T.C.

ESKİŞEHİR OSMANGAZİ ÜNİVERSİTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

2017-2018 Eğitim-Öğretim Yılı

152118631- ENGINEERING RESEARCH ON

SIGNAL APPS

FİNAL RAPORU

Proje Başlığı

“Mobil Uygulamalarda Güvenlik Analiz Sistemi”

Projeyi Hazırlayanlar:

“Mehmet Okan TAŞTAN, 152120131058, Bilgisayar Mühendisliği

Ahmet Sefa BULCA, 152120121017, Bilgisayar Mühendisliği

Sevda ÇİMEN, 152120131020, Bilgisayar Mühendisliği

Mustafa ŞAHİN, 152120121063, Bilgisayar Mühendisliği”

Danışman:

Yrd. Doç. Dr. Esra Nergis YOLAÇAN

Sektör Danışmanı (varsa):

“Evosoft Yazılım Bilişim Ltd. Şti, Ozan AYCAN”

İÇİNDEKİLER

Notlar:1) Tüm metinde kırmızı kısımlar açıklama amaçlı eklenmiş olup rapor tesliminde silinebilir.
2) Yeni sayfadan başlamış başlıklar önceki sayfa ile birleştirilmesin(içi doldurulmasa da). Rapor çıktısında da alt kısımlar ayrı kişiler tarafından değerlendirilecektir.

A. PLANLAMA
A.1. ÖZET(ABSTRACT) VE ANAHTAR KELİMELER(KEYWORDS)

ÖZET

Günlük hayatın önemli bir parçası haline gelen akıllı telefonlar, gün geçtikçe artan yetenekleri ve genişleyen özellikleri sayesinde bilgisayar sistemlerinin yerini almaya başlamıştır. Geliştirilen uygulamalar sayesinde günümüzdeki kullanımı fotoğraf makinesinden, navigasyon cihazına, hesap makinesinden, sağlık uygulamalarına kadar birçok farklı alanda birinci sırada yerini almaktadır. Teknolojik gelişmeler ile mobil cihazların maliyetlerinin düşmesi bu cihazlara erişimi de kolaylaştırmıştır. Akıllı telefonların kullanımının yaygınlaşması ile birlikte geliştirilen uygulamaların sayısı da artmış, ancak zararlı uygulamaların sayısında daha da büyük bir artış gözlenmiştir. Saldırganlar tarafından mobil cihazlara yönelik olarak gerçekleştirilen saldırılardan biri de mobil uygulamaların içerisine zararlı yazılımların enjekte edilmesidir [1]. Mobil cihazlara bulaşan söz konusu zararlı yazılımlar tüm rehber bilgisini ele geçirebilmekte, kullanıcıya ait mesajları okuyabilmekte, resim, video ve ses gibi multimedya dosyalarına erişebilmekte ve kullanıcıya ait konum bilgisini saldırganlara iletebilmektedir. Ayrıca sisteme bulaşmış olan zararlı yazılımlar, sahip oldukları yeteneklere göre; telefon görüşmelerini anlık olarak dinleme, kullanıcının bulunduğu ortamın dinlemesi, internet aktivitelerinin takip edilmesi gibi işlemleri gerçekleştirebilmektedir [1].

Zararlı yazılımlara 1970'lerden itibaren rastlanmaya başlanmasına rağmen 2004'te mobil dünyaya girmiştir. Kaydedilmiş ilk mobil zararlı yazılım olan Cabir Nokia 6600, N-Gage, Panasonic X700 ve Siemens SX1 gibi bluetooth'u açık cihazlar üzerinden yayılmaya başlamıştır. Bu solucan, zararsız olmasına rağmen telefona girdikten sonra ekranda 'Caribe' metni çıkarıyordu. Sonrasında ise yakınındaki bluetooth'u açık olan herhangi bir cihaza sıçrayarak yayılıyordu. Mobil sistemlerin gelişimi ile bu zararlı yazılımlar artış göstermiştir ve 2010 yılında Android platformu için ilk zararlı yazılım ortaya çıkmıştır. 'FakePlayer' isimli bu yazılım, Rusya'daki özel tarifeli numaralara pahalı SMS'ler gönderiyordu. 2016 yılında ise 'Pokemon Go' ile bir çok sahte uygulamayla birlikte 'adware' gibi zararlı yazılımlar yayılmıştır.

Bu denli artan güvenlik açığı sorunu günümüz mobil uygulamalarında büyük bir risk haline gelmiştir. Geliştirilmesi planlanan sistem ile bu risklerin son kullanıcıya raporlanması ve uygulamalar kullanılmadan önce bu risklerin test edebileceği bir ortam oluşturulması hedeflenmektedir. Bu işlem ise mobil uygulama güvenlik analizlerinde kullanılan dinamik analiz metodu ile gerçekleştirilecektir. Bu analiz yönteminin kullanılmasının sebebi ise; statik analiz metodunun aksine çalışan uygulamaların yaptıkları sistem çağrılarının, ağ trafiğinin, bellek kullanımının ve işlemci kullanımının analiz edilebilmesidir ve kod karıştırma gibi tekniklere karşı işlevsel olmasıdır. Çalışmamızın özgün değeri ise, geliştiricilerden çok son kullanıcılara hitap edecek olmasıdır. Bu amaç doğrultusunda dinamik analiz metodları kullanılarak kullanıcıya basit bir arayüz ile uygulamaların güvenlik analizi raporları sunulması hedeflenmektedir. Ulusal güvenliğin öneminin her gecen gün arttığı bir dönemde, çoğu yabancı kaynaklı olan zararlı yazılımların tespit edilmesi için yapılacak olan milli projenin, ülkemizde bu alandaki açığı kapatarak katkı sağlanması amaçlanmıştır.

Anahtar Kelimeler: Android, Dinamik Analiz, Statik Analiz, Mobil Uygulama Güvenliği, Zararlı Yazılım

ABSTRACT

In today's computing world, mobile devices are beginning to subsitute on the ever-increasing capabilities and expanding features of computer systems. With enhanced applications, today's usage takes place in many different areas, from cameras to navigation devices to calculators to healthcare applications. The increase in the use of mobile devices also resulted in the same increase in attacks against such devices. One of the attacks against mobile devices by attackers is the injection of malicious software into mobile applications [1]. The malicious software that is transmitted to the mobile devices can capture all the guide information, read the user's messages, access the multimedia files such as picture, video and audio and transmit the user's location information to the attackers.In addition, harmful software infected by the system can be classified according to their abilities like; listening to phone calls instantly, resting on the internet, continuing internet activities. User-related information about mobile applications has led to an increase in the development of malicious software for different abilities for mobile devices [1].

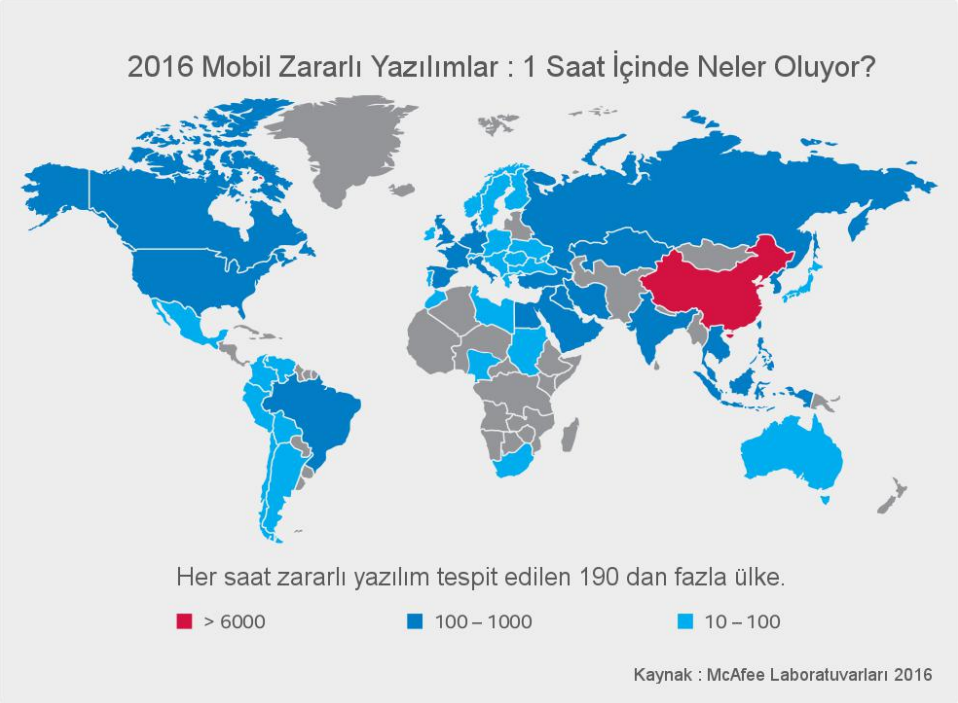
In 2004, the malware entered the mobile world, even though it began to be seen from the 1970s. Cabir Nokia 6600, N-Gage, Panasonic X700 and Siemens SX1 have begun to spread Bluetooth-enabled devices. Despite this worm being harmless,the text 'Caribe' appeared on the screen after it entered the phone. Later, the worm was spreading to any nearby device which device bluetooth properties was open. The development of the mobile systems that followed and the malicious software increased more and the first malicious software for Android appeared in 2010.This software, named 'FakePlayer', and it sent expensive SMS’s with special scheduled number to Russia. In 2016, a lot of malicious software case lived with Pokemon Go. These were scareware (software that does not have any benefits but promises to give the user the best protection against viruses), and adware (ad-supported computer software) that sweeps away any useless but promising viruses.

This increasing security vulnerability has become a major risk in today's mobile applications We intend to create an environment in which we can test these risks to the end user and test the reliability of the application and the risks it may create. We are going to do this using the dynamic analysis method used in mobile application security analysis. The reason why we use this analysis method is; the ability to analyze system calls, network traffic, memory usage, and processor usage of running applications, and thus maintains its function against techniques such as code blending. The value of our work is the system will usable for the end users as well as developers. For this purpose, using dynamic analysis methods, it is aimed to present security analysis reports of applications with a simple interface to the user. In a period of increasing national security precaution every other day, the national proj- ect that will be used to detect most foreign sources of malware is aimed at contributing by closing the gap in this area.

Keywords: Android, Dynamic Analysis, Static Analysis, Mobile Application Security, Malware

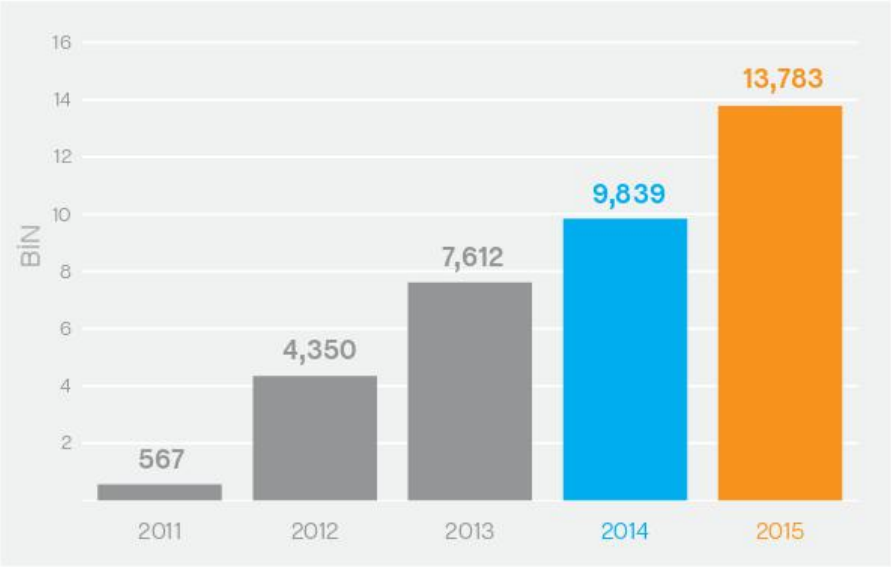
A.2. BİLGİ GEREKSİNİM BELİRLEME, PROBLEMİN TANIMLANMASI

Günümüzde, zararlı mobil uygulamalar konusunda bir farkındalık oluşmuş olmasına rağmen bu konuda bilinçli kullanıcı sayısı yeterli seviyeye ulaşamamıştır. Bu nedenle zararlı uygulamalar her geçen gün hem artmaya hem de yayılmaya devam etmektedir. 'McAfee Labs' sonuçlarına göre 2016 yılında her saat dünyada yayılan zararlı yazılımların ülkelere göre dağılımı Şekil 1'de gösterilmektedir [2]. Şekilde de görüldüğü üzere, mobil platformlardaki zararlı yazılımlar tüm dünyada farklı oranlarda da olsa etkili olmaktadır. Bu oran, diğer ülkelerle karşılaştırıldığında Türkiye için ortalama bir değerdedir ve tehdit altında olduğumuz görülmektedir. Bu nedenle ulusal güvenliğimiz göz önünde bulundurularak gerekli önlemlerin alınması gerekmektedir.



Şekil 1: 2016 Mobil Zararlı Yazılımlar: 1 Saat İçinde Neler Oluyor? [2]

Şekil 2'de verilen Symantec firmasının araştırmalarına göre 2014 yılında Android platformunda zararlı yazılım çeşidi bir önceki yıla göre %29 artmışken, 2015 yılında %40 artış göstermektedir [3].



Şekil 2: Android Zararlı Yazılım Çeşitlerinin Yıllara Göre Dağılımı [3]

Symantec firmasının 'Norton Mobile Insight' yazılımıyla yapılan taramalar sonucu 2014 yılında zararlı olarak sınıflandırılan uygulama sayısı bir önceki yıla göre 2227 artış gösterirken, 2015 yılında 3944 e çıkmıştır [3]. Araştırma sonucunda, zararlı yazılımlardaki artışın zararsız yazılımlara göre çok hızlı bir artış gösterdiği görülmüş ve elde edilen rakamlar Tablo 1'de gösterilmiştir.

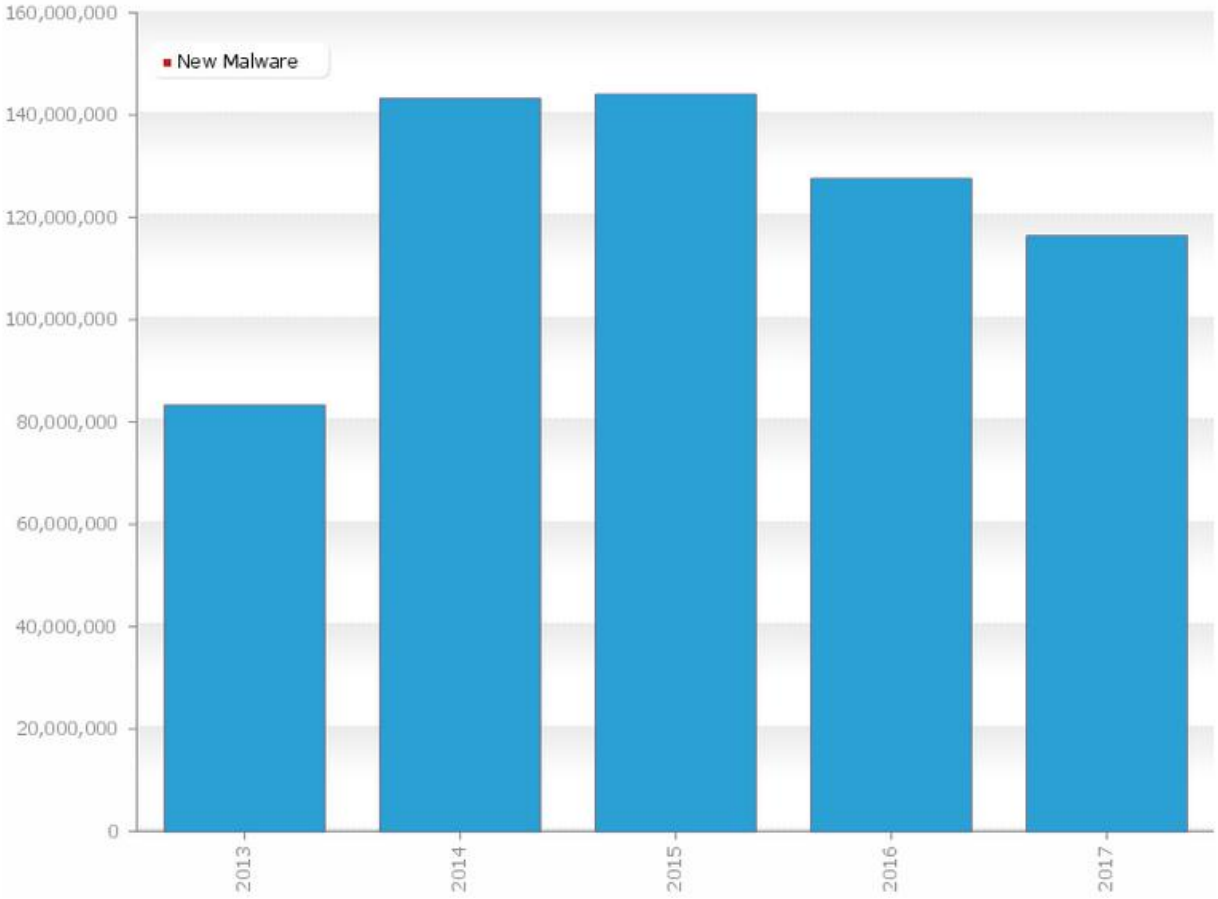
Tablo 1: Symactec Firmasının Mobil Uygulamalarda Güvenlik Analizi Sonuçları [3]

	2013	2014	2015
Toplam Analiz Edilen Uygulama Sayısı	6.1 Milyon	6.3 Milyon	10.8 Milyon
Zararlı Olarak Sınıflandırılan Uygulama Sayısı	0.7 Milyon	1.1 Milyon	3.3 Milyon
İstenmeyen Yazılım Olarak Sınıflandırılan Uygulama Sayısı	2.2 Milyon	2.3 Milyon	3.0 Milyon

Symantec firmasının istatistiklerine göre zararlı yazılımların yıllara göre sınıflandırılması Şekil 3'te gösterilmiştir [4]. Uygulamaların kullanıcıdan habersiz olarak gerçekleştirdiği eylemlerin yıllara göre oranları değişmesine rağmen daha çok zarar verme amaçlı genel tehditler, bilgi sızdırma ve kullanıcıyı izleme şeklinde yoğunlaştığı görülmektedir. Bunun yanında içerik gönderme ya da reklam gösterme gibi kullanıcıyı tehdit etmeyen ancak rahatsızlık verebilecek eylemler de gerçekleştirilebilmektedir.

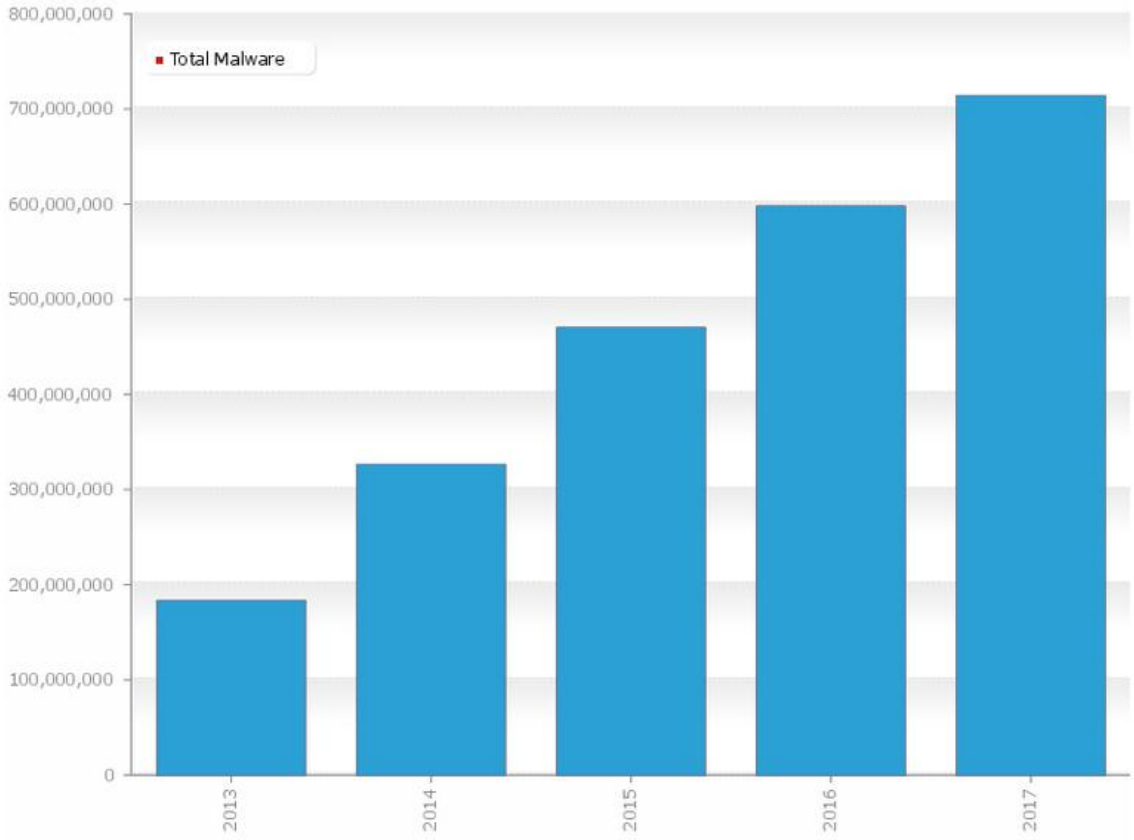
Şekil 3: Zararlı Yazılımlarda Bulunan Özelliklerin Yıllara Göre Dağılımı [4]

Güncel verilere baktığımızda ise 2017 yılında ortaya çıkan zararlı yazılım sayısı 115.000.000' un üzerindedir. Son 5 yılda her yıla ait ortaya çıkan zararlı yazılım sayısı Şekil 4' te gösterilmektedir.



Şekil 4: Son 5 Yılda Ortaya Çıkan Zararlı Yazılım Sayılarının Yıla Göre Grafiği [5]

Yine aynı firmanın istatistiklerine göre 2017 yılında AV-Test firmasının istatistiklerine göre toplam zararlı yazılım sayısı 700.000.000' un üzerindedir. Son 5 yılın toplam zararlı yazılım sayısı Şekil 5' te verilmektedir.



Şekil 5: Son 5 Yılın Toplam Zararlı Yazılım Sayısı [5]

A.2.1. Amaç

Yapılan araştırmalarda gösterildiği üzere her geçen gün mobil uygulamaların artması ve kullanımının yaygınlaşması ile birlikte, tehditler ve oluşturdukları riskler de artmaktadır. Bu tehditler doğrultusunda, bu çalışmada Mobil Uygulamalarda Güvenlik Analiz Sistemi projesinin geliştirilmesi planlanmaktadır. Bu proje ile birlikte test edilen mobil uygulamanın güvenlik açığı raporunun kullanıcıya sunulması hedeflenmektedir.

A.2.2. Konu ve Kapsam

Zararlı uygulamalar özellikle pazarda en çok paya sahip olması nedeni ile saldırılar için daha cazip hale gelen Android telefonları ve kullanıcılarını hedef almaktadır. Bu nedenle, bu projede android cihazlar için geliştirilen uygulamaların, son kullanıcı tarafından kullanılmaya başlamadan önce uygulamanın güvenirliliğinin ve oluşturabileceği risklerin test edebileceği bir ortam oluşturulması hedeflenmiştir. Mobil uygulamalarda güvenlik analizlerinde kullanılan iki tip yöntem vardır: statik analiz ve dinamik analiz. Statik analiz, uygulamaların tersine mühendislik yöntemleriyle kaynak kodlarına erişilip güvenliği tehdit edebilecek fonksiyonların kullanımının saptanmasıdır. Bu tip analiz yöntemi, kod karıştırma tekniklerine karşı zayıftır. Dinamik analiz methodlarında ise çalışan uygulamaların sistem çağrılarının, ağ trafiğinin, bellek kullanımının ve işlemci kullanımının analiz edilmesidir. Bu nedenle kod karıştırma gibi tekniklere karşı işlevini korur.

Bu proje güvenlik alanında ülkemizdeki çalışmalara katkı sağlayacak, otomatize edilmiş mobil güvenlik analiz sistemleri konusundaki açığı kapatacak, herkesin kolaylıkla kullanabileceği açık kaynak kodlu bir sistem olacaktır ve kullanıcıların bilgilerinin güvenliğini tehdit eden unsurları açığa çıkaracaktır.

Projeden herhangi bir kişi, kurum ve kuruluşlar faydalanabilecektir. Sistem için basit bir kullanıcı ara yüzü tasarlanması planlanmaktadır, böylelikle her yaş grubundan kullanıcı sistemden faydalanabilecektir. Proje, açık kaynak kodlu olarak internet ortamına sunulacaktır ve kar amacı güdülmemektedir. Ülkemizde güvenlik alanında yapılan çalışmalara katkı sağlaması amaçlanmaktadır.

