
# Cyber Advantage Uygulamamıza Modül Geliştirilme

Yazılım Mühendisliği Ana Bilim Dalı

Yüksek Lisans / Bitirme Projesi

Berke Kırşallıoba

Tez Danışmanı: Doç. Dr. Aytuğ Onan

Haziran 2023

# Yazarlık Beyanı

Ben, **Berke Kırşallıoba**, başlığı Cyber Advantage Uygulamamıza Modül Geliştirilme olan bu tezimin ve tezin içinde sunulan bilgilerin şahsıma ait olduğunu beyan ederim. Ayrıca:

- Bu çalışmanın bütünü veya esası bu üniversitede Yüksek Lisans / Doktora derecesi elde etmek üzere çalıştığım süre içinde gerçekleştirilmiştir.
- Daha önce bu tezin herhangi bir kısmı başka bir derece veya yeterlik almak üzere bu üniversiteye veya başka bir kuruma sunulduysa bu açık biçimde ifade edilmiştir.
- Başkalarının yayımlanmış çalışmalarına başvurduğum durumlarda bu çalışmalara açık biçimde atıfta bulundum.
- Başkalarının çalışmalarından alıntıladığımda kaynağı her zaman belirttim. Tezin bu alıntılar dışında kalan kısmı tümüyle benim kendi çalışmamdır.
- Kayda değer yardım aldığım bütün kaynaklara teşekkür ettim.
- Tezde başkalarıyla birlikte gerçekleştirilen çalışmalar varsa onların katkısını ve kendi yaptıklarımı tam olarak açıkladım.

Tarih: 16.06.2023

# Cyber Advantage Uygulamamıza Modül Geliştirilme

# Öz

Bu tez, bilgi sistemlerinin güvenliğini sağlamak için önemli bir adım olan Security Testing'in kapsamlı bir incelemesini sunmaktadır. Tezin amacı, güvenlik testinin önemini vurgulamak, yöntemlerini ve tekniklerini tanıtmak, sürecini ayrıntılı olarak açıklamak ve güvenlik açıklarının tespit edilmesi ve düzeltilmesi için kullanılan çeşitli araçları incelemektir. Tez, güvenlik testinin organizasyonlar için neden önemli olduğunu ve güvenlik açıklarının tespit edilmesi ve düzeltilmesi sürecinde nasıl yardımcı olduğunu açıklamaktadır.

Tezin kapsamı, güvenlik testinin temel prensipleri ve yöntemleri, pasif ve aktif güvenlik testleri, beyaz şapka, siyah şapka ve gri şapka yaklaşımları, manuel ve otomatik güvenlik testleri, bilgi toplama ve saldırı simülasyonu yöntemleri, zafiyet tarama ve penetrasyon testi araçları, istismar araçları, kod analizi araçları ve çeşitli güvenlik testi tekniklerini kapsamaktadır.

Tezin önemi, organizasyonların bilgi sistemlerini ve uygulamalarını güvenli tutmak için güvenlik testinin kritik bir bileşeni olduğunu vurgulamaktadır. Güvenlik testi, potansiyel güvenlik açıklarını tespit ederek organizasyonların risklerini azaltmalarına, veri güvenliğini ve gizliliğini sağlamalarına, saldırıları önlemelerine ve marka ile müşteri güvenini korumalarına yardımcı olur.

Tezin yapısal düzeni, giriş bölümü, literatür taraması, metodoloji, bulguların analizi, sonuçlar ve öneriler bölümlerini içerir. Her bir bölüm, güvenlik testi konusunda kapsamlı bir anlayış sağlamak için ilgili konuları ele almaktadır.

Bu tez, güvenlik testinin organizasyonlar için kritik bir süreç olduğunu ve güvenlik açıklarının tespit edilmesi ve düzeltilmesi için çeşitli yöntemlerin ve araçların kullanıldığını vurgulamaktadır. Tezin sonucunda, güvenlik testinin önemi ve sürekli iyileştirme çabalarının organizasyonların güvenlik durumunu güçlendirmesine nasıl katkı sağladığı özetlenmektedir.

# İçindekiler

# Bölüm 1

# Giriş

Bu araştırmanın amacı, Security Testing'in bir uygulamanın güvenlik açıklarını belirlemek, çeşitli saldırı vektörlerine karşı savunmasını değerlendirmek ve güvenlik düzeyini artırmak için nasıl kullanılabileceğini incelemektir. Ayrıca, Security Testing'in önemini vurgulayarak, organizasyonların güvenlik risklerini azaltmalarına ve hassas verileri korumalarına yardımcı olmayı hedeflemektedir. Bu araştırma, güvenlik testinin temel kavramları, yöntemleri, süreci, araçları ve en iyi uygulamaları hakkında ayrıntılı bir anlayış sağlayarak, bilgi teknolojileri alanında güvenliği önemseyen araştırmacılara, profesyonellere ve karar vericilere rehberlik etmeyi amaçlamaktadır.

Bu araştırmanın kapsamı, Security Testing'in temel kavramlarını, yöntemlerini, sürecini, araçlarını ve en iyi uygulamalarını kapsamaktadır. Ayrıca, farklı Security Testing türleri ve teknikleri de incelenmektedir. Araştırma, bilgi teknolojileri alanında güvenlik testi konusuna genel bir bakış sunmayı hedeflemektedir.

Ancak, bu araştırmanın bazı sınırlamaları da bulunmaktadır. Bunlar şunları içerebilir:

**Derinlemesine Teknik Ayrıntılar**: Araştırma, Security Testing'in temel kavramları ve yöntemleri üzerinde yoğunlaşsa da, belirli tekniklerin veya araçların ayrıntılı teknik detaylarına derinlemesine girmemektedir. Bu konuda daha fazla ayrıntı arayan okuyucular için ek araştırma yapılması gerekebilir.

**Uygulama Bağımlılığı**: Araştırma, Security Testing'in genel prensipleri ve yaklaşımları üzerinde durmaktadır. Ancak, farklı uygulama türleri veya endüstrilerdeki spesifik Security Testing gereksinimlerine derinlemesine odaklanmamaktadır. Bu nedenle, belirli bir uygulama veya endüstri için özelleştirilmiş yönergeler veya stratejiler arayan okuyucular için ek çalışmalar yapılması gerekebilir.

**Sınırlı Güncel Bilgi**: Bu araştırma, 2021 yılı itibarıyla mevcut olan bilgileri temel alır. Bilgi teknolojileri alanındaki güvenlik testi yöntemleri ve araçları sürekli olarak gelişmektedir. Dolayısıyla, daha güncel bilgilere veya yeniliklere ulaşmak için ek kaynaklara başvurulması önemlidir.

Bu sınırlamalar göz önüne alındığında, bu araştırmanın genel bir bakış sunmak ve temel bir anlayış sağlamak amacıyla değerli bir kaynak olduğunu belirtmek önemlidir. Ancak, belirli ihtiyaçları veya karmaşık senaryoları ele alan daha özelleştirilmiş çalışmaların da yapılması önerilir.

Bu araştırmanın önemi, güvenlik testinin önemini vurgulayarak ve Security Testing'in uygulanmasında bilgi sağlayarak birkaç açıdan ortaya çıkar:

Güvenlik Açıklarının Belirlenmesi: Araştırma, Security Testing'in bir uygulamanın güvenlik açıklarını belirlemek için önemli bir araç olduğunu vurgular. Güvenlik testi, potansiyel zayıflıkları tespit etmek ve saldırılara karşı savunmayı güçlendirmek için önleyici bir yaklaşım sunar. Bu nedenle, güvenlik testinin etkin bir şekilde uygulanması, organizasyonların siber saldırılara karşı daha dirençli olmalarını sağlayabilir.

Hassas Verilerin Korunması: Araştırma, Security Testing'in hassas verilerin korunması için kritik bir rol oynadığını vurgular. Bir uygulama güvenlik açıklarına sahip olduğunda, saldırganlar hassas verilere erişebilir ve kötü niyetli amaçlarla kullanabilir. Security Testing, bu tür riskleri tespit etmek ve düzeltici önlemler almak için kullanılarak, verilerin gizliliğini, bütünlüğünü ve erişilebilirliğini sağlamada önemli bir rol oynar.

İş Sürekliliği ve Müşteri Güveni: Güvenlik açıklarının istismar edilmesi, bir organizasyonun iş sürekliliğini ve itibarını ciddi şekilde etkileyebilir. Araştırma, Security Testing'in organizasyonların iş sürekliliğini sağlamak ve müşteri güvenini korumak için önemli bir faktör olduğunu belirtir. Security Testing ile güvenlik açıkları belirlenir ve düzeltici önlemler alınarak, organizasyonlar siber saldırılara karşı daha dirençli hale gelebilir ve müşteri güvenini artırabilir.

Uyumluluk ve Yasa ve Düzenlemelere Uygunluk: Birçok sektörde, belirli yasal ve düzenleyici gereksinimler vardır ve organizasyonlar bu gereksinimlere uyum

sağlamak zorundadır. Araştırma, Security Testing'in uyumluluk ve yasa ve düzenlemelere uygunluk süreçlerinde önemli bir rol oynadığını vurgular. Security Testing, güvenlik kontrollerinin etkinliğini değerlendirir ve gereksinimleri karşılamak için önemli bir adımdır.

Bu nedenlerle, Security Testing üzerine yapılan araştırmalar, organizasyonların güvenliklerini artırmalarına, siber saldırılara karşı korunmalarına ve hassas verileri korumalarına yardımcı olmak açısından büyük bir öneme sahiptir.

# Bölüm 2

# Security Testing Kavramları ve Temel İlkeler

Bu bölümde, Security Testing'in tanımı ve temel ilkeleri incelenir. Güvenlik testinin amacı ve faydaları üzerinde durulur. Ayrıca, Security Testing'in genel yöntemleri ve teknikleri hakkında bilgi verilir.

## 2.1 Security Testing nedir?

Security Testing, bir sistem, uygulama veya ağın güvenlik düzeyini değerlendirmek için gerçekleştirilen bir test sürecidir. Temel amacı, potansiyel güvenlik açıklarını tespit etmek, saldırılara karşı savunmasını değerlendirmek ve güvenlik düzeyini artırmaktır. Security Testing, bilgi sistemlerinin ve verilerin korunması için önemli bir adımdır ve siber saldırılara karşı savunma sağlamaya yardımcı olur.

Security Testing çeşitli yöntemler ve teknikler kullanır. Bunlar şunları içerebilir:

Zafiyet Taraması (Vulnerability Scanning): Bir sistemde veya ağda mevcut olan güvenlik açıklarını belirlemek için otomatik araçlar kullanılarak gerçekleştirilen taramadır. Bu taramalar, bilinen zafiyet veritabanlarını kullanarak sistemdeki açıkları tespit eder.

Penetrasyon Testi (Penetration Testing): Bir sisteme veya ağa saldırı benzeri davranışlar gerçekleştirerek güvenlik açıklarını tespit etmeyi amaçlar. Bu testler, sistemin güvenlik kontrollerini, ağa erişim düzeyini ve veri gizliliğini değerlendirmek için yapılan kontrollü saldırıları içerir.

Ağ Güvenlik Testi (Network Security Testing): Bir ağın güvenlik düzeyini değerlendirmek için gerçekleştirilen testlerdir. Ağ güvenlik testleri, ağa erişim kontrollerini, güvenlik duvarlarını, ağdaki zayıf noktaları ve saldırılara karşı koruma mekanizmalarını değerlendirir.

Web Uygulama Güvenlik Testi (Web Application Security Testing): Web uygulamalarının güvenlik açıklarını belirlemek için gerçekleştirilen testlerdir. Bu testler, SQL enjeksiyonu, cross-site scripting (XSS), güvenlik duvarı atlatma gibi yaygın web uygulama saldırılarına karşı savunmasını test etmeyi amaçlar.

Kod İncelemesi (Code Review): Uygulama veya sistemdeki yazılım kodlarının güvenlik açıklarını tespit etmek için gerçekleştirilen bir inceleme sürecidir. Bu inceleme, potansiyel zafiyetleri, hataları veya güvenlik açıklarını belirlemek için yazılım kodunu detaylı bir şekilde analiz eder.

Sosyal Mühendislik Testi (Social Engineering Testing): Saldırganların insanları manipüle ederek hassas bilgilere erişmelerine dayanan bir test türüdür. Sosyal mühendislik testleri, personelin güvenlik politikalarına uygun davranıp davranmadığını, dikkatsizlik veya güvenlik farkındalığı eksikliklerini değerlendirir.

```python
import tweepy

# Twitter API erişim anahtarlarınızı buraya girin
consumer_key = "YOUR_CONSUMER_KEY"
consumer_secret = "YOUR_CONSUMER_SECRET"
access_token = "YOUR_ACCESS_TOKEN"
access_token_secret = "YOUR_ACCESS_TOKEN_SECRET"

1 usage
def get_user_tweets(username):
    auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
    auth.set_access_token(access_token, access_token_secret)
    api = tweepy.API(auth)

    tweets = api.user_timeline(screen_name=username, count=10)  # İstenilen kullanıcının son 10 tweetini almak için

    for tweet in tweets:
        print(tweet.text)

username = "example_user"  # Verilerini çekmek istediğiniz Twitter kullanıcısının adını buraya girin

get_user_tweets(username)
```

Twitter API kullanarak tweet verisi çekme

Bu Security Testing yöntemleri ve teknikleri, bir sistemin güvenlik açıklarını belirlemek ve düzeltici önlemler almak için kullanılır. Security Testing, güvenlik

politikalarına ve standartlara uyumu değerlendirir ve sistemlerin siber saldırılara karşı savunmasını güçlendirir.

## 2.2 Güvenlik Testi'nin Amacı ve Faydaları

Güvenlik testinin temel amacı, bir sistem, uygulama veya ağın güvenlik düzeyini değerlendirmek ve potansiyel güvenlik açıklarını tespit etmektir. Bu testler, bir organizasyonun bilgi varlıklarını korumak, siber saldırılara karşı direnç sağlamak ve güvenlik politikalarının etkinliğini değerlendirmek için kullanılır. Güvenlik testlerinin aşağıdaki faydaları bulunmaktadır:

Güvenlik Açıklarının Tespiti: Güvenlik testleri, bir sistemde veya uygulamada potansiyel güvenlik açıklarını tespit ederek organizasyonun risk profilini belirlemeye yardımcı olur. Bu açıklar, kötü niyetli kişilerin saldırı girişimlerine maruz kalmadan önce tanımlanarak düzeltilir.

