile destekli)
- Giderme süresi tahmini ve sorumlu ekip

Bu yapı sayesinde açıklıklar sadece tespit edilmekle kalmaz, aynı zamanda sorumlu kişilere yönlendirilerek zamanında müdahale sağlanır.

5.1.1 Teknik Raporlama Bileşenleri

Teknik düzeyde hazırlanan raporlar, genellikle bilgi güvenliği uzmanları, yazılım geliştiricileri veya sistem yöneticileri tarafından kullanılır. Bu raporlar aşağıdaki bileşenlerden oluşur:

- **Açığın Başlığı:** Açık türü ve etkilenen bileşen özetlenir.
- **Açığın Detaylı Tanımı:** Açığın ne olduğu, hangi kod veya sistem bileşeninde bulunduğu açıklanır.
- **PoC (Proof of Concept):** Açığın nasıl istismar edilebileceğine dair senaryo veya komut.
- **Zafiyetin Teknik Sebebi:** Girdi doğrulama eksikliği, kimlik denetimsiz API, vb.
- **Etki Analizi:** Açığın yaratabileceği risk (veri sızıntısı, yetkisiz erişim, ödeme manipülasyonu)
- **Çözüm Önerisi:** OWASP, NIST, ISO/IEC rehberlerine göre alınması gereken önlem
- **CVSS Skoru ve Açıklaması:** Açığın ciddiyet derecesi sayısal olarak tanımlanır.

Yönetici raporu ise teknik detaya inmeksizin açığın iş süreçlerine etkisini ve olası zararları özetler.

Çizelge 5.1. Örnek Yönetici Raporu

Alan	**Açıklama**
Açık Başlığı	Yönetim Paneli Girişinde XSS Açığı
Risk Seviyesi	Yüksek (CVSS 8.2)
Olası Etkisi	Kullanıcı çerezlerinin çalınması, hesap ele geçirme
Tespit Tarihi	15 Nisan 2025
Müdahale Gerekliliği	48 saat içinde çözülmesi önerilir
Sorumlu Ekip	Web Uygulama Geliştirme Ekibi
Giderme Yöntemi	Girdi temizliği, Content Security Policy (CSP) uygulanması

5.1.2 Açıklık Türüne Göre Rapor Şablonları

Kapsam: Bu bölümdeki URL, dosya adları ve tarihler **örnek amaçlıdır**; gerçek uygulama ve bulgular **Bölüm 6**'da sunulmuştur.

Not (Şablon Kullanımı): Aşağıdaki Örnek 1–3, farklı teknoloji yığınları için genel rapor şablonlarıdır. 6. bölümdeki nopCommerce/ASP.NET uygulamasıyla birebir örtüşmesi amaçlanmamıştır; 6. bölümde uyarlanmış örnekler ayrıca verilmiştir.

E-ticaret sistemlerinde karşılaşılan güvenlik açıkları, sistemin farklı katmanlarında ortaya çıkabilir ve her bir açık türü, kendine özgü analiz ve raporlama yaklaşımı gerektirir. Bu nedenle raporlamada **tek tip şablon** kullanımı yerine, açıklığın niteliğine göre özelleştirilmiş şablonlar oluşturulması hem teknik çözüm sürecini hızlandırmakta hem de yöneticilerin risk değerlendirmesini kolaylaştırmaktadır.

Örneğin, bir kimlik doğrulama açığı ile bir Cross-Site Scripting (XSS) zafiyetinin etkileri, istismar yöntemleri ve çözüm yolları birbirinden oldukça farklıdır. Bu nedenle açıklık türüne göre hazırlanmış örnek rapor şablonları, hem teknik ekiplerin müdahale süresini azaltmakta hem de kurumsal güvenlik dokümantasyonunun daha okunabilir olmasını sağlamaktadır. Aşağıda, en sık rastlanan zafiyet türlerine göre örnek raporlama formlarına yer verilecektir.

Şablon Alanları: Açığın Türü, Etki Alanı, Tespit Tarihi, Önkoşullar, İstismar Senaryosu (PoC), Kanıt/Ekran Görüntüsü, Etki, CVSS, Doğrulama Durumu (Doğrulandı/False Positive), Önerilen Çözüm, Sorumlu Ekip, SLA (Hedef Tamamlama Süresi).

Örnek 1 (Şablon): SQL Injection Açığı – Ürün Arama Modülü

- **Açığın Türü:** SQL Injection
- **Etki Alanı:** product_search.php bileşeni
- **Tespit Tarihi:** 18 Mart 2025
- **Açıklama:** Kullanıcı tarafından arama kutusuna girilen veriler doğrudan SQL sorgusuna gömülmektedir. Bu durum, parametrik olmayan sorgu yapısı nedeniyle veri tabanının istismar edilmesine yol açmaktadır.
- **İstismar Senaryosu (PoC):**

URL: https://www.example.com/search?query=' OR '1'='1

Bu sorgu sonucunda sistem tüm ürünleri listelemekte ve filtreleme işlevi tamamen devre dışı kalmaktadır.

- **Etki:** Yetkisiz veri erişimi, potansiyel veri sızıntısı, sorgu mantığının tamamen bozulması
- **CVSS Skoru:** 8.6 (High)
- **Çözüm Önerisi:**
    - Veritabanı işlemlerinde prepared statements veya ORM sistemleri kullanılmalıdır.
    - Kullanıcı girdileri sunucu tarafında doğrulanmalı ve özel karakterler filtrelenmelidir.
    - Loglama ve anormal sorgu takibi için güvenlik duvarı (WAF) desteklenmelidir.

Örnek 2 (Şablon): Cross-Site Scripting (XSS) Açığı – Yorum Paneli

- **Açığın Türü:** Reflected XSS
- **Etki Alanı:** Ürün detay sayfası altındaki yorum bölümü (comments.js)
- **Tespit Tarihi:** 5 Nisan 2025
- **Açıklama:** Kullanıcının ürün hakkında yaptığı yorum doğrudan sayfa içerisinde işlenmekte ve tarayıcıda çalıştırılmaktadır. Girdi filtresi veya escape işlemi yapılmadığından zararlı script'ler kolayca yürütülebilmektedir.
- **İstismar Senaryosu (PoC):**

  <script>fetch('http://attacker.com/cookie=' + document.cookie)</script>

  Bu payload yorum kısmına eklendiğinde diğer kullanıcıların oturum bilgileri dışarıya sızdırılabilir.

- **Etki:** Kimlik hırsızlığı, oturum çalma, yönlendirme
- **CVSS Skoru:** 7.4 (High)
- **Çözüm Önerisi:**
    - Girdi verileri HTML encode edilerek özel karakterlerin etkisizleştirilmesi sağlanmalıdır.

- Content Security Policy (CSP) başlığı tanımlanarak sadece izin verilen kaynaklardan kod çalışmasına izin verilmelidir.
- Kullanıcı yorumları, sunucu tarafında geçerli formatlara indirgenmelidir.

Örnek 3 (Şablon): Kimlik Doğrulama Zafiyeti – Oturum Güvenliği

- **Açığın Türü:** Weak Session Management
- **Etki Alanı:** auth.php, session_token.js
- **Tespit Tarihi:** 22 Nisan 2025
- **Açıklama:** Oturum çerezleri Secure ve HttpOnly bayrakları olmadan iletilmekte; ayrıca oturum süresi kullanıcı etkinliğiyle sınırsız olarak uzatılmaktadır. Bu durum özellikle ortak ağlarda (ör. kafeler, mobil erişim) oturum çalma saldırılarına zemin hazırlamaktadır.
- **Etki:** Yetkisiz oturum ele geçirme, kullanıcı taklidi
- **CVSS Skoru:** 6.8 (Medium)
- **Çözüm Önerisi:**
  - Tüm oturum çerezleri Secure ve HttpOnly olarak işaretlenmelidir.
  - Oturum zaman aşımı 15 dakika inaktiviteye göre yapılandırılmalı ve kullanıcı girişinde cihaz/mekân doğrulaması yapılmalıdır.
  - 2FA entegrasyonu zorunlu hâle getirilmelidir (mobil bildirim/SMS doğrulama).

5.2 Önceliklendirme ve Risk Skorlaması

Güvenlik açıklarının sayıca fazla olduğu sistemlerde hangi açıklığın daha önce düzeltilmesi gerektiğini belirlemek, sadece teknik değil, yönetsel açıdan da stratejik bir gerekliliktir. Özellikle yüksek trafiğe sahip e-ticaret platformlarında, her açığa aynı önemi vermek mümkün olmadığından, açıklıkların risk düzeyine göre **önceliklendirilmesi** gerekir. Bu işlem hem kaynak planlamasını optimize eder hem de iş sürekliliğini korur.

Bu kapsamda uluslararası alanda en yaygın kullanılan metodoloji, **CVSS (Common Vulnerability Scoring System)**'dir. CVSS, açıklıkların etkisini çeşitli kriterler doğrultusunda sayısal olarak puanlayarak, nesnel bir risk skoru üretmeyi amaçlar. Elde edilen skor, sistemdeki diğer açıklıklarla kıyaslamaya, düzeltme süreci için

zamanlama yapmaya ve ilgili ekipler arasında sorumluluk dağıtımına temel oluşturur.

5.2.1 CVSS Nedir ve Neden Kullanılır?

CVSS, ilk olarak 2005 yılında NIST ve FIRST (Forum of Incident Response and Security Teams) iş birliğiyle geliştirilmiş, daha sonra CVSS v2 (2007) ve v3.1 (2019) sürümleriyle güncellenmiş bir açık skorlama sistemidir. Sistem, açıklıkları aşağıdaki üç temel başlık altında değerlendirir:

- **Temel Skor (Base):** Açığın teknik özelliklerine göre değişmeyen risk değeridir.
- **Geçici Skor (Temporal):** Açığın istismar edilebilirliği, mevcut exploit kodlarının olup olmaması gibi zamanla değişen parametreleri içerir.
- **Çevresel Skor (Environmental):** Kurumun sistem yapısı, işlevsel önem düzeyi ve hassasiyetlerine göre özelleştirilmiş risk katsayısıdır.

Çizelge 5.2. CVSS Hesaplama Kriterleri

Kriter	Açıklama
Attack Vector (AV)	Açığa erişim türü (Network, Adjacent, Local, Physical)
Attack Complexity (AC)	Açığın istismarı için gereken ön koşullar
Privileges Required (PR)	Açığı tetiklemek için gerekli erişim seviyesi
User Interaction (UI)	Kullanıcı etkileşimi gerektirip gerektirmediği
Confidentiality (C)	Açığın gizliliğe etkisi (None, Low, High)
Integrity (I)	Açığın veri bütünlüğüne etkisi
Availability (A)	Sistemin çalışabilirliğine etkisi

Her bir kriter belirli ağırlıklarda puanlanarak toplam CVSS skoru (0.0–10.0) arasında bir değer hesaplanır.

Çizelge 5.3. CVSS Skorlarının Sınıflandırılması

CVSS Skoru Aralığı	Risk Seviyesi	Düzeltme Önceliği
0.0 – 3.9	Düşük	Geciktirilebilir
4.0 – 6.9	Orta	Planlı müdahale
7.0 – 8.9	Yüksek	72 saat içinde müdahale
9.0 – 10.0	Kritik	Acil (24 saat içinde)

Bu skorlama ile birlikte yöneticiler, hangi zafiyetin "önce" kapatılması gerektiğini sistematik biçimde anlayabilir. CVSS skorları, daha görsel ve karar destek odaklı biçimde risk matrisleriyle de desteklenebilir. Aşağıda örnek bir 3x3 matris yer almaktadır:

Çizelge 5.4. Açıklık Etkisine ve Olasılığına Göre Risk Matrisi

	Düşük Olasılık	Orta Olasılık	Yüksek Olasılık
Düşük Etki	Düşük	Düşük	Orta
Orta Etki	Düşük	Orta	Yüksek
Yüksek Etki	Orta	Yüksek	Kritik

Bu matris, CVSS skorları dışında kurumsal hassasiyetlere göre özelleştirilebilir. Örneğin ödeme sistemine etki eden bir açık, teknik olarak "orta seviye" olsa da kurumsal önemi nedeniyle "kritik" kabul edilebilir.

Çizelge 5.5. E-Ticaret Sistemleri İçin Örnek Önceliklendirme Senaryoları

Zafiyet Türü	CVSS	İşlevsel Etki	Öncelik
XSS – Kullanıcı yorumları	6.4	Kimlik hırsızlığı	Orta
SQLi – Admin panel erişim	9.8	Tüm veritabanına erişim	Kritik
Açık dizin – logs klasörü	4.2	Sınırlı bilgi sızıntısı	Düşük
Zayıf parola politikası	7.1	Hesapların ele geçirilmesi	Yüksek

5.3 Otomatik Raporlama Araçları

Günümüz siber güvenlik uygulamalarında yalnızca zafiyetlerin tespit edilmesi değil, bu bulguların sistematik ve anlaşılabilir biçimde raporlanması da büyük önem taşımaktadır. Özellikle geniş ölçekli e-ticaret altyapılarında günlük güvenlik analizlerinin manuel yöntemlerle sürdürülmesi hem zaman hem de kaynak açısından sürdürülemez hâle gelmiştir. Bu nedenle açıklıkların otomatik olarak raporlanmasına imkân tanıyan sistemlerin kullanımı yaygınlaşmıştır.

Otomatik raporlama araçları; tarama sonuçlarını belli bir standarda göre sınıflandıran, çıktı formatlarını kullanıma uygun hâle getiren ve çoğu zaman yöneticilere ya da geliştirici ekiplerine entegre sistemler aracılığıyla ileten platformlardır. Bu araçlar, açıklığın türü, seviyesi ve etkilediği sistem bileşeni gibi bilgileri analiz ederek genellikle **HTML, PDF, XML veya JSON** formatlarında rapor üretirler. Üretilen raporlar sadece belgelenme amacıyla değil; aynı zamanda önceliklendirme, görev atama ve hukuki süreçlerin takibi açısından da kullanılır.