A.2.3. Literatür Özeti

Mobil uygulamalarda bilgi güvenliği alanında:

- Abraham, Schlecht, Dobrushin'in 'MobSF' ile otomatize edilmiş güvenlik analizi [7],
- Malik ve Khatter'in 'Strace' kullanarak sistem çağrılarını yakalama [9],
- Zheng, Sun ve Liu'nun 'Ptrace' kullanarak .so, .elf gibi dinamik yüklemeleri yakalama [10],
- Alzaylaee, Yerima ve Sezer'in mobil uygulamalarda otomatize edilmiş dinamik analiz [11],
- Reddy, Rajesh, Pareek ve Patil'in 'Dynaldroid' ile otomatize edilmiş dinamik analiz [12],
- Tchakounté ve Dayang' ın mobil uygulamalarda güvenlik araçlarının incelenmesi [13],
- Kabakuş, Doğru ve Çetin'in Android kötücül yazılım tespit ve koruma sistemleri incelenmesi [14],
- Kayabaşı ve Doğru'nun Mobil Cihazlarda Zararlı Yazılım Tespitinde Kullanılan Statik Analiz Araçları [15]

gibi bir çok çalışma yapılmıştır. Bu çalışmalardan [9] ve [10], dinamik analiz metoduyla analiz işlemi yapmaktadır fakat otomatize değildir. Abraham, Schlecht, Dobrushin ve Alzaylaee, Yerima ve Sezer'in çalışmalarında ise otomatize dinamik analiz yapılmaktadır fakat raporlamaları son kullanıcıya uygun değildir [7, 11].

Abraham, Schlecht, Dobrushin'in Mobile Security Framework ile analiz işleminde statik ve dinamik analiz yapılmaktadır [7]. Otomatize dinamik analiz yapılsa da raporlamaları son kullanıcıya uygun değildir.

Malik ve Khatter, monkeyrunner aracı başarılı bir şekilde çalıştırılmaktadır ve kötü amaçlı ve normal uygulama tarafından başlatılan sistem çağrıları 'Strace' ile kaydedilmektedir. Uygulamalar sürecinde 'Strace' komutunu çalıştırdıktan sonra, dikkate alınan uygulama tarafından çağrılan sistem çağrılarının log dosyasını alıp incelenmektedir [9]. Strace komutu sonucunda çıkan rapor da son kullanıcıya uygun değildir.

Zheng, Sun ve Liu' nun çalışmalarına göre bazı bilgiler veya kötü amaçlı davranışlar yalnızca çalışma zamanında elde edilebildiğinden DroidTrace, uygulamayı her ileri yürütme yolu üzerinden otomatik olarak çalıştırmaktadır ve dinamik yükleri analiz etmek için ptrace tabanlı dinamik analiz aracını kullanarak .so, .file gibi dosyaları yakalamaktadır [10]. 'Ptrace' de 'Strace' benzeri rapor üretmektedir ve bu da son kullanıcıya uygun değildir. Ayrıca 'Strace' ve 'Ptrace' araçları yönetici izni gerektirmektedir. Yönetici izni de son kullanıcılar için tehlikeli olabilir.

Alzaylaee, Yerima ve Sezer' in mobil uygulamalarda otomatize edilmiş dinamik analiz çalışmalarına göre; DynaLog, mevcut açık kaynak araçlarını kullanarak inşa edilmiş ve otomatikleştirilmiş bir analiz platformuna duyulan ihtiyaca göre düzenlenmiştir, çünkü çoğu mevcut çerçeveler ya kapalı kaynaktır ya da yalnızca aralıklı olarak analiz için online sunulmuştur [11]. DynaLog, DroidBox' a dayanan emulator tabanlı bir analiz sistemi içerir ve API çağrı imzalarının, bazı API sınıfları ve yöntemleri aracılığıyla etkinleştirilen çeşitli potansiyel olarak kötü amaçlı

davranışların loglarını tutmak için uygulamalara gömülmesini sağlayan bir araçlar modülü uygular. Çıktıları son kullanıcıya uygun değildir.

Dynaldroid, Android uygulamaların otomatik dinamik analizi için bir araçtır. Reddy, Rajesh, Pareek ve Patil'in çalışmalarının ana katkıları, tamamen otomatikleştirilmiş bir test durum nesilleri, sistematik olay tetikleme ve kötü amaçlı eylemleri tespit etmek için Android uygulamasının dinamik analizidir [12]. Kötü amaçlı yazılım örneklerinin analizine dayanarak, uygulamanın gerçekleştirdiği kötü amaçlı eylemlerin belirlenmesine yardımcı olan bir dizi desen çıkarılmıştır. Dynaldroid analiz sonuçları DroidBox analiz sonuçlarına oldukça benzemektedir fakat son kullanıcıya uygun olması için bu raporların anlamlandırılması gerekmektedir.

Yurt dışında yapılan otomatize dinamik analiz çalışmaları konusunda ülkemizde eksiklik yaşanmaktadır. [9], [10], [12] ve [15]. kaynaklarda olduğu gibi birçok güvenlik analiz aracı mobil cihaz üzerinde çalışmamaktadır. Bunun sebebi, çalışan uygulamaların sistem çağrıları gibi kritik işlemlerine ulaşmak için yönetici izni gerekiyor olmasıdır. Yapmayı hedeflediğimiz sistem yönetici iznine gerek duymadan uygulamaları mobil cihaz üzerinden analiz edebilecektir. Ayrıca tasarlanması düşünülen basit kullanıcı ara yüzü ve rapor sistemi sayesinde piyasadaki diğer sistemler ve uygulamalar arasında yer alabilecektir.

A.3. BEKLENEN FAYDA

A.3.1. Özgün Değer

Proje, diğer güvenlik uygulamalarından farklı olarak mobil cihaz üzerinden çalışacak, kullanıcıya basit bir ara yüz sağlayacak ve anlayabilecekleri şekilde bir rapor üretecektir. Ayrıca kullanıcılar Google Play Store üzerinden de uygulama seçip analiz edebileceklerdir.

Literatür incelemesi yapıldığında, önceki benzer çalışmalarda mobil cihaz üzerinden dinamik analiz yapılamamaktadır, sadece yönetici izni olan sanal makinelerde analiz işlemlerini yapılmaktadır [7, 11, 12]. Bu şekilde her kullanıcının telefonunun yönetici iznine sahip olması kullanıcıların bilgilerini tehlikeye atar. Zararlı yazılımlar, bu izinleri ele geçirdikleri takdirde her türlü dinleme, izin değiştirme, bozma işlemlerini yapabilirler. Yine benzer uygulamalarda kullanıcıların uygulamalarının kurulum dosyalarını (.apk) sisteme yüklemeleri gerekmektedir ve bu yapı her kullanıcıya uygun değildir, çünkü son kullanıcılar kurulum dosyalarını bulacak teknik bilgiye sahip olmayabilirler. Bu nedenle yapılması planlanan sistemde kullanıcılar 'Google Play Store' üzerinden seçtikleri uygulamayı analiz edebileceklerdir.

A.3.2. Yaygın Etki / Katma Değer

Yurt dışında yapılan otomatize dinamik analiz çalışmaları konusunda ülkemizde eksiklik yaşanmaktadır. [9], [10], [12] ve [15]. kaynaklarda olduğu gibi birçok güvenlik analiz aracı mobil cihaz üzerinde çalışmamaktadır. Bunun sebebi, çalışan uygulamaların sistem çağrıları gibi kritik işlemlerine ulaşmak için yönetici izni gerekiyor olmasıdır. Yapmayı hedeflediğimiz sistem yönetici iznine gerek duymadan uygulamaları mobil cihaz üzerinden analiz edebilecektir. Ayrıca tasarlanması düşünülen basit kullanıcı ara yüzü ve rapor sistemi sayesinde piyasadaki diğer sistemler ve uygulamalar arasında yer alabilecektir.

Yapılması planlanan sistemde kullanıcıların mobil cihazlarında yönetici iznine gerek duymadan, basit bir ara yüz ile Google uygulama marketinden seçebilecekleri uygulamaların analizi yapılabilecektir ve kurulum dosyası da yüklenebilecektir. Kullanıcılara basit bir ara yüz ile raporlama sunacaktır ve her tip kullanıcıya uygun olarak tasarlanacaktır. Bu raporda, analiz edilen uygulamanın zararlı olup olmadığı, zararlı ise ne kadar zararlı olduğu sınıflandırılarak kullanıcılara sunulacaktır. Zararlı yazılımlar özellikle kişilerin kimlik, resim veya kredi kartı gibi bilgilerini hedef alarak bilgi güvenliğinin en temel unsurlarından olan gizlilik ilkesini ihlal ederler. Bu anlamda, kişisel bilgi güvenliğini tehdit eden zararlı uygulamalar tespit edilerek, ulusal güvenliğimiz konusunda toplumsal ve ekonomik katkı sağlanması amaçlanmıştır.

A.4. YÖNTEM

Projede kapsamında geliştirilecek sistem, son kullanıcılara hitap edecektir ve dinamik analiz metodu tercih edilmiştir. Çalışan uygulamaların sistem çağrıları, uygulamanın istediği izinler ve ağ trafiği ancak dinamik analiz metodu ile ortaya çıkarılabilir.

Android işletim sistemi SELinux (Güvenliği Arttırılmış Linux) tabanlı bir işletim sistemi olduğu için SELinux ile aynı kullanıcı yapısına sahiptir [6]. Bu nedenle sistem çağrılarına ve tüm uygulamaların izinlerine erişmek için yönetici izni gerekmektedir. Mobil cihazlara yönetici izni vermek kullanıcılar için tehlikedir, çünkü sistem dosyalarını silebilir ve sisteme tam erişimi olduğu için herhangi bir zararlı yazılım, sistemin tam kontrolünü sağlayabilir.

Android işletim sistemi mimarisinde güvenliği sağlamak amacıyla uygulama izinleri yapısı kullanılmaktadır [7]. Her uygulama yüklendiği sırada kullanıcıdan kullanmak istediği izinleri ister. Android uygulamalarda geliştirme sırasında uygulamalara yeni izinler oluşturulabilir. Android işletim sistemindeki bu yapı bir güvenlik açığı olarak da değerlendirilebilir. Bu yapı sayesinde zararlı yazılımlar, erişilmesi istenilmeyen fonksiyonlara erişebilir. Bu da kullanıcılar için büyük tehlike oluşturmaktadır.

Android işletim sisteminin bu yapısından dolayı bir uygulama, diğer uygulamaların sistem çağrılarına erişemezler. İzinlerine erişmek için ise yine diğer uygulamalara erişim izni verilmesi gerekmektedir. Bu da kullanıcıların güvenlik sistemine güvenini azaltır. Önerilen projede bu probleme çözüm sağlayarak uygulamaların farklı bir platformda incelenerek, analiz sonucunun kullanıcıya rapor olarak iletilmesi düşünülmüştür.

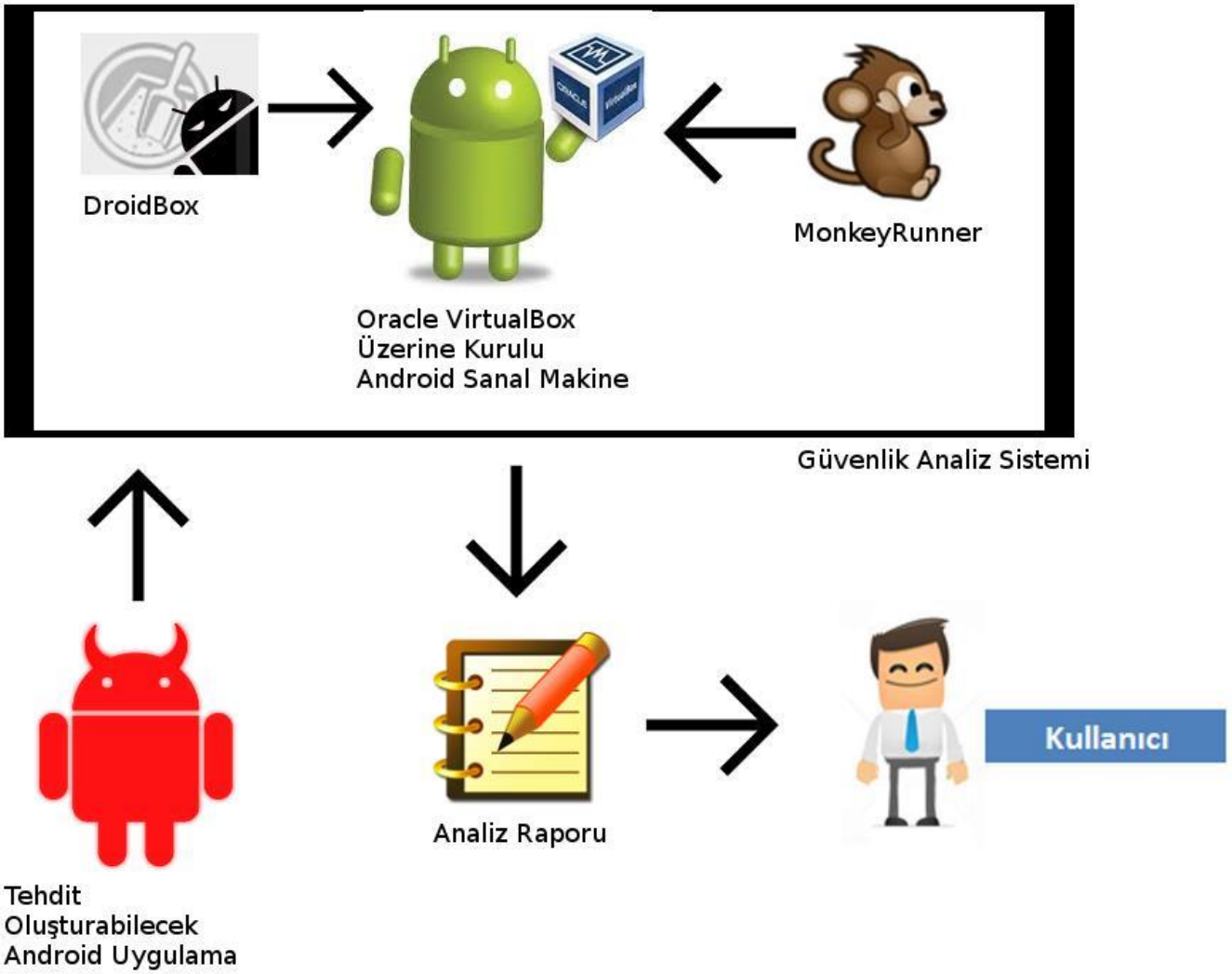
Oluşturulacak sistem, bir sunucu üzerinde çalıştırılacaktır. Sunucu üzerinde bulunan yönetici izinlerine sahip sanal makineye analiz edilmek istenilen uygulama yüklenecektir ve kullanılacaktır. Bu sayede uygulamanın sistem çağrıları yakalanacaktır. Bu aşamada 'Oracle VM VirtualBox' sanal makinesi üzerine kurulacak olan Android işletim sistemi kullanılacaktır. Ürettiği analiz raporları tasarlanacak mobil uygulamaya gönderilecektir. Kullanılacak bu yöntem sonucunda kullanıcıların mobil cihazlarında yönetici izni olmadan güvenlik analizi yapılabilecektir.

Sistem çağrılarını yakalamak için Droidbox aracı kullanılacaktır. Droidbox, android uygulamaların dinamik analizini yapmak amacıyla geliştirilmiş bir kum havuzu aracıdır. Gelen/giden ağ trafiği, dosya okuma/yazma işlemleri, giden sms ve arama takibi ve şifreleme işlemleri gibi kritik işlemleri kayıt altına alarak uygulamaların arka planda yaptıklarını elde etmeyi sağlar [7]. Yakaladığı işlemleri de Android geliştirici aracı olan logCat'e yazdırabilir. Elde edilecek bu bilgiler de basit bir ara yüz ile kullanıcıların anlayabileceği şekilde kullanıcıya sunulacaktır.

Oluşturulması planlanan sistem, kullanıcıların 'Google Play' uygulama marketinden yüklemeyi istedikleri uygulamayı veya 3.parti sitelerden indirdikleri uygulamaları analiz edebilecektir.

Dinamik analiz metodunda güvenlik analizi yapılacak uygulamanın sanal makine de kullanılması gerekmektedir ve bu işlem de kullanıcıların isteyecekleri bir durum değildir. Bu amaçla bu süreç otomatize edilecektir ve bu süreçte MonkeyRunner kullanılacaktır. MonkeyRunner, Android uygulamaları ve güvenlik analizi sistemlerinde en çok kullanılan test platformudur. MonkeyRunner sayesinde analiz edilecek uygulamaların aktiviteleri çalıştırılacak, butonlarına tıklanacak ve yaptığı sistem çağrıları yakalanacaktır.

Projenin çalışma yapısı Şekil 6' da gösterilmiştir. Kullanıcılar uygulamayı seçtiklerinde arka planda uygulama sanal makineye yüklenecek ve güvenlik analizi yapılacaktır. Bunun sonucunda çıkarılan analiz raporu, son kullanıcıya uygun bir şekilde dönüştürülecek ve kullanıcılara sunulacaktır. Bu aşamada mobil uygulama geliştirilecek ve bu uygulama üzerinden tüm işlemler gerçekleştirilebilecektir.



Şekil 6: Mobil Uygulamalarda Güvenlik Analiz Sistemi Çalışma Yapısı

A.5. ARAŞTIRMA OLANAKLARI

Yapılması planlanan sistemde kullanılacak teçhizat listesi, Tablo 2' de verilmiştir.

Tablo 2: Projede Kullanılacak Teçhizat Listesi

Projede Kullanılacak Mevcut Makine – Teçhizat Listesi (*)	
Adı/Modeli	**Projede Kullanım Amacı**
Sunucu	Sistem kurulacak ve servisi sağlayacak sunucu
Android İşletim Sistemli Akıllı Telefon	Uygulama yükleme, Google Play Store üzerinden uygulama arama ve uygulama analizi yapma gibi işlemleri yapacak mobil telefon

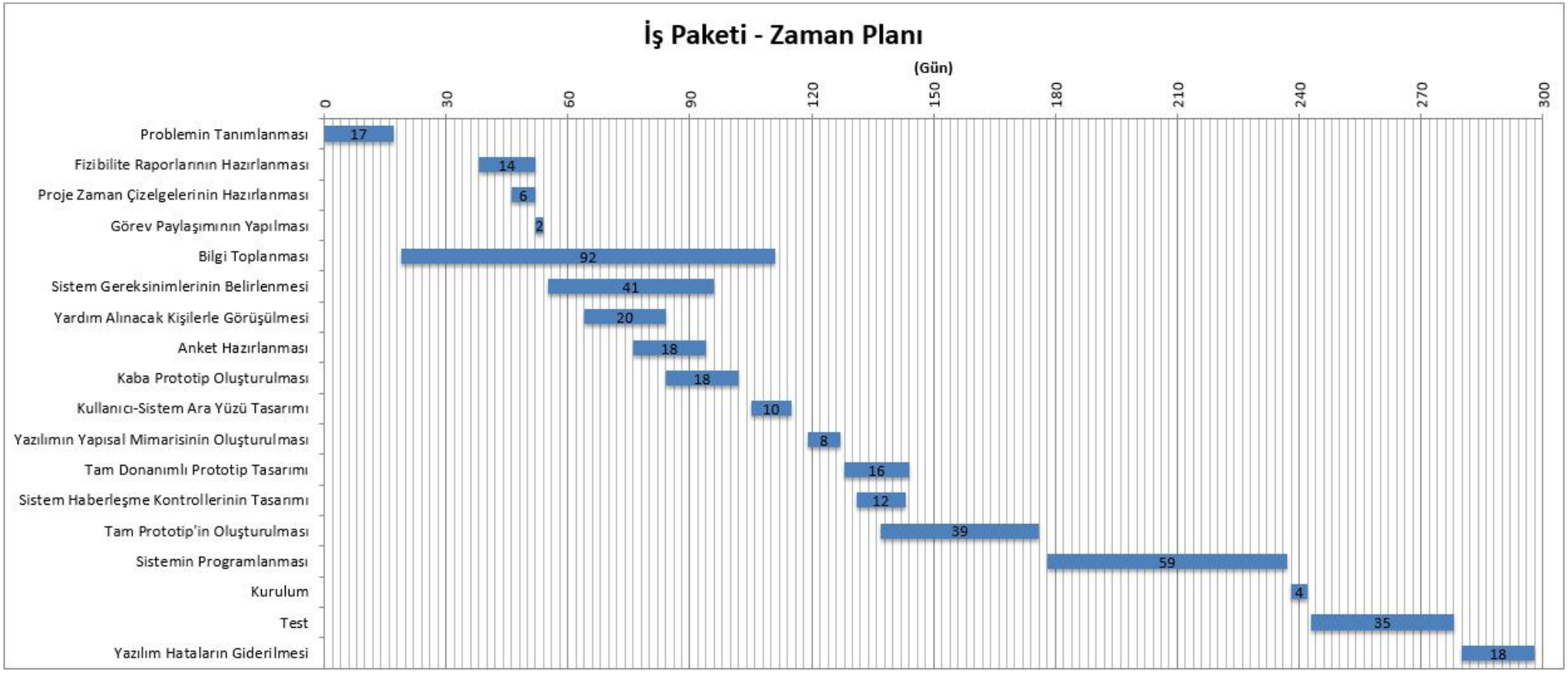
(*)Bu bölümde sadece, bölümümüzde veya tez yazarının da var olup projede kullanılacak olan makine-teçhizat belirtilmelidir

A.6. ÇALIŞMA TAKVİMİ:

A.6.1. İş Zaman Çizelgesi:

İş - zaman çizelgesinde planlama, analiz ve tasarım süreçlerini kapsayan süreçlerin zamana bağlı çalışma düzenleri belirtilmektedir. Her aşama için uygun olan çalışma süresi grup çalışanları ve araştırma olanaklari gözetilerek hesaplanmıştır.

Tablo 3: İş Paketi - Zaman Planı



A.6.2. Kişi - İş Açıklaması (Projede kim hangi işte görev alacak):

Projenin araştırılma safhasından tamamlanmasına kadar hesaplanan süre içerisinde yapılması gereken işler grup üyelerine atanmıştır. Grup üyelerinin proje geliştirme ve araştırma aşamalarında eşit rol alması ve gerçekleştirilecek projenin bir parçası olmasını sağlamak amacıyla görev atamaları aşağıdaki gibi yapılmıştır.