Savunma ve Güvenlik Önlemlerinin İyileştirilmesi: Güvenlik testleri, güvenlik kontrollerinin etkinliğini değerlendirir ve zayıf noktaları belirler. Bu, organizasyonun savunma mekanizmalarını ve güvenlik politikalarını güçlendirmek için iyileştirme adımları atmasına olanak sağlar.

Saldırılara Karşı Direnç Sağlama: Güvenlik testleri, saldırganların gerçek saldırılarını simüle ederek sistemin ne kadar dirençli olduğunu test eder. Bu, organizasyonun siber saldırılara karşı savunmasını güçlendirir ve saldırılara karşı hazırlıklı olmayı sağlar.

Hukuki ve Uyum Yükümlülüklerini Karşılamak: Bazı sektörlerde, organizasyonlar belirli güvenlik standartlarını karşılamak zorundadır. Güvenlik testleri, bu standartları yerine getirmek ve uyumluluğu sağlamak için önemli bir adımdır. Ayrıca, güvenlik testleri, veri koruma yasaları ve regülasyonlarına uyumu değerlendirmek için de kullanılabilir.

İş Sürekliliğini Sağlama: Güvenlik testleri, sistemlerin dayanıklılığını ve iş sürekliliğini değerlendirmek için kullanılır. Potansiyel güvenlik açıkları ve saldırı noktaları belirlenerek, organizasyonun siber saldırılara karşı daha dirençli olması ve hizmet kesintilerini en aza indirmesi sağlanır.

Müşteri ve Paydaş Güvenini Artırma: Güvenlik testlerinin düzenli olarak yapılması, organizasyonun müşterileri ve paydaşları için güvenilir bir ortak olduğunu gösterir. Bu, müşteri ve paydaş güvenini artırır ve rekabet avantajı sağlar.

Güvenlik testleri, organizasyonların bilgi varlıklarını korumak, siber saldırılara karşı direnç sağlamak ve güvenlik politikalarını etkin bir şekilde uygulamak için kritik bir adımdır. Bu testler, güvenlik açıklarını belirlemek ve düzeltici önlemler almak için önemli bir araçtır.

## 2.3 Security Testing Yöntemleri ve Teknikleri

Security Testing için kullanılan çeşitli yöntemler ve teknikler vardır. İşte yaygın olarak kullanılan bazı Security Testing yöntemleri ve teknikleri:

Zafiyet Taraması (Vulnerability Scanning): Zafiyet taraması, bir sistem, ağ veya uygulama üzerinde bilinen güvenlik açıklarını tespit etmek için otomatik tarayıcılar veya araçlar kullanarak gerçekleştirilen bir işlemdir. Bu taramalar, sistemdeki zafiyet veritabanlarını kullanarak potansiyel açıkları belirler. Zafiyet taramaları genellikle ağ güvenliği için kullanılır.

Penetrasyon Testi (Penetration Testing): Penetrasyon testi, bir sistem, uygulama veya ağın güvenlik açıklarını tespit etmek ve değerlendirmek için gerçek saldırılara benzer denemelerin yapıldığı kontrollü bir süreçtir. Bu testler, sistemlerin saldırılara karşı dayanıklılığını test etmek ve zayıf noktaları belirlemek için kullanılır. Penetrasyon testleri genellikle etik hackerlar veya güvenlik uzmanları tarafından gerçekleştirilir.

Web Uygulama Güvenlik Testi (Web Application Security Testing): Web uygulamalarının güvenlik açıklarını tespit etmek için gerçekleştirilen testlerdir. Bu testler, yaygın web uygulama saldırılarına karşı savunmayı değerlendirmek ve potansiyel açıkları tespit etmek için yapılan denemeleri içerir. Örnek olarak SQL enjeksiyonu, cross-site scripting (XSS), kimlik avı (phishing) gibi saldırı türlerini içeren testler yapılabilir.

Ağ Güvenlik Testi (Network Security Testing): Ağ güvenlik testi, bir ağın güvenlik düzeyini değerlendirmek ve zayıf noktaları belirlemek için gerçekleştirilen testlerdir. Bu testler, ağdaki güvenlik duvarları, ağ bileşenleri, erişim kontrolü ve ağa bağlı

cihazlar gibi unsurları içerir. Ağ güvenlik testleri, ağdaki açıkları tespit etmek ve ağa yönelik saldırılara karşı savunmayı güçlendirmek için kullanılır.

Mobil Uygulama Güvenlik Testi (Mobile Application Security Testing): Mobil uygulamaların güvenlik açıklarını belirlemek için gerçekleştirilen testlerdir. Bu testler, mobil uygulamanın kodunu, depolama alanını, veri iletişimini ve güvenlik kontrollerini değerlendirir. Mobil uygulama güvenlik testleri, güvenli mobil uygulama geliştirme sürecinin bir parçası olarak yapılmalıdır.

Sosyal Mühendislik Testi (Social Engineering Testing): Sosyal mühendislik testi, saldırganların insanları manipüle ederek güvenlik açıklarını kullanmalarını veya hassas bilgilere erişmelerini hedefleyen bir test türüdür. Bu testler, personelin güvenlik farkındalığını ve davranışını değerlendirir. Örnek olarak telefonla kimlik avı, sahte e-postalarla kimlik avı veya sosyal medya üzerinden saldırılar gibi senaryolar kullanılabilir.

```python
from instagram_scraper.app import InstagramScraper

1 usage
def scrape_user_posts(username):
    scraper = InstagramScraper()
    scraper.profile_page(username, 10)  # İstenilen kullanıcının son 10 gönderisini almak için

    for post in scraper.medias():
        print(post.url)

username = "example_user"  # Verilerini çekmek istediğiniz Instagram kullanıcısının adını buraya girin

scrape_user_posts(username)
```

Instagram API kullanarak gönderi verisi çekme

Bu yöntemler ve teknikler, Security Testing'in farklı yönlerini kapsayan ve güvenlik açıklarını belirleme ve düzeltme sürecinde önemli bir rol oynayan araçlardır. Farklı testlerin kullanımı, sistemlerin, uygulamaların veya ağların güvenlik düzeyini artırmak ve potansiyel riskleri azaltmak için entegre bir yaklaşımı gerektirebilir.

## 2.4 Security Testing Süreci ve Aşamaları

Security Testing süreci, genellikle aşağıdaki aşamalardan oluşur:

Planlama ve Hazırlık: Bu aşamada, Security Testing süreci için bir plan oluşturulur. Test hedefleri belirlenir, kaynaklar tahsis edilir ve testin kapsamı ve süresi belirlenir. Ayrıca, test ortamı ve gereksinimleri hazırlanır ve test için gerekli araçlar ve kaynaklar temin edilir.

Bilgi Toplama ve Analiz: Bu aşamada, hedef sisteme veya uygulamaya ilişkin bilgi toplanır. Bu bilgiler, ağ topolojisi, kullanılan teknolojiler, yetkilendirme ve kimlik doğrulama mekanizmaları, güvenlik politikaları vb. içerebilir. Bu bilgiler, testin kapsamını ve yöntemlerini belirlemek için analiz edilir.

Zafiyet Keşfi: Bu aşamada, hedef sistemin veya uygulamanın potansiyel güvenlik açıkları ve zafiyetleri tespit edilir. Bu aşamada, zafiyet taraması, penetrasyon testleri, kod incelemesi ve diğer güvenlik testi teknikleri kullanılabilir. Elde edilen sonuçlar, sistemin güvenlik açısından hassas noktalarını belirlemek için değerlendirilir.

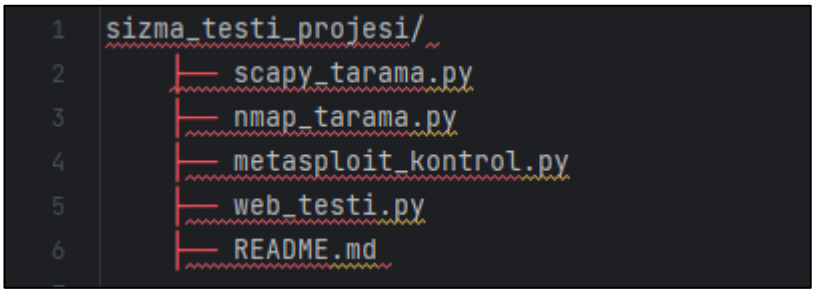
Açıkların Değerlendirilmesi ve Sınıflandırılması: Bu aşamada, tespit edilen güvenlik açıkları ve zafiyetler değerlendirilir ve önceliklendirilir. Açıklar, ciddiyet düzeyine, etkilenen bileşenlere ve potansiyel etkilere göre sınıflandırılır. Bu, düzeltilmesi gereken en kritik açıkların belirlenmesine yardımcı olur.

Saldırı ve Test Senaryolarının Yürütülmesi: Bu aşamada, tespit edilen güvenlik açıklarını test etmek ve saldırı senaryolarını uygulamak için testler gerçekleştirilir. Bu aşamada, penetrasyon testleri, web uygulama saldırıları, ağ saldırıları veya sosyal mühendislik gibi test senaryoları kullanılabilir. Test sonuçları dökümante edilir ve analiz edilir.

Sonuçların Değerlendirilmesi ve Raporlama: Bu aşamada, gerçekleştirilen testlerin sonuçları değerlendirilir ve bir test raporu hazırlanır. Raporda, tespit edilen güvenlik açıkları, sınıflandırmaları, etkileri ve önerilen düzeltici önlemler belirtilir. Bu rapor, sistem sahiplerine veya ilgili paydaşlara sunularak düzeltici eylemler için bir temel sağlar.

Düzeltici Eylemler: Test raporunda belirtilen güvenlik açıkları ve önerilen önlemler doğrultusunda, gerekli düzeltici eylemler alınır. Bu aşamada, güvenlik açıklarının kapatılması, sistem veya uygulamanın güvenlik kontrollerinin güçlendirilmesi ve diğer önleyici tedbirlerin uygulanması sağlanır.

Security Testing süreci, tekrarlanabilir ve sürekli bir yaklaşım gerektirir. Sistemlerde yapılan değişiklikler, yeni güvenlik tehditleri veya teknolojik gelişmeler, sürecin tekrarlanmasını ve güncellenmesini gerektirebilir.



```python
import os
# Proje klasörünü oluşturma
os.makedirs('sizma_testi_projesi')

# scapy_tarama.py dosyasını oluşturma
scapy_code = '''
from scapy.all import *

# Scapy kodlarını buraya ekleyin
'''
with open('sizma_testi_projesi/scapy_tarama.py', 'w') as file:
    file.write(scapy_code)

# nmap_tarama.py dosyasını oluşturma
nmap_code = '''
import nmap
# Nmap kodlarını buraya ekleyin
'''
with open('sizma_testi_projesi/nmap_tarama.py', 'w') as file:
    file.write(nmap_code)

# metasploit_kontrol.py dosyasını oluşturma
metasploit_code = '''
# Metasploit kontrol kodlarını buraya ekleyin
'''
with open('sizma_testi_projesi/metasploit_kontrol.py', 'w') as file:
    file.write(metasploit_code)

# web_testi.py dosyasını oluşturma
web_test_code = '''
import requests
# Web testi kodlarını buraya ekleyin
'''
with open('sizma_testi_projesi/web_testi.py', 'w') as file:
    file.write(web_test_code)

# README.md dosyasını oluşturma
readme_content = '''
# Sızma Testi Projesi

Bu proje, sızma testi için Python kullanımını göstermektedir.

## Proje Yapısı

- `scapy_tarama.py`: Ağ taramaları ve paket manipülasyonları için scapy kullanımı.
- `nmap_tarama.py`: Nmap taramaları için python-nmap kullanımı.
- `metasploit_kontrol.py`: Metasploit Framework kontrolü için Python kullanımı.
- `web_testi.py`: Web uygulamalarının test edilmesi için requests veya selenium kullanımı.

## Kullanım

Her bir Python dosyasını ilgili kütüphaneleri kurduktan sonra çalıştırabilirsiniz.

'''
with open('sizma_testi_projesi/README.md', 'w') as file:
    file.write(readme_content)
```

# Bölüm 3

# Security Testing Türleri

Bu bölümde, farklı Security Testing türleri ele alınır. Beyaz Şapka, Siyah Şapka ve Gri Şapka Testleri gibi farklı yaklaşımlar açıklanır. Fonksiyonel ve fonksiyonel olmayan Security Testing, pasif ve aktif Security Testing gibi farklı kategorilendirmeler üzerinde durulur.

## 3.1 Pasif ve Aktif Güvenlik Testleri

**Pasif Güvenlik Testleri**: Pasif güvenlik testleri, hedef sistem veya uygulamaya doğrudan müdahale etmeden yapılan testlerdir. Bu testler, sistemin veya uygulamanın güvenlik durumunu analiz etmek ve potansiyel zafiyetleri belirlemek için trafiği izleyerek, günlükleri inceleyerek veya verileri analiz ederek yapılır. Pasif güvenlik testleri, ağ trafiğini yakalamak, protokol analizi yapmak, veritabanı günlüklerini incelemek veya zafiyet taraması sonuçlarını analiz etmek gibi yöntemleri içerebilir. Bu testler, dışarıdan yapılan saldırıları simüle etmeyi değil, mevcut durumu değerlendirmeyi amaçlar.

**Aktif Güvenlik Testleri**: Aktif güvenlik testleri, hedef sistem veya uygulamaya doğrudan müdahale ederek yapılan testlerdir. Bu testler, gerçek saldırı senaryolarını taklit ederek sistemde veya uygulamada potansiyel güvenlik açıklarını tespit etmeyi amaçlar. Aktif güvenlik testleri, zafiyet taraması, penetrasyon testleri, web uygulama saldırıları ve sosyal mühendislik saldırıları gibi aktif saldırı senaryolarını içerir. Bu testler, bir saldırganın sisteme yönelik gerçek saldırılarını simüle etmeyi amaçlar ve zafiyetlerin gerçek dünya koşullarında nasıl sömürülebileceğini gösterir.

Her iki yaklaşım da farklı güvenlik açıklarını tespit etme ve değerlendirme yöntemlerini kullanır. Pasif güvenlik testleri daha gözlemleyici ve analitik bir

yaklaşımı benimserken, aktif güvenlik testleri daha saldırgan ve gerçek dünya senaryolarını taklit eden bir yaklaşımı benimser. Her iki yaklaşımın kullanımı, bir organizasyonun güvenlik durumunu kapsamlı bir şekilde değerlendirmek ve güvenlik açıklarını belirlemek için önemlidir.

