T.C.
BAHÇEŞEHİR ÜNİVERSİTESİ

ÖDEME SİSTEMLERİNDE MOBİL CİHAZLARDA (ANDROID) GÜVENLİK

Yüksek Lisans Tezi

NİHAT KARABACAK

İSTANBUL, 2019

T.C.
BAHÇEŞEHİR ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ
BİLGİ TEKNOLOJİLERİ

ÖDEME SİSTEMLERİNDE MOBİL CİHAZLARDA (ANDROID) GÜVENLİK

Yüksek Lisans Tezi

NİHAT KARABACAK

Tez Danışmanı: DR. ÖĞR. ÜYESİ AHMET NACİ ÜNAL

İSTANBUL, 2019

T.C.
BAHÇEŞEHİR ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ
BİLGİ TEKNOLOJİLERİ

Tezin Adı: ÖDEME SİSTEMLERİNDE MOBİL CİHAZLARDA (ANDROID) GÜVENLİK
Öğrencinin Adı Soyadı: Nihat Karabacak
Tez Savunma Tarihi: 7 Ağustos 2019

Bu tezin Yüksek Lisans tezi olarak gerekli şartları yerine getirmiş olduğu Fen Bilimleri Enstitüsü tarafından onaylanmıştır.

Dr.Öğr.Üyesi, Yücel Batu SALMAN
Enstitü Müdürü

Bu tezin Yüksek Lisans tezi olarak gerekli şartları yerine getirmiş olduğunu onaylarım.

Prof. Dr. Mehmet Alper TUNGA
Program Koordinatörü

Bu Tez tarafımızca okunmuş, nitelik ve içerik açısından bir Yüksek Lisans tezi olarak yeterli görülmüş ve kabul edilmiştir.

Jüri Üyeleri	İmzalar
Tez Danışmanı Dr. Öğr. Üyesi Ahmet Naci ÜNAL	----------
Üye Dr. Öğr. Üyesi Serkan AYVAZ	----------
Üye Dr. Öğr. Üyesi Tayfun ACARER	----------

ÖZET

ÖDEME SİSTEMLERİNDE MOBİL CİHAZLARDA (ANDROID) GÜVENLİK

Nihat KARABACAK

Bilgi Teknolojileri

Tez Danışmanı: Dr. Öğr. Üyesi. Ahmet Naci ÜNAL

Ağustos 2019, 55 Sayfa

Bu çalışma Android işletim sisteminin güvenliği ile ilgili özellikle ödeme sistemlerinde kullanımı konusunda araştırma, tasarım ve uygulamalı olarak çözüm üretmek amacıyla hazırlanmıştır.

Android, en yaygın kullanılan mobil işletim sistemidir. İlk yayınlandığı zaman, asıl amacı akıllı telefonlarda çalıştırmak olan Android işletim sistemi günümüzde bu amacı aşarak, birçok farklı türde cihaz üzerinde çalışır hale gelmiştir. Sadece akıllı telefonlar ve tabletler değil, akıllı saatler, akıllı TV'ler, endüstriyel sistemler ve hatta otonom araçlar gibi pek çok sistem Android tarafından yönetilir hale gelmiştir.

Konuşma ve mesajlaşmanın yanı sıra, internete erişmek, sosyal medya kullanmak, e-posta almak ve göndermek, oyun oynamak, multimedya içeriği oynatmak, çevrimiçi alışveriş, seyahat erişimi, çevrimiçi bankacılık, çevrimdışı ödeme sistemleri için modern akıllı telefonlar kullanılmaktadır. Bu gelişimin bir sonucu olarak ta, Android platformu kötü niyetli yazılımlar geliştiren,yeni güvenlik açıkları arayan suçlular ve gizli servisler için çok değerli bir hedef haline gelmiştir.

Bu tez özellikle ödeme sistemlerinde kullanılmaları nedeniyle Android akıllı telefonlara odaklanacak ve güvenli bir yazılım tasarımı üzerine tespitler, öneriler, tasarım ve uygulamada bulunacaktır .

Anahtar Kelimeler: Android, Güvenlik, Ödeme Sistemleri

ABSTRACT

SECURITY AT MOBILE DEVICES (ANDROID) IN PAYMENT INDUSTRY

Nihat KARABACAK

Information Technologies

Thesis Supervisor: Assist. Prof. Dr. Ahmet Naci ÜNAL

August 2019, 55 Pages

This study has been prepared for research, design and practical solutions related to the security of Android operating system especially in the use of payment systems.

Android is the most widely used mobile operating system. When it was first released, the Android operating system, whose main purpose was to run on smartphones, has now surpassed this goal and has been running on many different types of devices. Not only smartphones and tablets, but many systems such as smart watches, smart TVs, industrial systems and even autonomous vehicles have become Android-managed.

In addition to speaking and messaging, modern smartphones are used for accessing the internet, using social media, receiving and sending emails, playing games, playing multimedia content, online shopping, travel access, online banking, and offline payment systems. As a result of this development, the Android platform has become an invaluable target for criminals and secret services that develop malicious software, looking for new vulnerabilities.

This thesis will focus on Android smartphones especially because they are used in payment systems and will make determinations, recommendations, design and application on secure software design.

Keywords: Android, Security, Payment Systems

İÇİNDEKİLER

TABLOLAR

ŞEKİLLER

KISALTMALAR

AAPT : Android Asset Packaging Tool
AAR : Android Application Record
AES : Advanced Encryption Standard
AIDL : Android Interface Definition Language
AOT : Ahead-of-Time Compilation
API : Application Programming Interface
APK : Android Package Kit
ART : Android Run Time
DES : Data Encryption Standard
DEX : Dalvik Executable
ECC : Elliptic Curve Cryptograpy
GC : Garbage Collection
GCM : Google Cloud Messaging
HAL : Hardware Abstraction Layer
HCE : Host Card Emulation
HSM : Host Security Module
IPC : Inter Process Communication
ISO : International Standard Organisation
JIT : Just-in-Time Compilation
JNI : Java Native Interface
MAC : Message Authentication Code
MAP : Mobile Application Platform
MPA : Mobile Payment Application
NFC : Near Field Communication
RNS : Remote Notification Service
RSA : Rivest-Shamir-Adleman PKI
SDK : Software Development Kit
SE : SecureElement
SQL : Structured Query Language
SSL : Secure Socket Layer

TEE : Trusted Execution Environmnt

TLS : Transport Layer Security

TOE : Target Of Evaluation

TPM : Trusted Platform Module

UI : User Interface

WBC : WhiteBox Cryptograpy

1. GİRİŞ

Android sözcüğünün kelime anlamı ‘insan şeklindeki robot’ olarak verilmektedir. Bilgisayar terimi olarak ise ‘Cep telefonları için geliştirilen açık kaynak kodlu bir platform’ anlamındadır. Oxford English Dictionary ise android sözcüğünü ‘an automaton resembling a humanbeing’ yani ‘insanı andıran otomasyon’ olarak açıklamaktadır. Genel olarak ‘mobil telefon ve tabletlerde kullanılan bir işletim sistemi’ olarak tanımlanabilir.

Android, Google ve Open Handset Alliance tarafından geliştirilen bir işletim sistemidir. Kurucuları Andy Rubin, Rich Miner, Nick Sears ve Chris White'dır. Linux tabanlı açık kaynak kodlu ve ücretsiz bir işletim sistemidir (Heger, 2012). Uygulamaları Google Play Store üzerinden indirilmektedir. Birçok farklı sürümü mevcuttur. Her yeni sürümle birlikte cihazların yeni özelliklerini kullanmak pratikleşip, kolaylaşmakta ve daha güvenli hale gelmektedir.

İlk yayınlandığı zaman, asıl amacı akıllı telefonlarda çalıştırmak olan Android işletim sistemi günümüzde bu amacı aşarak, birçok farklı türde cihaz üzerinde çalışır hale gelmiştir. Sadece akıllı telefonlar ve tabletler değil, akıllı saatler, akıllı TV'ler, endüstriyel sistemler ve hatta otonom araçlar gibi pek çok sistem Android tarafından yönetilir hale gelmiştir.

Konuşma ve mesajlaşmanın yanı sıra, internete erişmek, sosyal medya kullanmak, e-posta almak ve göndermek, oyun oynamak, multimedya içeriği oynatmak, çevrimiçi alışveriş, seyahat erişimi, çevrimiçi bankacılık, çevrimdışı ödeme sistemleri için modern akıllı telefonlar kullanılmaktadır. Bu gelişimin bir sonucu olarak ta, Android platformu kötü niyetli yazılımlar geliştiren ve yeni güvenlik açıkları arayan suçlular ve gizli servisler için çok değerli bir hedef haline gelmiştir. Birçok Android cihaz düşük sürümlerde çalışır çünkü çoğu üretici yalnızca 18 ay ile 24 ay arasında destek sağlamaktadır. Bu destek süresinden sonra, artık belirli cihazlar için son Android sürümlerinin resmi bağlantı noktaları sağlanmamaktadır. Son Android sürümlerini temel alan çeşitli özel ROM'lardan birini kullanmak mümkün olsa da, çoğu kullanıcı bunları cihazlarına yükleyecek kadar teknolojik olarak bilgilendirilmemektedir.

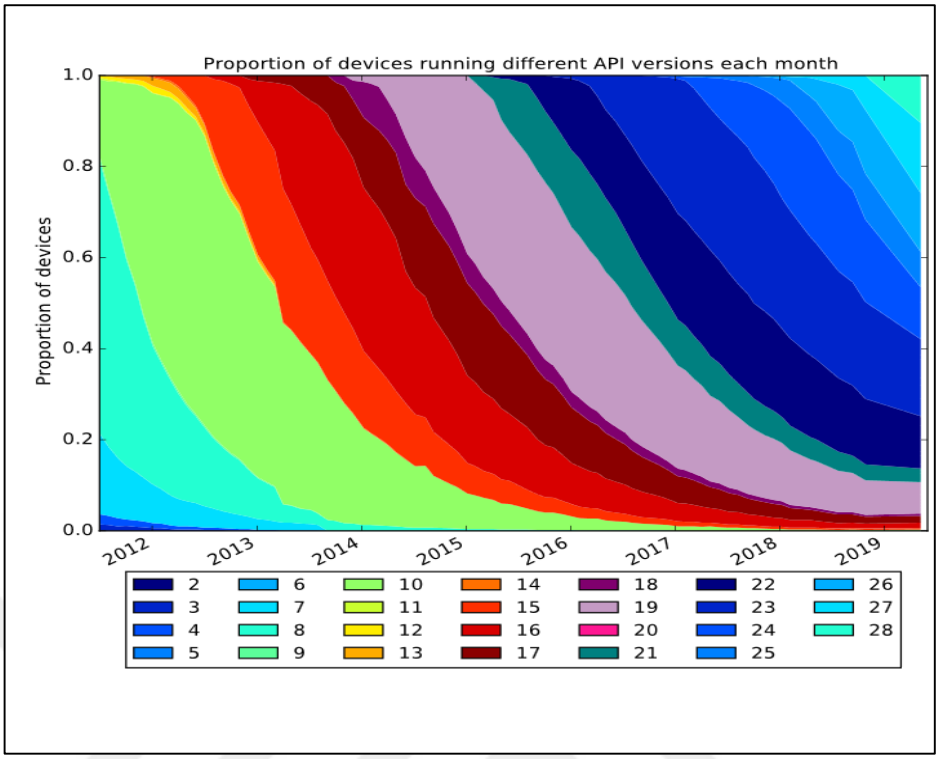
Android sürümlerinin bu parçalanması, güvenlikle ilgili güncellemelerle ilgili büyük bir sorundur, çünkü Google, Nexus aygıtları dışında doğrudan güvenlik düzeltmeleri sağlamamaktadır. Android işletim sistemi Kasım 2007' da yayınlanan Beta 1.0 sürümünden bugüne düzenli olarak güncellenerek, bugün itibari ile Ağustos 2018' de yayınlanan 9.0 Pie sürümüne ulaşmıştır. Android sürümlerinin dağılımı ve kod adları **Tablo 1.1** de verilmiştir (Dashboard, 2019).

Belirli bir sürümü belirtmenin farklı yolları vardır. Kod adı, gerçek Android sürümü veya Uygulama Programlama Arabirimi (API) sürümünü kullanmak mümkündür. KitKat ve düşük sürümleri genellikle desteklenmemektedir, bu da güvenlikle ilgili daha fazla güncelleme bulunmadığı anlamına gelmektedir. Bu nedenle dünya çapında kullanılan tüm cihazların% 50'den fazlası çeşitli saldırılara açıktır. Eski Android sürümlerindeki güvenlik açıklarının bile güncel olmasının nedeni budur. Pazardaki Android API Dağılımı **Şekil** 1.1**'** de verilmiştir (Vulnerabilities, 2015).

Tablo 1.1: Android sürümleri ve pazar oranları

Versiyon	Kod Adı	API	Dağıtım
1.0	Base	1	
1.1		2	
1.5	Cupcake	3	
1.6	Donut	4	
2.0 - 2.1	Eclair	5	
2.2 - 2.2.2	Froyo	8	
2.3	Gingerbread	10	0.3%
4.0.3 4.0.4	Ice Cream Sandwich	15	0.3%
4.1.x	Jelly Bean	16	1.2%
4.2.x		17	1.5%
4.3		18	0.5%
4.4	KitKat	19	6.9%
5.0	Lollipop	21	3.0%
5.1		22	11.5%
6.0	Marshmallow	23	16.9%
7.0	Nougat	24	11.4%
7.1		25	7.8%
8.0	Oreo	26	12.9%
8.1		27	15.4%
9	Pie	28	10.4%

Şekil 1.1 : Android API kullanım dağılımı

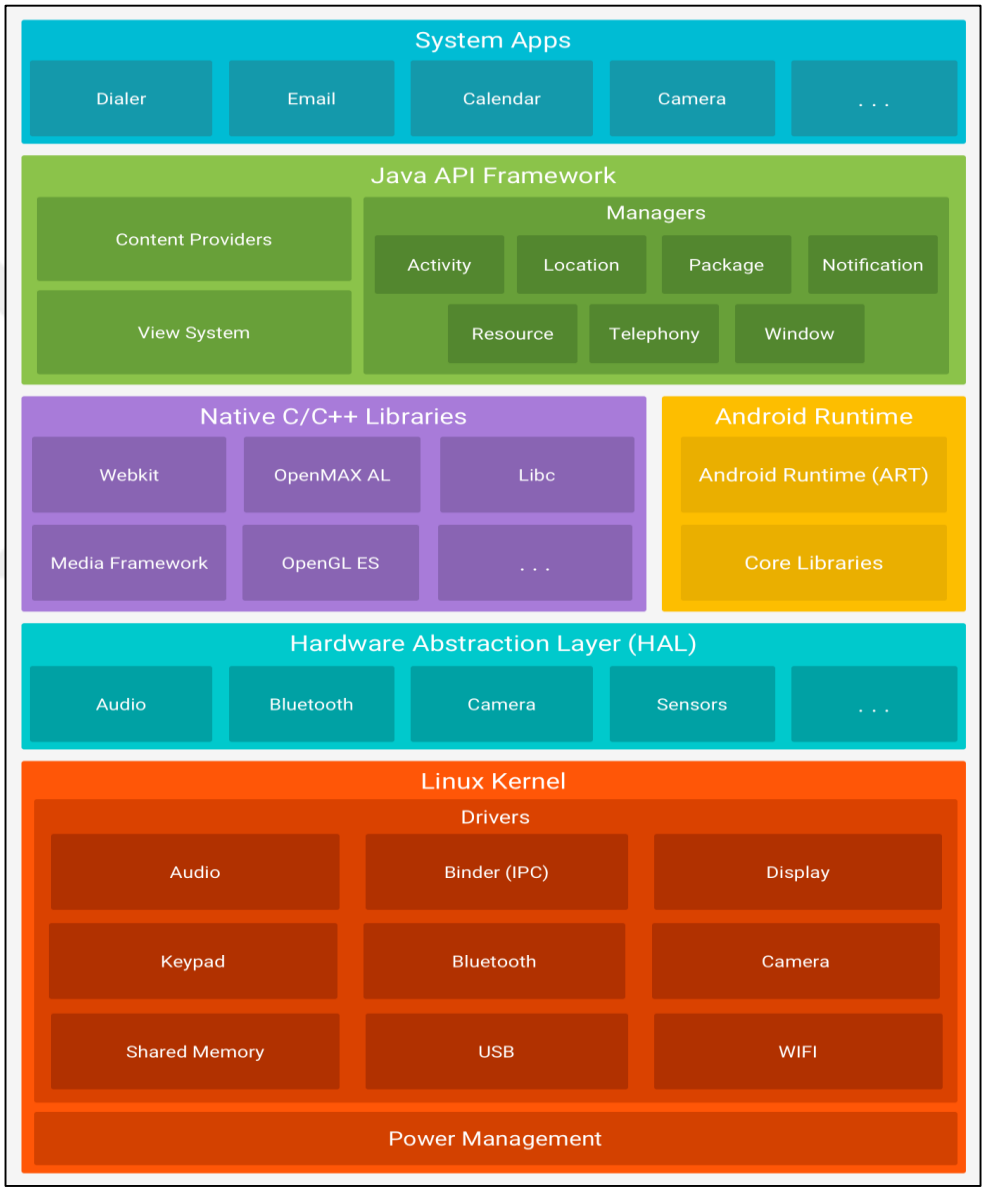


2. LİTERATÜR TARAMASI

2.1 ANDROID PLATFORMU

Android, çok çeşitli aygıtlar ve form faktörleri için oluşturulmuş, açık kaynaklı, Linux tabanlı bir yazılım yığınıdır. **Şekil 2.1** Android platformunun ana bileşenlerini göstermektedir (Dashboard, 2019).

Şekil 2.1 : Android yazılım yığını



2.1.1 Linux Çekirdeği

Android platformunun temeli Linux çekirdeğidir. Örneğin, Android Çalışma Zamanı

(ART), iş parçacığı ve düşük seviye bellek yönetimi gibi işlevselliklerin altında yatan Linux çekirdeğine güvenir. Bir Linux çekirdeğini kullanmak, Android'in temel güvenlik özelliklerinden yararlanmasını ve cihaz üreticilerinin iyi bilinen bir çekirdek için donanım sürücüleri geliştirmelerini sağlar. (Dashboard, 2019)

2.1.2 Donanım Soyutlama Katmanı

Donanım soyutlama katmanı (HAL), cihaz donanım yeteneklerini üst seviye Java API çerçevesine maruz bırakan standart arayüzler sağlar. HAL, her biri kamera veya bluetooth gibi belirli bir donanım bileşeni için bir arayüz uygulayan çoklu kütüphane modüllerinden oluşur. Bir çerçeve API cihaz donanımına erişmek için bir çağrı yaptığında, Android sistemi bu donanım bileşeni için ilgili kütüphaneyi yükler. (Dashboard, 2019)

2.1.3 Android Çalışma Zamanı

Android Çalışma Zamanı Modülü (ART), en az bellek alanı için optimize edilmiş Android için özel olarak tasarlanmış bir bytecode formatı olan DEX dosyalarını çalıştırarak, düşük bellek aygıtlarında birden fazla sanal makineyi çalıştırmak için yazılmıştır. Jack gibi araç zincirleri ile Java kaynakları Android platformunda çalışabilen DEX bayt kodunda derlenir.