Tablo 4: Zaman ve Sorumluluklarla İlgili Planlama Tablosu

Görev	Kişiler
Problemin Tanımlanması	OT, MŞ
Fizibilite Raporlarının Hazırlanması	OT, SB
Proje Zaman Çizelgelerinin Hazırlanması	SB, MŞ
Görev Paylaşımının Yapılması	MŞ, SÇ
Bilgi Toplanması	OT, SB, MŞ, SÇ
Sistem Gereksinimlerinin Belirlenmesi	OT, SB, MŞ, SÇ
Yardım Alınacak Kişilerle Görüşülmesi	SB, SÇ
Anket Hazırlanması	MŞ, SÇ
Kaba Prototip Oluşturulması	OT
Kullanıcı-Sistem Ara Yüzü Tasarımı	OT, MŞ
Yazılımın Yapısal Mimarisinin Oluşturulması	OT, MŞ, SÇ
Tam Donanımlı Prototip Tasarımı	OT, SB, SÇ
Sistem Haberleşme Kontrollerinin Tasarımı	MŞ, SB
Tam Prototip'in Oluşturulması	OT, SB
Sistemin Programlanması	OT, MŞ, SB, SÇ
Kurulum	MŞ, SÇ
Test	SB, SÇ
Yazılım Hataların Giderilmesi	SB, MŞ

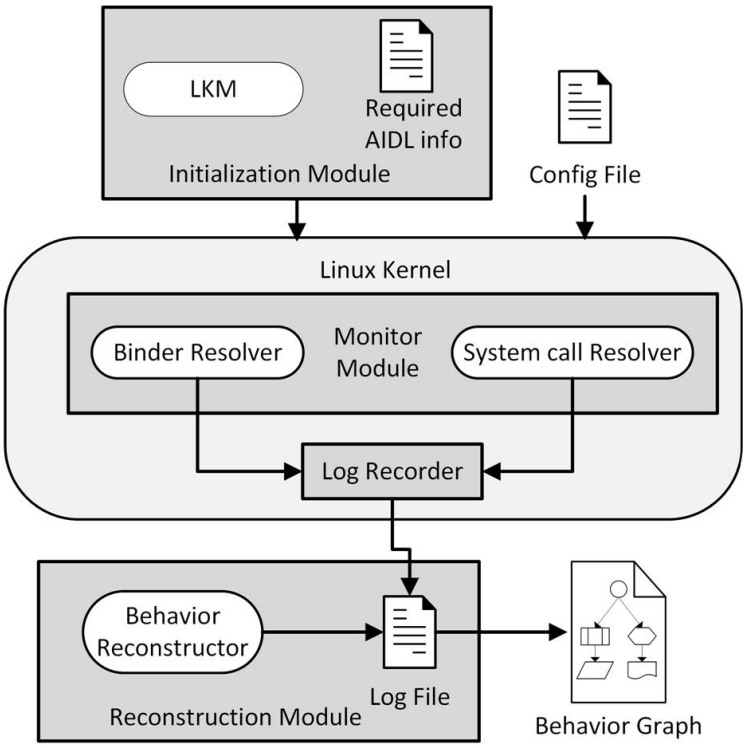
Mehmet Okan TAŞTAN : OT
Ahmet Sefa BULCA : SB
Mustafa ŞAHİN : MŞ
Sevda ÇİMEN : SÇ

B. ANALİZ
B.1. SİSTEM GEREKSİNİMLERİNİ ORTAYA ÇIKARMA YÖNTEM VE TEKNİKLERİ

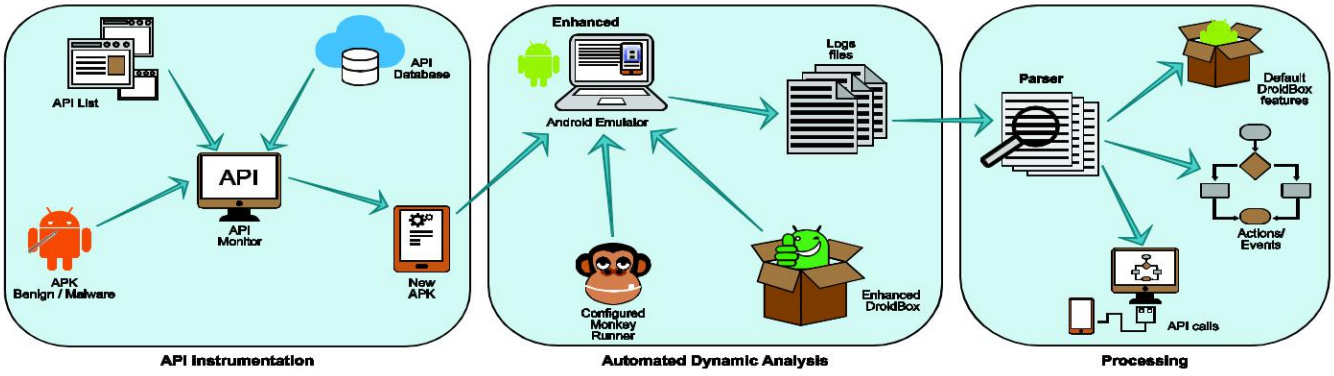
Sistem gereksinimleri; yazılı basılı belge inceleme, yüz yüze görüşme, gözlem, prototip ve hızlı uygulama tasarımı, veri akış şemaları ve iş akış şeması oluşturularak belirlenmiştir.

B.1.1. Yazılı Basılı Belge İnceleme

TaintDroid, VetDroid, DroidKungFu, Aurasium ve CopperDroid araçları kernel seviyesinde sistem çağrılarını toplarken, otomatize bir uygulama analizi yapmamaktadır. DroidRevealer, DroidBox, DroidScope, DynaLog ve MobSF gibi araçlar ise otomatize uygulama analizi yapmaktadır fakat raporları ve kullanımları son kullanıcıya uygun değildir. DroidRevealer mimarisi Şekil 7' de gösterilmektedir. MobSF (Mobile Security Framework) üzerinde statik ve dinamic analiz yapılabilmektedir. Tersine mühendislik araçları da barındırmaktadır fakat son kullanıcı uygulamaların yükleme dosyalarını (.apk) temin edecek seviyede olmayabilirler. Bu yüzden son kullanıcı için uygun değildir. Yapılması planlanan sistem, DynaLog ile benzerlik göstermektedir fakat son kullanıcıya daha uygun olacaktır ve uygulamalar Google Play Store üzerinden seçilerek yüklenebilecektir. DynaLog mimarisinde olduğu gibi analiz sistemine uygulama gönderilip, analiz sonuçları kullanıcıya sunulacaktır. Analiz işlemi için DroidBox kullanılacaktır. DynaLog mimarisi, Şekil 8' de gösterilmektedir.



Şekil 7: DroidRevealer Mimarisi [16]



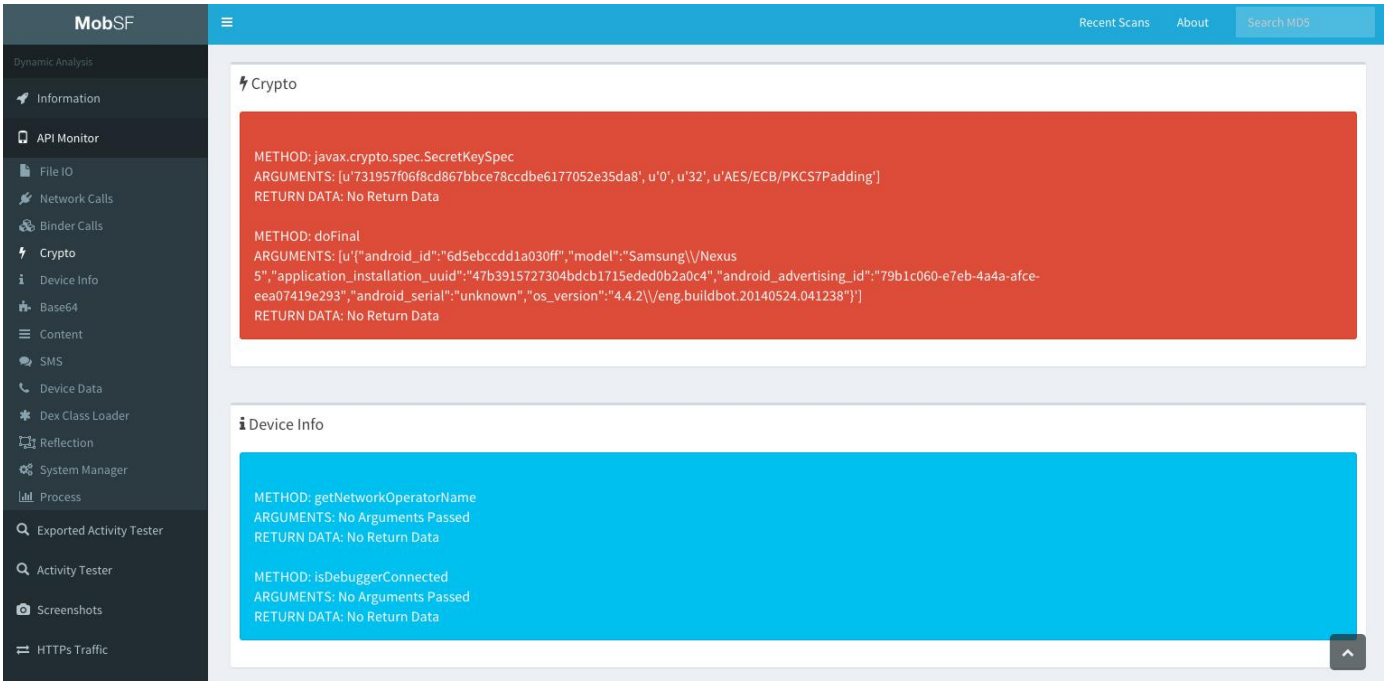
Şekil 8: DynaLog Mimarisi [11]

B.1.2. Yüz Yüze Görüşme

46 Software Design (Ankara) şirketinin kurucularından ve eski ortaklarından Sadık HANECİOĞLU ile görüşmeler sonucunda Google Play Store' dan uygulamaları alırken yaşanabilecek sorunlar hakkında ve çözümleri hakkında fikir alınmıştır. HTML parser araçlar ve web tarayıcı otomatizasyon işlemleri için bilgi alınmıştır.

B.1.3. Gözlem

MobSF ile birlikte DEXtoJar ve JD-Gui statik analiz araçları da gözlemlenmiştir. MobSF sisteminin dinamik analiz yapısı, araçları ve raporları incelenmiştir. Son kullanıcıya uygun hale nasıl getirilebileceği konusu incelenmiştir. Şekil 9' da da görüldüğü gibi oldukça gelişmiş bir analiz aracı olan MobSF' in analiz raporu son kullanıcıya uygun değildir.

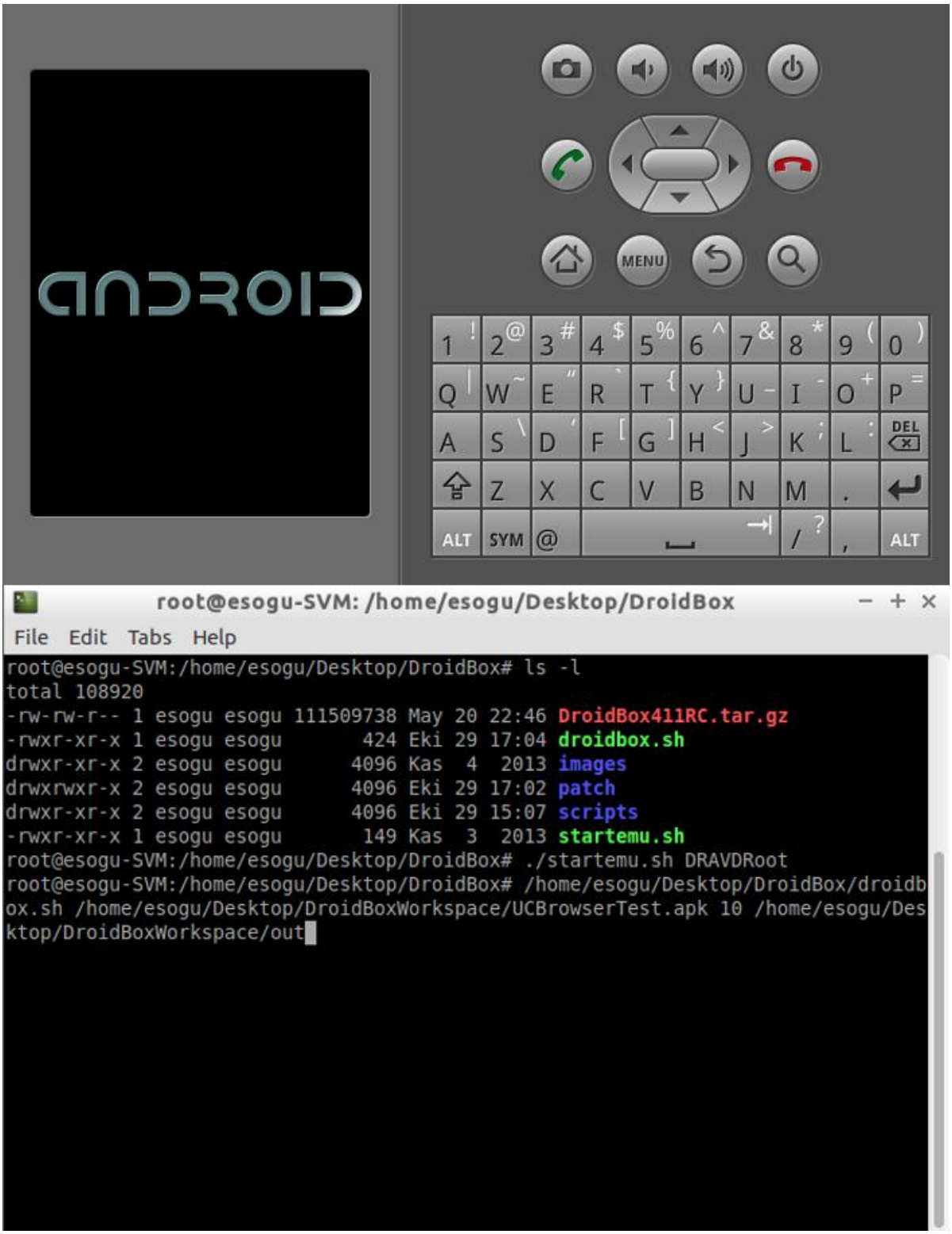


Şekil 9: MobSF Analiz Raporu [18]

B.1.4. Prototip ve Hızlı Uygulama Tasarım

Ubuntu 14.04 üzerine DroidBox aracı kurularak bazı uygulamalar analiz edilmiştir ve analiz raporu incelenmiştir. Aynı zamanda GPlayCli aracı ile Google Play Store' dan uygulamalar aratılmış ve indirilmiştir. Analiz işlemleri için DroidBox düzenlenmiştir. DroidBox' ın çalışması için uygulamayı ve analiz edilecek süreyi vermek yeterlidir fakat sistem çıktılarını istenilen klasöre almak ve DroidBox analiz sonucunu JSON formatında görüntülemek için DroidBox geliştirilmiştir.

İlk olarak DRAVDRoot ismiyle oluşturulan sanal makine başlatılmıştır ve UCBrowser uygulaması, bu makineye yüklenerek 10 saniye analiz edilmek üzere DroidBox ile çalıştırılmıştır. Bu işlemler Şekil 10' da gösterilmektedir. Bu işlemler daha sonra otomatize bir şekilde yapılacaktır.



Şekil 10: DroidBox Aracı Test Edilmesi

Şekil 11’ de gösterildiği gibi Droidbox çalışırken sistem çağrılarını toplamaktadır.

Şekil 11: DroidBox Aracı Test Edilmesi - 2

Şekil 12’ de görüldüğü gibi DroidBox analiz sonucu, JSON veri formatında görüntülenmektedir.


```
"com.uc.browser.bgprocess.bussiness.lockscreen.backgroundbussiness.dataprovider.
action.PROVIDE_DATA", "com.uc.browser.bgprocess.bussiness.zombieuser.ZombieUserS
tatsBroadcastReceiver": "com.uc.action.bgprocess.user_stats_alarm", "com.uc.brow
ser.bgprocess.BackgroundProcessBroadcastReveiver": "android.intent.action.ACTION
_SHUTDOWN", "com.ucweb.message.UcwebRegistrationReceiver": "android.intent.actio
n.PACKAGE_REMOVED", "com.uc.browser.bgprocess.bussinessmanager.lockscreen.datapr
ovider.MusicPushDataProvider": "com.uc.browser.bgprocess.bussiness.lockscreen.ba
ckgroundbussiness.dataprovider.action.PROVIDE_DATA", "com.uc.base.system.receive
rs.SysReceiver": "android.intent.action.PACKAGE_REPLACED", "com.uc.base.push.gcm
.GCMPushBroadcastReceiver": "com.google.android.c2dm.intent.REGISTRATION", "com.
uc.browser.googleanalytics.GoogleAnalyticsReceiver": "com.android.vending.INSTAL
L_REFERRER", "com.uc.base.push.PushBusinessBroadcastReceiver": "com.UCMobile.pus
h.DELETE", "com.uc.application.desktopwidget.UCWidgetProvider": "com.UCMobile.ap
pwidget", "com.ucmusic.notindex.RemoteControlEventReceiver": "android.intent.act
ion.MEDIA_BUTTON", "com.uc.application.facebook.push.FacebookPushBroadcastReceiv
er": "com.uc.action.push.gcm.dispatch", "com.ucmusic.notindex.NewAddCheckReceive
rShell": "android.intent.action.BOOT_COMPLETED", "com.uc.browser.business.openwi
fi.NetworkConnectReceiver": "android.net.conn.CONNECTIVITY_CHANGE", "com.uc.base
.push.gcm.GCMLocalCommandReceiver": "android.intent.action.BOOT_COMPLETED"}, "de
xclass": {"0.7062399387359619": {"path": "/data/app/com.UCMobile.intl-1.apk", "t
ype": "dexload"}}, "hashes": ["73ed8f8b04e5943722f76d1a8c828917", "8b387960f9656
8c2df967b3e62495a53ed0a4b9c", "2e514e0e8de065a8e2afd8f1c276b7f261f0530a096d82e32
8fb22a8a8d39267"], "closenet": {}, "phonecalls": {}}
```


Şekil 12: DroidBox Analiz Sonucu

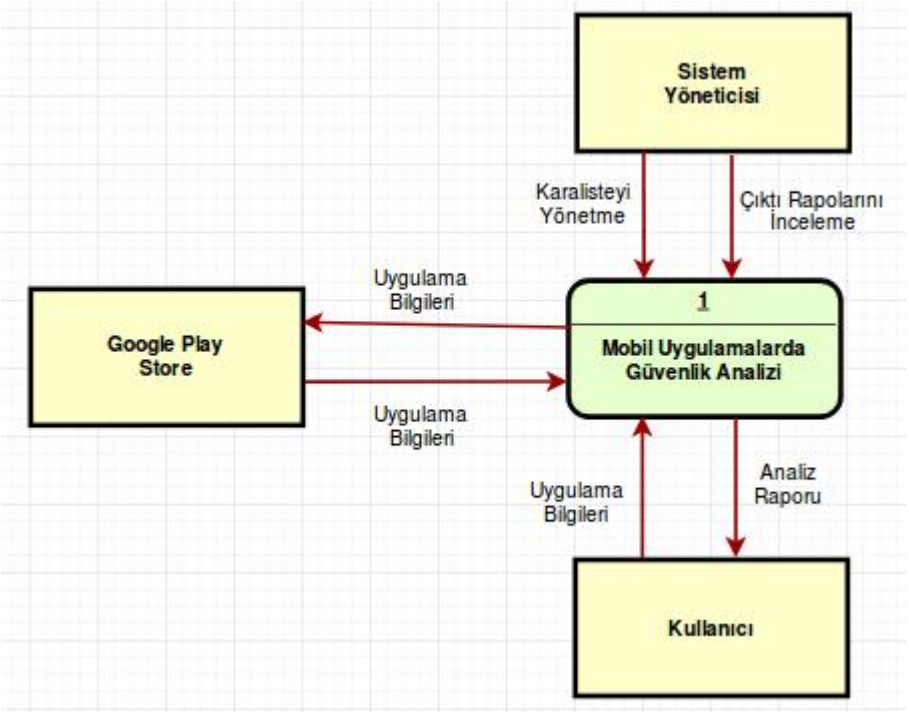
GPlayCli aracı ile ‘BlockBreaker’ kelimesi aratılmış ve Block Breaker Puzzle uygulaması başarılı bir şekilde sunucuya indirilmiştir. İlgili işlemler Şekil 13’ te gösterilmektedir.

```
motastan950MehmetPC ~ $rm -f ~/.cache/gplaycli/token
motastan950MehmetPC ~ $ gplaycli -s 'BlockBreaker'
Title                          Creator          Size     Downloads             Last Update   AppID                                          Version   Rating
BlockBreaker Syndra            Bullinzadu       34.84MB  100,000+ downloads    Sep 17, 2017  com.Bullinzadu.Syndra                          21        4.74
Brick Breaker Star: Space King Springcomes      17.30MB  5,000,000+ downloads  Aug 25, 2017  com.lt.latte.brick                             31        4.51
Break Bricks                   CanadaDroid      7.69MB   10,000,000+ downloads Aug 29, 2014  com.tongwei.blockbreaker                       10917     3.98
Bricks Breaker Puzzle          mobirix          24.52MB  1,000,000+ downloads  Nov 1, 2017   com.mobirix.swipebrick                         17110101  4.47
Balls Bounce                   MagiPlay Games   18.59MB  1,000,000+ downloads  Nov 2, 2017   ballz.brick.breaker.ballz.bounce.free          22        4.40
One More Brick                 Rifter Games     6.26MB   1,000,000+ downloads  Oct 20, 2017  com.riftergames.onemorebrick                   13300     4.64
Brick Blast Ball               Dong Studio      26.78MB  1,000,000+ downloads  Nov 3, 2017   com.dongstudio.brick                           47        4.66
BBTAN by 111%                  111%             34.55MB  10,000,000+ downloads Aug 7, 2017   com.crater.bbtan                               31        4.65
Balls Bricks Breaker           Arcade Game Maker 20.82MB 100,000+ downloads    Oct 21, 2017  balls.arcademaker.bricks.breaker.brounce.ballz 16102     4.64
Bricks Ball: Block Breaker game Arcade Games Team 14.74MB 1,000+ downloads     Sep 26, 2017  com.ArcadeTapTap.BallzGlow                     8         4.64
motastan950MehmetPC ~ $ gplaycli -d com.mobirix.swipebrick
Download complete
motastan950MehmetPC ~ $
```

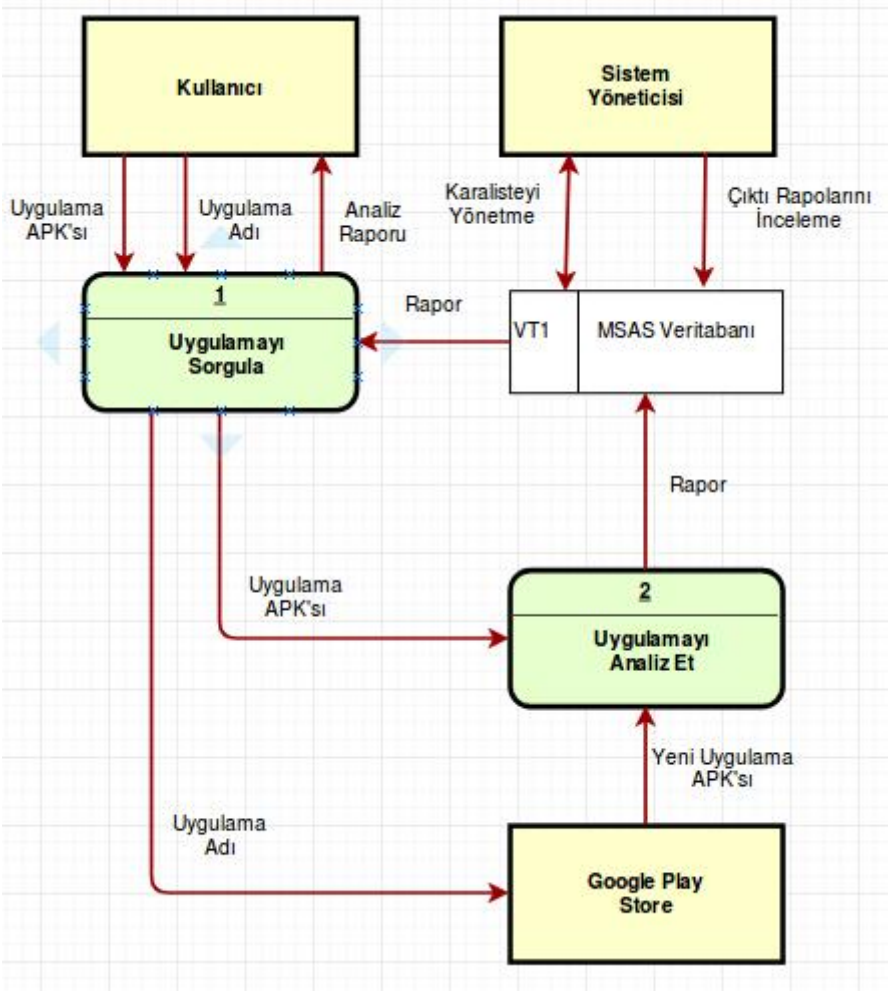
Şekil 13: GPlayCli Aracı İle Google Play Store Üzerinden Uygulama Arama ve İndirme

B.1.5. Veri Akış Şemaları

Son kullanıcı, uygulamayı sisteme yükledikten sonra veya uygulamayı Google Play Store' dan seçtikten sonra uygulama analiz edilip raporu kullanıcıya iletilecektir. Kavramsal ve mantıksal veri akış şemaları şekil 14 ve 15' te görülmektedir.



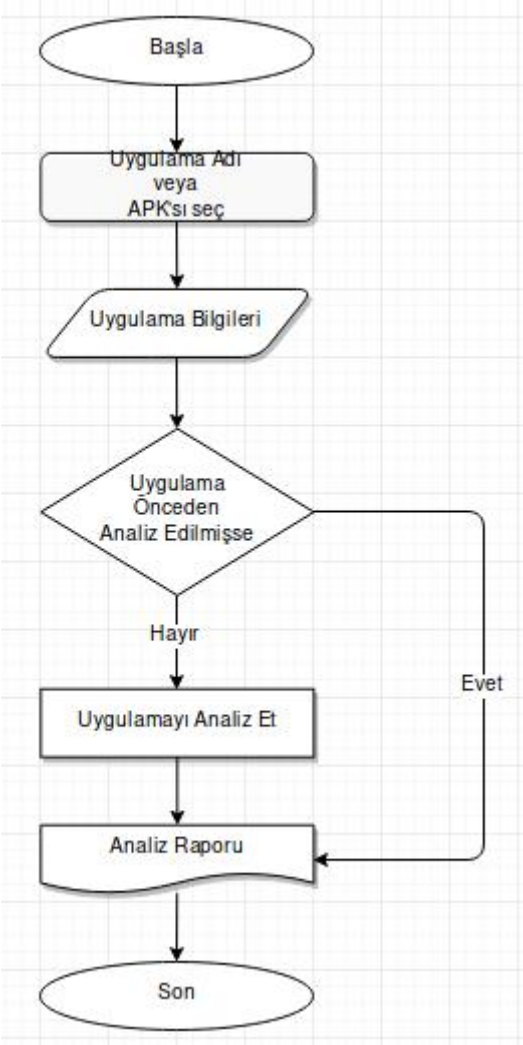
Şekil 14: Kavramsal Veri Akış Şeması



Şekil 15: Mantıksal Veri Akış Şeması

B.1.6. İş Akış Şeması

Sistemin iş akışı, Şekil 16' da gösterilmektedir. Sistem çalışmasından bitişine kadar olan veri akış yönü de gösterilmiştir.