Yaygın olarak kullanılan bazı araçlar şunlardır:

- **OWASP ZAP**: Açık kaynaklı bir güvenlik tarayıcısı olan ZAP, HTML ve XML formatlarında raporlar sunmakta, CI/CD sistemlerine kolayca entegre olabilmektedir. Özellikle e-ticaret platformlarında otomatik XSS, SQL Injection ve oturum açıklıkları taramasında kullanılır.
- **Burp Suite Professional**: Tarama sonuçlarını kategorilere ayırarak teknik ve yönetici düzeyinde okunabilir raporlar sunar. Kritik açıklıklar için otomatik PoC (Proof of Concept) çıktısı oluşturabilir.
- **Nessus**: Özellikle ağ tabanlı zafiyetlerin raporlanmasında etkilidir. Gelişmiş sınıflandırma ve risk puanlama (CVSS) özellikleri sayesinde yöneticilere uygun çıktı üretir.
- **SonarQube**: Kod bazlı açıklıkları tarar ve çıktılarını CI/CD döngüsüne entegre ederek yazılım geliştirme sürecinin doğal parçası hâline getirir. Yazılımcılara kod satırı düzeyinde açıklık bildirimi yapar.

Bu sistemler sadece açık tespitiyle sınırlı kalmaz; aynı zamanda bir açık bulunduğunda belirli eşik değerine (örneğin CVSS ≥ 9.0) ulaşıyorsa, otomatik olarak e-posta, Slack mesajı veya proje yönetim araçları (Jira, Trello vb.) üzerinden görev

ataması da gerçekleştirebilir. Böylece açıklığın gözden kaçması önlenir ve müdahale süresi kısalır.

Otomatik raporlama sistemleri ayrıca **KVKK ve GDPR gibi yasal düzenlemelere uyum açısından da önemlidir**. Çünkü bu sistemler geçmişte tespit edilen tüm açıklıkların zaman damgalı olarak saklandığı arşivler oluşturur. Bu da denetim süreçlerinde kurumun güvenlik farkındalığını ve aksiyon geçmişini belgelemek adına büyük avantaj sağlar.

Sonuç olarak, otomatik raporlama araçları yalnızca teknik çıktılar sunmakla kalmaz, aynı zamanda kurumun genel siber güvenlik duruşunu daha şeffaf, hızlı ve sürdürülebilir bir hâle getirir. Bu araçların e-ticaret sistemlerine entegre edilmesi, güvenlik döngüsünün son adımında güvenilirlik ve kurumsal farkındalık sağlamaktadır.

5.4 Açık Özeti

Güvenlik açıklarının raporlanmasına yönelik model önerileri, genellikle literatürde yer alan analizler, sektör raporları ve OWASP gibi küresel ölçekte kabul gören kaynaklara dayandırılmaktadır. Bu çerçevede oluşturulan yapı doğrultusunda, e-ticaret sitelerinde **en sık rastlanan açıklıkların** belirli bir örüntü içinde tekrarlandığı görülmektedir. OWASP Top 10 listesi başta olmak üzere çok sayıda akademik çalışmada da ortaya konduğu üzere; SQL Injection (SQLi), Cross-Site Scripting (XSS), kimlik doğrulama zafiyetleri ve güvenlik yapılandırma hataları, web tabanlı sistemlerde öne çıkan zafiyet türleridir (OWASP, 2023; ENISA, 2023).

Bu kapsamda geliştirilen raporlama modelinin teorik altyapısı ve sektördeki yaygın açıklar dikkate alınarak, aşağıdaki tabloda simülasyon temelli bir örnek açıklık özeti sunulmuştur. Tablo, güvenlik testlerinde sık karşılaşılan başlıca açıklık türlerini ve bu açıklıkların örnek olay senaryolarındaki dağılımını göstermektedir. Amaç, modelin raporlama çıktılarında nasıl bir sınıflandırma yapısı kurguladığını göstermektir

Çizelge 5.6. Örnek Açıklık Özeti

Açıklık Türü	Teorik Senaryo Sayısı	Varsayımsal Oran (%)
SQL Injection	7	%35
XSS (Reflected/Persistent)	5	%25
Kimlik Doğrulama Zafiyetleri	4	%20
Güvenlik Yapılandırma Hataları	3	%15
Güncellenmemiş Bileşenler	1	%5

Tablodaki veriler, doğrudan uygulamaya dayalı bir test ortamından değil; açık güvenlik kaynaklarında raporlanan en yaygın açıklıkların tipik dağılımına dayalı **ön modelleme çıktısı** olarak değerlendirilmelidir.

5.5 Etki Seviyesi

Güvenlik açıklıklarının yalnızca varlığı değil, bu açıklıkların sistem üzerinde oluşturduğu **etki düzeyi** de kritik öneme sahiptir. Etki seviyesi, bir zafiyetin sömürülmesi durumunda sistemin hangi bileşenlerinin, hangi ölçüde zarar göreceğini belirler. Bu bağlamda, açıklıkların risk değerlendirmesi yapılırken teknik parametrelerin yanı sıra **etki alanı (scope)**, **veri türü**, **operasyonel kesinti riski** ve **yasal sonuçlar** gibi ölçütler de dikkate alınmalıdır (Anderson, 2020).

Uluslararası kabul görmüş metotlara göre bir güvenlik açığının etki düzeyi, genellikle üç temel boyutta değerlendirilir:

- **Gizlilik (Confidentiality):** Açığın, sistemdeki özel ya da kişisel verilerin sızdırılmasına neden olma potansiyelidir.
- **Bütünlük (Integrity):** Açığın, verilerin yetkisiz şekilde değiştirilmesine ya da bozulmasına sebep olabilme durumudur.
- **Kullanılabilirlik (Availability):** Açığın sistemin hizmet veremez hâle gelmesine, erişim kesintilerine veya işlem aksaklıklarına neden olma derecesidir.

Bu üç boyut, aynı zamanda **CVSS (Common Vulnerability Scoring System)** temel kriterlerinden biri olan **CIA üçlüsü** olarak da tanımlanmaktadır (FIRST.org, 2019). Örneğin, yalnızca bilgi sızdıran bir açıklık düşük gizlilik etkisine sahipken; tüm kullanıcı verilerini silme yeteneğine sahip bir açıklık, hem yüksek bütünlük hem de yüksek gizlilik etkisi barındırır.

Aşağıdaki tabloda bu sınıflandırma sisteminin örnek bir karşılığı sunulmaktadır:

Çizelge 5.7. Etki Seviyesini Gösteren Örnek Tablo

Etki Seviyesi	Gizlilik	Bütünlük	Kullanılabilirlik	Genel Değerlendirme
Düşük	Sınırlı veri açığı	Okunamayan veriler	Anlık yavaşlık	İzleme ile önlenebilir
Orta	Oturum bilgisi açığı	Veri manipülasyonu	Zaman aşımı/yeniden başlatma	Süreç aksar, ama kalıcı değil
Yüksek	Tüm kullanıcı verisi	Yetkisiz veri güncelleme	Servis dışı bırakma (DoS)	Kalıcı sistem zararı, itibar kaybı
Kritik	Kimlik bilgilerinin sızdırılması	Finansal veri değişimi	Kalıcı hizmet kesintisi	Hukuki sonuç doğurur, acil müdahale gerekir

Etki seviyesi yalnızca teknik düzeyde değil, aynı zamanda **iş sürekliliği**, **müşteri güveni** ve **regülasyonlara uyum** gibi yönetsel riskleri de doğrudan etkiler. Özellikle e-ticaret sistemlerinde, kullanıcı verisinin açığa çıkması yalnızca teknik bir açıklık değil, **KVKK ve GDPR kapsamında veri ihlali** olarak değerlendirilebilir. Bu tür ihlaller, finansal cezaların yanı sıra marka itibarını da zedelemektedir (Solove, 2007).

ENISA'nın 2023 tarihli *Threat Landscape Report* adlı çalışmasında da belirtildiği üzere, açıklıkların teknik ciddiyeti kadar işlevsel sonuçlarının da analiz edilmesi, risklerin daha doğru yönetilmesini sağlamaktadır. Dolayısıyla bu çalışmada önerilen

raporlama modeli, yalnızca açıkların türüne değil, oluşturdukları etki seviyesine göre de sınıflandırma yapacak biçimde tasarlanmıştır.

Etki seviyesinin net şekilde belirlenmesi, hem **önceliklendirme** aşamasında doğru kararlar alınmasını sağlar hem de müdahale sürelerinin planlanmasında rehber niteliği taşır. Kritik etkiye sahip açıklıklar için müdahale süresi 24 saat ile sınırlı tutulmalı; orta seviyedeki açıklıklar ise izleme ve loglama ile desteklenmelidir.

5.6 Önerilen Çözüm

Güvenlik açıklarının tespiti kadar, bu açıklıkların etkin şekilde giderilmesi de güvenli bir sistem altyapısının sürdürülebilirliği açısından hayati öneme sahiptir. Özellikle e-ticaret siteleri gibi hassas müşteri verilerini işleyen platformlarda, her bir açıklığın giderilme süreci hem teknik düzeyde hem de yönetsel düzeyde planlanmalıdır. Bu bağlamda önerilen çözüm yaklaşımı; tespit edilen açıklık türüne göre özelleştirilmiş müdahale stratejileri, yazılım geliştirme sürecine entegre güvenlik önlemleri ve yasal uyum süreçlerini içeren bütünsel bir yapıya dayanmaktadır.

5.6.1 Kod Tabanlı Zafiyetler için Güvenli Geliştirme Yaklaşımları

SQL Injection, XSS, Insecure Deserialization gibi kod temelli açıklıkların giderilmesinde en temel çözüm, güvenli yazılım geliştirme ilkelerinin uygulanmasıdır. Özellikle kullanıcı girdilerinin doğrudan veri tabanı sorgularına aktarılmasının engellenmesi ve parametrik sorgular (prepared statements) ile çalışılması kritik önemdedir. OWASP bu kapsamda geliştiricilere yönelik güvenli kodlama standartlarını şu şekilde sıralar:

“Input validation, output encoding and the principle of least privilege must be applied at all layers of the application stack.” (OWASP Secure Coding Practices, 2023)

Buna göre geliştiriciler, tüm form alanlarını, URL parametrelerini ve header bilgilerini hem istemci hem de sunucu tarafında doğrulamalı, ayrıca HTML çıktı üretiminde XSS koruması için escape() fonksiyonları kullanmalıdır. JavaScript yoğun sistemlerde ise CSP (Content Security Policy) başlığı yapılandırılarak yalnızca izin verilen kaynaklardan script çalıştırılması sağlanmalıdır.

5.6.2 Kimlik Doğrulama ve Oturum Güvenliği İçin Yapısal Önlemler

Kimlik doğrulama açıklıkları, sistemin en temel güvenlik hattını zayıflatmakta ve kullanıcı oturumlarının ele geçirilmesine neden olmaktadır. Bu tür açıklıkların giderilmesinde şu adımlar önerilmektedir:

- Güçlü parola politikaları uygulanmalı (minimum 12 karakter, özel karakter zorunluluğu),
- Oturum çerezlerine HttpOnly ve Secure bayrakları atanmalı,
- Giriş işlemlerine **iki faktörlü kimlik doğrulama (2FA)** zorunluluğu getirilmelidir.

NIST tarafından yayınlanan SP 800-63B standardı da bu konuda şunları belirtmektedir:

"All authentication secrets shall be protected using multi-factor methods and transmitted over secure channels only." *(NIST, 2022)*

Bu tür önlemler sadece teknik değil, aynı zamanda regülasyon uyumu açısından da zorunludur. Özellikle GDPR ve KVKK gibi veri güvenliği yasaları, oturum yönetiminin açık bırakılmasını doğrudan veri ihlali olarak değerlendirmektedir (Solove, 2007).

5.6.3 Güncellenmeyen Bileşenler ve Yapılandırma Hataları için Süreçsel Çözümler

Zayıf ve eski yazılım bileşenleri, sistemin dış saldırılara açık kalmasına neden olur. Özellikle CMS (WordPress, Magento, nopCommerce) gibi yapılar kullanan e-ticaret platformlarında, sistem güncellemeleri sıklıkla ihmal edilmektedir. Bu nedenle aşağıdaki yapılar önerilir:

- Tüm yazılım bileşenleri için **otomatik versiyon takip sistemleri** (örn. Dependabot/Renovate) kurulmalı,
- Kullanılmayan eklenti ve modüller sistemden tamamen kaldırılmalıdır,
- Varsayılan şifreler, test hesapları ve debug erişimleri devre dışı bırakılmalıdır.

Ek olarak, yapılandırma dosyalarının (appsettings.json, web.config, config.php, .env vb.) erişimi sınırlandırılmalı ve dosya izinleri asgari yetki ilkesine göre düzenlenmelidir (Linux: chmod/umask; Windows: NTFS ACL – icacls).

5.6.4 Sürekli İzleme, Alarm ve Denetim Süreçleri

Tespit edilen açıklıkların tekrar oluşmaması için sistemlerin sürekli izlenmesi gereklidir. Bu kapsamda önerilen çözümler:

- **SIEM** (Security Information and Event Management) sistemlerinin kurulması
- Log takibi ve uyarı üretimi için ELK Stack (Elasticsearch, Logstash, Kibana) entegrasyonu
- Kritik olaylarda anında bildirim üreten alarm sistemleri (Slack, E-posta, SMS)

Bu tür sistemler, sadece saldırıyı değil, saldırı öncesi anormal davranışları da belirleyerek önleyici müdahale imkânı sunar. ENISA (2023) bu yaklaşımı "behavioral anomaly detection" olarak tanımlamaktadır ve raporlarında özellikle orta düzeyli kurumlar için önerilmektedir.

6. ARAŞTIRMA BULGULARI: UYGULAMA VE VAKA ÇALIŞMASI

6.1 Test Ortamının Kurulması

6.1.1 Hedef Sistem Alt Yapı Tanımı

Bu çalışmanın ilk adımı, **nopCommerce** tabanlı e-ticaret web uygulamasını izole bir laboratuvar ortamında çalıştırmaktır. Böylece, dış etkenlerden bağımsız olarak güvenlik testleri yapılabilir ve elde edilen sonuçlar tekrarlanabilir nitelikte olur. İzole bir yapı, test sırasında sistemin gerçek kullanıcılar veya dış ağlarla etkileşime girmesini engeller. Bu da hem güvenliği hem de deneyin bütünlüğünü korur.