## 3.2 Beyaz Şapka, Siyah Şapka ve Gri Şapka Yaklaşımları

**Beyaz Şapka (White Hat) Yaklaşımı**: Beyaz şapka yaklaşımı, güvenlik uzmanlarının etik sınırlar içinde ve organizasyonun izniyle gerçekleştirdiği güvenlik testlerini ifade eder. Beyaz şapkalar, sistemlerin veya uygulamaların güvenlik zafiyetlerini tespit etmek, açıkları analiz etmek ve düzeltici önlemler önermek için çalışırlar. Bu yaklaşım, organizasyonların güvenlik düzeyini artırmak ve saldırılara karşı dirençli hale getirmek amacıyla kullanılır. Beyaz şapka yaklaşımı, etik hackerlar, güvenlik danışmanları ve siber güvenlik uzmanları tarafından benimsenir.

**Siyah Şapka (Black Hat) Yaklaşımı**: Siyah şapka yaklaşımı, bilgisayar korsanlarının veya saldırganların, izinsiz olarak sistemlere veya uygulamalara saldırmak, güvenlik açıklarını sömürmek veya bilgi hırsızlığı yapmak amacıyla gerçekleştirdiği faaliyetleri ifade eder. Bu yaklaşım, kötü niyetli saldırıları ve siber suçları içerir. Siyah şapka saldırganları, organizasyonların güvenlik zayıflıklarından faydalanarak zararlı etkinliklerde bulunurlar. Siyah şapka yaklaşımı yasa dışıdır ve genellikle hukuki sonuçları vardır.

**Gri Şapka (Grey Hat) Yaklaşımı**: Gri şapka yaklaşımı, beyaz ve siyah şapka yaklaşımlarının bir karışımını ifade eder. Gri şapkalar, genellikle izinsiz veya yetkisiz bir şekilde bir sisteme veya uygulamaya erişirler, ancak saldırgan niyet taşımadıklarını ve güvenlik açıklarını keşfettiklerinde bunları paylaşmayı hedeflediklerini iddia ederler. Gri şapka yaklaşımı, güvenlik açıklarını ortaya çıkarmak, bilinç oluşturmak ve düzeltici önlemler alınmasını sağlamak için kullanılır. Bununla birlikte, gri şapka saldırılarının yasal durumu belirsiz olabilir ve organizasyonlar tarafından farklı şekillerde karşılanabilir.

Bu yaklaşımlar, güvenlik testleri ve siber güvenlik çalışmalarında farklı niyetler ve etik kurallar temelinde faaliyet gösteren kişileri ifade etmektedir. Beyaz şapka yaklaşımı genellikle kabul edilen ve etik olarak kabul edilen bir metodoloji olarak

kabul edilirken, siyah şapka yaklaşımı yasa dışı ve etik olmayan faaliyetleri ifade eder. Gri şapka yaklaşımı ise tartışmalı bir alandır ve etik ve yasal konuları içerir.

## 3.3 Manuel ve Otomatik Güvenlik Testleri

**Manuel Güvenlik Testleri**: Manuel güvenlik testleri, güvenlik uzmanlarının el ile gerçekleştirdiği testlerdir. Bu yaklaşım, insan beceri ve uzmanlığını kullanarak sistemleri veya uygulamaları analiz etmeyi ve güvenlik açıklarını tespit etmeyi amaçlar. Manuel güvenlik testleri genellikle derinlemesine inceleme, karmaşık senaryoların simülasyonu ve zafiyetlerin el ile kontrol edilmesi gibi yöntemleri içerir. Bu yaklaşım, özel durumları ve uygulamaları hedef alan özelleştirilmiş testler için tercih edilir. Manuel testler, daha esneklik sağlar ve spesifik saldırı senaryolarının analizini yapma yeteneği sunar.

**Otomatik Güvenlik Testleri**: Otomatik güvenlik testleri, bilgisayar tabanlı araçların kullanıldığı testlerdir. Bu yaklaşım, güvenlik açıklarını tespit etmek, zafiyet taraması yapmak veya güvenlik kontrollerini otomatik olarak değerlendirmek için önceden programlanmış araçları kullanır. Otomatik güvenlik testleri, genellikle daha geniş kapsamlı testlerde kullanılır ve hızlı sonuçlar elde etmek için büyük miktarda veri ve trafiği analiz edebilme yeteneğine sahiptir. Bu yaklaşım, tekrarlanabilirlik ve ölçeklenebilirlik sağlar. Otomatik test araçları, zafiyet taraması, güvenlik açığı tespiti, güvenlik kontrolleri değerlendirmesi ve log analizi gibi görevleri gerçekleştirebilir.

Her iki yaklaşımın da avantajları ve dezavantajları vardır. Manuel güvenlik testleri, insan odaklı analiz ve esneklik sağlar, ancak zaman alıcı olabilir ve insan hatalarına açık olabilir. Otomatik güvenlik testleri, hızlı sonuçlar ve ölçeklenebilirlik sağlar, ancak bazı spesifik senaryoları veya derinlemesine analiz gerektiren durumları atlayabilir. Genellikle, güvenlik test sürecinde hem manuel hem de otomatik testlerin bir kombinasyonu kullanılarak en iyi sonuçlar elde edilir.

## 3.4 Bilgi Toplama ve Saldırı Simülasyonu Yöntemleri

1. **Bilgi Toplama Yöntemleri**: Bilgi toplama yöntemleri, hedef sistem veya uygulama hakkında bilgi edinmek için kullanılan tekniklerdir. Bu yöntemler, sistemin veya uygulamanın yapısını, konfigürasyonunu, erişim noktalarını, kullanılan teknolojileri ve diğer önemli bilgileri belirlemeyi amaçlar. Bilgi toplama yöntemleri, saldırganların hedef sistem veya uygulama hakkında daha fazla bilgi edinmelerini ve saldırı senaryolarını geliştirmelerini sağlar. Bazı yaygın bilgi toplama yöntemleri şunlardır:
    - Ağ Taraması: Ağ taraması, hedef ağ üzerindeki sistemleri ve hizmetleri keşfetmek için kullanılan bir yöntemdir. Ağ taraması, IP adresleri, açık portlar, erişilebilir hizmetler ve ağda bulunan diğer bilgileri belirlemeyi amaçlar.
    - Veri Toplama: Veri toplama, hedef sistem veya uygulama hakkında bilgi elde etmek için kamuya açık kaynaklardan veya diğer kaynaklardan veri toplama işlemidir. Bu yöntem, hedef organizasyonun web sitesi, sosyal medya profilleri, bloglar, haberler ve diğer kaynaklardan bilgi toplamayı içerir.
    - Sosyal Mühendislik: Sosyal mühendislik, insanları manipüle ederek bilgi elde etmeyi amaçlayan bir yöntemdir. Saldırganlar, telefonda, e-postada veya yüz yüze görüşme gibi iletişim yöntemlerini kullanarak hedef organizasyonun çalışanlarından bilgi edinmeye çalışabilir.

```python
import socket

target_ip = "192.168.0.14"  # Hedef IP adresini buraya girin
start_port = 1  # Tarama yapılacak başlangıç portu
end_port = 100  # Tarama yapılacak bitiş portu

for port in range(start_port, end_port + 1):
    try:
        sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        sock.settimeout(1)  # Bağlantı zaman aşımını 1 saniye olarak ayarla
        result = sock.connect_ex((target_ip, port))
        if result == 0:
            print(f"Port {port} açık")
        sock.close()
    except socket.error:
        print(f"Hata port {port}'da oluştu")
```

Python ile port taraması yapma

2. Saldırı Simülasyonu Yöntemleri: Saldırı simülasyonu yöntemleri, gerçek saldırı senaryolarını taklit ederek sistemlerin veya uygulamaların güvenlik durumunu değerlendirmek için kullanılan tekniklerdir. Bu yöntemler, güvenlik açıklarını tespit etmek, zafiyetleri sömürmek ve saldırıya uğrayan sistemlerin tepkilerini değerlendirmek amacıyla kullanılır. Saldırı simülasyonu yöntemleri, saldırgan niyet taşımayan güvenlik uzmanları tarafından gerçekleştirilir ve organizasyonların savunma mekanizmalarını test etmeyi amaçlar. Bazı yaygın saldırı simülasyonu yöntemleri şunlardır:
    - Zafiyet Taraması: Zafiyet taraması, sistem veya uygulamalarda potansiyel güvenlik açıklarını tespit etmek için otomatik araçlar veya manuel analiz kullanarak yapılan bir yöntemdir. Zafiyet taraması, bilinen zafiyetleri ve güvenlik açıklarını tespit etmek için sistemleri veya uygulamaları tarar.
    - Penetrasyon Testi: Penetrasyon testi, bir sistem veya uygulama üzerinde gerçek saldırı senaryolarını simüle ederek güvenlik açıklarını tespit etmek için yapılan bir yöntemdir. Penetrasyon testi, saldırganların bakış açısını taklit eder ve gerçek saldırıların sonuçlarını değerlendirir.
    - Sosyal Mühendislik Saldırıları: Sosyal mühendislik saldırıları, hedef organizasyonun çalışanlarını manipüle etmek veya kandırmak için

kullanılan bir yöntemdir. Bu saldırılar, kullanıcıların güvenlik önlemlerini atlamalarını sağlamak veya hassas bilgilere erişim sağlamak amacıyla yapılır.

```python
from geopy.geocoders import Nominatim

1 usage
def get_location(address):
    geolocator = Nominatim(user_agent="my-app")  # Nominatim coğrafi konum hizmetini kullanmak için

    location = geolocator.geocode(address)

    if location:
        print("Adres: ", location.address)
        print("Enlem: ", location.latitude)
        print("Boylam: ", location.longitude)
    else:
        print("Konum bulunamadı")

address = "1600 Amphitheatre Parkway, Mountain View, CA"  # Coğrafi konumunu çekmek istediğiniz adresi buraya girin

get_location(address)
```

Coğrafi Konum Çıkarma

Bu yöntemler, güvenlik testlerinin etkinliğini artırmak ve organizasyonların güvenlik düzeyini değerlendirmek için önemlidir. Organizasyonlar, bu yöntemleri kullanarak güvenlik açıklarını tespit etmek ve düzeltici önlemler alarak saldırılara karşı dirençli hale gelmek için güvenlik testlerini düzenler.

# Bölüm 4

# Security Testing Araçları ve Teknikleri

Bu bölümde, Security Testing için kullanılan çeşitli araçlar ve teknikler incelenir. Otomatik Security Testing araçları, zafiyet tarama araçları, penetrasyon testi araçları, kod analizi araçları ve saldırı simülasyonu araçları hakkında bilgi verilir. Ayrıca, manuel Security Testing teknikleri de açıklanır, bu da hata enjeksiyonu, güvenlik protokolleri analizi, sosyal mühendislik saldırıları gibi yöntemleri içerir.

## 4.1 Zafiyet Tarama Araçları

Zafiyet tarama araçları, sistemlerde veya uygulamalarda potansiyel güvenlik açıklarını tespit etmek için kullanılan otomatik araçlardır. Bu araçlar, genellikle otomatik olarak güvenlik açıkları tarar, zafiyetleri saptar ve raporlar oluşturur. İşte yaygın olarak kullanılan bazı zafiyet tarama araçları:

1. Nessus: Nessus, popüler bir zafiyet tarama aracıdır. Ağ tabanlı zafiyetleri tarar, açık portları ve hizmetleri tespit eder ve sistemlerdeki güvenlik açıklarını belirler. Kapsamlı bir zafiyet veritabanı ve kullanıcı dostu bir arayüze sahiptir.
2. OpenVAS: OpenVAS (Open Vulnerability Assessment System), açık kaynaklı bir zafiyet tarama aracıdır. Ağ tabanlı zafiyetleri tarar, CVE veritabanını kullanarak güvenlik açıklarını tespit eder ve detaylı raporlar sunar.
3. Nikto: Nikto, web uygulamalarındaki güvenlik açıklarını tespit etmek için kullanılan bir araçtır. Web sunucusu üzerinde yapısal hataları, yanlış yapılandırılmış sunucu dosyalarını ve yaygın güvenlik açıklarını tarar.

```python
import http.client

1 usage
def get_webserver_fingerprint(host, port):
    try:
        conn = http.client.HTTPConnection(host, port)
        conn.request("GET", "/")
        response = conn.getresponse()
        server_banner = response.getheader('Server')
        if server_banner:
            return server_banner
        else:
            return "Server banner not found"
    except Exception as e:
        return f"Error: {str(e)}"

host = "example.com"  # Parmağınızı almak istediğiniz web sitesinin host adını buraya girin
port = 80  # Web sunucusunun port numarasını buraya girin (genellikle 80 veya 443)

fingerprint = get_webserver_fingerprint(host, port)
print(f"Fingerprint: {fingerprint}")
```

Web Sunucu Fingerprint

4. Acunetix: Acunetix, web uygulamaları için bir zafiyet tarama aracıdır. XSS (Cross-Site Scripting), SQL enjeksiyonu, CSRF (Cross-Site Request Forgery) gibi yaygın web güvenlik açıklarını tarar. Ayrıca performans testi ve taranan uygulamalar için raporlama sağlar.
5. Burp Suite: Burp Suite, güvenlik testleri için kullanılan bir paket araçtır. Web uygulamalarını tarar, istemci tarafı güvenlik açıklarını tespit eder, proxy sunucusu olarak kullanılabilir ve web uygulaması üzerinde aktif saldırılar gerçekleştirebilir.
6. Qualys: Qualys, ağ tabanlı güvenlik taramaları ve zafiyet yönetimi için bulut tabanlı bir platformdur. Geniş bir zafiyet veritabanı ve otomatik tarama özellikleri sunar.

Bu sadece bazı örneklerdir ve birçok farklı zafiyet tarama aracı bulunmaktadır. Seçilen araç, ihtiyaçlara, bütçeye ve sistemin veya uygulamanın gereksinimlerine bağlı olarak değişebilir.