ARTJIT yerine AOT kullandığından daha Dalvik'e göre hızlı olup aynı zamanda optimize edilmiş çöp toplama (GC) özelliğini de barındırmaktadır (Dashboard, 2019) .

2.1.4 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 yapılmıştır. Android platformu, bu yerel kütüphanelerin bazılarının işlevselliğini uygulamalara sunmak için Java framework API'leri sağlar (Dashboard, 2019).

Yerel kütüphanelerin ARM, x86 ve MIPs mimarileri için özel olarak geliştirilmesi ve derlenmesi gerekir.X86 işlemcilerle karşılaştırıldığında, ARM donanım mimarisi X86 mimarisine göre düşük güç tüketimine sahip olduğundan mobil cihazlar için daha uygundur (Maker, 2010).

Linux ile karşılaştırıldığında, Android kendi c kütüphanesini (Bionic) içermektedir

(Brady, 2008). Bu kütüphane glibc ile uyumlu değildir. Her bir iş parçacığının bellek tüketimini optimize eden ve yeni bir iş parçacığının başlangıç süresini azaltan özel bir iş parçacığı uygulaması içerir (Liang, 2010).

2.1.5 Java API Framework

Android işletim sisteminin tüm özellik kümeleri Java dilinde yazılmış API'ler aracılığıyla kullanılabilir. Bu API'ler Android uygulamaları oluşturmak için gereken yapı taşlarını oluşturur. Geliştiriciler, Android sistem uygulamalarının kullandığı aynı çerçeve API'lerine tam erişime sahiptir. (Dashboard, 2019)

2.1.6 Sistem Uygulamaları

Android, e-posta, SMS mesajları, takvimler, internet taraması, kişiler ve daha fazlası için bir dizi temel uygulama ile birlikte gelir. Platforma dâhil olan uygulamalar, kullanıcının yüklemeyi seçtiği uygulamalar arasında özel bir duruma sahip değildir.

Sistem uygulamaları hem kullanıcılar için uygulamalar olarak işlev görür, hem de geliştiricilerin kendi uygulamalarından erişebilecekleri temel yetenekler sunar (Dashboard, 2019) .

2.2 ANDROID UYGULAMA MİMARİSİ

Android Uygulama Mimarisi ile ilgili özet bilgi bu bölümde verilmiştir.

2.2.1 Mobil Uygulama

Mobil uygulama, masaüstü veya dizüstü bilgisayarlar yerine telefon, tablet, saat, bileklik gibi taşınabilir kablosuz iletişim ve disk kapasitesi bulunan akıllı cihazlar kullanılmak üzere geliştirilen yazılımdır. Mobil uygulamalar, cihazların sahip oldukları özellikleri ve yeteneklerinden yararlanmak için tasarlanmıştır. Mobil uygulamalar belirli bir platform için özel olarak oluşturulmuş ve kategorize edilmiştir. Uygulamalar aşağıdaki üç şekilde kategorize edilmektedir;

i. Web tabanlı uygulamalar
ii. Native uygulamalar
iii. Native ve Web Hibrit uygulamalar

2.2.2 Uygulama Bileşenleri

Android uygulamalarını geliştirmede kullanılan temel bileşenler şunlardır:

i. **Activity:** Android uygulamaların kullanıcı arayüzüyle ilgili bileşendir (Avtivity, 2019).

ii. **Service:** Arka planda yapılması gereken işlemleri gerçekleştiren bileşendir (Service, 2019).

iii. **Broadcast Receiver:** Sistemden ya da diğer uygulamalardan gelen mesajları dinleyen ve yanıt veren bileşenlerdir (Broadcast, 2019).

iv. **Content Providers:** Uygulamalar arası veri iletiminde kullanılan bileşenlerdir (Content, 2019).

v. **Fragment:** Aktivitelerden daha basit olan ancak aktivite gibi tek başlarına kullanılamayan bileşenlerdir (Fragment, 2019).

vi. **Views:** Ekrana çizdirilebilen ve kullanıcı etkileşimini sağlayan arayüz bileşenleridir (View, 2019).

vii. **Intent:** Bileşenler arası iletişimi sağlayan bileşen sınıfıdır (Intent, 2019).

viii. **Resources:** Uygulamada kullanılacak olan görseller, metinler, temalar ve animasyonların saklandığı bileşendir (Resources, 2019).

ix. **AndroidManifest:** Uygulama ismi, uygulama ikonu, uygulama içerisinde kullanılan aktiviteler, uygulamanın ihtiyaç duyacağı izinler, kullanılan servisler, kullanılan yayın alıcılar, intent öncelikleri, intent filtreleri, paket ismi gibi bilgiler içeren bir yapılandırma dosyasıdır (Manifest, 2019).

2.2.3 İzinler

Android uygulamaları belirli işlemleri yapabilmek için kullanıcıdan izin talep etmek zorundadır (Permissions, 2019). Bu izinlerkötü amaçlı yazılımlar tarafından kullanıldığı gibi kötü amaçlı yazılımların tespitinde de kullanılabilmektedir. İzinler dört gruba ayırlmaktadır:

i. **Normal izinler:** Uygulamanın, kulanıcının gizliliği ya da diğer uygulamaların güvenliği açısından risk teşkil etmeyen izinlerdir. Basit ve tehlikesiz izinler olup kullanıcı onayına tabi değillerdir.

ii. **Tehlikeli izinler:** Uygulamanın kullanıcıların özel bilgilerini içeren verilere veya

kaynaklara ihtiyaç duyan işlemleri içeren izinlerdir. Bu izinlerle kullanıcının kayıtlı verileriveya diğer uygulamaya ait dosyalar silinebilir, kopyalanabilir ve değiştirebilir. Kullanıcıya izin onay ekranı gösterilerek onayı alınır.

iii. **Özel izinler:** Normal ve tehlikeli izinlere benzemektedir. Tehlikeli olabilecek özelliklere sahip olan ve az kullanılan izinlerdir. Bu izinlere ihtiyaç duyulduğunda tehlikeli izinlerde olduğu gibi kullanıcıya çalışma zamanında izin talebinde bulunmalıdır.

iv. **İmza izinler:** Uygulamanın istediği imza iznin sistem tarafından otomatik olarak verilebilmesi için, aynı izne sahip ve aynı sertifika ile imzalanmış olan başka bir uygulamaya bu iznin verilmiş olması gerekmektedir. Sistem tarafından otomatik olarak verildiğinden bazı suistimallere sebep olabilmektedir.

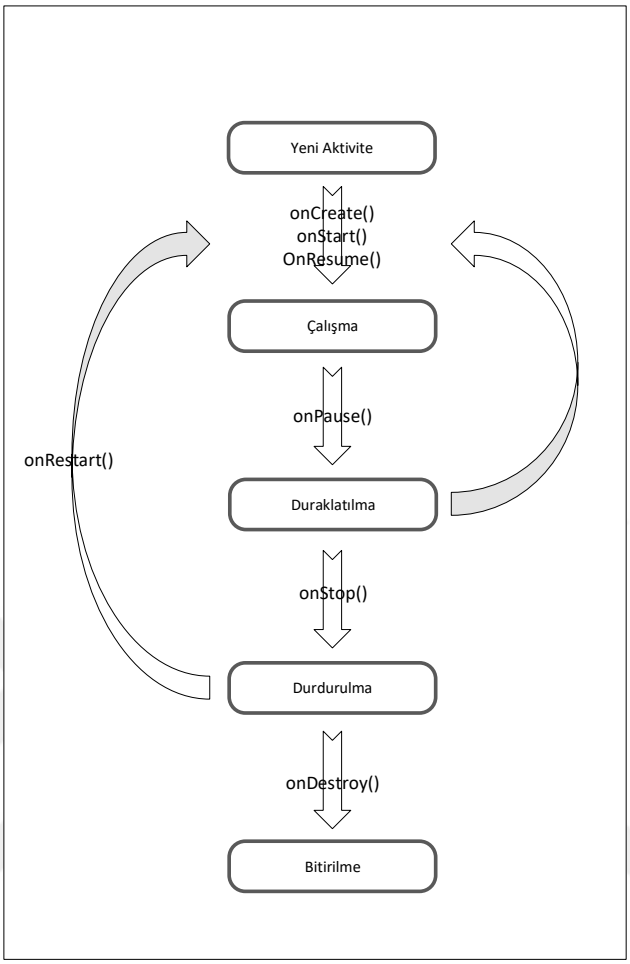
2.2.4 Yaşam Döngüsü

Android işletim sistemi, uygulamaları farklı dalvik sanal makinesi üzerinde farklı proseslerde çalıştırır. Android uygulamalarının dört temel durumu vardır.

i. Çalışma durumu (Running)
ii. Duraklatılmı durumu (Paused)
iii. Durdurulma durumu (Stop)
iv. Çalışma durumu (Destroyed).

Android uygulama yaşam döngüsü **Şekil 2.2** de gösterilmiştir.

Şekil 2.2 : Android uygulama yaşam döngüsü



Uygulama ilk çalıştığında onCreate(), onStart(), onRessume() metotları çalışarak uygulama Çalışma durumuna geçer.

Uygulamada kesinti durumu oluştuğunda onPause() metodu çalışır. Bu durumda bir süre bekledikten sonra onStop() metodu çalışarak Durdurulma durumuna geçer veya onResume() metodu ile Çalışma durumuna geçer.

Durdurulma durumundan onRestart(), onStart() ve onResume() metotları çalışarak Çalışma durumuna geçer veya onDestroy() metodu çalışarak Çalışma durumuna geçer ve uygulama öldürülür.

2.2.5 Uygulama Geliştirme Araçları

Android uygulamaları için Google tarafında resmi olarak desteklenen geliştirme dilleri Java, Kotlin ve C/C++ dilleridir. Geliştirme ortamı olarak yine Google tarafından geliştirilen olan Android Studio kullanılması önerilmektedir. Her durumda bir uygulamanın mağazada yayınlanabilmesi için mutlakayayın anahtarlarıyla imzalanması gerekmektedir (Release, 2019), Google Play hata ayıklama anahtarı imzasına sahip olan uygulamaları yayın ortamına kabul etmemektedir. Derleme sonucunda ortaya çıkan Android uygulaması uzantısı. apk şeklinde olan sıkıştırılmış bir dosyadır.

Bir uygulamayı derlemek ve imzalamak için sırasıyla aşağıdaki aşamalar takip edilir:

i. AAPT aracı projede yer alan bütün dosyalar bir araya getirilir.
ii. AIDL, IPC kullanarak haberleşmeyi sağlayan java arabirimlerini uygun formata çevirir.
iii. Java derleyicisi tüm java ve AIDL dosyalarını derler ve class dosyalarını oluşturur.
iv. Sınıf dosyaları üçüncü parti yazılımlarla beraber DEX aracı ile (dx komutu) Dalvik baytkodlarına çevrilir ve dex dosyasını oluşturur.
v. APKBuilder aracı, DEX dosyasını, derlenmiş kaynakları ve resim gibi derlemeye gerek olmayan kaynakları bir APK dosyası halinde sıkıştırır.
vi. APK dosyası cihaza yüklenmeden önce yayın veya hata ayıklama anahtarı ile imzalanır.
vii. Eğer APK dosyası yayın anahtarıyla imzalanmışsa ZIPAlign aracı kullanılır.

Android Studio gibi araçlar bütün bu aşamaları otomatik olarak gerçekleştirebilmektedir.

2.2.5.1 APK

Android Package Kit (APK), Android uygulamalarını dağıtmakta ve yüklemekte kullanılan dosya formatıdır. Bu dosyacihaza yüklemeye yapmak için gerekli tüm öğeleri barındırır. Google Play'den uygulama seçilerek, APK dosyası otomatik olarak indirilip yüklenebildiği gibi, APKPure (APKPure, 2019) veya APKMirror (APKMirror, 2019) gibi alternatif uygulama mağazalarından da edinebilmektedir. Google Play haricinde kanallardan veya elle yüklemekbazı avantajlara sahip olsa da aynı zamanda çok ciddi tehditler de içermektedir.

2.2.5.2 AAR

Android Application Record (AAR), yapısal olarak bir APK ile aynı olan derlenmiş bir kütüphane dosyasıdır. JAR dosyalarından farklı olarak içerisinde kaynak kod, kaynak dosyalar ve bildirimler dahil olmak üzere bir uygulama oluşturmak için gereken her şeyi içerebilir. Ancak, tek başına çalışmaz, projeye kütüphane olarak eklenerek kullanılır (Developer, 2019).

2.3 ANDROID VERİ SAKLAMA YÖNTEMLERİ

Android, uygulama verilerinin kaydedilmesiiçin çeşitli seçenekler sunar. Seçilecek depolama çözümü, verilerinbüyüklüğü, türü ve erişim özelliklerine göre değişebilir. Bu seçenekler şu şekilde sıralanabilir:

i. Dahili dosya depolama: Uygulamaya dosyalarının cihaz dosya sisteminde saklanması.
ii. Harici dosya depolama: Dosyaların harici dosya sisteminde saklanması.
iii. Paylaşılan tercihler: Özel ilkel verilerin anahtar / değer çiftlerinde saklanması.
iv. Veritabanları: Yapılandırılmış verilerin özel bir veritabanında saklanması.

Harici depolama dışındaki seçeneklerin uygulamaya özel seçenekler olup veriler diğer uygulamalartarafından erişilemez (Data Storage, 2019) dense de köklenmiş sistemlerde bu geçerli değildir.

2.3.1 SQLite

Veri depolama seçeneklerinde SQLite, kendi kendine yeten, sunucusuz, sıfır konfigürasyonlu, işlemsel bir SQL veritabanı motorunu uygulayan süreç içi bir kütüphanedir. SQLite kodu kamuya açık alandadır ve bu nedenle ticari veya özel herhangi bir amaç için ücretsizdir. SQLite, birçok yüksek profilli proje de dahil olmak üzere, güvenebileceğimizden daha fazla uygulamaya sahip, dünyanın en yaygın veritabanıdır (SQLite, 2019).

SQLite gömülü bir SQL veritabanı motorudur. Diğer birçok SQL veritabanının aksine, SQLite ayrı bir sunucu işlemine sahip değildir. SQLite doğrudan sıradan disk dosyalarını okur ve yazar. SQLite'yi Oracle / MSSQL yerine değil, fopen () / fwrite() / fread()

şeklinde düşünmek daha doğru olacaktır.

Başta SQLite olmak üzere, bütün depolama seçeneklerinde uygulama geliştiriciler verilerin güvenli olarak saklandığına inanarak önlem almamaktadır. Oysa köklenmesine bile gerek kalmadan bu veriler alınabilmektedir. Dolayısı ile her türlü veri saklama seçeneğinde verilerin şifrelenerek saklanması gerekmektedir. Sadece verilerin değil etiketlerin, tablo ve kolon adlarının da okunduğunda anlamsız olması güvenlik açısından çok önemlidir.
Yerel veritabanı kullanımlarını, geliştiriciler ara katman olarak şifreleme / şifre çözme özellikleri ekleyerek şifreli yerel veritabanı kullanımı sağlayarak gerçekleştirmelidir.

2.4 ANDROID GÜVENLİĞİ

İnternette gezinmek, sosyal ağ sitelerini güncellemek ve online alışveriş ve bankacılık işlemlerini yapmak için akıllı telefonlarını ve tabletlerini kullanan insanların sayısı arttıkça siber suçlular ve kötü amaçlı yazılımlar da yeni akıllı telefon tehditleri ve mobil tehditlerle mobil cihazları gün geçtikçe daha fazla hedeflemeye başlamıştır.

Modern uygulamalar genellikle saldırıya açık olduğu bilinen açık cihazlarda çalışır. Bu nedenle cihazlara uygulama geliştiricileri tarafından güvenilemez. Bu, geliştiricilerin, aygıt ortamından bağımsız olarak güvenli uygulamalar oluşturabilmesini gerekli kılmaktadır.
Bu gerekliliğin bir sonucu olarak, uygulamaların kendilerini korumalarını sağlayan araçlar, teknolojiler ve metodolojiler kullanmasızorunlu hale gelmektedir.

2.4.1 Saldırıların Hedefi Neden Android?

Günümüzde Google Play resmi uygulama mağazasında 2019 yılının ilk çeyreği itibariyla 2,1 milyon uygulama bulunmaktadır. Bunu yaklaşık 1,8 milyon uygulama ile Apple App Store izlemektedir (Statistics, 2019). 2019 yılında Sidney Üniversitesince yapılan çalışmada, Play Store da bulunan 1.000.000 uygulama içinde 2.040 potansiyel zararlıuygulama bulunurken, 40.000 uygulamada ise popüler uygulamalarla tehlikeli benzerlikler tespit edilmiştir. (Sydney, 2019) F-secure'un 2014 raporuna göre ise zararlı yazılımların %99'u Android platformunu hedef almaktadır (FSecure, 2014). Benzer

şekilde Kaspersky Lab tarafından algılanan tüm mobil kötü amaçlı yazılımların %99'unun Androidi hedefleyecek şekilde tasarlandığı tespit edildiği belirtilmiştir. (Kaspersky, 2016)

Android'i hedefleyen kötü amaçlı yazılımların sayısındaki bu yüksek artışın nedenleri şunlardır:

i. Android platformu, yeni akıllı telefonlarda en yaygın kullanılan işletim sistemi haline gelerek %80'in üzerinde pazar payına sahip olması.
ii. Android işletim sisteminin açık niteliği, uygulama geliştirme kolaylığı ve (resmi olmayan) uygulama pazarlarındaki geniş çeşitlilik.

2.4.2 Kötü Amaçlı Yazılımlar

Kötü amaçlı yazılımlar kullanıcının kaynaklarını kendi hedefleri için kullanmak amacıyla tasarlanmış yazılımlardır.

2.4.3 Kötü Amaçlı Yazılımların Hedefleri

Kötü amaçlıyazılımlar dört temel hedefe sahiptir. Kapsamına göre bir kötü amaçlı bir yazılımda bu hedeflerden sadece biri, hepsi ya da bazıları bir arada bulunabilir. Bunlar :

iii. Cihaz üzerinde yetki kazanımı
iv. Cihazı uzaktan kontrol edilebilme
v. Maddi kazanç sağlama
vi. Bilgi toplama

2.4.4 Kötü Amaçlı Yazılım Türleri

Günümüzde birçok farklı kategoride kötü amaçlıy azılımlar üretilmektedir. Android akıllı telefonlarda algılanan en yaygın kötü amaçlı nesneler, başlıca yedi gruba ayrılabilir:

i. **Spyware & Adware:** Amacı kişisel bilgileri (rehber, mesajlar, IMEI numarası vs.) alıp üçüncü taraflarla paylaşmak olan yazılımlardır. Bu taraflar bazen reklam şirketleri de olabilir. Bu yüzden bazı yerlerde Adware olarak ta geçerler (Felt, 2011).
ii. **Trojan ve Virüsler:** Zararsız uygulamalara entegre olarak cihazı kötü amaçları için kullanana zararlı yazılımlardır. Zararsız görünen bir uygulama yoluyla cihaza girerler. Uygulama kendi normal işlevlerini yerine getirirken Trojan / Virüs genellikle cihazı kökleyerek hassas dosyalara ve cihaz belleğine erişebilirler (Felt, 2011) (Arshad, 2016).

iii. **Phishing Uygulamaları:** Masaüstü uygulamalarına benzer şekilde, mobil cihazlar için üretilen zararsız görünümlü zararlı uygulamalardır. Genellikle bilinen bir uygulama ile aynı görünüme sahip olup arka planda kullanıcıbilgilerinin çalınması hedeflenir.

iv. **Mobil Botnet:** Çok sayıda birbirine bağlı zararlı yazılımdan oluşan ve haberleşmeyi bir C&C üzerinden gerçekleştirirler. Genellikle siber saldırı türü kötü amaçlı amaçlarla kullanılırlar (Ismail, 2017).