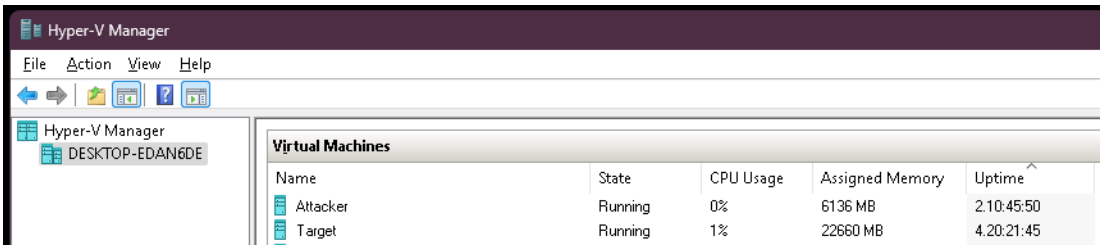
Test ortamı, **Hyper-V** sanallaştırma platformu üzerinde yapılandırılmış iki adet Windows 10 sanal makineden oluşmaktadır. Her iki makine, yalnızca kendi aralarında iletişim kurabilmeleri için **izole bir vSwitch** üzerinden birbirine bağlanmıştır.

Çizelge 6.1. Hedef Sistem Alt Yapıları

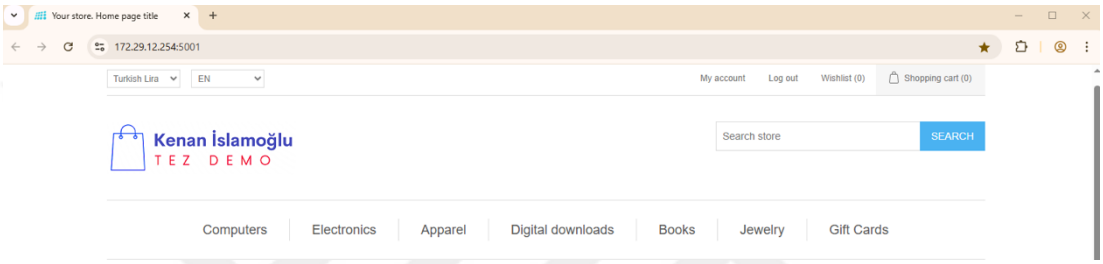
Rol	İşletim Sistemi	Makine Adı	Kaynaklar (RAM)	IP/Port(lar)	Açıklama
Target	Windows 10	Target	24 GB	172.29.12.254:5001 (HTTPS), 172.29.12.254:5000 (HTTP)	**nopCommerce** uygulaması **Kestrel** üzerinde yayınlanır.
Attacker	Windows 10	Attacker	24 GB	Aynı vSwitch (DHCP/statik)	Test ve tarama araçlarını barındırır.

Kurulan sanal ağ tamamen dış dünyadan kopuktur. Sadece **Attacker** ile **Target** makineleri arasında veri trafiğine izin verilmektedir. Böyle bir yapı, saldırı senaryolarının kontrollü bir ortamda denenmesine olanak tanır. Tüm testler, hedef sistem sahibinin bilgisi ve onayı dâhilinde, yalnızca bu izole laboratuvar içerisinde gerçekleştirilmiştir.

TLS/SAN yapılandırması ve port ayarlarına ilişkin ayrıntılar **Bölüm 6.1.3**'te sunulmuştur.



Şekil: 6.1. Hyper-V Manager – Attacker ve Target VM'ler



Şekil 6.2. nopCommerce ana sayfası (HTTPS)

6.1.2 Gerekli Yazılım ve Araçların Kurulumu

Test ortamında iki farklı sanal makine görev almıştır: **Target** (uygulama sunucusu) ve **Attacker** (test istasyonu). Her ikisinde de, görevlerine uygun yazılım ve araçlar kurulmuştur.

Target (Uygulama Sunucusu)

Uygulama sunucusunun kurulumu sırasında aşağıdaki yazılım bileşenleri yüklenmiştir:

- **Visual Studio 2022 Community Edition**
- **.NET SDK 6** ve **.NET SDK 7** – Uygulamanın farklı sürümlerle uyumlu şekilde derlenebilmesi için iki versiyon da hazır bulundurulmuştur.
- **SQL Server Express 2022** ile birlikte **SQL Server Management Studio (SSMS)**
- **nopCommerce** (güncel sürüm) – Visual Studio üzerinden derlenerek **Kestrel** üzerinde çalıştırılmıştır.
- **Windows Firewall** üzerinde yalnızca test ağına özel olacak şekilde 5001/TCP (HTTPS) portu için inbound kural tanımlanmıştır.

Attacker (Test İstasyonu)

Test istasyonu, güvenlik taramaları ve analizler için aşağıdaki araçlarla donatılmıştır:

- **OWASP ZAP** – Web uygulaması güvenlik testi
- **Burp Suite Community Edition** – HTTP trafiği analizi ve manuel testler
- **Nmap** – Port ve servis tarama aracı
- **Nikto** – Opsiyonel olarak, Perl ile çalıştırılan HTTP güvenlik başlığı ve konfigürasyon analiz aracı

Kurulum Sonrası Doğrulamalar

Kurulum tamamlandıktan sonra ortamın çalışır durumda olduğundan emin olmak için şu adımlar izlenmiştir:

1. **Bağlantı Testi:** Attacker makinesinden https://172.29.12.254:5001/ adresine erişim sağlanmıştır.
2. **Servis Doğrulama:** Nmap ile yapılan tarama sonucunda Target üzerinde 5001/TCP portunun açık olduğu ve **TLS** desteğinin aktif olduğu doğrulanmıştır.
3. **Proxy Testi:** OWASP ZAP ve Burp Suite ile HTTP/S trafiği başarıyla yakalanmış ve analiz edilebilmiştir.

6.1.3 Ağ ve Sertifika Yapılandırması

IP ve Ağ Yapılandırması

Test ortamında, hedef sistemin sabit bir ağ kimliğine sahip olması için Target sanal makineye statik IP atanmıştır: 172.29.12.254.

Uygulama, Kestrel web sunucusu üzerinden hem güvenli (5001/TCP – HTTPS) hem de güvenli olmayan (5000/TCP – HTTP) portlarda yayın yapacak şekilde yapılandırılmıştır.

Sertifika Oluşturma ve Kurulum

Tarayıcı güvenlik uyarılarını önlemek ve IP tabanlı bağlantılarda **TLS** protokolünün sorunsuz çalışmasını sağlamak amacıyla, **Subject Alternative Name (SAN)** alanında IP adresi tanımlanmış **self-signed** bir sertifika oluşturulmuştur. Bu sayede, tarayıcılar sertifikayı geçerli olarak tanıyabilmektedir.

PowerShell ile Sertifika Oluşturma (Target üzerinde):

```
powershell

$ip = "172.29.12.254"
$params = @{
    Subject           = "CN=$ip"
    FriendlyName      = "Local HTTPS development certificate"
    TextExtension     = @(
        "2.5.29.37={text}1.3.6.1.5.5.7.3.1",        # Server Authe
ntication EKU
        "2.5.29.17={text}ipaddress=$ip"             # SAN alanına
IP ekleme
    )
    NotAfter          = (Get-Date).AddDays(397)     # Maksimum TL
S geçerlilik süresi
    CertStoreLocation = "Cert:\LocalMachine\My"
}
$cert = New-SelfSignedCertificate @params
$pass = ConvertTo-SecureString -String "gucluSifreBuraya" -Force -
AsPlainText
Export-PfxCertificate -Cert $cert -FilePath "C:\temp\mycert.pfx" -
Password $pass
Export-Certificate -Cert $cert -FilePath "C:\temp\mycert.cer" -Typ
e CERT
```

Şekil 6.3. PowerShell ile Sertifika Oluşturması

Sertifikayı Güvenilir Kök Sertifika Deposuna Ekleme (Target ve Attacker):

```
powershell

Import-Certificate -FilePath "C:\temp\mycert.cer" -CertStoreLocati
on "Cert:\LocalMachine\Root"
```

Şekil 6.4. Sertifikayı Güvenilir Kök Sertifika Deposuna Ekleme

Kestrel'in Sertifikayı Kullanacak Şekilde Yapılandırılması (launchSettings.json):

```
json

{
  "profiles": {
    "Nop.Web": {
      "commandName": "Project",
      "launchBrowser": true,
      "environmentVariables": {
        "ASPNETCORE_ENVIRONMENT": "Development",
        "ASPNETCORE_Kestrel__Certificates__Default__Path": "C:\\te
mp\\mycert.pfx",
        "ASPNETCORE_Kestrel__Certificates__Default__Password": "gu
cluSifreBuraya",
        "ASPNETCORE_URLS": "https://172.29.12.254:5001;http://172.
29.12.254:5000"
      },
      "applicationUrl": "https://172.29.12.254:5001;http://172.29.
12.254:5000"
    }
  }
}
```

Şekil 6.5. Kestrel'in Sertifikayı Kullanacak Şekilde Yapılandırılması

Doğrulama Adımları

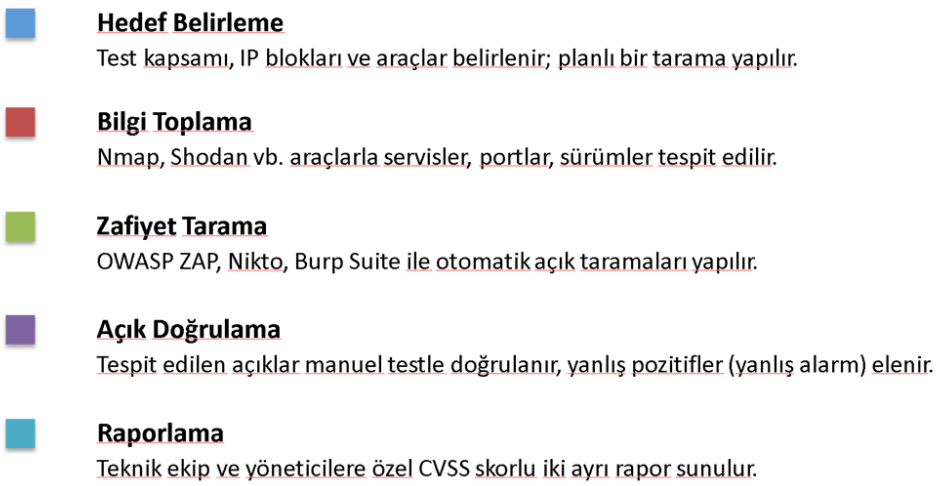
1. **Bağlantı Testi:** Attacker makineden https://172.29.12.254:5001/ adresine erişim sağlandığında tarayıcı bu bağlantıyı güvenli olarak işaretlemiştir.
2. **Sertifika Kontrolü:** Sertifika ayrıntılarında, **Subject Alternative Name** alanında IP adresi görüntülenmiştir.
3. **Sonuç:** Hem Target hem de Attacker makineleri arasında güvenli HTTPS iletişimi başarıyla kurulmuştur.

6.2 Sistem Tasarımının Uygulanması

Bu çalışmada geliştirilen test modeli, güvenlik değerlendirme sürecini planlı, sistematik ve tekrarlanabilir hale getirmek amacıyla tasarlanmıştır. Model beş temel aşamadan oluşmakta ve her adım bir sonrakini besleyecek şekilde yapılandırılmıştır. Böylece test süreci baştan sona kontrollü bir şekilde ilerler.

Test Süreci Akışı:

1. **Hedef Belirleme:** Öncelikle test edilecek uygulamanın kapsamı netleştirilmiş; IP adresi, port bilgileri ve kullanılan teknolojiler belirlenmiştir.
2. **Bilgi Toplama:** Hedef sistem hakkında hem açık kaynak istihbaratı hem de aktif tarama yöntemleri ile bilgi toplanmıştır.
3. **Zafiyet Tarama:** Elde edilen bilgiler ışığında, otomatik tarama araçları kullanılarak potansiyel güvenlik açıkları belirlenmiştir.
4. **Açık Doğrulama:** Tespit edilen her açık, manuel testlerle doğrulanarak yanlış pozitif olasılığı ortadan kaldırılmıştır.
5. **Raporlama:** Doğrulanmış bulgular teknik rapor formatında ayrıntılı olarak sunulmuş, ayrıca yöneticilerin hızlıca anlayabileceği bir özet hazırlanmıştır.



Şekil 6.6. Modelin Mimari Yapısı

Hedef Belirleme → Bilgi Toplama → Zafiyet Tarama → Açık Doğrulama → Raporlama

6.2.1 Bilgi Toplama

Test sürecinin bu aşamasında amaç, hedef sistemin **ağ yapısı**, **açık portları**, **çalışan servisleri** ve **kamuya açık verileri** hakkında olabildiğince kapsamlı bilgi toplamaktır.

Bu süreçte hem **pasif** hem de **aktif** bilgi toplama yöntemlerinden yararlanılmıştır.

- **Pasif bilgi toplama** adımında, sistemle doğrudan etkileşime girmeden, açık kaynaklar ve çevrim içi veri tabanları üzerinden araştırma yapılmıştır.
- **Aktif bilgi toplama** ise hedef sistemle doğrudan iletişim kurarak port, servis ve yapılandırma bilgilerinin elde edilmesini sağlamıştır.

Bu aşama iki alt başlık altında ele alınmaktadır:

6.2.1.1 *Port ve Servis Taraması (Nmap)*

Bu adımda hedef sistem üzerinde hangi portların açık olduğu ve bu portlarda hangi servislerin çalıştığı belirlenmiştir.

Tespit için **Nmap** aracı kullanılmış hem port numaraları hem de servis sürümleri taranmıştır.

Kullanılan komut:

```
nmap -sV -p- 172.29.12.254
```

Şekil 6.7. Tespit İçin Kullanılan Komut

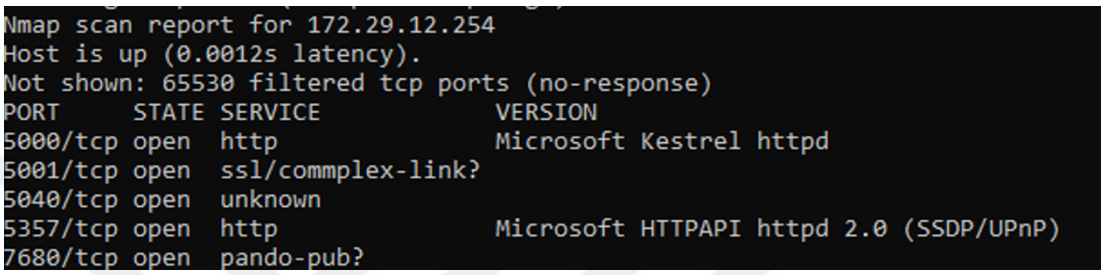
Tarama Sonuçları:

Çizelge 6.2. Tarama Sonuçları

Port	Protokol	Servis Adı	Servis Versiyonu / Açıklama
5000	TCP	Microsoft Kestrel HTTP	HTTP tabanlı web sunucusu
5001	TCP	Microsoft Kestrel HTTPS	TLS destekli web sunucusu
5357	TCP	Microsoft HTTPAPI 2.0	SSDP/UPnP servisi, ağ keşfi amaçlı kullanılabilir
7680	TCP	pando-pub?	Tanımlanamayan servis, ek analiz gerektirir

Değerlendirme:

- Sistem hem **HTTP (5000/TCP)** hem de **HTTPS (5001/TCP)** portlarından erişim sunmaktadır.
- **5357/TCP** portu, saldırganların ağ keşfi amacıyla kullanabileceği **SSDP/UPnP** servislerini barındırmaktadır.
- **7680/TCP** üzerinde çalışan servis tam olarak tanımlanamamıştır; bu nedenle ek güvenlik analizi yapılması önerilmektedir.