Şekil 16: İş Akış Şeması

B.2. SİSTEM GEREKSİNİMLERİ

B.2.1. İşlevsel Gereksinimler

Sistemin çalışması ve sürekliliği için gereken işlevsel gereksinimler şunlardır;

- Analiz raporları veritabanında tutulmalıdır.
- Test edilmek istenilen uygulama 'Google Play Store' dan aratılabilmelidir.
- APK uzantılı uygulama dosyaları da yüklenip test edilebilmelidir.
- Daha önce test edilmiş uygulamalar ve raporları görüntülenebilmelidir.
- Uygulama 'Google Play Store' üzerinden seçildiyse ve uygulamanın aynı sürümü daha önce analiz edilmişse, kullanıcıya raporu döndürülmelidir. Bu sayede zaman kaybının önüne geçilmiş olacaktır. Bu senaryoda eğer uygulamanın güncel sürümü var ise, o sürüm tekrar analiz edilip kullanıcıya raporu sunulacaktır.

B.2.2. Sistem ve Kullanıcı Ara yüzleri ile ilgili Gereksinimler

Sistem, sunucu üzerindeki android sanal makine üzerinde çalışacaktır ve kullanıcı ara yüzünü oluşturan uygulama, servis aracılığı ile sanal makineyle haberleşecektir.

Sisteme girdiler Google Play Store ve dosya yükleme ile dışarıdan gelecektir. Sistemin çıktıları, sunucu üzerindeki veritabanına saklanacaktır.

Kullanıcılar, uygulama analizini başlattıkları zaman uygulamanın indirilme veya yüklenme süresi ve analiz süresini bekleyeceklerdir. Bu aşamada kullanıcıları bekletmemek için uygulamanın analizi sonuçlandığında kullanıcılara Google Firebase sunucularından bildirim gönderilecek ve bildirimin içerisine gömülü olan rapor numarası ile ilgili rapor görüntülenecektir. Bir diğer alternatif ise analiz başlatıldığında kullanıcıya, gösterilecek rapor numarasının başlatma işlemi sonucunda döndürülmesidir.

Veriler gönderilirken, kaydedilirken ve alınırken JSON veri standartı kullanılacaktır.

B.2.3. Veriyle İlgili Gereksinimler

Analiz sonucunda oluşan veriler, her kullanıcının anlayabileceği şekilde olmadığı için filtrelendikten sonra veritabanına aktarılacaktır ve olabildiğince sade bir şekilde kullanıcıya sunulacaktır.

Sunucudaki veri kapasitesi sınırlı olduğu için ve de bazı uygulamaların boyutu çok büyük olduğu için uygulamaların 'apk' dosyaları analiz sonrasında silinecektir.

Veriler, 5 saatte bir yedeklenecektir ve yönetim panelinden excel dökümanı olarak alınabilecektir.

Veriler NoSQL veya ilişkisel veritabanında tutulabilir. Uygulamaları yayınlayan kişiler ile uygulamaları arasında ilişki oluşturulması planlanmaktadır. İlişkisel veritabanı tercih edilmiştir.

B.2.4. Kullanıcılar ve İnsan Faktörü Gereksinimleri, Güvenlik Gereksinimleri

Sistemi, uygulamayı indiren her kullanıcı kullanabilecektir. Her yaştan kullanıcının uygulamayı kolaylıkla kullanabilmesi hedeflenmektedir. Uygulama, Türkçe ve İngilizce olmak üzere iki dil seçeneği içerecektir.

Sistemin daha iyi analiz edebilmesi için, monkeyrunner aracının olası tüm durumları tetiklemesi hedeflenmektedir. Bu geliştirmeler yeni sürümler çıkarılarak sağlanacaktır.

Kötü niyetli kişiler, uygulamanın çalışmasını önlemek, aksatmak için sisteme saldırı yapabilirler. Bunun önlenmesi için uygulama analiz yapmak isteyen kullanıcıların IP adresleri tutulacaktır. Bir kullanıcının günlük işlem limiti 100 uygulama ile sınırlandırılacaktır. Spam yapan kullanıcıların IP'leri kara listeye alınacaktır ve uygulama analizi yapmaları ve raporları görüntülemeleri engellenecektir.

B.2.5. Teknik ve Kaynak Gereksinimleri, Fiziksel Gereksinimler

Önerilen proje, Kişisel Verilerin Korunması (6698) – Madde 245/A ve Elektronik Haberleşme Sektöründe Şebeke ve Bilgi Güvenliği Yönetmeliği – Madde 12'yi desteklemektedir. Proje, açık kaynak kodlu olarak sunulacaktır ve herhangi bir ticari kar amacı gütmemektedir. Projede kullanılacak Droidbox ve MonkeyRunner araçları da açık kaynak kodlu araçlardır ve ticari amaçla kullanılmaları yasaktır.

Sistem, Linux işletim sistemi üzerinde kurulu olan DroidBox aracı ile çalışacaktır. Kiralanacak sunucuya kurulacak sistem, uygulamaları analiz edecek ve bu uygulamaların analiz raporlarının depolamasında kullanılacaktır.

Teknik gereksinimler aşağıdaki gibidir:

- Ubuntu 14.04
- Android SDK r24.4.1 veya r23.0.2
- Android API 16 r5 ARM image
- DroidBox
- MonkeyRunner
- Python 2 ve 3 + matplotlib, numpy, scipy, tk paketleri, pip, pip3
- GPlayCli

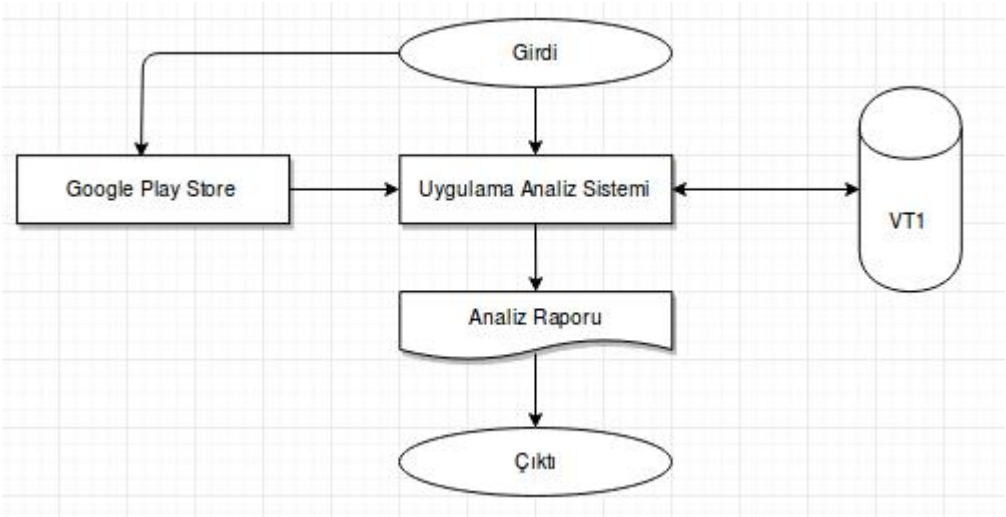
Bu aşamada sunucuya, yukarıdaki uygulama ve paketlerin kurulması gerekmektedir. Bu kurumların bazılarını AndroidTamer, BlackArch ve Santoku gibi mobil penetrasyon testi işletim sistemleri sağlamaktadır. Bunların yerine de 'docker' imajları kullanılabilir. Ali İKİNCİ'nin DroidBox docker imajı da Windows veya Linux işletim sistemleri üzerine kurulup kullanılabilir.

Uygulamaların Google Play üzerinden aratılıp sisteme indirilmesi için evozi veya apkpure gibi siteler kullanılabilir. Bunun için SeleniumHQ ve iMacros gibi tarayıcı otomasyon programları kullanılabilir fakat bu işlemlerin otomatize edilmesi ve 'HTML parser' lar kullanılarak listelerin alınması gerekmektedir. Bu işlemler yerine GPlayCli ile gerçek bir cihazın 'Android device id' si verilerek arama ve indirme işlemleri yapılabilmektedir.

Elektrik ve internet kesintisi sistemi kısıtlayan faktörlerdir fakat günümüzde sunucu hizmeti veren firmalar bu kesintilerin olmayacağını büyük ölçüde garantilerler.

C. TASARIM
C.1. SİSTEM TASARIMI

Sistem tasarımı, Şekil 17' de görüldüğü gibidir. Kullanıcı sisteme girdi olarak uygulama apk' sı veya uygulama adı girdikten sonra seçtiği veya yüklediği uygulama analiz sistemde analiz edildikten sonra veritabanına kayıt edilecektir. Daha sonra analiz raporu çıktı olarak kullanıcıya sunulacaktır.

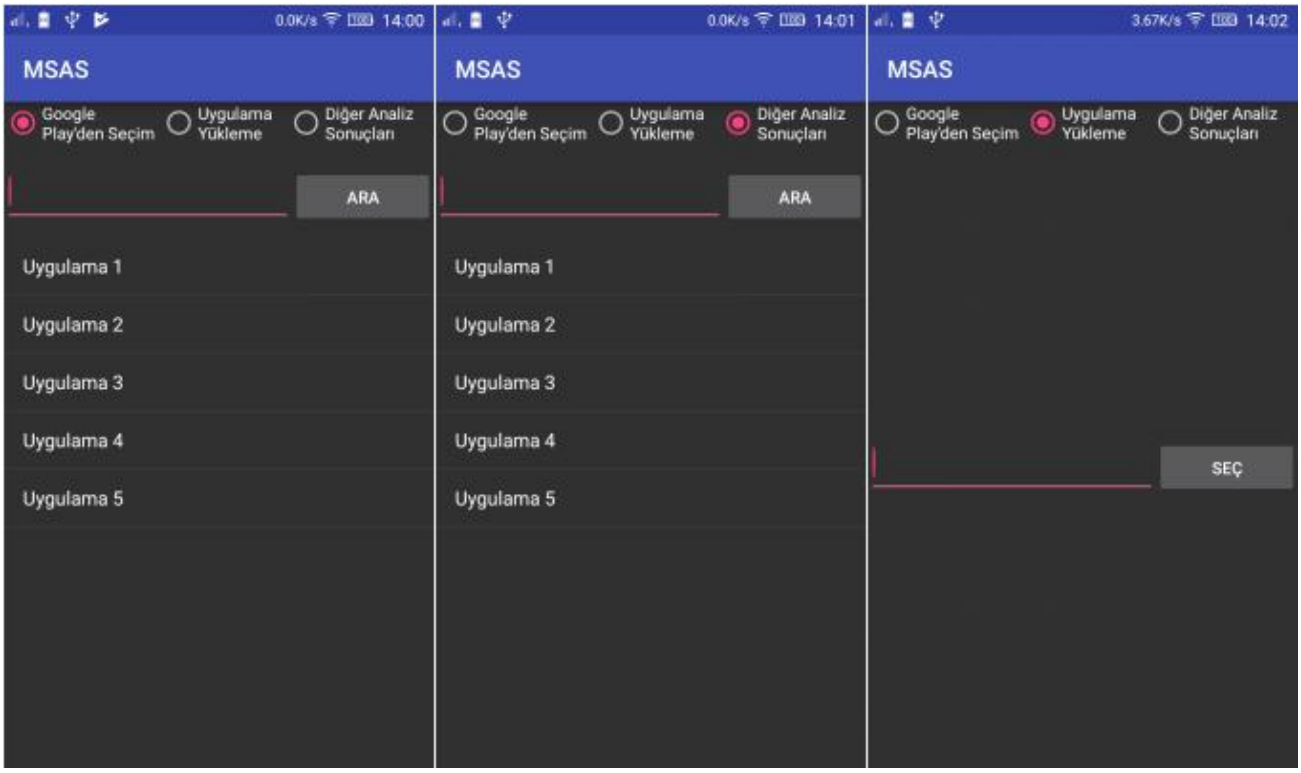


Şekil 17: Sistem Tasarımı

C.2. KULLANICI VE SİSTEM ARA YÜZÜ TASARIMLARI

C.2.1. Kullanıcı Veri Girişi / Seçim Ara Yüzleri

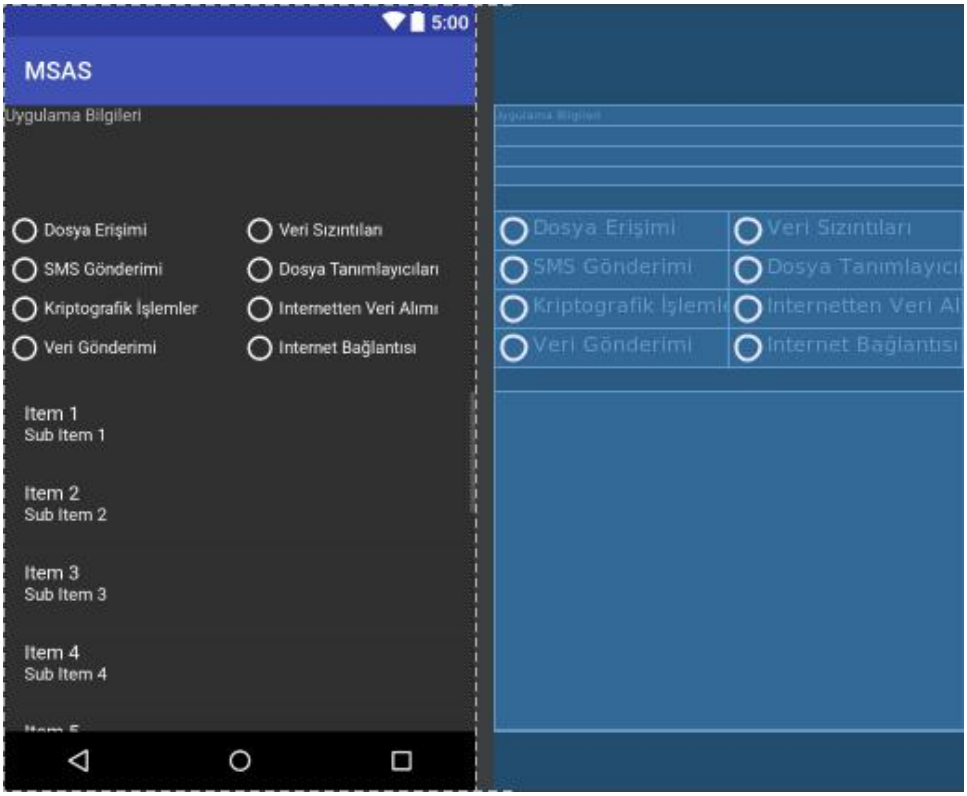
Kullanıcıların üst kısımdan seçtiği radio butonlara göre alt kısım şekillenecektir ve veri girişini gerçekleştirecektir. 'Google Play'den Seçim' butonuna tıkladığında arama kısmına aramak istediği uygulamayı yazıp o uygulamayı analiz ettirecektir. 'Uygulama Yükleme' butonunu seçtiğinde de kullanıcı kendi cihazında bulunan 'apk' dosyasını seçip uygulama analizini başlatacaktır. 'Diğer Analiz Sonuçları' butonuna tıkladığında ise eski analiz raporları listelenecektir ve istediği analiz raporunu seçip inceleyebilecektir.



Şekil 18: Analiz İçin Uygulama Seçim ve Analiz Sonuçları Listeleme Ekranı Tasarımı

C.2.2. Raporlama / Bilgilendirme Ara Yüzleri

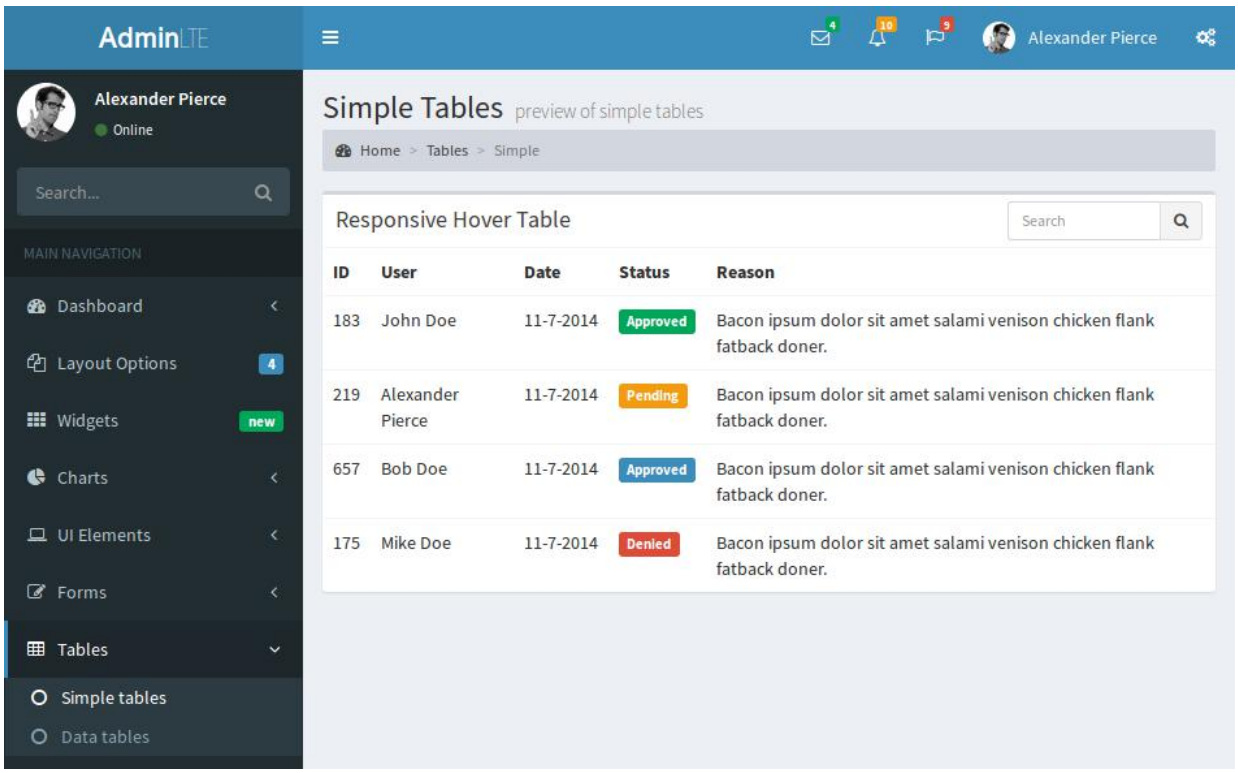
Raporlama arayüzünde analiz sonuçlarından gelen bilgiler, ilgili 'radio buton' lar ile filtrelenebilecektir. Filtre işlemleri sonucunda ilgili analiz sonuçları, altındaki listede görüntülenecektir.



Şekil 19: Uygulama Analiz Raporu Detaylı Görüntüleme Ekranı Tasarımı

C.2.3. Diğer Sistemler Veri Alış / Veriş Ara Yüzleri

Sistem yöneticisi için yönetim paneli AdminLTE yönetim paneli kullanılarak yapılacaktır. Sistem yöneticisi buradan analiz sonuçlarını, IP adreslerini ve kara listeyi görüp yönetebilecektir. AdminLTE şablonu Şekil 20' de gösterilmektedir.



Şekil 20: Yönetim Paneli Şablonu ve Tablo Tasarımı

C.3. TEST TASARIMI

C.3.1. Gereksinim Analizlerinden Teste Yönelik Hedeflerinin Detaylandırılması

Sistem, servis aracılığıyla mobil cihazlar ve yönetim paneliyle haberleşecektir. Aynı zamanda analiz sonuçlarının kullanıcılara bildirilmesi için mobil bildirim kullanılacaktır. Bu nedenle haberleşme ve analiz işlemleri için servislerin sorunsuz bir şekilde çalışması gerekmektedir. Buna bağlı olarak uygulama analizi, analiz sonucunun ayrıştırılması, servislerin çalışması, bildirim gönderilmesi, uygulama indirilmesi için blackbox testleri (fonksiyonellik testi, duman testi, kullanılabilirlik testi, yükleme testi) ve birim testleri yapılacaktır.

Uygulamanın analiz edilip son kullanıcıya raporu ulaşma süresi çok uzun olmamalıdır. Buna bağlı olarak da performans testleri yapılacaktır.

C.3.2. Fonksiyonel Test Tasarımı

Sistem, aşağıdaki fonksiyonları yerine getirecektir:
- Uygulamaları analiz etme
- Rapor detayı görüntüleme
- Rapor listesi görüntüleme
- Kullanıcıların IP engelini kaldırma
- Google Play Store' dan uygulama arama
- Google Play Store' dan seçilen uygulamanın sunucuya indirilmesi
- Uygulama APK' sı yükleme
- Uygulama ekleme, güncelleme, silme
- Uygulama geliştiricisi ekleme, silme, güncelleme

Yukarıda belirtilen fonksiyonlar için birim testi ve kara kutu testleri yapılacaktır. Her bir fonksiyon için 3 adet farklı girdi ve çıktının doğruluğu test edilecektir. Uygulama analizi için 3 adet uygulama analiz edilecek ve sonucun çıktığı test edilecektir.

C.3.3. Performans Test Tasarımı

Sistem için performans, uygulamanın analiz edilme süresi ve raporun düzenlenip kullanıcıya gönderilme süresine bağlıdır. Uygulamalar 10-30 saniye çalıştırılacaktır. Bu süreleri belirlerken DroidBox analiz örneklerine ve taranan uygulamalarda 'MainActivity' açıldığında işlemlerin en erken ve en geç yapılış sürelerine bakılmıştır. Bazı uygulamalarda 10 saniyede internete erişimi, sms gönderimi gibi işlemler yapılamamaktadır. 5 adet uygulama test edilmiştir.

Analiz işlemi için uygulamanın Android sanal makineye yüklenme, açılma süreleri, raporun döndürülmesi, ayrıştırılması olmak üzere toplamda 5 dakikayı geçmemelidir. Honeynet projesinin dockerized (docker imajına dönüştürülmüş) halde bulunan DroidBox imajında uygulamalar 10 saniye analiz edildiklerinde analiz işleminin tamamlanma süresi yaklaşık 5 dakika sürmektedir.

DroidBox, sunucumuz üzerinde direk kurulu olacağı için en geç 5 dakika süreceği tahmin edilmektedir. Buna bağlı olarak performans testi için 3 adet uygulama test edilip son kullanıcıya teslim edilme süresi hesaplanacaktır.

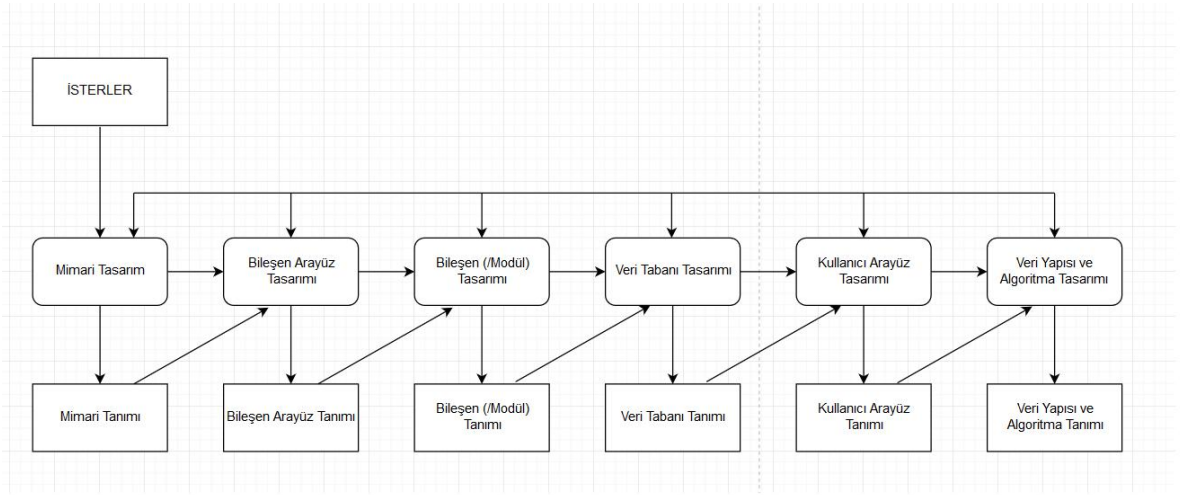
C.3.4. Kabul Testleri Tasarımı

Sistemin kullanıcı kabul testlerini geçebilmesi için aşağıdaki fonksiyonları yerine getirmesi gerekmektedir:
- Google Play Store' dan uygulama arama
- Seçilen uygulamanın analiz edilmesi
- Analiz raporunun görüntülenmesi
- Diğer analiz edilen uygulamaların raporlarının görüntülenmesi

Takip eden C4-C6 başlıklarındaki var olması gereken tasarımların ortalamaları alınır.