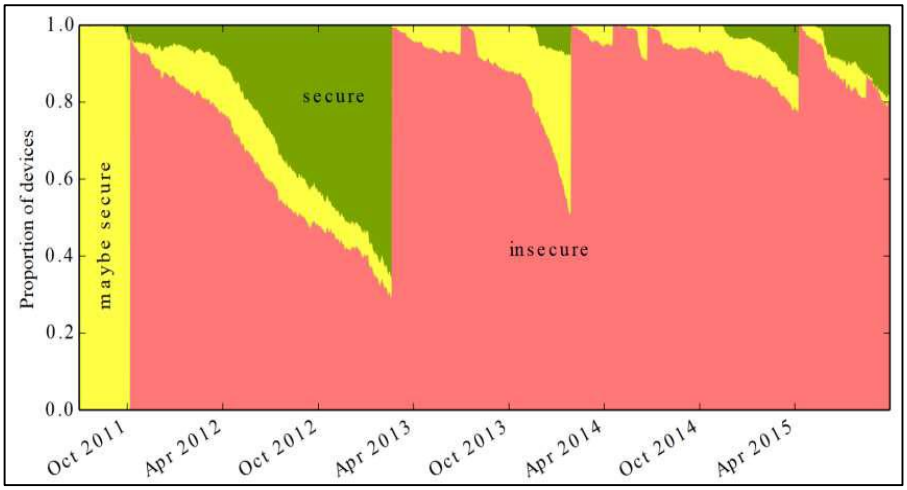
v. **Rootkit ve Backdoor:** Bulaştığı cihazlarda süper kullanıcı izni almaya çalışan ve böylece cihaz üzerinde sınırsız yetkiye sahip olmayı amaçlayan yazılımlardır (Ismail, 2017). Bunun sonucunda cihazdaki varsa antivirüs yazılımları da dahil olmak üzere bütün uygulamaları durdurabilir, silebilir ve yeni yazılımlar indirebilirler. Rootkitler sayesinde cihaz uzaktan yönetilebilir ve istenen işlemler kullanıcıya farkettirmeden gerçekleştirilebilir (Arshad, 2016).

vi. **Ransomware ve Cryptolocker:** Bulaştığı cihazlarda genellikle önemli belgeleri ya da cihazın tamamını yalnızca kendisi tarafından çözülebilecek şekilde şifreleyen ve şifrenin çözülmesi için kullanıcıdan finansal karşılı talep eden, bu talep karşılanmadığında verileri silen yazılımlardır (Arshad, 2016).

vii. **Cryptominer:** Bulaştığı cihazlarda kullanıcının bilgisi dışında, özellikle cihazın işlemcisini zincir madenciliği için kullanan yazılımlardır. Cihazın yavaşlaması ve pil ömrünün azalması gibi yan etkileri vardır.

Kaspersky (Kaspersky, 2016), McAfee (McAfee, 2018) ve Avast (Avast, 2019)gibi güvenlik firmaları yıllık raporlarında, Google Play mağazasında bile kötü amaçlı programların bulunduğunu ve yıllar içinde değişen zararlı yazılım türlerini, saldırı eğilimlerini belirtmektedir. **Şekil** 2.3 Cambridge Universitesinde yapılan bir araştırma sonucunda ortaya çıkan, sahada güvensiz çalışan Android cihazların oranının tahminini göstermektedir (Vulnerabilities, 2015).

Şekil 2.3 : Güvensiz çalışan Android cihazların tahmini oranı

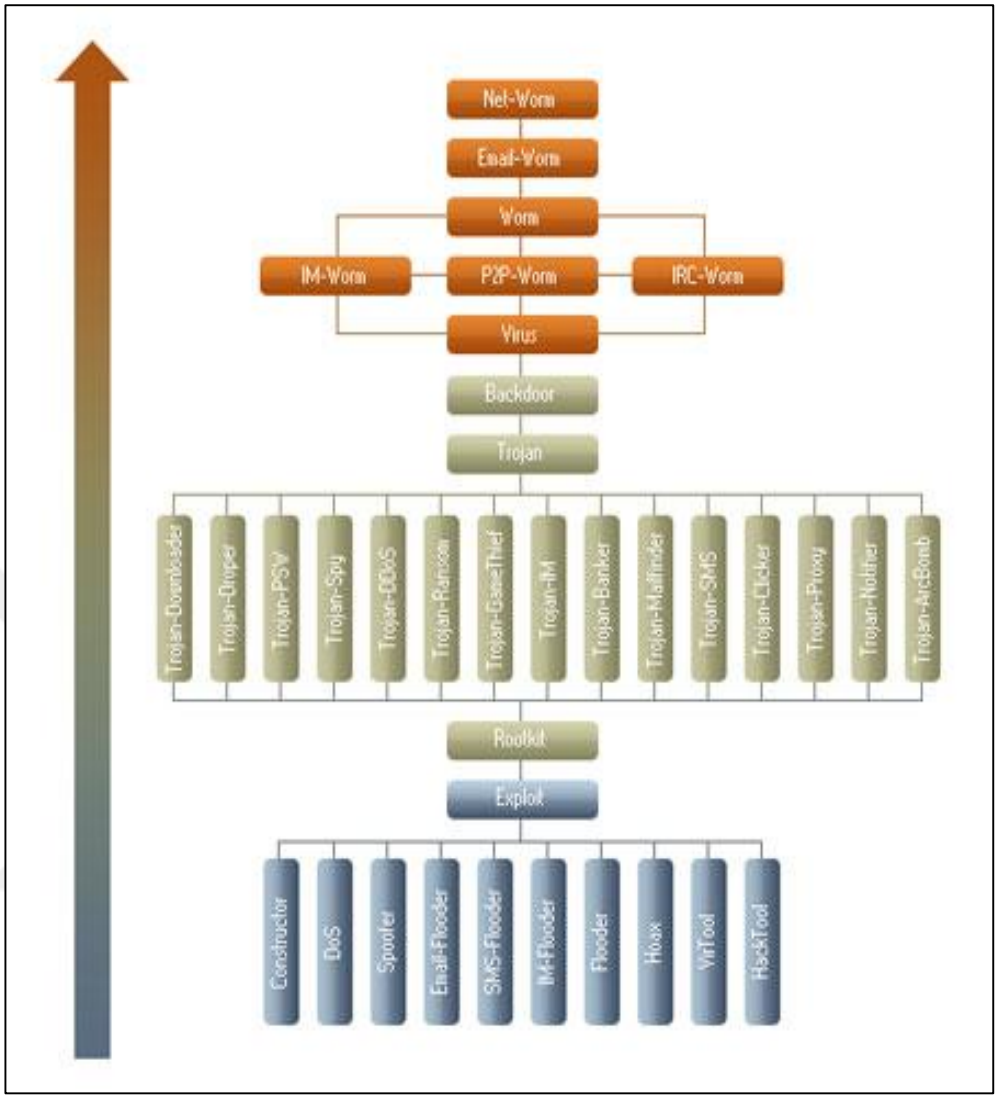


2.4.5 Kötü Amaçlı Yazılım Sınıflandırma Ağacı

Kaspersky Lab, kötü amaçlı yazılım öğelerini, kullanıcıların etkinliklerine göre sınıflandırarak tüm kötü amaçlı yazılım çeşitlerini ve Kaspersky'nin antivirüs motoru tarafından algılanan, potansiyel olarak istenmeyen nesneleri sınıflara ayırır. Diğer antivirüs tedarikçileri de sınıflandırmada Kaspersky'nin kullandığı sınıflandırma sistemini temel almaktadır (Kaspersky, 2018). Kaspersky'nin sınıflandırma sistemi, algılanan her nesneye **Şekil** 2.4 ile gösterilen "sınıflandırma ağacında" açık bir tanım ve belirli bir konum atar. Bu şemada:

i. En az tehdit oluşturan davranış türleri, şemanın alt kısmında gösterilir.
ii. Daha yüksek bir tehdit oluşturan davranış türleri, şemanın üst kısmında görüntülenir.

Şekil 2.4 : Kötü amaçlı yazılım sınıflandırma ağacı



2.4.6 Akıllı Telefon Cihaz Güvenliği

InfoWorld'e göre, tüm akıllı telefonlarda üç temel güvenlik unsuru vardır. Mobil cihaz kullanıcısı olarak ilk yapılması gereken, bu katmanların farkında olup onları cihazlarınızda etkinleştirmektir: (KSmart, 2018)

i. **Cihaz Koruması:** Cihazın kaybolması veya çalınması durumunda verilerin uzaktan "silinmesine" izin vermek.

ii. **Veri Koruması:** Kurumsal verilerin aynı cihazda veya kişisel ağda çalışan kişisel uygulamalara aktarılmasını engellemek

iii. **Uygulama Yönetimi Güvenliği:** Uygulama içi bilgilerinizi ele geçirilmeye karşı korumak.

Güvenli bir uygulamanız yoksa işletim sisteminin ne kadar güvenli olduğunun önemi yoktur.

2.4.7 Akıllı Telefon Kullanım Güvenliği

Siber güvenlik uzmanları tarafından kullanıcılar için güvenli kullanım önerileri şunlardır: (KSafety, 2019)

i. Halka açık WIFI kullanılmamalıdır
ii. Sosyal medya uygulamaları ve şifreleri dikkatli kullanılmalıdır.
iii. Bağlantıların gerçek olduğu varsayılmamalıdır
iv. Sahte iletişimlere karşı dikkatli olunmalıdır
v. URL kontrol edilmelidir
vi. Şifreleme kullanın
vii. Güvenli bilgisayarlar ve internet bağlantıları kullanılmalıdır
viii. Asıl kredi kartı veya banka kartı kullanılmamalıdır
ix. İşletim sistemi güncellenmeli, Antivirüs ve Güvenlik duvarı uygulamaları kullanılmalıdır

2.5 KRİPTOGRAFİ

Kriptografi taraflar arasında mesajların güvenli bir şekilde iletilmesi ile ilgilenen bir bilimdir. Sadece mesajın şifrelenerek doğrulanması değil aynı zamanda değişiklşiklere karşı korunmasını da amaçlar.

2.5.1 Kriptografik Algoritmalar

Doğrudan finansal etkileri olduğunu için ödeme sistemlerinde , gerek çevrimdışı gerekse çevrimiçi işlemler için pek çok kriptografik sistem kullanılmaktadır.

Ödeme sistemlerinde kullanılan Kriptografilk algoritmalar bu bölümde özet olarak verilmiştir.

2.5.1.1 Gizli Anahtarlı Kriptografik Algoritmalar

Gizli anahtarlı kriptografik algoritmalaren eski kriptografik algoritmalardan biridir. Bu algoritmalarda sifreleme ve sifre çözme birbirinin tersi seklinde olup iki işlem için de aynı anahtar kullanılmaktadır. Simetrik anahtarlı kripto sistemler olup kısaca akış

şöyledir:

i. Öncelikle taraflar aralarında gizli anahtarı tespit ederler ve güvenli bir şekilde paylaşırlar.
ii. Gönderici taraf şifreleme algoritmasına girdi olarak açık metin ve gizli anahtarı kullanarakşifreli metni elde eder.
iii. Alıcı taraf ise sifre çözme algoritmasına girdi olarak şifreli metini ve gizli anahtarıkullanarak açık metni elde eder.

Gizli anahtarlı algoritmalarda anahtarın boyu ve anahtarın gizli tutulması çok önemlidir. Ödeme sistemlerinde en çok kullanılan algoritmalar DES ve AES tir.

2.5.1.2 Açık Anahtarlı Kriptografik Algoritmalar

Açık anahtarlı kriptografik algoritmalar modern kriptografik algoritmalardır. Bu algoritmalarda sifreleme ve sifre çözme için gizli ve sçık olmsk üzere farklı anahtarlar kullanılmaktadır. Asimetrik anahtarlı kripto sistemler olup kısaca akış şöyledir:

i. Öncelikle gönderici tarafların her birikendi özel ve açık anahtarlarını yaratır, özel anahtarı saklı tutarak açık anahtarı diğer tarafla paylaşır.
ii. Gönderici şifreleme algoritmasına girdi olarak açık metin ve alıcının açıkanahtarını kullanarakşifreli metni elde eder.
iii. Alıcı sifre çözme algoritmasına girdi olarak şifreli metini ve kendi gizli anahtarını kullanarak açık metni elde eder.
iv. B taraf şifreleme algoritmasına girdi olarak açık metin ve açıkanahtarı kullanarak şifreli metni elde eder.
v. Alıcı taraf ise sifre çözme algoritmasına girdi olarak şifreli metini ve gizli anahtarıkullanarak açık metni elde eder.

Açık anahtarlı algoritmalarda anahtarın boyu ve üretimi yöntemi çok önemlidir. Ödeme sistemlerinde en çok kullanılan algoritmalar RSA ve ECC tir.

2.5.1.3 Özet Algoritmaları

Özet fonksiyonları tek yönlü sıkıştırma ve kolay hesaplanabilme özelliklerine sahip , anahtarlı ve anahtarsız çalışabilen fonksiyonlardır. Ödeme sistemlerinde en çok

kullanılan özetleme fonksiyonları SHA türevleri ve MD5 tir.

2.5.2 Anahtar Güvenliği

Kriptografide anahtarların gizli tutulması ve açığa çıkarılmaması çok önemlidir.
Bu bölümde ödeme sistemlerinde kullanılan anahtar güvenliği sağlama metotları özet olarak verilmiştir.

2.5.2.1 Donanımsal Anahtar Güvenliği

Kriptografik anahtarların güvenli bir şekilde kullanılmasını sağlayan donanımlardır. Temel amacı, kriptografik anahtarları izinsiz kullanımın zor veya imkansız olduğu şekilde saklamaktır. Tüm mobil cihazlarda bulunmadığından, kullanılması durumunda cihaz bağımlılığı oluşturmaktadır. Bu donanımların başlıcaları şunlardır:

i. TPM, özellikle kripto hesaplama yapmak için oluşturulmuş bir donanım parçasıdır. İşletim sisteminin geri kalanından fiziksel olarak izole edilir ve çoğunlukla ana kart üzerinde ayrılmış bir entegredir.
ii. TEE, yonga kümesinde TPM gibi çalışan ancak çipin geri kalanından fiziksel olarak izole olmayan bir alandır.
iii. SE, bir akıllı kart veya SIM kart gibi, kurcalamaya karşı dayanıklı güvenli deposudur.

Mobil cihazlarda bulunabilen donanımların aksine, yine donanımsal bir anahtar güvenliği aygıtı olan HSM, sunucu tarfında şifreleme işlevlerini gerçekleştirir.

Ödeme Sistemlerinde en çok kullanılan HSM aygıtlarıdır. Sunucu uygulamalarında HSM tipine ve desteklediği komut setine göre özelleştirme yapılması gerekir. Ağ modunda, HSM istemcilerden gelen bağlantı isteklerini bekleyen bir TCP portunu dinleyen bir TCP / IP sunucusu gibi davranır.

Kriptografik bir işlem yapacağı zamanuygulamalar, TCP / IP kullanarak HSM'ye bağlanır ve ardından HSM'nin belirtimine göre komutu oluşturur ve ardından yürütme için HSM'ye gönderir. HSM gerekli görevi yerine getirecek ve sonucu uygulamaya geri verir. Tüm şifreleme, şifre çözme işlemleri HSM içinde güvenle gerçekleşir.

Fiziksel bağlantı gerektiren konsol modunda HSM, normalde ağ modunda mümkün olmayan anahtar bileşenlerin üretilmesi, anahtar bileşenlerden anahtarın üretilmesi gibi özel yönetsel işlevleri yerine getirir.

HSM temelde aşağıdaki işlevleri yerine getirmek için kullanılır:

i. Anahtar Saklama
ii. Anahtar Üretimi, Değişimi, Birleştirilmesi
iii. Kimlik ve Kriptogram Doğrulama
iv. Rastgele Sayı Üretimi
v. PIN Çevirimi, Saklanması
vi. Dijital İmza Üretimi ve Doğrulama

HSM ayrıca hassas verileri Simetrik veya Asimetrik algoritmalar şifreleme ve şifre çözme işlemlerini gerçekleştirmek için kullanılabilir.

i. RSA. EC
ii. DES, AES
iii. Elektronik Kod Kitabı (ECB)
iv. Şifreli Blok Zinciri (CBC)
v. CBC MAC içeren sayaç (CCM)
vi. HMAC

2.5.2.2 Yazılımsal Anahtar Güvenliği

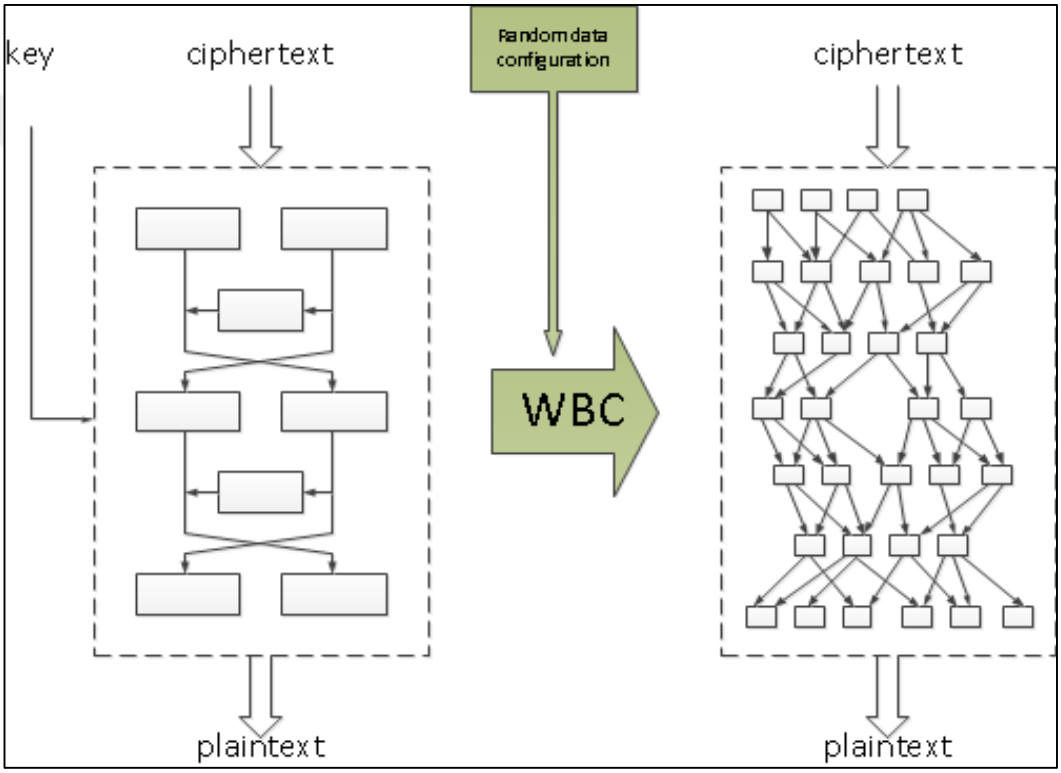
Kriptografik anahtarların güvenli bir şekilde kullanılmasını sağlayan yazılım yaklaşımına ve uygulamasına verilen isimdir. Temel amacı, özellikle simetrik kriptografik anahtarları koda çevirerek, boyutunu arttırarak karmaşık bir şekilde saklamak ve kullanmaktır. Akademik olarak ilk çalışma 2002 yılında Chow, Eisen, Johnson ve van Oorschot (SAC 2002) tarafından yapılmıştır.