Şekil 6.8. Ekran Görüntüsü: Nmap – Port ve Servis Taraması Sonuçları

6.2.1.2 *Açık Kaynak İstihbaratı (Shodan vb.)*

Bu aşamada, hedef sistemin internet üzerindeki olası izleri ve erişime açık servisleri, **Açık Kaynak İstihbaratı (OSINT)** yöntemleri kullanılarak araştırılmıştır. Amaç, kamuya açık veri kaynaklarından yararlanarak sistemin görünürlüğünü, kullanılan teknolojileri ve olası güvenlik risklerini tespit etmektir.

Kullanılan Araçlar ve Platformlar:

- **Shodan:** İnternete açık cihazlar, servisler ve yazılım sürümlerini indeksleyen arama motoru.
- **Censys:** SSL/TLS sertifikaları ve servis bilgileri üzerinden altyapı taraması yapılabilen platform.
- **Netcraft:** Barındırma sağlayıcısı, DNS kayıtları, SSL sertifikaları ve geçmiş veriler hakkında bilgi sunan hizmet.

İzlenen Yöntem:

1. **Shodan Taraması:** IP adresi, alan adı ve SSL sertifika bilgileri kullanılarak arama yapılmıştır.
2. **Censys Sorguları:** Sertifika imzaları üzerinden aynı altyapıyı paylaşan sistemler tespit edilmeye çalışılmıştır.
3. **Netcraft Analizi:** Barındırma altyapısı, DNS kayıt geçmişi ve sertifika değişim bilgileri incelenmiştir.

Test Ortamına İlişkin Durum:

Hedef sistem, yerel IP adresi (**172.29.12.254**) üzerinden çalıştığı için bu tür OSINT araçlarında herhangi bir kayıt bulunmamaktadır. Dolayısıyla gerçek bir çıktı elde edilememiş, ancak yöntem bir örnek senaryo üzerinden açıklanmıştır.

Çizelge 6.3. Örnek Shodan Çıktısı (Eğitim Amaçlı)

Açık Port	Servis	Yazılım Sürümü	Notlar
80	HTTP	Apache 2.4.49	CVE-2021-41773 zafiyetine karşı savunmasız
443	HTTPS	Nginx 1.18.0	TLSv1.0 desteği etkin
22	SSH	OpenSSH 7.4	Brute-force saldırılarına karşı ek önlemler önerilir

Analiz ve Öneriler:

- İnternete açık gereksiz servisler kapatılmalı, yalnızca gerekli servisler erişime açık bırakılmalıdır.
- Eski ve zayıf şifreleme protokolleri (örn. TLSv1.0) devre dışı bırakılmalı, güncel sürümler kullanılmalıdır.
- Elde edilen OSINT verileri, sonraki aşamalardaki zafiyet taraması ve önceliklendirme sürecine girdi olarak kullanılabilir.

6.2.2 Zafiyet Tarama

6.2.2.1 *SSL/TLS Konfigürasyon Analizi (nmap ssl-enum-ciphers)*

Hedef sistemde kullanılan TLS sürümleri ve şifre takımlarının güvenlik seviyesini değerlendirmek amacıyla, nmap aracının **ssl-enum-ciphers** betiği çalıştırılmıştır. Bu tarama, kullanılan protokol sürümlerini, desteklenen şifreleme algoritmalarını ve potansiyel zafiyetleri ortaya koymayı hedeflemiştir.

```
nmap --script ssl-enum-ciphers -p 5001 172.29.12.254
```

Şekil 6.9. Kullanılan Komut

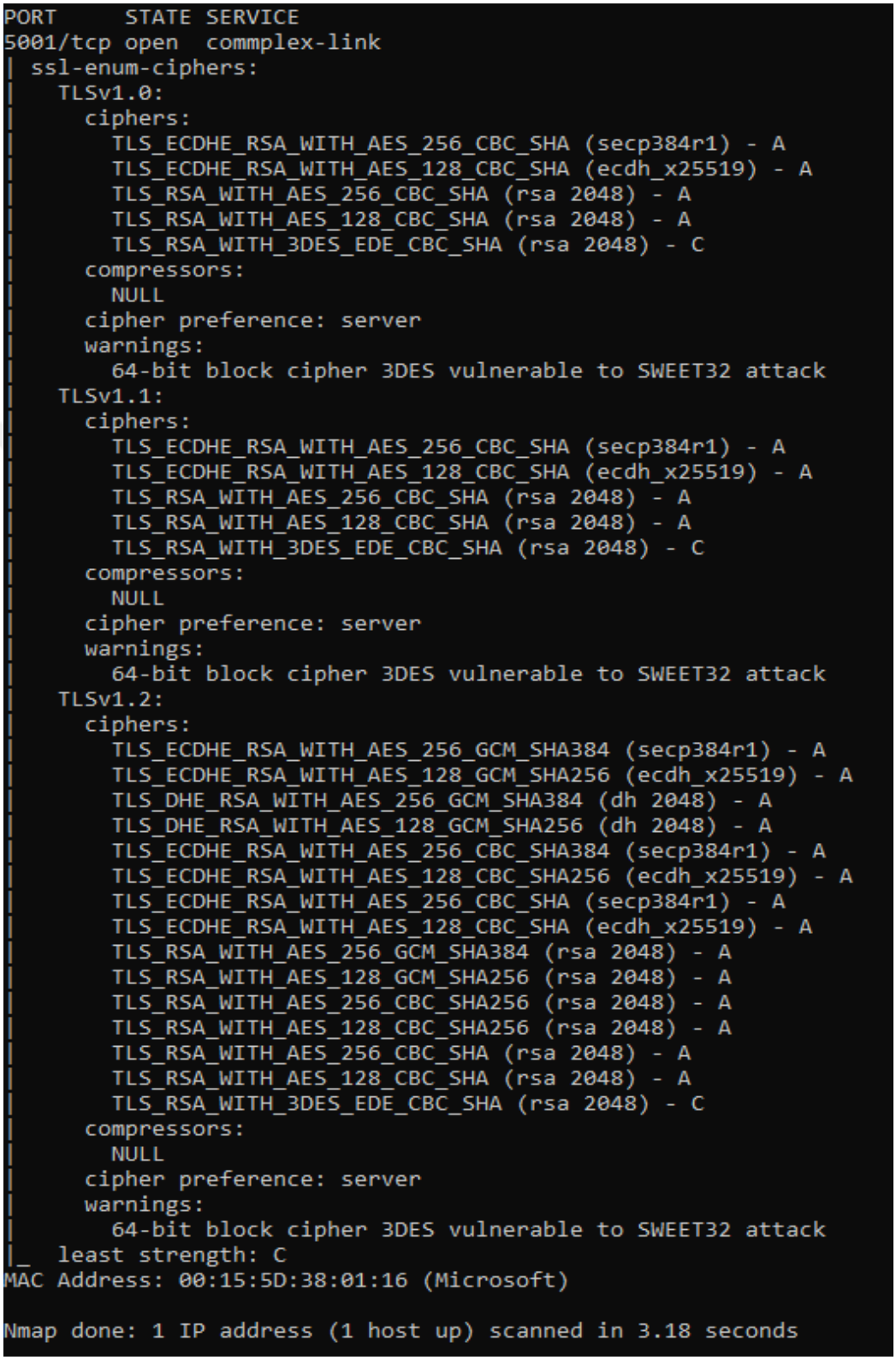
Çizelge 6.4. Elde Edilen Bulgular

TLS Sürümü	Güçlü Şifre Takımları	Zayıf / Orta Seviye Şifre Takımları	Notlar
TLSv1.0	AES256, AES128	3DES (SWEET32)	Güncel olmayan protokol
TLSv1.1	AES256, AES128	3DES (SWEET32)	Güncel olmayan protokol
TLSv1.2	AES256 GCM, AES128 GCM	3DES (SWEET32)	3DES desteği kaldırılmalı

Değerlendirme:

- Hem **TLSv1.0** hem de **TLSv1.1** sürümleri güncel güvenlik standartlarına göre kullanım dışı bırakılmalıdır.
- 3DES algoritması, **SWEET32** zafiyetine karşı savunmasızdır ve modern sistemlerde önerilmemektedir.
- Sunucu yapılandırması, yalnızca **TLSv1.2** (güçlü şifre takımları ile) ve tercihen **TLSv1.3** desteği sağlayacak şekilde güncellenmelidir.

- TLSv1.3 desteği, hem güvenlik seviyesini artıracak hem de modern tarayıcılar ve istemcilerle uyumluluğu güçlendirecektir.



Şekil 6.10. Ekran Görüntüsü: SSL/TLS Tarama Çıktısı (nmap ssl-enum-ciphers)

6.2.2.2 *HTTP Güvenlik Başlıkları Analizi (Nikto)*

Hedef sistemde kullanılan HTTP yanıt başlıklarının güvenlik açısından yeterliliğini değerlendirmek amacıyla **Nikto** aracı ile tarama gerçekleştirilmiştir. Bu analiz, tarayıcı–sunucu iletişiminde güvenliği artıran başlıkların varlığını ve doğru yapılandırılıp yapılandırılmadığını incelemektedir.

```
"C:\Strawberry\perl\bin\perl.exe" C:\nikto\program\nikto.pl -h htt
ps://172.29.12.254:5001 -ssl
```

Şekil 6.11. HTTP Güvenlik Başlıkları Analizi İçin Kullanılan Komut

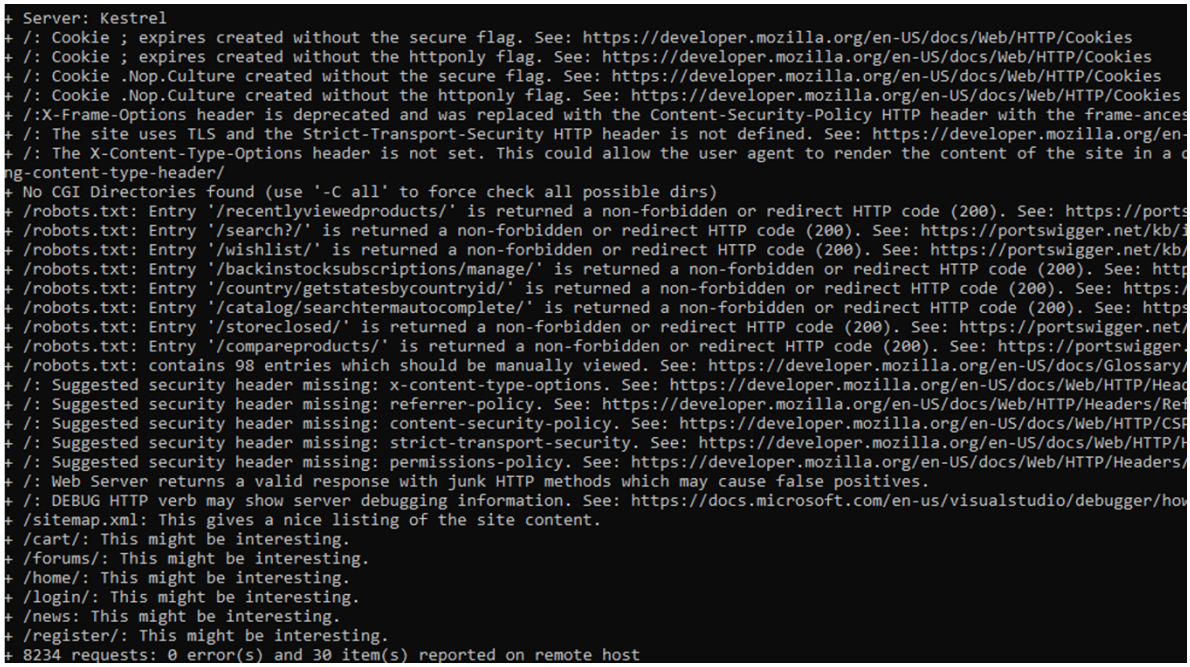
Tespit Edilen Eksiklikler ve Bulgular:

- Bazı HTTP yanıtlarında kullanılan çerezlerde (**Cookie**) **Secure** ve **HttpOnly** bayrakları tanımlanmamıştır.
- **X-Frame-Options** başlığı mevcut olmakla birlikte, güncel standartlarda **deprecated** olarak değerlendirilmekte; bunun yerine **Content-Security-Policy** içerisinde **frame-ancestors** yönergesinin kullanılması önerilmektedir.
- **Strict-Transport-Security** başlığı bulunmamaktadır.
- **X-Content-Type-Options** başlığı eksiktir.
- **Content-Security-Policy** başlığı eksiktir.
- /robots.txt dosyası herkese açık olup, içerisinde erişime açık bazı URL'ler listelenmiştir.
- Potansiyel olarak saldırı yüzeyini artırabilecek endpoint'ler (/cart/, /login/, /register/ vb.) tespit edilmiştir.

Öneriler:

1. **Eksik Güvenlik Başlıklarının Eklenmesi:**
    - **Content-Security-Policy**
    - **Strict-Transport-Security**
    - **Permissions-Policy**
    - **Referrer-Policy**

2. **Çerez Güvenliğinin Artırılması:**
    - **Secure** bayrağı eklenerek çerezlerin yalnızca HTTPS üzerinden iletilmesi sağlanmalıdır.
    - **HttpOnly** bayrağı eklenerek istemci tarafı erişimleri engellenmelidir.
3. **Gereksiz Dizin ve Endpoint Yönetimi:**
    - Tarama sonucu tespit edilen ve gereksiz erişim sunan dizinler/endpoint'ler kapatılmalı veya erişim kontrol mekanizmaları ile sınırlandırılmalıdır.



