



AÇIK KAYNAKLI YAZILIM UYGULAMALARI

141180056 RENEZ ERİM ÖZTÜRK
141180061 FURKAN ŞOLPAN

FİNAL RAPORU

GAZİ ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ

BM496 BİLGİSAYAR PROJESİ II

HAZİRAN 2018
ANKARA

SECURITY CHECKER

**RENEZ ERİM ÖZTÜRK
FURKAN ŞOLPAN**

PROJE DANIŞMANI: ÖĞR.GÖR.DR. OKTAY YILDIZ

ÖZET

Günümüzde mobil cihazların sayısı gittikçe artmaktadır. Mobil cihazların kullanımının artmasıyla beraber, mobil cihazlar için kullanılmak üzere mobil uygulamaların sayısı da artış göstermektedir. Uygulamalar içerisinde kullanıcının telefonunun kontrolünü ele geçirmekten, hassas bilgilerini çalmaya kadar birçok farklı türde kötücül yazılım bulunmaktadır. Bir Android telefona Google Play Store ya da farklı bir uygulama mağazası aracılığıyla uygulamalar yüklenebilmektedir. Bu çalışmada, uygulama mağazalarından kullanıcıların daha rahat bir şekilde yararlanabilmesi için Security Checker isimli uygulama geliştirilmiştir.

Bu uygulama, makine öğrenmesi yöntemlerini kullanarak kötücül uygulama tespiti yapabilmektedir. Makine öğrenmesi için kullanılan Android uygulama veri setinde uygulamalardan çıkarılan, 14 nitelik grubuna ayrılmış bulunan toplamda yaklaşık 2 milyon adet nitelik bulunmaktadır. Bu nitelik gruplarından kullanılan apicall ve izinler nitelikleri algoritmamızda kullanılmak üzere ön işlemlerden geçirilerek 138 adet apicall ve 96 adet izin niteliği seçilmiştir. Makine öğrenmesi algoritmalarında kullanılan toplam nitelik 234 adettir. Daha sonra nitelik seçimi ile nitelik sayısı 79'a indirilmiştir.

Veri setinde 44786 normal, 8958 kötücül uygulama bulunmaktadır. Veri setinden apicall ve izin niteliklerini bulundurmayan örnekler ve aynı olan örnekler atıldıktan sonra veri setinde 11490 normal, 3848 kötücül uygulama olmak üzere toplam 15338 uygulama örneği kalmıştır. Makine öğrenmesi algoritmalarında veri seti ile elde edilen doğruluk oranları sırasıyla NaiveBayes %71,07, Knn %87,26, RandomForest %91,22, DecisionTree %85,37. Tespit sisteminde kullanılmak üzere bu doğruluk oranlarına bakılarak RandomForest algoritması seçilmiştir.

İÇİNDEKİLER

1. GİRİŞ	4
1.1. Projenin Amacı.....	4
2. Android Güvenlik Konusunda Yapılan Çalışmalar	5
3. Kullanılacak Araç ve Yöntemler	7
3.1. Android Studio	7
3.2. Sunucu	8
3.3. Veri Seti.....	9
3.3.1. Veri Toplama İşlemi.....	9
3.4. Nitelik Seçimi.....	10
3.5. Makine Öğrenmesi	12
3.5.1. Naive Bayes.....	12
3.5.2. KNN (K Nearest Neighborhood)	13
3.5.3. Random Forest	14
3.5.4. Decision Tree	15
3.6. Özel Kimlik Kod	16
4. Gerçekleştirilen Uygulama.....	17
4.1. Arayüz Tasarımı	17
4.2. CPU Tüketimi Modülü.....	22
5. Sonuç ve Değerlendirme	23
KAYNAKLAR.....	24

