GAZİ ÜNİVERSİTESİ
Gazi
1926

# TÜRKİYE'DEKİ KAMU KURUMLARININ ANDROİD UYGULAMALARININ GÜVENLİK ZAFİYETLERİNİN ANALİZİ

**Kaya Emre CİBALIK**

**YÜKSEK LİSANS TEZİ**

**ADLİ BİLİŞİM ANA BİLİM DALI**

**GAZİ ÜNİVERSİTESİ**

**BİLİŞİM ENSTİTÜSÜ**

**KASIM 2019**

Kaya Emre CİBALIK tarafından hazırlanan "TÜRKİYE'DEKİ KAMU KURUMLARININ ANDROİD UYGULAMALARININ GÜVENLİK ZAFİYETLERİNİN ANALİZİ" adlı tez çalışması aşağıdaki jüri tarafından OY BİRLİĞİ ile Gazi Üniversitesi Adli Bilişim Ana Bilim Dalında YÜKSEK LİSANS TEZİ olarak kabul edilmiştir.

**Danışman:** Dr. Öğr. Üyesi Cemal KOÇAK

Bilgisayar Mühendisliği, Gazi Üniversitesi

Bu tezin, kapsam ve kalite olarak Yüksek Lisans Tezi olduğunu onaylıyorum.

**Başkan:** Dr. Öğr. Üyesi Hilal KAYA

Bilgisayar Mühendisliği, Ankara Yıldırım Beyazıt Üniversitesi

Bu tezin, kapsam ve kalite olarak Yüksek Lisans Tezi olduğunu onaylıyorum.

**Üye:** Doç. Dr. İbrahim Alper DOĞRU

Bilgisayar Mühendisliği, Gazi Üniversitesi

Bu tezin, kapsam ve kalite olarak Yüksek Lisans Tezi olduğunu onaylıyorum.

Tez Savunma Tarihi: 28/11/2019

Jüri tarafından kabul edilen bu tezin Yüksek Lisans Tezi olması için gerekli şartları yerine getirdiğini onaylıyorum.

.....

Doç. Dr. Aslıhan TÜFEKCİ

Bilişim Enstitüsü Müdürü

## ETİK BEYAN

Gazi Üniversitesi Bilişim Enstitüsü Tez Yazım Kurallarına uygun olarak hazırladığım bu tez çalışmasında;

- Tez içinde sunduğum verileri, bilgileri ve dokümanları akademik ve etik kurallar çerçevesinde elde ettiğimi,
- Tüm bilgi, belge, değerlendirme ve sonuçları bilimsel etik ve ahlak kurallarına uygun olarak sunduğumu,
- Tez çalışmasında yararlandığım eserlerin tümüne uygun atıfta bulunarak kaynak gösterdiğimi,
- Kullanılan verilerde herhangi bir değişiklik yapmadığımı,
- Bu tezde sunduğum çalışmanın özgün olduğunu,

bildirir, aksi bir durumda aleyhime doğabilecek tüm hak kayıplarını kabullendiğimi beyan ederim.

Kaya Emre CİBALIK
28/11/2019

# TÜRKİYE'DEKİ KAMU KURUMLARININ ANDROİD UYGULAMALARININ GÜVENLİK ZAFİYETLERİNİN ANALİZİ

(Yüksek Lisans Tezi)

Kaya Emre CİBALIK

GAZİ ÜNİVERSİTESİ

BİLİŞİM ENSTİTÜSÜ

Kasım 2019

## ÖZET

Son yıllarda akıllı telefon kullanımının hızlı bir şekilde artmasıyla birlikte mobil uygulama pazarı da hızlı bir gelişme gösterdi. Mobil uygulama pazarının iki oyuncusu Android ve iOS işletim sistemleri yaygın bir şekilde kullanılmaktadır. Ancak Android'in hem açık kaynak bir yazılım olması hem de daha rahat bir geliştirme ortamı sunması nedeniyle iOS karşısında kullanım farkını oldukça artırmış durumdadır. Android'in bu kadar yaygın kullanılması zararlı yazılım geliştiricileri ve saldırganlar için de ciddi bir motivasyon kaynağı oluşturmuştur. Bu sebeple Android uygulamalarının güvenliği konusu ciddi bir problem olmuştur. Akıllı telefonların kullanımının yaygınlaşması hassas verilerin mobil dünyaya daha hızlı bir şekilde kaymasına neden olmuştur. Bu tez çalışması, Türkiye'deki kamu kurumlarının mobil uygulamalarının güvenlik zafiyetlerinin tespiti ve bu zafiyetlerin nasıl giderileceği konusunda yapılmış ilk çalışmalardan birisidir. Uygulamaların test edilmesi için bir test modeli önerilmiştir. Bu çalışma kapsamında, seçilen uygulamalar statik ve dinamik analiz süreçlerinden geçirilerek test edilmiştir. Bununla birlikte SSL/TLS bağlantılarının güvenliği konusunda son güncel Android sürümünde analizler yapılmıştır. Yapılan analiz sonuçları detaylı bir şekilde raporlanmıştır. Tez çalışmasında, uygulamalara MiTM (ortadaki adam) saldırıları yapılmış ve zafiyetli olup olmadıkları durumu tespit edilmiştir. Çalışmanın sonucunda, mobil uygulama geliştiricilerinin ve kamu kurumlarının mobil uygulamalarındaki güvenlik açıklarının farkındalığı artıracağı yönünde anlamlı hale getirilmiştir.

Bilim Kodu : 92401

Anahtar Kelimeler :Android, Uygulama Güvenliği, Zafiyet Analizi, Mobil İşletim Sistemleri

Sayfa Adedi : 61

Danışman : Dr. Öğretim Üyesi Cemal KOÇAK

# VULNERABILITY ANALYSIS OF ANDROID APPLICATIONS OF PUBLIC INSTITUTIONS IN TURKEY

(M. Sc. Thesis)

Kaya Emre CİBALIK

GAZI UNIVERSITY

INFORMATICS INSTITUTE

November 2019

## ABSTRACT

With the huge increase in smartphone usage in recent years, the mobile application market has also developed rapidly. The two players of the mobile application market are widely used in Android and iOS operating systems. However, Android is both open source software and offers a more comfortable development environment, which has greatly increased the use of iOS. The wide usage of Android is a serious motivation for malware developers and attackers. For this reason, the security of Android applications has been a serious problem. The wide usage of smartphones has caused sensitive data to shift to the mobile world more quickly. This study is one of the first studies on this area that determination of the mobile application security weakness of public institutions in Turkey and on how to fix these weaknesses. A test model is proposed for testing applications. Within the scope of this study, the selected applications were tested through static and dynamic analysis processes. On the other hand, the latest version of Android has been analyzed about the security of SSL / TLS connections. The results of the analysis are reported in detail. In this thesis, MiTM (man in the middle) attacks were conducted to the applications and it was determined whether they were vulnerable or not. As a result of this study, it was made meaningful that the security gaps in mobile applications developers and public institutions would increase awareness.

Science Code : 92401

Keywords : Android, Application Security, Vulnerability Analysis, Mobile Operating Systems

Page Number : 61

Supervisor : Dr. Cemal KOÇAK

# TEŞEKKÜR

Tezin hazırlanması aşamasında bana her türlü desteği veren danışmanım Sayın Dr. Cemal KOÇAK' a, yüksek lisans ve tez çalışmam boyunca göstermiş olduğu anlayış ve sabır ile bana her zaman destek olan eşime, desteklerini esirgemeyen mesai arkadaşlarıma, yüksek lisans yapmamız için her daim bizi teşvik eden Sağlık Bilgi Sistemleri Genel Müdürü Sayın Dr. Mustafa Mahir ÜLGÜ' ye ayrıca sonsuz teşekkürlerimi sunmayı bir borç bilirim.

# İÇİNDEKİLER

**Sayfa**

ÖZET ..... iv

ABSTRACT ..... v

TEŞEKKÜR ..... vi

İÇİNDEKİLER ..... vii

ÇİZELGELERİN LİSTESİ ..... x

ŞEKİLLERİN LİSTESİ ..... xi

RESİMLERİN LİSTESİ ..... xiii

SİMGELER ve KISALTMALAR ..... xiv

1.GİRİŞ ..... 1

2.ANDROID İŞLETİM SİSTEMİ ve MİMARİSİ ..... 7

2.1.Android İşletim Sisteminin Tarihi ..... 7

2.1.1.Android 1.0 öncesi versiyonların gelişimi ..... 7

2.1.2.Android 1.0 ve sonraki versiyonları ..... 8

2.2.Android İşletim Sistemi Mimarisi ..... 10

2.2.1.Güç yönetimi ..... 12

2.2.2.Linux çekirdeği ..... 12

2.2.3.Donanım soyutlama katmanı ..... 12

2.2.4.Android çalışma zamanı ..... 12

2.2.5.Yerel C/C++ kütüphaneleri ..... 13

2.2.6.Java uygulama programlama ara yüzü çatısı ..... 13

2.2.7.Sistem uygulamaları ..... 14

2.3.Android Uygulama Zafiyetleri ..... 14

2.3.1.Ağ bağlantılarının güvenliği ..... 14

2.3.2.Depolama erişiminin güvenliği ..... 15

2.3.3.Webview üzerinde ataklar ..... 16

**Sayfa**

2.3.4.SQLite veri tabanı güvenliği ..... 16
2.3.5.İzin yönetimi ..... 17
2.3.6.Komut çalıştırma zafiyeti ..... 19
3.GERÇEKLEŞTİRİLEN ZAFİYET TESTİ ÇALIŞMASI ..... 21
3.1.Android Uygulama Yapısı ..... 23
3.2.Test Aşamasında Kullanılan Araçlar ..... 23
3.2.1.Burp suite ..... 23
3.2.2.Mitmproxy ..... 28
3.2.3.Kali linux ..... 30
3.2.4.AndroBugs framework ..... 30
3.2.5.MobSF ..... 30
3.2.6.Apktool ..... 30
3.2.7.Android debug bridge (Adb) ..... 30
3.2.8.Android studio ..... 31
3.2.9.Drozer ..... 34
3.3.Test Aşamaları ..... 35
3.3.1.Statik analiz ..... 43
3.3.2.Dinamik analiz ..... 44
3.3.3.Test ortamı ..... 44
3.3.4.Test edilen mobil uygulama veri kümesi ..... 45
4.TEST SONUÇLARI ve ANALİZ ..... 47
4.1.Statik Analiz Test Sonuçları ..... 47
4.2.Dinamik Analiz Test Sonuçları ..... 49
4.3.Tartışma ..... 54
5.SONUÇ VE ÖNERİLER ..... 55
KAYNAKLAR ..... 57

**Sayfa**

ÖZGEÇMİŞ..... 61

# ÇİZELGELERİN LİSTESİ

**Çizelge** **Sayfa**

Çizelge 2.1. Android sürüm isimlendirmeleri..... 8

Çizelge 2.2. Android versiyonlarının linux çekirdeğine göre karşılıkları..... 9

Çizelge 3.1. Araçların zafiyetleri test koşulları..... 43

Çizelge 3.2. Uygulamaların versiyon ve indirilme sayıları..... 45

Çizelge 4.1. Statik analiz test sonuçları..... 47

Çizelge 4.2. Sertifika pinleme test sonuçları..... 48

Çizelge 4.3. Tehlikeli olabilecek izin istekleri..... 48

Çizelge 4.4. Dinamik analiz test sonuçları..... 49

# ŞEKİLLERİN LİSTESİ

**Şekil** **Sayfa**

Şekil 2.1. Android 1.0 öncesi versiyonların görünümü..... 8
Şekil 2.2. Android 10'dan bir görünüm..... 10
Şekil 2.3. Android işletim sistemi mimari yapısı [24]..... 11
Şekil 2.4. Android 5.0 öncesi izin modeli [31]..... 18
Şekil 2.5. Android yeni izin modeli [31]..... 19
Şekil 3.1. Önerilen test modeli..... 22
Şekil 3.2. Apk dosyası içeriği..... 23
Şekil 3.3. Apktool ile apk dosyasının kullanımı..... 23
Şekil 3.4. Burp ana ekranı görüntüsü..... 24
Şekil 3.5. Burp proxy ayarlarına giriş..... 25
Şekil 3.6. Burp options ve proxy aktif edilmesi..... 26
Şekil 3.7. Proxy IP ayarlarının yapılandırılması..... 27
Şekil 3.8. Android cihazı proxy ayarlarının yapılması..... 28
Şekil 3.9. Mitmproxy çalışma mantığı [43]..... 29
Şekil 3.10. Adb devices örnek çıktısı..... 31
Şekil 3.11. Adb shell örnek bağlantı..... 31
Şekil 3.12. Avd manager ana ekranı..... 32
Şekil 3.13. Donanım seçim adımı..... 32
Şekil 3.14. İşletim sistemi seçim ekranı..... 33
Şekil 3.15. Avd özet görüntüsü adımı..... 33
Şekil 3.16. Androbugs apk testi komut ekranı..... 35
Şekil 3.17. Androbugs test raporu..... 36
Şekil 3.18. Mobsf ilk çalıştırılma ekranı..... 36
Şekil 3.19. Mobsf apk dosyası yükleme web arayüzü..... 37
Şekil 3.20. Mobsf analiz aşaması..... 38

**Şekil** **Sayfa**

Şekil 3.21. Mobsf analiz test raporu çıktısı..... 38

Şekil 3.22. config dosyasının varlığının test edilmesi sonuç çıktısı..... 39

Şekil 3.23. Uygulama-2 webview zafiyeti statik analiz..... 40

Şekil 3.24. Uygulama-2 aktivitelerin tespiti..... 40

Şekil 3.25. Uygulama-2 aktivitenin script ile çalıştırılması..... 40

Şekil 3.26. Uygulama-8'in ssl trafiğinin vekil sunucuyla yakalanması..... 41

# RESİMLERİN LİSTESİ

**Resim** **Sayfa**

Resim 3.1. Drozer ajanı ana ekran görüntüsü ..... 34

Resim 3.2. Port yönlendirme adımı ..... 35

Resim 4.1. Uygulama-1'e yapılan MITM saldırısı ..... 51

Resim 4.2. Drozer ilk konsol bağlantısı ..... 52

Resim 4.3. Drozer uygulama listelerinin görülmesi ..... 52

Resim 4.4. Zararlı kodun hazırlanması-1 ..... 53

Resim 4.5. Zararlı kodun hazırlanması-2 ..... 53

Resim 4.6. Zararlı kodun uygulamaya enjekte edilmesi ..... 54

Resim 4.7. Zararlı kodun başarısız olduğunun görülmesi ..... 54

# SİMGELER ve KISALTMALAR

Bu çalışmada kullanılan kısaltmalar, açıklamaları ile birlikte aşağıda sunulmuştur.

| Kısaltmalar | Açıklama |
|---|---|
| **ADB** | Android hata-ayıklama köprüsü (Android debug bridge) |
| **API** | Uygulama programlama ara yüzü |
| **APK** | Android paketi (Android package) |
| **ART** | Android çalışma zamanı (Android run time) |
| **AVD** | Android sanal makineler (Android virtual devices) |
| **CN** | Ortak isim (Common name) |
| **DEX** | Dalvik çalıştırılabilir dosya (Dalvik executable) |
| **ES** | Gömülü sistemler (Embedded systems) |
| **HAL** | Donanım soyutlama katmanı (Hardware abstraction layer) |
| **HPKP** | HTTP açık anahtar iğneleme (HTTP public key pinning) |
| **HTTP** | Hypertext transfer protocol |
| **HTTPS** | Secure hypertext transfer protocol |
| **IDE** | Entegre geliştirme ortamı (Integrated development environment) |
| **iOS** | Apple mobil cihaz işletim sistemi |
| **IoT** | Internet cihazları **(**Internet of things) |
| **IP** | İnternet protokolü (Internet protocol) |
| **JIT** | Tam zamanında (Just in time) |
| **MITM** | Ortadaki adam saldırısı (Man in the middle) |
| **NVD** | Ulusal zafiyet veri tabanı (National vulnerability database) |
| **NDK** | Yerel geliştirme kiti (Native development kit) |
| **OpenGL** | Açık grafik kütüphanesi (Open graphics library) |
| **OpenSSL** | Açık kaynak güvenli soket katmanı (Open secure socket layer) |
| **OSVDB** | Açık kaynak zafiyet veritabanı (Open sourced vulnerability db) |
| **OWASP** | Açık web uygulamaları güvenliği projesi |
| **RFC** | Yorum isteği (Request for comments) |

| Kısaltmalar | Açıklama |
|---|---|
| **SAN** | Konu alternatif adı (Subject alternative name) |
| **SDK** | Yazılım geliştirme kiti (Software development kit) |
| **SEE** | Sqlite şifreleme eklentisi (Sqlite encryption extension) |
| **SMS** | Kısa mesaj servisi (Short message service) |
| **SNI** | Sunucu adı göstergesi (Server name indication) |
| **SSL** | Güvenli soket katmanı (Secure socket layer) |
| **TLS** | Taşıma katmanı güvenliği (Transport layer security) |
| **TCP** | İletim kontrol protokolü (Transmission control protocol) |
| **TÜBİTAK** | Türkiye Bilimsel ve Teknolojik Araştırma Kurumu |
| **XSS** | Siteler arası komut çalıştırma (Cross-site scripting) |

# 1. GİRİŞ

Mobil uygulamaların sayısının ve çeşitliliğinin artmasıyla birlikte, hassas kişisel ve finansal bilgilere mobil ortam kullanılarak erişilmesi ve depolanması durumunu da beraberinde getirdi ve bununla birlikte mobil uygulama kullanımı da artmıştır. Mobil uygulamaların kullanılması için hazırlanan işletim sistemi ortamı olarak Android, iOS ve Windows işletim sistemleri bulunmaktadır. Android ve iOS mobil işletim sistemleri içinde en yaygın kullanılan platform Temmuz 2019 yılı itibariyle %76,08 oranla Android İşletim Sistemidir. Bu istatistik bize aynı zamanda piyasadaki akıllı cihazların (bu kapsama mobil telefon, tablet, IoT cihazları dâhil) büyük bir bölümünün Android kullandığı anlamına geldiğini göstermektedir [1]. Bu sebeple yaygın olarak kullanılan cihazların kötü niyetli saldırganlar tarafından hedef alınması ve yapılan saldırının daha büyük ölçekte bir sorun teşkil etmesi de kaçınılmaz hale gelmektedir. Aynı zamanda zararlı yazılım geliştiricileri ve kötü niyetli saldırgan ve/veya saldırgan gruplar tarafından kullanıcıların günlük hayatta kullandığı faydalı uygulamaların zayıf yönlerini istismar etme eğilimlerinde de artış kaçınılmaz olmaya başlamıştır. Mobil uygulamalarda ne tür zafiyetler [2] olduğuna dair çalışmalar yapılmıştır. Güvenlik zafiyetlerinin sadece tespit edilmesi değil, aynı zamanda da uygulama sahibine konuyla ilgili bilgi verilerek ilgili zafiyetin giderilmesini sağlamak önemlidir. Çünkü açık kalan her zafiyet kötü amaçlı kişiler tarafından istismar edilmesi mümkün olmaktadır [3]. Tez çalışması kapsamında test yapılacak uygulamaların yaygınlığı ve kapsamı göz önünde bulundurulduğunda, çalışmada Kamu kurumlarının vatandaşa hizmet veren ve Google uygulama mağazasından indirilme sayısı da dikkate alınarak seçilen uygulamalar kullanılmıştır. Böylelikle araştırmanın Türkiye'deki kamu kurumlarının mobil uygulamalarının güvenlik durumunu ortaya koyan bir çıktısı olacağı değerlendirilmektedir. Bu araştırma kapsamında kamu kurumlarının vatandaşa hizmet veren uygulamaların zafiyetleri tespit edilecek ve tespit edilen zafiyetlerin nasıl ortadan kaldırılması gerektiği konusunda alınması gereken önlemlerden bahsedilecek. Seçilen Android uygulamasının ne gibi zafiyetleri olduğu tespit edilip detaylı bir şekilde raporlanmıştır. Ulusal Tez Merkezi (tez.yok.gov.tr) ve TÜBİTAK'ın ilgili dizinleri incelendiğinde, ülkemiz özelinde böyle bir çalışmanın yapılmadığı görülmüştür. Bu ihtiyaca binaen ilgili çalışmanın yapılması gerekliliği ortaya çıkmıştır.

Araştırma kapsamında yapılan incelemelerde bazı ölçütler kullanılacaktır. Bu ölçütler, OWASP ‘ın mobil uygulamaların test edilmesinde kullandığı aşağıda verilen kontrol listesine [4] göre yapılmıştır.

- Veri Depolanması ve Gizlilik
- Kimlik Doğrulama ve Oturum Yönetimi
- Ağ İletişimi
- Kod Kalitesi ve Yapılandırma Ayarları
- Tersine Mühendisliğe karşı dayanıklılık

Yukarıda geçen 5 adet ölçüte göre uygulamalar test edilip, test edilen uygulamaların 5 kategoride tespit edilen kontrol durumlarına göre zafiyeti olup olmadığı tespit edilecektir.

Literatürde, Shezan ve diğ. [5], Bangladeş Hükümeti’nin son günlerde kendi yerel uygulama mağazasına (EATL) mobil uygulama geliştirilmesi için geliştiricileri teşvik kampanyaları düzenlemiş. Bu kapsamda, kısa sürede birçok uygulama geliştirilmiş ve ilgili 3.parti markete uygulamalar kazandırılmıştır. İlgili çalışmada, seçilen 29 uygulamalarda 8 yaygın zafiyetin taranması ve bunların detaylı sonuçları konusunda deneysel bir çalışma yapılmıştır. Bu çalışmada yapılan testler, Depolama Erişimindeki Zafiyet, Webview’deki atak, Android SQLite Veri tabanı şifrelemesi, Uygulama amacındaki zafiyet, Reklam Modülü Analizi, Eskimiş API versiyonu testi ve Hassas API, SMS ve Telefon Aramalarının İzlenmesi, Android Debug Modundaki Ataklar olarak 8 kategoride zafiyet taranmıştır. Uygulamaların statik ve dinamik analizlerinin yapılması aşamasında açık kaynak bazı araçlar kullanılmıştır. Bu araçlar, Androbugs, SandDroid ve Qark’tır. Her bir araç sonucunda bulunan açıklıklar tablo halinde gösterilmiştir. Yapılan çalışmanın sonuç kısmında verilen bilgilere göre, Webview açıklığının taranan birçok uygulamada bulunduğu, Reklam ve Depolama erişimi açıklığının ise diğer yüksek oranlı bulunan açıklıklar olarak listelendiğinden bahsetmiştir.

Umansakar [6], ilgili çalışmada Android işletim sistemindeki çeşitli zafiyetlerin taraması yapılmış. Bu zafiyetleri tespit etmek için, bir framework (çatı) oluşturulup, Android işletim sistemindeki ve uygulamalardaki zafiyetler ve kötücül uygulamaların tespiti yapılmıştır. Bu çatı, OWASP Droid Fusion ile oluşturulmuş, NVD ve OWASP Mobile Top 10’daki prosedürler ve bilgiler kullanılmıştır. OWASP Mobile Top 10, NVD ve OSVDB Veri tabanında listelenen zafiyetler analiz edilmiş ve bu çalışmanın esas konusu bu zafiyetleri analiz etmek ve anlamak ve mevcut tespit etme metotlarının geliştirilmesi için tavsiyeler verilmiştir. Tasarlanan çatıda 3 bölüm bulunuyor; Statik analiz, Zafiyet analizi ve Dinamik

analiz. Statik analiz aşamasında, apk dosyası smali, apktool araçları ile tersine derlenir. Dex dosyası önce jar sonra java dosyasına çevrilir. Daha sonra kod analizcisi, statik olarak kodu, izinleri, API çağırımlarını ve manifest dosyasını analiz eder. Eğer kötücül bir davranış sezilirse, dinamik analiz araçlarına verilir ve sonuçlar izlenir. Çalışma sonucunda test edilen uygulamalarda birçok zafiyet tespit edilmiş. Bu zafiyetler OWASP Top 10'a göre listelenmiştir.