## 4.2 Penetrasyon Testi Araçları

Penetrasyon testi araçları, sistemlerin veya uygulamaların güvenlik açıklarını tespit etmek ve gerçek saldırı senaryolarını simüle etmek için kullanılan araçlardır. Bu

araçlar, güvenlik uzmanları tarafından kullanılarak hedef sistemlerin güvenlik seviyelerini değerlendirmek için kullanılır. İşte yaygın olarak kullanılan bazı penetrasyon testi araçları:

1. Metasploit Framework: Metasploit Framework, popüler bir penetrasyon testi aracıdır. Geniş bir saldırı modülü koleksiyonuna sahiptir ve farklı saldırı senaryolarını uygulayarak sistemlere veya uygulamalara sızma yeteneklerini simüle eder.
2. Nmap: Nmap (Network Mapper), ağ keşfi ve güvenlik taraması için kullanılan bir araçtır. Ağdaki sistemleri tespit etmek, açık portları taramak, hizmetlerin sürümlerini belirlemek ve diğer ağ bilgilerini toplamak için kullanılır.

```python
import subprocess

1 usage
def run_nmap_scan(target_ip, options):
    command = f"nmap {options} {target_ip}"
    process = subprocess.Popen(command, shell=True, stdout=subprocess.PIPE, stderr=subprocess.PIPE)
    output, error = process.communicate()

    if error:
        print(f"Hata: {error.decode('utf-8')}")
    else:
        print(output.decode('utf-8'))


target_ip = "127.0.0.1"  # Hedef IP adresini buraya girin
scan_options = "-p 1-100 -sV"  # Nmap tarama seçeneklerini burada belirtin

run_nmap_scan(target_ip, scan_options)
```

Python ile NMAP' taraması yapma

3. Wireshark: Wireshark, ağ analizi ve paket yakalama için kullanılan bir araçtır. Ağ trafiğini izlemek, paketleri analiz etmek ve ağdaki güvenlik açıklarını tespit etmek için kullanılır.

```python
import socket

# Sunucunun IP adresi ve port numarası
SERVER_IP = '127.0.0.1'
SERVER_PORT = 12345

# Soket oluşturma
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# Sunucuya bağlanma
sock.connect((SERVER_IP, SERVER_PORT))
print('Sunucuya bağlanıldı.')

while True:
    # Kullanıcıdan veri girişi alınması
    message = input('Mesajınızı girin (q ile çıkış): ')

    if message == 'q':
        break

    # Mesajın sunucuya gönderilmesi
    sock.sendall(message.encode())

    # Sunucudan gelen yanıtın alınması
    response = sock.recv(1024).decode()
    print('Sunucudan gelen yanıt:', response)

# Soketin kapatılması
sock.close()
```

Soket Programlama

```python
from scapy.all import *

1 usage
def packet_sniffer(packet):
    if packet.haslayer(TCP):
        src_ip = packet[IP].src
        dst_ip = packet[IP].dst
        src_port = packet[TCP].sport
        dst_port = packet[TCP].dport
        payload = packet[TCP].payload

        print(f"Source IP: {src_ip}")
        print(f"Destination IP: {dst_ip}")
        print(f"Source Port: {src_port}")
        print(f"Destination Port: {dst_port}")
        print(f"Payload: {payload}")

sniff(filter="tcp", prn=packet_sniffer, count=10)
```

Paket Sniffing

4. Burp Suite: Burp Suite, web uygulamaları için bir penetrasyon testi aracıdır. Proxy sunucusu olarak kullanılabilir, web uygulaması üzerinde güvenlik açıklarını tespit eder, aktif saldırılar gerçekleştirebilir ve otomatik tarayıcıyla zafiyet taraması yapabilir.
5. Aircrack-ng: Aircrack-ng, kablosuz ağ güvenliği testleri için kullanılan bir araçtır. Kablosuz ağlardaki şifreleme anahtarlarını kırmak, şifreli trafiği izlemek ve kablosuz ağların güvenlik açıklarını tespit etmek için kullanılır.

```python
from scapy.all import *

1 usage
def scan_wireless_networks(interface):
    ssid_set = set()

    # Kablosuz ağ taraması için probe isteği gönderme
    def handle_probe_response(packet):
        if packet.haslayer(Dot11ProbeResp) or packet.haslayer(Dot11Beacon):
            ssid = packet[Dot11Elt].info.decode()
            if ssid not in ssid_set:
                ssid_set.add(ssid)
                print(f"SSID: {ssid}")

    # Kablosuz ağları taramak için sniff fonksiyonunu kullanma
    sniff(iface=interface, prn=handle_probe_response, timeout=10)

interface = "wlan0"  # Kablosuz ağ taraması yapmak için kullanılacak ağ arabirimini buraya girin

scan_wireless_networks(interface)
```

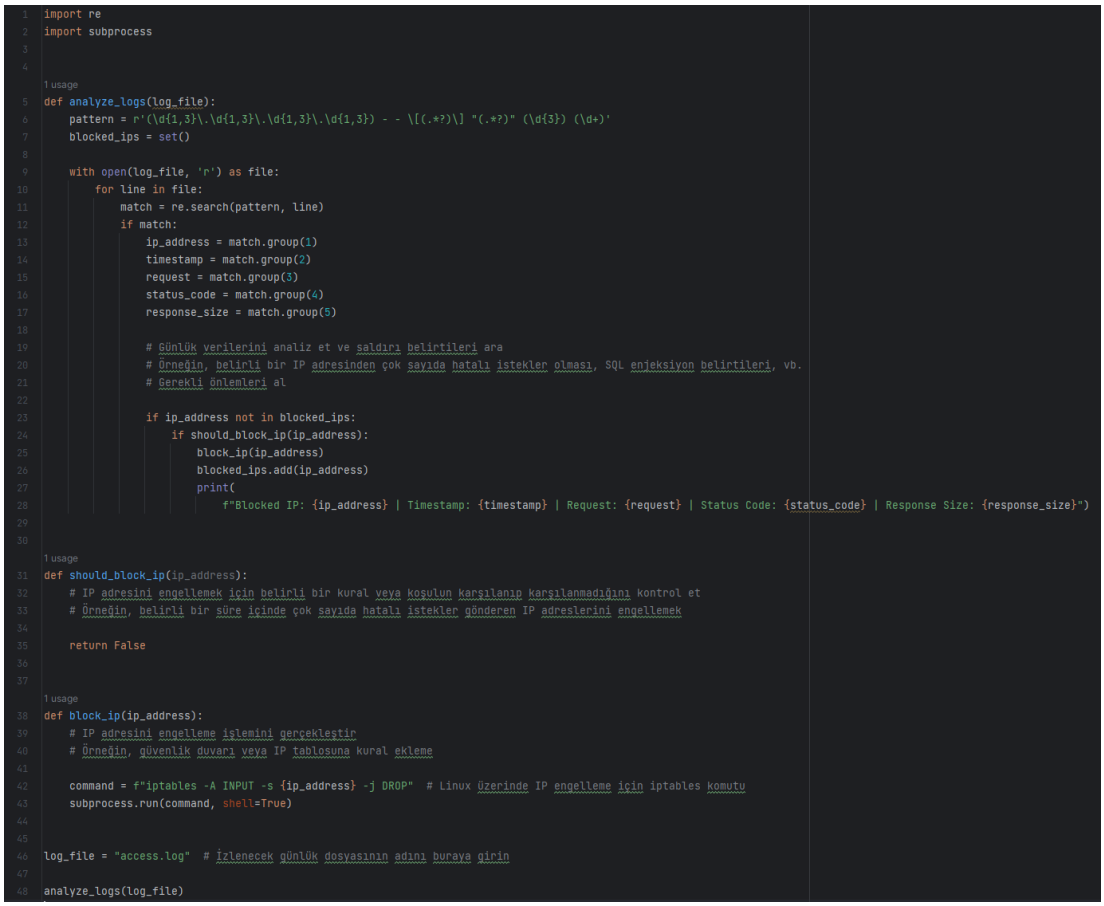
Kablosuz Ağ Taraması

6. Sqlmap: Sqlmap, SQL enjeksiyonu saldırılarını otomatik olarak gerçekleştiren bir araçtır. Web uygulamalarındaki SQL enjeksiyon açıklarını tespit etmek ve veritabanına erişim sağlamak için kullanılır.

Bu sadece bazı örneklerdir ve birçok farklı penetrasyon testi aracı bulunmaktadır. Seçilen araç, ihtiyaçlara, hedef sistem veya uygulamanın özelliklerine ve güvenlik testi hedeflerine bağlı olarak değişebilir.

```python
import re


1 usage
def analyze_logs(log_file):
    pattern = r'(\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}) - - \[(.*?)\] "(.*?)" (\d{3}) (\d+)'

    with open(log_file, 'r') as file:
        for line in file:
            match = re.search(pattern, line)
            if match:
                ip_address = match.group(1)
                timestamp = match.group(2)
                request = match.group(3)
                status_code = match.group(4)
                response_size = match.group(5)

                # Günlük verilerini analiz et ve saldırı belirtileri ara
                # Örneğin, belirli bir IP adresinden çok sayıda hatalı istekler olması, SQL enjeksiyon belirtileri, vb.
                # Gerekli önlemleri al

                print(
                    f"IP: {ip_address} | Timestamp: {timestamp} | Request: {request} | Status Code: {status_code} | Response Size: {response_size}")


log_file = "access.log"  # İzlenecek günlük dosyasının adını buraya girin

analyze_logs(log_file)
```

Günlükleri İzleme



```python
import re
import subprocess


1 usage
def analyze_logs(log_file):
    pattern = r'(\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}) - - \[(.*?)\] "(.*?)" (\d{3}) (\d+)'
    blocked_ips = set()

    with open(log_file, 'r') as file:
        for line in file:
            match = re.search(pattern, line)
            if match:
                ip_address = match.group(1)
                timestamp = match.group(2)
                request = match.group(3)
                status_code = match.group(4)
                response_size = match.group(5)

                # Günlük verilerini analiz et ve saldırı belirtileri ara
                # Örneğin, belirli bir IP adresinden çok sayıda hatalı istekler olması, SQL enjeksiyon belirtileri, vb.
                # Gerekli önlemleri al

                if ip_address not in blocked_ips:
                    if should_block_ip(ip_address):
                        block_ip(ip_address)
                        blocked_ips.add(ip_address)
                        print(
                            f"Blocked IP: {ip_address} | Timestamp: {timestamp} | Request: {request} | Status Code: {status_code} | Response Size: {response_size}")


1 usage
def should_block_ip(ip_address):
    # IP adresini engellemek için belirli bir kural veya koşulun karşılanıp karşılanmadığını kontrol et
    # Örneğin, belirli bir süre içinde çok sayıda hatalı istekler gönderen IP adreslerini engellemek

    return False


1 usage
def block_ip(ip_address):
    # IP adresini engelleme işlemini gerçekleştir
    # Örneğin, güvenlik duvarı veya IP tablosuna kural ekleme

    command = f"iptables -A INPUT -s {ip_address} -j DROP"  # Linux üzerinde IP engelleme için iptables komutu
    subprocess.run(command, shell=True)


log_file = "access.log"  # İzlenecek günlük dosyasının adını buraya girin

analyze_logs(log_file)
```

IP Engelleme

## 4.3 İstismar Araçları

İstismar araçları, güvenlik testi veya penetrasyon testi sırasında kullanılan ve bilinen güvenlik açıklarını veya zafiyetleri istismar etmek için tasarlanmış araçlardır. Bu araçlar, güvenlik uzmanlarının sistemlerin veya uygulamaların güvenlik seviyelerini test etmek ve gerçek saldırı senaryolarını simüle etmek için kullanılır. İşte bazı yaygın istismar araçları:

1. Metasploit Framework: Metasploit Framework, hem penetrasyon testi aracı hem de istismar aracı olarak kullanılabilir. Geniş bir saldırı modülü koleksiyonuna sahip olan Metasploit, bilinen güvenlik açıklarını istismar etmek için kullanılan çeşitli saldırı vektörlerini sağlar.
2. ExploitDB: ExploitDB, bir veritabanı ve arama motoru olarak kullanılan bir kaynak olup, bilinen güvenlik açıklarını içeren istismar kodlarını içerir.

ExploitDB, belirli bir zafiyeti hedefleyen istismarlar aramak ve kullanmak için kullanılır.

3. SET (Social Engineering Toolkit): SET, sosyal mühendislik saldırılarını otomatikleştiren bir araçtır. Bu araç, e-posta, telefon veya phishing gibi sosyal mühendislik saldırıları gerçekleştirmek için kullanılır. SET, kullanıcıları manipüle ederek hassas bilgilere erişim sağlamayı hedefler.
4. BeEF (Browser Exploitation Framework): BeEF, web tarayıcılarını hedefleyen saldırılar için kullanılan bir istismar aracıdır. BeEF, XSS (Cross-Site Scripting) açıklarını istismar etmek ve tarayıcı üzerinde kontrole sahip olmak için kullanılır.
5. sqlmap: sqlmap, web uygulamalarında SQL enjeksiyonu açıklarını istismar etmek için kullanılan bir araçtır. Bu araç, SQL enjeksiyonu saldırılarını otomatik olarak gerçekleştirir ve veritabanına erişim sağlamayı amaçlar.
6. Hydra: Hydra, çoklu oturum açma saldırıları için kullanılan bir araçtır. Bir hedef sistem veya uygulamada kullanıcı adı ve şifre kombinasyonlarını deneyerek oturum açma bilgilerini bulmayı hedefler.

Bu sadece bazı örneklerdir ve birçok farklı istismar aracı bulunmaktadır. Ancak, bu tür araçların yasal ve etik kullanımı önemlidir. Sadece yetkili testler ve izin alınmış sistemler üzerinde kullanılmalıdır.

## 4.4 Kod Analizi Araçları

Kod analizi araçları, yazılım kodlarının güvenlik açıklarını tespit etmek, kaliteyi artırmak ve hataları bulmak için kullanılan araçlardır. Bu araçlar, yazılım geliştirme sürecinde kullanılarak kod tabanını analiz eder ve potansiyel zafiyetleri veya kod hatalarını tespit eder. İşte bazı yaygın kod analizi araçları:

1. SonarQube: SonarQube, açık kaynaklı bir kod analizi platformudur. Farklı programlama dilleri için destek sağlar ve kod tabanını statik analiz, kod kapsamı, güvenlik açıkları ve kod kalitesi açısından tarar. Koda ilişkin detaylı raporlar ve metrikler sunar.
2. Veracode: Veracode, bulut tabanlı bir kod analizi ve güvenlik testi platformudur. Uygulama kodunu statik analiz, dinamik analiz ve bileşen

analizi yöntemleriyle tarar. Potansiyel güvenlik açıklarını tespit eder ve raporlar oluşturur.