C.4. YAZILIM TASARIMI

Yazılım tasarımı için Şekil 21' de gösterilen tasarım taslağı tercih edilmiştir. Tasarım için bu döngü uygulanmıştır.



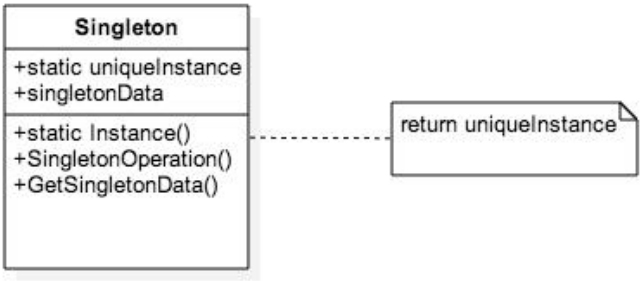
Şekil 21: Yazılım Tasarımı İş Planı

C.4.1. Gereksinime Bağlı Tasarım Kalıp(lar)ı Seçimi

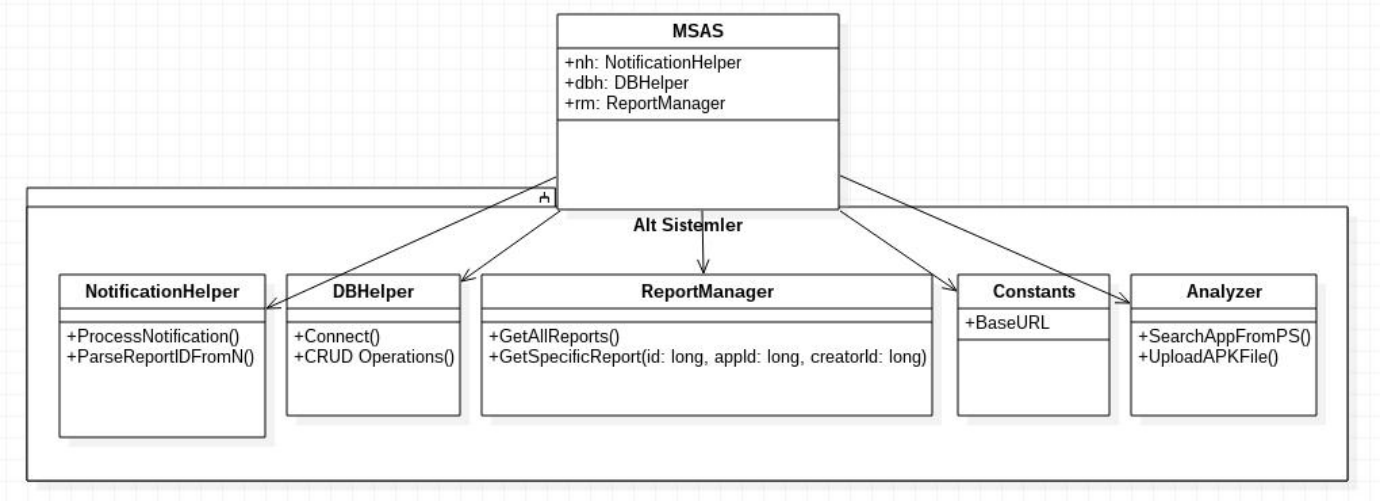
Yapılması planlanan sistem için 'Facade' ve 'Singleton' tasarım kalıpları tercih edilmiştir. Veritabanı işlemleri ve bildirim yollama işlemi için sınıflar singleton yapılacaktır ve analiz, rapor yönetimi gibi işlemler için de facade tasarım örüntüsü kullanılacaktır.

C.4.2. UML Kullanarak Tasarımı Diyagramları Oluşturma

UML diagramları StarUML kullanılarak oluşturulmuştur. Sisteme ait Facade tasarım kalıbı ve örnek bir singleton sınıf yapısı aşağıdaki gibidir.



Şekil 22: Singleton Sınıf Yapısı



Şekil 23: Mobil Uygulamanın Genel UML Diagramı

C.5. VERİ TABANI TASARIMI

C.5.1. Veri Tabanı İsterler Dokümanı

Veri tabanı içerisinde kullanıcılar ve bunlara bağlı olarak taranan uygulamalar ayrıca uygulama analiz ve raporları tutulacaktır. Bu kısımdaki kullanıcı tarafından beklenen isterler:

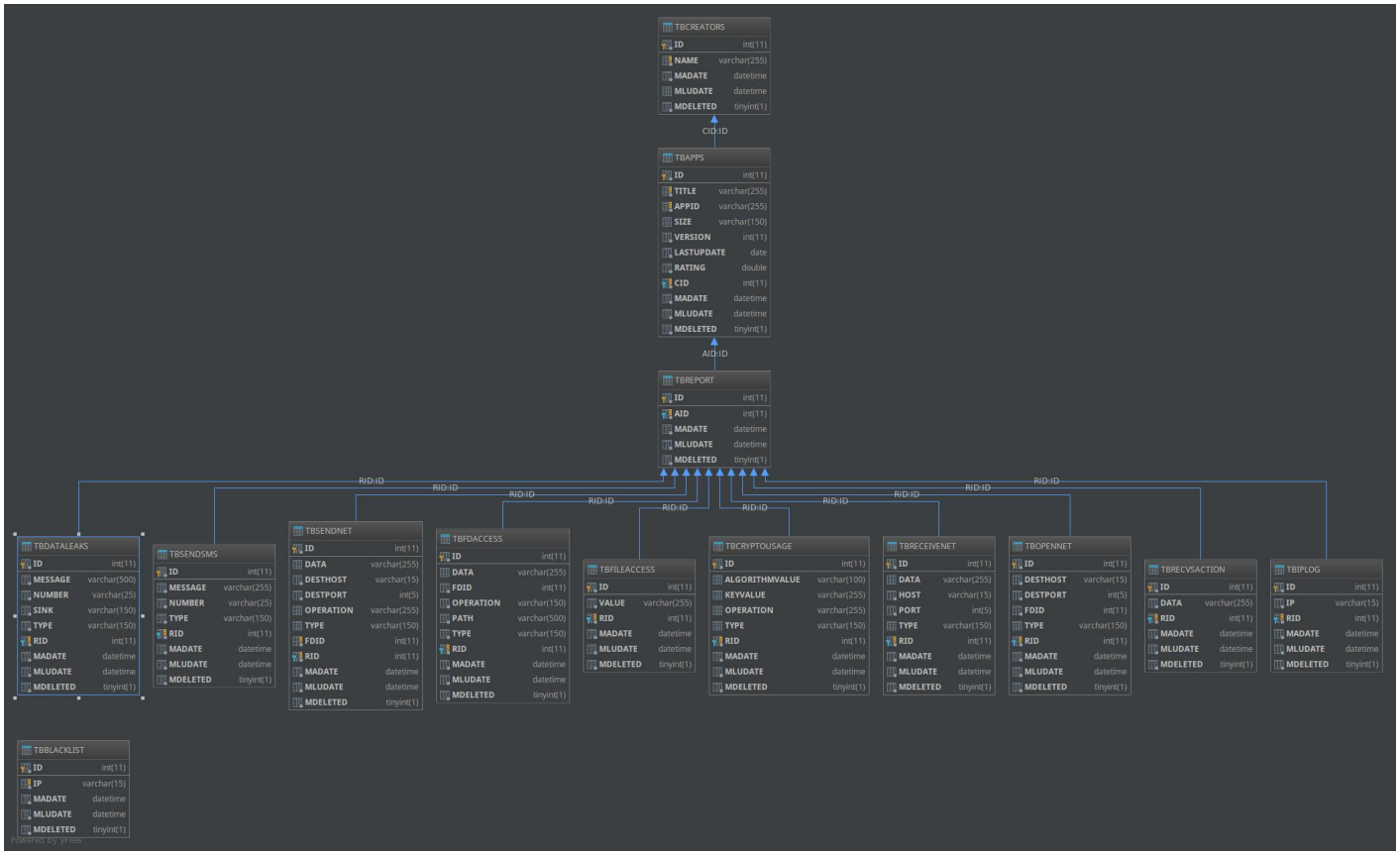
Analiz işlemini yapan kullanıcı için gerekli bilgilerin tutulması, bu bilgiler içerisinde yüklenme, güncellenme gibi işlemlerin tarih ve saat verileri olmalıdır. Aynı zamanda kullanıcıya bağlı olarak uygulamanın silinip silinmediği bilgisinin de tutulması gerekmektedir.

Bunun dışında uygulama için gereken isterler:

Her uygulamanın ID bilgisinin olması, isim, uygulama ID'si, uygulamanın boyut bilgisi, versiyon ve en son güncelleme bilgisinin tutulası gerekmedir. En başta belirtilen ID bilgisi taranan uygulamalar arasında bir eşsizliğin sağlanması açısından Primary Key olarak belirlenmelidir.
Uygulamalar için versiyon bilgileri tutulmalı ve yüklenen aynı uygulamanın bu verisi karşılaştırılarak eğer yeni bir versiyon eklenmemiş ise var olan analiz raporu döndürülmelidir. Bu şekilde aynı işlemlerin tekrar yapılmasından kaçınılmış ve etkili bir kullanım sağlanabilmektedir.

Son olarak da analiz raporu için belirlenen isterler şunlardır:
Her raporun bir ID değeri olmalıdır. Aynı zamanda rapor içerisinde analiz edilen uygulamanın ID değeri barındırılmalıdır.
Diyagram tasarımlarında iki farklı yaklaşım bulunmaktadır. İlk olarak json formatındaki verilerin birden fazla tablo içerisinde tutulması yöntemidir. Bu yöntem içerisinde her tablo için gelen veriler parse edilerek kolonlara yazılmaktadır. İkinci yöntem ise json formatindaki verilerin işlem bazında ayrıştırılmadan rapor id'leri ile saklanmasıdır. İki yöntem de uygulama aşamasında test edilecektir. Etkili ve performanslı olan yöntem seçilecektir.

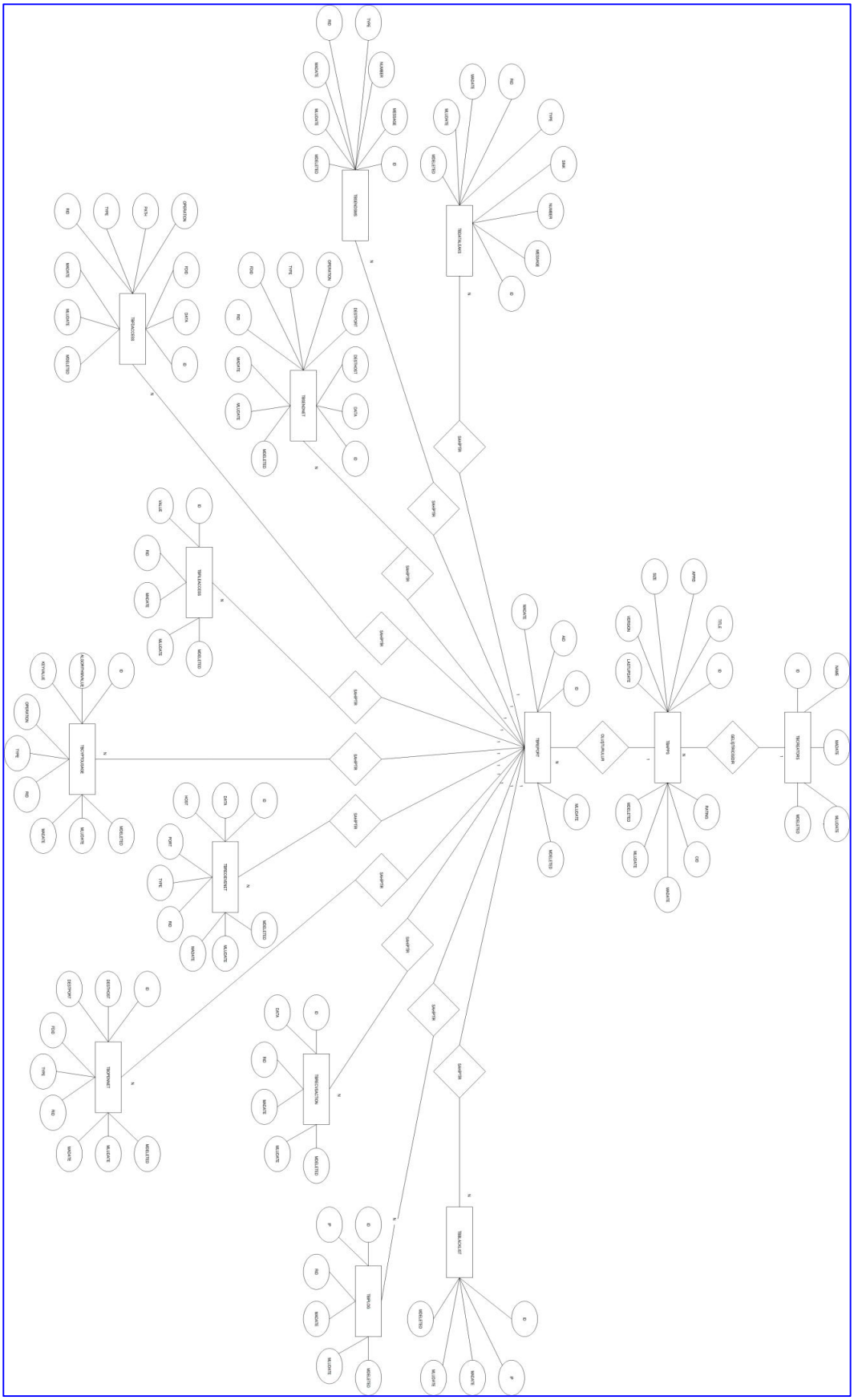
Veritabanı olarak ilişkisel veritabanı veya ilişkisel olmayan (NoSQL) veritabanı kullanılabilir. İlişkisel veritabanında veritabanı şeması Şekil 24' te gözükmektedir.



Şekil 24: İlişkisel Veritabanı Şeması

C.5.2. E/R Diyagramı

İsterlere bağlı olarak tasarlanan ER diagramı Şekil 25’ te gösterilmektedir. Her bir uygulama, bir geliştiriciye aittir. Bir geliştiricinin birden fazla uygulaması olabilir. Her uygulamanın birden fazla raporu olabilir fakat bir rapor sadece bir uygulamaya aittir. Raporun birden fazla filtresi (dosya erişimi, sms gönderimi vb.) olabilir fakat her bir filtre sadece bir rapora aittir.



Şekil 25: ER Diagramı

C.5.3. Stored Procedures, Triggerrs, Kısıtların Tasarımı

Prosedür tasarımları aşağıdaki gibidir:

- Uygulama geliştiricisi ekleme, silme, güncelleme, arama
- Uygulama bilgileri ekleme, silme, güncelleme, arama
- Versiyon değişikliği kontrolü
- Kara listeye kayıt ekleme ve silme
- Rapora bağlı tabloların kayıt prosedürleri

Eğer bir uygulama analiz raporu siliniyorsa, o uygulamaya bağlı oluşturulan işlem tablolarındaki (dosya erişimi vb.) bilgilerin de silinmesi gerekmektedir. Bu işlemler için trigger kullanılacaktır.

Kullanıcıların gün içinde tarayabilecekleri uygulama sayısı sınırlandırılmalıdır. Kara liste, sadece yönetici hesabı tarafından düzenlenebilmedir.

D. UYGULAMA

D.1. Geliştirilen Sistemin Sistem Tasarımlarını Karşılanmasının Değerlendirilmesi

Geliştirilen sistem, kullanıcı ara yüzleri tasarımı, veritabanı tasarımı ve performans testlerini karşılamaktadır. Sistemde beklenmedik sorunlardan dolayı değişiklikler yapılmıştır ve veritabanı ve sistem genel yapısına eklemeler yapılmıştır. Detaylar ve şekiller D.2, D.3 ve D.4 başlıklarında verilecektir.

Sistemde web servisler ile DroidBox bağlantısı ve çalıştırılması sağlanamamıştır. Linux işletim sistemli bir sunucu kullanıldığı için 'stdout' dosya tanımlayıcısı değiştirilmesi ve '>' operatörü ile çıktının istenilen dosyaya aktarılması test edilmiştir fakat başarılı olmamıştır. Bu nedenle sisteme kuyruk yapısı eklenmiştir.

Kuyruk yapısıyla web servislerden gelen istekler, veritabanında kuyruğa eklenmektedir ve sistemde çalışan 4 adet kuyruk yöneticisi bu kuyrukları sırayla işlemektedir. Bu 4 kuyruk; arama kuyruğu, indirme kuyruğu, analiz etme kuyruğu ve bildirim gönderme kuyruğudur. Bu kuyrukları yöneten bir de yönetici sınıf tanımlanmıştır. Facade tasarım örüntüsü de bu şekilde sağlanmıştır.

Kuyruklarda işlemlerin tamamlanması, kullanıcıya 'Google Firebase Push Notifications' ile bildirim yollanarak sağlanmıştır. Bildirimde arka planda gönderilen veriler sayesinde hangi işlemin tamamlandığı ekranda gösterilebilmektedir. Bu sayede kullanıcının uygulamayı sürekli açık tutmasına gerek kalmamaktadır. Google Firebase Push Notifications bildirim ve veriyi aynı anda gönderme işlemi Şekil 26' da gösterilmektedir. Eğer analiz sonucu ise rapor numarası, diğer durumda ise arama tamamlanma bildirimi olacağı için aranan anahtar kelime gönderilmektedir.

```
if isAnalysis:
    data = {'to': deviceId, 'notification':{'title': title, 'body': message, 'sound':'default', "click_action": "com.esogu.lab.msas.DetailsActivity"}, 'data': {'reportId': report_id}}
else:
    data = {'to': deviceId, 'notification':{'title': title, 'body': message, 'sound':'default', "click_action": "com.esogu.lab.msas.MainActivity"}, 'data': {'searchKey': message}}
```

Şekil 26: Google Firebase Push Notifications Bildirim ve Veri Gönderme

Sistemi test etmek için 3 adet uygulama analiz edilmiştir, 6 adet kelime Google Play' den aratılmıştır. Web servislere yapılabilecek istekler de test edilmiştir. Performans testi ve birim testleri için detaylar D.3 başlığında verilecektir.

D.2. Kullanıcı ve Sistem Ara yüzü Gerçeklemeleri

Mobil uygulama arayüzleri şekil 27' deki gibi gerçekleştirilmiştir.

Şekil 27' de mobil uygulamanın 16 adet ekran görüntüsü (soldan sağa doğru) bulunmaktadır. İlk ekran görüntüsünde Google Play üzerinden 'Esogu' kelimesi aratılmıştır. Sonucunda 3 adet uygulama listelenmiştir. Analiz işlemi yapmak için kullanıcıların liste üzerindeki uygulamalardan herhangi birine uzun basması gerekmektedir. Tıklamayla analiz başlatılmamasının nedeni, hatalı tıklamalarda analiz başlatılmasının önlenmesidir.

İkinci ekran görüntüsünde ise analiz edilen diğer uygulamaların sonuçları görüntülenmektedir. 'ESOGU Yemekhane' uygulaması test işlemleri sırasında 3 kere analiz edilmiştir. Tüm olası sonuçları gösterebilmek için 'DroidBox Test' uygulaması test edilmiştir. Üçüncü ve dördüncü resimde bu iki uygulamanın analiz raporu özeti görüntülenmektedir. Uygulamanın tehlikeli olup olmadığı, uygulamanın rehbere erişimi, resimlere erişimi ve video erişiminden belirlenmiştir. Resim düzenleme uygulamaları resimlere erişeceği için tehlikeli olmayabilir fakat olma ihtimali düşünüldüğü için basit bir karar mekanizması oluşturulmuştur.

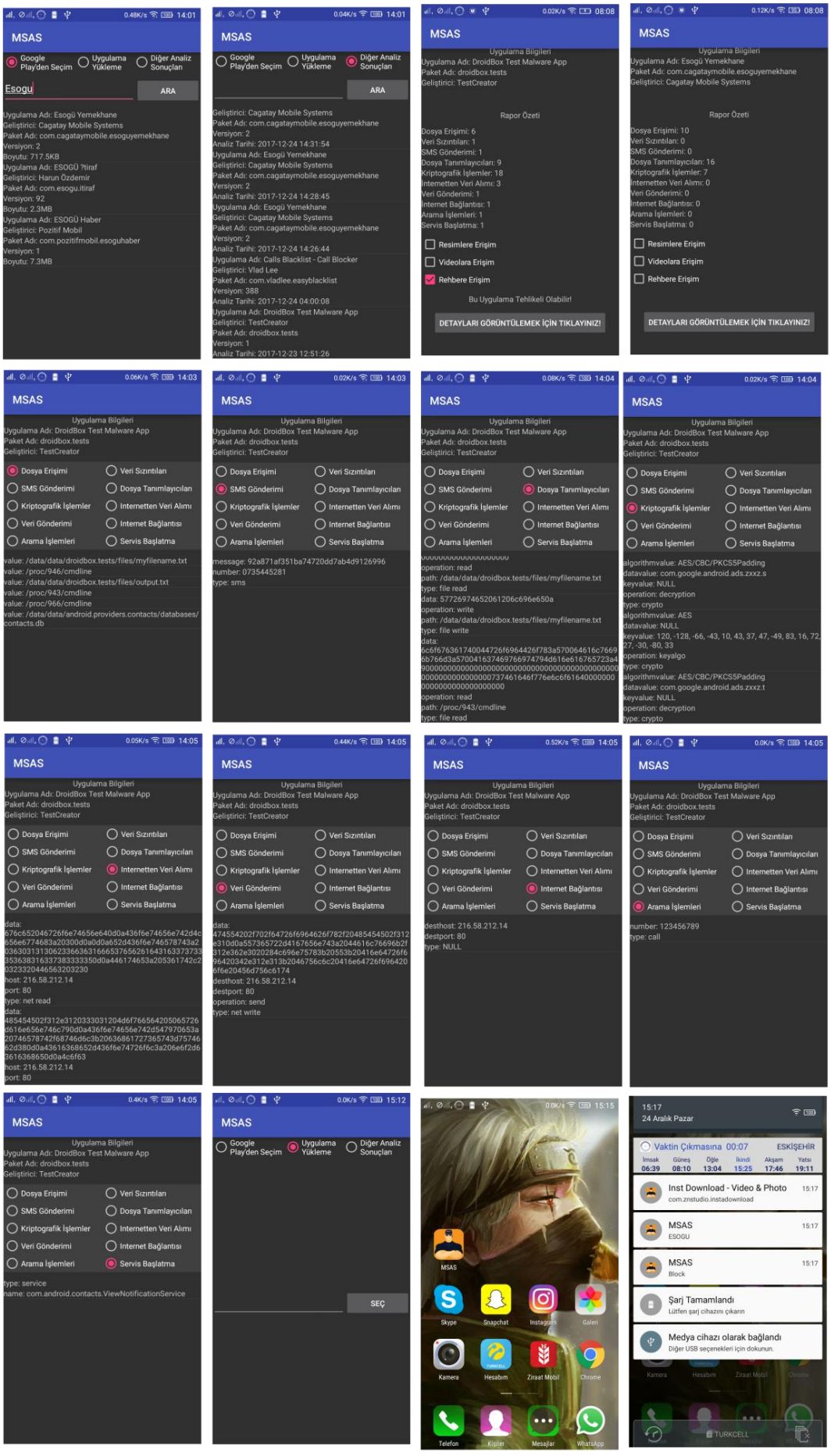
Detayları görüntülemek için üçüncü ve dördüncü uygulamalardaki detayları görüntüleme butonuna basılması yeterlidir. Daha sonra 5, 6, 7, 8, 9, 10, 11, 12 ve 13. ekran görüntülerinde görüntülenen filtreler açılacaktır. Filtreler ve açıklamaları aşağıdaki gibidir:

- Dosya Erişimi: Bu izin ile Android uygulamalar cihaza bağlı bulunan tüm bellek tiplerini yönetebiliyorlar. Aslında genellikle uygulamalar sadece kendileriyle ilgili olan işlemleri yerine getirmek için bu izine ihtiyaç duyuyorlar. Bu izni verdiğiniz zaman uygulamalar telefon belleğine ya da SD kartlara ekstra bilgileri yükleyip silebiliyor. Fotoğraf, video veya pdf dosyaları ele geçirildiğinde şantaj yapılabilecek bilgiler elde edebilirler.
- Veri Sızıntıları: İnternet tarayıcısındaki geçmişe ve çerezlere erişerek kimlik ve şfre bilgilerini ele geçirebilir. Veri sızıntıları aynı zamanda sms ile veri gönderimi ve internet üzerinden veri gönderimi de olabilir.
- SMS Gönderimi: Bu izin uygulamanın istenilmeyen SMS almasına ve göndermesine olanak sağlar. Dolayısıyla kullanıcının istemediği bir yere ücretli abonelik sağlanabilir. Finansal kaybın yanı sıra suç örgütlerinin bir parçası haline de gelinebilir.
- Dosya Tanımlayıcıları: Dosya tanımlayıcıları, dosyalara ve internete erişimi göstermektedir. Her erişimde bir dosya tanımlayısının kaydı görüntülenir. Dosya erişimi, internetten veri okuma ve yazma işlemleri verilerinin ortak alanda toplanmış halidir.
- Kriptografik İşlemler: Bu işlemler, dosyalarınızın şifrelenmesine ve tekrar erişememenize neden olabilir. Aynı zamanda başka bir uygulama ve veriyi de gizleme işlemleri yapılabilir.
- İnternetten Veri Alımı: Uygulama ile internetten konumunuza ulaşarak nerede olduğunuzu bilir. Birçok şey olabilir, örneğin hırsızlar sizin evde olmadığınız zamanı öğrenebilirler.
- Veri Gönderimi: İnternet üzerinden yapılan veri gönderme işlemlerini ve gönderilen veriyi gösterir.
- İnternet Bağlantısı: İnternete bağlanıp bağlanmadığını, bağlandıysa hangi ip ve port üzerinden bağlandığını gösterir. Kullanıcıların izni olmadan zararlı sitelere bağlantı sağlanıyorsa açığa çıkarılabilir.
- Arama İşlemleri: Kullanıcıdan izinsiz olarak herhangi bir telefon aranıp, tuşlamalar ile veri gönderilebilir ya da kullanıcılar finansal zarara girebilir.
- Servis Başlatma: Zararlı uygulamalar, kendilerini internete bağlanmıyor gibi gösterip diğer uygulamaların internet, sms vb. Izinlerini kullanabilir. Ekranda istenmeyen bildirimler, reklamları göstermek için de istenmeyen servisleri çalıştırabilir.