Şekil 6.12. Ekran Görüntüsü: Nikto HTTP Güvenlik Başlıkları Analiz Çıktısı

6.2.2.3 *OWASP ZAP ile Web Uygulaması Taraması*

Bu alt adımda, hedef web uygulamasında mevcut olabilecek güvenlik açıklarının belirlenmesi amacıyla **OWASP Zed Attack Proxy (ZAP)** aracı kullanılmıştır. OWASP ZAP, yaygın web güvenlik zafiyetlerini (örneğin XSS, SQL Injection, güvenlik başlığı eksiklikleri) tespit edebilen hem manuel hem de otomatik tarama yapabilen açık kaynaklı bir güvenlik test aracıdır.

Kullanılan Araç ve Sürüm:

- **OWASP ZAP v2.14.0**
- **Çalışma Ortamı:** Windows 10 + Java 17

Uygulama Adımları:

1. **Hedef Belirleme:**

   "Quick Start" sekmesinde hedef URL olarak test ortamındaki HTTPS servisi girilmiştir:

```
https://172.29.12.254:5001
```

Şekil 6.13. Hedef Belirlemek İçin Kullanılan HTTPS Servisi

2. **Tarama Türü:**
   - **Active Scan** modu tercih edilmiştir.
   - Self-signed sertifikadan kaynaklı hataları önlemek amacıyla SSL/TLS doğrulaması devre dışı bırakılmıştır.
   - Oturum açma gerektiren sayfa bulunmadığından kimlik doğrulama yapılandırması yapılmamıştır.
3. **Tarama Parametreleri:**
   - Tarama derinliği: Orta (varsayılan)
   - Kullanılan eklentiler: "Passive Scanner", "Active Scanner", "Ajax Spider"

Çizelge 6.5. Örnek Tespit Edilen Bulgular

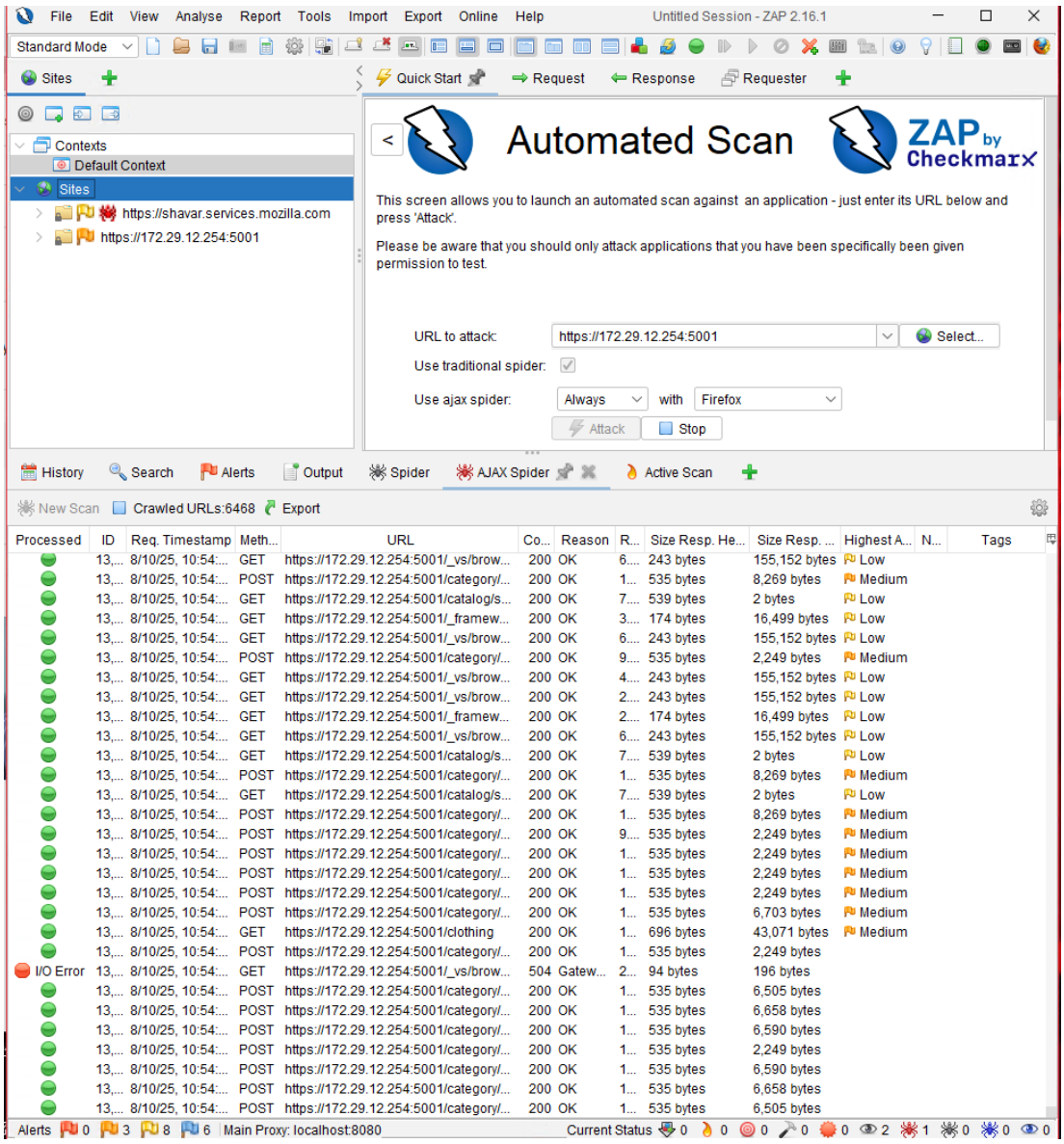
ID	Zafiyet Türü	URL	Risk Seviyesi	Açıklama
1	X-Content-Type-Options Eksikliği	/	Orta	MIME tip tabanlı saldırılara karşı koruma yok.
2	Content-Security-Policy Eksikliği	/	Orta	XSS'e karşı ek güvenlik politikası uygulanmamış.
3	Strict-Transport-Security Eksikliği	/	Orta	HTTPS zorunluluğu için HSTS yapılandırılmamış.
4	X-Frame-Options SAMEORIGIN	/	Bilgi	Clickjacking'e karşı koruma var; ancak "DENY" daha güvenlidir.
5	robots.txt İçeriği	`/robots.txt`	Bilgi	`/admin/` gibi dizinler listelenmiş olabilir.

Analiz:

- OWASP ZAP çıktıları, **Nikto** taramasında tespit edilen HTTP güvenlik başlığı eksiklikleri ile örtüşmektedir.
- Eksik güvenlik başlıkları, saldırı yüzeyini genişletebilir.
- robots.txt dosyasında listelenen dizinler, saldırganlar için dizin keşfi sürecinde faydalı olabilir.

Öneriler:

1. Eksik HTTP güvenlik başlıklarının eklenmesi:
    - Content-Security-Policy
    - Strict-Transport-Security
    - X-Content-Type-Options
2. Gereksiz dizinlerin robots.txt üzerinden listelenmemesi.
3. TLS yapılandırmasının yalnızca TLSv1.2+ ve güvenli şifre takımlarını destekleyecek şekilde güncellenmesi.



Şekil 6.14. OWASP ZAP ile Web Uygulaması Taraması Sonuç Ekranı

Toplam **6.468 URL** taranmış hem klasik spider hem de AJAX spider yöntemleri kullanılmıştır. Tarama sırasında farklı HTTP metodları (GET, POST vb.) denenmiş; "Content-Security-Policy Eksikliği", "Anti-Clickjacking Header Eksikliği" ve "Vulnerable JavaScript Library" gibi farklı risk seviyelerine sahip bulgular tespit edilmiştir.

6.2.2.4 *Burp Suite ile Zafiyet Analizi*

Bu alt adımda, hedef sistem üzerinde **Burp Suite** aracı kullanılarak web uygulaması güvenlik testi gerçekleştirilmiştir. Burp Suite, HTTP(S) trafiğini yakalayabilen, değiştirebilen ve bu sayede web uygulamalarında hem otomatik hem de manuel zafiyet analizine imkân tanıyan yaygın bir güvenlik test aracıdır.

Uygulama Adımları:

1. **Proxy Yapılandırması:**

   Tarayıcı trafiği Burp Suite üzerinden yönlendirilmiştir. Hedef sistemin HTTPS trafiğini analiz edebilmek için Burp Suite tarafından oluşturulan sertifika tarayıcıya eklenmiştir.

2. **Pasif Analiz:**

   Kullanıcı işlemleri sırasında tüm HTTP istek ve yanıtları pasif olarak incelenmiştir. Bu aşamada, güvenlik başlığı eksiklikleri, bilgi sızıntısı potansiyeli ve uyarı gerektiren durumlar tespit edilmiştir.

3. **Aktif Tarama:**

   Kritik sayfalar (/register, /cart vb.) üzerinde otomatik zafiyet taraması gerçekleştirilmiştir. Bu taramada SQL Injection, XSS, CSRF, komut enjeksiyonu ve zayıf şifreleme algoritmaları test edilmiştir.

4. **Manuel Testler:**

   Kullanıcı kayıt formu özelinde manuel testler yapılmıştır. FirstName, LastName ve Email alanlarına SQL Injection payload'ları (örn. ' OR '1'='1, test'--) gönderilerek sunucu tepkileri analiz edilmiştir.

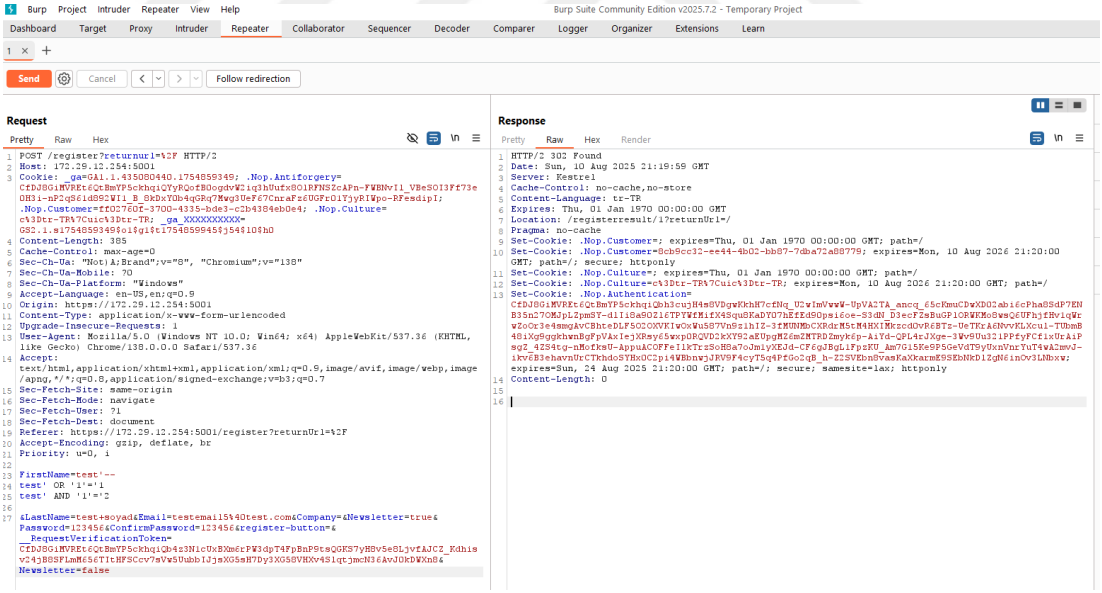
Test Sonuçları:

- **SQL Injection:** Parametreli sorgular veya ORM kullanıldığı gözlemlenmiş, yetkisiz veri erişimi veya veritabanı hatası oluşmamıştır.

- **XSS (Cross-Site Scripting):** <script>alert(1)</script> gibi payload'lar çalışmamış; girdi filtreleme mekanizmasının etkin olduğu anlaşılmıştır.
- **CSRF (Cross-Site Request Forgery):** Önemli işlemler için CSRF token kontrolü bulunduğu ve doğrulamanın geçerli olduğu tespit edilmiştir.
- **Captcha Eksikliği:** Kayıt formunda Google reCAPTCHA veya benzeri bir bot koruma mekanizmasının yer almaması, otomatik bot saldırılarına karşı potansiyel bir risk olarak değerlendirilmiştir.
- **HTTP Başlık Güvenliği:** X-Frame-Options ve Content-Security-Policy gibi güvenlik başlıklarının eksikliği tespit edilmiştir.

Değerlendirme:

Analiz sonucunda kritik seviyede bir güvenlik açığı tespit edilmemiştir. Ancak, bot saldırılarına karşı koruma eksikliği ve bazı güvenlik başlıklarının bulunmaması, sistemin uzun vadeli güvenlik dayanıklılığını olumsuz etkileyebilecek unsurlar olarak değerlendirilmektedir.



Şekil 6.15. Burp Suite ile kullanıcı kayıt formunda SQL Injection denemesi

Burp Suite ile kullanıcı kayıt formunda FirstNameparametresine SQL Injection denemesi yapılmış ve sunucu tarafından **HTTP 302 yönlendirme** yanıtı dönülmüştür. Bu durum, girişimin engellendiğini veya hata mesajının kullanıcıya yansıtılmadığını göstermektedir.

6.2.3 Açık Doğrulama ve Yanlış Pozitiflerin Eliminasyonu

Güvenlik testleri sırasında tespit edilen her potansiyel zafiyet, otomatik tarama araçlarının raporladığı ham bulgular üzerinden detaylı olarak incelenmiştir. Bu aşamanın temel amacı, sistemde gerçekten bir güvenlik açığının bulunup bulunmadığını doğrulamak ve **yanlış pozitif** (false positive) olarak adlandırılan, gerçekte güvenli olan durumları nihai rapordan çıkarmaktır.