1. GİRİŞ

Gerçekleştirilen Security Checker adlı uygulama, yeni kurulan uygulamaları tespit eden bir servisten ve android telefonda bulunan tüm uygulamaları tarayan bir modülden oluşmaktadır. Bu uygulama tarafından ele alınan bir uygulamanın çıkartılan özel kimlik kodu ve izin listesi JSON formatında Spring ile kurulan sunucuya gönderilerek kötücül uygulama olup olmama riski öğrenilmektedir. Sunucu tarafında apk uzantılı dosyalar üzerinden tespit işlemi yapılarak sonuçlar, çıkartılan özel kimlik kodu ile birlikte kayıt altına alınmaktadır. Uygulama üzerinde kötücül olarak tespit edilmiş uygulamalar kullanıcı tarafından silinse de silinmese de bir liste içinde tutulmaktadır. Kullanıcı kötücül olmadığını bildiği bir uygulamayı bu liste üzerinden bize bildirebilmektedir. Sunucu üzerinde tutulan kayıtlar ve bildirilen yanlış tespitler kullanılarak veri seti ilerleyen zamanlarda güncellenebilecektir.

1.1. Projenin Amacı

Android telefonlarda çalışan uygulamalar, android uygulama mağazalarından elde edilmektedir. Google Play Store bu mağazaların başında gelmektedir. Play Store'dan indirilen uygulamalar Google Play Protect sistemi ile daha mağazaya yüklenirken taranmaktadır. Ancak diğer android uygulama mağazalarında böyle bir sistem bulunmamaktadır. Bu mağazalar yüksek miktarda kötücül uygulamalar içermektedir. Bu uygulamalar yüklendiği telefonu ele geçirmekten, telefon üzerindeki özel verileri (telefon rehberi, banka numaraları, resimler ...vb.) çalmaya kadar birçok tipte bulunmaktadır. Gerçekleştirilen uygulama, herhangi bir uygulama mağazasından kurulan uygulamaların izin listelerini ve apicall sayılarını makine öğrenmesi teknikleri ile inceleyerek, kurulan uygulamanın kötücül uygulama olma riski taşıyıp taşımadığını tespit etmektedir. Bu uygulama sayesinde, kullanıcıların Google Play Store ve diğer android uygulama mağazalarından güvenli bir şekilde yararlanabilmeleri amaçlanmaktadır.

2. Android Güvenlik Konusunda Yapılan Çalışmalar

Sanz ve arkadaşları 2014 yılında yapmış oldukları Anomaly Detection Using String Analysis for Android Malware Detection adlı çalışmada, düşük hata oranına sahip kötücül (malware) yazılım tespit çalışması yapmışlardır. Anomali tespit sistemleri yüksek hata oranlarına (özellikle False Positive oranları) sahiptir. Bunun aksine bu çalışmada yapılan deneysel testlerde düşük hata oranları ile karşılaşmıştır. Bu önerilen yöntem malware örneklerinin toplanması zorunluluğunu azaltarak kolaylık sağlamaktadır. Çünkü normal uygulamalar train için kullanılmaktadır. Bu yöntemde uygulamaların string datalarından çıkarılan özellikler kullanıldığından uygulamanın kurulması önlenmektedir. Bunun yanı sıra malware uygulamalar kurulduktan sonra malware kodları internet üzerinden de indirebildiği için ancak dinamik bir izleme yöntemi bu uygulamaları tespit edebilmektedir [1].

Saracino ve arkadaşları 2016 yılında yapmış oldukları MADAM: Effective and Efficient Behavior-based Android Malware Detection and Prevention adlı çalışmada, malware uygulamalar davranışsal yönlerine bakılarak küçük gruplara ayrılmıştır. Önerilen MADAM adlı sistem uygulamalardan 4 farklı seviyede nitelik analiz etmektedir. Bu seviyeler kernel, application, user ve package'dir. MADAM düşük seviye işlem gücü ve pil tüketimi ile 4 farklı seviyede davranışsal bir analiz yapan host-based malware tespit sistemidir [2].