On dördüncü ekran görüntüsünde ise APK dosyası seçilerek uygulama analizi başlatma işlemi yapılabilir. Son kullanıcılara uygun değildir. Yine de teknik bilgisi yeterli kullanıcılar bu seçeneği kullanabilir.

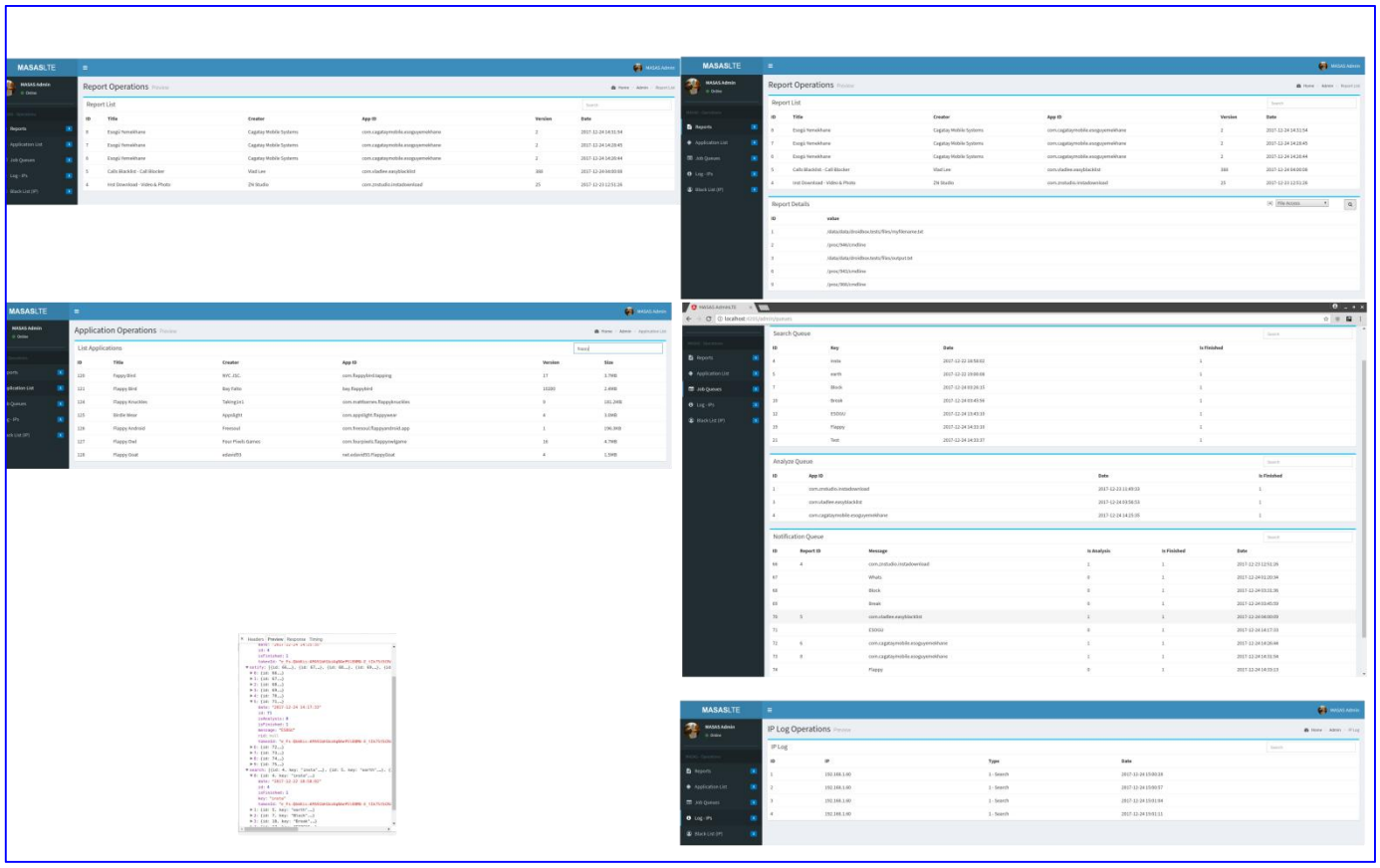
On beşinci ekran görüntüsünde uygulamanın ikonu görüntülenmektedir.

On altıncı ekran görüntüsünde ise 1 adet analiz sonucu ve 2 adet arama işleminin tamamlanma sonucu, bildirim olarak telefona gönderilmiştir. İsminde 'MSAS' yazan bildirimler, altındaki kelimelerin arama sonucunu belirtmektedir. İsminde Uygulama adı yazan bildirimler ise analiz sonucunun bittiğini göstermektedir. Tıklanarak ilgili detaylar görüntülenebilir. Eğer uygulama açıkken bildirim gelirse, 'Analiz Sonucu Tamamlandı!' ve 'Arama Sonucu Tamamlandı!' şeklinde gözükmektedir.



Şekil 27: Mobil Uygulama Ekran Görüntüleri

Şekil 28’ de de yönetim panelinin 6 adet ekran görüntüsü (soldan sağa doğru) gösterilmektedir.



Şekil 28: Yönetim Paneli Ekran Görüntüleri

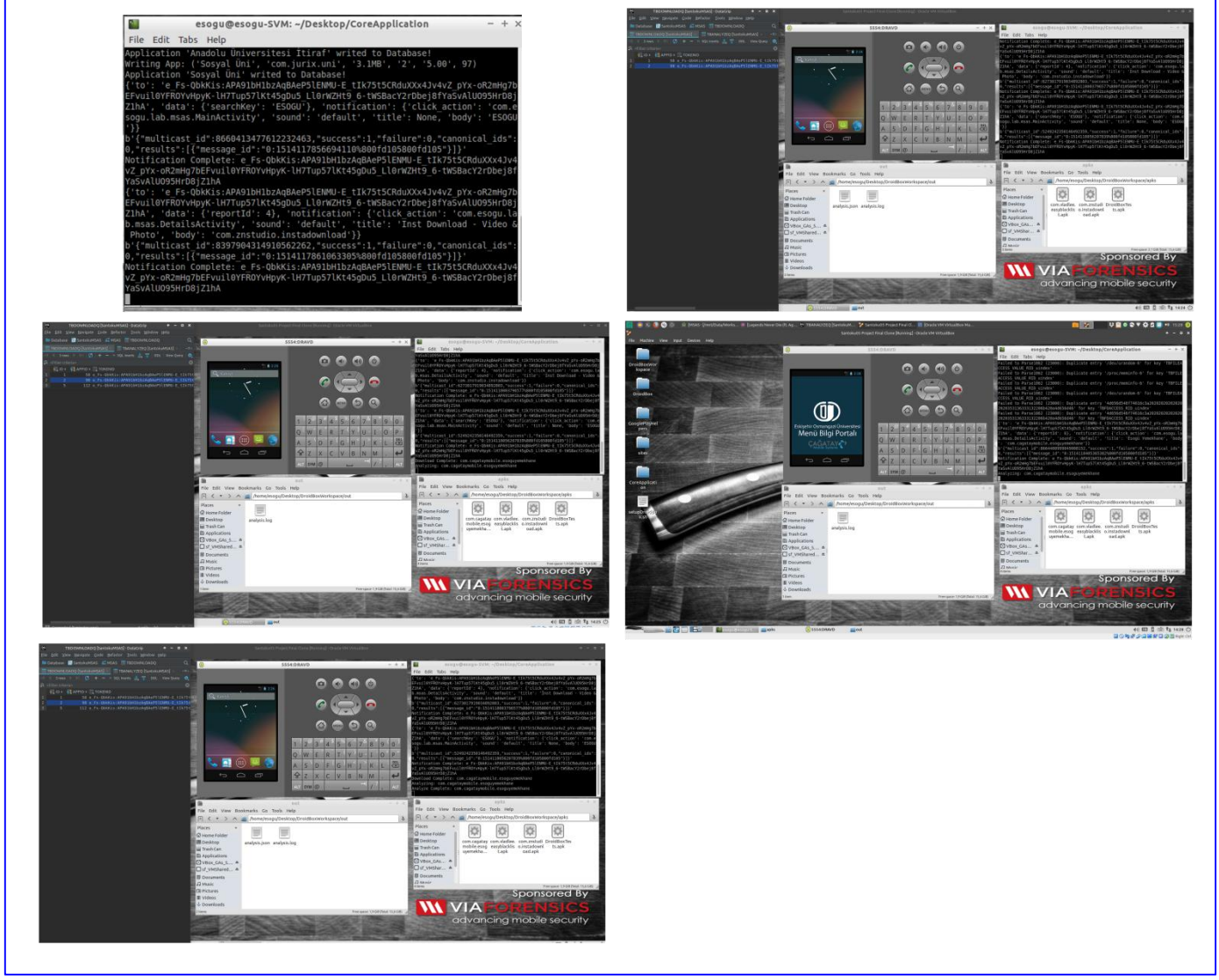
Birinci ekran görüntüsünde, analizi tamamlanan uygulamanın raporları görüntülenmektedir. Herhangi bir rapora tıklandığında ise ikinci ekran görüntüsünde olduğu gibi filtreler, ve yanında da rapor numarası görüntülenmektedir. Seçilen filtreye ait rapor detayı da ikinci açılan tabloda görüntülenmektedir.

Üçüncü ekran görüntüsünde ise uygulama listesi görüntülenebilmektedir ve arama yapılabilmektedir.

Dördüncü şekilde ise arama, indirme ve bildirim kuyrukları görüntülenmektedir. Sunucu tarafında bu kuyrukları işleyen Job’ lar bulunmaktadır. Her bir kuyruğun işlemi bittiğinde ise ‘isFinished’ sütunu 1 olmaktadır. Job’ ların detayları Yazılım Gerçeklemeleri başlığında detaylı açıklanacaktır. İş kuyrukları, mobil cihazların Google Firebase üzerinden bildirim gönderilmek için kullanılan cihaza özel numaraları (tokenId) ile birlikte kayıt edilmektedir. Beşinci ekran görüntüsünde ‘tokenId’ gösterilmektedir.

Kuyruğa eklenen her bir iş, IP kaydı tablosunda (IPLog) tutulmaktadır. Altıncı şekilde de bu kayıt tablosu gösterilmektedir. Yönetici tarafından sisteme saldırı yapılan IP’ ler kara listeye alınabilir.

Sunucu tarafı ana uygulama ile ilgili 5 adet ekran görüntüsü Şekil 29’ te gösterilmektedir.



Şekil 29: Sunucu Tarafı Ana Uygulama Ekran Görüntüleri

İlk ekran görüntüsünde Google Play den arama işlemi sonrası sonuçların veritabanına kaydedilme işlemi ve bildirim yollanma sonucu görüntülenmektedir. Buradaki terminal ekranı ana uygulamanın çalışma ekranıdır.

İkinci ve üçüncü ekran görüntüsünde ise ‘Esogu Yemekhane’ uygulaması analiz kuyruğuna sokulmuştur. Üçüncü ekran görüntüsünde görüldüğü gibi uygulama indirilmiş ve analiz işlemine başlanacaktır.

Dördüncü ekran görüntüsünde görüldüğü gibi ‘Esogu Yemekhane’ uygulaması analiz edilmektedir.

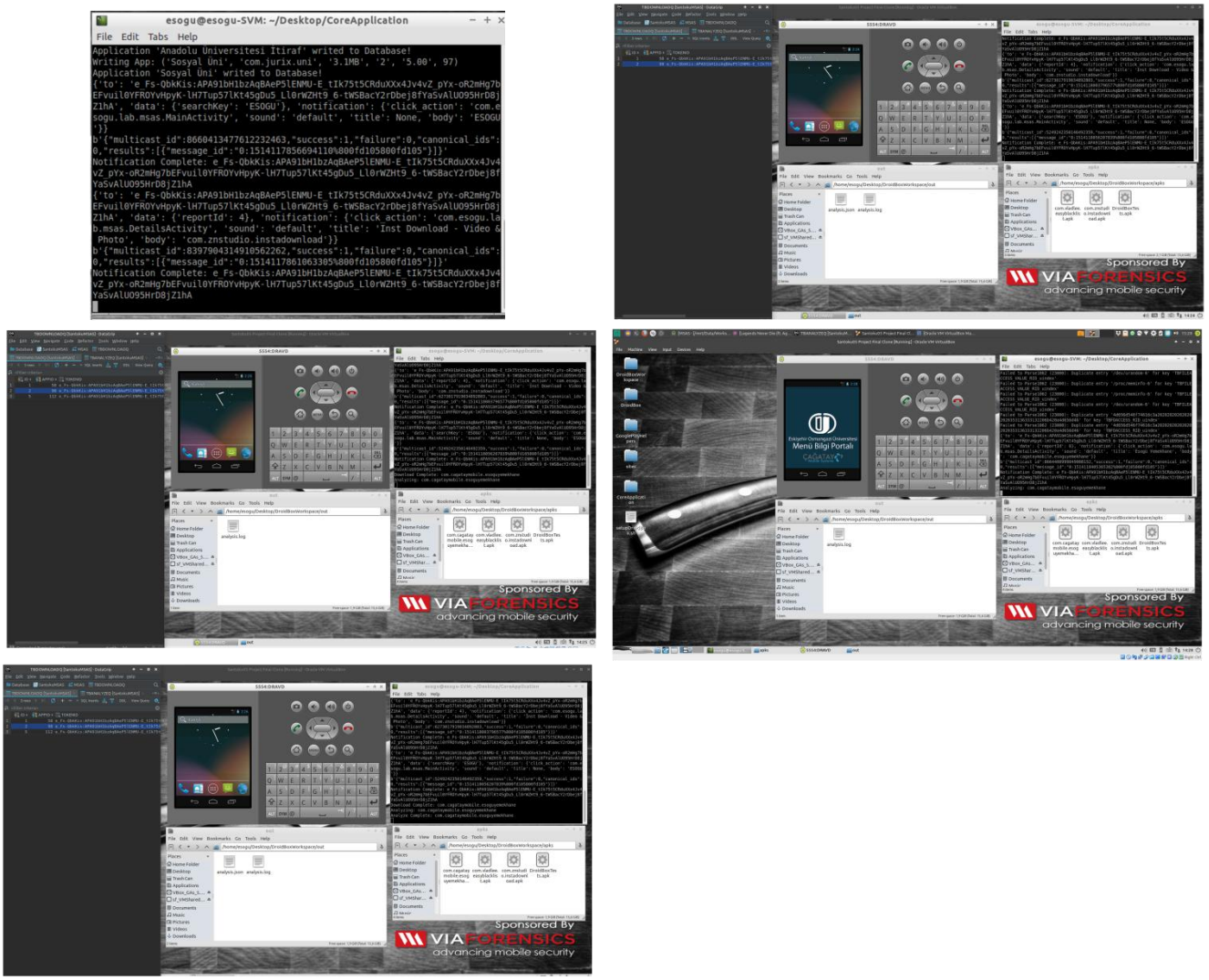
Beşinci ve son ekran görüntüsünde ise analiz işlemi tamamlanmış ve tamamlanma işleminin bittiği ana uygulama ekranında gösterilmektedir.

D.3. Gerçeklenen Testler

Sistem tasarımında yapılması planlanan tüm testler başarıyla sonuçlanmıştır. Kara kutu (BlackBox) testleri, performans testi ve birim testleri yapılmıştır. 3 uygulama analiz edilmiştir ve her bir filtre görüntüleme, kaydetme, uygulamaları analiz etme, analiz sonucunu ayrıştırma, kullanıcıya bildirim gönderme, yönetim panelinden ve mobil uygulamadan tüm analiz sonuçlarını görüntüleme, Google Play üzerinden arama yapma ve analiz işlemi için seçilen uygulamayı indirme işlemlerinin hepsi de başarılı bir şekilde sonuçlanmıştır.

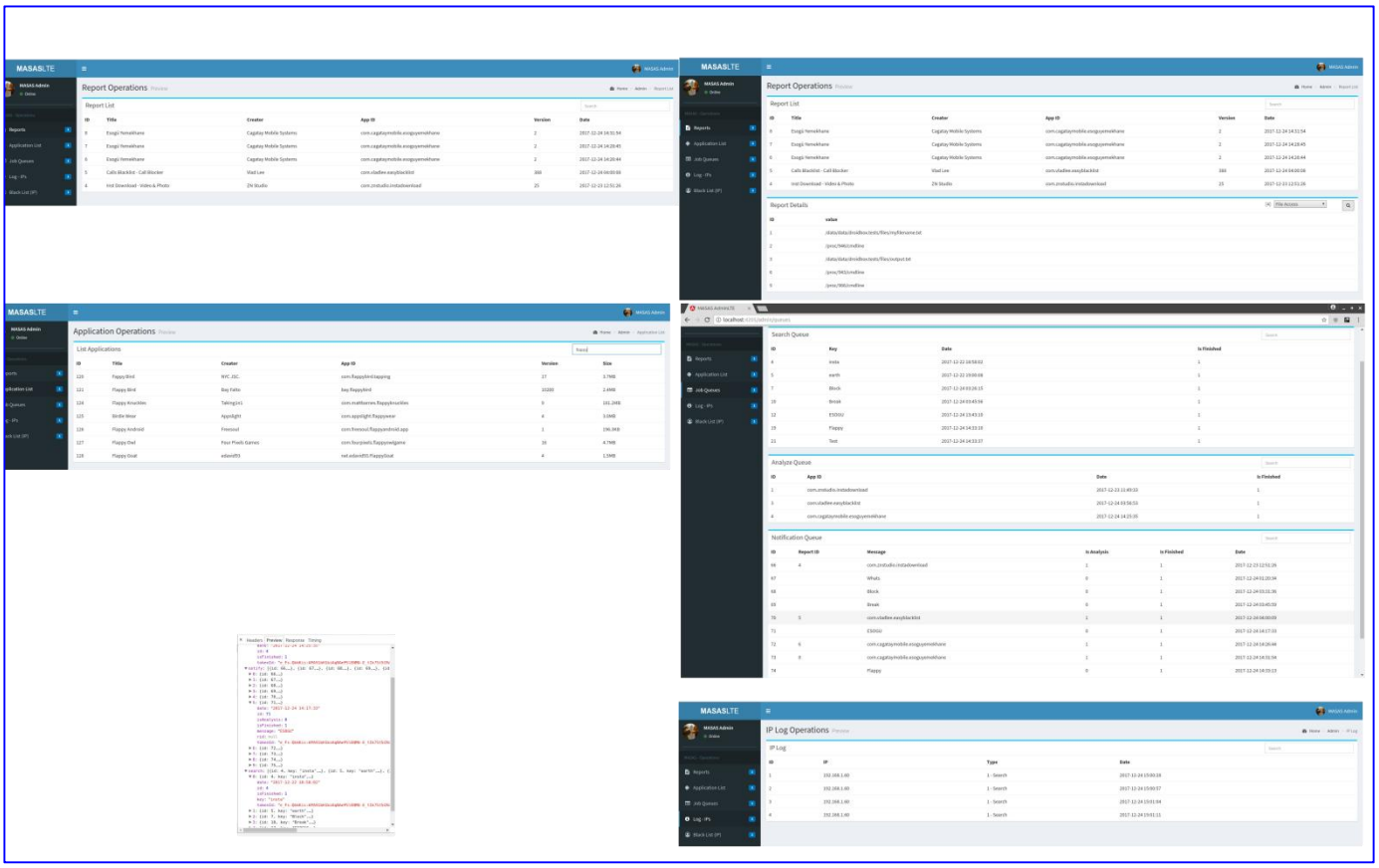
Yapılan performans testlerinde 3 uygulama 30 saniye analiz edilmiştir ve ortalama 95 saniye sürmüştür. Performans testinde de hedeflendiği gibi 5 dakikanın altında analiz işlemi tamamlanmıştır ve kullanıcıya bildirim gönderilmiştir.

Sunucu tarafında çalışan uygulamaya yapılan kara kutu testleri Şekil 30' da gösterilmektedir. Bu ekran görüntülerinde uygulama arama sonucunun veritabanına yazılması, kullanıcıya bildirim gönderilmesi, analiz işlemi başlatılması, uygulama indirilmesi, analiz sonucunun ayrıştırılması ve veritabanına kaydedilmesi gösterilmektedir.



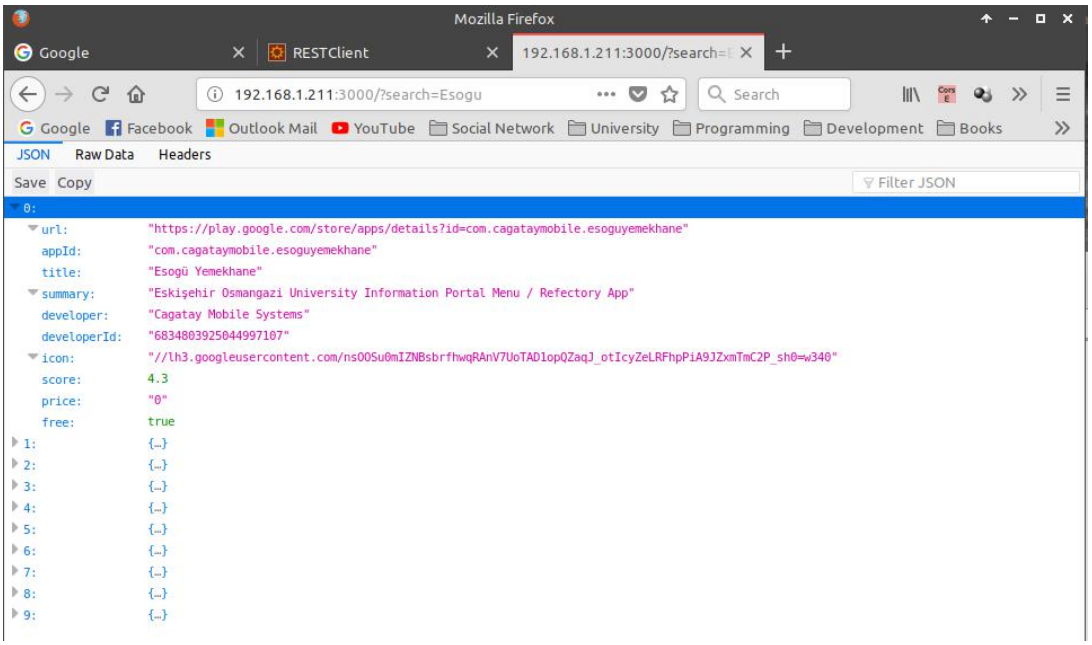
Şekil 30: Sunucu Tarafında Yapılan Kara Kutu Testleri

Yönetim panelinde yapılan kara kutu testleri Şekil 31' de gösterilmektedir. Bu ekran görüntülerinde ise tüm raporları görüntüleme, filtreleri seçme, analiz kuyruklarını görüntüleme, ip kaydını görüntüleme ve mobil cihazdan gelen bildirim için kullanılan tokenID' nin doğrulanması görüntülenmektedir.