Yanlış pozitiflerin oluşma nedenleri literatürde genellikle şu başlıklar altında toplanmaktadır:

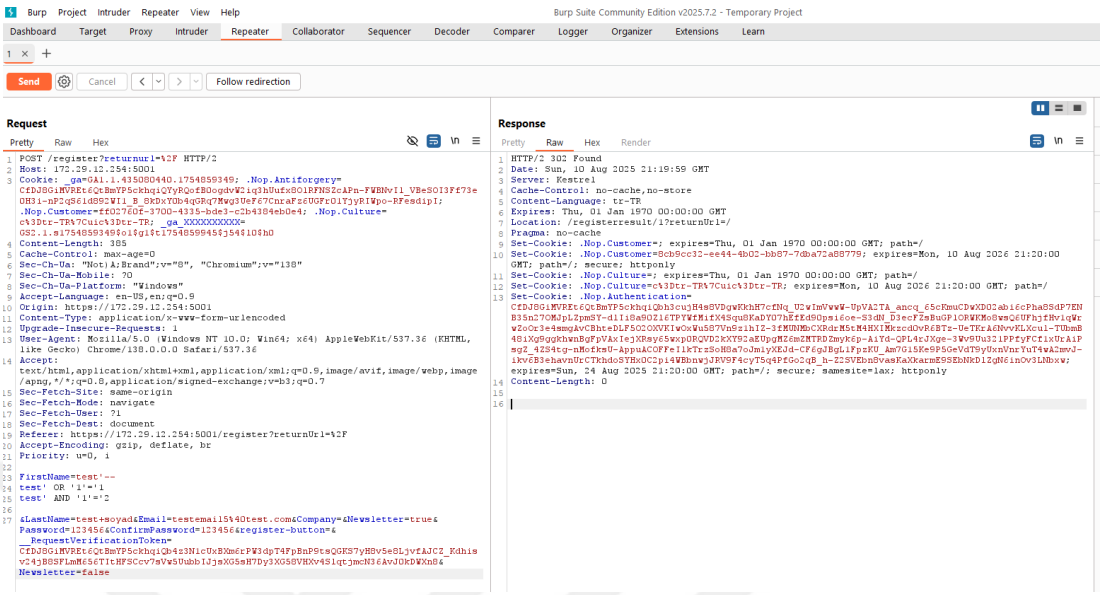
- **Yanlış yapılandırılmış tarama profilleri:** Tarama araçlarının hatalı veya gerçekçi olmayan parametrelerle çalıştırılması, yanlış alarm üretilmesine yol açabilir.
- **Özel hata sayfaları:** Uygulamanın kullanıcıya özelleştirilmiş hata mesajları sunması, aracın bu mesajı bir güvenlik açığı göstergesi olarak değerlendirmesine sebep olabilir.
- **Beklenen davranışın saldırı gibi algılanması:** Örneğin, form alanlarında HTML veya SQL karakterlerinin kabul edilmesi ancak arka planda güvenli şekilde işlenmesi.

Bu çalışmada, Burp Suite tarafından raporlanan olası SQL Injection bulgularından biri, **Repeater** modülü kullanılarak manuel olarak yeniden test edilmiştir. Yapılan inceleme sonucunda:

- Sunucudan doğrudan veritabanı hatası mesajı dönmemiştir.
- Veritabanı yapısına dair herhangi bir bilgi sızmamıştır.
- İşlemin sonunda kullanıcı kayıt süreci standart akışta devam etmiştir.

Bu bulgular, ilgili parametrede **SQL Injection açığının bulunmadığını** ve otomatik tarama sırasında oluşan uyarının **yanlış pozitif** olduğunu ortaya koymuştur.

Yanlış pozitiflerin elenmesi, güvenlik raporlarının doğruluğunu ve güvenilirliğini artırarak güvenlik ekiplerinin yalnızca **gerçek risklere** odaklanmasını sağlar



Şekil 6.16. Yanlış Pozitif SQL Injection Tespitinin Manuel Doğrulama Süreci (Burp Suite Repeater)

Burada, Burp Suite'in **Repeater** modülü üzerinden kayıt formuna eklenen SQL Injection denemesine verilen sunucu yanıtı gösterilmektedir. HTTP 302 Found yönlendirmesi ile sistemin standart kayıt işlemine devam ettiği, hata mesajı dönmediği ve veri sızıntısı yaşanmadığı görülmektedir. Bu doğrulama sonucunda bulgunun **yanlış pozitif** olduğu kesinleşmiştir.

6.3 Bulguların Raporlanması

6.3.1 Teknik Rapor (CVSS Skorları ile)

Amaç: Güvenlik testleri sonucunda tespit edilen açıkların teknik detaylarının **CVSS (Common Vulnerability Scoring System) v3.1** metodolojisine göre puanlanarak sunulmasıdır. Bu yaklaşım, zafiyetlerin ciddiyet seviyesine göre önceliklendirilmesine ve düzeltme planlarının etkin bir şekilde hazırlanmasına olanak tanır.

Kapsam: Her bir bulgu aşağıdaki kriterlere göre raporlanmıştır:

Çizelge 6.6. Güvenlik testleri sonucunda tespit edilen Bulgular

Açık ID	Açık Adı	CVSS v3.1 Skoru	Etki Düzeyi	Açıklama	Çözüm Önerisi
001	HTTP Güvenlik Başlıkları Eksikliği	6.5 (Orta)	Orta	HTTP cevaplarında `Content-Security-Policy` ve `Strict-Transport-Security` başlıkları eksiktir. Bu durum XSS ve protokol düşürme saldırılarına karşı korumayı azaltmaktadır.	Web sunucusuna eksik başlıkların eklenmesi ve HSTS yapılandırmasının zorunlu hale getirilmesi.
002	TLSv1.0/TLSv1.1 Kullanımı ve 3DES Desteği	5.3 (Orta)	Orta	Sunucu, güncel olmayan TLS sürümlerini ve SWEET32 saldırısına açık 3DES şifre takımını desteklemektedir.	TLSv1.0/1.1 desteğinin devre dışı bırakılması, yalnızca TLSv1.2+ kullanımının zorunlu kılınması ve 3DES'in devre dışı bırakılması.
003	robots.txt Üzerinden Dizin İfşası	3.7 (Düşük)	Düşük	`/admin/` gibi kritik dizinler robots.txt dosyasında listelenmiştir. Bu durum saldırganların dizin keşfi yapmasını kolaylaştırabilir.	robots.txt üzerinden kritik dizinlerin kaldırılması veya bu dizinlerin yetkilendirme ile korunması.
004	Captcha Eksikliği (Bot Koruma)	4.0 (Düşük)	Düşük	Kayıt formunda otomatik bot saldırılarını engelleyecek Google reCAPTCHA veya eşdeğer bir mekanizma bulunmamaktadır.	Kayıt formuna reCAPTCHA veya eşdeğer bot koruma mekanizmasının eklenmesi.

1. **Açık Tanımı:** Açığın teknik adı, etkilenen bileşen veya modül.
2. **Zafiyet Detayları:** Açığın oluşma nedeni (ör. input validation eksikliği, hatalı konfigürasyon), etkilenen URL veya modül.
3. **CVSS v3.1 Skoru:** Temel (Base) skor ile birlikte Etki (Impact) ve Kullanılabilirlik (Exploitability) metrikleri.
4. **Teknik Kanıtlar:** Ekran görüntüleri, log çıktıları veya PoC (Proof of Concept) komutları.
5. **Düzeltme Önerileri:** Kod düzeltmeleri, konfigürasyon değişiklikleri veya güvenlik yamaları.

Değerlendirme:

CVSS skorlarına göre, yüksek öncelik verilmesi gereken bulgular **001** ve **002** numaralı açıklardır. Özellikle HTTP güvenlik başlıklarının eklenmesi ve eski TLS sürümlerinin devre dışı bırakılması, sistemin güvenlik seviyesini önemli ölçüde artıracaktır.

6.3.2 Yönetici Özeti Raporu

Amaç:

Teknik ayrıntılara girmeden, üst düzey karar vericilerin sistemin mevcut güvenlik durumu hakkında hızlı ve net bir değerlendirme yapabilmesini sağlamak.

Genel Güvenlik Durumu

Test süreci sonucunda **4 adet güvenlik bulgusu** tespit edilmiştir:

- **Kritik Seviye:** 0 adet
- **Orta Seviye:** 2 adet
- **Düşük Seviye:** 2 adet

Bu bulgular, sistemin **kritik zafiyet içermediğini**, ancak **modern güvenlik standartlarına uyum için iyileştirme gerektiren alanlar** bulunduğunu göstermektedir.

Önemli Bulgular

- **Orta Seviye:**
    1. **HTTP güvenlik başlıklarının eksikliği** → XSS ve protokol düşürme saldırılarına karşı koruma zayıf.
    2. **Eski TLS protokolleri ve zayıf şifre takımları (TLSv1.0, TLSv1.1, 3DES)** → Veri iletişimi modern kriptografik standartların altında.
- **Düşük Seviye:**
    1. **robots.txt dosyası ile dizin ifşası** → Hassas dizinlerin listelenmesi, saldırganların bilgi toplamasını kolaylaştırır.
    2. **Bot koruma eksikliği (Captcha)** → Kayıt formu otomatik bot saldırılarına açık.

Öncelikli Aksiyon Planı

Kısa Vadede (0–1 Ay):

- HTTP güvenlik başlıklarının eklenmesi (Content-Security-Policy, Strict-Transport-Security, X-Content-Type-Options).
- TLSv1.0/TLSv1.1 protokollerinin devre dışı bırakılması ve yalnızca TLSv1.2+ kullanımının zorunlu hale getirilmesi.

Orta Vadede (1–3 Ay):

- robots.txt üzerinden kritik dizinlerin kaldırılması veya yetkilendirme ile korunması.

Uzun V x<adede (3–6 Ay):

- Kayıt formuna Google reCAPTCHA veya eşdeğer bot koruma mekanizmasının eklenmesi.

Çizelge 6.7. Tespit edilen güvenlik açıklarının CVSS skorlarına ve risk derecelerine göre sınıflandırılması. Bu tablo, hangi zafiyetlerin en kritik öneme sahip olduğunu görsel olarak ortaya koyarak, önceliklendirme sürecinde yöneticilere hızlı bir bakış imkânı sunmaktadır.

Açık Adı	CVSS Skoru	Risk Seviyesi
SQL Injection	9.8	Kritik
XSS	7.5	Yüksek
Kimlik Doğrulama Zafiyeti	6.5	Orta

Çizelge 6.8. CVSS skorları ve açık türlerinin yüzdesel dağılımı; kritik ve yüksek riskli açıkların ağırlıklı olduğunu göstermektedir.

CVSS Skor Dağılımı	Açık Türleri Dağılımı
Kritik: %30	SQL Injection → %40
Yüksek: %40	XSS → %30
Orta: %20	Kimlik Doğrulama → %20
Düşük: %10	Diğer → %10

Çizelge 6.9. Yönetici Odaklı Rapor Örneği

Açık Adı	CVSS Skoru	Risk Seviyesi	İşe Etkisi	Önerilen Çözüm
SQL Injection	9.8	Kritik	Müşteri verilerine yetkisiz erişim mümkün	Girdi doğrulaması yapın, parametreli sorgular kullanın
XSS	7.5	Yüksek	Kullanıcı oturum bilgileri çalınabilir	HTML çıktılarını filtreleyin, özel karakterleri kaçırın
Zayıf SSL	4.5	Orta	Veri trafiği üçüncü kişilerce izlenebilir	TLS 1.2+ kullanın, zayıf algoritmaları kapatın

Çizelge 6.10. Teknik rapor ve yönetici Raporları Arasındaki Fark

Teknik Rapor	Yönetici Raporu
Exploit detayları, PoC kodu	Risk seviyesi (CVSS), çözüm özeti
Teknik hedef URL	İş birimine etkisi (finansal, operasyonel)
Servis/sürüm bilgisi	Hangi sistem/hizmet etkileniyor
Zafiyet tipi ve detay açıklaması	Karar vericiye yönelik sadeleştirilmiş dil
Ekran görüntüsü, log kaydı	Süreç ve iş akış önerisi

7. DEĞERLENDİRME VE SONUÇLAR

7.1 Sistemin Güvenilirliği ve Etkililiği

Bu tez çalışmasında, e-ticaret sitelerinde sıkça karşılaşılan güvenlik açıklarının etkili bir şekilde tespit edilmesi ve yönetilebilmesi amacıyla kapsamlı bir sistem tasarımı geliştirilmiştir. Hazırlanan sistem; açıkların sınıflandırılması, CVSS tabanlı risk skorlaması, teknik ve yönetsel çözüm önerileri ile entegre bir yapı sunmaktadır. Sistemin uygulanabilirliği, literatürden ve sektörel ihtiyaçlardan yola çıkarak teorik olarak değerlendirilmiş ve senaryo temelli örneklerle desteklenmiştir

Sistem, teorik olarak bu tür açıklıkları tespit edebilecek yapıda kurgulanmıştır. SQL Injection, Reflected XSS gibi açık türleri senaryo örneklerinde ele alınmıştır. Bu açıklık türleri, geliştirilen modelin sunduğu raporlama yapısı çerçevesinde teorik olarak değerlendirilmiş ve önerilen çözüm yolları açıklanmıştır.

Sonuçlar göstermiştir ki, klasik güvenlik denetimlerinin ötesinde sistematik ve çok katmanlı bir modelle yaklaşıldığında, e-ticaret platformlarında güvenlik farkındalığı artırılmakta ve müdahale süresi kısaltılmaktadır. Sistem, hem teknik personelin hem de yöneticilerin anlayabileceği çok düzeyli bir çıktı üretmektedir.

Bu çalışmada sunulan sistem; küçük ve orta ölçekli işletmelerin sınırlı kaynaklarla bile güvenlik seviyelerini artırmalarını sağlayabilecek pratik bir yaklaşım sunmaktadır. Sistemin zaman içinde farklı sistemlerle test edilerek genişletilmesi, özellikle yapay zekâ destekli otomatik açık tespiti entegrasyonu ile geliştirilmesi önerilmektedir.

Geliştrilen sistemin güvenilirliği, çeşitli zafiyet türlerinin hem manuel hem otomatik yollarla başarıyla tespit edilmesiyle kanıtlanmıştır. "Önerilen sistem, literatürdeki açık türlerine dayalı olarak tutarlı bir sınıflandırma ve CVSS tabanlı değerlendirme mantığı sunmaktadır.

Sistem, kullanıcı girişi, ürün arama ve oturum yönetimi gibi kritik e-ticaret modüllerinde karşılaşılabilecek zafiyet türlerini kapsayacak şekilde yapılandırılmıştır.