Thomas Blasing ve arkadaşları 2010 yılında An Android Application Sandbox System for Suspicious Software Detection adlı çalışmada, Android Application Sandbox (AASandbox) kullanılarak hem statik hem de dinamik analiz yaparak android'te tehlikeli olabilecek uygulamaları tespit etmektedir. Statik analiz uygulamayı yüklemeyen kötü amaçlı yazılım kalıplarını tarayarak tespit etmeye çalışır. Tespit etmek için şifre çözümü, virüs kalıplarını eşleştirme, sistem çağrılarını analiz eder. Önceden var olan kalıpların tespit edilmesi için hızlı ve kolay bir yöntemdir. Fakat yeni çıkan kötücül yazılımları tespit edememektedir. Dinamik analiz tamamen izole edilmiş bir ortamda

uygulamanın yaptığı etkileşimleri inceler ve bunları kaydeder. Çalıştığı süre boyunca yapılan dosya değişikliklerini, internet kullanımlarını, yaptığı işlemler, sistem çağrılarını inceleyerek dinamik analiz gerçekleştirilir. Hem sandbox hem de tespit etme algoritmaları bulut üzerinden haberleşebilirler. Google Play Store benzer bir uygulama indirme mağazasında kötücül yazılımlar paylaşılarak tespiti sağlanır [3].

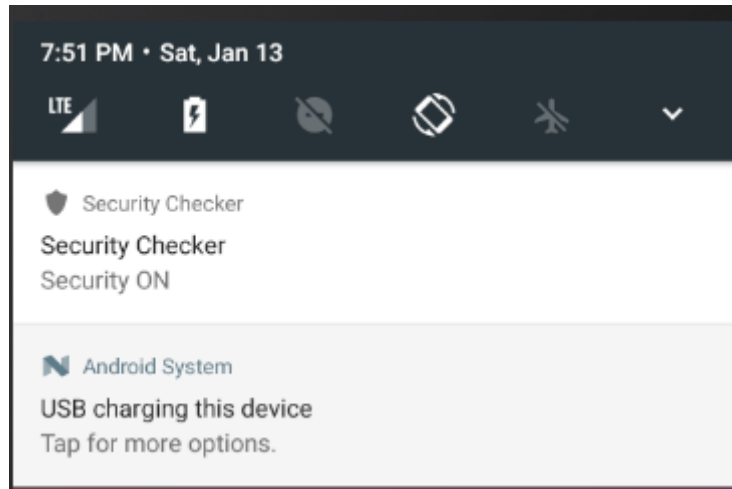
Takamasa Isohara ve arkadaşları 2011 yılında Kernel-based Behavior Analysis for Android Malware Detection adlı çalışmada, davranışsal bir malware tespiti yapmışlardır. Uygulamaların hatalarını ve güvenliğini sağlamak için Dalvik sanal makinesindeki logcat uygulaması pek çok geliştirici tarafından uygulanmaktadır. Fakat yeteri kadar güvenli olmadığından dolayı tüm faaliyetleri, davranışları ve güvenlik anlamında tam koruma sağlanabilmesi için çekirdek tabanlı davranış analizi uygulanmalıdır. Bu çalışmada çekirdek tabanlı analiz ile uygulamaların kötücül olmasına karar vermek üzerine yapılmıştır. Bu yapılan sistem Linux işletim sistemi üzerinde bir adet log toplayıcı ve bu logları analiz eden iki adet katmanı vardır. Log toplayıcı çekirdek katmanında günlük olan olayları, sistem çağrıları kaydeder. Çekirdek düzeyde günlük olayları not etmesinden dolayı büyük miktarda veri elde edilir. Aynı zamanda Android sistemde çalışan tüm uygulamaları etkinliklerini de içermektedir. Analiz kısmında ise bütün uygulamaların etkinliklerini bir işlem ağacına alır ve zaman zaman zararlı bir uygulama olup olmadığını algılamak adına düzenli aralıklarla uygulamanın imzasını eşleştirme yapar. Aynı zamanda aygıtta depolanan bazı verileri otomatik olarak toplar ve kişisel bilgi sızıntısını algılamak adına bir imza haline dönüştürür. Bu yapılan prototip sistemde 230 adet uygulama değerlendirilmiştir. Çalışma sonunda kötücül özellik gösteren uygulamalar tespit edilmiştir. Bu uygulamalarda 37 uygulama bilgileri sızdırmak üzere, 14 uygulama yabancı kodlar çalıştırmaya çalışan ve 13 adette kullanıcıya zarar vermeye çalışan uygulamalar bulunmuştur [4].