Algoritmayı kod boyutunda oluşturduğundan, elle tersine mühendislik için çok uzun zaman gerekmektedir. Örneklem çeşitlendirmesinden dolayı ölçeklenebilir saldırı mümkün değildir.

Diğer yandan işlevi değil anahtarı gizlemeyi amaçlaşdığından, saldırganlar tüm işlevi kopyalayabilir ve harici olarak çalıştırabilirler. Kod kaldırma saldırılarına açıktır. Aynı zamanda olgunlaşmamış bir teknoloji olmasının yanısıra, daha şimdiden otomasyon, yan kanal, hata enjeksiyonu gibi çeşitli ataklar ortaya çıkmış bulunmaktadır (Baek, et al., n.d.).

Beyaz kutu şifreleme şeması **Şekil** 2.5 ile verilmiştir (WBC, 2002).

Şekil 2.5 : Beyaz kutu şifreleme şeması



Beyaz Kutu Şifrelemesi, bir saldırgan tarafından tamamen gözlemlenebilir ve değiştirilebilir bir mobil cihaz gibi bir yürütme ortamında kriptografik algoritmaların güvenli bir şekilde uygulanmasını açıklar. Bir algoritmanın dahili işleme verilerinin bir saldırgan tarafından kullanılamadığı kara kutu şifrelemesinden farklıdır. Beyaz kutu ortamı, kriptografik algoritmaların uygulanmasına ek kısıtlamalar getirmektedir. Geleneksel kriptografide kara kutu saldırısı, saldırganın algoritmayı bilerek ve giriş ve çıkışları izleyerek, ancak yürütmenin görünür olmasını izleyerek kriptografik anahtarı elde etmeye çalıştığı durumu açıklar.

Beyaz kutu kriptografisi, saldırganın her şeyi gözleyebildiği, hedef sistem uygulamasının tüm yönlerine erişebildiği ve kripto algoritması hakkında kara kutu bilgisine sahip olabileceği daha şiddetli tehdit modeline hitap eder. Beyaz kutu şifreleme, bir yazılım uygulamasında gizli anahtarların ifşa edilmesini önlemeyi amaçlar.

3. VERİ VE YÖNTEM

Tez konusu olan ödeme sistemlerinde mobil güvenlik, ödeme sistemlerinde kullanılan Android uygulamalarının güvenliğinin, uygulama seviyesinde ve cihaz üzerinde sağlanması için bir tasarım ve bu tasarımı gerçekleştirerek testini içerecektir.

Bu nedenle ödeme sisteminin bütün bileşenleri değil de sistemin güvenlik açısından en önemli parçasını oluşturan, müşteri tarafından kullanılacak olan mobil uygulama ve bu uygulamanın kullandığı sunucu arka uç servislerinin güvenliği ile ilgilenecektir.

Birinci yazılım Android cihazlarda çalışacak uygulamaların kullanacağı bir güvenlik kütüphanesidir. Bu kütüphane AAR uzantılı bir SDK'dır. Diğer yandan güvenlik sistem olarak bir bütün olduğundan, sunucu üzerindeki arka uç servislerinin yazılımı da yapılacaktır

3.1 ANDROID SALDIRILARI

Android güvenlik kütüphanesinin geliştirilmesi için öncelikle tehditlerin ve saldırıların tariflenmesi gerekmektedir. Bu açıdan baktığımızda saldırı tipleri aşağıdaki gibi özetlenebilinir;

i. Statik Analiz Uygulamaları
ii. Dinamik Analiz Uygulamaları
iii. Köklenme Uygulamaları
iv. Ayrıştırma / Sökme / ÇıkarmaUygulamaları
v. Emülasyon Uygulamaları
vi. Hata Ayıklama Uygulamaları
vii. Kod İzleme Uygulamaları
viii. API İzleme Uygulamaları
ix. Kancalama Uygulamaları

3.1.1 Köklendirme

Köklendirme (Rooting), en çok Android akıllı telefonları ve tabletleri olan aygıtlar üzerinde root erişimi veya ayrıcalıklı kontrol kazanma sürecini tanımlamak için

kullanılan bir terimdir. Köklendirme, Linux ortamlarını temel alan cihazlarda da yapılabilir. Kilit açma (Unlocking) ve hapsedilme (Jailbreaking) gibi terimlere benzese de, kavramsal olarak köklenme bu terimlerden oldukça farklıdır. Köklendirme, normal bir kullanıcının işletim sistemi ortamı için yönetici düzeyinde izinlere sahip olmasını sağlar. Android cihazlar söz konusu olduğunda, güvenlik mimarisini atlatmaya yardımcı olsa da potansiyel olarak ciddi sorunlara neden olabilir.

Köklenme, bir nevi işletim sistemininin kontrolünü açma işlemidir. Köklendirme işletim sisteminin güvenlik sınırlamasını ortadan kaldırır ve cihazın üzerinde tam kontrol imkanı sağlar. Bu sayede tüm kaynaklara sistem düzeyinde erişim sağlanır. Dosyalara, belleğe, çevre birimlerine ve arayüzlere erişim sağlanabilir. Tüm işletim sistemi koruması kök salması nedeniyle geçersiz kılınır. Köklenme, işletim sisteminin biraçığı kullanılarak elde edilebildiği gibi uygulama ile de yapılabilir. Unutulmaması gereken en tehlikeli saldırıların cihaz köklenerek başladığıdır.

Bir cihazı köklendirmek, cihazın kök dizininde Superuser'e haklar vermek için dosya sistemini değiştirir. Böylece Android'e tam erişim sağlanır, kök kullanıcı ile cihazda sınırsız izin elde edilir.

Köklenme işlemi çok temel saldırı tipi olduğundan,bir istismar paketinin alıcısından, uzman kara kutu suçlu saldırganına cok farklı uzmanlık seviyelerindesaldırgan özellikleri olabilir. Köklenme sonucunda cihaz üzerinde tam kontrolsağlanacağından ölçeklenebilir saldırılar düzenlenebilir.

3.1.2 APK / AAR Tersine Mühendislik

Yazılım sistemlerinde tersine mühendislik(Reverse Engineering) temelde, elde bulunan bir yazılımın; içeriğinin, yapısının, akışının veya teknolojilerin çeşitli analiz araçları ile keşfedilmesi, anlaşılmasıdır.

Tersine mühendislik, nesneyi çoğaltmak veya geliştirmek için nasıl çalıştığını görmek için bir nesneyi ayırır. Yazılım özelinde tersine mühendislik, bir yazılımın makine kodunun bir program dili ifadeleri kullanılarak, yazıldığı kaynak koduna geri döndürülmesini içerir. Tersine mühendislik saldırı amaçlı kullanılabildiği gibi, saldırılara

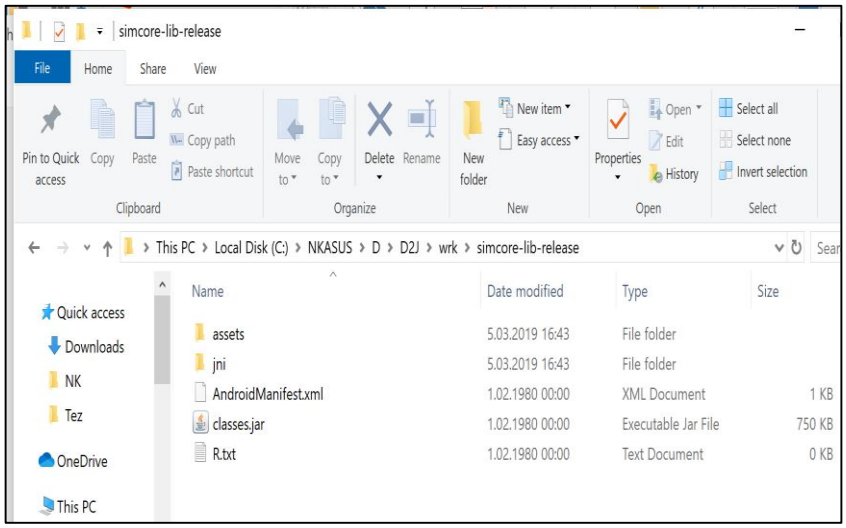
karşı güvenlik önlemi alabilmek vesaldırıları incelemek ve çalışma mantığını anlamak için de kullanılabilir.

Benzer şekilde kötü amaçlı Android uygulamalarınıda tersine mühendislik araçları ile incelemek ve elde edilen verilere ve bilgilere göre geliştirilen saldırılara karşı önlem almak mümkündür. Bu amaçla yapılan çalışmaların çoğunda uygulamaların veri setleri incelenerek, makina öğrenmesi ve yapay zeka kullanımı ile uygulamanın sunucularda skorlanması şeklindedir.

Bu çalışmada ise amaç bundan farklı olarak Android uygulamalarına eklenecek bir kütüphane ile yerel olarak tehditlerin teşhis edilmesi, saldırıların tespit edilmesi ve durdurulması için yöntemler geliştirilmesi, hassas veri ve anahtarların korunmasıdır.

AAR dosyası sıkıştırılmış formatta olup, uzantıları değiştirilerek sıkıştırma uygulamaları aracılığıyla içeriği çıkarılabilir . **Şekil** 3.1 te bu dosyalar gösterilmektedir. Bu yöntem ile çıkarılan kaynakların çoğu okunabilir formatta değildir.

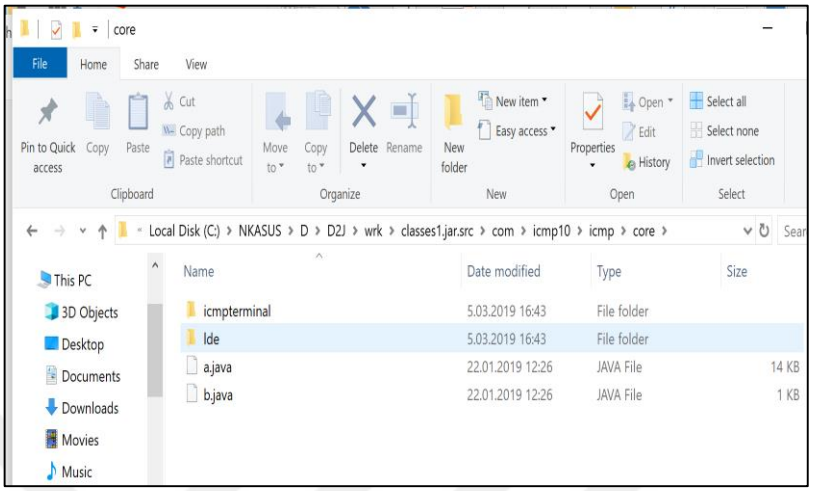
Şekil 3.1 : AAR Uzantısı değiştirilerek elde edilen dosyalar



Kütüphanenin kaynak kodlarının okunabilir hale gelmesi için dönüştürücü araçların kullanılması gerekir. Java baytkodlarını içeren .jar uzantılı dosya, APKTool (APKTool, 2018) dönüştürücü uygulaması kullanılarak Java kaynak kodları elde edilir.

Şekil 3.2 te bu dosyalar gösterilmektedir.

Şekil 3.2 : APKTool ile elde edilen dosyalar



Böylece orijinali;

```
    public GetAllTerminalParameterResult getAllParameters(String dtId) {
        return ldeBusinessService.getAllParameters(dtId);
    }
```

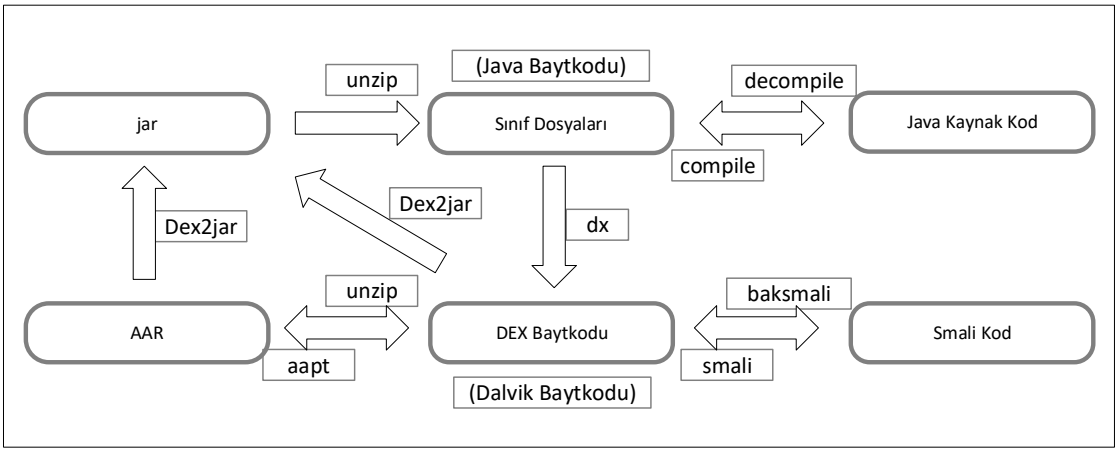
Olan kaynak koda;

```
  public com.icmp10.icmp.core.lde.b d()
  {
    return this.i.b();
  }
```

şeklinde ulaşılabilir.

APKTool kullanılarak elde edilen smali kodlar değiştirilerek yeniden AAR yaratılabilir. Bir APK veya AAR dosyasından kaynak kodları çıkarma, kaynak kodlarından derleyerek yeniden APK veya AAR haline işlemleri **Şekil 3.3** ile gösterilmiştir.

Şekil 3.3 : Uygulama derleme ve kaynak koda dönüştürme adımları



3.1.3 Hata Ayıklama

Uygulama içindeki hatalara "bug", hata ayıklama işlemine de "debug", bu işlemi yapan uygulamalara "debugger" denir. Android özelinde Hata ayıklama modu, fizksel veya sanal cihazın üzerindeki USB ile Hata Ayıklama seçeneğinin açılarak; cihazın ileri seviye sistem komutlarını çalıştırabilmesini sağlar.

Genel olarak bütün Android cihazlarda geliştirici moduna, Seçenekler menüsünden, Cihaz Hakkında veya Yazılım Bilgisi menüsüne gidilerek, "Build Numarası" seçeneğine belirli bir sayıda dokunularak geçilir. AndroidManifest.xml dosyasında android:debuggable="true" ise uygulama veya kütüphane için Hata Ayıklama Modunda derlenir, aksi taktirde yayın modunda olacağından hata ayıklama yapılamaz.

Gerek uygulamanın hata verdiği durumlarda gerekse uygulamadan beklenmedik değerler alındığında hatta programlamaya yeni başlayanlar kodların nasıl çalıştığını daha iyi anlayabilmek adına hata ayıklama işlemleri geliştirici için büyük kolaylıklar getirmektedir. Fakat bu işlem sırasında sistem dış saldırılara maruz kalabilmektedir, hele yayın ortamındaki kullanıcıların cihazları için bu durum çok daha tehlikelidir.

Güvenlik amacıyla hazırlanacak kütüphanenin, Android Hata Ayıklama Köprüsünü ve Uygulamanın Hata Ayıklama modunda olup olmadığını tespit etmesi ve ona göre davranması gerekmektedir.

3.1.4 Emülasyon

Bir sistemin başka bir sistem üzerinde çalıştırılmasına emülasyon (Emulation) denir ve bu tür yazılımlara da emülatör denir. Böylece Windows üzerinde Android Cihaz çalıştırabilirsiniz. Android Studio aracılığıyla sanal android cihazı üreterek kullanmak çok kolaylaşmıştır.

Güvenlik amacıyla hazırlanacak kütüphanenin, cihazın Emülatör modunda olup olmadığını tespit etmesi ve ona göre davranması gerekmektedir.

3.1.5 Kancalama

Kancalama (Hooking), mevcut çalışan işleme kötü niyetli rutinler enjekte etme işlemidir. Kötü amaçlıyazılım araçları, hedef yazılımın kaynak kodunu elde etmeden çalışma anında prosesin akışına erişip, akışı kontrol edebilir, değişkenleri değiştirebilirler. Fakat bunun için kancalanacak fonksiyonların bilinmesi veya bulunması gerekir.

Kancalama yapılabilecekler aşağıda verilmiştir:

i. Çağrılar incelenip ve kaydedilebilir,
ii. Çağrılar engellenebilir,
iii. Parametreler değiştirilebilir,
iv. İletişimi kötü niyetli bir sunucuya yönlendirilebilir,
v. Güvenlik ihlali belirten bir değerler doğru gibi ayarlanabilir,
vi. Şifre kontrolünden sonra her zaman doğru dönülebilir,
vii. Bir işlevi değiştirilebilir,
viii. Şüpheli dosyalar uygulamadan gizlenebilir.

Başlıca kancalama çeşitleri şunlardır:

i. Metod kancalama
ii. Giriş noktası kancalama
iii. API kancalama
iv. Sistem çağrısı kancalama

Güvenlik amacıyla hazırlanacak kütüphanenin, cihaz üzerinde kancalama uygulaması olup olmadığını ve prosess haritasında tehlikeli bir işlem olup olmadığını tespit etmesi ve ona göre davranması gerekmektedir.

Kritik noktalarda genel geçer öngörülebilinecek sistem çağrıları yerine yerel karşılıkları geliştirilip, kullanılmalıdır. Aynı zamanda özellikle JNI katmanında kancalama karşıtı tedbirlere sahip olmalıdır.

3.1.6 Sömürü

"Exploit" kelimesinin sözlük anlamı "sömürmek"tir. İşletim sistemindeki bir açığın kötü niyetle kullanılması işlemidir. Bir sistem üzerinde keşfedilmiş olan açığın kullanılabilmesi için geliştirilen yazılımlar veya scriptlerdir. İstismarın çalışma şekli oldukça basittir: kötü amaçlı bir uygulama, Android cihazına indirilir ve kurulur; kullanıcıların girmesine gerek olmadan gerekli izinler verilir. Buradan kötü amaçlı yazılımlar sayesinde, kullanıcının farkında olmadan tıklama kiti, tuş vuruşlarını kaydetme, kimlik bilgilerini ve anahtarları açığa çıkarmak, elde etmekmümkün olabilir.