Chan ve diğ. [7] bu çalışmada Android uygulamalarındaki yetki yükseltme ve ön tanımlı olarak tanımlanan izinlerin dışına çıkma saldırılarının tespit edilmesi üzerine bir deneysel çalışmadır. Öneriken zafiyet kontrol sistemi, uygulamanın yetki yükseltme davranışları gösterip göstermediğini kontrol etmektedir. Bu çalışma kapsamında 1038 uygulama indirilmiş, 217 uygulamanın ilgili zafiyete sahip olduğu tespit edilmiştir.

Argudo ve diğ. [8] ilgili çalışmada, Android Uygulamalarındaki Gizlilik Zafiyeti Analizi için bir yöntem önerilmiştir. Metodoloji, OWASP Mobil Güvenlik Projesi, Open Android Güvenlik Değerlendirme Metodolojisi ve bazı iyi uygulamaların birleştirilmesi ile oluşturulmuştur. Önerilen değerlendirme metodu şu bölümlerden oluşmakta; Bilgi toplanması, Giriş ve şifreleme, Bilgi Transferi, Android Uygulama geliştirmesi. Deneysel çalışma birçok çeşitli aracın kullanılması ile zafiyet analizi yapıldığını anlatıyor ve ilgili test Ekvador Cumhuriyetindeki kamu kurumlarının uygulamalarının test edilmesini kapsamaktadır. Bilgi Toplama aşamasında Openvas, Zanti ve Nmap araçları; giriş ve şifreleme aşamasında Zanti ve BurpSuite; Bilgi Transferi aşamasında Wireshark ve Burpsuite; Android Uygulama geliştirme aşamasında ise Apk Extractor, AndroBugs, SanDroid ve Drozer araçları kullanılmıştır.

Kaka ve diğ. [9] İlgili çalışmada, mobil bankacılık uygulamalarının Man in the Middle (Ortadaki adam) saldırılarına karşı zafiyetinin olup olmadığının test edilmesini kapsamaktadır. İlgili çalışma, Hindistan'daki 19 mobil bankacılık uygulamasının Ortadaki adam saldırısına karşı zafiyeti olup olmadığı kontrol edildiğinde, Uygulamaların %90'ının bu zafiyete karşı açık olduğu tespit edilmiştir. Daha da fazlası, bazı mobil uygulamaların kullanıcıların bilgilerini HTTP protokolü üzerinden transfer ettiği tespit edilmiş. Bu çalışma, mobil bankacılık uygulamalarındaki MitM atağının engellenmesi üzerine tavsiyeler içermektedir. Sonuç kısmında ise, zafiyeti olan uygulamaların kullandığı SSL sürümünün zayıflığından kaynaklandığı ve güncel SSL/TLS sürümünün uygulamalarına enjekte edildiğinde herhangi bir sorun olmayacağı değerlendirilmiştir. Android uygulamalarında

SSL/TLS yapılandırma hatalarının neler olduğu ve geliştiricilerin bu hataların nasıl önüne geçebileceğine dair kapsamlı bir çalışma [10] yapılmıştır.

Chang ve diğ. [11], ilgili çalışmada, Android uygulama zafiyetlerini tespit etmek için *Vulnerability Parser* adında statik analiz tabanlı sistem geliştirmiştir. Bu sistem, APK tekrar sıkıştırıcı, Manifest.xml ayrıştırıcı, Zafiyet vektörü ve Dexayrıştırıcı bileşenlerini içermektedir. Zafiyet Vektörü ile farklı zafiyet alanları için kapsamlı tespit kuralları oluşturulmuştur. Bu çalışmada WebView zafiyetinin Dosya erişimi ve Aktarılan Bileşenler üzerine vaka analizi yapılmıştır.

Wu ve diğ. [12], çalışmada literatürde pek yer verilmeyen Android sistem zafiyetleri üzerine bir araştırma yapmışlardır. 2015 ağustos ayından bu yana 3 yıl içerisindeki Android Security Bulletin programındaki 2,179 zafiyetinin ve 1,349 kamuya açık zafiyet giderme yönteminin incelendiği ilk çalışma olduğu belirtilmiştir. Sonuç olarak, zafiyetli modülleri inceleme için 3 analizör, zafiyet giderme modülü, zafiyet deseni modüllerini içeren hiyerarşik veri tabanı yapısı bulunan otomatik analiz çatısı oluşturulmuştur.

Literatürde, SSL/TLS uygunsuz yapılandırılmasıyla ilgili çalışmalar mevcuttur. İlgili çalışmada [13], Android uygulamalarının ağ iletişimi başlattıkları süreçlerde işletim sisteminde kullanılan web tarayıcılarının SSL hata uyarıları oluştuğu durumdaki davranışları ve neden olduğu zafiyetler üzerine saha çalışması gerçekleştirilmiştir. Web tarayıcıların SSL uyarılarının zararlı yazılım ve oltalama saldırılarının önüne geçmek adına önemli olduğu ancak güvenlik uzmanları ve sistem mimarları dışında son kullanıcıların bu uyarıyı görmezden geldiği görülmüştür. Google Chrome'un web tarayıcısının verdiği uyarılardan kullanıcıların %70,2'sinin bu uyarıyı önemsemeyip bağlantıya izin verdikleri tespit edilmiştir. Sonuç kısmında, Tarayıcı güvenlik uyarılarının başarılı bulunduğu, ancak uyarıların verimliliği hususunun demografik yapıya göre değişkenlik gösterdiği belirtilmiştir.

Fahl ve diğ. [14], SSL/TLS yapılandırma hatalarından ilk olarak bu çalışmayla ortaya atmışlardır. Mallodroid adında bir araç geliştirilmiş ve bu araç statik kod analizi ile zafiyetleri tespit etmektedir. Daha sonra, zafiyet barındırdığını tespit ettikleri 100 uygulamanın dinamik analizle testi gerçekleştirilmiş. Bu yöntemin zayıf yanı, statik analiz ile olması muhtemel zafiyetleri tespit ettikten sonra bunu onaylamak için kullanılan dinamik analiz sürecinde manuel bir yöntem kullanmalarıdır. Ayrıca çalışmalarında eski sürüm Android işletim Sistemi kullanmaları da eksiklik olarak değerlendirilmektedir. Diğer bir

çalışmada [15], yazarlar SMV-HUNTER adında SSL/TLS yapılandırma hatalarını tespit eden otomatize analiz aracı geliştirmişlerdir. Uygulamaları ve SSL kütüphanelerinin kapsamını Android 2.3.6, iOS 4.2.1 işletim sistemlerinde gerçekleştirmişler. Bu çalışmada da eski sürüm mobil işletim sistemleri kullanılmıştır. 2019 yılında yapılan bir diğer çalışmada [16], DCDroid adında statik ve dinamik analiz süreçlerini otomatize eden araç geliştirilmiştir. Bu yaklaşımın eksik noktası ise, 2019 yılında en son kullanılan Android sürümünün 9.0 olmasına rağmen, testlerde Android 6.0 sürümünde cihazlar kullanılmış olmasıdır.

Bu tez çalışmasında literatüre yapılacak katkı;

- Kamu kurumlarının Android uygulamalarının testi ve test sonuçlarının anlamlandırılması,
- Android mobil uygulamalarının ağ iletişimde kullandığı SSL/TLS protokolündeki yapılandırma hatalarının tespit edilmesi,
- Android'in yayınlanan ve güncel sürümleri olan Android 8.0 ve Android 9.0 Pie sürümü üzerinde çalışmalar yapılmıştır.

Bu tez çalışmasında, 1. Bölümde tezin konusu, amacı ve önemi konularında genel bilgiler verilmiştir. Tez çalışmasının literatür taraması bu bölümde verilmiştir. Tezin 2. Bölümünde Android işletim sisteminin mimari yapısı, tarihsel gelişimi ve mobil uygulamalarda oluşabilecek zafiyetler, bu zafiyetlerin oluşmaması için alınabilecek önlemlere yer verilmiştir. 3. Bölümde bu tez kapsamında gerçekleştirilen test araçları ve test ortamı ayrıntılı bir şekilde anlatılmıştır. Test yöntemi, kullanılan araçlar ve veri kümesi de bu bölüm içerisinde verilmiştir. 4.bölümde, gerçekleştirilen testin sonuçları ve bu sonuçların analizi detaylı bir şekilde verilmiştir. 5. Bölümde ise, sonuç ve yapılan çalışma konusunda ileriye dönük önerilere yer verilmiştir.

# 2. ANDROID İŞLETİM SİSTEMİ ve MİMARİSİ

Bu bölümde Android işletim sisteminin tarihi ve platform mimarisi hakkında genel bilgiler verilmiştir. İşletim sistemi mimari yapısı, uygulamalarda yapılan konfigürasyon hataları ve bunun sonucunda oluşabilecek güvenlik zafiyetlerinin detaylı anlatımına yer verilmiştir.

## 2.1. Android İşletim Sisteminin Tarihi

2003 yılının Ekim ayında "akıllı telefon" terimi herkes tarafından kullanımı yaygınlaşmadan önce Palo Alto, Kaliforniya da Android Şirketi kuruldu. Şirketin dört kurucusu Rich Miner, Nick Sears, Chris White ve Andy Rubin'dir. Kurucularından Rubin, "*sahibinin konumunu ve tercihlerini daha iyi bilen daha akıllı mobil cihazlar*" geliştireceğini halka duyurdu [17]. 2004'te şirket yatırımcılarına, kameraya yüklenebilen bir Android'in bir bilgisayara kablosuz bağlantı üzerinden bağlantının nasıl olması gerektiğini gösteren adımlar atılmıştır. Kablosuz olarak bağlanılan bu bilgisayar verileri aldıktan sonra kameradaki fotoğrafları bulut mimarisinde çalışan bir sunucuya çevrim içi olarak depolayabileceği bir "Android Veri Merkezi'ne" bağlantısını sağlayacaktır. Ancak işler dijital kamera sektöründe istedikleri gibi gitmedi ve dijital kamera pazarı bu gelişmelere uyumlu çalışmayı kabul etmemiştir. Robin, kameralar için ürettikleri işletim sisteminin aynısını artık mobil cihazlar için kullanılabileceğini açıkca duyurmuştur. Belki de Android için en büyük gelişme olarak sayabileceğimiz durum oluştu ve Google firması 2005 yılında Android'in ana firmasını en az 50 milyon dolar karşılığında satın aldı [17]. Rubin ve şirketin diğer kurucu üyeleri, işletim sistemini geliştirmeye devam etme kararı alarak Google'da yani firmanın yeni sahibinde çalışmaya devam ettiler. Linux'u Android işletim sisteminin için ana bileşen olarak kullanma kararı alındı ve bu aynı zamanda Android' in diğer cep telefonu üreticilerine ücretsiz olarak sunulabilmesinin de önünün açılmasını ifade edilmekteydi.

### 2.1.1. Android 1.0 öncesi versiyonların gelişimi

23 Eylül 2008 tarihinde Android'in ilk ticari versiyonu olan Android 1.0 duyuruldu. Ancak 1.0 versiyonundan önce üzerinde çalışılan ve ticari olmayan sürümleri de mevcuttur.

"m3-rc20a," "m3-rc37a," "m5-rc14." Versiyonlarının görünümü Şekil 2.1' deki gibidir [18]. Bu versiyonlarla ilgili SDK'ların da detayına Android resmi web sitesinden [19] görülebilir ve herkesin erişimine açık durumda bulunmaktadır.

Şekil 2.1. Android 1.0 öncesi versiyonların görünümü

### 2.1.2. Android 1.0 ve sonraki versiyonları

Android 1.0 versiyonu 5 Kasım 2007 yılında beta sürümü olarak yayınlamıştır ve bu sürümün herhangi bir adı yoktur. Ancak Android 1.1 ile birlikte (9 Şubat 2009 tarihi itibariyle) artık versiyonların her birinin bir isimle adlandırılarak geldiği görülmektedir. Örneğin: Android 1.1 versiyonunun "Petit Four" ismiyle geldiğini gözlemlenmektedir. Bu versiyondan sonra ise her Android işletim Sistemi sürümü bir şeker adıyla anılmaktadır [20] ve bu liste Çizelge 2.1'deki gibidir. Bu isimlendirmeler ta ki Android 10 sürümüne kadar devam etmekte ve sonunda Android 10 ile birlikte artık bu isimlendirmeden vazgeçildiği Çizelge 2.1'de görülmektedir.

Çizelge 2.1. Android sürüm isimlendirmeleri

| **Android Sürümleri** | **Sürüm İsmi** |
|---|---|
| Android 4.4 | Kitkat |
| Android 5.0 | Lollipop |
| Android 6.0 | Marshmallow |

Çizelge 2.1(devam)

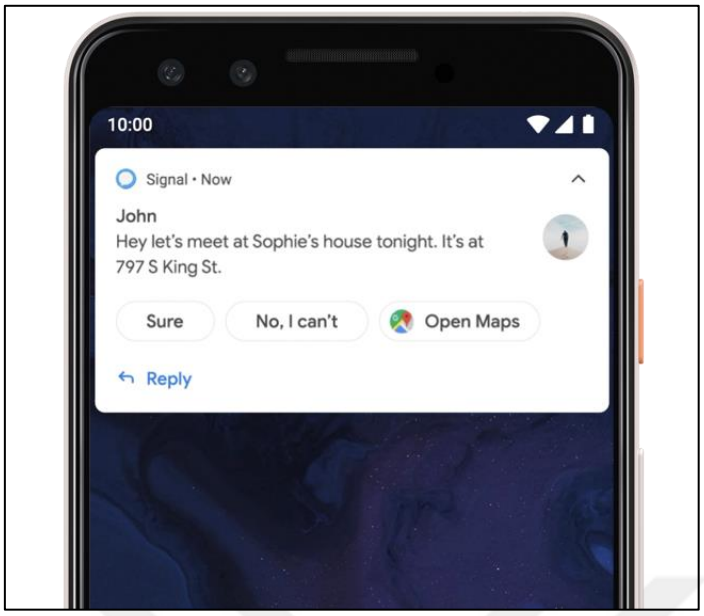
| | |
|---|---|
| Android 7.0 | Nougat |
| Android 8.0 | Oreo |
| Android 9.0 | Pie |
| Android 10 | isimsiz |

Android sürümlerine göre Linux çekirdeği versiyonları Çizelge 2.2'de gösterilmiştir [21].

Çizelge 2.2. Android versiyonlarının linux çekirdeğine göre karşılıkları

| **Android Versiyonu** | | **API** | **Linux Çekirdek** |
|---|---|---|---|
| 1.5 | Cupcake | 3 | 2.6.27 |
| 1.6 | Donut | 4 | 2.6.29 |
| 2.0/1 | Eclair | 5-7 | 2.6.29 |
| 2.2.x | Froyo | 8 | 2.6.32 |
| 2.3.x | Gingerbread | 9,10 | 2.6.35 |
| 3.x.x | Honeycomb | 11-13 | 2.6.36 |
| 4.0.x | Ice Cream Sandwich | 14, 15 | 3.0.1 |
| 4.1.x | Jelly Bean | 16 | 3.0.31 |
| 4.2.x | Jelly Bean | 17 | 3.4.0 |
| 4.3 | Jelly Bean | 18 | 3.4.39 |
| 4.4 | Kit Kat | 19, 20 | 3.10 |
| 5.x | Lollipop | 21, 22 | 3.16.1 |
| 6.0 | Marshmallow | 23 | 3.18.10 |
| 7.0 | Nougat | 24 | 4.4.1 |
| 7.1 | Nougat | 25 | 4.4.1 |
| 8.0 | Oreo | 26 | 4.10 |
| 8.1 | Oreo | 27 | 4.10 |
| 9.0 | Pie | 28 | 4.4, 4.9 ve 4.14 |

Android işletim sisteminin ve akıllı cihazların geldiği mevcut durumu görmek ve ilk sürümüyle karşılaştırma yapabilmek adına aşağıdaki Şekil 2.2'de gösterilmektedir [22].




Şekil 2.2. Android 10'dan bir görünüm

## 2.2. Android İşletim Sistemi Mimarisi

Android mobil cihazlar için Linux çekirdeği tabanlı Google firması tarafından geliştirilen bir yazılım platformudur [23]. Geliştiricilerin Google tarafından geliştirilmiş Java tabanlı kütüphanelerinin olduğu Java programlama dili kullanılarak yönetilebilir kod yazmalarına imkân tanır. Android'in mimari yapısı aşağıdaki Şekil 2.3'de gösterildiği gibidir [24].

Şekil 2.3. Android işletim sistemi mimari yapısı *[24]*

Şekil 2.3'de görüldüğü gibi bu mimarideki temel bileşenler aşağıda verilmiştir [24];

- Güç Yönetimi (Power Management)
- Linux Çekirdeği (Linux Kernel)
- Donanım Soyutlama Katmanı (Hardware Abstraction Layer)
- Yerli C/C++ Kütüphaneleri (Native C/C++ Kütüphaneleri)
- Android Çalışma Zamanı (Android Runtime)
- Java Uygulama Programla Arayüzü Çatısı (Java API Çatısı)
- Sistem Uygulamaları (System Apps)

Bu mimarideki temel bileşenler aşağıda alt başlıklar halinde açıklanmıştır [24].

### 2.2.1. Güç yönetimi

Android Cihazının güç yönetiminin yapıldığı katmandır. Bu katmanda ilgili cihaz için tanımlanan güç parametreleri çalışır. Her cihaz için farklı giriş ve çıkış değerlerinin belirlendiği ve yazılım kaynaklı güç yönetimi veya güç tasarrufu işlemlerinin yapıldığı yerdir [24].

### 2.2.2. Linux çekirdeği

Android platformunun temeli Linux çekirdeğine dayanır. Örnek vermek gerekirse, Android Çalışma Zamanı (Android RunTime), birden fazla iş yönetimi veya düşük seviye bellek yönetimi gibi fonksiyonların altında Linux çekirdeğinin temeli bulunmaktadır. Android, Linux çekirdeği kullanarak bu çekirdeğin kilit ve/veya çok kiritik güvenlik özelliklerinden de yararlanmaktadır. Aynı zamanda sürücü üretimi ve geliştirilmesi için de önemli bir rol oynamaktadır [24].

### 2.2.3. Donanım soyutlama katmanı

Donanım soyutlama katmanı, cihaz donanım yeteneklerini üst seviye Java API çatısına hizmet veren standart ara yüzler sağlar. Bu katman, her biri kamera veya benzeri modüllerin özel bir donanım bileşeni için bir ara yüz uygulayan birden fazla kütüphane modüllerinden oluşur. Çatı programlama ara yüzü cihaz donanımına erişmek için bir istek yaptığında, Android sistemi bu donanım bileşeninin çalışması için özellikli kütüphane modül ve/veya modüllerini yüklemektedir [24].

### 2.2.4. Android çalışma zamanı

Android versiyon 5.0 (API 21) veya daha yüksek versiyon cihazlarda, her uygulama kendi prosesinde ve Android Çalışma Zamanı (ART) kopyasıyla çalışır. Android'in 5.0'dan eski versiyonlarında ART fonksiyonu Dalvik Sanal Makinesi yardımıyla yerine getirilmektedir. ART, düşük bellek alanı üzerinde minimum kaynak kullanımı için optimize edilmiş bir bit kodu dosyası olan DEX dosyalarını çalıştırmak suretiyle bir veya birden çok sanal makinenin çalışması için yazılmıştır. Araç zincirleri oluşturma, örneğin Jack (Java kaynak kodlarını DEX bit kodların a çevirip çalıştırma işlemi) gibi durumlar Android platformunda çalıştırılabilmektedir [24].

ART'in en temel özelliklerinden bazıları şunlardır:

- Çalışma zamanı sırasında (Ahead-of-Time) ve tam zamanında (Just in Time-JIT) derleme
- Optimize edilmiş çöp dosyası koleksiyonu (Garbage Collection)
- Android 9'da (API seviye 28) ve daha yüksek versiyonlarda, Dalvik Executable format (DEX) dosyalarının daha kompakt makine koduna dönüştürülmesi.
- Daha iyi hata ayıklama fonksiyonu
- Android sürüm 5.0'dan önce (API level 21), Dalvik Android çalışma zamanı olarak çalışıyordu. Uygulamanız ART'de çalışıyorsa, Dalvik'te de çalışabilmektedir [24].

### 2.2.5. Yerel C/C++ kütüphaneleri

ART ve HAL gibi birçok çekirdek Android sistem bileşeni ve hizmeti, C ve C++ dilinde yazılmış yerel kütüphaneler gerektiren yerel kodlardan üretilmiştir. Android platformu, bu yerel kütüphanelerin bazılarının işlevselliğini uygulamalara sunmak için Java çatı API' leri sağlar. Örnek vermek gerekirse, uygulamanızda 2D ve 3D grafiklerin çizilmesi ve işlenmesi için destek eklemek için Android çatısının Java OpenGL API'sinden OpenGL ES'ye erişilebilmektedir.

C veya C++ kodu gerektiren bir uygulama geliştiriyorsanız, bu yerel platform kitaplıklarının bazılarına doğrudan yerel koddan erişmek için Android NDK'yi kullanabilmektedir [24].

### 2.2.6. Java uygulama programlama ara yüzü çatısı

Android işletim sisteminin tüm özellik kümeleri Java dilinde yazılmış API'ler aracılığıyla kullanılabilir. Bu API'ler, aşağıdaki bileşenleri içeren çekirdek modüler sistem bileşenlerinin ve hizmetlerinin yeniden kullanımını basitleştirerek Android uygulamaları oluşturmanız için gereken temel dökümünü oluşturmaktadır [24]:

Listeler, ızgaralar, metin kutuları, düğmeler ve hatta gömülebilir bir web tarayıcısı dâhil olmak üzere bir uygulamanın kullanıcı ara yüzünü oluşturmak için kullanabileceğiniz zengin ve genişletilebilir bir *İzleme Sistemi*;

Yerel dizeler, grafikler ve düzen dosyaları gibi kod dışı kaynaklara erişim sağlayan bir *Kaynak Yöneticisi;*

Tüm uygulamaların durum çubuğunda özel uyarıları görüntülemesini sağlayan bir *Bildirim Yöneticisi;*

Uygulamaların yaşam döngüsünü yöneten ve ortak bir gezinme arka yığını sağlayan bir *Etkinlik Yöneticisi;*

Uygulamaların Rehber uygulaması gibi diğer uygulamalardan verilere erişmesini veya kendi verilerini paylaşmasını sağlayan *İçerik Sağlayıcıları;*

Geliştiriciler, Android sistem uygulamalarının kullandığı aynı çatı API'lerine tam erişime sahip bir şekilde yazılım geliştirmektedirler [24].