Bu çalışmalarla birlikte kötücül yazılım tespiti hakkında yapılan çalışmalar [5-15] incelenmiştir.

3. Kullanılacak Araç ve Yöntemler

3.1. Android Studio

Gerçekleştirilen uygulama Android Studio üzerinde geliştirilmiştir. Android 5.0 ve üzerinde çalışmaktadır. Sade ve pratik bir arayüz hedeflenmiştir. Bu uygulama herhangi bir uygulama kurulduğunda uyanacak şekilde arka planda çalışmaktadır. Arka planda çalışması için bir Foreground Service kullanılmıştır. Ön yüzde kullanıcıya görünerek işlem yapan bu servis Android İşletim Sistemi tarafından kaynak yetersizliği ile karşılaşıldığında sonlandırılmaz. Foreground servisin çalışması Şekil 3.1.1. de görüldüğü gibidir.



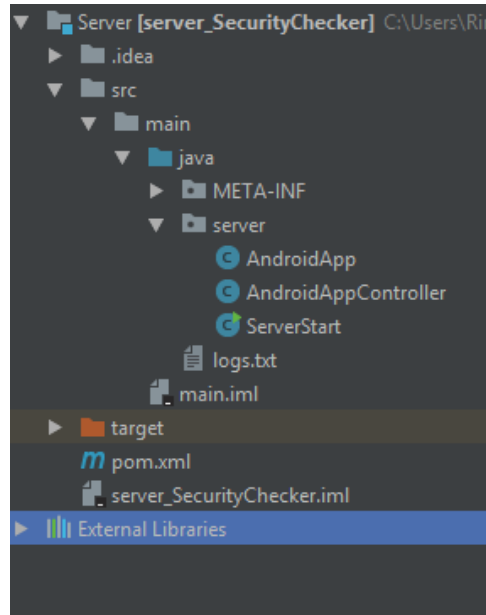
Şekil 3.1.1. Çalışır durumdaki Foreground Service

Yeni kurulan uygulamanın veya tüm cihaz taranırken diğer tüm uygulamaların özel kimlik kodu çıkarılmıştır. Çıkarılan özel kimlik kodu sunucuya JSON formatında gönderilmiştir. Veri AsyncTask kullanarak gönderilmiştir. Bu uygulama arayüzünün sunucudan cevap beklenirken kitlenmemesini sağlamaktadır. Bu uygulama, kötücül olma potansiyeline sahip uygulamaları kullanıcıya bildirerek silmek için izin istemektedir. Kullanıcı kötücül uygulamayı silmese de riskli uygulama Blacklist isimli listeye eklenmektedir. Kullanıcı kötücül olmadığını bildiği yanlış tespitleri bu liste üzerinden sunucuya bildirebilmektedir.

3.2. Sunucu

Gerçekleştirilen uygulamanın elde ettiği yeni verileri kaydetmesi için bir sunucu geliştirilmiştir. Sunucu için işletim sistemi bağımsız bir şekilde Java dilinde yazılmış açık kaynaklı bir framework olan Spring kullanılmıştır. Spring localhost:8080 üzerinde bir yerel sunucu kurmaktadır. VBox'ta Kali Linux üzerinde kurulu olan sunucuya bilgisayarın 90 portu yönlendirilerek uygulama testleri gerçekleştirilmiştir. Testler sırasında Android emulator Windows üzerinde çalıştırılmıştır ve bilgisayarın 90 portuna istekler atılmıştır.