3.1.7 Kurcalama / Yeniden Paketleme

Kurculama (Tampering) uygulama ile kod seviyesinde oynanabilmesidir. Yeniden paketleme (Repackaging) , Mobil Uygulamanın yetkisiz bir kişi tarafından değiştirilmesi vefarklı amaçlarla özelleştirilmesidir. Saldırganın; APK dosyasını açarak, smali kodunu değiştirerek APK dosyasını yeniden derleyip, imzalayıp bir mobil cihazda çalışır hale getirmesidir.

Kurcalama ve yeniden paketleme karşıtı alınabilecek bazı önlemler aşağıda verilmiştir:

i. Çalışma zamanı bütünlüğü kontrolleri
ii. Kendi kendini denetleme
iii. İmzalama anahtarlarının kontrolü
iv. APK paketini ve ikili kod karma değerlerinin kontrol edilmesi
v. Cihaz bağlama

3.1.8 Cihaz Bağlama

Saldırganlar uygulamayı, veriyi ve kodunu boşaltmak ve başka bir ortamda çalıştırmak isteyebilir. Bu sebeple mobil uygulamayı kurulduğu cihaza özel olacak şekilde bağlamak gerekir. Burada amaç her bir mobil cihaz için cihazın içindeki verilerden benzersiz bir parmak izi (Mobile Device Fingerprint) üretmektir. Bu parmak izi anahtar olarak verilerin şifrelenmesinde kullanılabilir.

Kullanılacak cihaza özel veriler şunlar olabilir:

i. Ekran özellikleri
ii. Cihaz yapı numarası
iii. İşletim sistemi versiyonu
iv. Kurulu eklentiler
v. Cihaz numarası
vi. IMEnumarası
vii. MACnumarası

Buna karşı köklenme ve kancalama yapılarak, cihaz bağlamaya karşı saldırılar yapılabilir:

i. Cihazın içindeki benzersiz bir parmak izi üretim mekanizması bulunabilir
ii. Üretildiği yer dışında kullanılan cihazın parmak izi değerine müdahale edilebilir
iii. Standart Android emülatörlerinin yaptığı gibi, bilinen tüm cihaz özellikleri işletim sistemi düzeyinde kopyalanarak sahtekarlıkta kullanılabilir

3.1.9 Saldırı Araçları

Başlıca Android saldırı araçlarıaşağıda verilmiştir;

i. Rooting: Towelroot
ii. Inspection Tool:Androguard
iii. Disassembler: IDA, JEB
iv. Debugger: GDB, JDWP
v. Decompiler: JEB
vi. Instrumenting: ADBI, DDI, Frida,Substrate, Xposed

3.1.10 Gizleme / Şaşırtma

Obfuscation, karşılığı gizlemek, şaşırtmak olan bir işlemdir. Bu işlem, yazılımda, kodların anlaşılabilirliğinin daha az olması için yapılmaktadır. Temel amacı ise güvenlik ve gizliliktir. Bu işlem, kodun işleyişine bir etkide bulunmaz, sadece kodun anlaşılabilirlik derecesini zorlaştırır. Bir nevi kodumuzukilitleme işlemidir.

Obfuscator dediğimiz araçlar ise, yazılan kodların anlaşılabilirliği daha zor hale getiren, yani gizleme işlemini yapan uygulamalardır. Obfuscation işleminin tersini yapan

uygulamalar da mevcuttur.

Şaşırtma, kodu okumayı zorlaştırmak için kullanılan tipik bir kod koruma tekniğidir. Şaşırtma, sınıfların ve yöntemlerin adını okunamayan veya anlamsız karakterlere dönüştürerek başkalarının koduanlamalarını zorlaştırır.

Nesneye yönelik programlama her yerde yararlıdır çünkü kodu okumak, uyarlamak veya ölçmek için çeşitli avantajlar sunar. Bununla birlikte, modüllerin bu şekilde programlanabilmesi, birçok izlemeyi çalıştırılabilir hale getirir ve tersine mühendisler bu izleri orijinal kaynak kodunu yeniden oluşturmak için mümkün olduğu kadar iyi kullanırlar. Bu nedenle, programcılar analizin zorlaşması için bir programın içindekileri anlamakta güçlük çeken birkaç teknik geliştirmelidir. Bu teknik, bir kodu analiz ve kurcalamaya karşı daha dirençli hale getiren, ancak işlevselliğini koruyan kod için bir veya daha fazla dönüşüm uygular.

3.1.10.1 Kod ve Değer Gizleme

Saldırganın kod analizini yapmasını zorlaştırmayı hedefler. Kod içindeki fonksiyon, metod ve değişkenlerin isimlerini anlamsız hale getirerek uygulanır

i. İsim gizliliği: İsimleri anlamsız hale getirmek
   sifrele (bayt [] acik); iken m1 (bayt [] p1) yapmak.
   Bilgi kaybı yaşanacak olsa da, elle iyileştirme mümkündür.
ii. Dize şaşırtma: Dizeleri anlaşılmaz hale getirmek
   String algo = “AES / CBC / PKCS5Padding”; iken
   String s1 = m2 (“802e8) Qjrq_zgF) 2LUx”) yapmak.
   Şifreleme statik ise otomatik şifre çözme mümkündür.
iii. Aritmetik dönüşümler kullanarak değerlerin saklanması

3.1.10.2 Akış Gizleme

Kaynak kod yapılarıyla uyumlu olmayan ikili dosyaya atlama ifadeleri eklenerek ve decompiler analiz motorlarından yararlanarak, uygulamada ayrışmayı önlemek ve mantığı gizlemektir. Binary analiz ve kısmi yeniden yapılandırma ile anlaşılması zor da olsa mümkündür. Kullanılbilecek başlıca yöntemler şu şekildedir :

i. Şifrelenmiş sınıfların dinamik olarak sınıf yaratılması.
ii. Metod ve fonksiyonların tekrarlanması
iii. Şifreleme için metod ve fonksiyonların yeniden yazılması
iv. Native kod kullanılması
v. Sahte kod ve satır içi kod kullanılması

3.1.11 Gizleme Araçları

Kötü amaçlı yazılım araçları gibi iyi amaçlı yazılı araçları da vardır:

i. ProGuard
ii. DexGuard
iii. DexProtector

Bu araçlar aşağıdaki koruma mekanizmaları ile mobil uygulamaların güvenliklerini arttırmayı amaçlar :

i. Dize Şifreleme
ii. Sınıf Şifreleme
iii. Varlık Şifreleme
iv. Erişim Gizleme
v. Yerel Kod Şifreleme
vi. Yerel Kod Gizleme

3.2 ÖDEME SİSTEMLERİNDE MOBİL

Mobil cihazlar ortaya çıktıktan çok kısa bir süre sonra ödeme sistemlerinde kullanılmaya başlanmışlardır. İlk önce SMS gönderimi gibi basit olan bu kullanımlar cihazların akıllanması ile birlikte uygulama seviyesine ulaşmış ve sektör içim vazgeçilmez hale gelmiştir.

3.2.1 Mobil Cüzdan

Mobil cüzdan, akıllı cihazlar ile banka hesapları, kredi kartı veya sanal kart tanımlanarak para transferi, fatura ödeme ve alışveriş işlemlerini yapabilmeyi sağlayan mobil cihaz

uygulamalarıdır. Fiziki olarak taşıdığımız cüzdanın sanallaştırılmış ve bir mobil uygulamaya çevrilmiş hali olarak ta düşünülebilir.

3.2.2 Elektronik Para

Elektronik para, fiziksel kağıt para değerinin sayısal olarak bir ortamda tutulmasıdır. Dijital para elektronik olarak, kurumun merkezi veritabanında, bulut sistemlerde, akıllı kart veya mobil cihaz üzerinde tutulabilir.

Elektronik para değeri bulunduğu ortama göre farklı şekillerde kullanılabilir. İçerisine para değerinin sayısal olarak yüklenebildiği ön ödemeli, çevrimdışı çalışan merkezi sunucuya bağlanmadan işlem yapılabilen sistemler olduğu gibi, içerisine para yüklenmeyen, işlemin çevrimiçi olarak merkezi sunucu tarafından onaylandığı sistemler de vardır.

3.2.3 Akıllı Kart

Akıllı kart, elektrik enerjisi ile çalışan içerisinde mantıksal-lojik hesaplamalar yapabilen mikro işlemci, sınırlı bellek alanı gibi birimleri bulunan silikon tabanlı entegre elektronik devre bütünüdür. Elektrik ve iletişimin kart okuyucunun kart kontaklarına teması ile sağlandığı temaslı kartlar (ISO 7816), kart okuyucunun manyetik alanına girmesi ile sağlandığı temassız kartlar (ISO 14443) vardır.

3.2.4 NFC

NFC terimi İngilizce “Near Field Communication” kelimelerinin baş harflerinin kısaltılmış halidir. Yakın alan iletişimi olarak Türkçeye çevirebileceğimiz yeni nesil bir RF radyo frekans dalgaları üzerinden kablosuz iletişim protokolü setidir. Teknolojinin geliştirilmesinde temassız iletişim sağlayan akıllı kartlar sayesinde olmuştur.

Veri iletiminin yavaş ve kısa mesafede olması dezavantaj gibi görülse de ödeme sistemleri için yeterli kapasitede ve güvenlik açısından avantajlı olmaktadır. Desteklenen modlar:

i. NFC Emülasyon Modu,mobil cihazın telefonun temassız akıllı kart gibi davranmasıdır.
ii. NFC Terminal Modu mobil cihazın terminal gibi çalışmasıdır.
iii. NFC Peer to Peer Mod, iki cihazın karşılıklı veri iletimi yapmasıdır.

3.2.5 HCE

HCE terimi ingilizce “Host-Based Card Emulation”kelimelerinin kısaltılmasından oluşmaktadır. Host based kelimeleri mobil cihazın işletim sistemini ifade etmektedir. HCE işlevselliği için akıllı cihazda 4.4.2 üzeri işletim sistemi, NFC çip yongası ve temassız iletişimi sağlayan anteni bulunmalıdır.

3.2.6 Mobil Ödeme

Mobil cüzdan içerisindeki elektronik paranın kullanılması ile gerçekleşen ödeme işlemlerine mobil ödeme denir. Ödemeler Karekod, SMS, OCR tabanlı olabildiği gibi, NFC / HCE tabanlı da gerçekleştiriebilir. Fiziksel ödemeden daha pratik, kullanım sınırlaması bulunmayan ve güvenli bir sistem olması nedeniyle hem bireysel hem de kurumsal olarak günümüzde kullanılan bir ödeme şeklidir.

3.2.7 Çok Faktörlü Doğrulama

Kimlik doğrulama modelleri gelişimi ve içerdiği faktörler itibariyle sırasıyla şöyledir:

i. Tek faktörlü doğrulama: Bildiğin bir şey; kullanıcı şifresi gibi
ii. İki faktörlü doğrulama: Sahip olduğun bir şey; OTP cihazı, Cep telefonu gibi
iii. Üç faktörlü doğrulama: Olduğun bir şey; Biyometrik doğrulama iris, parmak izi gibi

Mobil dünyada bankacılık uygulamaları genelde cep telefonuna gönderilen tek kullanımlık şifreler ileiki faktörlü doğrulama kullanmaktadır. Çok güvenli olduğu düşünülen bu yönteme karşı bile saldırılar başarılı olmuştur.

Örneğin Zitmo program ailesi bu amaçla tasarlanmış ve Zbot (ZeuS) ile birlikte çalışarakiki faaktörlü kimlik doğrulama sistemlerini aşabilmiştir:

i. Zbot kullanıcı adını ve parolayı çalarak virüslü bilgisayardan bankacılık sistemine girer.
ii. Para transferi esnasında Zitmo görevi devralarak işlem onay kodunu siber suçlulara iletir.

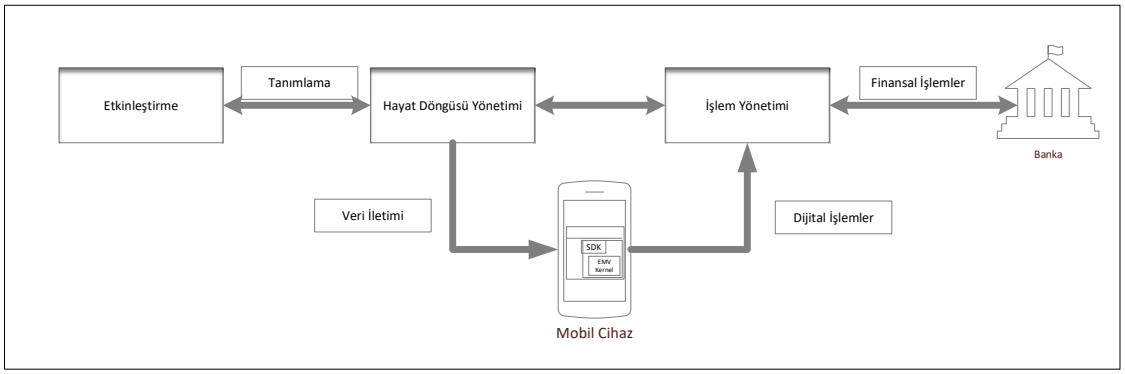
4. BULGULAR

Ödeme sistemlerinde güvenli Android uygulaması için önerilen sistem, Mobil Ödeme Uygulamasının geliştirme ve kişiselleştirme süreçlerinde kullanılacak bir yazılım geliştirme kitidir. Güvenliğe bütünsel olarak yaklaşmak gerektiğinden tasarlanan sistem, sunucu arka uç servisleri ve Android SDK olmak üzere iki bileşenden oluşmalıdır.

Sunucu servisleri, Etkinleştirme, Hayat Döngüsü Yönetimi ve İşlem Yönetimi hizmetlerini içeren bir platformu olarak düşünülmelidir.

SDK, Mobil Ödeme Uygulaması'nın geliştirilmesi sırasında kütüphane olarak kullanılır. Gereksinim duyulan yüksek güvenlikli şifreleme algoritmaları gerçekleştirmek ve hassas verileri gizlemek / korumak için kullanılır. SDK, içinde yüksek güvenlikli şifreleme algoritmaları uygulanan yerel kütüphaneyi (JNI) barındırmalıır. Bu algoritmalar aynı zamanda hazır bulunan Android kütüphaneleri kullanılmadan Java katmanında da geliştirilmelidir. SDK, yetkisiz değişiklik ve düzenlemelere karşı önlemler alınarak geliştirilmelidir. SDK ve sunucu arka uç servisleri, güvenli bir taşıma kanalı üzerinden iletişim kurmalıdır. **Şekil 4.1** sistem bileşenlerini göstermektedir.

Şekil 4.1 : Önerilen sistem bileşenleri



4.1 NOTASYONLAR

M, mesaj veya açık metin anlamına gelir. C şifrelenmiş metin anlamına gelir. k, şifreleme veya şifre çözme işlemi için kullanılan anahtarı ifade eder.

E, C'yi üretmek için k kullanarak M'yi şifrelemek anlamına gelir.
$E_k(M) = C$

D, M'yi üretmek için k kullanarak C'nin şifresini çözmek anlamına gelir.
$D_k(C) = M$

Bir iletiyi şifrelemenin ve daha sonra şifresini çözmenin sonucu orijinal açık metne ulaşmak olduğu için, aşağıdaki formül geçerli olmalıdır:
$D_k(E_k(M)) = M$

4.2 KULLANILAN ANAHTARLAR

Sistemde kullanılacak anahtarlar ve algoritmalar aşağıda verilmiştir

Mobil Anahtarlar

i. Mobile_CFD
ii. Mobile_MAC

Mobil Oturum Anahtarları

i. MobileSK_CFD
ii. MobileSK_MAC

CLOUD_EC_INIT Anahtarı

i. CLOUD_EC_PUB
ii. CLOUD_EC_PRV

SDK_EC_KEY_PAIR Anahtarı

i. SDK_EC_PUB
ii. SDK_EC_PRV
iii. Agreement Key

WBC_KEY

LDE_KEY

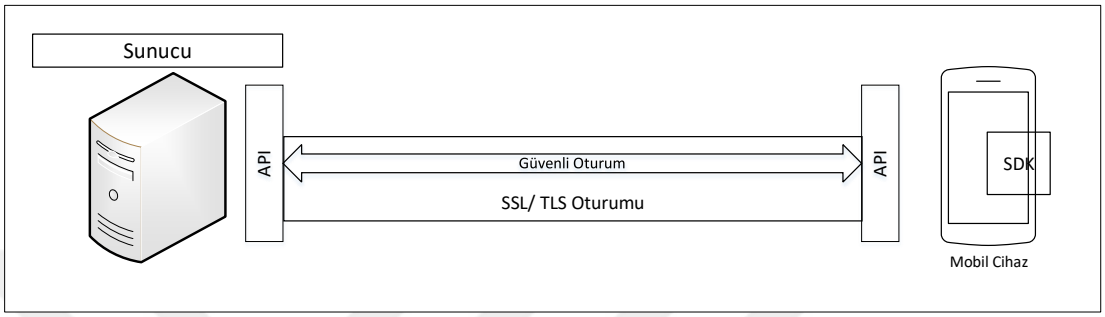
4.3 SDK VE SUNUCU GÜVENLİK TASARIMI

4.3.1 SSL / TLS

SSL / TLS iletişimi, SDK ve Uzaktan Yönetim'in başlatılması sırasında kullanılmalıdır.

Şekil 4.2, SDK ve Sunucu arasındaki iletişimi göstermektedir.

Şekil 4.2 : SDK ve sunucu iletişimi



Sertifika Sabitleme kavramı, MITM saldırılarına karşı olan bu SSL / TLS bağlantıları için olmazsa olmaz bir kuraldır.

SDK, çeşitli arabirimler tarafından kullanılan herhangi bir SSL / TLS bağlantısı için Sertifika Sabitleme özelliğini kullanmalıdır. SDK, aşağıdaki onaylamaları gerçekleştirmek için gereken güvenilir bilgilerin bir listesini tutmalıdır:

i. Sunucu Sertifikasında tanımlanan Ortak Adı, SSL / TCP bağlantısını kurmak için kullanılan Sunucu adına göre doğrulanması.
ii. Sunucu Sertifikasını referans Sertifikasına göre kontrol edilmesi
iii. Sunucu Sertifikasının Sertifika Zincirinde tanımlanan tüm sertifikaları güvenilir bir referans değerleri listesine göre kontrol edilmesi

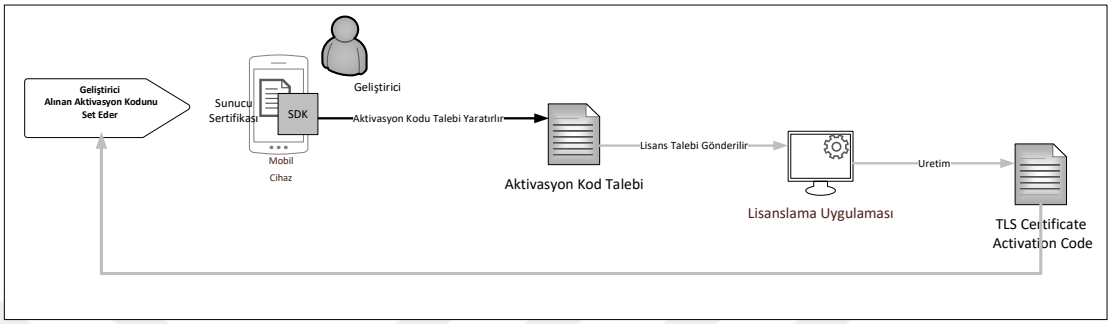
4.3.2 TLS Sertifika Aktivasyon Kodu

SDK'nın farklı sertifikalarla kullanılmasını önlemek için aşağıdaki adımlar atılmalıdır:

i. MPA geliştiricisi sertifikayı SSL / TLS için SDK yöntemine göre ayarlar.
ii. MPA geliştiricisi, SDK yöntemini çağırarak aktivasyon kodu isteği oluşturur ve lisanlamak için gönderir.
iii. Daha sonra lisanslama aracını kullanarak bu sertifika için benzersiz bir aktivasyon kodu oluşturulur.
iv. MPA geliştiricisi, SDK'yı aldığı benzersiz aktivasyon kodunu kullanarak açar.