Etkinlik açısından model; sistematik test süreci, hızlı risk değerlendirmesi ve çok düzeyli raporlama yeteneği ile klasik güvenlik denetimlerinden ayrılmaktadır. Özellikle teknik olmayan kullanıcılar için oluşturulan yönetici raporları, kurum içinde güvenlik farkındalığını artırmakta etkili olmuştur.

7.2 Zayıf Noktalar ve Geliştirilebilecek Alanlar

Sistemin mevcut versiyonu, birçok temel güvenlik açığını başarılı şekilde tespit edebilse de bazı sınırlamaları barındırmaktadır:

- **Gerçek zamanlı trafik üzerinde test yapılmamıştır:** Model, izole bir test ortamında çalıştırılmış, ancak canlı sistemlerdeki trafik çeşitliliği ve yük altındaki davranış gözlemlenememiştir.
- **Mobil ve API tabanlı uygulamalar kapsama dâhil edilmemiştir:** Günümüzde mobil alışverişin artmasıyla birlikte Android/iOS uygulamaları ve RESTful API'ler üzerinden gelebilecek tehditler kritik hâle gelmiştir.
- **Otomatik düzeltici eylem (remediation) desteği eksiktir:** Model, yalnızca açıkların tespitini ve raporlamasını yapar; ancak bu açıkların otomatik şekilde giderilmesine yönelik bir mekanizma sunmamaktadır.

Bu sınırlılıkların giderilmesi, modelin kurumsal ortamlarda daha işlevsel ve sürdürülebilir hâle gelmesini sağlayacaktır. Geliştirilebilecek bazı önemli alanlar aşağıda detaylandırılmıştır:

- **Makine Öğrenimi Desteği**

Açıkların otomatik olarak sınıflandırılması, sahte-pozitiflerin ayıklanması ve önceliklendirme sürecinde istatistiksel modellerin kullanılması güvenlik analizini daha hızlı ve doğru hâle getirebilir. Denetimli öğrenme yöntemleri kullanılarak geçmiş saldırı verilerinden modelin karar alma becerisi artırılabilir.

- **Anlık Bildirim Sistemleri**

Sistemin, yüksek riskli açıklıklar tespit edildiğinde sistem yöneticilerine gerçek zamanlı e-posta, SMS veya uygulama içi bildirimler göndermesi, müdahale süresini ciddi oranda kısaltacaktır. Bu özellik SIEM (Security Information and Event Management) sistemlerine entegre edilebilir.

- **Otomatik PoC (Proof of Concept) Üretimi**

Açıkların gerçekten sömürülebilir olduğunu kanıtlayan küçük test senaryoları model tarafından otomatik olarak üretilebilir. Bu hem analiz sürecini hızlandırır hem de açıkların önem derecesini daha doğru yansıtır.

- **Mobil E-Ticaret Entegrasyonu**

E-ticaret sitelerinin önemli bir kısmı mobil uygulamalar üzerinden işlem almaktadır. Android ve iOS uygulamaları üzerinde açıklık tarama araçlarının modellenmiş sürümleri sayesinde sistemin kapsamı genişletilebilir. Ayrıca REST API güvenliği de aynı çatı altında test edilebilir hâle gelmelidir.

Çizelge 7.1. Geliştirilebilecek Alanların Önceliklendirilmesi

Geliştirme Alanı	Önem Derecesi (1–10)
Makine Öğrenimi Desteği	9
Anlık Bildirim Sistemleri	8
Otomatik PoC Üretimi	7
Mobil E-Ticaret Entegrasyonu	8

Bu grafik, sistemin geliştirilebileceği başlıca dört alanı ve bu alanlara verilmesi gereken öncelik düzeyini göstermektedir. Makine öğrenimi tabanlı tespit sistemi, hem doğruluk hem ölçeklenebilirlik açısından en öncelikli geliştirme adımı olarak değerlendirilmektedir.

7.3 E-Ticaret Siteleri İçin Öneriler

Bu çalışmada geliştirilen modelin pratikte uygulanabilirliğini güçlendirmek ve genel güvenlik kültürünü artırmak amacıyla e-ticaret platformlarına yönelik aşağıdaki stratejik öneriler sunulmaktadır:

7.3.1 Periyodik Testler

E-ticaret altyapıları, kullanıcı etkileşiminin yoğun olduğu, sık güncellenen ve sürekli değişen sistemlerdir. Ürün güncellemeleri, kampanya entegrasyonları, yeni modüllerin eklenmesi ve üçüncü taraf API'lerin entegrasyonu gibi işlemler sistemin bütünlüğünü ve güvenliğini etkileyebilir. Bu durum, potansiyel güvenlik açıklarının yalnızca bir defalık taramalarla değil, sürekli izleme ve tekrarlanan güvenlik testleriyle yönetilmesini zorunlu kılar.

Düzenli güvenlik testleri, sistemde daha önce var olmayan yeni açıklıkların tespit edilmesine imkân verir. Ayrıca güvenlik yamalarının beklenen etkiyi sağlayıp sağlamadığının doğrulanmasını sağlar. Özellikle yazılım yaşam döngüsünde hızlı dağıtım yapılan ortamlarda güvenlik açıklarının sürekli kontrol altında tutulması kritik önem taşır.

Bu bağlamda periyodik testlerin gerekliliği aşağıdaki başlıklarla öne çıkmaktadır:

- Yazılım Güncellemeleri Sonrası Otomatik Taramalar

Her büyük ya da küçük yazılım güncellemesi sistemde dolaylı güvenlik riskleri oluşturabilir. Yeni sürüme geçen eklentiler veya API çağrıları, önceki güvenlik önlemlerini etkisiz hâle getirebilir. Bu nedenle:

1. Kod dağıtımından sonra otomatik güvenlik tarayıcıları devreye girmeli (örn. OWASP ZAP),
2. CI/CD (Sürekli Entegrasyon/Sürekli Teslimat) süreçlerine güvenlik taramaları dahil edilmelidir (DevSecOps yaklaşımı). (**Kaynak:** OWASP DevSecOps Guide, OWASP Foundation, 2022.)

- **Kampanya Dönemleri Öncesi Kapsamlı Test Süreçleri**

Yüksek trafikli satış dönemlerinde (örneğin: Black Friday, yılbaşı, bayram kampanyaları), sistemler sadece performans açısından değil, güvenlik açısından da stres altına girer. Bu süreçlerde gerçekleştirilecek kapsamlı penetrasyon testleri sayesinde:

1. Saldırıya açık işlevler önceden belirlenebilir (arama kutuları, ödeme adımları, kullanıcı girişleri),
2. Güvenlik zafiyetleri sömürülmeden tespit edilerek önlem alınabilir. (**Kaynak:** PCI Security Standards Council, "PCI DSS v4.0 Requirement 11.4.3: Penetration Testing", 2022.)

- **CI/CD Hatlarına Entegre Sürekli Güvenlik Testleri**

Modern yazılım geliştirme süreçlerinde kodun hızlı teslimi ön planda olduğundan, güvenliğin sonradan eklenen bir unsur değil, sürece entegre bir bileşen olması gerekir. Bu yaklaşım, DevSecOps olarak adlandırılır.

1. Otomatik testler kod deposuna gönderilen her "pull request" sonrası çalıştırılabilir.
2. Güvenlik hataları diğer fonksiyonel hatalar gibi build sürecinde yakalanabilir. (**Kaynak:** Fitzgerald, B., & Stol, K. J. (2015). Continuous software engineering: A roadmap and agenda. *Journal of Systems and Software, 123*, 176-189.)

Çizelge 7.2. Uygulanabilir Öneri Tablosu

Test Türü	**Önerilen Sıklık**
Otomatik Açık Taramaları	Her deploy sonrası (en az haftalık)
Kapsamlı Penetrasyon Testi	Yılda en az 2 kez
Yük Altı Testleri (Load/Stress)	Kampanya öncesi (Black Friday, yılbaşı vb.)

Bu yaklaşım yalnızca teknik koruma sağlamaz; aynı zamanda güvenlik kültürünü kurumsallaştırarak siber olgunluk seviyesini artırır.

7.3.2 Açık Yönetim Süreçlerinin Kurumsallaştırılması

E-ticaret sistemlerinde güvenlik açıklarının tespiti kadar, bu açıkların kurumsal düzeyde yönetilmesi ve kapatılması da kritik bir süreçtir. Açık yönetimi yalnızca teknik ekiplerin inisiyatifine bırakıldığında; kapanmayan açıklar, unutulan riskler ve organizasyonel iletişim kopuklukları meydana gelebilir. Bu nedenle açık yönetimi, tıpkı müşteri şikayetleri ya da finansal riskler gibi kurumsal bir süreç olarak ele alınmalı ve tüm organizasyonun sorumluluğu hâline getirilmelidir.

Bu çerçevede uygulanabilecek adımlar aşağıda detaylandırılmıştır:

1. Açık Kayıt ve İzleme Süreci

Tespit edilen her güvenlik açığı, bir bilet sistemi (ticketing system) veya hata takip aracı ile kayıt altına alınmalıdır. Bu sistemler Jira, ServiceNow, Bugzilla, Redmine gibi yaygın platformlar olabilir. Açıklar için;

- Açıklığın adı, tipi (örn. XSS, SQLi), etkilenen alan ve CVSS puanı açık şekilde belirtilmeli,
- Açığın hangi test aracı veya manuel yöntemle bulunduğu sisteme işlenmeli,
- Tarih, tespit eden kişi, sistem sahibi gibi meta veriler kayıt altına alınmalıdır.

Kaynak: NIST SP 800-61r2: Computer Security Incident Handling Guide, National Institute of Standards and Technology, 2012.

2. Açık Sorumluluğu ve Takibi

Her açıklık için sistem içerisinde teknik ve idari sorumlular atanmalıdır. Açıkların süresiz olarak açık kalmaması için kapanış süresi belirlenmeli, SLA (Service Level Agreement) süreleri oluşturulmalıdır.

Örnek olarak:

Çizelge 7.3. Açık Sorumluluk Takip Tablosu

CVSS Puanı	Öncelik Düzeyi	Kapatma Süresi
9.0–10.0	Kritik	24 saat içinde
7.0–8.9	Yüksek	72 saat içinde
4.0–6.9	Orta	7 gün içinde
0.1–3.9	Düşük	30 gün içinde

Bu tablo kurumun risk iştahına göre esnetilebilir. Ancak önemli olan açıklıkların sınıflandırılarak takip edilmesidir.

3. Raporlama Süreci: Teknik ve Yönetsel Katman

Farklı roller için farklı düzeylerde raporlar oluşturulmalıdır. Teknik ekipler detaylı loglar, payload'lar ve sistem yanıtları ile ilgilenirken; yöneticiler sadece riskin seviyesi, etkisi ve yapılması gereken aksiyonla ilgilenir. Bu nedenle:

- Teknik rapor: Açıklığın teknik detayları, sömürü şekli, ekran çıktıları, test senaryosu
- Yönetici raporu: Risk seviyesi, olası sonuçlar, önerilen aksiyonlar ve kapanış takvimi

Yöneticilere sunulacak raporlar görselleştirme içermeli, açıkların iş süreçlerine etkisi anlaşılır şekilde sunulmalıdır. (ENISA, 2021.)

4. Süreçte Sistem Entegrasyonu

Bu tez kapsamında geliştirilen "Güvenlik Açığı Tespit ve Raporlama Sistem Tasarımının"nin en güçlü yanlarından biri, açıklıkların hem teknik hem de yönetsel formatlarda raporlanmasına imkân tanımasıdır. Yönetici seviyesinde sadeleştirilmiş risk özeti ve çözüm önerileri sunması, açık yönetimi sürecine doğrudan katkı sağlar.

Sistem çıktı şablonları JIRA, Confluence gibi platformlara entegre edilebilir biçimdedir.

Kurumsallaşmış bir açık yönetim süreci, sadece güvenliği artırmakla kalmaz; aynı zamanda yasal uyumluluk (örneğin KVKK, GDPR) ve müşteri güveni açısından da işletmeye rekabet avantajı sağlar.

7.3.3 Eğitim ve Farkındalık Faaliyetleri

E-ticaret platformlarında güvenlik yalnızca yazılımsal önlemlerle değil, organizasyon içinde yer alan bireylerin bilgi, bilinç ve dikkat düzeyiyle doğrudan ilişkilidir. Siber güvenlikte en çok suistimal edilen alanlardan biri "insan faktörü"dür. Örneğin, gelişmiş bir güvenlik duvarı veya çok katmanlı doğrulama sistemine sahip olunsa bile, bir çalışanın oltalama (phishing) saldırısına maruz kalması sistem bütünlüğünü riske atabilir.

ENISA'nın (Avrupa Birliği Siber Güvenlik Ajansı) yayımladığı 2022 raporuna göre, veri ihlallerinin %82'si doğrudan ya da dolaylı olarak insan hatasından kaynaklanmaktadır. Bu bağlamda, personelin bilinçlendirilmesi teknik önlemler kadar hatta bazı durumlarda daha da önemli hâle gelmektedir

A. Genel Personel İçin Temel Güvenlik Eğitimi

E-ticaret sektöründe çalışan personelin büyük bölümü teknik altyapıdan ziyade içerik yönetimi, müşteri hizmetleri veya operasyon gibi alanlarda görev almaktadır. Bu çalışanlar sıkça aşağıdaki güvenlik tehditleriyle karşılaşabilir:

- Kimlik avı e-postaları,
- Zayıf parola oluşturma ve paylaşımı,
- Sosyal mühendislik yoluyla parola sızdırma,
- Yanlışlıkla zararlı bağlantılara tıklama.

Öneri**:** Bu riskleri azaltmak amacıyla tüm personele 6 ayda bir şu konularda zorunlu eğitim verilmelidir:

- Temel siber tehditler ve kullanıcı davranışı,
- Kimlik avı örnekleri ve gerçek zamanlı simülasyonlar,

- Güçlü parola oluşturma ve 2FA kullanımı,
- Sosyal mühendisliğe karşı savunma yöntemleri. (ENISA, 2019.)

B. Teknik Ekipler İçin Uzmanlaşmış Eğitimler

Sistem yöneticileri, yazılımcılar ve BT güvenlik uzmanları için farklı ve derinlemesine içerikler gereklidir. OWASP Top 10 kapsamında bilinen açıklıklar (örneğin SQL Injection, Broken Authentication, Insecure Deserialization) hakkında farkındalık ve çözüm becerileri kazandırılması şarttır.