### 2.2.7. Sistem uygulamaları

Android İşletim Sistemi, e-posta, SMS mesajları, takvimler, internet taraması, kişiler ve daha fazlası için bir dizi çekirdek uygulama ile birlikte gelir. Platforma ön tanımlı olarak yüklü gelen uygulamalar, kullanıcının daha sonradan uygulama marketlerinden özellikle yüklediği uygulamalar arasında farklı bir statüsü yoktur. Böylece kullanıcının sonradan mobil cihazına indirdiği bir başka uygulama varsayılan web tarayıcısı veya diğer varsayılan klavye vb. gibi uygulamaların yerine geçip kullanılabilmektedir. (Sistemin Ayarlar uygulaması gibi bazı istisnalar bu kapsam dışındadır).

Sistem uygulamaları hem kullanıcılar için standart uygulama olarak çalışır, hem de geliştiricilerin kendi uygulamalarından erişebilecekleri belli başlı kabiliyetler sunmasına olanak sağlar. Örneğin, geliştirilecek uygulama SMS mesajı göndermesi gerekiyorsa, bu fonksiyonu geliştiricinin kendisinin tekrar yazması veya oluşturması gerekmez; bunun yerine, uygulamada tanımlı belirtilen bir alıcıya bir mesaj iletmek için ön tanımlı olarak gelen SMS uygulaması çağırıp bu işlev bu şekilde yerine getirebilmektedir [24].

## 2.3. Android Uygulama Zafiyetleri

Bu bölümde Android uygulamalarındaki zafiyetler detaylı bir şekilde ana başlıklar halinde ifade edilecektir. Bu kategoriler; Ağ bağlantılarının güvenliği, Depolama Erişiminin güvenliği, SQLite veri tabanı güvenliği, İzin Yönetimi, Webview üzerinde ataklar, Komut Çalıştırma Zafiyeti şeklinde alt başlıklarda ele alınmıştır.

### 2.3.1. Ağ bağlantılarının güvenliği

Test edilen tüm uygulamalar kullanıcının işlemleri için dış dünya ile bağlantı kurma ihtiyacı bulunmaktadır. Uygulamaların dış dünya ile bağlantı kurarken bu bağlantıların güvenli protokoller ve ayarlar kullanılarak yapılması gerekmektedir. Temel olarak; kullanıcılara ait hassas verilerin görüntülenmesi, değiştirilmesi ve işlenmesi durumlarında bağlantının güvenliği açısından şifreli haberleşme protokolleri kullanılmalıdır. Eğer uygulama bir web isteği göndererek bu işlemlerini gerçekleştiriyorsa bu bağlantılar HTTPS (Hyper Text

Transport Protocol Secure) kullanılarak yapılmalıdır. HTTPS protokolü RFC 2818 [25] RFC dokümanında 2000 yılında tanımlanmıştır. Bağlantının HTTPS olması da güvenli olduğu anlamına gelmemektedir. Güvenli ve en güncel kriptolama tekniklerinin kullanıldığı ve tavsiye edilen protokol sürümleri kullanılmalıdır. Örneğin; Bir SSL zafiyeti olan Heartbleed zafiyetinin Bangladeş özelinde yaptığı etkiler [26] çalışmasında detaylı bir şekilde verilmiştir. Heartbleed zafiyeti 2014 yılında OpenSSL ile ilgili bir yapılandırma zafiyetini ilgilendirmektedir. SSL protokolünde gerek tasarım gerekse protokolün zayıflıkları kullanılarak istismar edildiği durumlar Ca' Foscari University of Venice Üniversitesinde yapılan bir çalışmada [27] sistematik bir şekilde incelenmiştir.

### 2.3.2. Depolama erişiminin güvenliği

Android işletim sistemi, uygulamaların verilerini yazması için kullanabileceği alanları tanımlamıştır. Bu alanlar, harici depolama alanı, dâhili depolama alanı, Paylaşılan ortak tercihler ve SQLite veri tabanı alanlarıdır. Dâhili depolama alanında uygulamanın kendi yarattığı dosyalara sadece uygulamanın kendisi ulaşabilir. Ancak bazı durumlarda geliştiriciler uygulama izinleri konusunda farklı ayarlamalar yapmaktadırlar. MODE_WORLD_WRITABLE ve MODE_WORLD_READABLE şeklinde izin hakkı tanımlanmış dosyalar oluşturulmuşsa, bu dosyalar diğer tüm uygulamalar tarafından ulaşılabilir veya değiştirilebilir duruma gelmektedir. Bu durum güvenlik açığında sebep olmaktadır. Çözümü ise, MODE_WORLD_WRITABLE ve MODE_WORLD_READABLE izin haklarıyla dosya kullanımından kaçınmak, dâhili depolama alanında dosya kullanılacaksa farklı kodlama ve şifreleme mekanizmalarıyla bu dosyaları saklamak şeklinde olmalıdır.

Bazı uygulamalar oluşturdukları dosyaları harici depolama alanına yazmaktadırlar. Harici depolama alanına yazılan dosyalar hem okunabilir hem de yazılabilir şekilde saklanmaktadır. Eğer bu dosyalar içerisinde hassas verileri bulunuyorsa, başkaları tarafından kolayca ele geçirilebilir hale gelmiş durumda olacaktır. Eğer android cihazının root'lama işlemi yapıldıysa, artık cihaz üzerindeki SQLite veri tabanı dahi açık ve herkesin okuyabileceği şekilde kullanıma maruz bırakılmış olacaktır [28]. Bu gibi durumların çözümü, geliştiricilerin hem dosya okuma hem de dosya yazma işlerinde harici depolama alanlarını kullanmaması gerekmektedir. Uygulamalar büyük boyutlu ve hassas veri barındırmayan dosyalarını harici depolama alanlarında saklaması önerilmektedir.

### 2.3.3. Webview üzerinde ataklar

Webview diğer web tarayıcıları gibi Chrome, Mozilla vb. web içeriklerini kullanıcıya ulaştırılmasını ve görüntülenmesini sağlamakla sorumludur. Web içeriklerini kolay bir şekilde görüntülemek için kullanılır. Ancak piyasada kullanılan diğer web tarayıcıları kadar çok özelliğe sahip değildir. Webview genelde uygulamanın kullanıcı ara yüzünde daha yüksek erişim kontrolü ihtiyacı olduğu durumlarda kullanılır. Uygulama içerisine gömülü web içeriklerinin gösterilmesini mümkün kılar. Webview kendi başına zafiyet içeren bir alan değildir. Önemli olan bu Webview içerisinde görüntülenecek olan içeriğin mahiyetidir. Uygulamanın webview içerisinde 3.parti bir içeriğin görüntülenmesi yüksek ihtimalle zarar verici olabilir. Bunun gibi durumlar için Android içerisinde tasarlanmış kumhavuzu işlemleri yürütülmektedir. Ancak Webview işlemi tek işlem (single-process) olarak çalıştığından dolayı kumhavuzu işlemleri etkili bir şekilde kullanılamamaktadır. Bu sebeple, Webview içerisinde zararlı bir içerik çalıştırılmışsa, bu zararlı uygulamanın sahip olduğu tüm haklara da sahip olacaktır. 3.parti içeriklerin Webview'de çalıştırılmadığından kesinlikle emin olmak gerekmektedir. Bu içerikler en azından onaylı ve sterilize edilmiş içerikler olmalıdır.

Webview tarafında karşılaşılabilecek en kritik zafiyetler [28], SQL Enjeksiyonu, Siteler arası komut dosyası çalıştırma (XSS) şeklindedir. Bu tür saldırılar ciddi zararlı saldırılardır. Özellikle bu tür saldırılar, uygulamanın veya web sayfasının kullanıcı girdilerini düzgün bir şekilde kontrol etmemesinden kaynaklanmaktadır. Bu tür saldırıların çözümü, hiçbir kullanıcı girdisine güvenmeden, hepsinin kontrol edilip, sterilize edilmesidir.

### 2.3.4. SQLite veri tabanı güvenliği

Uygulamaların yerel verilerinin depolanması için en çok kullanılan alan Android içerisindeki SQLite veri tabanıdır. Kullanılan cihazların bir şekilde bozulması veya kısmen zarar görmesi durumlarında veri tabanı vasıtasıyla geri döndürülme olasılığı bulunmakta ve bu verilerin kurtarılması adına ciddi bir önem arz etmektedir. Eğer cihaz root işlemine tabi tutulduysa veya hata ayıklama modu aktif edildiyse, SQLite veri tabanındaki veriler çok kolay bir şekilde ADB aracı vasıtasıyla yetkisiz bir şekilde görüntülenebilir, elde edilebilir. Bu zafiyete örnek bir durum [30] çalışmada verilmiştir. Bu tür bir zafiyet, kimlik hırsızlığı, sahtekârlık, itibar zedelenmesi gibi durumlara yol açabilmektedir. Bu tarz durumların çözümü ise, SQL veri tabanına güvenlik için bir katman daha eklenerek, SQLite Encyrption Extension(SEE) yapısının aktive edilmesi gerekmektedir.

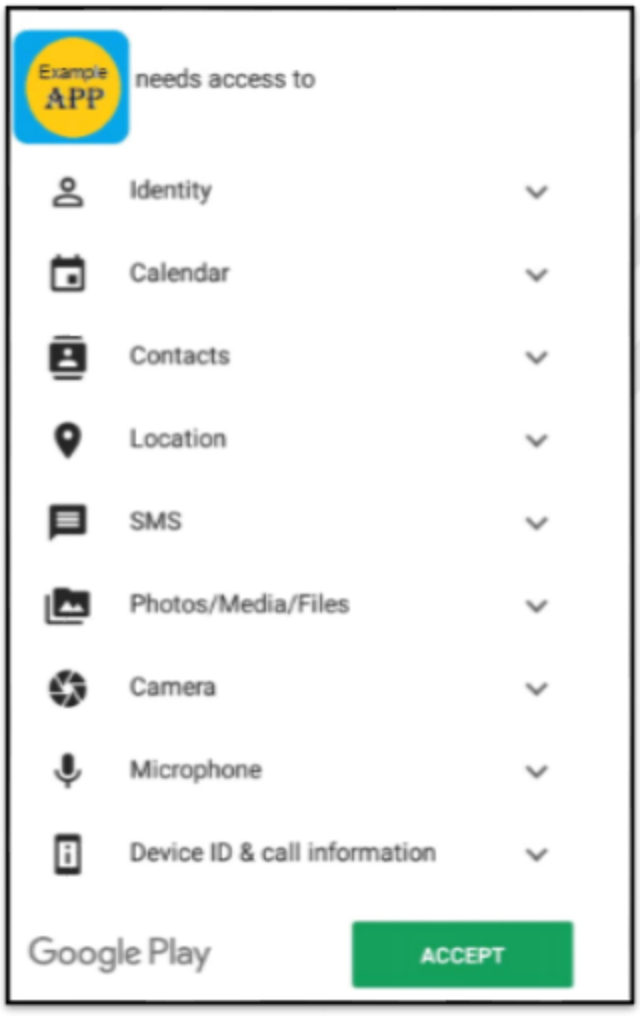
### 2.3.5. İzin yönetimi

Android uygulamaları, APK dosyasında uygulama geliştiricisinin önceden tanımlamış olduğu izinleri talep etmekte ve kullanmaktadır. Bu izin isteklerinin tanımlanmış olduğu dosya AndroidManifest.xml dosyasıdır. Örnek bir izin xml dosyası aşağıda gösterilmiştir.

```
<?xml version="1.0" encoding="utf-8" standalone="no"?><manifest
xmlns:android="http://schemas.android.com/apk/res/android"
android:hardwareAccelerated="true" package="tr.gov.saglik.ekipportal"
platformBuildVersionCode="7" platformBuildVersionName="1.0.5">
   <supports-screens android:anyDensity="true" android:largeScreens="true"
android:normalScreens="true" android:resizeable="true" android:smallScreens="true"
android:xlargeScreens="true"/>
   <uses-permission android:name="android.permission.INTERNET"/>
   <uses-permission
android:name="android.permission.ACCESS_NETWORK_STATE"/>
   <uses-permission android:name="android.permission.WAKE_LOCK"/>
   <uses-permission
android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
   <uses-permission android:name="android.permission.READ_PHONE_STATE"/>
   <uses-permission
android:name="android.permission.READ_EXTERNAL_STORAGE"/>
   <uses-permission android:name="android.permission.RECORD_AUDIO"/>
   <uses-permission android:name="android.permission.CALL_PHONE"/>
   <uses-feature android:name="android.hardware.camera"/>
   <uses-permission android:name="android.permission.CAMERA"/>
   <uses-permission
android:name="com.sec.android.provider.badge.permission.READ"/>
   <uses-permission
android:name="com.sec.android.provider.badge.permission.WRITE"/>
   <uses-permission
android:name="com.htc.launcher.permission.READ_SETTINGS"/>
<uses-permission
android:name="com.htc.launcher.permission.UPDATE_SHORTCUT"/>
   <uses-permission
android:name="com.sonyericsson.home.permission.BROADCAST_BADGE"/>
android:name="com.anddoes.launcher.permission.UPDATE_COUNT"/>
   <uses-permission
android:name="com.majeur.launcher.permission.UPDATE_BADGE"/>
   <uses-permission
android:name="com.huawei.android.launcher.permission.READ_SETTINGS"/>
   <uses-permission
android:name="com.huawei.android.launcher.permission.WRITE_SETTINGS"/>
</application>
</manifest>
```

Örnek xml dosyasında android.permission ile başlayıp devam eden satırlarda uygulamanın izin istekleri yer almaktadır.

Android işletim sisteminde izin yönetimi durumunu Android 5.0 sürümü (API 22) öncesi ve sonrası olarak değerlendirmek daha doğru olacaktır. Çünkü Android 5.0 ve öncesi sürümlerinde yüklemek istenilen uygulamanın istediği tüm izinleri vermek durumundaydınız. Aksi hale uygulama yükleme işlemi başarısızlıkla sonuçlanmaktaydı [31]. Bu durumu Şekil 2.4' de verilmiştir.

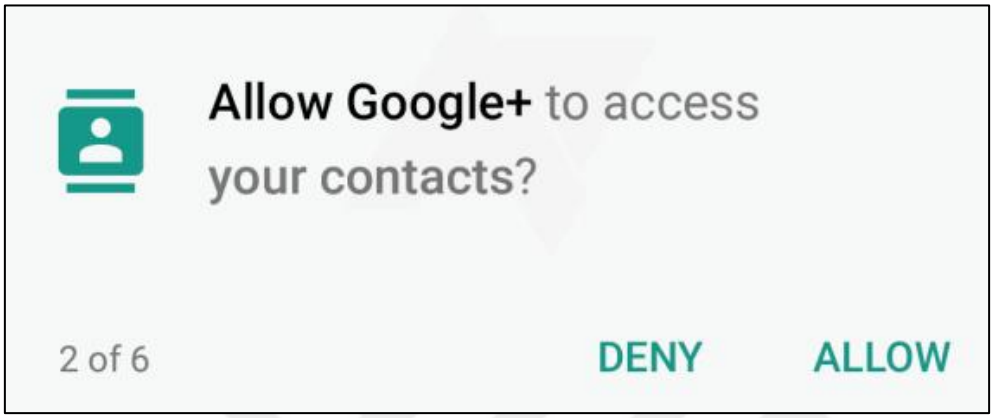



Şekil 2.4. Android 5.0 öncesi izin modeli *[31]*

Android 5.0 sürümü sonrası, Android M izin modeli yapısı getirildi. Bu izin modelinde uygulama çalışma zamanında ihtiyacı olan izinleri talep etmekte, kullanıcı eğer isterse sadece çalışma zamanındaki izinleri uygulamaya verebilmektedir. Bu durumla birlikte, kullanıcı kurulum aşamasında tüm izinleri uygulama vermemekte, uygulama çalışma

zamanında izinlerini ihtiyaç olduğunda kullanıcının karşısına Şekil 2.5' de olduğu gibi çıkarmaktadır.

Arora ve Peddoju, ilgili çalışmasında [32], uygulamaların izin isteklerine ve ağ trafiğine göre bir model oluşturulmuş ve gereksiz, fazla ve kritik izinlerin ne gibi zararlı aktivitelere yol açtığını detaylı bir çalışma gerçekleştirmiştir.




Şekil 2.5. Android yeni izin modeli *[31]*

### 2.3.6. Komut çalıştırma zafiyeti

Uygulamalar harici bir web sayfası ile iletişim kurarken http gibi açık yazı iletişim protokollerini kullanırsa veya HTTPS protokolünü kullanıp yapılandırma adımlarında yapılandırma hataları yapılmışsa veya zafiyetli, eski kalmış SSL/TLS sürümlerini kullanmışsa (Bknz. 2.3.1 Ağ bağlantılarının güvenliği) ortadaki adam saldırılarına maruz kalabilmektedir. Ortadaki adam saldırılarına maruz kalındıktan sonra, kötü niyetli saldırganlar tarafından uzaktan komut çalıştırma saldırıları gerçekleştirilebilmektedir. Bu tarz uzaktan kod çalıştırma ataklarının geniş ölçekte analizleri, Choi ve Kim'in ilgili çalışmasında [33] detaylı bir şekilde yapılmıştır. Bu çalışmada, devre dışı bırakılabilir komut saldırısı atakları, dosya üzerine yazma zafiyetleri ve komut tetikleme noktaları zafiyetleri açısından uygulamalar üzerinde analizler yapılmıştır. 4718 adedi Google uygulama mağazasından, 2967 adedi 3. parti bir uygulama mağazasından ve 1369 adedi önceden tanımlı yüklenmiş sistem uygulamaları seçilmiştir.

# 3. GERÇEKLEŞTİRİLEN ZAFİYET TESTİ ÇALIŞMASI

Test edilecek uygulamaların seçimi (Bknz. Giriş)'nden sonra uygulama dosyasının temin edilmesi gerekmektedir. Android uygulamaları mobil cihaz üzerinde APK uzantılı dosyaların cihaza indirilip kurulması ile gerçekleştirilir. Android uygulamaları APK uzantılı arşiv dosyalarıdır. Test edilecek uygulamaların dosyalarını apkpure.com web sitesinden temini sağlanmıştır. Test yöntemi iki şekilde gerçekleştirilmiştir. Bu yöntemler, statik ve dinamik analiz yöntemleridir.

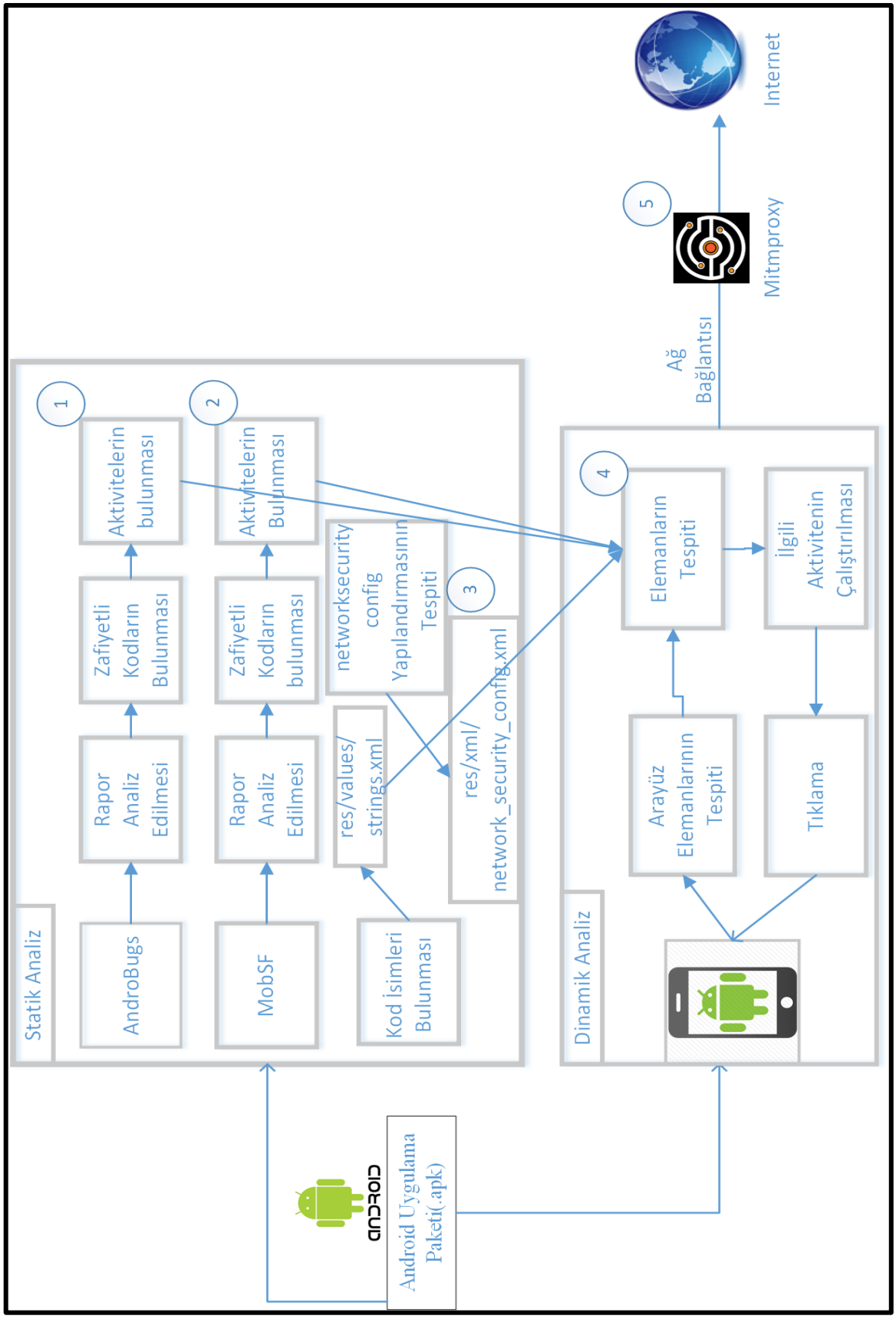
Bu tez çalışmasında, seçilen mobil uygulamaların testi Şekil 3.1'de gösterildiği şekilde önerilen test modeliyle gerçekleştirilmiştir.

Statik analiz aşamasında APK dosyası önce tersine mühendislik yöntemleri ile kaynak koduna getirilmeye çalışılır, tersine çevrilen dosyalar düzgün bir şekilde kategorize edilir ve kategorize edilen dosyalar uygun bir sistematikle analiz edilmeye ve zafiyet olup olmadığı kontrol edilmiştir.

Dinamik analiz aşamasında ise; uygulamanın çalıştırılma zamanında kontrol edilmesi gerekmektedir. Bu aşamada uygulama iki yöntemle edilebilir, birincisi test olarak kullandığınız Android bir cihaz kullanılabilir, ikincisi ise test ortamı oluşturularak sanal bir ortam kullanılabilir. Bu çalışmada Android 9.0 Pie sürümü test edileceğinden ve piyasadan bu sürümdeki cihazların temini ve olası zarar görmesi hesaba katıldığından dolayı sanallaştırma ortamında test işlemleri gerçekleştirilecektir. Ayrıca sanallaştırma ortamı daha rahat ve kolay bir şekilde uygulama silme, ekleme, cihazın yedeğinin alınması vb. durumlar için daha ideal bir ortamdır.

Uygulama apkpure.com sitesinden elde edildikten sonra sanal ortamdaki cihaza kurulumu gerçekleştirilmiştir. Uygulamanın yerel depolama alanında oluşturduğu dosyaları analiz edilmesi, uygulamayı kaldırdıktan sonra da yerel depolama alanında bıraktığı dosyaları analiz edilmesi, uygulamanın ağ trafiğinin izlenmesi, hassas verilerin depolama veya ağ trafiğinde olup olmadığının tespiti işlemleri yapılmıştır.