3. Checkmarx: Checkmarx, otomatik statik kod analizi aracıdır. Yazılım kodunu tarar, potansiyel güvenlik açıklarını tespit eder ve hataları raporlar. Farklı programlama dilleri ve çevreleri için destek sağlar.
4. Fortify: Fortify, birçok programlama dili için statik kod analizi sağlayan bir güvenlik aracıdır. Yazılım kodunu tarar, güvenlik açıklarını tespit eder ve raporlar sunar. Ayrıca hatalı kod kalıplarını ve güvenlik standartlarına uygunluğu kontrol eder.
5. Coverity: Coverity, yazılım kodunun statik analizini yaparak güvenlik açıklarını ve hataları tespit eden bir araçtır. Yüksek hassasiyetle potansiyel sorunları belirler ve geliştiricilere detaylı analiz raporları sunar.
6. PMD: PMD, Java ve diğer bazı diller için kullanılan bir kod analizi aracıdır. Kod tabanını tarar, hataları ve kod kalitesi sorunlarını tespit eder. Yazılım geliştirme sürecinde kullanılarak kod standardını iyileştirir ve hataları önler.

Bu sadece bazı örneklerdir ve birçok farklı kod analizi aracı bulunmaktadır. Seçilen araç, kullanılan programlama dili, projenin gereksinimleri ve tercihlere bağlı olarak değişebilir.

## 4.5 Güvenlik Testi Teknikleri

Güvenlik testi teknikleri, sistemlerin veya uygulamaların güvenlik açıklarını tespit etmek ve değerlendirmek için kullanılan çeşitli yöntemlerdir. İşte bazı yaygın güvenlik testi teknikleri:

1. Zafiyet Taraması: Bir sistemde veya uygulamada potansiyel güvenlik açıklarını tespit etmek için otomatik olarak tarayıcılar, araçlar veya yazılımlar kullanarak yapılan bir tekniktir. Örneğin, açık port taraması, zayıf şifrelerin denetlenmesi, güvenlik duvarı taraması gibi işlemler gerçekleştirilir.
2. XSS (Cross-Site Scripting) Testi: XSS, web uygulamalarında bulunan bir güvenlik açığıdır. Bu test, web uygulamalarında kullanıcı tarafından sağlanan verilerin nasıl işlendiğini kontrol ederek, XSS saldırılarına olanak tanıyan açıkları tespit etmeyi amaçlar.

3. SQL Enjeksiyonu Testi: SQL enjeksiyonu, web uygulamalarında yaygın bir güvenlik açığıdır. Bu test, web uygulamalarının kullanıcı tarafından sağlanan verileri doğru bir şekilde işlediğini ve güvenlik açıklarına neden olabilecek SQL sorgularını önlediğini kontrol eder.
4. Kaba Kuvvet Saldırıları: Kaba kuvvet saldırıları, oturum açma bilgilerini veya şifreleri tahmin etmek için otomatik olarak deneme yapılmasını içerir. Bu test, sistem veya uygulamanın zayıf şifre politikalarını tespit etmek ve gerekli önlemleri almak için kullanılır.
5. Sosyal Mühendislik Testleri: Sosyal mühendislik testleri, kullanıcıların manipüle edilmesi veya yanıltılması yoluyla hassas bilgilere erişimi hedefler. Bu testler, phishing e-postaları, telefon aramaları veya sahte web siteleri gibi yöntemleri kullanarak kullanıcı farkındalığını değerlendirir.

```python
from linkedin_v2 import linkedin

# LinkedIn API erişim anahtarlarınızı buraya girin
access_token = "YOUR_ACCESS_TOKEN"
client_id = "YOUR_CLIENT_ID"
client_secret = "YOUR_CLIENT_SECRET"

1 usage
def get_user_profile(user_id):
    authentication = linkedin.LinkedInAuthentication(client_id, client_secret, access_token)
    api = linkedin.LinkedInApplication(token=authentication.get_access_token())

    profile = api.get_profile(member_id=user_id)

    print(profile)

user_id = "example_user_id"  # Verilerini çekmek istediğiniz LinkedIn kullanıcısının ID'sini buraya girin

get_user_profile(user_id)
```

LinkedIn API kullanarak veri çekme

6. Güvenlik Protokolleri Analizi: Bu test, ağ protokollerini veya güvenlik mekanizmalarını analiz ederek güvenlik açıklarını tespit etmeyi amaçlar. Örneğin, SSL/TLS protokolü üzerinde yapılan analizlerde zayıf şifreleme algoritmaları, sertifika sorunları veya yapılandırma hataları gibi güvenlik açıkları tespit edilebilir.
7. Yetkilendirme ve Kimlik Doğrulama Testleri: Bu testler, sistem veya uygulamanın kullanıcı yetkilendirme ve kimlik doğrulama süreçlerini değerlendirir. Örneğin, güçlü şifre politikaları, oturum süreleri ve oturum yönetimi gibi konular kontrol edilir.

8. Güvenlik Ayarları Değerlendirmesi: Bu testler, sistem veya uygulamanın güvenlik ayarlarını ve konfigürasyonlarını kontrol eder. Örneğin, güvenlik duvarı yapılandırması, erişim kontrolleri, güncelleme politikaları gibi konular değerlendirilir.

Bu sadece bazı örneklerdir ve güvenlik testi teknikleri çeşitli alanlarda ve senaryolarda kullanılabilir. Gerçek bir güvenlik testi, uygulamanın veya sistemin spesifik gereksinimlerine ve tehditlerine uygun olarak planlanmalı ve uygulanmalıdır.

# Bölüm 5

# Security Testing'in Önemi

Security Testing'in önemi, bir organizasyonun, kullanıcıların ve müşterilerin güvenliğini sağlamak, maliyetli veri ihlallerini önlemek ve yasal düzenlemelere uyum sağlamak için vazgeçilmezdir. Güvenliği önemseyen bir yaklaşım, siber saldırılara karşı koruma sağlar ve güvenli bir iş ortamı yaratır.

## 5.1 Veri Güvenliği ve Gizlilik

Veri güvenliği ve gizlilik, bilgi teknolojileri ve bilişim sistemlerindeki verilerin korunması ve yetkisiz erişimden, manipülasyondan veya ifşadan korunmasıdır. Bu kavramlar, kullanıcıların ve kuruluşların hassas bilgilerini korumak için önemlidir. İşte veri güvenliği ve gizlilik hakkında daha fazla bilgi:

Veri Güvenliği: Veri güvenliği, verilerin bütünlüğünü, doğruluğunu ve kullanılabilirliğini sağlamak için alınan önlemleri ifade eder. Bu, verilerin yetkisiz erişimden, veri kaybından veya bozulmadan korunması anlamına gelir. Veri güvenliği, güçlü kimlik doğrulama, şifreleme, erişim kontrolleri, güvenlik duvarları ve güvenlik yazılımları gibi çeşitli güvenlik önlemlerini içerir.

Veri Gizliliği: Veri gizliliği, kişisel bilgiler, ticari sırlar, müşteri verileri veya diğer hassas bilgilerin yetkisiz kişiler tarafından erişilemez veya ifşa edilemez şekilde korunmasıdır. Veri gizliliği, verilerin doğru kişilere sınırlı bir şekilde erişilebilmesini sağlayan gizlilik politikaları, veri şifreleme, veri maskeleme, erişim denetimleri ve kullanıcı yetkilendirme gibi önlemleri içerir.

Veri güvenliği ve gizliliği, hem bireyler hem de kuruluşlar için önemlidir. Aşağıdaki nedenlerle veri güvenliği ve gizliliği sağlanmalıdır:

Kişisel Mahremiyetin Korunması: Veri güvenliği ve gizliliği, kişisel bilgilerin yetkisiz kişilerin eline geçmesini önler. Bu, kişisel mahremiyetin korunmasını sağlar ve kullanıcıların güven duygusunu artırır.

İşletme Rekabet Avantajı: Ticari sırların veya patentli bilgilerin korunması, bir işletmenin rekabet avantajını sürdürmesine yardımcı olur. Veri güvenliği ve gizliliği sağlamak, işletmenin stratejik bilgilerini korur ve rakipler tarafından kullanılmasını önler.

Yasal ve Düzenleyici Uyumluluk: Birçok ülkede, veri koruma yasaları ve düzenlemeleri bulunmaktadır. Veri güvenliği ve gizliliği sağlamak, bu yasalara ve düzenlemelere uyum sağlamayı gerektirir. Aksi takdirde, kuruluşlar yasal sorunlarla karşılaşabilir ve cezai yaptırımlara tabi tutulabilir.

Müşteri Güveni ve İmaj: Veri güvenliği ve gizliliği sağlamak, müşterilerin güvenini kazanmada önemlidir. Müşteriler, kişisel ve finansal bilgilerinin güvende olduğunu bilmek isterler. Güvenli bir veri ortamı sağlamak, müşteri sadakatini artırır ve kuruluşun itibarını korur.

Veri İhlallerinin Önlenmesi: Veri güvenliği ve gizliliği sağlamak, veri ihlallerinin önlenmesine yardımcı olur. Veri ihlalleri ciddi maliyetlere, itibar kaybına ve hukuki sonuçlara neden olabilir. Veri güvenliği ve gizliliği sağlamak, bu riskleri azaltır veya ortadan kaldırır.

Sonuç olarak, veri güvenliği ve gizliliği, bireylerin ve kuruluşların bilgilerinin korunmasında kritik bir rol oynar. Bu, hem kişisel mahremiyeti korumak hem de işletmelerin güvenilirlik ve rekabet avantajını sürdürmek için önemlidir.

## 5.2 Saldırılar ve Tehditler

Bilgi teknolojileri ve bilişim sistemleri üzerinde çeşitli saldırılar ve tehditler bulunmaktadır. Bu saldırılar ve tehditler aşağıdaki gibi sınıflandırılabilir:

DoS (Hizmet Dışı Bırakma) Saldırıları: DoS saldırıları, hedef sistem ya da ağa yoğun talepler göndererek kaynakları tüketmeyi amaçlar. Bu sayede hedef sistemin normal

faaliyetlerini yerine getirememesi sağlanır. Örnek olarak, ping saldırıları, botnet saldırıları ve kaynak tüketme saldırıları verilebilir.

Veri Sızdırma ve İhlal Saldırıları: Bu saldırılar, hassas bilgilere yetkisiz erişim sağlamayı hedefler. Örnek olarak, veri hırsızlığı, veri sızıntısı ve veri ifşası saldırıları verilebilir. Bu saldırılar, kullanıcı kimlik bilgilerini, finansal bilgileri, ticari sırları veya kişisel verileri hedef alabilir.

Zararlı Yazılım Saldırıları: Zararlı yazılım saldırıları, kötü niyetli yazılımların hedef sistemlere veya kullanıcılara bulaşmasını hedefler. Bu yazılımlar, virüsler, solucanlar, truva atları, fidye yazılımları ve casus yazılımlar gibi farklı formlarda olabilir.

Kimlik Avı (Phishing) Saldırıları: Kimlik avı saldırıları, sahte web siteleri, sahte e-postalar veya SMS'ler aracılığıyla kullanıcıların hassas bilgilerini elde etmeyi amaçlar. Saldırganlar, kullanıcıları yanıltmak için güvenilir kuruluşların veya kişilerin kimliklerini taklit ederler.

SQL Enjeksiyon Saldırıları: SQL enjeksiyonu, hedef uygulamanın veritabanına kötü niyetli SQL ifadeleri göndererek veritabanına yetkisiz erişim sağlamayı amaçlar. Bu saldırılar, web uygulamalarında güvenlik açıklarını sömürerek veri çalma, veri değiştirme veya hizmeti dışarıda bırakma gibi sonuçlar doğurabilir.

XSS (Cross-Site Scripting) Saldırıları: XSS saldırıları, hedef web uygulamalarına kötü niyetli komutlar veya betikler ekleyerek kullanıcıların tarayıcılarında çalıştırmayı amaçlar. Bu saldırılar, kullanıcıların tarayıcı oturumlarını ele geçirme, oturum çalma veya kullanıcıları yanıltma gibi sonuçlar doğurabilir.

Fiziksel Saldırılar: Fiziksel saldırılar, sistemlere veya ağ altyapısına doğrudan fiziksel erişim sağlayarak zarar vermeyi amaçlar. Örnek olarak, donanım veya kablo kesme, donanım manipülasyonu ve fiziksel hırsızlık saldırıları verilebilir.

Ağ Saldırıları: Ağ saldırıları, hedef ağa yönelik saldırıları içerir. Örnek olarak, ağ kesme saldırıları, ağ üzerinde casusluk veya paket yakalama saldırıları, ağ altyapısının manipülasyonu ve ağa yetkisiz erişim sağlama saldırıları verilebilir.

Bu sadece birkaç örnek olup, saldırı ve tehditlerin çeşitliliği oldukça fazladır. Her geçen gün yeni saldırı yöntemleri ve tehditler ortaya çıkmaktadır. Bu nedenle,

güvenlik önlemlerini sürekli güncellemek ve saldırılara karşı savunma mekanizmalarını iyileştirmek önemlidir.

## 5.3 Yasal ve Düzenleyici Uyum

Yasal ve düzenleyici uyum, bir kuruluşun ilgili yasalara, düzenlemelere ve standartlara uygun şekilde faaliyet göstermesi anlamına gelir. Bu uyum, belirli sektörlerdeki yasal gerekliliklere ve düzenlemelere tabi olan kuruluşlar için önemlidir. İşletmeler, ilgili yasalara ve düzenlemelere uymakla yükümlüdür ve uyum sağlamak için çeşitli adımlar atmalıdır.

Yasal ve düzenleyici uyumun önemi aşağıdaki şekillerde açıklanabilir:

Yasal Yaptırımlardan Kaçınma: Yasal ve düzenleyici uyum sağlamayan kuruluşlar, hukuki sorunlarla karşılaşabilir ve yasal yaptırımlara maruz kalabilir. Yasalara ve düzenlemelere uymak, bu tür yaptırımlardan kaçınmayı ve hukuki riskleri azaltmayı sağlar.

İtibarın ve Güvenin Korunması: Yasal ve düzenleyici uyum, kuruluşun itibarını korumada önemli bir rol oynar. Müşteriler, tedarikçiler ve iş ortakları, yasalara ve düzenlemelere uygun olarak faaliyet gösteren bir kuruluşa güven duyarlar. Uyumluluk, müşteri memnuniyetini artırabilir ve iş ilişkilerinin sürdürülebilirliğini sağlayabilir.

Müşteri Verilerinin Korunması: Birçok sektörde müşteri verilerinin korunması yasal bir gerekliliktir. Yasal ve düzenleyici uyum, müşteri verilerinin gizliliğini ve güvenliğini sağlamaya yönelik önlemleri içerir. Bu, müşterilerin kişisel bilgilerinin yetkisiz erişim, ifşa veya kötüye kullanımdan korunmasını sağlar.