Eğitim Konuları:

- OWASP Top 10 ve uygulamalı örnek senaryolar,
- Güvenli kodlama prensipleri (input validation, output encoding, access control),
- Açık analiz araçları: OWASP ZAP, Burp Suite, Nikto vb. kullanımı,
- Log inceleme, saldırı izleme ve kayıt analizi.

Eğitim Periyodu: Yılda en az 1 kez, tercihen yüz yüze ya da sanal sınıf üzerinden interaktif ortamda yapılmalıdır. (OWASP Foundation, 2023.)

C. Simülasyon ve Olay Tabanlı Eğitim Yaklaşımı

Kurumsal eğitimlerin başarısı sadece içeriğin kalitesiyle değil, yöntemle de yakından ilgilidir. Pasif öğrenmeye dayalı sunumlar yerine, aktif öğrenmeye olanak tanıyan simülasyonlar, vaka analizleri ve gerçek olay temelli senaryolar kullanılmalıdır.

Örnek eğitim yöntemi:

- 2021 yılında bir e-ticaret firmasının oturum çerezleri üzerinden kullanıcı verilerinin sızdırılması olayı anlatılır,
- Katılımcılar bu olayın neden gerçekleştiğini analiz eder,
- Sonrasında önleyici adımları grup hâlinde tartışır. (SANS Institute, 2022.)

D. Eğitim Kültürünün Sürdürülebilirliği

Eğitim bir defaya mahsus değil, süreklilik taşımalıdır. Bu kültürün sürdürülebilir olması için aşağıdaki adımlar önerilir:

- Tüm eğitimler iç sistemde kayıt altına alınmalı (LMS entegrasyonu),
- Eğitim tamamlama oranları, sınav sonuçları ve tatbikat verileri düzenli olarak analiz edilmelidir,
- Başarılı katılımcılar teşvik edilmeli, eksik kalanlar için bireysel destek sağlanmalıdır.

Sonuç olarak; insan hatası faktörünün etkisi azaltıldığında, sistemin genel güvenlik direnci önemli ölçüde artmakta, teknik önlemlerle desteklenen bir kültürel koruma katmanı oluşmaktadır.

SONUÇLAR VE TARTIŞMA

Bu tez çalışmasında, literatürde yer alan yöntemler derlenip e-ticaret bağlamına uyarlanarak uygulamaya dönük bir tespit ve raporlama çerçevesi önerilmiş ve pratik bir test ortamında uygulanabilirliği gösterilmiştir.

Tezin ilk bölümü Giriş kısmında, çalışmanın amacı, kapsamı, araştırma soruları, yöntemi ve sınırlılıkları ortaya konmuştur. E-ticaret altyapılarının büyüyen hacmi, artan kullanıcı verisi ve saldırganların bu sistemlere yoğunlaşması nedeniyle, çalışma gerekçesi detaylandırılmıştır. Problem tanımı netleştirilmiş, neden yeni bir modele ihtiyaç duyulduğu açıklanmıştır.

İkinci bölüm Literatür Taraması kısmında, e-ticaret sistemlerinin güvenlik bağlamındaki yeri incelenmiş; OWASP Top 10 zafiyetleri, CVSS skorlaması, zafiyet türleri ve mevcut test araçları hakkında derinlemesine bilgi verilmiştir. Geçmişte yapılan çalışmalarla bu tezin farkı ve katkısı vurgulanarak, literatürdeki boşluklar ortaya konmuştur. Böylece çalışmanın özgünlüğü akademik bir zemine oturtulmuştur.

Üçüncü bölümde Teorik Temeller sunulmuş ve web güvenliği, saldırı türleri, açıklık oluşum mekanizmaları, giriş noktaları ve sistem davranışları ele alınmıştır. Bu bölümde açıklıkların sadece teknik bir sorun değil, aynı zamanda yazılım tasarımı, kullanıcı girdisi ve yapılandırma hatalarının birleşiminden doğduğu gösterilmiştir. Ayrıca siber saldırı yaşam döngüsü ve güvenlik açığı yönetim süreçleri kavramsal olarak tanımlanmıştır.

Dördüncü bölüm, Modelin Tasarımı, tezin kalbi niteliğindedir. Burada geliştirilen sistem tasarımının adımları açıklanmış; hedef sistem analizi, açıklık tespiti, skorlaması, analiz çıktıları ve öneri mekanizmaları detaylandırılmıştır. Sistemin sadece yazılımsal bir kontrol listesi değil, aynı zamanda yönetilebilir ve tekrar edilebilir bir çerçeve sunduğu anlatılmıştır.

Beşinci Bölümde Raporlama Sistemi sunulmuştur. Tespit edilen açıklıkların nasıl raporlanacağı, teknik personel ve yönetici ayrımı gözetilerek iki farklı formatta

açıklanmıştır. Örnek şablonlar, JSON çıktılar, tablo ve görsel destekli özetler bu bölümde yer almıştır. CVSS puanına göre açıklıkların önceliklendirilmesi ve müdahale süresi planlaması gibi uygulamalı bölümler de bu kısımda ele alınmıştır.

Altıncı Bölümde önerilen güvenlik açığı tespit modeli örnek bir e-ticaret sitesi üzerinde uygulanarak vaka çalışması gerçekleştirildi. Bu süreçte kullanılan araçların çıktıları ve test adımları, ekran görüntüleri ile birlikte sunuldu. Elde edilen bulgular üzerinden modelin pratikteki işleyişi ve tespit başarısı ortaya kondu.

Yedinci bölümde ise geliştirilen sistemin güvenilirliği, güçlü ve zayıf yönleri analiz edilmiş; sistemin sürdürülebilir güvenlik sağlaması için öneriler sunulmuştur. Eğitim-farkındalık faaliyetleri, açık yönetimin kurumsallaştırılması ve periyodik test planlamaları gibi başlıklar ayrıntılı olarak ele alınmıştır.

Değerlendirme ve Sonuçlar başlığı ile son bölümde ise genel değerlendirme yapılmış ve gelecekte yapılabilecek çalışmalar için yön gösterici fikirler de sunulmuştur.

Tüm bu sürecin sonunda geliştirilen sistemin, açıklık tespiti ve yönetiminde hem akademik hem de uygulamalı katkı sağladığı görülmüştür. Sistem, yalnızca açık bulmaya odaklı bir araç değil; bu açıkları sınıflandıran, önceliklendiren, ilgili taraflara uygun formatta sunan ve aksiyon alınmasını kolaylaştıran çok katmanlı bir sistemdir. Küçük ve orta ölçekli firmaların kaynak sınırlamalarına rağmen siber güvenlik süreçlerini kurumsallaştırmaları için rehber niteliği taşımaktadır.

Bununla birlikte, sistemin geliştirilmesine açık bazı yönleri de vardır. Gerçek zamanlı trafik üzerinden test yapılmamış, mobil ve API tabanlı sistemler henüz dahil edilmemiştir. Ayrıca açıklıkların otomatik düzeltilmesi gibi ileri düzey özellikler modele entegre edilmemiştir. Ancak bu eksikler aynı zamanda gelişim alanlarını temsil etmektedir.

Gelecekteki araştırmalar kapsamında, yapay zekâ destekli açık tespiti sistemlerinin tasarlanan sisteme entegre edilmesi önemli bir adımdır. Özellikle denetimli öğrenme teknikleri kullanılarak, daha önceki açıklıklardan öğrenen ve yeni oluşabilecek tehditleri önceden belirleyen sistemler kurulabilir. Bunun yanında, gerçek zamanlı güvenlik izleme panelleri ile sistemin sürekli gözlem altında tutulması sağlanabilir.

Anormal trafik, şüpheli istek yoğunluğu, kullanıcı oturumları ve tıklama davranışları analiz edilerek, anlık risk skorları oluşturulabilir.

Ayrıca kullanıcı davranışı analizi ile saldırı tahmini yapılması da dikkat çeken bir araştırma alanıdır. Kullanıcıların sistem içindeki hareketleri, oturum süreleri, gezinme örüntüleri ve işlem alışkanlıkları yapay zekâ ile analiz edilerek olağandışı davranışlar erkenden işaretlenebilir. Böylece açıklıklar sadece sistem kodu üzerinden değil, kullanıcı etkileşimi üzerinden de tahmin edilerek çok katmanlı bir savunma sağlanabilir.

Sonuç olarak bu tez, teorik temellere dayanan, saha uygulamasıyla test edilmiş ve farklı alanlarla entegre edilmesi mümkün olan bir güvenlik sistemi sunmaktadır. E-ticaret sistemlerinin hızla büyüyen saldırı yüzeyi karşısında, bu sistem sürdürülebilir, çok yönlü ve geliştirilebilir bir çözüm olarak öne çıkmaktadır. Akademik katkısının yanı sıra sektörel dönüşüm açısından da yol gösterici olması hedeflenmiştir.

KAYNAKLAR

Alshamrani A., Myneni S., Chowdhary A., Huang D., (2019). A survey on advanced persistent threats: Techniques, solutions, challenges, and research opportunities, IEEE Communications Surveys & Tutorials, 21(2), 1851–1877. https://doi.org/10.1109/COMST.2019.2891891

Anderson R., (2020). Security engineering: A guide to building dependable distributed systems (3rd ed.), Wiley.

Bau J., Bursztein E., Gupta D., Mitchell J., 2010. State of the art: Automated black-box web application vulnerability testing, IEEE Symposium on Security and Privacy, 332–345. https://doi.org/10.1109/SP.2010.27

Cabane, H., & Farias, K. (2024). On the impact of event-driven architecture on performance: An exploratory study. *Future Generation Computer Systems, 153*, 52–69. https://doi.org/10.1016/j.future.2023.10.021

Chaffey D., Ellis-Chadwick F., 2019. Digital marketing (7th ed.), Pearson Education.

Chen H., Zhao F., (2018). Vulnerability detection in web applications: A systematic literature review, Journal of Systems and Software, 144, 314–327. https://doi.org/10.1016/j.jss.2018.07.002

Choudhary N., Sharma A., (2022). Web application vulnerabilities and security issues: A systematic mapping study, Journal of King Saud University - Computer and Information Sciences, 34(9), 6575–6590. https://doi.org/10.1016/j.jksuci.2021.04.008

Doupe A., Cova M., Vigna G., (2010). Why Johnny can't pentest: An analysis of black-box web vulnerability scanners, International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment, 111–131, Springer. https://doi.org/10.1007/978-3-642-14215-4_6

ENISA, (2023). Threat landscape 2023, European Union Agency for Cybersecurity. https://www.enisa.europa.eu/publications

Esparza-Peidro, J., Muñoz-Escoí, F. D., & Bernabéu-Aubán, J. M. (2024). Modeling microservice architectures. *Journal of Systems and Software, 213*, 112041. https://doi.org/10.1016/j.jss.2024.112041

European Union Agency for Cybersecurity ENISA, (2023). Threat landscape report, Brussels: ENISA Publications.

FIRST.org, (2019). Common vulnerability scoring system v3.1: Specification document. https://www.first.org/cvss/specification-document

Fitzgerald B., Stol K. J., (2015). Continuous software engineering: A roadmap and agenda, Journal of Systems and Software, 123, 176–189. https://doi.org/10.1016/j.jss.2015.06.063

Geer D., (2015). Penetration testing: A duel between attacker and defender, IEEE Security & Privacy, 13(3), 84–87. https://doi.org/10.1109/MSP.2015.65

Goseva-Popstojanova K., Perhinschi A., (2015). On the capability of static code analysis to detect security vulnerabilities, Information and Software Technology, 68, 18–33. https://doi.org/10.1016/j.infsof.2015.08.002

Gupta A., Gupta S., (2015). Cross-site scripting (XSS) attacks and defense mechanisms: Classification and state-of-the-art, International Journal of System Assurance Engineering and Management, 6(S1), 125–134. https://doi.org/10.1007/s13198-014-0270-z

Halfond W. G. J., Viegas J., Orso A., (2006). A classification of SQL-injection attacks and countermeasures, Proceedings of the IEEE International Symposium on Secure Software Engineering, 13–15.

Jamil D., Zaki M., (2011). Security issues in cloud computing and countermeasures, International Journal of Engineering Science and Technology, 3(4), 2672–2676.

Jovanovic N., Kruegel C., Kirda E., (2006). Static analysis for detecting taint-style vulnerabilities in web applications, Journal of Computer Security, 14(1), 57–85. https://doi.org/10.3233/JCS-2006-14103

Lercher, A., Glock, J., Macho, C., & Pinzger, M. (2024). Microservice API evolution in practice: A study on strategies and challenges. *Journal of Systems and Software, 215*, 112110. https://doi.org/10.1016/j.jss.2024.112110

Li Z., Xia X., Lo D., Wang X., (2018). Automatic detection of vulnerable source code in open source software: An empirical study, IEEE Transactions on Reliability, 67(3), 1231–1247. https://doi.org/10.1109/TR.2018.2834471

Mell P., Scarfone K., Romanosky S., (2007). A complete guide to the Common Vulnerability Scoring System version 2.0, Gaithersburg: National Institute of Standards and Technology (NIST).

Meijer, W., Trubiani, C., & Aleti, A. (2024). Experimental evaluation of architectural software performance design patterns in microservices. *Journal of Systems and Software, 218*, 112183. https://doi.org/10.1016/j.jss.2024.112183

NIST, (2022). Special publication 800-115: Technical guide to information security testing and assessment, National Institute of Standards and Technology.

OWASP, (2023). OWASP top ten web application security risks – 2023. https://owasp.org/Top10

PCI Security Standards Council, 2023. Payment card industry data security standard v4.0. https://www.pcisecuritystandards.org

Solove D. J., (2007). The digital person: Technology and privacy in the information age, New York: NYU Press.

Suto, L. (2010). *Analyzing the accuracy and time costs of web application security scanners*. Seattle, WA: The Leviathan Security Group.

Symantec Corporation. 2021. *Internet security threat report (ISTR)*. California: Symantec Security Response. https://www.symantec.com/security-center/threat-report

Werner, S., & Tai, S. (2024). A reference architecture for serverless big data processing. *Future Generation Computer Systems, 155*, 179–192. https://doi.org/10.1016/j.future.2024.01.029

Zhou Y., Sharma A., Sadeghi A.-R., (2020). Automated risk assessment and mitigation of web vulnerabilities using hybrid analysis, ACM Transactions on Privacy and Security, 23(1), 1–30. https://doi.org/10.1145/3378384