Zafiyet tespiti çalışmasında test için kullandığımız araçların güncel olması, uygulama paket dosyasının (APK) test için yeterli olması yani kaynak koda ihtiyacı olmaması ve yapılan testleri detaylı bir şekilde raporlayabilmektedir.




Şekil 3.1. Önerilen test modeli

## 3.1. Android Uygulama Yapısı

Android uygulaması apk uzantılı bir arşiv dosyasıdır. Örnek bir arşiv dosyasının içerisindeki dosyalar Şekil 3.2'deki gibidir. Bu arşiv dosyasını, sıradan bir arşiv dosyasının çıkartıldığı gibi çıkarmak mümkün değildir. Çünkü dosyalar kodlanmış ve insan tarafından okunabilir halde değildir. APK dosyasını apktool [33] gibi araçlarla tersine çevirip, okunabilir hale getirilmesi gerekmektedir.

Şekil 3.2. Apk dosyası içeriği

APK dosyasının direkt olarak varsayılan arşiv uygulaması ile çıkarılması durumunda, xml dosyaları ve diğer dosyalar kodlanmış şekilde karşımıza çıkmaktadır. Bu şekilde herhangi bir veri elde edilmemekte ve okunamamaktadır. Bu yüzden apktool ile Şekil 3.2'deki gibi örnek bir uygulama dosyası tersine çevrilmiştir.

```
C:\Users\3MR3>apktool d C:\Users\3MR3\Documents\ekip.apk
I: Using Apktool 2.4.0 on ekip.apk
I: Loading resource table...
I: Decoding AndroidManifest.xml with resources...
S: WARNING: Could not write to (C:\Users\3MR3\AppData\Local\apktool\framework), us
Data\Local\Temp\ instead...
S: Please be aware this is a volatile directory and frameworks could go missing, p
-path if the default storage directory is unavailable
I: Loading resource table from file: C:\Users\3MR3\AppData\Local\Temp\1.apk
I: Regular manifest package...
I: Decoding file-resources...
I: Decoding values */* XMLs...
I: Baksmaling classes.dex...
```

Şekil 3.3. Apktool ile apk dosyasının kullanımı

## 3.2. Test Aşamasında Kullanılan Araçlar

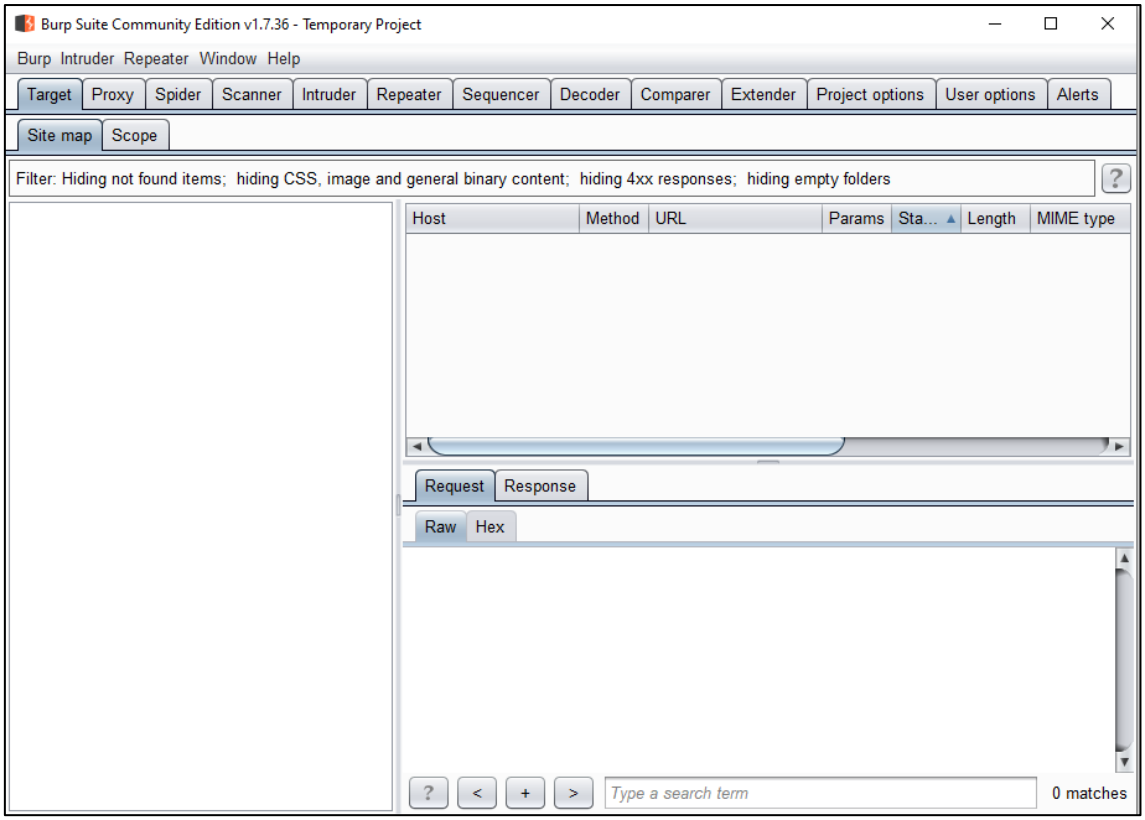
Test aşamasında; Proxy için Burp Suite [35], Mitmproxy [36], Kali Linux sanal makinesi [37], Androbugs Framework [38], MobSF [39], apktool [34], adb [40], Android Studio [41] ve Drozer [42] araçları kullanılmıştır.

### 3.2.1. Burp suite

Burp Suite aracı [35], uygulama içerisinden gerçekleştirilen tüm web isteklerinin yakalanması ve incelenmesi için kullanılmıştır. Burp aracının yapılan web isteklerini

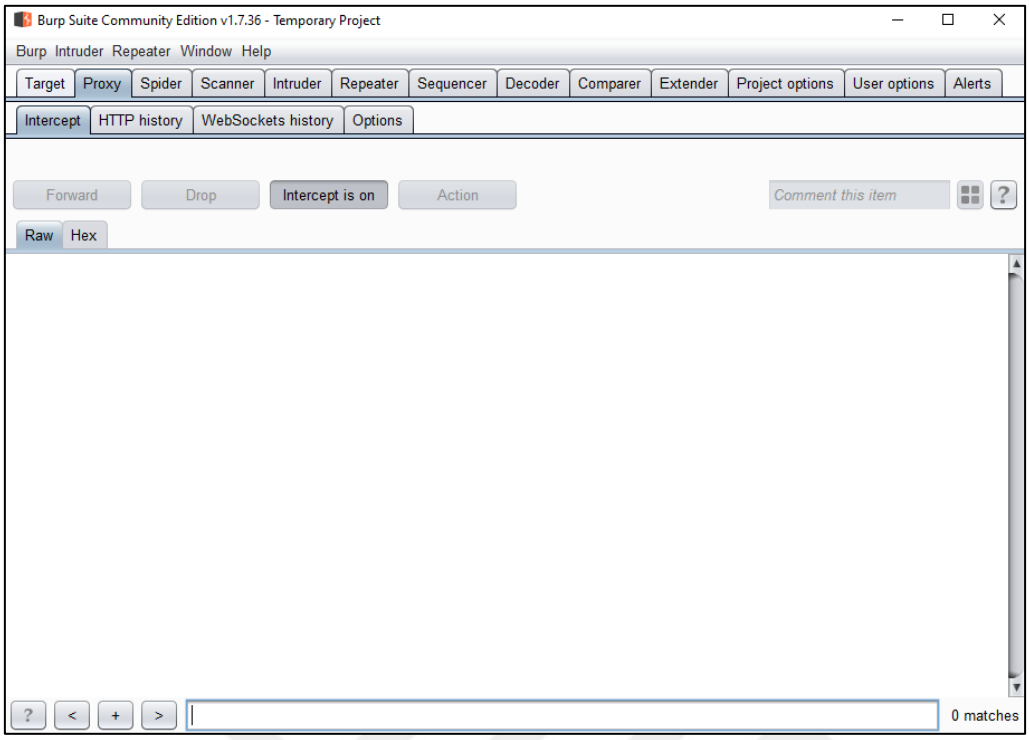
yakalaması için uygun şekilde yapılandırmak gerekmektedir. Burp yapılandırması aşağıdaki adımları takip ederek varsayılan proxy olarak ayarlanmıştır.

1. Burp aracı ilk açıldığında Şekil 3.4'de gösterildiği gibi ana ekran görüntülenmiştir.



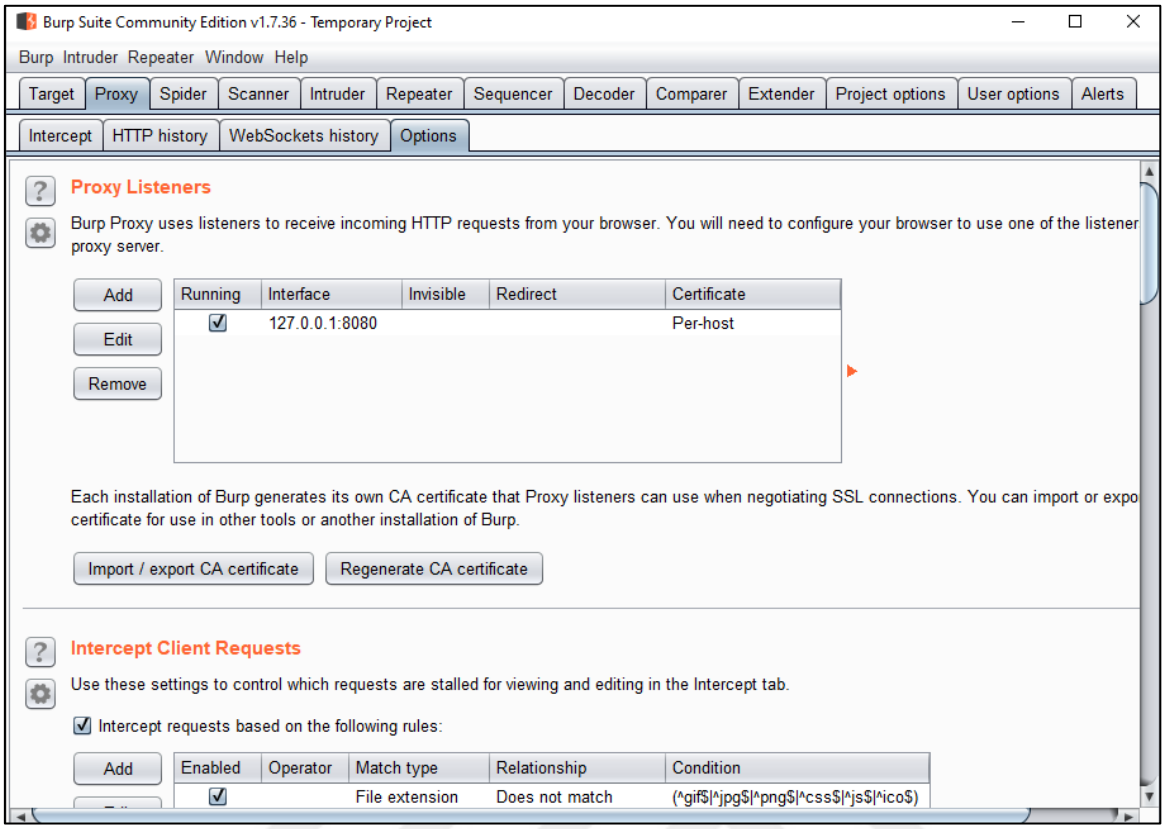

Şekil 3.4. Burp ana ekranı görüntüsü

2. Burp proxy ayarlarını yapmak için Ana menülerden "Proxy" tabına sonrasında ise "Options" tabına tıklanması gerekmektedir. Şekil 3.5 ile ilgili ekran gösterilmiştir.



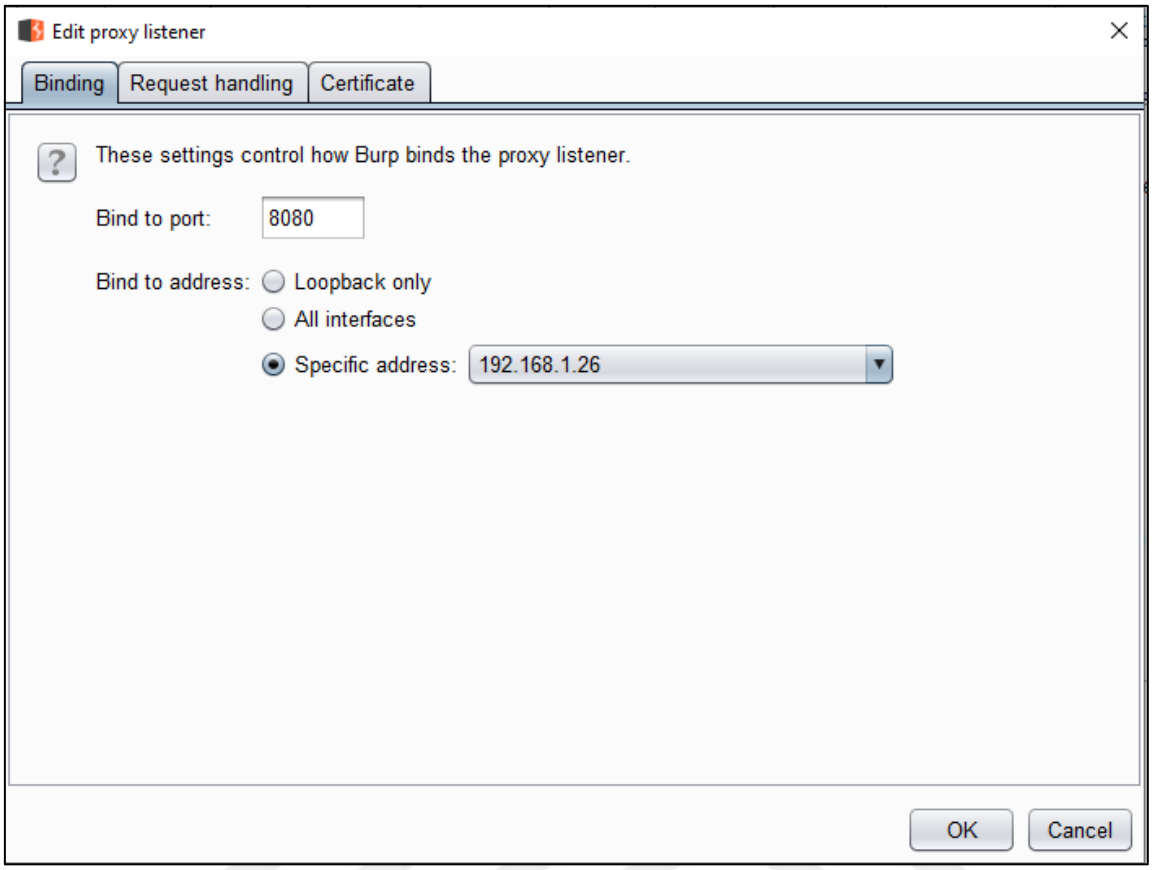

Şekil 3.5. Burp proxy ayarlarına giriş

3. “Options” tabında varsayılan olarak seçili gelen “127.0.0.1:8080” ayarı seçili ve aktif edilmiş bir şekilde olması gerekmektedir. Bu ayarın anlamı, Burp 8080 portundan kendisine gönderilen trafiği dinleyip yakalayacaktır. Bu sayede trafik öncelikle proxy üzerine gelecek, incelenip analiz edildikten sonra hedefe iletilecektir. Şekil 3.6’deki gibi ayarların kontrolü yapılmalıdır.



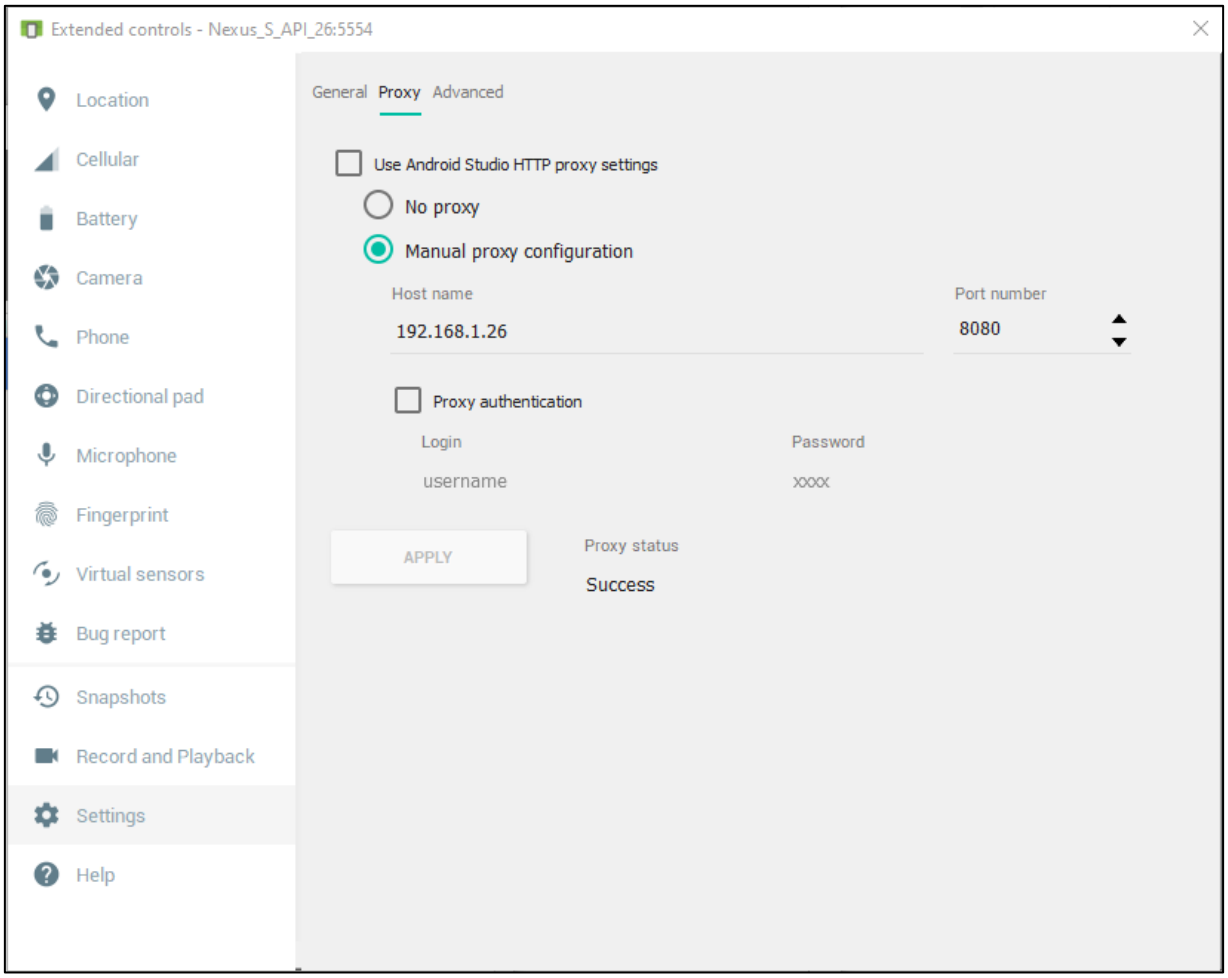

Şekil 3.6. Burp options ve proxy aktif edilmesi

4. Test çalışmamızda bilgisayarımızın Ethernet kartının IP adresi üzerinden haberleşecek olması dolayısıyla IP adresi ayarını ona uygun şekilde yapılması gerekmektedir. Bu konfigürasyonda bilgisayarımızın IP adresi "192.168.1.26" olarak Şekil 3.7 üzerindeki gibi seçilmiştir. Seçim yapıldıktan sonra "OK" ile ayar onaylanmıştır.




Şekil 3.7. Proxy IP ayarlarının yapılandırılması

Ayarlar bu şekilde yapıldıktan sonra proxy yapılandırması tamamlanmıştır. Artık sanallaştırma ortamındaki android cihazının da proxy'e trafiğini yönlendirecek şekilde ayarlanması gerekmektedir. İlgili ayarlar Şekil 3.8'de görüldüğü şekilde olmalıdır.

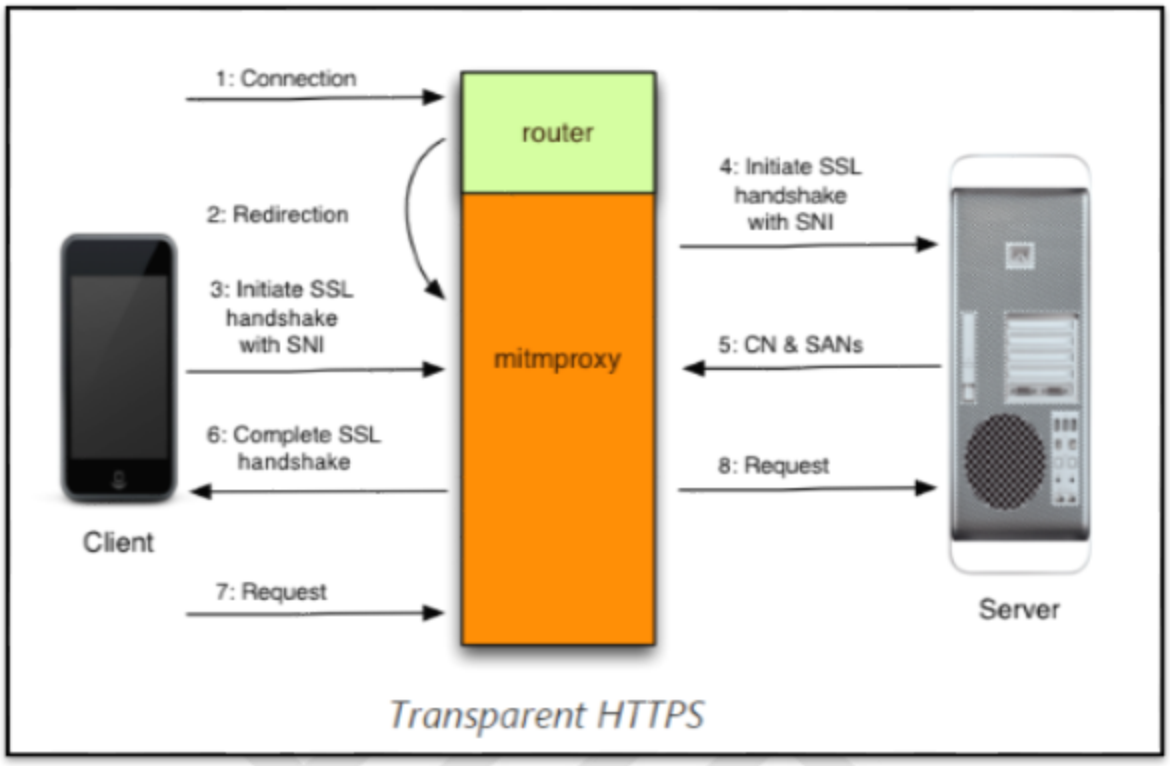



Şekil 3.8. Android cihazı proxy ayarlarının yapılması

İlgili ayarlar resimdeki gibi yapıldıktan sonra "Apply" butonu tıklanarak, sanal cihazın proxy'imiz ile haberleşip haberleşmediği anlaşılmaktadır. "Success" sonucu alınarak testin başarılı geçtiği görülmüştür.