Bir saldırgan SDK'yı orijinal sertifika değil farklı bir sertifika ile kullanmaya çalıştığında, SDK orijinal sertifika ile uyuşmayan aktivasyon kodunu reddeder. TLS Sertifika Aktivasyon Kodu akışının Üretimi ve Paylaşımı **Şekil 4.3** ile gösterilmiştir.

Şekil 4.3 : Sertifika aktivasyon akışı



4.3.3 Mobil Anahtarlar

Mobile_CFD şifreleme için kullanılan anahtardır. Mobile_MAC veri bütünlüğü için kullanılak anahtardır, Her ikisi sunucu tarafından SDK'nın başlatılması sırasında, rassal olarak üretilerek, varlık alışverişinin bir parçası olarak SDK'ya aktarılan 256 bit AES anahtarlardır. MobileSK_CFD ve MobileSK_MAC olan Mobil Oturum Anahtarları, oturum tanımlayıcısı kullanılarak ana mobil anahtarlardan elde edilir. Mobil oturum anahtarlarının oluşturulması sırasında, HMAC256 algoritması kullanılır,

i. Session_ID ve Mobile_CFD ile MobileSK_CFD üretilir.
ii. Session_ID ve Mobile_MAC ile MobileSK_MAC üretilir.

Mobil oturum anahtarları, SSL / TLS katmanı üzerinden iletilen uygulama verilerine uygulanan ek bir şifreleme katmanı ve veri bütünlüğü için kullanılır. Mobil oturum anahtarları, SDK'nın uzaktan yönetimi sırasında iletişimi güvence altına almak için kullanılır:

i. MobileSK_CFD, verilerin şifrelenmesi için kullanmak (Gizlilik).
ii. MobileSK_MAC, verilerde değişim kontolü için kullanılır (Bütünlük).

4.3.4 Uzaktan Bildirim Sistemi

Sunucu ve SDK arasındaki iletişimde, Uzaktan Yönetim işlemlerini desteklemek için iki iletişim kanalı kullanılır:

i. Bir RNS Protokolü kullanarak tetikleme işlemi

ii. Mesajlaşma Protokolü kullanarak veri alışverişi

Uzaktan Bildirim Hizmeti (RNS), sunucu arafından SDK'ya gönderilen bir mesajın iletimini (bir itme mekanizması kullanarak) desteklemek için kullanılır. Sistem taşıma katmanları ve servislerinin (RNS'nin kendisi gibi) kullandığı temel güvenlik hizmetlerine ek olarak uygulama düzeyinde veri alışverişini güvence altına alır.

SDK bir uzaktan bildirim mesajı tarafından tetiklendiğinde, SDK tarafından sunucuya bir veri bağlantısı kurulur:

i. Karşılıklı bir kimlik doğrulama protokolü her iki taraf tarafından tanımlanır ve yürütülür.
ii. SDK, sunucu'yu taşıma katmanını kullanarak doğrular (SSL / TLS).
iii. Sunucu, uygulama düzeyinde üretilen ve verilen bir Kimlik Doğrulama Kodu kullanarak SDK'yi doğrular.

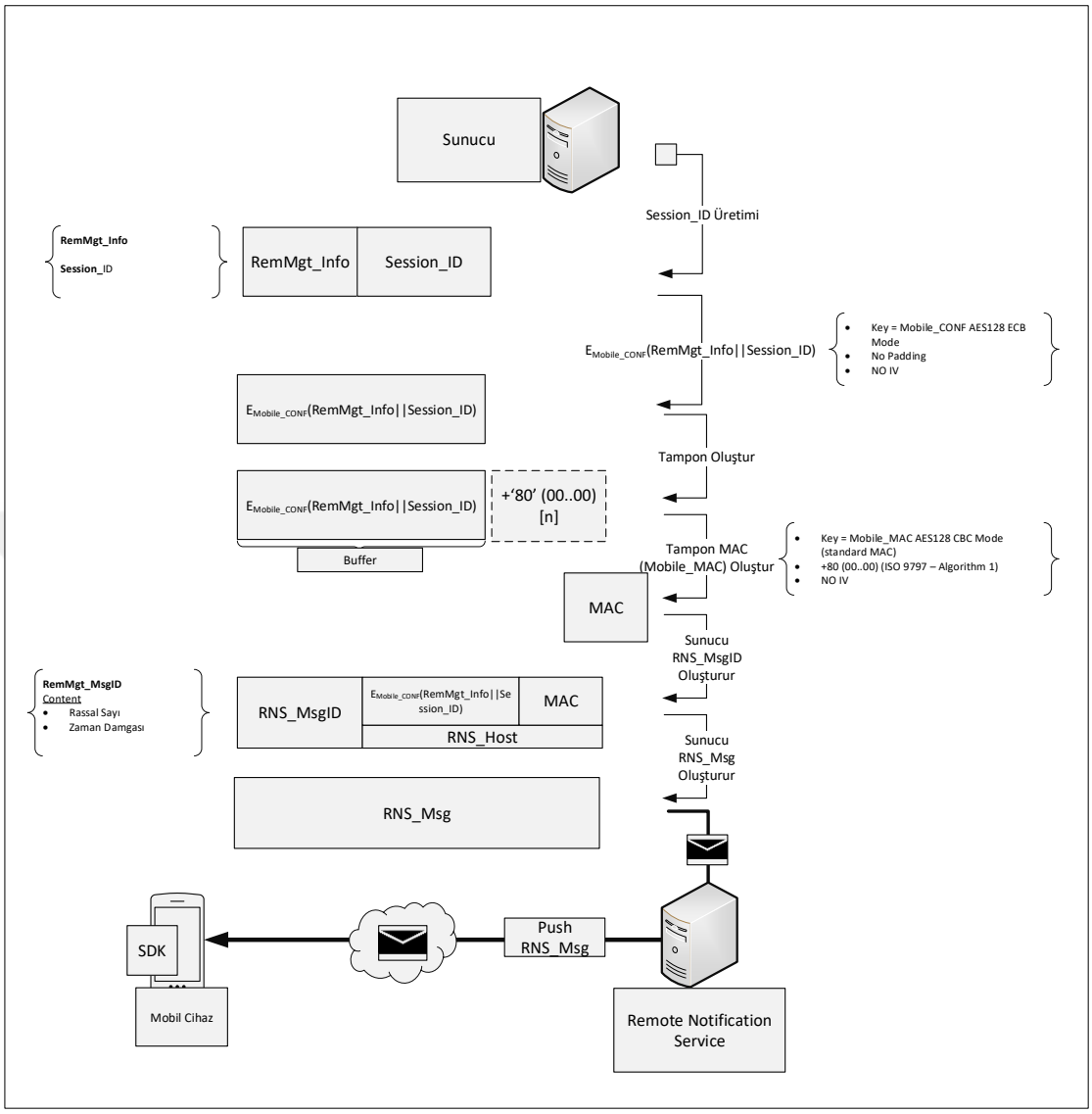
Mesajlaşma Protokolü, sistem tarafından, sunucu ve SDK arasındaki iletişim kanalı için karşılıklı kimlik doğrulama işlemi Uzaktan Bildirim Hizmeti (RNS) kullanılarak başlatılan bir oturum bağlamında başarıyla tamamlandıktan sonra kullanılan protokolü sunar.

RNS protokolü aşağıdaki gibi bölünmüştür:

i. RNS Mesajı - Sunucu, Sunucu tarafından bir mesajı hazırlamak ve SDK'ya iletmek için kullanılan protokolü belirler.
ii. RNS Mesajı - SDK, Uzaktan Bildirim Servisi'ni (RNS) kullanarak Sunucu tarafından basılan bir mesajı almak ve işlemek için SDK tarafından kullanılan protokolü belirler.

Sunucu'nun Uzaktan Bildirim Servisi (RNS) kullanarak bir mesajı SDK'ya hazırlamak ve iletmek için kullandığı protokolü **Şekil 4.4** ile gösterilmektedir.

Şekil 4.4 : Sunucu RNS protokolü



Mobil Anahtarlar, SDK'nın ilk kurulum sırasında Sunucu ve SDK arasında paylaşılır. RemMgt_Info ve Session_ID, Mobile_CFD kullanılarak E_{Mobile_CFD} (RemMgt_Info || Session_ID) oluşturmak için şifrelenir.

Bir Mesaj Kimlik Doğrulama Kodu (MAC) , bir Tampon üzerinden Mobil Anahtar (bütünlük kodu yaratma için kullanılır) kullanılarak hesaplanır.

Sunucu bir RNS Mesaj Tanımlayıcısı (RNS_MsgID) oluşturur ve aşağıdakileri içeren bir mesaj RNS_Msg iletmek için Uzaktan Bildirim Hizmetini (RNS) kullanır:

i. RNS_MsgID

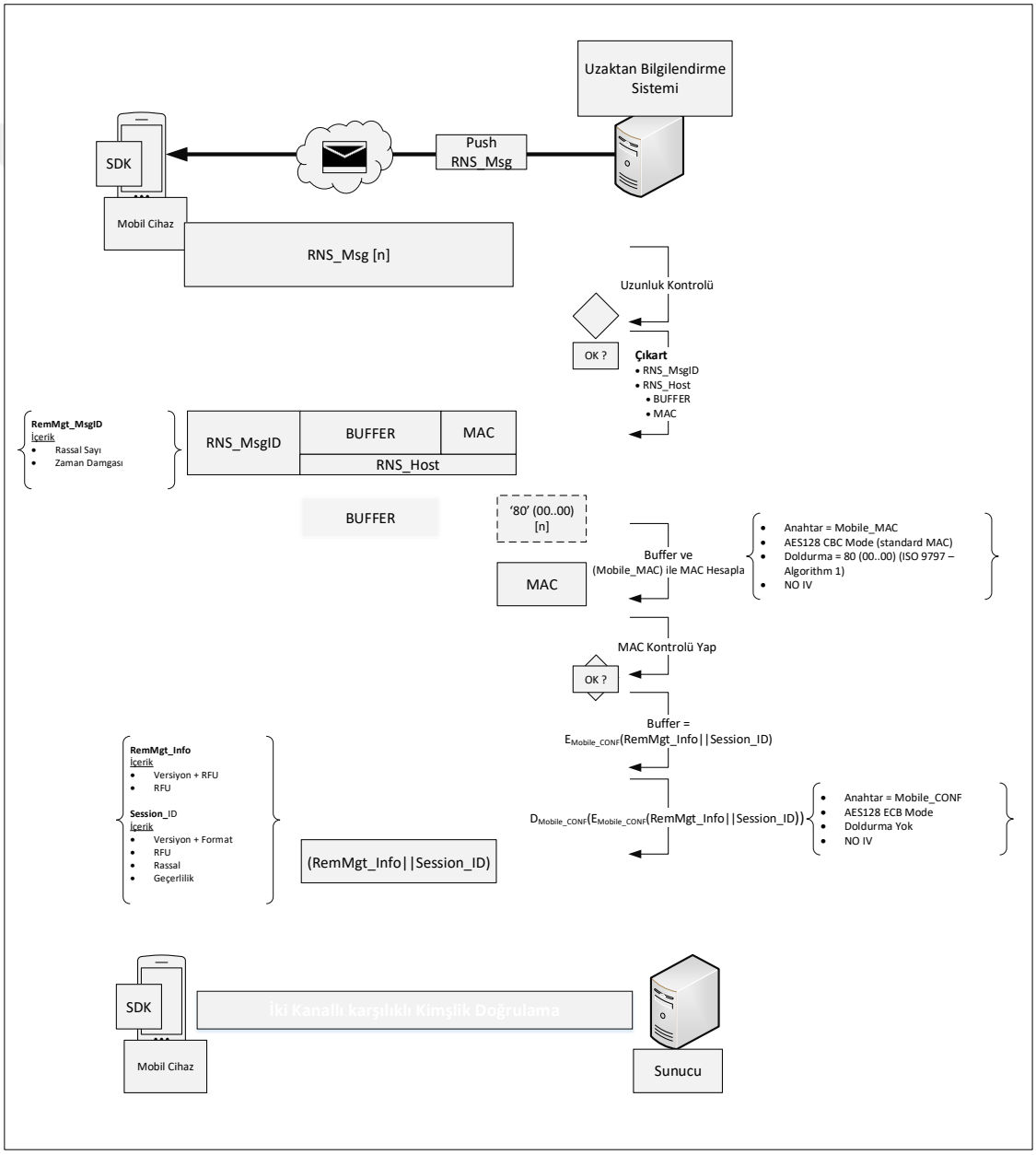
ii. RNS_Host

E_{Mobile_CFD}(RemMgt_Info||Session_ID)

MAC

SDK'nın SSL / TLS kullanarak ikinci bir kanal üzerinden bir Sunucu'ya veri hazırlamak ve bir veri göndermek için RNS_Msg aldıktan sonra SDK tarafından kullanılan protokol **Şekil 4.5** ile gösterilmektedir.

Şekil 4.5 : SDK RNS protokolü



SDK, RNS_Msg uzunluğunu kontrol ettikten sonra aşağıdakileri elde eder:

i. RNS Mesaj Tanımlayıcısı (RNS_MsgID)
ii. Arabellek
iii. MAC

Arabellek üzerinden MAC hesaplar ve değeri RNS_Msg içinde verilen MAC ile karşılaştırır

SDK E_{Mobile_CFD}(RemMgt_Info || Session_ID) şifresini çözer ve Sunucu ile Karşılıklı Kimlik Doğrulama işlemini desteklemek için Session_ID'yi kullanır.

Başarılı kimlik doğrulamasının ardından, Mobil Oturum Anahtarları (MobileSK_CFD ve MobileSK_MAC), çeşitlendirici olarak Session_ID kullanılarak Mobil Anahtarlardan elde edilir.

4.3.5 SDK ve Sunucu Karşılıklı Kimlik Doğrulama

SDK ile Sunucu arasında karşılıklı kimlik doğrulama aşağıdaki gibi yapılır:

i. SDK, Sunucu'ya bağlanır.
ii. SDK, Sunucu'yu Sertifika Sabitleme ile iletişim katmanı düzeyinde (SSL / TLS) doğrular.
iii. SDK bir Kimlik Doğrulama Kodu (MPA_AUTH) oluşturur ve bu Kimlik Doğrulama Kodunu SSL / TLS bağlantısı üzerinden Sunucu'ya sağlar.
iv. Sunucu, SDK'yi doğrulamak için bu Doğrulama Kodunu kontrol eder.
v. Sunucu kendi kimlik doğrulamasını hesaplar, karşılıklı kimlik doğrulama işlemini tamamlamak için Kimlik Doğrulama Kodunu karşılaştırır ve doğrular

MPA_AUTH, Sunucu'ya bağlanırken SDK tarafından üretilen bir Kimlik Doğrulama Kodudur. Bu kod, 32 baytlık bir değer oluşturmak için aşağıdaki kimlik doğrulama faktörleri üzerinden hesaplanan bir özettir (SHA 256):

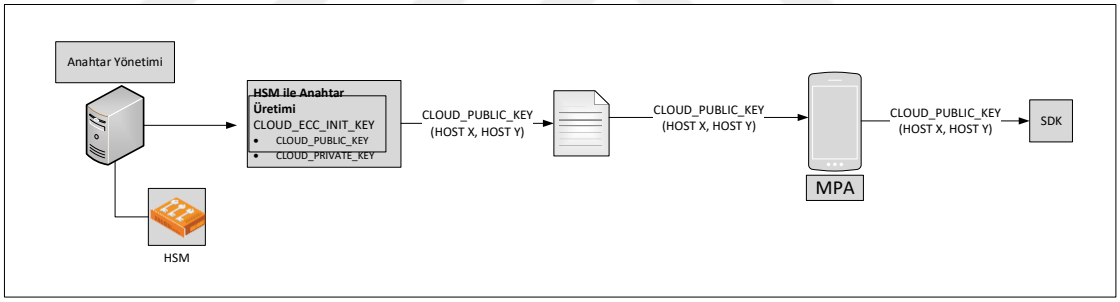
i. MPA_ID kayıt sırasında Sunucu tarafından oluşturulur ve SDK başlatma işlemi sırasında alır.
ii. Session_ID sunucu tarafından üretilir; SDK Uzaktan Yönetim işlemiyle alır. Her oturum için, Sunucu tarafından yeni bir Session_ID oluşturulur.
iii. Mobil Parmak İzi başlatma sırasında SDK'dan Sunucu'ya gönderilen Mobil Cihaz Bilgileri.

Özünde, SDK'nın doğrulanması için, Sunucu bir Uzaktan Bildirim Hizmeti kullanarak bir birinci kanalın üzerinden şifrelenmiş bir oturum tanımlayıcısını gönderir. SDK, bu oturum tanımlayıcısını kullanarak bir Kimlik Doğrulama Kodu oluşturur ve bunu bir Sunucu'ya (Başlangıç sırasında sağlanan ve SDK'da depolanan bir URL_RM kullanarak) ikinci bir kanal kullanarak SSL / TLS üzerinden iletir. Sunucu, karşılıklı kimlik doğrulama işlemini tamamlamak için kimlik doğrulama kodunu doğrular.

4.3.6 Sunucu Anahtarlarının Üretilmesi ve Paylaşılması

CLOUD_EC_PUB (HOST_X, HOST_Y) ve CLOUD_EC_PRV 'den oluşan CLOUD_EC_INIT adlı anahtar, sunucuda HSM aracılığıyla oluşturulur, ardından CLOUD_EC_PUB (HOST_X, HOST_Y) olarak SDK parametresi olarak kullanması için MPA geliştiricisi ile paylaşılır. **Şekil 4.6** bu akışı açıklamaktadır.