Sektörel Standartlara Uygunluk: Bazı sektörlerde, belirli standartlara uyum sağlamak yasal bir gerekliliktir. Örneğin, sağlık sektöründe HIPAA (Sağlık Sigortası Taşınabilirlik ve Sorumluluk Yasası), finans sektöründe PCI DSS (Kredi Kartı Endüstrisi Veri Güvenliği Standardı) gibi standartlar vardır. Yasal ve düzenleyici uyum, bu sektörel standartlara uygunluğu sağlamak için önemlidir.

İş Sürekliliği ve Risk Yönetimi: Yasal ve düzenleyici uyum, iş sürekliliğini sağlamak ve riskleri yönetmek için bir çerçeve sunar. Yasal gerekliliklere uyum, işletmenin sürekliliğini sağlamak ve potansiyel riskleri azaltmak için önleyici tedbirler almayı içerir.

Yasal ve düzenleyici uyum, kuruluşların toplumun ve müşterilerin güvenini kazanması, hukuki riskleri azaltması ve iş faaliyetlerini düzenli ve sorunsuz bir şekilde sürdürebilmesi için önemlidir.

## 5.4 Marka ve Müşteri Güveni

Marka ve müşteri güveni, bir kuruluşun müşterileri tarafından güvenilir, saygın ve değerli olarak algılanmasıdır. Müşteri güveni, müşterilerin bir markaya, ürüne veya hizmete olan inançlarını ve sadakatlerini ifade eder. Bu güven, müşterilerin bir markaya bağlılık göstermesini, tekrar satın almayı tercih etmesini ve olumlu bir marka deneyimi yaşamasını sağlar.

Marka güveni ve müşteri güveni, aşağıdaki unsurları içerir:

Kalite ve Güvenilirlik: Müşteri güveni, bir markanın ürün veya hizmetlerinin kalitesine ve güvenilirliğine olan inançla ilişkilidir. Müşteriler, kaliteli ürünler sunan ve güvenilir hizmetler sağlayan markalara güvenirler. Bu güven, müşterilerin markayı tercih etmesini ve sadık bir müşteri tabanı oluşturmasını sağlar.

Müşteri Deneyimi: Müşteri güveni, müşteri deneyimiyle de yakından ilişkilidir. Müşteriler, olumlu bir deneyim yaşadıklarında markaya olan güvenlerini artırır. İyi bir müşteri deneyimi, müşterilerin beklentilerini karşılamak, sorunları çözmek ve müşterilere değer sağlamak üzerine odaklanır.

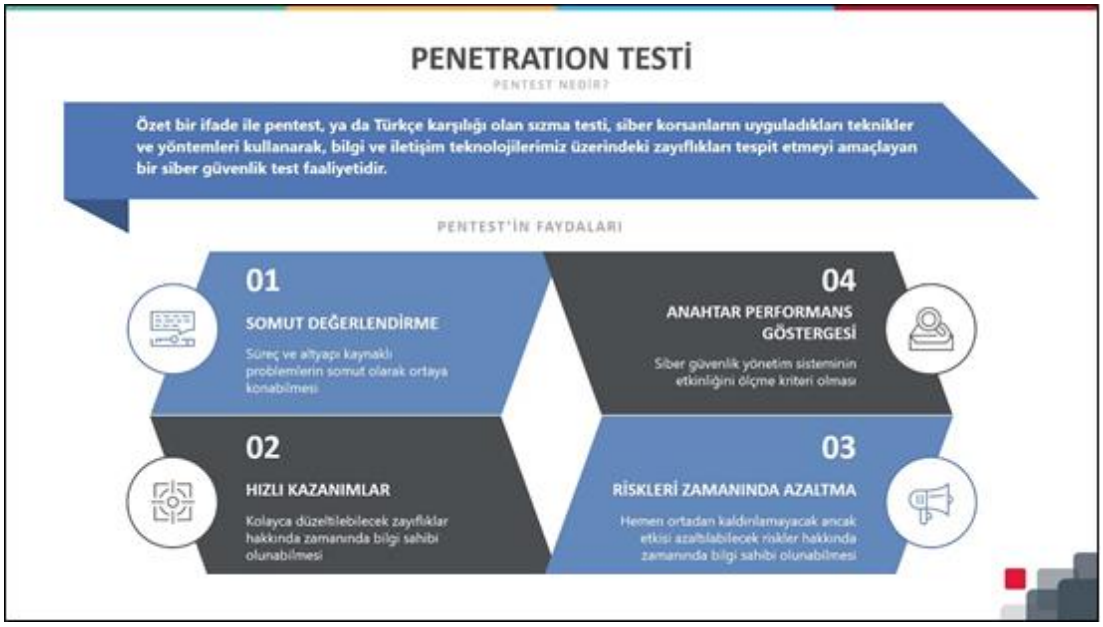
İtibar ve Saygınlık: Marka güveni, bir markanın itibarını ve saygınlığını yansıtır. Marka, dürüstlük, etik değerlere uygunluk, şeffaflık ve müşteri odaklılık gibi unsurlarla ilişkilendirilirse, müşteriler markaya olan güvenlerini artırır. İyi bir itibar, müşterilerin markanın sözlerine inanmasını ve markayla bağ kurmasını sağlar.

Veri Güvenliği ve Gizlilik: Müşteri güveni, kişisel veri güvenliği ve gizliliği ile de ilişkilidir. Müşteriler, kişisel bilgilerinin güvende olduğunu ve markanın bu bilgileri

doğru şekilde kullanacağına güvenirler. Veri güvenliği ve gizliliği sağlamak, müşterilerin güvenini artırır ve markaya olan bağlılığı destekler.

Müşteri İlişkileri: Müşteri güveni, markanın müşterileriyle kurduğu ilişkilerle de ilişkilidir. Müşterilere değer veren, onları dinleyen ve ihtiyaçlarını anlayan bir marka, müşteri güvenini artırır. İyi bir müşteri ilişkisi, müşterilerin markaya olan güvenini ve bağlılığını pekiştirir.

Marka ve müşteri güveni, uzun vadeli başarı için önemlidir. Güvenilir bir marka, müşterilerin tercih ettiği bir marka haline gelir ve rekabetçi bir avantaj sağlar. Müşteri güveninin olumlu yönde yönetilmesi, marka sadakatini artırır, müşteri memnuniyetini sağlar ve olumlu bir marka imajı oluşturur.



# Bölüm 6

# Security Testing Süreci

Bu bölümde, Security Testing süreci adım adım açıklanır. Hedef belirleme ve kapsam tanımı, test planlama, test senaryolarının tasarımı, Security Testing'in uygulanması ve sonuçların analizi, raporlama ve sonuçların değerlendirilmesi, düzeltici önlemler ve sürdürülebilirlik gibi aşamalar detaylı bir şekilde ele alınır.

## 6.1 Hedef Belirleme ve Kapsam Tanımı

Hedef Belirleme ve Kapsam Tanımı, bir güvenlik testi sürecinde yapılacak çalışmanın hedeflerini ve kapsamını belirlemeyi içerir. Bu aşama, testin odaklanacağı sistemlerin, uygulamaların veya ağ bileşenlerinin seçimini ve testin hedeflerini tanımlamayı içerir.

Hedef Belirleme aşamasında aşağıdaki adımlar takip edilir:

Hedef Sistemlerin Belirlenmesi: Güvenlik testi yapılacak sistemlerin veya uygulamaların belirlenmesi önemlidir. Bu sistemler, ağ altyapısı, sunucular, veritabanları, web uygulamaları veya mobil uygulamalar gibi çeşitli bileşenleri içerebilir. Testin odaklanacağı sistemin belirlenmesi, kaynak ve zaman yönetimi açısından önemlidir.

Test Kapsamının Tanımlanması: Güvenlik testi kapsamının belirlenmesi, hangi test tekniklerinin kullanılacağını, hangi saldırı senaryolarının simüle edileceğini ve hangi güvenlik açıklarının hedefleneceğini belirler. Bu aşamada, belirlenen hedefler, sistem bileşenleri, işlevler ve kullanılan teknolojiler dikkate alınarak testin kapsamı belirlenir.

Hedeflerin ve Kısıtlamaların Belirlenmesi: Güvenlik testinin hedefleri netleştirilir ve testin yapılacağı ortamdaki kısıtlamalar belirlenir. Örneğin, testin zaman sınırlamaları, kaynakların kullanılabilirliği, test ortamının yeterliliği gibi faktörler dikkate alınır.

Hedeflerin ve kısıtlamaların belirlenmesi, test sürecinin etkin ve verimli olmasını sağlar.

Kapsam Tanımı aşamasında ise aşağıdaki adımlar izlenir:

Test Alanının Belirlenmesi: Güvenlik testinin kapsamında hangi sistemlerin veya uygulamaların yer aldığı belirlenir. Bu adım, test ekibinin hangi bileşenlere odaklanacağını ve hangi alanlarda test yapılacağını belirler. Örneğin, bir web uygulaması güvenlik testi için, kullanıcı yetkilendirme, veri girişi, güvenlik kontrolleri gibi belirli alanlar kapsama alınabilir.

Kapsamın Sınırlarının Belirlenmesi: Testin kapsamının sınırları belirlenir ve hangi ağ veya sistemlerin test dışı bırakılacağı belirtilir. Bu, test ekibinin hangi bileşenleri test edeceğini ve hangi bileşenlerin dışında bırakılacağını açıklar. Örneğin, üretim ortamı gibi kritik sistemler, testin dışında tutulabilir.

Hedef Belirleme ve Kapsam Tanımı, güvenlik testinin etkin ve verimli bir şekilde gerçekleştirilmesini sağlar. Bu aşamaların doğru bir şekilde yapılandırılması, testin odaklanmasını, kaynakların etkin kullanılmasını ve sonuçların daha sağlam bir şekilde değerlendirilmesini sağlar.

## 6.1 Test Planlama ve Raporlama

Test Planlama ve Raporlama, güvenlik testi sürecinin önemli aşamalarından biridir. Test planlama aşamasında, testin nasıl yapılacağı, hangi adımların izleneceği, kaynakların ve zamanın nasıl kullanılacağı gibi detaylar belirlenir. Raporlama aşamasında ise test sonuçları değerlendirilir ve bir rapor oluşturulur.

Test Planlama aşamasında aşağıdaki adımlar izlenir:

Test Amaçları ve Hedefleri: Testin amacı ve hedefleri belirlenir. Bu, hangi güvenlik açıklarının tespit edilmek istendiği, hangi saldırı senaryolarının simüle edileceği gibi bilgileri içerir.

Test Stratejisi: Güvenlik testi için kullanılacak yöntemler, teknikler ve araçlar belirlenir. Testin manuel olarak mı yapılacağı yoksa otomatik araçlarla mı destekleneceği gibi kararlar alınır.

Test Kaynakları: Test için gereken kaynaklar, ekip üyeleri, test ortamı, test verileri gibi unsurlar belirlenir. Bu kaynakların temin edilmesi ve kullanılması için planlama yapılır.

Test Zamanlaması: Testin hangi zaman aralığında gerçekleştirileceği, test aşamalarının sıralaması ve süreleri belirlenir. Testin tamamlanması için gerekli zaman dilimi planlanır.

Raporlama aşamasında ise aşağıdaki adımlar takip edilir:

Test Sonuçlarının Değerlendirilmesi: Gerçekleştirilen güvenlik testinin sonuçları analiz edilir. Tespit edilen güvenlik açıkları, zafiyetler, hatalar ve riskler değerlendirilir.

Test Raporunun Oluşturulması: Test sonuçlarının özetlendiği ve ayrıntılı bir şekilde raporlandığı bir belge hazırlanır. Bu rapor, tespit edilen güvenlik açıkları, risk değerlendirmesi, saldırı senaryoları, önerilen çözüm önerileri ve iyileştirme önerilerini içerir.

Raporun Sunumu ve İletişimi: Rapor, ilgili paydaşlara sunulur. Bu paydaşlar, sistem sahipleri, yöneticiler, geliştiriciler veya diğer ilgili ekipler olabilir. Raporun anlaşılır ve etkili bir şekilde sunulması, alınacak önlemlerin hızla uygulanmasını sağlar.

Test planlama ve raporlama, güvenlik testi sürecinde testin doğru bir şekilde yapılmasını, sonuçların etkili bir şekilde değerlendirilmesini ve önlemlerin alınmasını sağlar. Bu aşamaların düzgün bir şekilde uygulanması, testin başarılı ve verimli olmasını sağlar.

## 6.3 Güvenlik Testi Uygulama

Güvenlik testi uygulaması, güvenlik testi sürecinin gerçekleştirilmesini içerir. Bu aşamada, belirlenen test planına göre test adımları uygulanır, test teknikleri ve araçları kullanılır ve güvenlik açıklarının tespit edilmesi için saldırı senaryoları simüle edilir.

Güvenlik testi uygulaması aşağıdaki adımları içerir:

Güvenlik Testi Ortamının Hazırlanması: Test için uygun bir ortam oluşturulur. Bu, test edilecek sistem veya uygulamanın kurulumu, konfigürasyonu ve gerektiğinde izole bir ortamın oluşturulmasını içerir. Test ortamı, gerçek üretim ortamından ayrı tutulmalıdır.

Test Senaryolarının Yürütülmesi: Belirlenen güvenlik test senaryoları uygulanır. Bu senaryolar, önceden tanımlanan güvenlik açıklarını hedefleyen saldırıları simüle etmek için kullanılır. Örneğin, XSS (Cross-Site Scripting) saldırılarını tespit etmek için kullanıcı girişlerine kötü niyetli kod enjekte edilebilir.

Araçların Kullanımı: Güvenlik testi için çeşitli araçlar kullanılabilir. Bu araçlar, zafiyet tarama araçları, penetrasyon testi araçları, istismar araçları, kod analizi araçları gibi çeşitli kategorilere ayrılabilir. Bu araçlar, test sürecini otomatize etmek, hızlandırmak ve daha kapsamlı sonuçlar elde etmek için kullanılır.

Veri Toplama ve Analiz: Test sırasında elde edilen veriler toplanır ve analiz edilir. Bu veriler, tespit edilen güvenlik açıkları, hatalar, zafiyetler, sistemin tepkileri ve kullanıcı girişleridir. Bu veriler, test sonuçlarının değerlendirilmesi ve raporlama aşamasında kullanılır.