### 3.2.2. Mitmproxy

Mitmproxy aracı [36], uygulamaların SSL/TLS bağlantılarında yapılandırma hataları olup olmadığını tespit etmek amacıyla kullanılmıştır. Burp yerine mitmproxy aracının seçilme nedeni ise, bu aracın havada (on-the-fly) bir şekilde SSL/TLS bağlantısında trafiğin arasına girerek uygulamanın ulaşmak istediği hedefteki web sitesinin SSL sertifikasını anlık olarak istek yapıldığında üretip, bu sertifika ile uygulamayı kandırıp araya girebilmektedir. Temel olarak mimarisi aşağıdaki Şekil 3.9' daki gibidir.




Şekil 3.9. Mitmproxy çalışma mantığı *[43]*

Mitmproxy şu şekilde çalışır;

a. İstemci, sunucuyla bağlantı kurar.
b. Yönlendirici, bağlantıyı aynı ana bilgisayarın yerel bağlantı noktasını dinleyen mitmproxy'ye yönlendirir. Mitmproxy daha sonra orijinal varış yerinin ne olduğunu belirlemek için yönlendirme mekanizmasına danışır.
c. İstemci uzak sunucuyla konuştuğuna inanıp ve TLS bağlantısını başlatır. Bağlandığı ana bilgisayar adını belirtmek için SNI kullanır.
d. Mitmproxy sunucuya bağlanır ve istemci tarafından belirtilen SNI ana bilgisayar adını kullanarak bir TLS bağlantısı kurar.
e. Sunucu, araya girme sertifikası oluşturmak için gereken CN ve SAN değerlerini içeren eşleme sertifikasına yanıt verir.
f. Mitmproxy, araya girme sertifikası oluşturur ve istemcinin TLS el sıkışmasını 3. adımda duraklattığını sürdürür.
g. İstemci isteği, kurulan TLS bağlantısı üzerinden gönderir.
h. Mitmproxy, isteği sunucuya 4. adımda başlatılan TLS bağlantısı üzerinden iletir. Bu şekilde aslında trafik şifreli olduğunu düşünerek son kullanıcı bağlantıyı başlatır.

### 3.2.3. Kali linux

Kali Linux dağıtımı [37], Offensive Security firması tarafından geliştirilen ve penetrasyon testlerinde kullanılmak amacıyla ortaya çıkarılmış tam donanımlı bir Debian dağıtımıdır. Bu dağıtım içerisinde test için kullandığımız AndroBugs Framework ve MobSF araçlarının kurulumları yapılmıştır. Bu araçlar Kali Linux üzerinde kurularak, ilgili testler icra edilmiştir.

### 3.2.4. AndroBugs framework

Androbugs Framework [38], Android uygulamalarının zafiyet testlerinde kullanılan en çok bilinen test aracıdır. Bu araç, uygulamaların zafiyet testlerini kod seviyesinde tespit etmektedir. Uygulama kodları öncelikle tersine mühendislik yöntemleri ile kaynak koduna çevrilmeye çalışılır. Kaynak kodundaki tüm fonksiyonlar zafiyet belirtilerine göre kategorize edilip anlamlandırılır ve sonucunda bir rapor ile çıktısı oluşturulur. Bu aracın en önemli özellikleri arasında, kaynak kod içerisinde zafiyete yol açabilecek kod veya kod parçacıklarının tespiti, çalıştırılması tehlikeli olabilecek komutların çalıştırılıp çalıştırılmadığının kontrolü, uygulamanın kendi güvenliğini kontrol edip etmediğinin tespiti ve uygulamanın internetteki hedef sistemlerine nasıl bağlandığının analizi olarak öne çıkmaktadır. Test sonucunda detaylı rapor vermesi de tercih edilen özellikler arasında bulunmaktadır.

### 3.2.5. MobSF

MobSF [39], Android/ iOS/ Windows platformunda çalışan uygulamalar için kullanılan uygulama zafiyet analizi ve zararlı uygulama analizi yapan otomatize bir araçtır. Bu aracın en önemli özelliklerinden birisi, hem web ara yüzünün bulunması hem de statik ve dinamik olarak verilen uygulamaların test edilebilmesidir. Bu tez çalışmasında MobSF aracı hem statik analiz sürecinde hem de dinamik analiz sürecinde kullanılmıştır.

### 3.2.6. Apktool

Apktool [34], Android uygulamaları, 3.parti uygulamalar ve kapalı/paketlenmiş dosyaların tersine mühendislik ile geri döndürülmesi için kullanılan bir araçtır. Bu tez çalışmasında manuel analiz süreçlerinde bu araçtan faydalanılmıştır.

### 3.2.7. Android debug bridge (Adb)

Android Hata Ayıklama Köprüsü [40], istenilen bir cihazla iletişim kurulmasını sağlayan komut satırı tabanlı bir araçtır. Adb komutu ile uygulamalara komut satırından müdahale

edilebilir, komut gönderilebilir ve hata ayıklama adımlarında kullanılabilir. Bu tez çalışmasında sanallaştırma ortamındaki Android cihazına uygulama yükleme, logların takibi ve uygulamaların işlemler bittikten sonra kaldırılması için kullanılmıştır. Adb komutları şu şekildedir;

*adb devices –l* >> devices komutu şuanda bağlı bulunan sanal veya donanımsal cihazların listesini gösterir. Örnek ekran görüntüsü Şekil 3.10' da verilmiştir. Örnekte görüldüğü gibi emulator-5554 cihazı ulaşılabilir ve bağlı durumdadır.

```
C:\Users\3MR3>adb devices -l
List of devices attached
emulator-5554          device product:sdk_gphone_x86 model:Android_SDK_built_for_x
86 device:generic_x86 transport_id:1
```

Şekil 3.10. Adb devices örnek çıktısı

*adb install ornekuygulama.apk* >> ornekuygulama uygulamasının karşı tarafa yüklenmesini sağlar.

*adb pull /system/uygulama/log.txt* >> pull komutu karşı sistemden dosya almak için kullanılır. Örnek komutta cihazdan log.txt dosyası alınmıştır.

*adb push ornekuygulama.apk /sdcard/ornek.apk* >> push komutu karşı cihaza dosya göndermekte kullanılır. Örnek komutta cihaza ornekuygulama.apk dosyası gönderilmiştir.

*adb shell* >> Shell komutu ile cihazın komut satırına bağlantı gerçekleştirilir. Örnek bağlantı ekranı Şekil 3.11 ile gösterilmiştir.

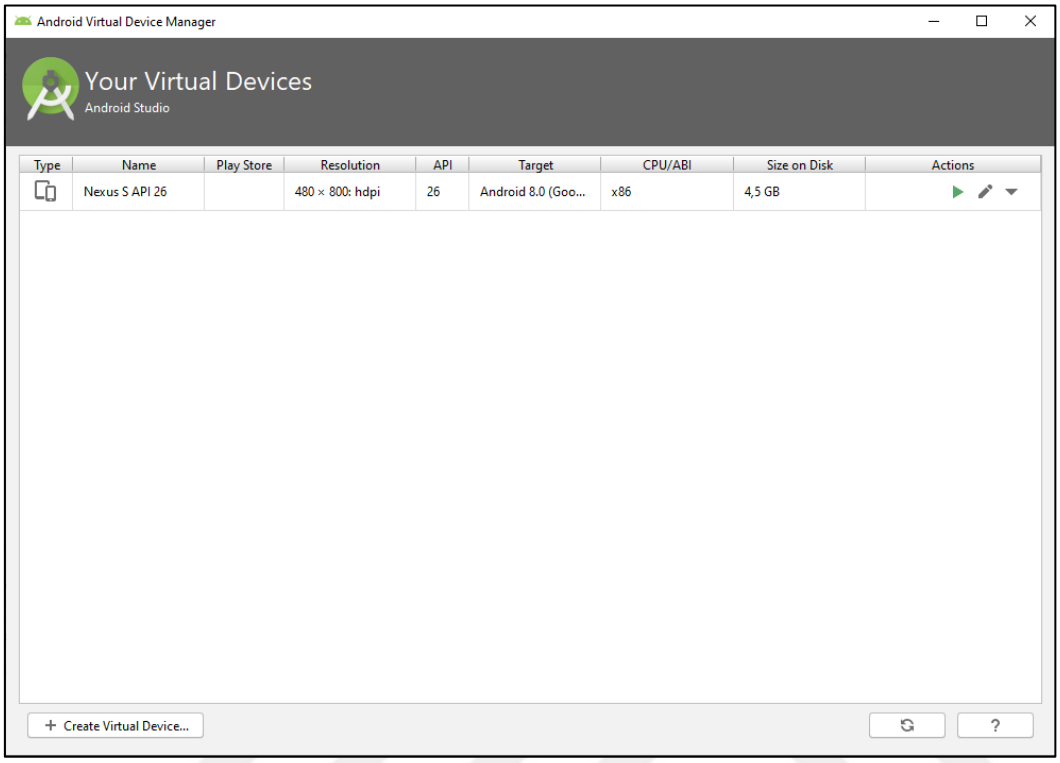
```
C:\Users\3MR3>adb shell
generic_x86:/ $ id
uid=2000(shell) gid=2000(shell) groups=2000(shell),1004(input),1007(log),1011(adb),1015(sdcard_rw),1028(sd
card_r),3001(net_bt_admin),3002(net_bt),3003(inet),3006(net_bw_stats),3009(readproc) context=u:r:shell:s0
```

Şekil 3.11. Adb shell örnek bağlantı

### 3.2.8. Android studio

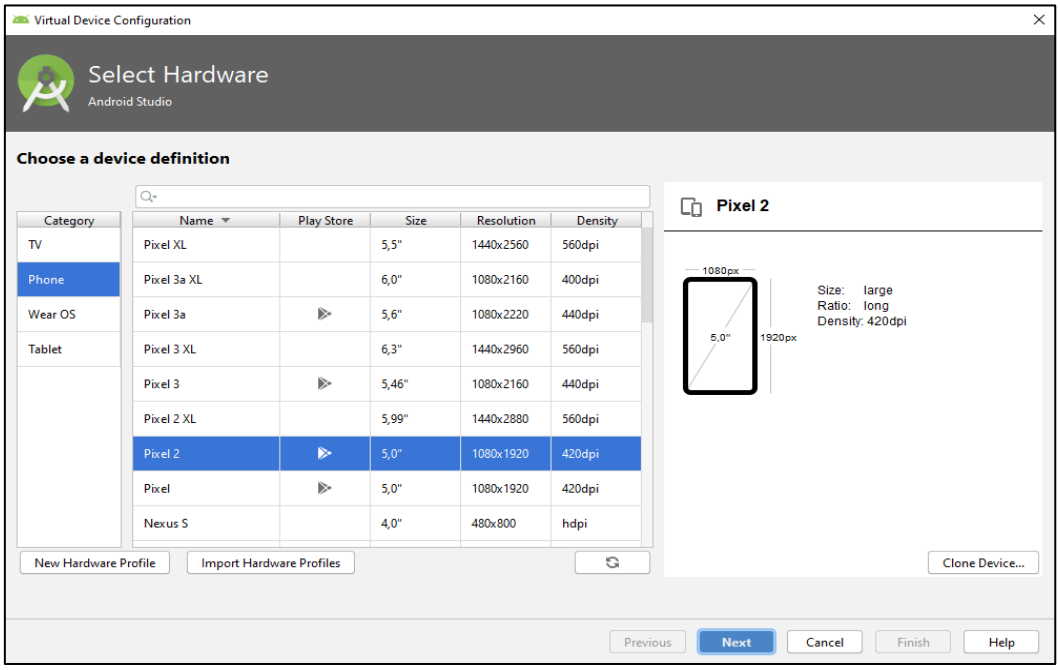
Android Studio [41], android uygulama geliştirme, uygulama analizi ve Sanal makine oluşturulmasını sağlayan parçası olan AVD ile resmi bir IDE ortamıdır. Android Studio üzerinde Sanal makine oluşturma adımları şu şekilde gerçekleştirilmiştir.

Android Studio ana ekranında Tools> AVD Manager tabları seçilerek Şekil 3.12'deki gibi Android Sanallaştırma ana ekranına geçiş yapılır.



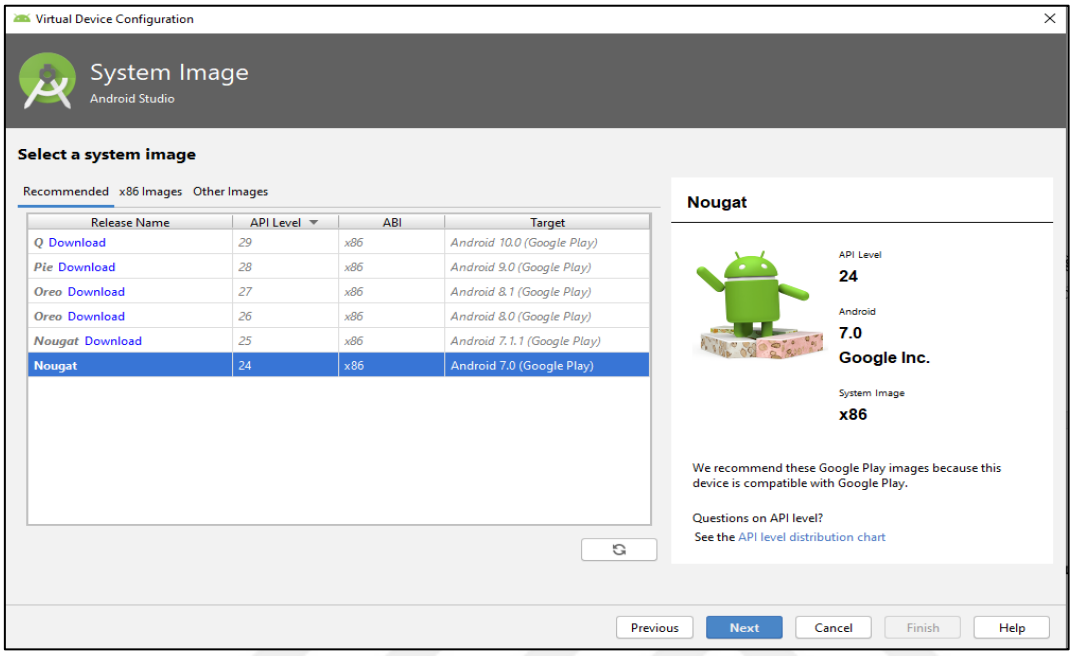

Şekil 3.12. Avd manager ana ekranı

Bu ekrandan en altta bulunan "Create Virtual Device" adımından sanal cihaz oluşturma isteği yapılır. Donanım seçim adımı Şekil 3.13'de gösterilmiştir.



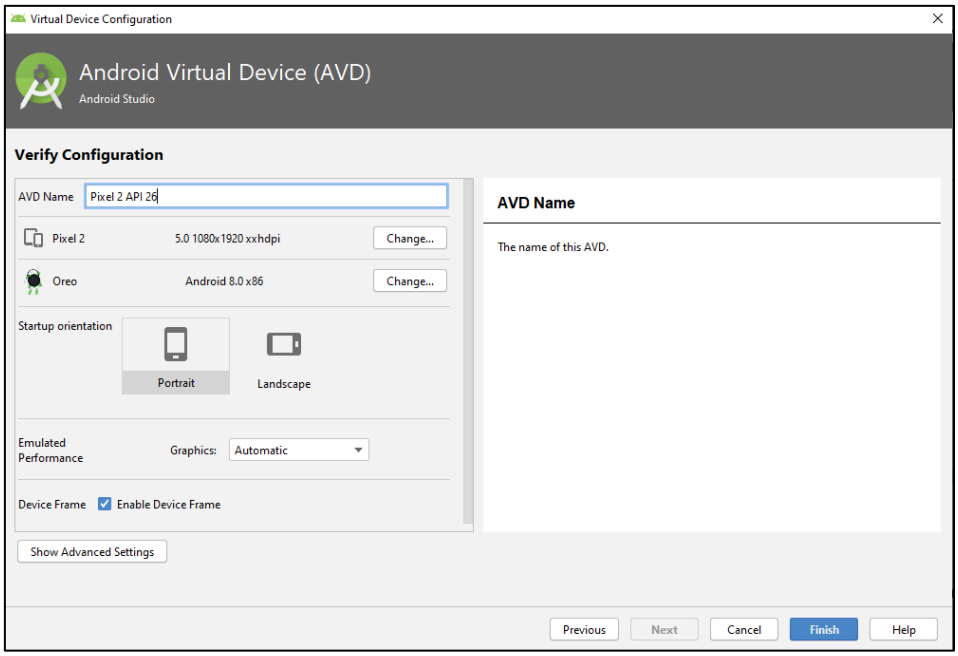

Şekil 3.13. Donanım seçim adımı

Akıllı telefon, tablet, giyilebilir teknolojiler ve TV gibi cihazların seçimi bu adımda gerçekleştirilir. Bu adımdan "Phone" sekmesinden ekran boyutu ve çözünürlüğü gibi özelliklere bağlı olarak bir cihaz seçilir. "Next" ile bir sonraki ekrana geçiş yapılır.




Şekil 3.14. İşletim sistemi seçim ekranı

Şekil 3.14'de görüldüğü gibi işletim sistemi seçim ekranında istenilen işletim sistemi seçimi yapılır. Bu tez çalışması kapsamında Android 8.0 veya Android 9.0 sürümleri seçilmiştir. Seçim yapıldıktan sonra "Next" ile bir sonraki ekran olan Şekil 3.15'e geçilmiştir.




Şekil 3.15. Avd özet görüntüsü adımı

Bu adımda, cihazın isim verme işlemleri, ekran yönü, grafik arayüz performansı seçimi gibi detay ayarların seçimi yapılır. Örnek Şekil 3.14' de görüldüğü üzere Android 8.0 x86 imajı seçilmiş, Pixel 2 cihazının simulator performansı seçimi yapılmıştır. Tüm ayarlar tamamlandıktan sonra "Finish" ile cihaz seçimi işlemi bitirilmiştir. Oluşturulan cihaz AVD manager ana ekranında gözlemlenebilmektedir.

### 3.2.9. Drozer

Drozer aracı [41], dinamik analiz testlerinde kullanılmıştır. Statik analiz testinde elde edilen muhtemel zafiyetli kodların, manuel dinamik analiz testinde kod enjeksiyonu, saldırı yüzeylerini tespit etme, zafiyetli servisleri istismar edilmesi amacıyla için kullanılmıştır. Drozer aracının iki bileşeni bulunmaktadır. Bir bileşen apk dosyası olarak test edilecek olan cihaza Resim 3.1'deki gibi yüklenmiştir. Diğer bileşen ise, saldırı yapacak olan ortama "konsol" bileşeni olarak yüklenmiştir.

Resim 3.1. Drozer ajanı ana ekran görüntüsü

Drozer bileşeninin, kendi makinenizdeki parçası olan konsol parçasını aktif edebilmek için, port yönlendirme işlemi yapılması gerekmektedir. Bu port yönlendirme işleminden sonra,

konsol ile ajan uygulama arasındaki bağlantı sağlanmıştır. Bu yönlendirme işlemi Resim 3.2'de verilmiştir.

```
C:\Users\3MR3>adb forward tcp:31415 tcp:31415
31415
```

Resim 3.2. Port yönlendirme adımı

### 3.3. Test Aşamaları

Uygulamalar iki yöntemle test edilmiştir. Öncelikle statik analiz yöntemiyle kod seviyesinde analizler yapılmıştır. Bu aşamada çıkan muhtemel zafiyetli kodlar ve durumlar dinamik analiz aşamasında uygulama çalışma zamanında onaylanıp ilgili zafiyetin varlığı gözlemlenmiştir.

Bu tez kapsamında önerilen test modeli (Bknz. Şekil 3.1 Önerilen test modeli) kapsamında 1, 2, 3, 4 ve 5 numaralı aşamaların detayları şu şekilde verilmiştir.

*1 Numaralı Aşama:*

Bu aşamada uygulamalar Androbugs Framework ile zafiyet testine tabi tutulmuştur. Androbugs aracı, Kali makine üzerinde kurulmuştur. Seçilen bir uygulamanın testi için komut satırında aşağıdaki Şekil 3.16'daki gibi komut yürütülmüştür.

```
root@kali:/opt/mobile/AndroBugs_Framework# python androbugs.py -f /root/Desktop/DATASET/[redacted].apk -o /root/Desktop/Rapor/Uygulama1.txt
*********************************************************************
**   AndroBugs Framework - Android App Security Vulnerability Scanner  **
**                         version: 1.0.0                              **
**    author: Yu-Cheng Lin (@AndroBugs, http://www.AndroBugs.com)      **
**             contact: androbugs.framework@gmail.com                  **
*********************************************************************
```

Şekil 3.16. Androbugs apk testi komut ekranı

*-f : test edilecek apk dosyasının tam adresi*

*-o : test raporunun hangi dizine kaydedileceğinin belirtilmesi*

İlgili komutu yürüttükten sonra, Androbugs belirttiğimiz dizinde bir rapor üretmiş olacaktır. İlgili raporda Kritik, Uyarı ve Bilgi seviyesinde zafiyet testi seviyelerinde rapor karşımıza çıkmaktadır. Test edilen Uygulama-1 uygulamasının test raporunun bir bölümü aşağıdaki Şekil 3.17'de verilmiştir.