Şekil 4.6 : Sunucu EC anahtar oluşturulması ve paylaşımı



4.3.7 SDK ve Sunucu Anahtar Üretimi

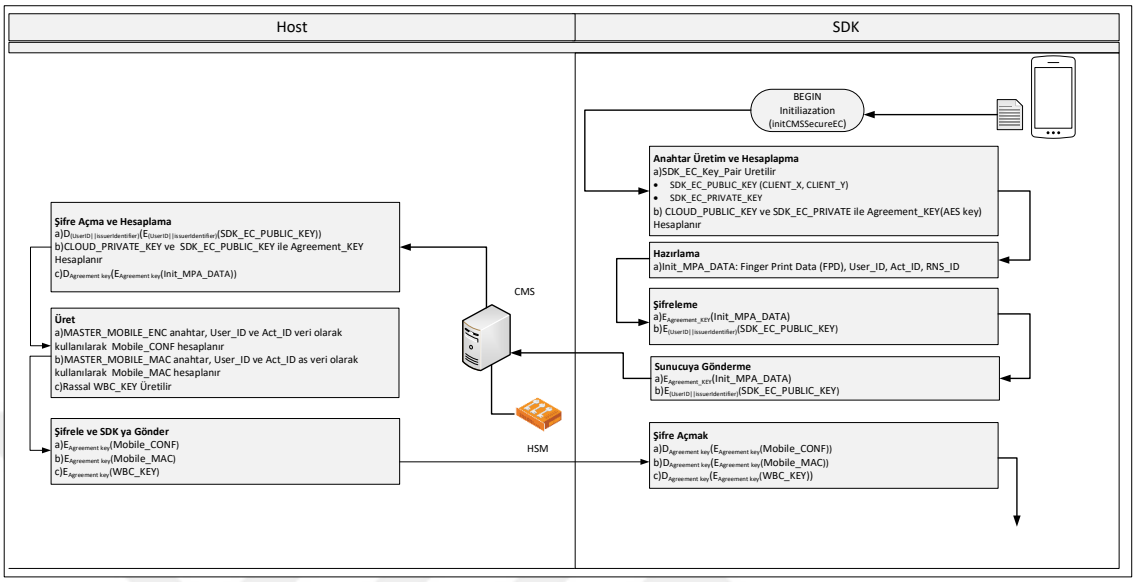
İlk kurulum işleminde, Agreement_KEY (256 bitAES) oluşturmak için Eliptik Eğri Şifreleme (ECDH) Anahtar Sözleşmesi protokolünü kullanılır. SDK ile Sunucu arasında kuruluma özel hassas verileri şifrelemek ve korumak için Agreement_KEY kullanılır. SDK ile Sunucu arasında şifrelenmiş veriler aşağıda belirtilmiştir.

i. **SDK'dan Sunucuya Gönderilen Veriler:** Activation_ID, Versiyon ve Mobil Cihaz Parmakizi bilgileridir.

ii. **Sunucu'dan SDK'ya Gönderilen Veriler:** Mobil Conf Key, Mobil MAC Key, WBC Key, MPA_ID ve URL bilgileridir.

Şekil 4.7 Sunucu ve SDK / MPA arasındaki anahtar değişimini göstermektedir.

Şekil 4.7 : SDK ve sunucu arasında anahtar oluşturma



SDK ile Sunucu arasında anahtar değişimi yapmak için ECDH Anahtar Anlaşması protokolünü kullanılır.

SDK, SDK_EC_PUB (CLIENT_X, CLIENT_Y) ve SDK_EC_PRV'den oluşan SDK_EC_KEY_PAIR üretir, daha sonra SDK, SDK_EC_PRV ve CLOUD_EC_PUB değerlerini kullanarak Agreement_KEY 'yi hesaplar.

Init_MPA_DATA SDK tarafından hazırlanır ve Agreement_KEY ile şifrelenir. SDK_EC_PUB, SHA256 algoritması kullanılarak Kullanıcı Kimliği ve Düzenleyici Tanımlayıcısının birleştirilmesiyle şifrelenir.

Şifreli Init_MPA_DATA ve şifreli SDK_EC_PUB, SDK'dan Host_'a URL_CMS aracılığıyla gönderilir. Aynı zamanda Kullanıcı Kimliği ve Düzenleyici Tanımlayıcısı da URL_CMS ile gönderilir.

Sunucu Kullanıcı Kimliği ve Düzenleyici Tanımlayıcısı kullanarak SDK_EC_PUB şifresini çözer. Sunucu SDK_EC_PUB'yi edindikten sonra, CLOUD_EC_PRV ve SDK_EC_PUB kullanarak Agreement_KEY'i hesaplar. Sunucu ve SDK aynı

Agreement_KEY'e sahip olduklarından birbirleriyle güvenli bir şekilde iletişim kurabiliyor hale gelir.

Sunucu Mobile_CFD, Mobile_MAC ve rassal WBC_KEY oluşturur ve hepsini Agreement_KEY ile şifreler, ardından SDK'ya gönderir.

SDK şifreli Mobile_CFD, Mobile_MAC ve rasgele WBC_KEY'i aldığında, anahtarlar SDK tarafından Agreement_KEY kullanılarak şifresi açılır.

4.4 SDK YEREL GÜVENLİK TASARIMI

4.4.1 JNI / Yerel Kütüphane

Java Native Interface (JNI), Java Virtual Machine'de (JVM) çalışan Java kodunun başka dillerde yazılmış olan yerel uygulamalar ve kütüphaneler tarafından aranmasını ve çağrılmasını sağlayan bir programlama çerçevesidir.

Sistemin JNI kütüphanesi C, C++ ve Assembler programlama dillerinde yazılmalıdır. Kriptografik algoritmalar JNI kütüphanesinde uygulanmalıdır. Şifreleme algoritmaları ancak JNI ve SDK arasında ve JNI ile MPA arasında başarılı Karşılıklı Kimlik Doğrulama sonrasında kullanılabilir hale gelmelidir.

JNI Kütüphanesinde uygulanması gereken şifreleme algoritmaları aşağıda listelenmiştir:

i. **AES (ECB, CBC, CTR)**
   a. AES ECB ve CBC verileri şifrelemek için kullanılır.
   b. AES CTR sunucu ve SDK arasında paylaşılan verileri şifrelemek / korumak için kullanılır.

ii. **DES (ECB, CBC)**
   a. DES ECB ve CBCişlem sırasında kullanılır.

iii. **DES3 (ECB, CBC)**
   a. DES3ECB ve CBCişlem sırasında kullanılır.

iv. **MAC (AES, DES3)**
   a. MAC AES sunucu ve SDK arasında veri bütünlüğü korumak için kullanılır.
   b. MAC DES3, işlem sırasında kriptogram üretmek için kullanılır.

v. RSA

a. RSA, CDA imzası ve fDDA imzası hesaplamak için kullanılır.

vi. SHA1

a. SHA1, CDA imzasını ve fDDA imzasını hesaplamak için kullanılır.

vii. SHA256

a. SHA256 Kimlik Doğrulama Kodu, Anahtar Dağıtım anahtarı, Mobil Parmakizi verileri, Cihaz Bağımlı Anahtarı, vb. oluşturmak için kullanılır.

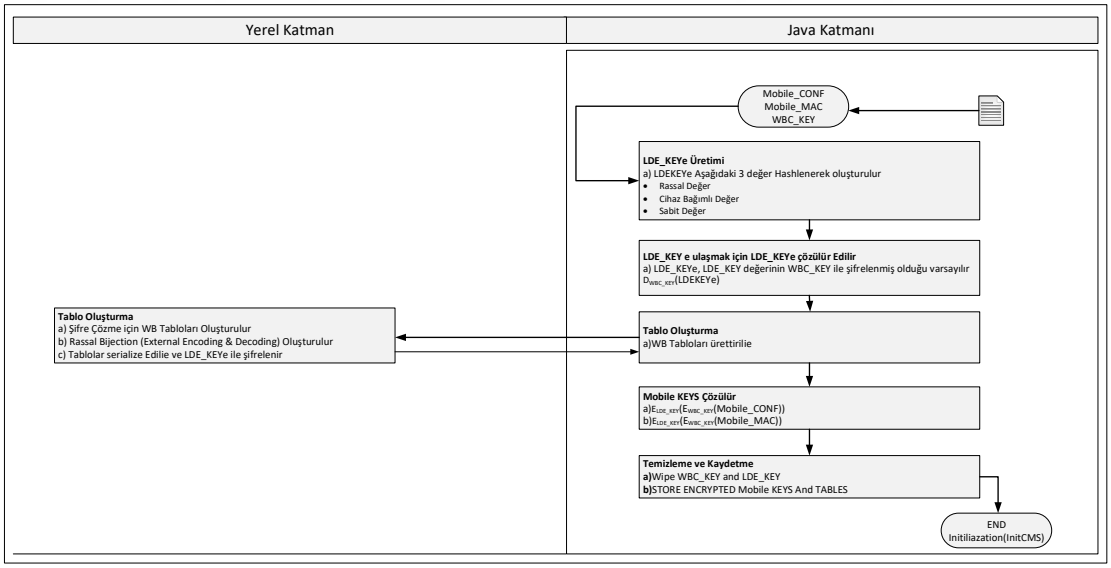
viii. Beyaz Kutu Şifreleme

JNI kütüphanesi, şifreleme ve özetleme algoritmalarının Java katmanından daha hızlı gerçekleştirilmesini sağlar, diğer yandan kolayca tahmin edilerek çözülmesini önler.

4.4.2 SDK WBC ve LDE Başlatma İşlemi

SDK WBC & LDE başlatma işlemi, SDK Mobile_CFD, Mobile_MAC ve WBC_KEY'yi aldıktan sonra başlar **Şekil 4.8** , SDK WBC ve LDE başlatma işlemini göstermektedir.

Şekil 4.8 : SDK WBC ve LDE oluşturma



SDK, SHA256 algoritması ile rastgele değer, cihaza bağlı değer ve sabit kodlanmış değer kullanarak LDE_KEYe oluşturur. LDE_KEYe, WBC_KEY ile şifrelenmiş LDE_KEY biçimi olarak kabul edilir.

Bir sonraki adımda, LDE_KEYe WBC_KEY kullanılarak şifrelenmiştir ve şu anda açık LDE_KEY SDK'da bulunmaktadır.

Mobil anahtarlar (Mobile_CFD ve Mobile_MAC) WBC_KEY ile şifrelenir. Şifreli mobil anahtar tekrar LDE_KEY ile şifrelenir. "E_{LDE_KEY} (E_{WBC_KEY} (MobileKeys))" olarak gösterilmiştir.

SDK, Yerel katmandan WB tabloları ister. Daha sonra Yerel katman, WBC_KEY için şifre çözücü olarak WB tabloları ve harici kodlama ve kod çözme için rasgele önyükleme tablolarını oluşturur. Beyaz kutu tabloları ve rastgele önyükleme tabloları LDE_KEYe ile şifrelenir ve SDK'ya gönderilir. Sonra WBC_KEY yerel katmanda temizlenerek silinir.

Şifrelenmiş mobil anahtarlar, WB tabloları ve LDE_KEY şifreli rasgele önyükleme tabloları SDK'da depolanır. Daha sonra java katmanında WBC_KEY ve LDE_KEY temizlenerek silinir.

4.4.3 Beyaz Kutu Şifreleme

Tasarlanan beyaz kutu şifreleme akışı bu bölümde açıklanmıştır.

Dahili ve Harici kimlik doğrulama sırasında oturum anahtarı üretimi Java ve JNI arasında gerçekleştirilir. Başlatma adımında, sunucu tarafından rastgele bir WBC_KEY oluşturulur ve Sözleşme Anahtarı Şifrelemesi ile SDK'ya aktarılır.

WBC_KEY, aşağıdaki tabloları oluşturmak için Java'dan JNI'ye iletilir:

i. Bir WBC tablosu (Tbl) +
ii. Harici bir önyükleme kodlama tabloları (EE) +
iii. Harici bir önyükleme kod çözme tabloları (ED)

"EE", F işlevi içindir.
"Tbl" sadece şifre çözme içindir.
"ED", F ^ (- 1) işlevi içindir.

Notasyonlar:

• Kodlama notasyonu X_EE olarak yazılmıştır (X, F ile kodlanmıştır).

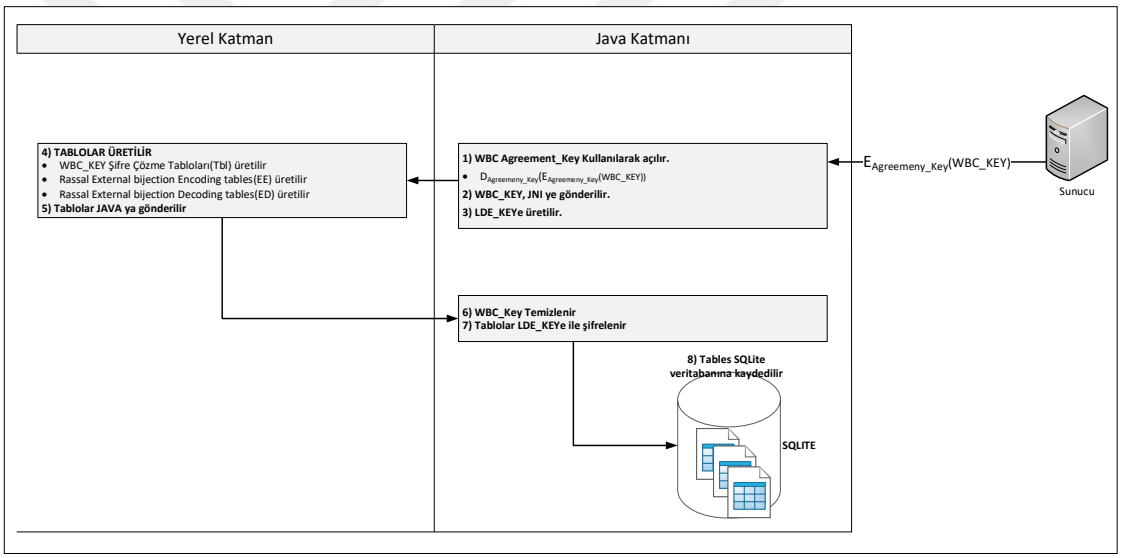
• Kod çözme notasyonu X_DE olarak yazılmıştır (X, F ^ (- 1) ile deşifre edilmiştir).

Bu işlem sırasında, mümkün olduğu kadar erken WBC_KEY değeri temizlenerek silinir.

Java, WBC tablolarını, EE tablolarını ve ED tablolarını serileştirilmiş, biçimlendirilmiş ve LDE_KEYe ile şifrelenmiş şekilde alınırr ve SQLite'a eklenir.

Şekil 4.9, WBC anahtarını ve tablo akışı oluşumunu göstermektedir.

Şekil 4.9 Beyaz kutu şiftreleme akışı



Uygulama başlarken sırasında Java, Tabloları SQLite'den okur ve JNI'ya iletir. JNI formatları kontrol eder ve tabloları yükler.

WBC tabloları, WBC_ KEY ve rasgele oluşturulmuş dış önyüklemelere karşılık gelir.

Rastgele oluşturulmuş bu dış önyüklemelerin bir sonucu olarak, JNI, aynı WBC_ KEY değeri için bile her arama için tablo işlevi farklı değerler döndürür.

Bu nedenle, Sunucu her bir Aktivasyon için rastgele WBC_Key üretir.

Her bir Aktivasyon için, WBC tabloları da farklıdır.

Her bir Aktivasyon için, bu tablolar yaşam döngüsü boyunca veritabanında tutulur.

4.4.4 MPA / SDK ve JNI Karşılıklı Kimlik Doğrulama

Global Platform Card Spesifikasyon Sürümü 2.1.1, Güvenli Kanal Formatı 2 bölümünde açıklandığı gibi Karşılıklı Kimlik Doğrulama, aynı zamanda Güvenli Kanal yaratacak şekilde geliştirilmelidir.

Spesifikasyondan farklı olarak, her adım için DES3 algoritması yerine AES algoritmasını kullanılmalıdır.

Karşılıklı doğrulama ile başlatma işlemi sırasında iki farklı güvenli kanal kurulur

i. MPA ve JNI kütüphanesi (Harici Kanal)
ii. SDK ve JNI kütüphanesi (Dahili Kanal)

Karşılıklı kimlik doğrulama işlemleri başarıyla tamamlandıktan sonra JNI, şifreleme işlevlerini yerine getirmeye başlar, aksi takdirde bu işlevleri yerine getirmez.

Güvenli Kanallar, JNI ve SDK'nın izinsiz kullanılmasını önler. SDK ve MPA, JNI'yi doğrular ve JNI ayrıca SDK ve MPA'yı doğrular.

Güvenli kanallar JNI ile herhangi bir veri değişimi için kullanılabilir olmasına rağmen, performans nedeniyle sadece WBC Key dönüşümü için kullanılmalıdır.

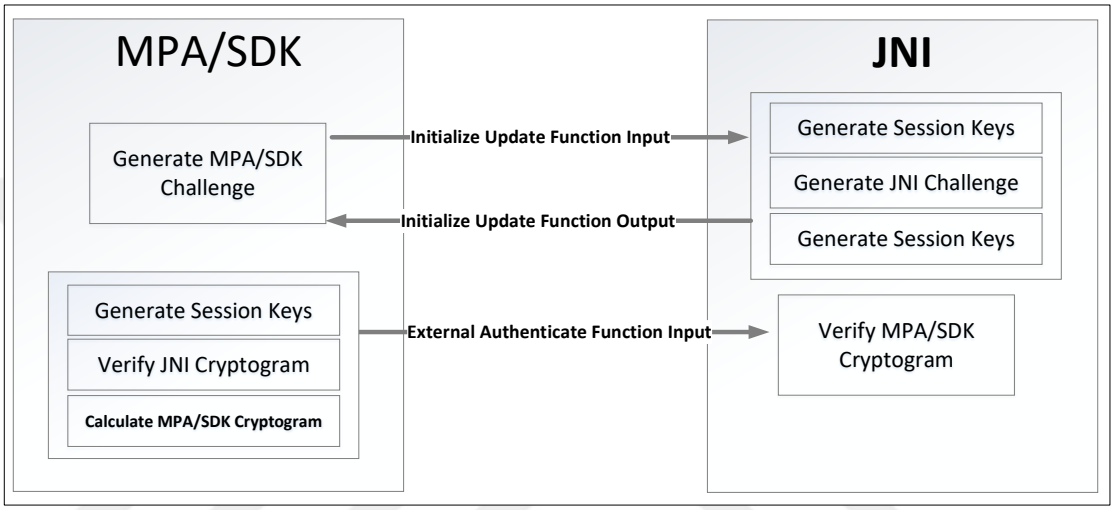
Karşılıklı Kimlik Doğrulama için İlk Kimlik Doğrulama Anahtarları dinamik olarak oluşturulur. Her iki taraf aynı temel limlik doğrulama anahtarlarına sahiptir.

MPA ve JNI ile harici kimlik doğrulama sırasında, MPA ve JNI'nın kimliğini doğrulamak için İmza Aktivasyon Kodu kullanılır. SDK, MPA'nın orijinal imzayla imzalanmış olup olmadığını kontrol eder. MPA, yeniden paketleme kullanarak bir saldırgan tarafından özelleştirilir veya değiştirilirse ve orijinal imzayla imzalanmamışsa, JNI, MPA'nın

kimliğini doğrulayamaz. JNI, harici kimlik doğrulama olmadan hizmet vermeye başlamaz.

Şekil 4.10, MPA / SDK ve JNI arasında gerçekleşen karşılıklı kimlik doğrulamasını göstermektedir.