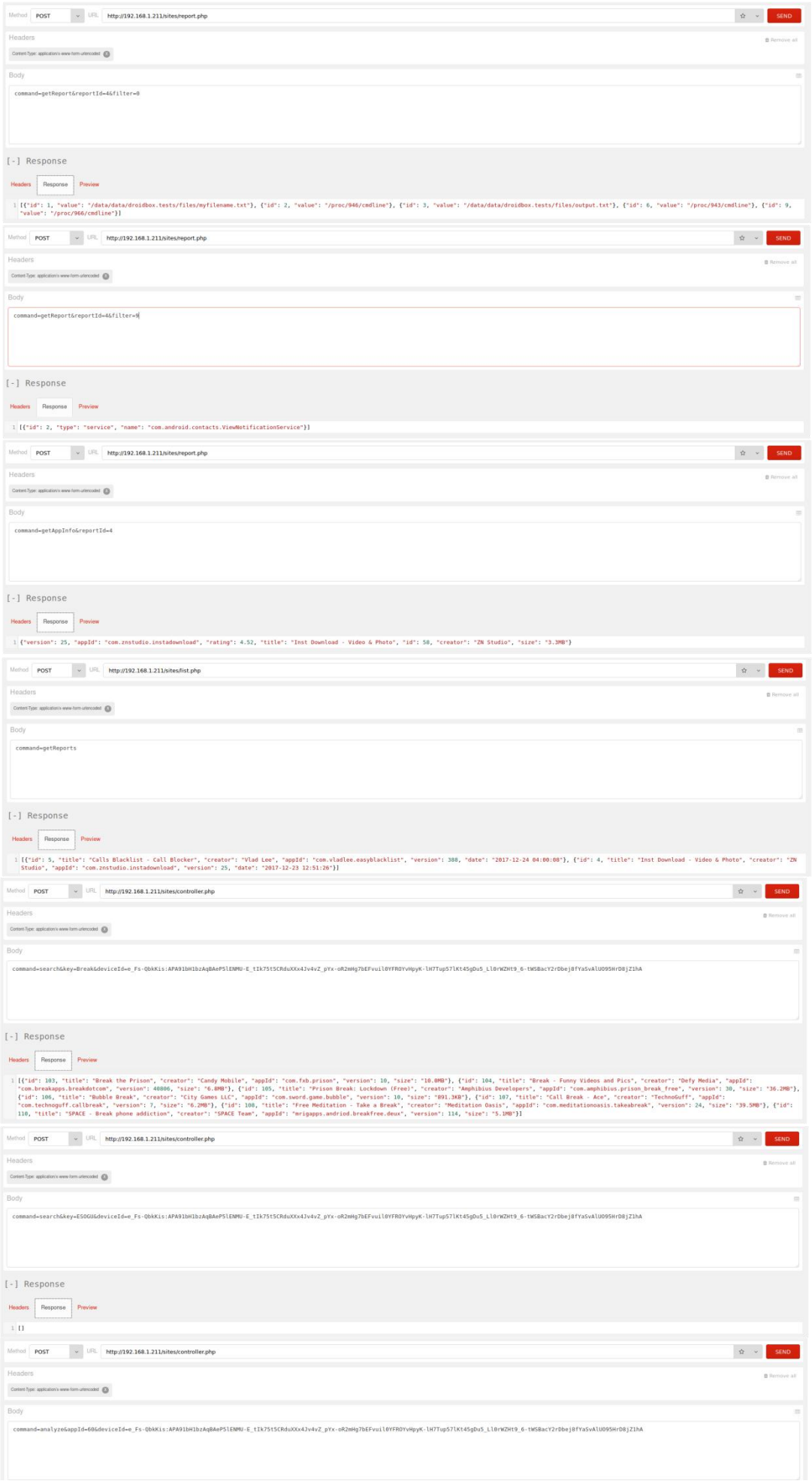
Şekil 31: Yönetim Paneli Kara Kutu Testleri

Google Play Scrapper' ın fonksiyonellik testi ise Şekil 32' de gösterilmektedir.



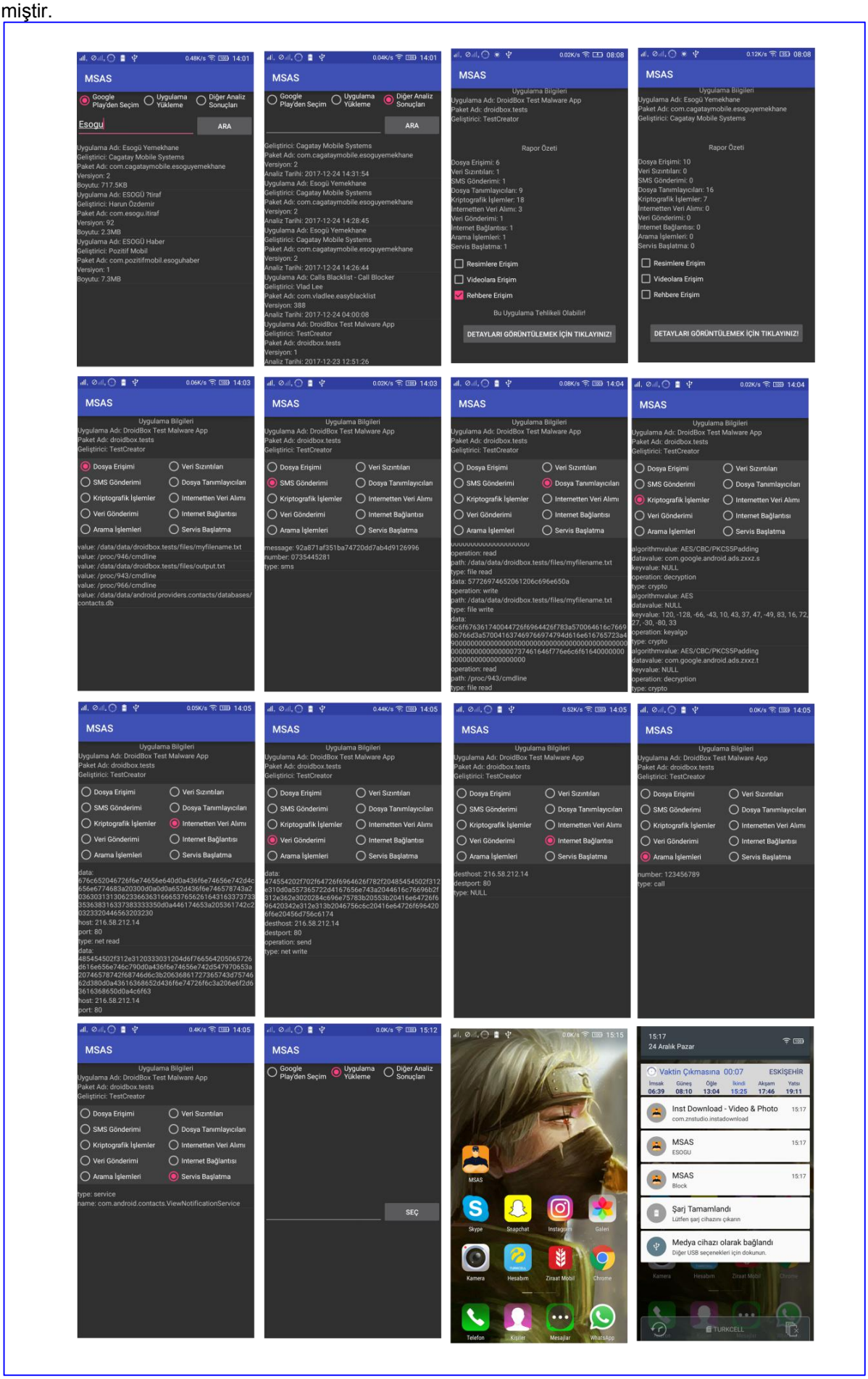
Şekil 32: Google Play Scrapper Fonksiyonellik Testi

Web servisler için yapılan fonksiyonellik testlerinden bazıları Şekil 33' te gösterilmektedir. Servislere yapılabilecek her bir istek ve çıktısı test edilmiştir. İsteklerle ilgili detaylar D.4 başlığında verilecektir.



Şekil 33: Web Servislere Yapılan Kara Kutu Testleri

Mobil uygulamada yapılan kara kutu testleri Şekil 34' te gösterilmektedir. Google Play üzerinden arama yapma, analiz sonuçlarını görüntüleme, raporların detaylarını ve filtreleri görüntüleme, bildirim alma ve görüntüleme işlemleri test edilmiştir.



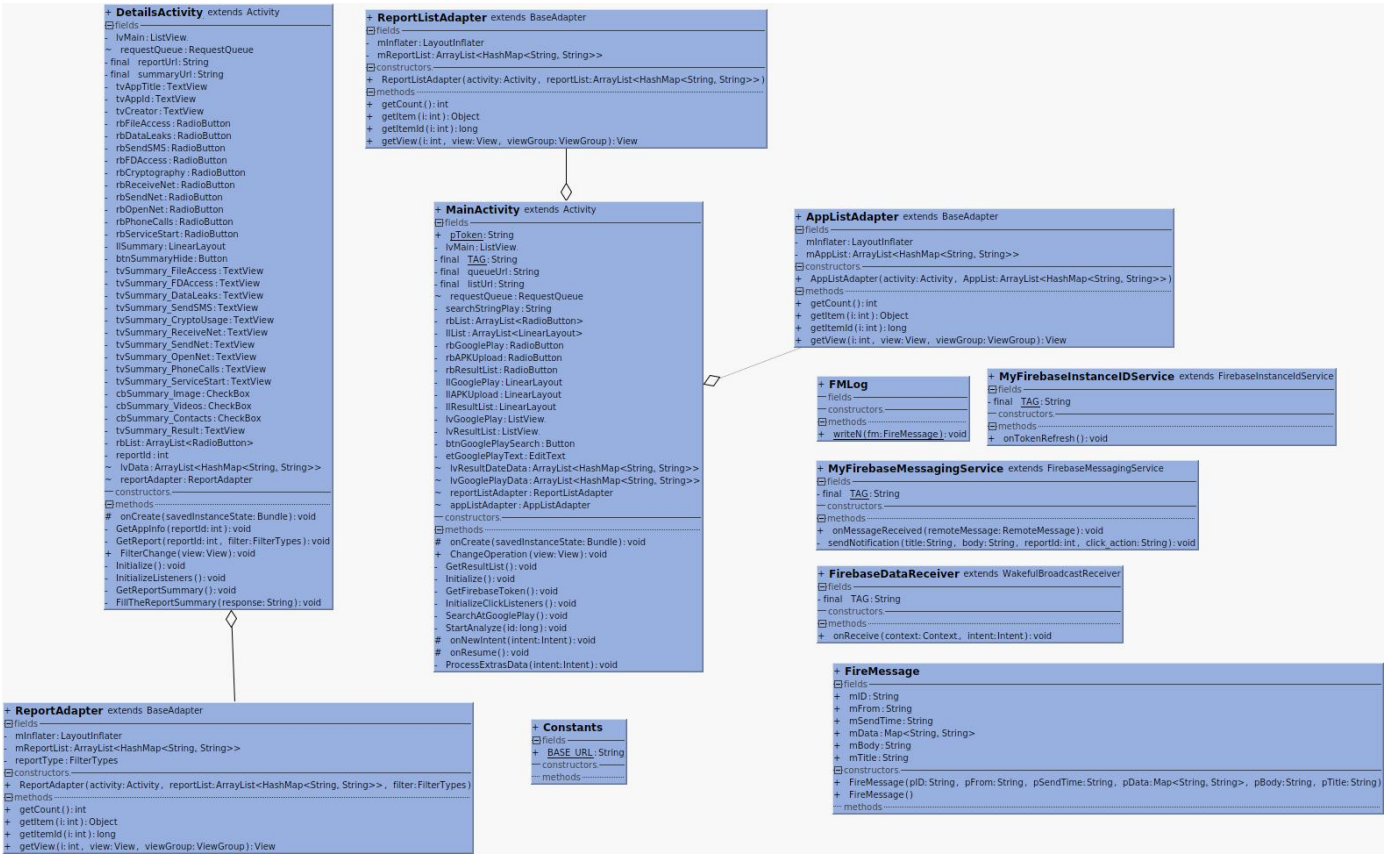
Şekil 34: Uygulama Arama, Analiz Etme, Bildirim Alma, Rapor Görüntüleme, Filtreleri Görüntüleme İşlemleri İçin Yapılan Kara Kutu Testleri

D.4. Yazılım/Veri Tabanı/Donanım Gerçeklemeleri

Yazılım gerçeklemesinde aşağıdaki zorluklarla karşılaşılmıştır:

- Projede kullanılan GooglePlayCli ve Google Play Unofficial Python API, arama sonucunda oluşan çıktıyı Web servis üzerinden istek atan kullanıcıya döndürmemektedir. Bu nedenle farklı Google Play apileri araştırılmıştır ve sonucunda GooglePlay Scrapper bulunmuştur. Bu araç ise nodejs üzerinde çalışmaktadır. Arama yapmaktadır fakat indirme işlemini yapmamaktadır. Bu nedenle proje için yeni bir yapı geliştirilmiştir (Kuyruk Yapısı). Bu kuyruk yapısıyla indirme ve arama işlemleri sadece kuyruğa eklenmiştir ve sunucuda çalışan kuyruk işlerini yöneten uygulamalar sayesinde bu kuyruklar kolaylıkla işlenebilmiştir.
- DroidBox, root kullanıcısı üzerinde çalışırken Apache Web Server kullanıcısı root olamamaktadır. DroidBox da kullanıcı üzerinde çalıştırılarak bu sorun çözülmüştür.
- DroidBox' ın çalışması için Android SDK nın eski bir sürümü gerekmektedir. Bu eski sürümden Android API 16 r5 ARM image indirilmelidir. Bu aşama için Android API 16 r5 indirilip emulator derlenmiştir. Yeni SDK'lar üzerinde çalışmayınca tekrar eski SDK sürümüne geçilmiştir.
- DroidBox, HoneyNet projesinde Docker imajı olarak da kullanılabilmektedir. Fakat docker imajının çalışması, bu imajda çalışan sistem ile sanal makinenin bağlantı süresi çok uzun olduğu için kendi sunucumuz üzerinde DroidBox kurulumu gerçekleştirilmiştir. Kurulum sonucunda DroidBox geliştirilerek çıktı dosyası düzenlenmiştir ve istenilen klasöre çıktı vermesi sağlanmıştır.
- Analiz sonucu cihaza bildirildikten sonra uygulama kapalı ise tıklama sonucunda ilgili rapor gösterilememektedir. Bunun nedeni ise Google Play Push Notifications bildiriminin içerisinde veri gönderilmemesidir. Veri anahtarı da bildirime eklenerek bu sorun çözülmüştür.

Mobil yazılımın UML sınıf şeması şekil 35' te verilmektedir.



Şekil 35: Mobil Uygulama UML Sınıf Şeması

Mobil uygulamada bildirimleri yönetmek için FMLog, InstanceIDService, MessagingService, DataReceiver ve FireMessage sınıfları oluşturulmuştur.

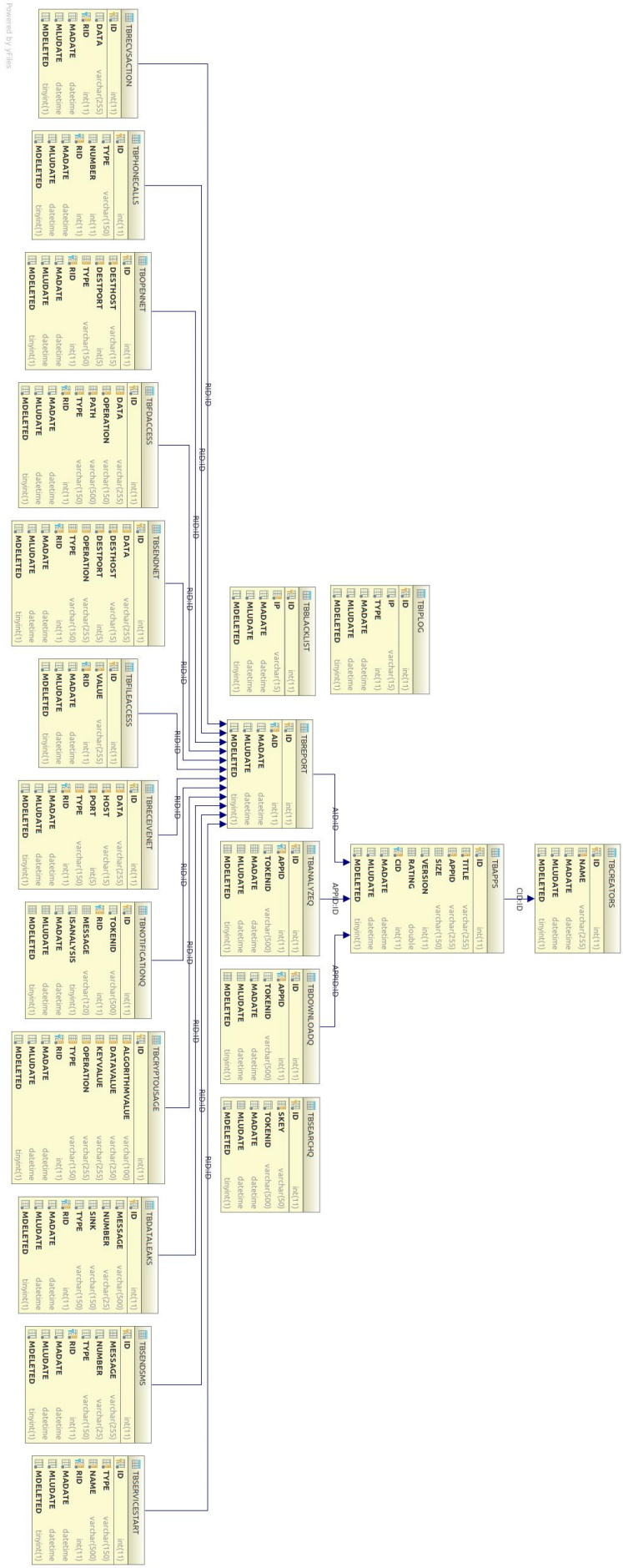
FMLog sınıfı gelen bildirimlerin mobil cihaz hafızasında kaydının tutulmasını sağlamaktadır. Bildirim içeriği ayrıştırıldıktan sonra FireMessage sınıfından bir nesneye doldurulmaktadır.

MessagingService, bildirim geldiğinde çalıştırılan fonksiyona sahiptir.

InstanceIDService ise mobil cihaza ait token id' yi üretmektedir.

FirebaseDataReceiver sınıfı ise mobil uygulama kapalı olsa bile gelen bildirimlerin işlenmesini sağlamaktadır.

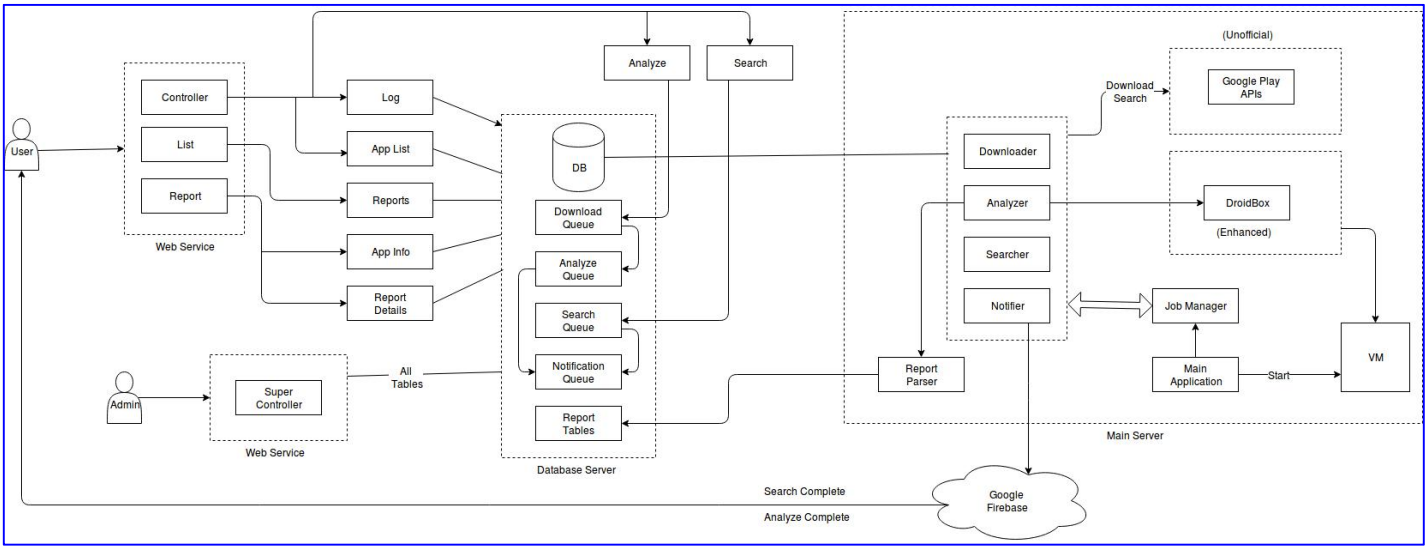
Kuyruk yapısı oluşturulduğu için sistemin veri tabanı tasarımı şekil 36’ da olduğu gibi değiştirilmiştir.



Şekil 36: Veri Tabanı Şeması

İndirme kuyruğu, arama kuyruğu, analiz kuyruğu ve bildirim kuyruğu olmak üzere toplamda 4 adet kuyruk tablosu oluşturulmuştur. Bildirim tablosundaki 'isAnalysis' sütunu, arama tamamlanma bildirimi mi yoksa analiz tamamlanma bildirimi mi gönderileceğini belirler.

Sistemin genel çalışma yapısı Şekil 37' de gösterilmektedir.



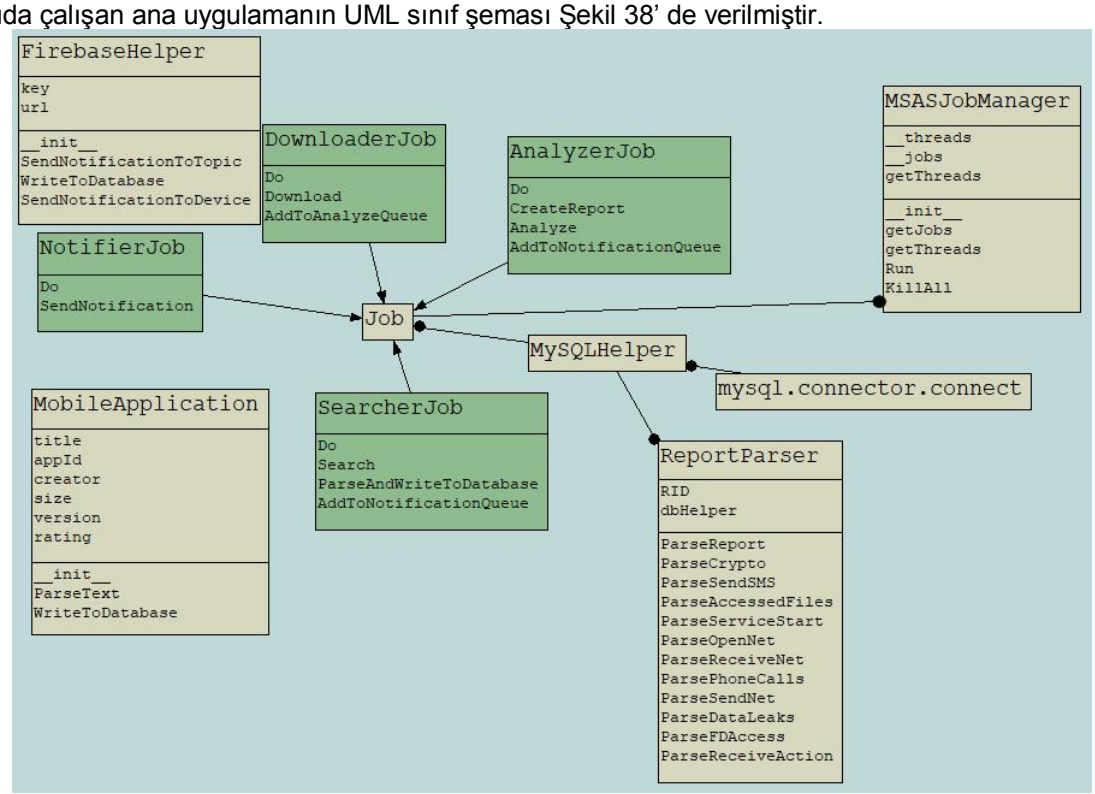
Şekil 37: Sistemin Genel Çalışma Yapısı

Sistem, application server, main server ve database server olmak üzere 3 farklı sunucuda dağıtık olarak çalıştırılabilir. Google Play API ve DroidBox da farklı sunuculara taşınabilir. Analiz işlemi tek bir DroidBox üzerinden yapılmaktadır. Farklı sunuculara birden fazla DroidBox kurulup, bu analiz sistemlerini yöneten bir yapı ile eş zamanlı olarak analiz kuyrukları işlenebilir. Dağıtık olarak çalıştırılabilecek alt sistemler, kesikli çizgi içine alınmıştır.