Aşağıdaki Şekil 3.17'de de görüleceği üzere, raporun ilk sayfalarından alınan çıktıda öncelikle "Kritik" seviyedeki zafiyetler verilmiştir. Test raporunda Kritik zafiyetlerden <SSL Security> altında görülen <SSL Implementation Checking (SSL Uygulama

Yapılandırma Kontrolü)" adımında uygulamanın ALLOW_ALL_HOSTNAME_VERIFIER veya Allow_All_Hostname_Verifier sınıflarının varlığı belirtilmiştir. Bu sınıfların varlığı, uygulamanın hedefteki SSL sertifikasının CN name'ini kontrol etmemesi sonucunu doğurmaktadır. Bu durumda SSL bağlantılarında MiTM atağı gerçekleştirilebilme olasılığı oluşmaktadır.

```
[Critical] <Command> Runtime Command Checking:
           This app is using critical function 'Runtime.getRuntime().exec("...")'.
           Please confirm these following code secions are not harmful:
               => Lcom/objsys/asn1j/runtime/s;->i()Ljava/lang/String; (0x22) --->
                    Ljava/lang/Runtime;->exec(Ljava/lang/String;)Ljava/lang/Process;
               => Ltr/gov/tubitak/uekae/esya/api/common/tools/Ini;->a()Ljava/util/Properties; (0x54) --->
                    Ljava/lang/Runtime;->exec(Ljava/lang/String;)Ljava/lang/Process;
               => Ltr/gov/tubitak/uekae/esya/api/common/tools/Ini;->a()Ljava/util/Properties; (0x100) --->
                    Ljava/lang/Runtime;->exec(Ljava/lang/String;)Ljava/lang/Process;
               => Ltr/gov/tubitak/uekae/esya/api/common/tools/Ini;->a()Ljava/util/Properties; (0x122) --->
                    Ljava/lang/Runtime;->exec(Ljava/lang/String;)Ljava/lang/Process;
[Critical] <#BID 64208, CVE-2013-6271#> Fragment Vulnerability Checking:
           'Fragment' or 'Fragment for ActionbarSherlock' has a severe vulnerability prior to Android 4.4 (API 19).
           Please check:
           (1)http://developer.android.com/reference/android/os/Build.VERSION_CODES.html#KITKAT
           (2)http://developer.android.com/reference/android/preference/PreferenceActivity.html#isValidFragment(java.lang.String)
           (3)http://stackoverflow.com/questions/19973034/isvalidfragment-android-api-19
           (4)http://securityintelligence.com/new-vulnerability-android-framework-fragment-injection/
           (5)http://securityintelligence.com/wp-content/uploads/2013/12/android-collapses-into-fragments.pdf
           (6)https://cureblog.de/2013/11/cve-2013-6271-remove-device-locks-from-android-phone/
               You MUST override 'isValidFragment' method in every "PreferenceActivity" class to avoid Exception throwing in Android 4.4:
                   Lcom/jeremyfeinstein/slidingmenu/lib/app/SlidingPreferenceActivity;
               All of the potential vulnerable "fragment":
                   Lcom/seneka/edosya/ForwardItemDetailFragment;
                   Lcom/seneka/edosya/PdfFragment;
                   Lcom/seneka/edosya/PoolItemDetailFragment;
[Critical] <KeyStore><Hacker> KeyStore Protection Checking:
           The Keystores below seem using "byte array" or "hard-coded cert info" to do SSL pinning (Total: 1). Please manually check:
               => Lcom/loopj/android/http/MySSLSocketFactory;->getKeystore()Ljava/security/KeyStore; (0x16) --->
                    Ljava/security/KeyStore;->load(Ljava/io/InputStream; [C)V
[Critical] <SSL_Security> SSL Implementation Checking (Verifying Host Name in Fields):
           This app does not check the validation of the CN(Common Name) of the SSL certificate ("ALLOW_ALL_HOSTNAME_VERIFIER" field or
           "AllowAllHostnameVerifier" class).
           This is a critical vulnerability and allows attackers to do MITM attacks with his valid certificate without your knowledge.
           Case example:
           (1)http://osvdb.org/96411
           (2)http://www.wooyun.org/bugs/wooyun-2010-042710
           (3)http://www.wooyun.org/bugs/wooyun-2010-052339
```

Şekil 3.17. Androbugs test raporu

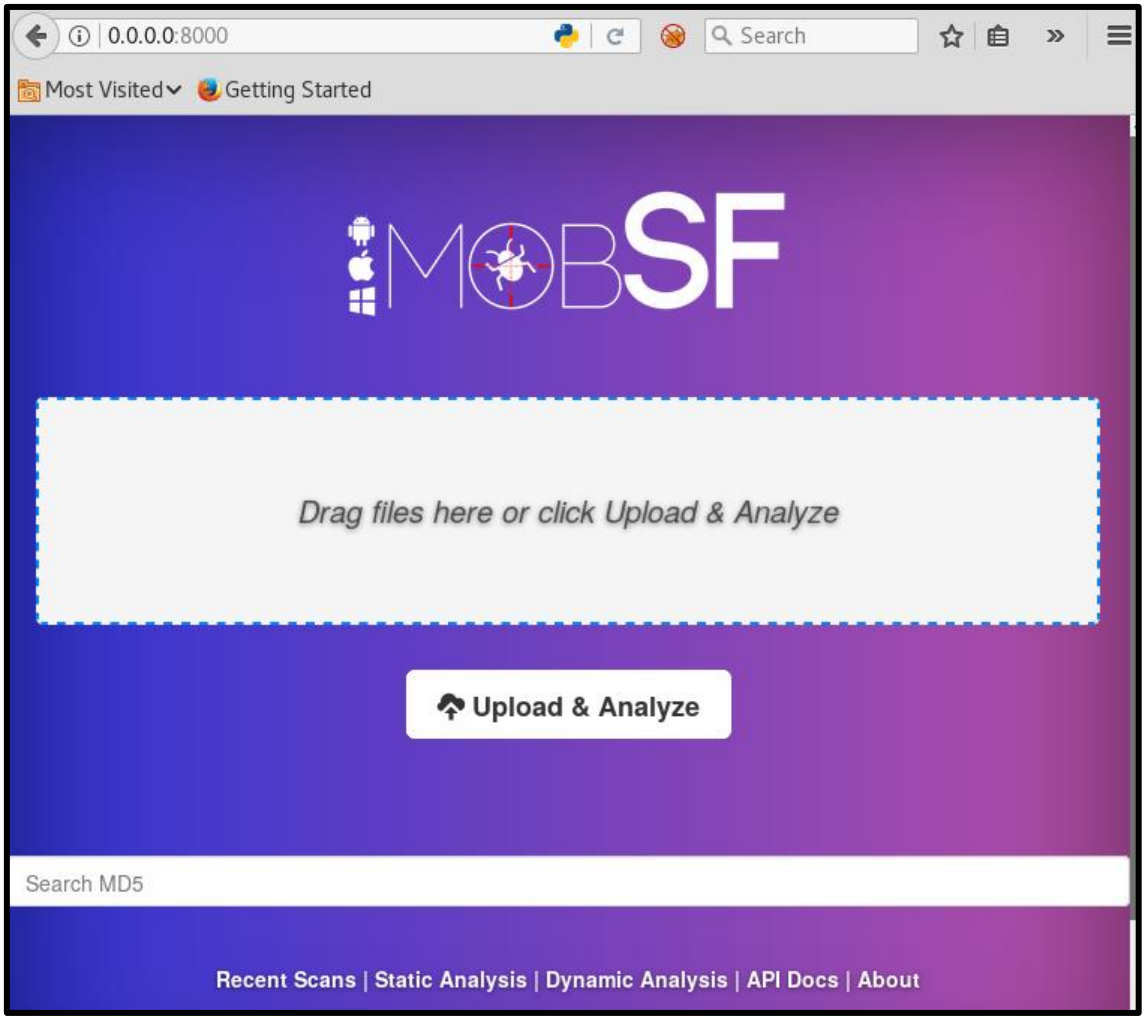
*2 Numaralı Aşama:*

Bu aşamada uygulamalar MobSF aracı kullanılarak statik analiz testine tabi tutulmuştur. Mobsf aracı Kali Linux makinesi üzerinde kurulmuştur. Bu aracın, web ara yüzü bulunmaktadır. Mobsf aracının başlatılması aşaması Şekil 3.18'de gösterildiği şekilde yapılmıştır. Aracın kurulum dosyalarının olduğu dizinde, "*run.sh*" ön tanımlı betiğinin çalıştırılması sonucunda, komut satırında görüleceği üzere "*http://0.0.0.0:8000*" adresinde uygulama çalışır hale gelmiştir.

```
root@kali:/opt/mobile/Mobile-Security-Framework-MobSF# ./run.sh
[2019-11-30 21:52:53 +0300] [2531] [INFO] Starting gunicorn 19.9.0
[2019-11-30 21:52:53 +0300] [2531] [INFO] Listening at: http://0.0.0.0:8000 (2531)
[2019-11-30 21:52:53 +0300] [2531] [INFO] Using worker: threads
[2019-11-30 21:52:53 +0300] [2534] [INFO] Booting worker with pid: 2534
```

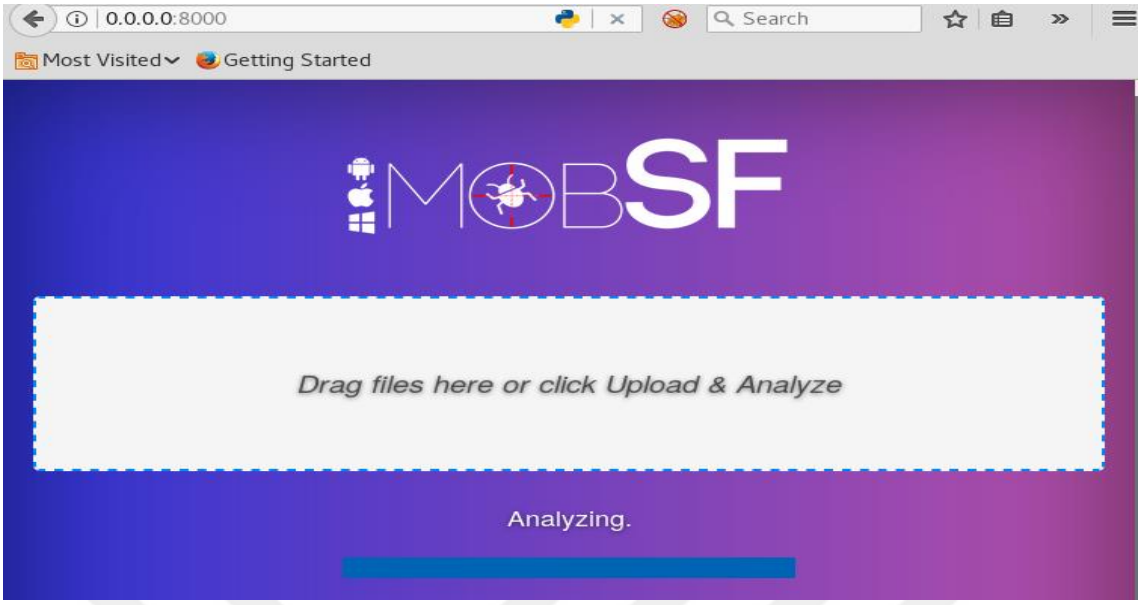
Şekil 3.18. Mobsf ilk çalıştırılma ekranı

Uygulamaların APK dosyaları web arayüzü vasıtasıyla ilgili araca Şekil 3.19'da görülen ara yüzden "Drag files here or click Upload and Analyze(Uygulama dosyasını ya bu Alana sürükleyin ya da Upload butonuna tıklayarak ilgili dosyası seçin)" adımını kullanarak APK dosyası Mobsf aracına analiz için yüklenmiştir.



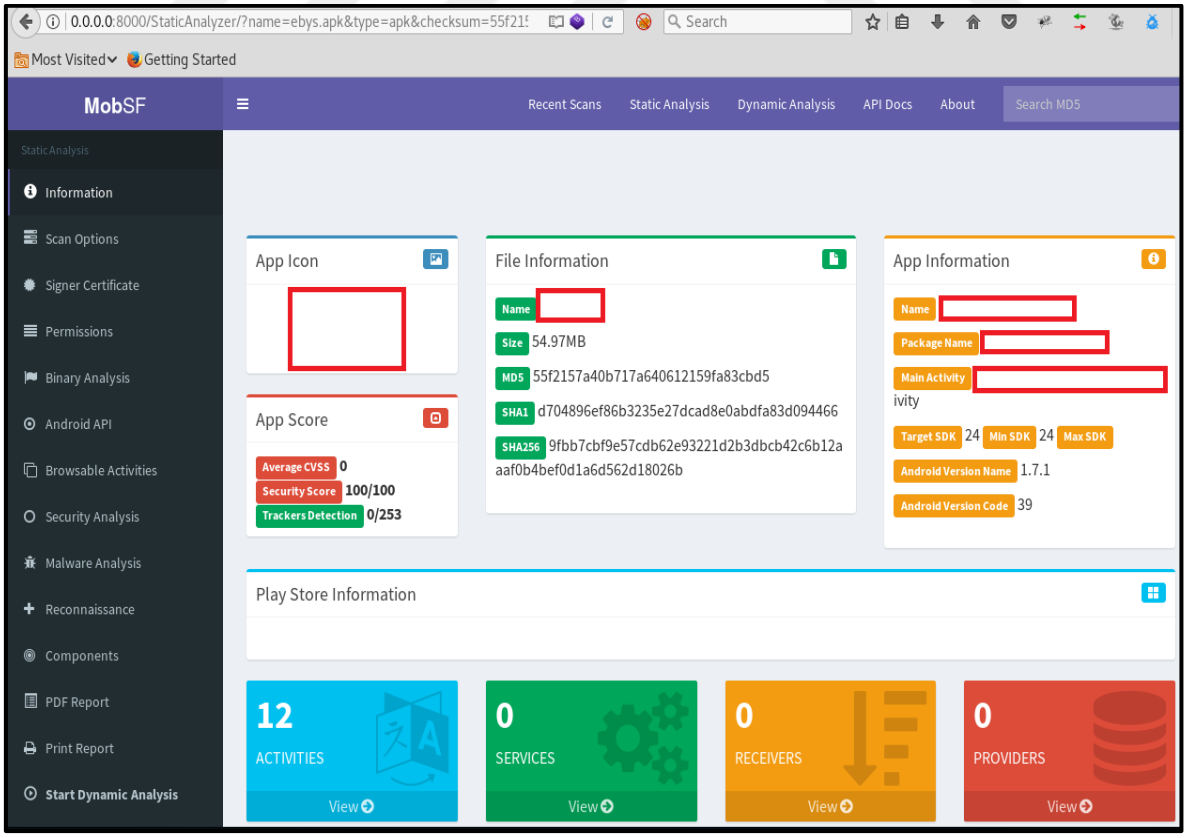

Şekil 3.19. Mobsf apk dosyası yükleme web arayüzü

Uygulama dosyası seçilip, yüklendiğinde Şekil 3.20'deki ekran ile dosyanın uygulama yüklenmesi ve analiz edilmesi adımı beklenmektedir.




Şekil 3.20. Mobsf analiz aşaması

Uygulama test edildikten sonra, Şekil 3.21'deki gibi bir test sonuç ekranı karşımıza gelmektedir. Burada uygulamanın genel bilgileri, izin yönetimi, taranabilen aktiviteler, güvenlik analizi, zararlı analizi ve uygulamanın parçaları olarak ana menüler halinde test raporu verilmiştir.




Şekil 3.21. Mobsf analiz test raporu çıktısı

*3 Numaralı Aşama:*

Bu aşamada, Android 7.0 sürümünden sonra tanımlanan ve ağ bağlantılarının güvenliğine odaklanan konfigürasyon dosyası *"network_security_config.xml"* dosyasının varlığı ve eğer mevcutsa konfigürasyon detayları analizi ve tespiti yapılmıştır.

APK dosyasının içeriğini görüntülemek için apktool aracından faydalanılmıştır. Bir Apk dosyasının apktool ile kullanımı önceki bölümlerde anlatılmıştır (Bknz. Şekil 3.3.). Sonrasında ilgili xml dosyasının varlığının tespiti amacıyla bir python programlama dilinde bir kod yazılmıştır. Yazılan python kodu aşağıda verilmiştir.

```
#!/usr/bin python
import os.path
if os.path.isfile('res/xml/network_security_config.xml'):
   print ("Dosya mevcut")
else:
   print ("Dosya mevcut degil")
```

İlgili kodu uygulama dosyalarının apktool ile oluşturulduğu ana dizininde çalıştırdığımızda dosyanın bulunmadığı görülmüş ve komut çıktısı "File not exist (Dosya bulunamadı)" çıktısı Şekil 3.22'de verilmiştir.

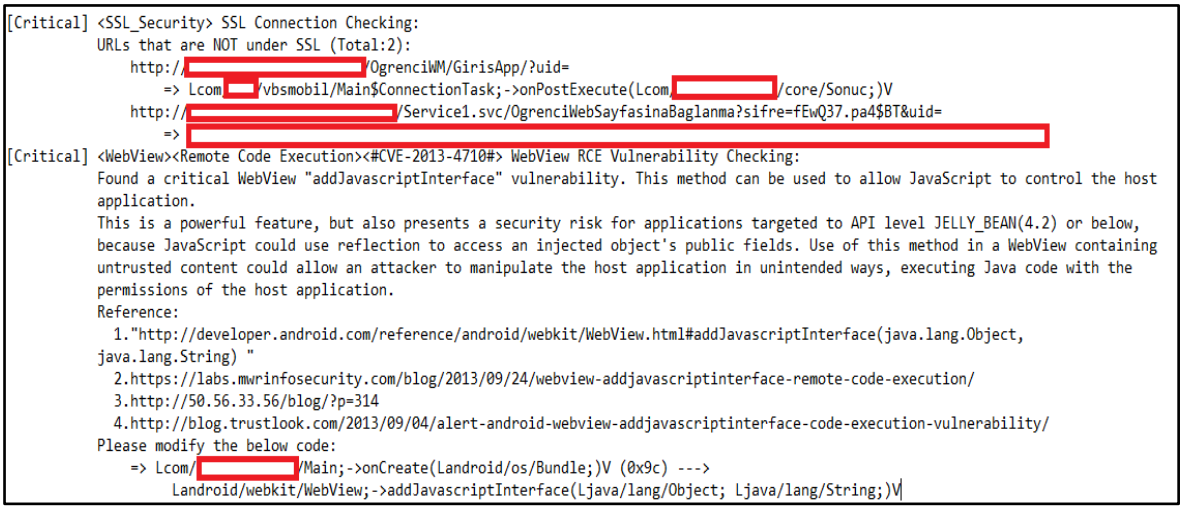
```
root@kali:~/Desktop/DATASET/[redacted]# python search.py
Dosya mevcut degil
```

Şekil 3.22. config dosyasının varlığının test edilmesi sonuç çıktısı

*4 Numaralı Aşama:*

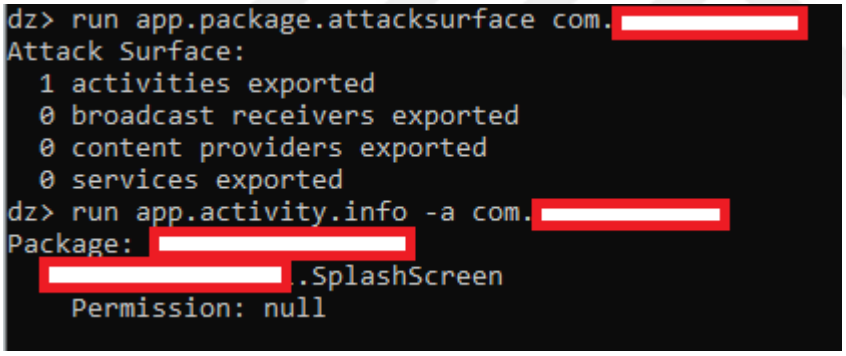
Bu aşamada, statik analiz süreçlerinden elde edilen sonuçlara göre işlem yapılmıştır. Statik analiz süreçlerinde kullanılan açık kaynak araçların çıktılarının yorumlanması bu adımda yapılmıştır. Kod seviyesi analizlerden elde edilen muhtemel zafiyetli kodların tespiti ve uygulama çalışma zamanında ilgili kodların tetiklenmesi üzerine yapılan çalışmaları kapsamaktadır.

Statik analiz sürecinde muhtemel Webview zafiyeti olan Uygulama-2'nin kod çalıştırma zafiyetinin testi manuel olarak yapılmıştır. Androbugs'ın raporundan kırpılan görüntü Şekil 3.23'de verilmiştir.



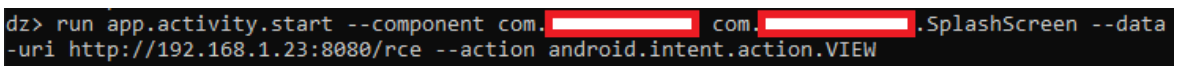

Şekil 3.23. Uygulama-2 webview zafiyeti statik analiz

Muhtemel Webview javascript zafiyetinin tetiklenmesi amacıyla, uygulamanın çalıştırdığı aktivitelerin tespit edilmesi gerekmektedir. Aktivitelerin tespiti amacıyla Drozer (Bknz. 3.2.9. Drozer) aracı kullanılmıştır. Bu araçla, uygulamanın aktiviteleri aşağıdaki Şekil 3.24'de verilmiştir.




Şekil 3.24. Uygulama-2 aktivitelerin tespiti

Uygulamanın 1 aktivitesinin olduğu tespit edilmiştir. Bu aktivite üzerinden zafiyetli kod gönderimi yapılmıştır. Şekil 3.25'de ilgili durum simule edilmiştir. Burada "*http://192.168.1.23:8080/rce*" dizininde Kali makinesinde zararlı kod çalıştırılmaktadır (Bknz. Resim 4.4. zararlı kodun hazırlanması).

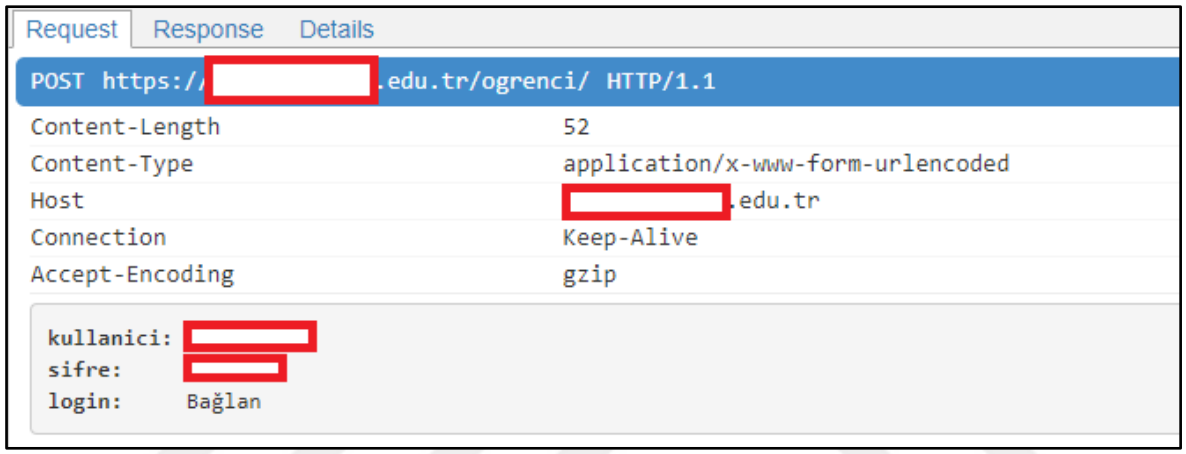



Şekil 3.25. Uygulama-2 aktivitenin script ile çalıştırılması

Uygulama zararlı kodla beraber çalıştırılmış ancak uygulamanın zararlı koda zafiyeti olmadığı tespit edilmiştir. Buradan çıkarılacak sonuç, statik analiz süreçlerinin mutlaka uygulamanın çalışma zamanında yapılan testlerle onaylanması gerekliliğidir.

*5 Numaralı Aşama:*

Bu aşamada uygulamaların ağ bağlantılarının vekil sunucu ile araya girilmesi suretiyle yakalanması ve incelenmesi adımlarını kapsamaktadır. Bu aşamada, statik analiz adımında SSL/TLS bağlantılarında yapılandırma hataları veya yapılandırma eksiklikleri bulunan uygulamaların tespiti çalışma zamanında yapılmıştır. Android cihazının vekil sunucu ayarlarının yapılması önceki bölümlerde verilmiştir (Bknz. Şekil 3.8 android cihazının vekil sunucu ayarlarının yapılması). Örnek olarak, Uygulama-8 üzerinde yapılan test sonucu vekil sunucumuza gelen SSL trafiği Şekil 3.26'da verilmiştir.




Şekil 3.26. Uygulama-8'in ssl trafiğinin vekil sunucuyla yakalanması

Şekil 3.26'da da görüldüğü üzere, ağ bağlantısı https ve Android cihazı üzerine vekil sunucunun sertifikasının da yüklenmemesine rağmen açık bir şekilde kullanıcının girdiği kullanıcı adı ve şifre kısımları görülebilmektedir.