Sorunların Raporlanması: Test sırasında tespit edilen güvenlik açıkları ve zafiyetler ayrıntılı bir şekilde raporlanır. Bu raporlar, ilgili paydaşlara sunulur ve gerekli önlemlerin alınması için kullanılır. Raporlar, tespit edilen sorunların ne kadar ciddi olduğunu, etkilerini ve çözüm önerilerini içermelidir.

Güvenlik testi uygulaması, belirlenen hedeflere ulaşmak ve güvenlik açıklarını tespit etmek için stratejik bir şekilde planlanmalı ve doğru araçlar ve teknikler

kullanılmalıdır. Ayrıca, testin etkili bir şekilde gerçekleştirilmesi için güncel güvenlik standartları ve en iyi uygulamalar göz önünde bulundurulmalıdır.

## 6.4 Bulguların Analizi ve Değerlendirilmesi

Bulguların analizi ve değerlendirilmesi, güvenlik testi sonuçlarının anlamlı bir şekilde yorumlanması ve değerlendirilmesini içerir. Bu aşama, tespit edilen güvenlik açıklarının ciddiyetinin, etkilerinin ve olası risklerin belirlenmesini sağlar.

Bulguların analizi ve değerlendirilmesi aşağıdaki adımları içerebilir:

Güvenlik Açıklarının Sınıflandırılması: Tespit edilen güvenlik açıkları, ciddiyet ve etki düzeylerine göre sınıflandırılır. Bu sınıflandırma, açıkların potansiyel zararını ve sisteme olan etkisini değerlendirmeyi sağlar. Örneğin, kritik seviyedeki açıkların derhal çözülmesi gerekebilirken, düşük seviyedeki açıklar daha düşük bir öncelik düzeyine sahip olabilir.

Risk Değerlendirmesi: Tespit edilen güvenlik açıkları ve zafiyetler, sisteme olan risk düzeyleri değerlendirilir. Bu, açıkların kullanılması durumunda ortaya çıkabilecek riskleri ve potansiyel etkileri belirlemeyi sağlar. Risk değerlendirmesi, açıkların önceliklendirilmesi ve düzeltme çalışmalarının planlanmasında rehberlik eder.

Çözüm Önerileri: Güvenlik açıklarının çözümü için öneriler sunulur. Bu öneriler, tespit edilen açıkların nasıl giderilebileceği, güvenlik önlemlerinin nasıl güçlendirilebileceği veya sistemde yapısal değişikliklerin nasıl yapılması gerektiği gibi bilgileri içerir. Çözüm önerileri, güvenlik açıklarının etkili bir şekilde kapatılmasını ve sistemin daha güvenli hale getirilmesini sağlar.

Raporlama: Bulguların analizi ve değerlendirmesi sonucunda bir rapor oluşturulur. Bu rapor, tespit edilen güvenlik açıkları, risk değerlendirmesi, çözüm önerileri ve diğer önemli bulguları içerir. Rapor, ilgili paydaşlara sunulur ve güvenlik açıklarının giderilmesi ve önlemlerin alınması için yol haritası sağlar.

Bulguların analizi ve değerlendirilmesi aşaması, güvenlik testi sonuçlarının anlamlı bir şekilde değerlendirilmesini ve önlemlerin alınmasını sağlar. Bu aşama, tespit edilen

güvenlik açıklarının ciddiyetinin doğru bir şekilde anlaşılmasını ve uygun tedbirlerin alınmasını sağlar.

## 6.5 Düzeltici Önlemler ve Sürdürülebilirlik

Düzeltici önlemler ve sürdürülebilirlik, güvenlik testi sonuçlarına dayanarak tespit edilen güvenlik açıklarının giderilmesi ve sistemin uzun vadeli güvenliğinin sağlanması için alınan önlemleri ifade eder. Bu aşamada, tespit edilen güvenlik açıklarının düzeltilmesi, güvenlik önlemlerinin güçlendirilmesi ve sürekli olarak güvenli bir durumun sürdürülmesi hedeflenir.

Düzeltici önlemler ve sürdürülebilirlik aşağıdaki unsurları içerebilir:

Güvenlik Açıklarının Düzeltme: Tespit edilen güvenlik açıkları, hatalar veya zafiyetlerin düzeltilmesi için gereken önlemler alınır. Bu, güvenlik açıklarının kaynağının belirlenmesi, düzeltici eylemlerin planlanması ve uygulanması sürecini içerir. Örneğin, bir uygulamadaki SQL enjeksiyonu zafiyetinin giderilmesi için gerekli kod düzeltmeleri yapılabilir.

Güvenlik Politikalarının Güncellenmesi: Güvenlik politikaları ve prosedürler, tespit edilen güvenlik açıklarının göz önüne alınarak güncellenir. Bu, organizasyonun güvenlik politikalarını ve prosedürlerini revize etmeyi ve güvenlik açıklarının tekrarlanmamasını sağlamayı içerir.

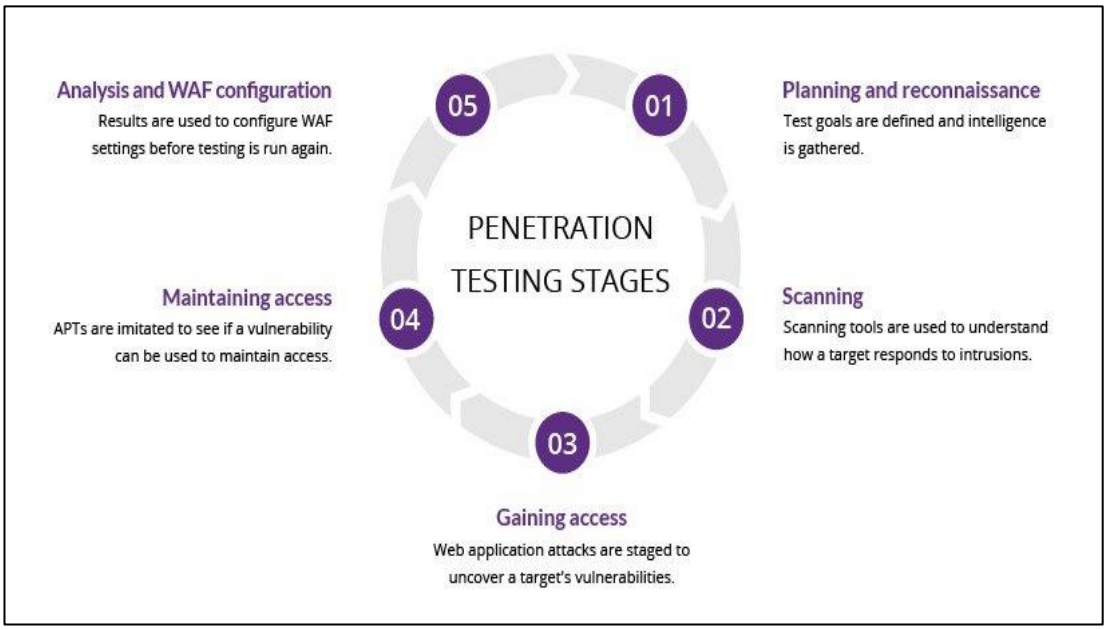
Eğitim ve Farkındalık: Sistem kullanıcıları, geliştiriciler ve diğer ilgili personel için güvenlik eğitimleri ve farkındalık programları düzenlenir. Bu, güvenlik bilincinin artırılması, doğru güvenlik uygulamalarının benimsenmesi ve güvenlik açıklarının önlenmesi için eğitimli bir topluluk oluşturmayı hedefler.

Sürekli İzleme ve İyileştirme: Güvenlik testi sonuçlarının sürekli olarak izlenmesi, yeni güvenlik açıklarının tespiti ve önlenmesi için sürekli bir çaba gerektirir. Güvenlik önlemlerinin etkinliğini değerlendirmek ve iyileştirmek için düzenli olarak güvenlik testleri ve denetimler yapılır.

Sürdürülebilirlik, güvenlik önlemlerinin tek seferlik bir çözüm olmaktan ziyade sürekli olarak uygulanmasını ve güncellenmesini gerektirir. Güvenlik açıklarının

düzeltilmesi ve sistemin güvenli bir durumda tutulması için sürekli bir çaba sarf edilmelidir.

Düzeltici önlemler ve sürdürülebilirlik, güvenlik testi sonuçlarının değerlendirilmesi ve önlemlerin alınması sürecinin kritik bir parçasını oluşturur. Bu aşama, tespit edilen güvenlik açıklarının düzeltilmesini sağlar ve sistemin güvenliğini sürdürülebilir bir şekilde sağlamak için gereken adımları atar.



# Bölüm 7

# Security Testing Best Practices

Bu bölümde, etkili Security Testing için en iyi uygulamalar incelenir. Test ortamının hazırlanması ve yönetimi, test verilerinin gizliliği ve güvenliği, Security Testing senaryolarının gerçekçiliği ve kapsamlılığı, Security Testing'in sürekli iyileştirilmesi gibi konulara odaklanılır.

## 7.1 Test Ortamının Kurulması ve Yönetimi

Test ortamının kurulması ve yönetimi, güvenlik testlerinin gerçekleştirileceği uygun bir ortamın oluşturulması ve bu ortamın etkili bir şekilde yönetilmesini içerir. Test ortamı, gerçek üretim ortamından bağımsız olarak oluşturulan ve kontrol edilen bir ortamdır. Bu ortam, güvenlik testlerinin gerçekleştirilmesi için gerekli olan sistemler, ağlar, yazılımlar ve verilerin bulunduğu bir yapıyı kapsar.

Test ortamının kurulması ve yönetimi aşağıdaki adımları içerebilir:

İzole Edilmiş Ortam Oluşturma: Güvenlik testlerinin gerçekleştirileceği ortam, gerçek üretim ortamından izole edilmelidir. Bu, test ortamının, gerçek sistemlere veya verilere zarar vermeden güvenlik testlerinin yapılabilmesi için ayrı bir altyapıya sahip olmasını sağlar. Sanal makineler, konteynerler veya ayrı ağ segmentleri gibi teknolojiler kullanılarak izolasyon sağlanabilir.

Gerçekçi Veri Kullanımı: Test ortamında gerçekçi veriler kullanılması önemlidir. Bu, testlerin gerçek dünya senaryolarını yansıtması ve gerçek sisteme benzer koşullar altında gerçekleştirilmesini sağlar. Ancak, gizlilik ve uyumluluk gereksinimlerine uygun şekilde verilerin gizliliğinin ve korunmasının sağlanması önemlidir. Anonimleştirme veya sahte veri kullanımı gibi teknikler tercih edilebilir.

Test Ortamının Yönetimi: Test ortamı, düzenli olarak güncellenmeli ve yönetilmelidir. Bu, sistem bileşenlerinin ve yazılımlarının güncel ve güvenli bir şekilde tutulmasını sağlar. Aynı zamanda, test ortamının izin verilen kişiler tarafından erişilebileceği ve gerektiğinde hızlı bir şekilde geri yüklenebileceği bir yedekleme ve kurtarma stratejisi oluşturulmalıdır.

Çevre Simülasyonu: Test ortamı, gerçek üretim ortamını mümkün olduğunca doğru bir şekilde simüle etmelidir. Bu, farklı ağ yapılandırmalarının, güvenlik duvarlarının, erişim kontrollerinin ve diğer bileşenlerin test edilebilmesini sağlar. Ayrıca, test ortamı, gerçek saldırı ve tehdit senaryolarını taklit ederek sistemin savunmasının test edilmesini sağlamalıdır.

Yetkilendirme ve Erişim Kontrolleri: Test ortamına erişim, sadece yetkili kişilere verilmelidir. İlgili personel, test ortamında gerçekleştirilecek güvenlik testleri konusunda uygun yetkilendirmelere sahip olmalıdır. Gerekli güvenlik kontrolleri uygulanmalı ve gerektiğinde izinler düzenlenmelidir.

Test ortamının kurulması ve yönetimi, güvenlik testlerinin etkili bir şekilde gerçekleştirilebilmesi ve güvenlik açıklarının tespit edilip düzeltilmesi için önemlidir. Bu süreç, güvenlik testlerinin doğru bir şekilde uygulanmasını ve sonuçların güvenilirliğini sağlar.

## 7.2 Test Senaryolarının Tasarımı

Test senaryolarının tasarımı, güvenlik testlerinin planlanması ve gerçekleştirilmesi için kullanılan senaryoların oluşturulması sürecidir. Test senaryoları, belirli güvenlik açıklarını veya tehditleri hedefleyen ve test eden ayrıntılı planlar ve adımlar içerir. Bu senaryolar, gerçek dünya senaryolarını yansıtmalı, farklı güvenlik zafiyetlerini kapsamalı ve test edilecek sistemin özelliklerine göre özelleştirilmelidir.

Test senaryolarının tasarımı aşağıdaki adımları içerebilir:

Hedeflenen Tehditlerin Belirlenmesi: Öncelikli olarak, test senaryolarının belirlenmesinde hedeflenen tehditler ve güvenlik açıkları belirlenmelidir. Bu, sistemin maruz kaldığı potansiyel tehditleri ve saldırı vektörlerini anlamak için risk analizi ve

tehdit modellemesi yapmayı içerir. Örnek olarak, XSS, SQL enjeksiyonu, yetkilendirme zaafiyetleri gibi güvenlik açıkları hedeflenebilir.

Senaryo Hedeflerinin ve Kapsamının Belirlenmesi: Her bir test senaryosu için net hedefler belirlenmeli ve senaryonun kapsamı tanımlanmalıdır. Bu, senaryonun test edeceği özellikler, bileşenler veya alanları belirlemeyi içerir. Örneğin, bir web uygulamasında XSS senaryosuyla, kullanıcı giriş alanlarının güvenliğini test etmek hedeflenebilir.

Adımların Tanımlanması: Her test senaryosu için adımlar ayrıntılı olarak tanımlanmalıdır. Bu adımlar, senaryonun nasıl gerçekleştirileceğini ve hangi aksiyonların alınacağını belirlemelidir. Örnek olarak, XSS senaryosunda, kullanıcı giriş alanlarına potansiyel zararlı kodlar ekleyerek XSS saldırısını simüle etmek gibi adımlar olabilir.

Test Verilerinin Hazırlanması: Test senaryolarının etkili bir şekilde gerçekleştirilebilmesi için uygun test verileri hazırlanmalıdır. Bu, senaryonun gerektirdiği verilerin oluşturulması veya temin edilmesini içerir. Örneğin, bir SQL enjeksiyonu senaryosunda, uygun SQL ifadeleriyle test verileri hazırlanmalıdır.