Bu sunucuda veri seti ile önceden eğitilen tespit sistemi bulunmaktadır. Android uygulama tarafından cihaz üzerinde yüklü olan uygulamaların özel kimlik kodu çıkarılarak sunucuya gönderilmektedir. Gelen özel kimlik kodu sunucu tarafından tespit sistemi ile önceden taranmış uygulamaların kimlik kodları arasında aratılır ve uygulamanın kötücül veya normal olduğu geri dönüş olarak gönderilmektedir. Sunucu tarafında uygulamadan yanlış tespit olarak bildirilen uygulamaların kaydı saklanmaktadır. Gelecek günlerde bu kayıt dosyaları kullanılarak veri seti güncellenebilecektir. Şekil 3.2.1.'de sunucunun sınıf hiyerarşisi verilmiştir.



Şekil 3.2.1. Spring Framework Sınıf Hiyerarşisi

3.3. Veri Seti

Bu uygulama, makine öğrenmesi yöntemleri kullanarak zararlı yazılım tespiti yapabilmektedir. Makine öğrenmesi için kullanılan Android uygulama veri seti Dr. Wei Wang'ın internet sitesinde 2015 yılında yayınlanmıştır [16]. Veri seti 46891 normal uygulama ve 9662 kötücül uygulama içermektedir. Bu veri setinde Android uygulamalardan çıkarılan, 14 nitelik grubuna ayrılmış bulunan toplamda yaklaşık 2 milyon adet nitelik bulunmaktadır. Bu nitelik gruplarından kullanılan apicall ve izin nitelikleri algoritmamızda kullanılmak üzere ön işlemlerden geçirilerek 138 adet apicall ve 96 adet izin niteliği seçilmiştir. Makine öğrenmesi algoritmalarında kullanılan toplam nitelik 234 adettir.

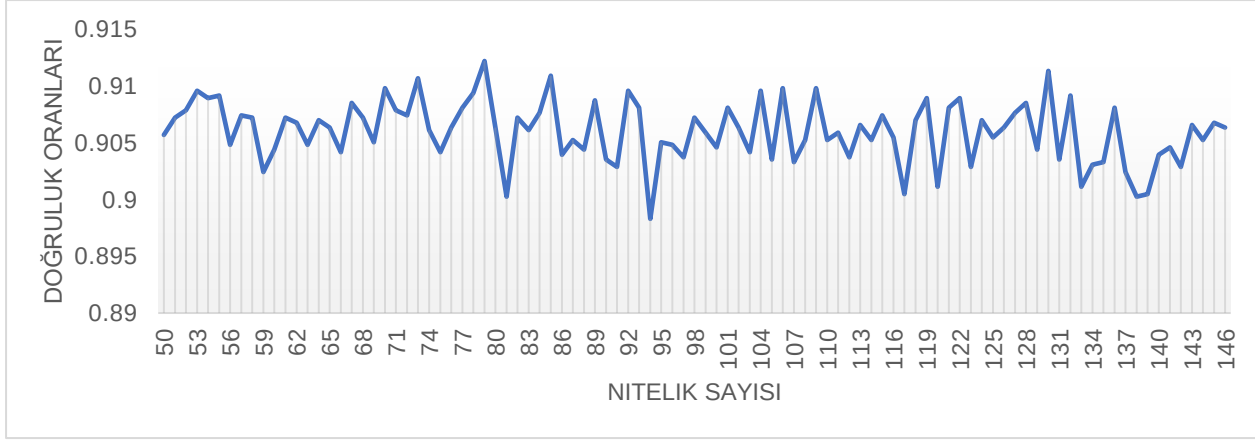
Veri setinden apicall ve izin niteliklerini bulundurmeyen örnekler ve yinelenen örnekler atıldıktan sonra veri setinde 11490 normal, 3848 kötücül uygulama olmak üzere toplam 15338 uygulama örneği kalmıştır. Bu veri setinin %70'i train, %30'u test için kullanılmıştır.