Şekil 4.10 : MPA / SDK ve JNI karşılıklı kimlik doğrulama



4.4.5 SDK Üzerindeki Ortam Kontrolleri

ADB Etkinleştirme Algılama, Kök Algılama ve Kanca Algılama ortam denetimleri Yerel Katman ve Java Katmanında uygulanmıştır. Herhangi bir ortam kontrolü pozitif olarak tespit edilirse, JNI hizmet vermek için sonlandırılır.

4.4.5.1 Köklenme Algılama

Java bölümünde, köklenme algılama mekanizması, Bilinen Sistemsiz Kök Dosyalarını kontrol etmek şeklinde uygulanır.

JNI kütüphanesi bölümünde, kök algılama mekanizması, potansiyel su, BusyBox, Magisk Dosyaları ve diğer şüpheli girişler için multithreaded dosya denetimi yapılarak uygulanmalıdır. Benzer şekilde rasgele su uygulama kontrolleri de yapılmalıdır.

Karşılıklı Kimlik Doğrulama sırasında köklenme mutlaka kontrol edilmelidir. Ayrıca JNI Native Layer'da kök tespiti de rastgele ve multithreaded kontrol edilmelidir.

Kök algılama, Mobil Uygulama Durumu durumu değiştiğinde (Duraklat veya DevamEt), çalışma zamanında da kontrol edilmelidir.

Uygulama etkinliğini sürdürdüğünde, kök kontrolü yapılmalıdır. Uygulamanın Etkinlik Durumu ApplicationLifecycleCallbacks geri çağrı kullanılarak geri bildirilmelidir.

4.4.5.2 Kancalama Algılama

Hem JNI kitaplığında hem de Java Katmanında uygulanan kanca algılama mekanizmaları uygulanmalıdır.

Java bölümünde, kanca algılama mekanizması uygulanması aşağıdaki gib olmalıdır :

i. Bilinen Kancalama Paketlerinin Kurulumu Kontrolü,
ii. StackTrace'te Bilinen Kancalama İşlemlerini Kontrol Etme,
iii. Bilinen Bağlama Nesnelerini Kontrol Etme

JNI kütüphane kısmında, kanca tespit mekanizması uygulanması aşağıdaki gib olmalıdır:

i. Multithreaded olarak Frida yerel portları denetlenir.
ii. Multithreaded oalrak Bellek Haritasında şüpheli işlemler kontrol edilir.
iii. Bu tespit yöntemleri JNI kütüphanesinde ASM dilinde, bir köklendirme aracı kullanarak kanca takılmasını önlemek için uygulanır.

4.4.5.3 Ortam Kontrolü Zamanları

Ortam kontrolleri aşağıdaki durumlarda özellikle yapılmalıdır :

i. Seçili etkinlikler'in geri çağrıları sırasında,
ii. Önemli işlemler sırasında,
iii. RNS mesajı alındığında,
iv. Uygulama durumu çalışma zamanında değiştiğinde (Devam veya Duraklat)

4.4.5.4 Ortam Kontrolü PozitifTespit Davranışı

Ortam kontrolü'nde herhangi bir pozitif durum tespit edildiğinde, JNI sonlandırılmalıdır. Ardından, Kimlik Doğrulama işlemi başarısız hale getirilerek, karşılıklı kimlik doğrulama gerçekleşmez ve JNI herhangi bir şifreleme işlevini yerine getiremez hale gelmelidir.

4.4.6 Güvenli Yerel Veritabanı

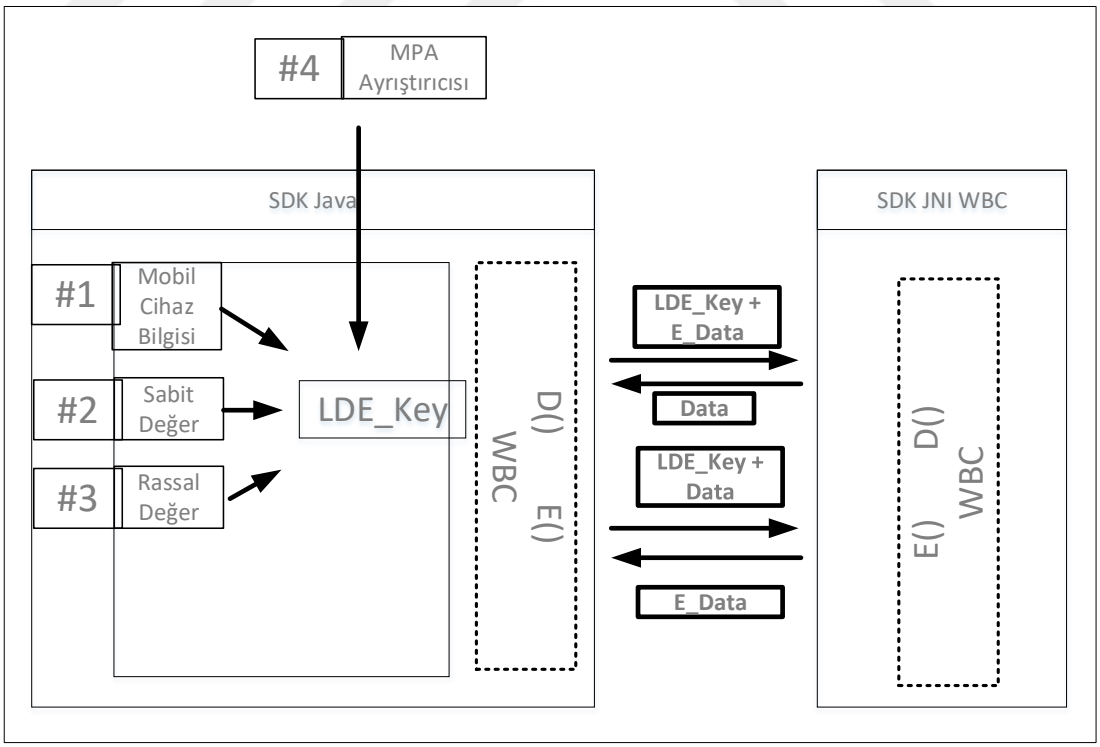
Android saklama ortamlarında, verileri açık olarak değil, şifreli olarak saklamak gerekmektedir. Böylece veritabanında depolanan verilere erişim, kullanıcının anahtarı bilmesine gerek kalmadan MPA tarafından herhangi bir zamanda güvenli bir şekilde yapılabilir. SDK bunun için arayüzlere sahip olmalıdır.

Yerel veritabanı şifrelemesi en etkin şekilde beyaz kutu şifreleme (WBC) sistemi kullanarak saklanmalıdır. Depolama anahtarının (ST_Key) aşağıdaki bilgilerden çeşitlendirilmiş ve dinamik bir şekilde oluşturmak doğru olacaktır:

i. SDK nın ürettiği Mobil cihaz bilgisi [Mobil Cihaz Parmak İzi (LDE_FGP)],
ii. SDK kaynak koduna gömülü sabit değer [SDK Sabiti (LDE_SDK)]
iii. SDK tarafından rasgele üretilmiş değer [SDK Rasgele Değeri (LDE_RND)],
iv. MPA kaynak koda gömülü ayrıştırıcı [Mobil Uygulama Filigranı (LDE_MPA)]

Şifrelenmiş Yerel Veritabanı (LDE) için işlevsel görünüm **Şekil 4.11** ile verilmiştir.

Şekil 4.11 : LDE işlevsel görünüm



5. TARTIŞMA VE SONUÇ

Bugün akıllı telefonlar, kullanıcılara müzik, oyun, dijital görüntüleme, küresel konumlandırma sistemleri, sosyal ağ, İnternet tarama, e-posta ve yazılı mesajlaşma dahil olmak üzere sürekli genişleyen bir dizi özellik ve yetenek sunmaktadır.

Akıllı telefonların elektronik ödemeler endüstrisi üzerinde önemli bir etkiye sahip olmaları beklenmektedir; çünkü günlük işlemleri hantal nakitten kartlara ve diğer elektronik ödeme biçimlerine taşımak için acil ve güçlü bir ihtiyacı doldurmaktadırlar.

Dört ana mobil ödeme türü öngörülmektedir:

i. Kart Emülasyonu: Mobil cihazın temassız mobil bir kart gibi kullanılması,
ii. Mobil Web Ödeme: Sanal bir kartla işlem gerçekleştirmek için mobil cihazın tarayıcı veya uygulama yeteneklerinin kullanılması,
iii. Mobil Uzaktan Ödeme: Mobil cihazın güvenli sanal ödeme işleminde kullanılması,
iv. Mobil Kart Kabulü: Mobil cihazın satış noktası terminali olarak kullanılması.

Mutlak bir güvenliğin olmadığı dünyada, mobil uygulamalarda güvenliği katmanlar halinde gerçekleştirerek, zaman ve uzayda uzaklık saldırganın maliyetini yükseltmek mümkündür.

Hassas verilerin sınırlı bir şekilde tutulması, işlemlerin çevrimiçi olması sayesinde arka uçta risk yönetimi ve güvenlik kontrolleri yapılması da gerekmektedir.

Unutulmamalıdır ki, güvenlik hareketli bir hedeftir ve geliştiriciler süreci saldırganların ve saldırıların ötesine taşımalı ve sürekli güvenlik değerlendirmesi yapmalıdır.

Akıllı telefonlar güvenli platformlar değildir, Android işletim sistemi sürekli tehdit ve düzenli saldırı altındadır. Özellikle mobil ödeme uygulamaları ek güvenlik unsurları gerektirmektedir. Güvenli uygulamalar geliştirme ihtiyacı sürekli yeni kavramlar ortaya çıkmaktadır. Bu anlamda güvenlik sonsuz bir çabadır, bu bakış açısıyla sürdürülebilir değerlendirme ve dönüşüm potansiyel tehditleri öngörmeye, saldırıları tanımlamaya ve

etkileri azaltmaya yardımcı olacaktır.

Saldırılara karşı yapılacak çalışmalarda göz önüne alınması gerekenler şunlardır:

i. Saldırganlara karşı sonsuz psikolojik ve teknolojik savaş gerekmektedir
ii. Yeni bir sürüm yayınlamak için geliştirici zamanı / çabası gerekmektedir
iii. Saldırganın zarar görmesi / engellenmesi sağlanmalıdır
iv. Saldırganın zaman ve finansal bütçesi zorlanmalıdır
v. Varlıklar bölümlere ayrılarak hasarlar sınırlanmalıdır
vi. Çevik geliştirme döngüsü kullanılaraki saldırganların önünde kalınmalıdır

Tasarımın gerçekleştirilmesi sonucunda, Android cihazların ödeme sistemlerinde kullanılmasını sağlayan bir SDK geliştirilmiştir.

SDK'nın ödeme sistemlerindeki güvenlik değerlendirmesi bir Güvenlik Labaratuvarı tarafından yapılmış olup, testleri başarı ile tamamlamıştır (Khan, 2019) .

Bu değerlendirme için SDK içindeki fonksiyonları yerine getirmek için ayrıca güvenlik önlemi olmayan basit bir uygulama kullanılmıştır.

Sızma testi sonuçları, üçüncü taraf yazılım koruma araçları ve SDK'da uygulanan güvenlik mekanizmalarının üst düzey bir güvenlik düzeyi sağladığını göstermiştir.

Karışık yerel katmanlarda uygulanan şifreleme işlevleri ve WBC kullanımı, genel güvenlik seviyesini arttırmıştır.

TOE'nin çevre kontrollerini hedef alan penetrasyon testleri, TOE'nin kullanıma hazır rootkit'leri ve dinamik ikili enstrümantasyon araçlarını tespit edebildiğini göstermiştir.

Hassas verilerinin çıkarılması gibi daha ileri saldırılar için bu güvenlik kontrollerini atlamak için bu araçların tersine mühendislik ve kişiselleştirilmesinin gerekli olduğu bulunmuştur.

Bu güvenlik kontrollerini atlamanın her bir tekil aktivasyon için kabaca iki hafta çaba gerektirdiği tahmin edilmiştir.

Öte yandan, Java katmanındaki tersine mühendislik ve ikili araçların bir saldırganın anahtar hiyerarşisinde en üst seviyede bulunan kriptografik varlıkların bazılarını çıkarmasına yardımcı olabileceği belirlenmiştir.

Bu anahtarlardan yararlanmak için, saldırgana eninde sonunda hassas verileri elde etmek için bir fırsat verebilecek anahtar türetme mekanizmalarını ortaya çıkarmak için yerel katmanlarda ek ters mühendislik çalışmaları yapılması gerektiği görülmüştür.

Sonuç olarak çevre kontrollerini atlamak da dahil olmak üzere her bir tekil aktivasyon için altı haftadan daha fazla çaba harcanacağı tahmin edilmiştir.

KAYNAKÇA

Diğer Yayınlar

APKMirror, 2019. *ApkMirror.* [Çevrimiçi]
Available at: https://www.apkmirror.com/
[Erişildi: 01 07 2019].

APKPure, 2019. *ApkPure.* [Çevrimiçi]
Available at: https://apkpure.com
[Erişildi: 01 07 2019].

APKTool, 2018. *Github APK Tool.* [Çevrimiçi]
Available at: https://github.com/iBotPeaches/Apktool
[Erişildi: 01 07 2019].

Arshad, S. S. M. A. K. A. a. A. M., 2016. Android malware detection & protection: a survey.

Avast, 2019. *Avast.* [Çevrimiçi]
Available at: https://blog.avast.com/avast-mobile-threat-predictions
[Erişildi: 01 07 2019].

Avtivity, A., 2019. *Android Developers Activity.* [Çevrimiçi]
Available at: https://developer.android.com/reference/android/app/Activity
[Erişildi: 01 07 2019].

Baek, C. H., Cheon, J. H. & Hong, H., tarih yok White-box AES implementation revisited. 18(3).

Brady, P., 2008. *Android Anatomy and Physiology.* -, Google I/O Developer Conference.

Broadcast, A., 2019. *Android Developers BroadcastReceiver.* [Çevrimiçi]
Available at:
https://developer.android.com/reference/android/content/BroadcastReceiver
[Erişildi: 01 07 2019].

Content, A., 2019. *Android Developers Content Providers.* [Çevrimiçi]
Available at: https://developer.android.com/guide/topics/providers/content-providers.html
[Erişildi: 01 07 2019].

Dashboard, A., 2019. *Android.* [Çevrimiçi]
Available at: https://developer.android.com/about/dashboards/index.html
[Erişildi: 01 07 2019].

Data Storage, A., 2019. *Android Developers Data Storage.* [Çevrimiçi]
Available at: https://developer.android.com/guide/topics/data/data-storage.html
[Erişildi: 01 08 2019].

Developer, A., 2019. *Android Developer AAR.* [Çevrimiçi]
Available at: https://developer.android.com/studio/projects/android-library
[Erişildi: 01 07 2019].

Felt, A. P. F. M. C. E. H. S. a. W. D., 2011. A survey of mobile malware in the wild. 1st ACM workshop on Security and privacy in smartphones and mobile devices.

Fragment, A., 2019. *Android Developers Fragment.* [Çevrimiçi]
Available at: https://developer.android.com/reference/android/app/Fragment
[Erişildi: 01 07 2019].

FSecure, A., 2014. *FSecure.* [Çevrimiçi]
Available at: https://www.f-secure.com/documents/996508/1030743/Mobile_Threat_Report_Q1_2014.pdf
[Erişildi: 01 07 2019].

Heger, D. A., 2012. *Mobile Devices-An Introduction to the Android Operating Environment Design, Architecture, and Performance Implications.* -: DHTechnologies (DHT).

Intent, A., 2019. *Android Developers Intent.* [Çevrimiçi]
Available at: https://developer.android.com/reference/android/content/Intent
[Erişildi: 01 07 2019].

Ismail, N. S. S. H. Y. R. a. A. M. F., 2017. General Android Malware Behaviour Taxonomy.

Kaspersky, 2016. *Kaspersky.* [Çevrimiçi]
Available at: https://www.kaspersky.com.tr/resource-center/preemptive-safety/android-security-tips
[Erişildi: 01 07 2019].

Kaspersky, 2018. *KasperskyMC.* [Çevrimiçi]
Available at: https://www.kaspersky.com.tr/resource-center/threats/malware-classifications
[Erişildi: 01 07 2019].

Khan, A., 2019. *CAST Security Evaluation for Simant mPOS CTLS on COTS SDK v2.0.4.0 for Android OS 5.0+,* Delft: Brightsight, B.V..

KSafety, 2019. *Kaspersky.* [Çevrimiçi]
Available at: https://www.kaspersky.com.tr/resource-center/preemptive-safety/money-online
[Erişildi: 01 07 2019].

KSmart, 2018. *Kaspersky Smartphone Threats.* [Çevrimiçi]
Available at: https://www.kaspersky.com.tr/resource-center/threats/smartphones
[Erişildi: 01 07 2019].

Liang, 2010. *System Integration for the Android Operating System.* -, National Taipei University.

Maker, F. C. Y., 2010. *A Survey on Android vs. Linux.* -, University of California.

Manifest, A., 2019. *Android Developers Manifest.* [Çevrimiçi]
Available at: https://developer.android.com/guide/topics/manifest/manifest-intro
[Erişildi: 01 07 2019].

McAfee, 2018. *McAfee.* [Çevrimiçi]
Available at: https://www.mcafee.com/enterprise/en-us/assets/reports/rp-mobile-threat-report-2019.pdf
[Erişildi: 01 07 2019].

Permissions, A., 2019. *Android Developers Permissions.* [Çevrimiçi]
Available at: https://developer.android.com/guide/topics/permissions/overview
[Erişildi: 01 07 2019].

Release, A., 2019. *Android Developers Publish.* [Çevrimiçi]
Available at: https://developer.android.com/studio/publish/preparing
[Erişildi: 01 07 2019].

Resources, A., 2019. *Android Developers Resources.* [Çevrimiçi]
Available at: https://developer.android.com/reference/android/content/res/Resources
[Erişildi: 01 07 2019].

Service, A., 2019. [Çevrimiçi]
Available at: https://developer.android.com/reference/android/app/Service
[Erişildi: 01 07 2019].

SQLite, 2019. *SQLite.* [Çevrimiçi]
Available at: https://www.sqlite.org/about.html
[Erişildi: 01 08 2019].

Statistics, A., 2019. *Statista.* [Çevrimiçi]
Available at: https://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores/
[Erişildi: 01 09 2019].

Sydney, A., 2019. *Sydney Universitesi.* [Çevrimiçi]
Available at: https://sydney.edu.au/news-opinion/news/2019/06/24/over-2-000-fake-android-apps-discovered.html
[Erişildi: 01 07 2019].

View, A., 2019. *Android Developer Views.* [Çevrimiçi]
Available at: https://developer.android.com/reference/android/view/View
[Erişildi: 01 07 2019].

Vulnerabilities, 2015. *AndroidVulnerabilities.* [Çevrimiçi]
Available at: http://www.androidvulnerabilities.org/press/2015-10-08
[Erişildi: 01 07 2019].

WBC, 2002. *White Box Crypto.* [Çevrimiçi]
Available at: http://www.whiteboxcrypto.com/
[Erişildi: 01 07 2019].