Senaryoların Doğrulama ve Dokümantasyonu: Tasarlanan test senaryoları doğrulanmalı ve gerekli dokümantasyonlar hazırlanmalıdır. Bu, senaryoların doğru bir şekilde çalıştığının ve gereken sonuçları ürettiğinin doğrulandığı anlamına gelir. Ayrıca, senaryoların tekrarlanabilirliğini sağlamak ve sonuçların izlenebilirliğini sağlamak için dokümantasyonlar oluşturulmalıdır.

Test senaryolarının tasarımı, güvenlik testlerinin etkili bir şekilde gerçekleştirilebilmesi için önemlidir. Doğru tasarlanmış senaryolar, sistemin güvenlik açıklarını tespit etmeye ve düzeltici önlemlerin alınmasına yardımcı olur.

## 7.3 Test Sonuçlarının Raporlanması

Test sonuçlarının raporlanması, gerçekleştirilen güvenlik testlerinin sonuçlarının toplanması, analiz edilmesi ve ilgili paydaşlara aktarılması sürecidir. Bu raporlar, test sonuçlarının ayrıntılı bir şekilde sunulmasını, tespit edilen güvenlik açıklarının ve zafiyetlerin açıklanmasını ve önerilen düzeltici önlemlerin belirtilmesini içerir.

Test sonuçlarının raporlanması aşağıdaki unsurları içerebilir:

Rapor Yapısı ve Formatı: Raporlar, okunabilir ve anlaşılır bir yapıya sahip olmalıdır. Raporun formatı, genellikle bir giriş bölümü, metodoloji açıklaması, test sonuçlarının ayrıntılı bir listesi, bulguların analizi, risk değerlendirmesi, önerilen düzeltici önlemler ve sonuçlar bölümlerini içerir. Raporun kullanılacak dil ve terimlerin anlaşılır olmasına dikkat edilmelidir.

Tespit Edilen Güvenlik Açıkları ve Zafiyetler: Rapor, test sırasında tespit edilen güvenlik açıklarını ve zafiyetleri ayrıntılı bir şekilde listelemelidir. Her bir açığın tanımı, etkisi, önemi ve risk seviyesi belirtilmelidir. Rapor ayrıca, açığın nasıl teşhis edildiği ve tekrarlanabilirliği için gerekli adımları içermelidir.

Risk Değerlendirmesi: Tespit edilen güvenlik açıklarının risk seviyesinin değerlendirilmesi önemlidir. Rapor, açıkların potansiyel etkisini, olası saldırı senaryolarını ve sistemin genel güvenlik durumunu göz önünde bulundurarak risk değerlendirmesi yapmalıdır. Bu, açıkların önceliklendirilmesine ve düzeltici önlemlerin planlanmasına yardımcı olur.

Önerilen Düzeltici Önlemler: Rapor, tespit edilen güvenlik açıklarının çözümü için önerilen düzeltici önlemleri içermelidir. Bu önlemler, açıkların giderilmesi veya riskin azaltılması için alınması gereken adımları, güvenlik düzeltmelerini ve en iyi uygulamaları içermelidir. Önerilen önlemler, etkinlikleri artırmak ve gelecekteki saldırıları önlemek için ayrıntılı olarak açıklanmalıdır.

Sonuçlar ve Özet: Raporun son bölümü, test sonuçlarının özetlenmesini ve genel bir değerlendirme yapılmasını içerir. Burada, testin amaçlarına ulaşılıp ulaşılmadığı, genel güvenlik durumu ve önerilerin uygulanması için takip edilmesi gereken adımlar gibi konulara değinilir.

Test sonuçlarının doğru ve ayrıntılı bir şekilde raporlanması, güvenlik açıklarının tespit edilmesi ve düzeltilmesi için önemlidir. Bu raporlar, ilgili paydaşlar tarafından kullanılabilir ve güvenlik önlemlerinin alınması ve sistem güvenliğinin iyileştirilmesi için temel oluşturur.

## 7.4 Güvenlik Testi Sürekliliği ve İyileştirme

Güvenlik testi sürekliliği ve iyileştirme, bir organizasyonun güvenlik önlemlerini sürekli olarak test etme, güvenlik açıklarını belirleme ve sürekli olarak güvenlik durumunu iyileştirme çabalarını içerir. Bu süreç, organizasyonun güvenlik açıklarını hızlı bir şekilde tespit etmesini, düzeltici önlemler almalarını ve gelecekteki saldırıları önlemek için sürekli olarak güvenlik önlemlerini geliştirmesini sağlar.

Güvenlik testi sürekliliği ve iyileştirme aşağıdaki unsurları içerebilir:

Periyodik Güvenlik Testleri: Organizasyonlar, belirli periyotlarda güvenlik testlerini gerçekleştirmelidir. Bu, düzenli olarak güvenlik açıklarını tespit etmek ve sistemlerdeki zayıflıkları belirlemek için yapılan testlerdir. Bu testler, organizasyonun güvenlik durumunu değerlendirmesine ve güvenlik açıklarını düzeltmesine yardımcı olur.

Sürekli İzleme ve Tehdit Değerlendirmesi: Organizasyonlar, sürekli olarak sistemlerini izlemeli ve potansiyel tehditleri değerlendirmelidir. Bu, anormallikleri tespit etmek, saldırı girişimlerini belirlemek ve erken aşamada güvenlik açıklarını saptamak için yapılır. Bu sayede organizasyon, tehditlere karşı daha hızlı ve etkili bir şekilde tepki verebilir.

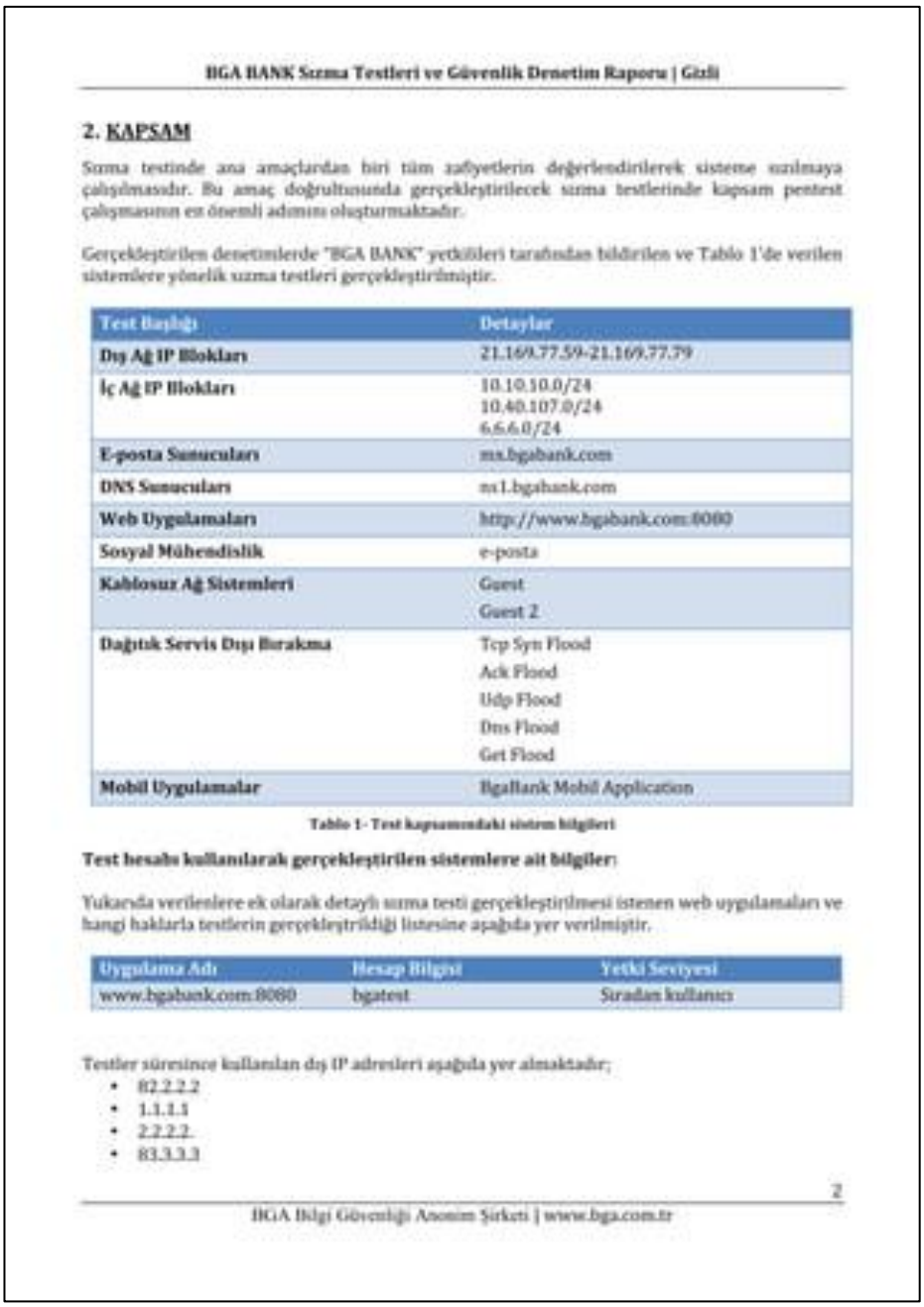
Güvenlik İyileştirme Planları: Güvenlik testlerinin sonuçlarına dayanarak organizasyonlar, güvenlik açıklarını düzeltmek için iyileştirme planları oluşturmalıdır. Bu planlar, tespit edilen güvenlik açıklarının önceliklendirilmesini, düzeltici önlemlerin belirlenmesini ve uygulanmasını içerir. İyileştirme planları, organizasyonun güvenlik durumunu sürekli olarak geliştirmesine yardımcı olur.

Eğitim ve Farkındalık Programları: Organizasyonlar, güvenlik testi sürekliliği ve iyileştirmesini desteklemek için personel eğitimleri ve farkındalık programları düzenlemelidir. Bu, çalışanların güvenlik konularında bilinçlenmelerini, güvenli uygulama ve iş süreçlerine uyumlarını artırmalarını ve güvenlik açıklarını bildirmelerini teşvik eder.

Denetim ve Uyumluluk Kontrolleri: Organizasyonlar, güvenlik testi sürekliliği ve iyileştirmesini desteklemek için düzenli olarak denetimler yapmalı ve uyumluluk

kontrollerini uygulamalıdır. Bu, güvenlik politikalarının ve standartların uygun şekilde uygulandığını, güvenlik süreçlerinin etkin olduğunu ve ilgili düzenlemelere uyum sağlandığını doğrulamayı içerir.

Güvenlik testi sürekliliği ve iyileştirme, organizasyonların güvenlik açıklarını azaltma ve güvenlik seviyelerini sürekli olarak artırma konusunda önemli bir rol oynar. Bu süreç, organizasyonların güvenliği tehdit eden zayıflıkları belirlemelerine, düzeltici önlemler almalarına ve gelecekteki saldırıları önleme konusunda proaktif bir yaklaşım benimsemelerine yardımcı olur.



BGA BANK Sızma Testleri ve Güvenlik Denetim Raporu | Gizli

2. KAPSAM

Sızma testinde ana amaçlardan biri tüm zafiyetlerin değerlendirilerek sisteme sızılmaya çalışılmasıdır. Bu amaç doğrultusunda gerçekleştirilecek sızma testlerinde kapsam pentest çalışmasının en önemli adımını oluşturmaktadır.

Gerçekleştirilen denetimlerde "BGA BANK" yetkilileri tarafından bildirilen ve Tablo 1'de verilen sistemlere yönelik sızma testleri gerçekleştirilmiştir.

| Test Başlığı | Detaylar |
|---|---|
| Dış Ağ IP Blokları | 21.169.77.59-21.169.77.79 |
| İç Ağ IP Blokları | 10.10.10.0/24<br>10.40.107.0/24<br>6.6.6.0/24 |
| E-posta Sunucuları | mx.bgabank.com |
| DNS Sunucuları | ns1.bgabank.com |
| Web Uygulamaları | http://www.bgabank.com:8080 |
| Sosyal Mühendislik | e-posta |
| Kablosuz Ağ Sistemleri | Guest<br>Guest 2 |
| Dağıtık Servis Dışı Bırakma | Tcp Syn Flood<br>Ack Flood<br>Udp Flood<br>Dns Flood<br>Get Flood |
| Mobil Uygulamalar | Bgalbank Mobil Application |

Tablo 1- Test kapsamındaki sistem bilgileri

Test hesabı kullanılarak gerçekleştirilen sistemlere ait bilgiler:

Yukarıda verilenlere ek olarak detaylı sızma testi gerçekleştirilmesi istenen web uygulamaları ve hangi haklarla testlerin gerçekleştirildiği listesine aşağıda yer verilmiştir.

| Uygulama Adı | Hesap Bilgisi | Yetki Seviyesi |
|---|---|---|
| www.bgabank.com:8080 | bgatest | Sıradan kullanıcı |

Testler sürecince kullanılan dış IP adresleri aşağıda yer almaktadır;

- 82.2.2.2
- 1.1.1.1
- 2.2.2.2
- 83.3.3.3

2

BGA Bilgi Güvenliği Anonim Şirketi | www.bga.com.tr

# Kaynaklar

Stuttard, D., Pinto, M. (2007). The Web Application Hacker's Handbook.

Erickson, J. (2003). Hacking: The Art of Exploitation.

Kennedy, D., O'Gorman, J., Kearns, D., Aharoni, M. (2011). Metasploit: The Penetration Tester's Guide.

Schneier, B. (1996). Applied Cryptography: Protocols, Algorithms, and Source Code in C.

Sikorski, M., Honig, A. (2012). Practical Malware Analysis: The Hands-On Guide to Dissecting Malicious Software.

OWASP (Open Web Application Security Project). https://owasp.org/

SANS Institute Reading Room. https://www.sans.org/reading-room/

SecurityTube. https://www.securitytube.net/

Krebs on Security. https://krebsonsecurity.com/

Schneier on Security. https://www.schneier.com/

Coursera. https://www.coursera.org/

edX.. https://www.edx.org/

Cybrary. https://www.cybrary.it/

Offensive Security. https://www.offensive-security.com/

CERT/CC (Computer Emergency Response Team Coordination Center). https://www.cert.org/

MITRE Corporation. https://www.mitre.org/

ENISA (European Union Agency for Cybersecurity). https://www.enisa.europa.eu/

StackExchange Information Security. https://security.stackexchange.com/

Reddit /r/netsec. https://www.reddit.com/r/netsec/

HackerOne Forum. https://hackerone.com/community

# Ekler

# Proje İçerisinden Görüntüler