Tez çalışmasında, uygulamaların ağ bağlantıları yapıp yapmadığı, eğer ağ bağlantısı ile çalışıyorsa bunun tespit edilip bağlantının güvenli kanallar kullanılıp kullanılmadığı hususları incelenmiştir. Bu hususlar, uygulama tersine mühendislik yöntemleri ile kaynak koduna döndürülerek kaynak koddaki fonksiyon, sınıf ve ifadelerin incelenmesi ile gerçekleştirilmiştir. Yapılan çalışma kapsamında uygulamalar eğer http bir bağlantı kullanıyorlarsa bu durum da ayrıca raporlanmıştır. HTTPS bağlantılarda ise, bağlantıda kullanılan protokollerin güncelliği, kullanılan fonksiyon ve ilgili kaynak kütüphanelerinin doğru kullanılıp kullanılmadığı durumlarına bakılmıştır. Bu bağlamda, SSL/TLS protokollerinin kullanıldığı durumlarda kod seviyesinde şu kontroller yapılmıştır;

*SSL/TLS Yapılandırma Kontrolü*: Eğer düzdün bir şekilde kontrol edilmediyse; SSL/TLS onaylama hatası oluşur. Bu kontrolün yapıldığı aşamada,

ALLOW_ALL_HOSTNAME_VERIFIER veya AllowAllHostnameVerifier, SSLCertificateSocketFactory and SSLSessionCache metotlarının içinde getInsecure() fonksiyonunun bulunup bulunmadığı kontrolleri yapılmıştır [16].

*SSL/TLS Sertifika Onaylama Kontrolü:* Eğer düzgün bir şekilde kontrol edilmediyse; self-signed, süresi dolmuş veya site adıyla uyuşmayan sertifikaların kullanıldığı durumlar oluşacak ve uygulama saldırıya maruz kalmış olacaktır. Bu aşamada kontrol edilen metotlar; "X509Certificate" kütüphanesi altında bulunan "checkClientTrusted", "checkServerTrusted" "getAcceptedIssuers" metotlarının kullanılıp kullanılmadığı kontrol edilmiştir [16].

*SSL/TLS Sertifika Pinleme Durumu Kontrolü:* HPKP, X509 Sertifikasının SebjectPublicKeyInfo kısmının özet fonksiyonu bilgisinin ön bilgi olarak uygulama içerisinde belirtilmesi durumudur. Uygulamada eğer HPKP koruması yapıldıysa, "res/xml/network_security_config.xml" dosyası mevcuttur ve sunucu tarafında kullanılmış olan sertifikanın açık anahtarının özet fonksiyonu çıktısı bu yapılandırma dosyası içerisinde belirtilmiş demektedir. HPKP durumu aktif edildiğinde, artık uygulama şifreli bağlantı başlattığı durumlarda sunucudan gelen sertifikanın özet bilgisi ile uygulama kodu içerisinde bulunan özet bilgisini karşılaştıracak ve eşleşmesi durumunda bağlantıyı kabul edip, hassas verilerin gönderimini yapacaktır. Bu tür bir koruma, ortadaki adam saldırılarının [36] önüne geçmekte ciddi bir önlem olacaktır.

*SSL/TLS hata durumunda çalışma kontrolü:* SSL/TLS bağlantılarında sertifikalar uyuşmadığında hata durumu oluşur. Eğer sertifikaların uyuşmadığı durumlarda kullanılan "onReceivedSSLError" sınıfında "proceed()" fonksiyonu kullanılmışsa, bu durum bağlantının devam etmesine ve ilgili hatanın görmezden gelinmesine yol açar. Bu sebeple, eğer kodlarda bu sınıfın altında güvensiz olan "proceed()" fonksiyonu kullanıldıysa uygulama saldırılara açık hale gelmekte ve sertifika kontrolü yapmadığı anlaşılmaktadır. SSL hata durumunda mobil uygulamaların devam etme durumunun oluşturduğu zafiyetlerin araştırılması ve tespit edilmesi Liu ve diğ.'nin [37] ilgili çalışmasında detaylı bir şekilde ele alınmıştır.

Bu tez çalışmasında, depolama alanında eğer dosya yazılıyorsa izin hakları kontrol edilmiştir. Dâhili depolama alanı, harici depolama alanı ve SQLite veri tabanında şifresiz veri saklama durumları da ayrıca kontrol edilmiştir. Ayrıca uygulamaların

AndroidManifest.xml dosyalarındaki izin istekleri incelenmiş ve hassas durumlar oluşturabilecek izin istekleri tespit edildiği durumlar raporlanmıştır.

### 3.3.1. Statik analiz

Statik analiz sürecinde seçtiğimiz uygulamaların kod seviyesinde analizleri yapılmıştır. Kullanılan iki açık kaynak kodlu araç statik analiz sürecinde kullanılmıştır. Kullanılan araçların hangi zafiyetleri kontrol edip edemediğine dair çizelge Çizelge 3.1' de gösterilmiştir.

Çizelge 3.1. Araçların zafiyetleri test koşulları

| Zafiyetler | AndroBugs | MobSF |
|---|---|---|
| Depolama Alanındaki Zafiyet | ✓ | ✓ |
| Webview zafiyeti | ✓ | ✓ |
| SQLite Veri Tabanı kriptolaması | ✓ | ✓ |
| Intent Zafiyet Kontrolü | × | ✓ |
| Reklam Modülü Analizi | × | ✓ |
| API kontrolü | ✓ | ✓ |
| SSL Sertifika Pinleme Kütüphanesi kontrolü | × | ✓ |
| SSL/TLS hata durumundaki zafiyet | ✓ | × |
| SMS ve Telefon çağrıları kontrolü | ✓ | ✓ |
| Hata ayıklama modundaki saldırılar | ✓ | ✓ |

Statik analiz aşamasında kullanılan araçların tam olarak test edemediği durum olan SSL/TLS bağlantılarındaki zafiyetler kısmı manuel olarak ayrıca test edilmiştir. Android 7.0 sürümünden sonra, uygulamaların kriptolu bağlantıları için daha geniş tanımlamalar yapılabileceği ve Android'in yerel kütüphaneleri arasına giren network_security_config.xml dosyası ortaya çıkarılmıştır. Bu xml dosyası ile çok daha detaylı bir şekilde ağ ayarları düzenlenebilmekte ve yerel kütüphane olduğundan dolayı herhangi bir uyumsuzluk durumları oluşmamaktadır. Örnek bir xml dosyası aşağıda verilmiştir. Bu dosyanın var olup olmadığı ve eğer varsa içerisindeki bilgilerin incelenip raporlanması kısmı için manuel olarak incelemeler yapılmıştır.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest ... >
    <application
android:networkSecurityConfig="@xml/network_security_config"