Bu yapıda istekler kuyruklara ekleme yaparken ana sunucudaki JobManager' ın yönettiği 4 adet Job, bu kuyrukları sırasıyla işlemektedir.

GooglePlay API olarak 3 adet API kullanılabilmektedir. Google Play Scrapper, daha hızlı ve daha fazla bilgi sunmasına rağmen indirme işlemini yapmamaktadır. Bu nedenle 'egirault - GooglePlay API' kullanılmıştır.

Ana sunucuda çalışan ana uygulamanın UML sınıf şeması Şekil 38' de verilmiştir.



Şekil 38: Ana Uygulamanın UML Sınıf Şeması

Web servisler ve yardımcı dosyaları şekil 39' da verilmiştir. Servislerin hepsi POST isteği almaktadır.

appinfo.py	12/23/2017 9:57 PM	PY File	2 KB
applist.py	12/24/2017 4:21 AM	PY File	2 KB
controller.php	12/24/2017 3:59 PM	PHP File	3 KB
download.py	12/24/2017 4:56 AM	PY File	1 KB
list.php	12/24/2017 4:06 AM	PHP File	1 KB
logger.py	12/24/2017 3:48 PM	PY File	1 KB
MySQLHelper.py	12/22/2017 7:57 PM	PY File	1 KB
MySQLHelper.pyc	12/22/2017 7:49 PM	PYC File	2 KB
report.php	12/23/2017 9:46 PM	PHP File	2 KB
report.py	12/23/2017 11:42 ...	PY File	3 KB
reportlist.py	12/24/2017 1:28 AM	PY File	2 KB
search.py	12/23/2017 8:32 PM	PY File	1 KB
summary.php	12/28/2017 2:58 PM	PHP File	1 KB
summary.py	12/28/2017 9:14 PM	PY File	5 KB
superapplist.py	12/25/2017 3:46 AM	PY File	2 KB
supercontroller.php	12/25/2017 4:14 AM	PHP File	3 KB
superiplog.py	12/25/2017 4:30 AM	PY File	2 KB
superqueuelist.py	12/25/2017 4:56 AM	PY File	3 KB

Şekil 39: Web Servisler ve Yardımcı Dosyaları

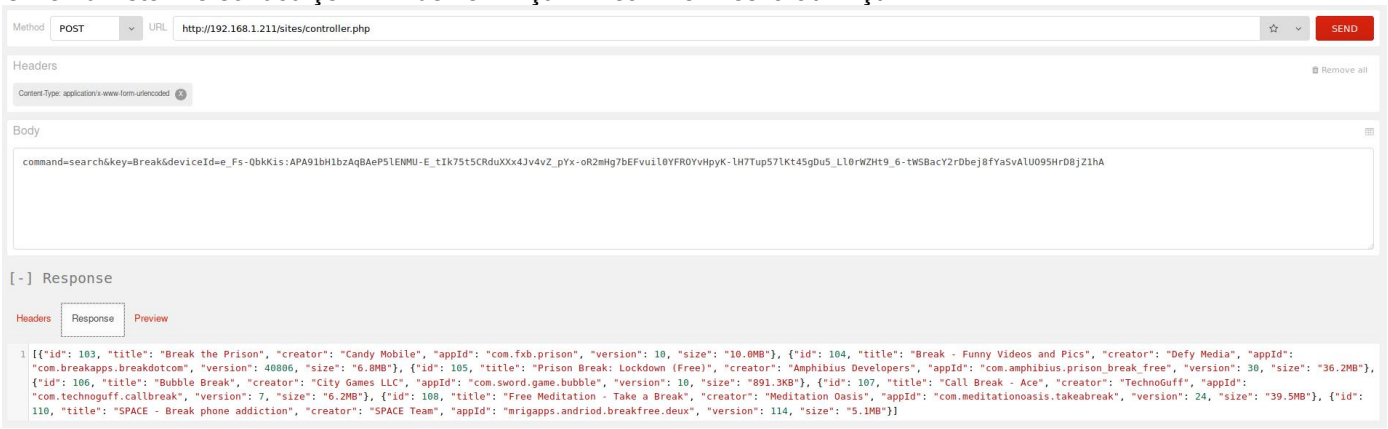
Gönderilebilecek olası komutlar ve aldıkları parametreler:

- **analyze** (appId, deviceId) - controller.php: Analiz edilecek uygulamanın AppID si ve analiz isteği yollayan mobil cihazın Google Firebase bildirimi için cihaz id sini parametre olarak alır. Analiz işlemi başlatmak için kullanılır.
- **search** (key, deviceId) - controller.php: Aranacak kelimeyi ve arama isteği yollayan mobil cihazın Google Firebase bildirimi için cihaz id sini parametre olarak alır. Google Play' den arama işlemi başlatmak için kullanılır.
- **getReports** - list.php: Geçmiş analiz sonuçlarını görüntülemek için kullanılır.
- **getReport** (reportId, filter) - report.php: Analiz sonucunu detaylı olarak görüntülemek için kullanılır. Rapor numarası ve detayı istenilen filtrenin numarasını alır. Filtreler şekil 40' ta verilmiştir. 'Dosya İşlemleri' 0 olacak şekilde artarak gitmektedir.
- **getAppInfo** (reportId) - report.php: Analiz raporuna ait uygulamanın bilgilerini almada kullanılır. Rapor numarasını parametre olarak alır.
- **getIPLog** - supercontroller.php: IP kayıtlarını görüntülemek için kullanılır (Yönetim Paneli).
- **getBlackList** - supercontroller.php: Kara Listeyi görüntülemek için kullanılır (Yönetim Paneli).
- **getQueues** - supercontroller.php: Kuyrukları görüntülemek için kullanılır (Yönetim Paneli).
- **getSummary** (reportId) - summary.php: Rapor özeti görüntülemek için kullanılır. Rapor numarasını parametre olarak alır.

```
public enum FilterTypes {
    DosyaIslemi,
    VeriSizintisi,
    SMSGonderimi,
    DosyaTanimlayicilari,
    Kripto,
    NetAlimi,
    NetGonderimi,
    NetBaglantisi,
    Cagri,
    ServisBaglantisi
}
```

Şekil 40: Filtreler

Örnek bir istek ve sonucu şekil 41' de verilmiştir. 'Break' kelimesi aratılmıştır.



Şekil 41: Örnek Arama İsteği ve Sonucu

Ana uygulamayı çalıştırmak için bulunduğu dizine gidip:

python3 main.py

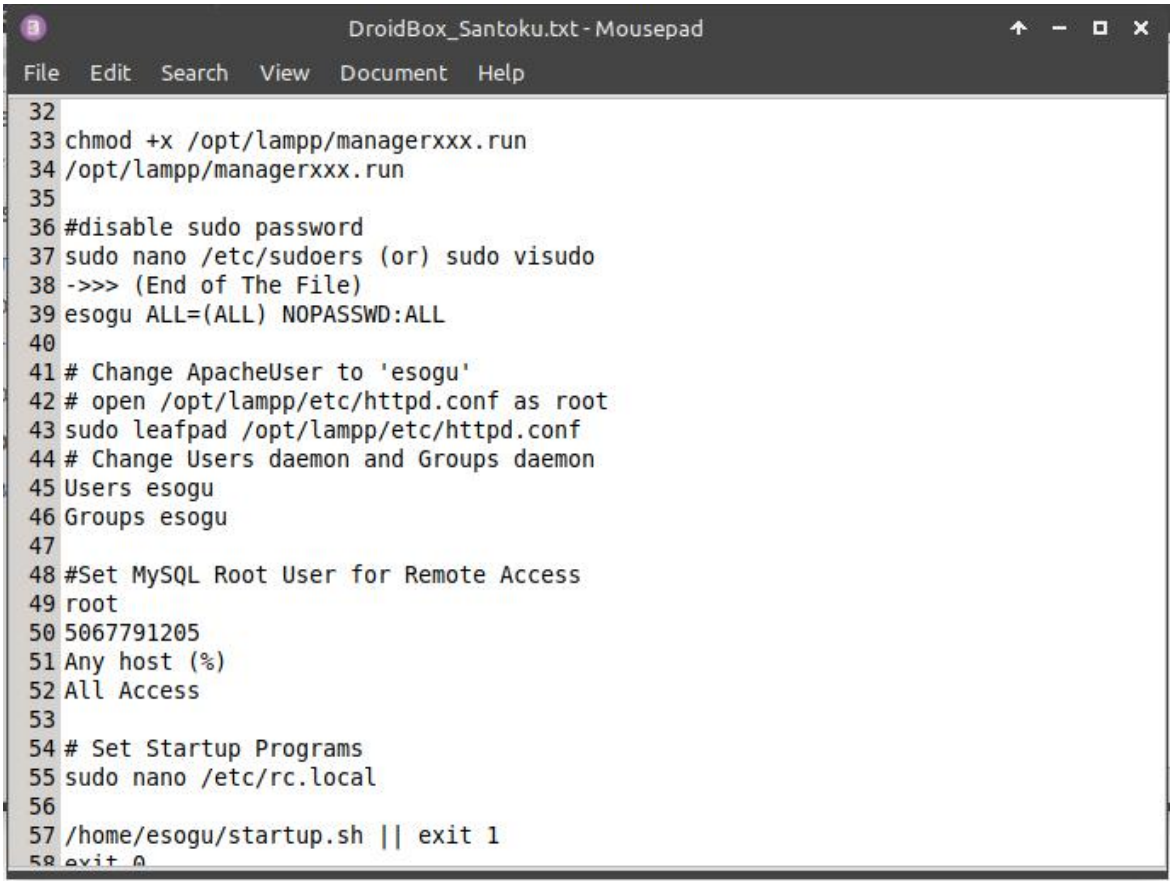
komutunun çalıştırılması yeterlidir. Bu komut sonucunda ana uygulama (JobManager vb.) çalıştırılacaktır.

Sistemi çalıştırmadan önce PATH değişkenlerinin atanmış olması gerekmektedir. Eğer atanmamışsa, atama işlemlerini ve sanal makineyi çalıştıracak şekilde shell script yazılmıştır. Şekil 42' de bu script gösterilmektedir.

```
export PATH=$PATH:/usr/share/android-sdk/sdk/tools
export PATH=$PATH:/usr/share/android-sdk/sdk/platform-tools
cd /home/esogu/Desktop/DroidBox
/home/esogu/Desktop/DroidBox/startemu.sh DRAVD
```

Şekil 42: setupDroidBox.sh Dosyası İçeriği

Sistem 'Santoku Linux' üzerinde çalıştırılmaktadır. Santoku linux, bir çok mobil tersine mühendislik araçlarını, android studio'yu ve android sdk r23.0.2'yi barındırmaktadır. Ubuntu 14.04 tabanlıdır. Kurulum sonrası gerekli paketler ve ayarlar kaynak kodlar ile birlikte verilecektir. Şekil 43' te bu komut ve ayarların bir kısmı gösterilmektedir.



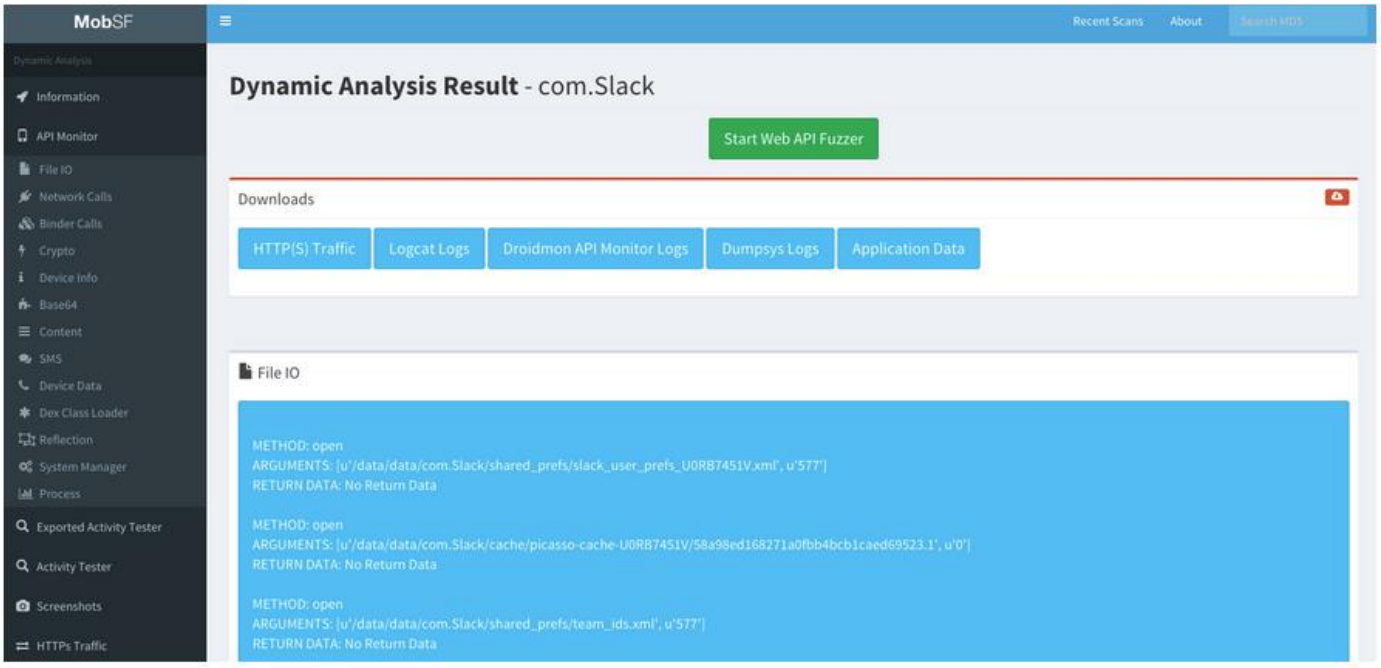
Şekil 143: Santoku Kurulum Sonrası Komutlar ve Ayarlar

E. SONUÇ VE ÖNERİLER

Sistemin çıktılarını diğer analiz sistemleriyle karşılaştıracak olursak:

DynaLog ve Dynaldroid sistemlerinde raporlar son kullanıcıya uygun değildir, kullanıcıların uygulamaların '.apk' uzantılı kurulum dosyalarını sisteme vermeleri gerekmektedir ve bu da son kullanıcıya uygun değildir.

MobSF' e bakıldığında raporları son kullanıcıya uygun değildir ve sisteme '.apk' uzantılı kurulum dosyalarının yüklenmesi gerekmektedir. Örnek bir MobSF analiz raporu Şekil 44' te gösterilmektedir.

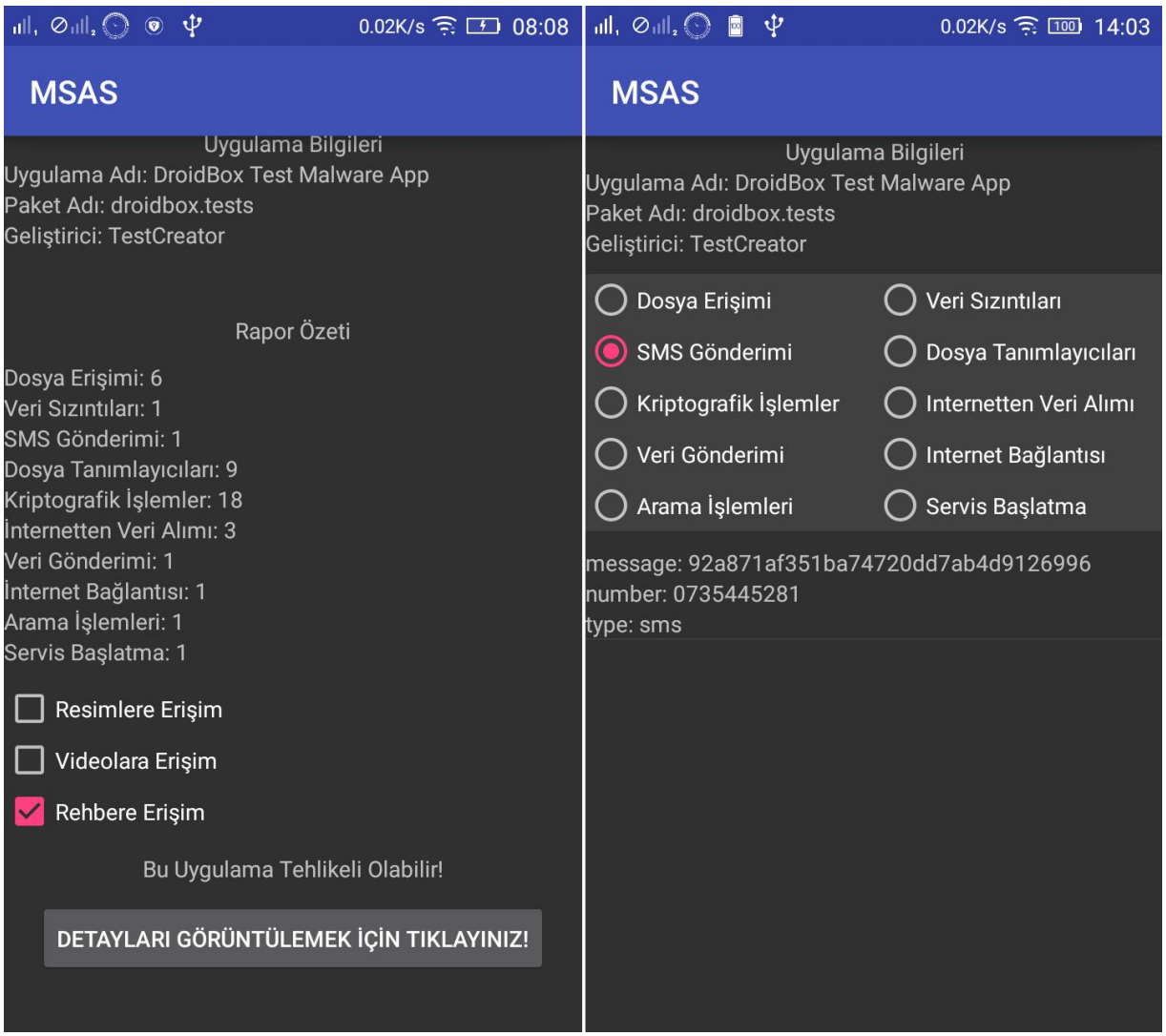


Şekil 44: MobSF Dosya Erişimi Analiz Sonucu [17]

Sanddroid, statik analiz, dinamik analiz ve risk analizi olmak üzere 3 adet analiz methodunu uygulamaktadır. Geliştiriciler için sistemimiz ile kıyaslandığında çok daha gelişmiştir ve raporu daha uygundur fakat son kullanıcı için kullanımı ve raporları uygun değildir. Kullanıcıların sisteme '.apk' uzantılı kurulum dosyalarını yüklemeleri gerekmektedir. Örnek bir Sanddroid analiz sonucunun sınıflandırılması Şekil 45' te gösterilmektedir.

Şekil 45: Sanddroid Analiz Sonucu Sınıflandırılması [18]

Geliştirilen sistemde ise kullanıcılar Google Play Store üzerinden arama yaptırarak uygulamalarını analiz edebilmektedir. Çıktıya bir karar mekanizması eklenerek test edilen uygulamanın rehbere, resimlere ve videolara erişip erişmediği de gösterilmektedir. Buna bağlı olarak uygulamanın tehlikeli olabileceği de bildirilmektedir. Sistemin analiz sonucu Şekil 46' da gösterilmektedir. Geliştirilen sistem, Sanddroid' e göre daha az uygulama analiz sonucu görüntülese de son kullanıcıya daha uygun bir sistemdir.



Şekil 46: MSAS Analiz Raporu Örneği

Sonuç olarak son kullanıcıların diğer analiz sistemlerine göre daha kolay kullanabileceği bir analiz sistemi geliştirilmiştir. Ülkemizde güvenlik alanındaki çalışmalarına katkı sağlanmıştır.

Sistem; statik, dinamik ve görselleştirme ile zararlı yazılım analiz methodlarının birlikte kullanılmasıyla daha da geliştirilebilir. Sistemde dinamik analiz işlemi için sadece 'MainActivitiy' açılmaktadır fakat dinamik analiz methodunun etkili olabilmesi için önce statik analiz methoduyla uygulamadaki tüm aktivite ve butonlar çıkarılıp daha sonra dinamik analiz teknikleriyle tüm eventler tetiklenmelidir.

Statik analiz teknikleri sonucunda uygulamanın dosyalara erişim potansiyeli de ortaya çıkarılabilir. Uygulama izinleri de kullanıcıya sunulabilir.

Analiz sonuçlarının kullanıcıya daha da uygun hale getirilmesi için karar mekanizması geliştirilebilir. Analiz sonuçları sınıflandırılabilir ve zararlı ve zararsız olarak uygulamalar ayrılabilir. Zararlı olan uygulamalar kullanıcı isteğine bağlı olarak, kullanıcının mobil cihazında yüklü ise kaldırılabilir.

KAYNAKLAR

1. YALÇINKAYA, M , ALTINOK, B , GÜRDAL, M , AKDOĞAN, M , KÜÇÜKSİLLE, E . (). Zararlı Yazılım Yayma Aracı Olarak Mobil Uygulamaların Kullanılması: Pokemon Go Örneği. Mehmet Akif Ersoy Üniversitesi Fen Bilimleri Enstitüsü Dergisi, 8 (Özel 1), 88-96. Retrieved from http://mehmetakif.dergipark.gov.tr/makufebed/issue/29469/316717.
2. Snell, B. (2016). Mobile threat report: What's on the horizon for 2016. Intel Security and McAfee, published March, 1.
3. Wood, P., & Haley, K. (2016). Symantec Internet Security Threat Report 2016, Volume 21 Symantec Corp., Mountain View, CA, USA, Tech. Rep. https://www.symantec.com/content/dam/symantec/docs/reports/istr-21-2016-en.pdf [Erişim 08 Mayıs 2017].
4. Wood, P., & Haley, K. (2015). Symantec Internet Security Threat Report 2015, Volume 20 Symantec Corp., Mountain View, CA, USA, Tech. Rep. https://www.symantec.com/content/dam/symantec/docs/security-center/archives/istr-15-april-volume-20-en.pdf [Erişim 08 Mayıs 2017].
5. AvTest. Malware. https://www.av-test.org/en/statistics/malware/ [Erişim 30 Aralık 2017]
6. Mert Sarıca. Android Zararlı Yazılım Analizi. https://www.mertsarica.com/android-zararli-yazilim-analizi/ [Erişim 16 Mayıs 2017].
7. Dalziel, H., & Abraham, A. (2015). Automated Security Analysis of Android and iOS Applications with Mobile Security Framework. Syngress.
8. Elenkov, N. (2014). Android security internals: An in-depth guide to Android's security architecture. No Starch Press.
9. Malik, S., & Khatter, K. (2016). System Call Analysis of Android Malware Families. Indian Journal of Science and Technology, 9(21).
10. Zheng, M., Sun, M., & Lui, J. C. (2014, August). DroidTrace: a ptrace based Android dynamic analysis system with forward execution capability. In Wireless Communications and Mobile Computing Conference (IWCMC), 2014 International (pp. 128-133). IEEE.
11. Alzaylaee, M. K., Yerima, S. Y., & Sezer, S. (2016, June). DynaLog: An automated dynamic analysis framework for characterizing android applications. In Cyber Security And Protection Of Digital Services (Cyber Security), 2016 International Conference On (pp. 1-8). IEEE.
12. Reddy, K. P., Pareek, H., & Patil, M. U. (2015, February). Dynaldroid: A system for automated dynamic analysis of Android applications. In Recent Advances in Electronics & Computer Engineering (RAECE), 2015 National Conference on (pp. 124-129). IEEE.
13. Tchakounté, F., & Dayang, P. (2013). Qualitative Evaluation of Security Tools for Android. International Journal of Science and Technology, 2(11), 754-838.
14. Kabakuş, A. T., Doğru, İ. A., & Çetin, A. (2015). Android kötücül yazılım tespit ve koruma sistemleri. Erciyes Üniversitesi Fen Bilimleri Dergisi, 31, 9-16.
15. KAYABAŞI, G., & DOĞRU, İ. A. Mobil Cihazlarda Zararlı Yazılım Tespitinde Kullanılan Statik Analiz Araçları.
16. Ruan, H., Fu, X., Liu, X., Du, X., & Luo, B. (2017, July). Analyzing Android Application in Real-Time at Kernel Level. In Computer Communication and Networks (ICCCN), 2017 26th International Conference on (pp. 1-9). IEEE.
17. M. (2017, October 16). Mobile-Security-Framework-MobSF. Retrieved November 05, 2017, from https://github.com/MobSF/Mobile-Security-Framework-MobSF
18. Chandan Kumar. (26 Aralık 2016). 7 Mobile App Scanner to Find Security Vulnerabilities. https://geekflare.com/mobile-app-security-scanner/ [Erişim 29 Aralık 2017]