3.3.1. Veri Toplama İşlemi

İncelenen uygulamalardan veri setine uygun bir şekilde veri çıkartılması işlemi birkaç farklı adımdan oluşmaktadır. İlk adımda apktool isimli uygulamanın 2.3.3 versiyonu kullanılarak apk uzantılı uygulama dosyaları tersine mühendislik işlemi yapılarak açılmaktadır. Açılan bu dosyada uygulamanın AndroidManifest.xml dosyası ve classes.dex dosyası bulunmaktadır. classes.dex dosyası ayrıca decode (baksmali) işlemine alınarak smali dosyalarına parçalanmaktadır. Smali, java dosyaları ile compile edilmiş android dosyaları olan dex (Dalvik Executable) dosyaları arasında bir dosya tipidir. Smali dosyalarından çok hızlı bir şekilde kod incelemesi yapılabilir.

İkinci adımda AndroidManifest.xml dosyalarından incelenen uygulamanın Paket adı ve elimizdeki veri setine uygun olarak kullandığı izinler çekilmektedir. Paket adı daha sonra

yüksek doğruluk oranı yakalanan en iyi skor elde edilen 79 nitelik kullanılmıştır. Tablo 3.4.2.'de seçilen 79 nitelik verilmiştir.



Tablo 3.4.1. Nitelik sayısına göre elde edilen doğruluk oranlarının karşılaştırılması

Nitelik Adı		
Landroid/accounts	Landroid/view/accessibility	CALL_PHONE
Landroid/bluetooth	Landroid/view/inputmethod	CAMERA
Landroid/hardware	Landroid/webkit	CHANGE_NETWORK_STATE
Landroid/inputmethodservice	Landroid/widget	CHANGE_WIFI_MULTICAST_STATE
Landroid/location	Lcom/android/internal/app	CHANGE_WIFI_STATE
Landroid/media	Lcom/android/internal/telephony	GET_ACCOUNTS
Landroid/net	Lcom/android/music	GET_TASKS
Landroid/net/wifi	Lcom/android/soundrecorder	MANAGE_ACCOUNTS
Landroid/net/wifi/p2p	Lcom/cooliris/app	MODIFY_AUDIO_SETTINGS
Landroid/nfc	Lcom/cooliris/media	NFC
Landroid/opengl	Lcom/cooliris/picasa	READ_CALENDAR
Landroid/os	Lcom/cooliris/wallpaper	READ_CONTACTS
Landroid/print	Lcom/coremedia/iso	READ_HISTORY_BOOKMARKS
Landroid/provider	Lcom/google/common/io	READ_PHONE_STATE
Landroid/speech/tts	Ljava/net	READ_PROFILE
Landroid/support/v4/content	Lorg/apache/http/impl/client	READ_SOCIAL_STREAM
Landroid/support/v4/net	ACCESS_COARSE_LOCATION	READ_SYNC_SETTINGS
Landroid/support/v4/print	ACCESS_FINE_LOCATION	READ_USER_DICTIONARY
Landroid/support/v4/view	ACCESS_LOCATION_EXTRA_COMMANDS	RECORD_AUDIO
Landroid/support/v4/view/accessibility	ACCESS_MOCK_LOCATION	REORDER_TASKS
Landroid/support/v4/widget	ACCESS_WIFI_STATE	SEND_SMS
Landroid/telephony	AUTHENTICATE_ACCOUNTS	USE_CREDENTIALS
Landroid/telephony/gsm	BATTERY_STATS	VIBRATE
Landroid/text/format	BLUETOOTH	WRITE_CONTACTS
Landroid/util	BLUETOOTH_ADMIN	WRITE_EXTERNAL_STORAGE
Landroid/view	BROADCAST_STICKY	WRITE_SETTINGS
		WRITE_SYNC_SETTINGS

Tablo 3.4.2. Seçilen 79 nitelik

3.5. Makine Öğrenmesi

Kötücül uygulamaların tespitinde Statik Analiz yöntemi kullanılmıştır [9]. Android uygulamaların apicall sayıları ve izin listeleri çıkartılarak kullanılan veri setine uygun şekilde makine öğrenmesi modeli üzerinde tahmin edilmiştir.