    </application>
</manifest>
```

Network_security_config.xml dosyası ile yapılabilecek özelleştirmeler aşağıda verilmiştir.

İşletim sisteminde varsayılan olarak yüklü gelen güvenilen sertifikaların listesinin kısıtlanması veya çevrimdışı üretilmiş(self-signed) sertifikanın güvenilir sertifika olarak eklenmesi, Hata ayıklama modunda güvenli bir şekilde testlerin gerçekleştirilmesinin sağlanması, Açık şifresiz bir şekilde gönderilen trafiğin engellenmesi, Sertifika pinlemesinin yapılabilmesi, bu sayede sunucu tarafındaki sertifikanızın özet fonksiyonu bilgisinin uygulamada kullanılması sağlanarak, araya girme ve trafiği izleme saldırılarının ortadan kaldırılmasının sağlanması özellikleri gerçekleştirilebilmektedir.

### 3.3.2. Dinamik analiz

Dinamik analiz sürecinde Android studio ortamında manuel olarak testler gerçekleştirilmiştir (Bknz. Şekil 3.1.). AndroBugs Framework ve MobSF sadece statik analiz araçları olarak kullanılmıştır. Bu aşamada statik analizden alınan çıktıların, çalışma zamanında uygulamaya gerekli girdileri yaparak test edilmesi sağlanmıştır.

### 3.3.3. Test ortamı

Gerçekleştirilecek test ortamında 1 adet Windows 10 host makinesi, üzerinde Vmware sanallaştırma ortamında çalışan Kali Linux sanal makinesi, Windows 10 makinesi üzerinde kurulu Android Studio, Android Sanal Makine Yöneticisi (AVD Manager), Android Sanal makine üzerinde çalışan Android 9.0 ve Android 8.0 sürümlerinde iki sanal Android cihazı, Burp Suite Community edition, Mitmproxy, adb tools, apktool araçları bulunmaktadır. Kali Linux sanal makinesi üzerinde kurulu apktool, Androbugs Frameworks, MobSF framework bulunmaktadır.

### 3.3.4. Test edilen mobil uygulama veri kümesi

Tez kapsamında, kamu kurumlarının Android işletim sisteminde çalışan mobil uygulamaları test edilmiştir. Test sürecine karar verilirken, uygulamaların vatandaşlar tarafından kullanım yoğunluğuna, uygulama marketinden indirilme sayısına, uygulama içerisinde erişilen verilerin kritiklik durumuna bakılmıştır. Bu sebeple test edilen uygulamaların etkisinin kamu kurumlarının Test sonuçlarında uygulama isimleri verilmemiştir. Uygulamaların kritik zafiyete sahip olanlarından bahsedilirken "Uygulama1" gibi isimlerle adlandırılmıştır. Bu durum hem kamu kurumlarının kritik öneme sahip zafiyetlerinin kamuya açık bir şekilde belirtilmesinden kaçınılmıştır. Uygulamaların indirilme sayıları ve versiyon bilgisi Çizelge 3.2'de verilmiştir.

Çizelge 3.2. Uygulamaların versiyon ve indirilme sayıları

| **Uygulama** | **Uygulama Versiyonu** | **Yüklenme Sayısı** |
|---|---|---|
| Uygulama-1 | 1.7.1 | 50Bin+ |
| Uygulama-2 | 2.2 | 10Milyon+ |
| Uygulama-3 | 2.1.2 | 100Bin+ |
| Uygulama-4 | 1.0.11 | 500Bin+ |
| Uygulama-5 | 6.7.0 | 5Milyon+ |
| Uygulama-6 | 1.2.41 | 100Bin+ |
| Uygulama-7 | 1.0.5 | 10Bin+ |
| Uygulama-8 | 1.5.5 | 50Bin+ |

# 4. TEST SONUÇLARI ve ANALİZ

Uygulamaların testi iki aşamada yapıldığından dolayı her aşamanın test sonuçları ayrı başlık altında verilmiştir. Test kapsamında elde edilen sonuçlar da yine her test başlığının kendi alanında ifade edilmiştir.

## 4.1. Statik Analiz Test Sonuçları

Statik analiz aşamasında, AndroBugs Framework ile MobSF araçları kullanılarak gerçekleştirilen testler ve bu testlerin sonuçları aşağıdaki Çizelge 4.1' de verilmiştir.

Çizelge 4.1. Statik analiz test sonuçları

| | Açık Şifresiz Ağ trafiği | Şifreli Ağ Trafiğinde yapılandırma hatası | WebView kod çalıştırma zafiyeti | WebView yerel dosya kontrol zafiyeti | Depolama Alanındaki Zafiyetler | Tehlikeli İzin İsteği | Dâhili Alanda Geçici Dosya oluşturma |
|---|---|---|---|---|---|---|---|
| Uygulama-1 | X | X | | | | X | |
| Uygulama-2 | X | X | X | | | X | |
| Uygulama-3 | X | X | X | | | X | |
| Uygulama-4 | | X | | | X | X | |
| Uygulama-5 | X | X | X | | X | X | X |
| Uygulama-6 | X | | | X | | X | |
| Uygulama-7 | | X | X | | X | X | |
| Uygulama-8 | X | X | X | X | | X | |

Test sonuçlarını incelediğimizde, uygulamaların ağ trafiğinde yaptığı işlemlerin birçoğunun açık şifresiz bir şekilde gerçekleştirildiği gözlenmiştir. Şifreli yapılan trafik kayıtlarında ise, kullanılan protokollerin yapılandırma hatalarından dolayı zafiyetlere açık olduğu gözlenmiştir. Şifreli bağlantılarda, uygulamalarda 3.parti kütüphane kullanımı olduğu gözlenmiş ve bu kütüphanelerin düzgün bir şekilde fonksiyonlarda kullanılmadığı görülmüştür.

Otomatik test yapan araçlarla yapılan testlerin dışında, manuel olarak da testler gerçekleştirilmiştir. Bu testlerde HTTPS trafiği kullanan uygulamaların HPKP (Bknz 2.3.1 Ağ bağlantılarının güvenliği) kullanılma durumları incelenmiştir. Uygulamaların kullandığı şifreli trafiğin test sonuçları Çizelge 4.2 ile gösterilmiştir.

Çizelge 4.2. Sertifika pinleme test sonuçları

| | HPKP aktif | HPKP yok |
|---|---|---|
| Uygulama-1 | | X |
| Uygulama-2 | | X |
| Uygulama-3 | | X |
| Uygulama-4 | | X |
| Uygulama-5 | X | |
| Uygulama-6 | X | |
| Uygulama-7 | | X |
| Uygulama-8 | | X |

Test sonuçlarına baktığımızda birçok uygulamanın HPKP yapılandırma ayarlarını yapmadığı açıkça görülmektedir. Uygulama geliştiricilerinin, şifreli trafiğinin güvenliğini korumak için bu ayarları yapması büyük önem arz etmektedir. Önceki bölümlerde detaylı anlatıldığı gibi, düzgün bir şekilde HPKP kullanılan uygulamaların şifreli trafiğine saldırganların müdahale etmesi ihtimali oldukça düşüktür.

Statik analiz sürecinde tespit edilen tehlikeli izin istekleri detaylı bir şekilde Çizelge 4.3 ile verilmiştir.

Çizelge 4.3. Tehlikeli olabilecek izin istekleri

| Tehlikeli İzin istekleri | Uygulama-1 | Uygulama-2 | Uygulama-3 | Uygulama-4 | Uygulama-5 | Uygulama-6 | Uygulama-7 | Uygulama-8 |
|---|---|---|---|---|---|---|---|---|
| WRITE_EXTERNAL_STORAGE | Var | | Var | Var | Var | Var | Var | Var |
| READ_EXTERNAL_STORAGE | Var | | Var | Var | Var | | Var | Var |
| READ_PHONE_STATE | | Var | | | | | Var | |
| CHANGE_WIFI_STATE | | | Var | | | | | |
| ACCESS_MOCK_LOCATION | | | | Var | | | | |
| ACCESS_COARSE_LOCATION | | | | Var | Var | Var | | |
| CALL_PHONE | | | | | Var | Var | Var | |
| MANAGE_ACCOUNTS | | | | | Var | | | |
| BLUETOOTH | | | | | | | Var | |
| CHANGE_NETWORK_STATE | | | | | | | | Var |

Tehlikeli olabilecek izin istekleri çizelgesi incelendiğinde, harici alana dosya yazma(WRITE_EXTERNAL_STORAGE) ve harici alandan dosya okuma(READ_EXTERNAL_STORAGE) isteklerinin yoğunluğu dikkat çekmektedir. Harici depolama alanında diğer tüm uygulamaların da erişimi olduğu ve o alandaki verinin kötü niyetli kişiler tarafından da eklenmiş veya değiştirilmiş olabilme ihtimali değerlendirildiğinde, özellikle harici depolama alanından dosya okuma (READ_EXTERNAL_STORAGE) izni uygulamayı ve aynı zamanda uygulamanın kurulu olduğu mobil cihazı da ciddi riske atabilecek durumdadır.

İzin isteklerinden, CALL_PHONE izin isteği, uygulamanın telefon aramaları yapma, telefon görüşmelerini başlatıp sonlandırma yetkisidir. Bu yetki, mobil cihaz sahibinin haberi olmadan telefon görüşmelerinin başlatılıp sonlandırılmasına yol açmaktadır. Kötü niyetli kişiler tarafından istismar edilebilir. BLUETOOTH izin isteği, mobil cihazın başka kablosuz cihazlarla bağlantı kurmasına, bu yolla veri transferi yapabilmesine yol açabilmektedir. CHANGE_NETWORK_STATE izin isteği, mobil cihazın ağ durumunu kontrol edebilmeye imkan tanımaktadır. Ağ trafiğinin değiştirilmesi, yeni bir ağa bağlanma isteği gibi kötüye kullanılabilecek durumlar teşkil edebilmektedir.

## 4.2. Dinamik Analiz Test Sonuçları

Dinamik analiz aşamasında statik analiz test sonuçları girdi olarak kabul edilmektedir. Statik analiz sürecinde bulunan zafiyetler, yapılandırma hataları dinamik analiz sürecinde test edilip onaylanmıştır. Bu analiz sürecinde, ağ iletişimindeki zafiyetlere ve depolama alanındaki yapılandırma hatalarına odaklanılmıştır. Çünkü statik analiz sürecinde kod seviyesinde tespit edilen diğer hususların bu süreçte de ayrıca kontrol edilmesine ihtiyaç duyulmamıştır.

Manuel olarak yapılan testler sonucunda, aşağıdaki Çizelge 4.4'de dinamik analiz test sonuçları verilmiştir.

Çizelge 4.4. Dinamik analiz test sonuçları

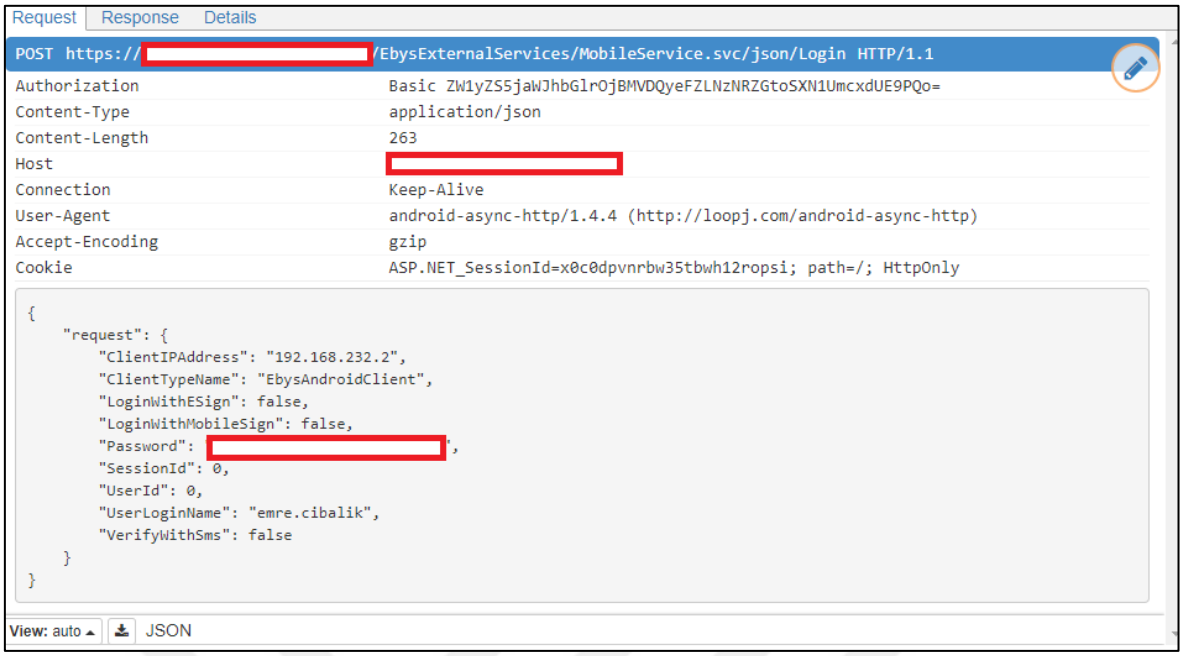
| | HTTP bağlantıların varlığı | Şifreli trafikte MITM saldırısı varlığı | MITM saldırısına karşı güvenli |
|---|---|---|---|
| Uygulama-1 | X | X | |
| Uygulama-2 | X | | X |
| Uygulama-3 | X | | X |

Çizelge 4.4 (devam)

| | | | |
|---|---|---|---|
| Uygulama-4 | | | X |
| Uygulama-5 | X | | X |
| Uygulama-6 | X | | X |
| Uygulama-7 | | X | |
| Uygulama-8 | | X | |

Dinamik analiz test sonuçlarının statik analiz test sonuçlarıyla karşılaştırıldığında, SSL/TLS yapılandırmalarındaki hataların uygulamanın çalışma zamanındaki testlerinde başarısız olduğu görülmüştür. Statik analizde, Uygulama-2, Uygulama-3, Uygulama-4, Uygulama-5 için kod seviyesinde şifreli bağlantı zafiyeti tespit edilmesine rağmen, çalışma zamanında yapılan testlerde MITM saldırısı başarılı olamamıştır. Statik analizde ulaşılan sonuçların dinamik analizde tekrar teste tabi tutulmasının aslında gerekliliği de buradan gelmektedir. Statik analiz sürecinde, kullanılan araçlarla ve manuel testlerle uygulama APK dosyası tersine mühendislik yöntemleriyle kaynak koduna dönüştürülmeye çalışılmıştır. Ancak geliştiricilerin kodlama yaparken almış olabileceği bazı önlemler nedeniyle yapılan işlemlerde her zaman tamamen aynı kaynak koduna dönülmesi mümkün değildir. Ayrıca, uygulama geliştirilirken kullanılan 3.parti kütüphaneler, fonksiyonlar gibi bileşenler de her zaman tersine mühendislik yöntemleriyle kaynağına döndürülmeleri de mümkün olamamaktadır. Dinamik analiz yöntemleri ile aslında bir nevi son test yapılması ihtiyacı gerekmektedir. Bu yüzden, bu tez çalışmasında iki adımlı bir test yöntemi kullanılmıştır. Ayrıca kullanılan araçların ağ trafiğindeki SSL/TLS yapılandırmalarında HPKP mekanizmasındaki testte eksikliklerinin olduğu tespit edilmiş ve bu değerlendirme sonucunda ağ trafiğindeki şifreli bağlantıların testi manuel olarak da icra edilmiştir.

Başarılı bir MITM saldırısının gerçekleştirildiği, Uygulama-1 için verilen ekran görüntüsü Resim 4.1' ile verilmiştir. Uygulamaların hassas veri içeren kısımları gizlenerek verilmiştir.

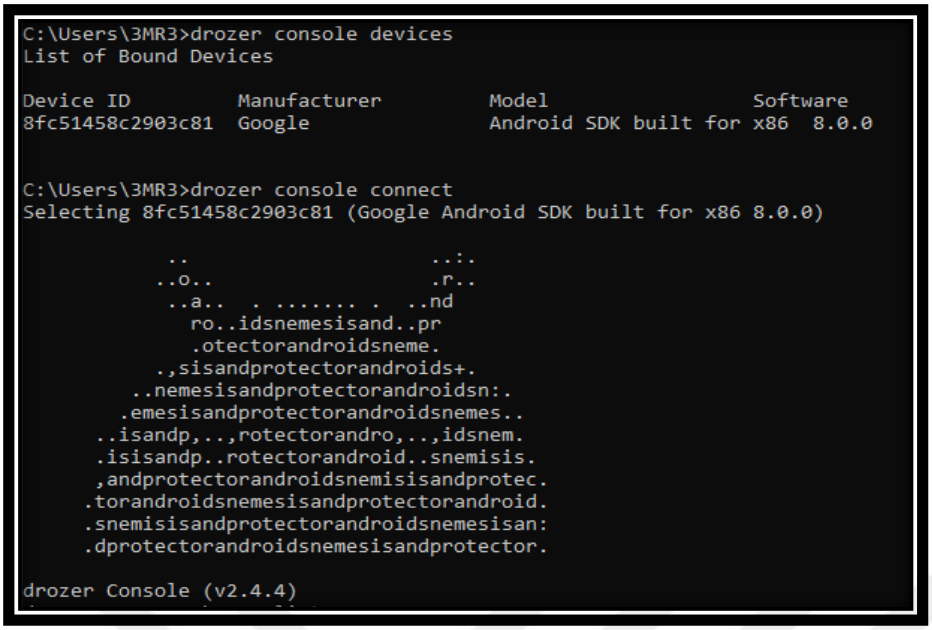



Resim 4.1. Uygulama-1'e yapılan MITM saldırısı

Şifreli olarak giden ve hassas veriler içeren (Kullanıcı adı, Parola gibi.) trafiğin kötü niyetli kişiler tarafından araya girilerek okunabildiği örneği, Android çalışma zamanı testlerinde yukarıdaki Uygulama-1 uygulamasının çıktısındaki gibi olmaktadır. Bağlantı https olmasına rağmen, SSL sertifika kontrolünde yapılandırma hatası olduğundan dolayı, uygulama bağlantıyı kabul edip, veriyi göndermiştir.

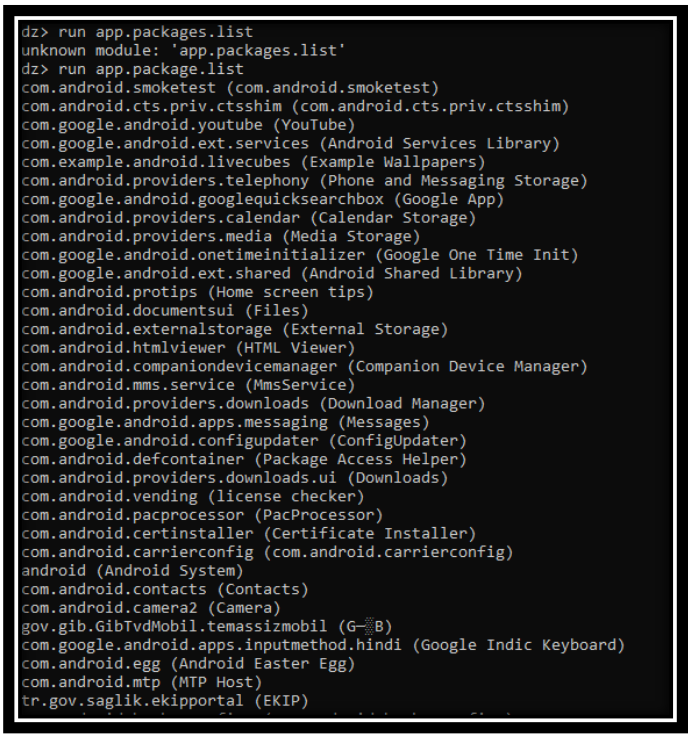
Statik analiz aşamasında muhtemel WebView kod çalıştırma ve WebView yerel dosya kontrol zafiyeti dinamik analiz testleri sonucunda yanlış pozitif (false positive) olduğu görülmüştür. İlgili zafiyetlerin dinamik analiz sonucunda bulunmadığı onaylanmıştır. Uygulama-7 üzerinde gerçekleştirilen çalışmanın detayları aşağıda gösterilmiştir.

Öncelikle, drozer konsol bileşeni ile cihaza Resim 4.2'deki gibi bağlanılmıştır. Bağlantı ekranında görüldüğü gibi, Android 8.0 versiyonundaki cihazımız ve Cihaz ID'si görülmektedir.



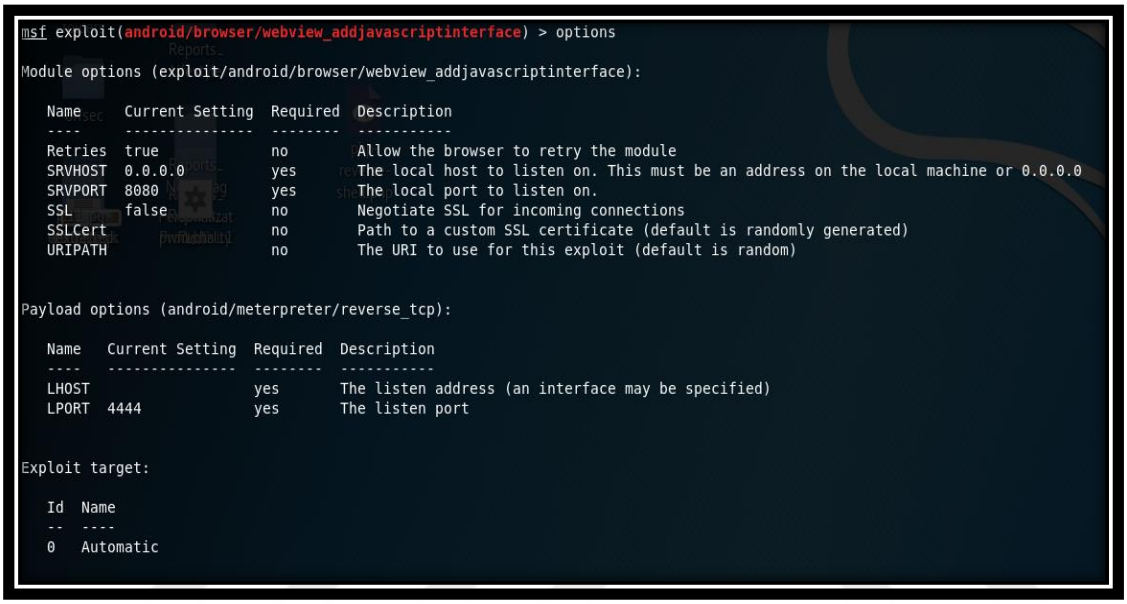

Resim 4.2. Drozer ilk konsol bağlantısı

Zafiyeti test edeceğimiz cihazdaki tüm uygulamalarında listesi, Resim 4.3'de görüldüğü gibi drozer konsolundan "*run app.package.list*" komutu yardımıyla elde edilmiştir. Test edeceğimiz uygulama paketini listeden seçerek, listede görüldüğü gibi bir yazım diliyle kullanmamız gerekmektedir. Bu yüzden, bu ekrandan test edilecek uygulamanın tam paket adresi kopyalanarak alınmıştır.




Resim 4.3. Drozer uygulama listelerinin görülmesi

Resim 4.4'de, Kali Linux makinesi kullanılarak Webview zafiyetini gerçekleştirilecek uygun zararlı modül seçilip ilgili ayarlar yapılmıştır. Burada seçilen modül, test edeceğimiz Webview javascript testi için kullanılan Kali Linux makinesi içerisinde kurulu olarak gelen Metasploit Framework'dür. İlgili çatının kullanımı için, komut satırından "*msf console*" komutu yürütülmelidir. Bu komut başarıyla yürütüldükten sonra, kullanılacak olan zararlı modülü "*use Android/browser/webview_addjavascriptinterface*" komutuyla çağırılmıştır.

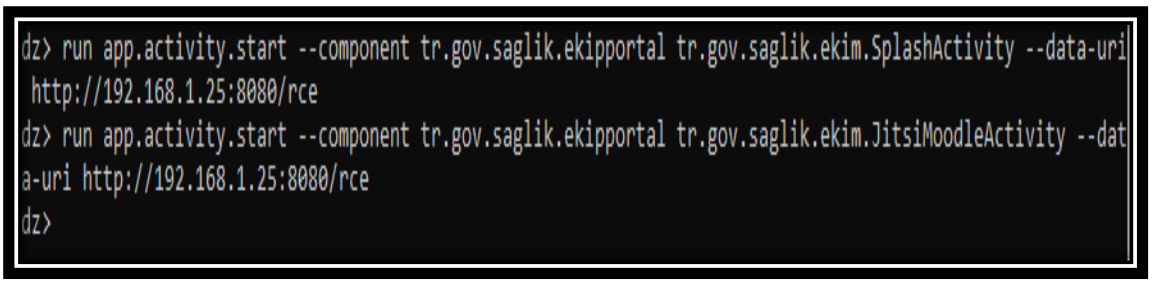



Resim 4.4. Zararlı kodun hazırlanması-1

İlgili zararlı modülün çalıştırılması için bazı bilgilerin girilmesi gerekmektedir. Zararlı kodun çalıştırılacağı dosya yolu, IP adresi, TCP portu bilgileri Resim 4.5 de ayarların nasıl yapıldığı gösterilmiştir. Bu adımda, Kali makinesinde zararlı javascript kodu eklenmiş ve bu adresin ziyaret edilmesi durumunda zararlı kodun çalışması durumu oluşturulmuştur.

Resim 4.5. Zararlı kodun hazırlanması-2

Resim 4.6'da zararlı kodun ilgili uygulama enjekte edilmesi durumu gösterilmiştir. Bu adımda, önceki adımda tespit edilen uygulamanın aktivitilerine, drozer yardımıyla Kali üzerinde oluşturduğumuz zararlı kodun barındırıldığı adres enjekte edilmiştir. Uygulamanın iki aktivitesi olduğu için, iki aktiviteye de zararlı kodun enjekte edilmesi denenmiştir.




Resim 4.6. Zararlı kodun uygulamaya enjekte edilmesi

Resim 4.7'de, zararlı kod enjekte edildikten sonra, Kali makinesi üzerinde saldırı yapılan android cihazından oluşan trafiğin gelip gelmediği kontrolü yapılmıştır. Yapılan kontrollerde zararlı aktivitenin uygulamada bir zafiyete yol açmadığı tespit edilmiştir.

Resim 4.7. Zararlı kodun başarısız olduğunun görülmesi

### 4.3. Tartışma

Önerilen test modeline göre yapılan testler sonucunda, iki aşamalı testin neden yapılması gerektiği hususunun önemine dikkat çekilmek istenmiştir. Statik analiz testi sonucunda ortaya çıkan Webview zafiyetlerinin, dinamik analiz aşamasında manuel olarak yapılan testler sonucunda bulunmadığı onaylanmıştır. Bu durumdan yola çıkarak, sadece statik analiz testi yapılarak uygulamanın bir zafiyete açık olduğunun belirtilmesinin doğru olmadığı görüşü savunulmuştur.

# 5. SONUÇ VE ÖNERİLER

Bu tez çalışmasında, kamu kurumlarının mobil işletim sistemi uygulamalarının güvenlik zafiyetleri incelenmiştir. Yapılan çalışmada, seçilen mobil uygulamalar öncelikle statik analiz sürecinden geçirilmiş, bu aşamada kod seviyesinde zafiyet tespitleri yapılmıştır. Kod seviyesinde zafiyet bulunduran kütüphaneler, fonksiyonlar ve sınıflar dinamik analiz sürecinde girdi olarak kabul edilerek dinamik analiz çalışması icra edilmiştir. Dinamik analiz sürecinde, muhtemel zafiyetli kodların tetiklenmesi için emülator ortamında çalışmalar yapılmıştır. Bulunan zafiyetlerin nasıl giderileceği hususunda yapılandırma önerilerine de tez çalışması içerisinde yer verilmiştir. Yapılan sıralı testler sonucunda, uygulamaların birçoğunda ağ iletişimi hususunda yapılandırma hataları, yapılandırma eksiklikleri ve yanlış protokoller kullanımları gözlenmiştir.

Bu tez çalışmasında, kamu kurumlarının Android uygulamalarının zafiyetleri incelenmiştir. Android 7.0 sürümünden sonra, uygulamaların ağ bağlantılarının güvenliğinin sağlanması için geliştirilen network_security_config.xml konfigürasyonunun kullanımı önemli hale gelmiştir. Gerçekleştirilen çalışma, bu alanda yapılan ilk çalışmalardan biridir. Çalışmanın sonucunda, kamu kurumlarının Android mobil uygulamalarının güvenlik durumlarının incelenip anlamlı hale getirilmesi, seçilen uygulamaların şifreli ağ trafiğinde (SSL/TLS) yapılandırma hatalarının ve yapılandırma eksikliklerinin tespit edilmesi için bir test modeli önerilmesi ve yapılan bu çalışmanın Android en son güncel (Android 8.0 ve Android 9.0) sürümleri kullanılarak gerçekleştirilmesi adına literatüre katkıda bulunulmuştur.

Tespit edilen zafiyetler, mobil uygulama geliştiricileri ve kamu kurumlarının mobil uygulama güvenliği yönünde anlamlı bir çıktı olarak verilmiştir. Buradan çıkarılan sonuç, mobil uygulama geliştiricilerinin kullandığı kütüphaneleri, işletim sisteminin yeni gelen özelliklerini ve kullandığı ağ iletişim protokollerinin zayıflıklarını ve güçlü yanlarının iyi analiz etmeleri gerektiğidir. Bu bağlamda tez çalışmasının Türkiye'deki mobil uygulama güvenliğinde farkındalığı oluşturacağı düşünülmüştür.

# KAYNAKLAR


[1] İnternet: Holst, A. Mobile operating systems' market share worldwide from January 2012 to July 2019, 2019 September 2019. URL: https://www.statista.com/statistics/272698/global-market-share-held-by-mobile-operating-systems-since-2009/, Son Erişim Tarihi: 25.10.2019.

[2] Jignesh Joshi, Chandresh Parekh, "Android smartphone vulnerabilities: A survey," in *2016 International Conference on Advances in Computing, Communication, & Automation (ICACCA) (Spring)*, Dehradun, India, 2016.

[3] Ankita Khandelwal; A K Mohapatra, "An insight into the security issues and their solutions for android phones," in *2015 2nd International Conference on Computing for Sustainable Global Development (INDIACom)*, New Delhi, India, 2015.

[4] İnternet: OWASP Mobile Security Testing Guide,"2019. URL: https://www.owasp.org/index.php/OWASP_Mobile_Security_Testing_Guide, Son Erişim Tarihi: 24.10.2019.

[5] Faysal Hossain Shezan ; Syeda Farzia Afroze ; Anindya Iqbal, "Vulnerability detection in recent Android apps: An empirical study," in *2017 International Conference on Networking, Systems and Security (NSysS)*, Dhaka, Bangladesh, 2017.

[6] Umansakar, "Analysis of latest vulnerabilities in android," in *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, Udupi, India, 2017.

[7] Patrick P.F.Chan; Lucas C.K. Hui; S.M. Yiu, "A privilege escalation vulnerability checking system for android applications," in *2011 IEEE 13th International Conference on Communication Technology*, Jinan, China, 2011.

[8] Alejandro Argudo; Gabriel López; Franklin Sánchez , "Privacy vulnerability analysis for Android Applications: A practical approach," in *2017 Fourth International Conference on eDemocracy & eGovernment (ICEDEG)*, Quito, Ecuador, 2017.

[9] Srikanth Kaka; V. N. Sastry ; Rajib Ranjan Maiti, "On the MitM vulnerability in mobile banking applications for android devices," in *2016 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS)*, Bangalore, India, 2017.

[10] Xuetao Wei, Michael Wolf, "A Survey on HTTPS Implementation by Android Apps: Issues and Countermeasures," *Applied Computing and Informatics*, pp. 101-117, 2017.

[11] Chang, Y., Liu, B., Cong, L., Deng, H., Li, J., & Chen, Y., "Vulnerability Parser: A Static Vulnerability Analysis System for Android Applications," *Journal of Physics: Conference Series*, vol. 1288, no. 1, 2019.

[12] Wu, D., Gao, D., Cheng, E. K., Cao, Y., Jiang, J., & Deng, R. H., "Towards understanding Android system vulnerabilities: Techniques and insights," in *Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security*, Auckland, New Zealand, 2019.

[13] Akhawe, D., & Felt, A. P., "Alice in warningland: A large-scale field study of browser security warning effectiveness," in *Proceedings of the 22nd USENIX Security Symposium.*, Washington, D.C., USA, 2013.

[14] Sascha Fahl, Marian Harbach, Thomas Muders, Lars Baumgärtner, Bernd Freisleben, Matthew Smith, "Why eve and mallory love android: an analysis of android SSL (in)security," in *CCS '12 Proceedings of the 2012 ACM conference on Computer and communications security*, Raleigh, North Carolina, USA, 2012.

[15] David Sounthiraraj, Justin Sahs, Garrett Greenwood, Zhiqiang Lin, Latifur Khan, "SMV-HUNTER: Large Scale, Automated Detection of SSL/TLS Man-in-the-Middle Vulnerabilities in Android Apps," in *Network and Distributed System Security Symposium*, San Diego, CA, USA, 2014.

[16] Yingjie Wang, Xing Liu, Weixuan Mao, Wei Wang, "DCDroid: Automated Detection of SSL/TLS Certificate Verification Vulnerabilities in Android Apps," *URC-AIS 2019: ACM TURC Conference on Artificial Intelligence and Security*, 2019.

[17] İnternet: Callaham, J. The history of Android OS: its name, origin and more. URL: https://www.androidauthority.com/history-android-os-name-789433/, Son Erişim Tarihi: 01.11.2019.

[18] İnternet: Amadeo, R. A History of Pre-Cupcake Android Codenames. URL: https://www.androidpolice.com/2012/09/17/a-history-of-pre-cupcake-android-codenames/, Son Erişim Tarihi: 20.10.2019.

[19] İnternet: SDK Archives. URL: https://developer.android.com/sdk/older_releases.html, Son Erişim Tarihi: 21.10.2019.

[20] İnternet: About the platform. URL: https://developer.android.com/about/versions/10, Son Erişim Tarihi: 19.10.2019.

[21] İnternet: Siemer, R. Which Android runs which Linux kernel?. URL: https://android.stackexchange.com/questions/51651/which-android-runs-which-linux-kernel, Son Erişim Tarihi: 22.10.2019.

[22] İnternet: Cuthbertson, S. Welcoming Android 10!. URL: https://android-developers.googleblog.com/2019/09/welcoming-android-10.html, Son Erişim Tarihi: 20.10.2019.

[23] Kirthika, B., Prabhu, S., Visalakshi, S., "Android Operating System: A Review," *International Journal of Trend in Research and Development*, 2015.

[24] İnternet: Platform Architecture. URL: https://developer.android.com/guide/platform, Son Erişim Tarihi: 25.10.2019.

[25] İnternet: Internet Engineering Task Force(IETF), HTTP over TLS. URL: https://tools.ietf.org/html/rfc2818, Son Erişim Tarihi: 25.10.2019.

[26] Moniruz Zaman, Delwar Alam, Touhid Bhuiyan, Tanjila Farah, "A Study of the Effects of Heartbleed Vulnerability in Bangladesh," in *The Fourth International*

*Conference on Computer Science, Computer Engineering, and Education Technologies (CSCEET2017)*, Bangladesh, 2017.

[27] Calzavara, S., Focardi, R., Nemec, M., Rabitti, A., & Squarcina, M., "ostcards from the Post-HTTP World: Amplification of HTTPS Vulnerabilities in the Web Ecosystem," *IEEE*, p. 0, 2019.

[28] Montealegre, C., Njuguna, C.R., Malik, M.I., Hannay, P., & McAteer, I.N., "Security vulnerabilities in android applications," in *Proceedings of the 16th Australian Information Security Management Conference*, Perth, Australia: Edith Cowan University., 2018.

[29] Tongbo Luo, Hao Hao, Wenliang Du, Yifei Wang, and Heng Yin., " Attacks on WebView in the Android system," in *Proceedings of the 27th Annual Computer Security Applications Conference (ACSAC '11)*, New York, NY, USA, 2011.

[30] İnternet: Android Hacking & Security - Part 9 : Insecure Local Storage. URL: https://resources.infosecinstitute.com/android-hacking-security-part-10-insecure-local-storage/, Son Erişim Tarihi: 15.10.2019.

[31] Nikhil Chiluka ; Ashish Kumar Singh ; Rajesh Eswarawaka, "Privacy and Security Issues Due to Permissions Glut in Android System," in *2018 International Conference on Inventive Research in Computing Applications (ICIRCA)*, Coimbatore, India, 2018.

[32] Anshul Arora, Sateesh K Peddoju, "NTPDroid: A Hybrid Android Malware Detector using Network Traffic and System Permissions," in *2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/ 12th*, New York, NY, USA, 2018.

[33] Hyunwoo Choi, Yongdae Kim, "Large-Scale Analysis of Remote Code Injection Attacks in Android Apps," *Security and Communication Networks*, 2018.

[34] İnternet: A tool for reverse engineering Android apk files. URL: https://ibotpeaches.github.io/Apktool/, Son Erişim Tarihi: 25.10.2019.

[35] İnternet: Burp Suite Community Edition. URL: https://portswigger.net/burp/communitydownload, Son Erişim Tarihi: 22.10.2019.

[36] Cortesi, A., Hils, M., & Kriechbaumer, T., "mitmproxy: A free and open source interactive HTTPS proxy," 2010.

[37] İnternet: Kali Linux Downloads – Virtual Images, Offensive Security. URL: https://www.offensive-security.com/kali-linux-vm-vmware-virtualbox-image-download/, Son Erişim Tarihi: 15.10.2019.

[38] Y.-C. Lin, "Androbugs Frameworks: An Android Application Security Vulnerability Scanner," Blackhat Europe, 2015.

[39] Dalziel, H., & Abraham, A., "Automated Security Analysis of Android and IOS Applications with Mobile Security Framework," *Syngress*, 2015.

[40] İnternet: Android Debug Bridge (adb), Android. URL: https://developer.android.com/studio/command-line/adb, Son Erişim Tarihi: 19.10.2019.

[41] İnternet: Create and manage virtual devices, Android. URL: https://developer.android.com/studio/run/managing-avds, Son Erişim Tarihi: 19.10.2019.

[42] İnternet: Drozer, FSecureLabs. URL: https://github.com/mwrlabs/drozer, Son Erişim Tarihi: 22.10.2019.

[43] İnternet: How mitmproxy works. URL: https://docs.mitmproxy.org/stable/concepts-howmitmproxyworks/, Son Erişim Tarihi: 22.10.2019.

[44] Ankita R Chordiya; Subhrajit Majumder ; Ahmad Y Javaid, "Man-in-the-Middle (MITM) Attack Based Hijacking of HTTP Traffic Using Open Source Tools," in *2018 IEEE International Conference on Electro/Information Technology (EIT)*, Rochester, MI, USA, 2018.

[45] Yang Liu, Chaoshun Zuo, Zonghua Zhang, Shanqing, Guo, Xinshun Xu, "An automatically vetting mechanism for SSL error-handling vulnerability in android hybrid Web apps," *World Wide Web Special Issue on Security and Privacy of IoT*, vol. 1, no. 21, pp. 127-150, 2018.

# ÖZGEÇMİŞ

## Kişisel Bilgiler


Soyadı, adı : CİBALIK, Kaya Emre
Uyruğu : T.C.
Doğum tarihi ve yeri : 16.10.1989, Tokat
Medeni hali : Evli
Telefon : 0 (312) 471 83 50
Faks : 0 (312) 471 83 50
e-mail : kayaemre.cibalik@gazi.edu.tr

## Eğitim

| Derece | Eğitim Birimi | Mezuniyet Tarihi |
|---|---|---|
| Yüksek Lisans | Gazi Üniversitesi / Adli Bilişim | Devam ediyor |
| Lisans | Gazi Üniversitesi / Elektr-Elektronik Müh. | 2011 |
| Lise | Sivas Fen Lisesi | 2006 |

## İş Deneyimi

| Yıl | Yer | Görev |
|---|---|---|
| 2016-Halen | Sağlık Bakanlığı | Bilişim Uzmanı |
| 2014-2016 | Aselsan A.Ş. | Mühendis |
| 2011-2014 | Bilsangrup Ltd. Şti. | Ağ Uzmanı |

## Yabancı Dil

İngilizce,

## Yayınlar

Cibalık K.E., Koçak C., "Kamu Kurumlarının Android Uygulamalarının Güvenlik Zafiyetlerinin İncelenmesi", ISMS 7th International Symposium on Academic Studies in Science, Engineering and Architecture Sciences", Ankara, Turkey, 2019.

## Sertifikalar

Offensive Security Certified Professional (OSCP) – 21.04.2019
CCNA Cyber Ops – 27.03.2018
Certified Ethical Hacker (CEH) - 15.07.2017

*GAZİLİ OLMAK AYRICALIKTIR…*