Bu veri seti kullanılarak makine öğrenmesi algoritmaları üzerinde denemeler yapılmıştır. Denemelerde NaiveBayes, KNN, RandomForest ve DecisionTree algoritmaları kullanılmıştır. Bu veri setindeki 15338 örneğin %70'lik kısmı modeli eğitmek amacıyla kalan %30'luk kısmını ise test etmek amacıyla kullanıldı. Makine öğrenmesi için Python dili ve Python üzerinde çalışan scikit-learn kütüphanesi kullanılmıştır. Bu çalışmada scikit-learn kullanılarak makine öğrenmesi sonuçları karşılaştırılmıştır. Daha sonra Nitelik Seçimi yapılarak aynı makine öğrenmesi algoritmaları eski sonuçlar ile karşılaştırılmıştır.

3.5.1. Naive Bayes

NaiveBayes algoritması Thomas Bayes tarafından bulunan olasılıkçı matematiksel bir yaklaşımdır. Bir olayın olasılık ilkelerine göre hesaplanarak gelen verinin kategorisini tespit etmeyi amaçlamaktadır. Naive bayes için her parametrenin istatistik açıdan birbirinden bağımsız olması gerekmektedir. Daha önceden verilen verilere göre olasılık hesabı hesaplayarak sınıflarına ayırmaktadır. Gelen data üzerinden oluşturulmuş olasılık hesaplarına göre işlemler yapılarak sınıfı tahmin edilir. Naive Bayes modelleri için parametre tahmini, maksimum olasılık yöntemini kullanır. Şekil 3.5.1.1. de Naive Bayes modelinin genel formülü verilmiştir.

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Şekil 3.5.1.1. Naive Bayes genel formülü

$P(A|B)$; B olayı gerçekleştiği durumda A olayının meydana gelme olasılığıdır

$P(B|A)$; A olayı gerçekleştiği durumda B olayının meydana gelme olasılığıdır

$P(A)$ ve $P(B)$; A ve B olaylarının önsel olasılıklarıdır

A ve B'nin önsel olasılığı bayes teoremine öznellik katar. A olayı hakkında bir bilgi edinmeden A olasılığı hakkında bilgi sahibi olunabilir. Bunun yanında $P(B|A)$ ardıl olasılıktır, veriler toplanmaya başladıktan sonra A'nın gerçekleşmiş olduğu durumlarda B'nin gerçekleşme olasılığı hakkında tahmin yürütülebilir. Naive Bayes en büyük avantajlarından birisi sınıflandırma yapmak için küçük veri setinin yeterli olmasıdır. Bu veri set ile sınıflandırma ve tahmin işlemlerini büyük doğrulukta yapabilmektedir.

3.5.2. KNN (K Nearest Neighborhood)

KNN, sınıflandırma ve regression için kullanılan bir yöntemdir. KNN en yakın komşuyu bulmayı amaçlar. KNN sınıflandırması verilen verinin önceden öğretilmiş olan verilere olan uzakları hesaplanarak en yakın olan komşular bulur. Komşuların çoğunluk olduğu sınıf gönderilen verinin sonucunu belirler. KNN regression ise en yakın komşu değerlerinin ortalamasını alarak gönderilen verinin sınıfını belirleme işlemidir.

KNN uzaklık hesaplarken 3 farklı uzaklık formülü kullanmaktadır. Bunlar Öklid, Manhattan ve Gauss teoremidir. Şekil 3.5.2.1. de uzaklık hesaplama formülleri verilmiştir.

Euclidean	$\sqrt{\sum_{i=1}^k (x_i - y_i)^2}$
Manhattan	$\sum_{i=1}^k x_i - y_i $
Minkowski	$\left(\sum_{i=1}^k (x_i - y_i)^q \right)^{1/q}$

Şekil 3.5.2.1. Knn için uzaklık bulma algoritmaları

Her komşuya olan uzaklıkları hesaplama işlemi uzun sürmesinden dolayı uzun zaman bir yöntemdir. Sınıflandırma türleri arasında en yakın ve doğru sonucu vermektedir. Büyük çapta veriler için uzun sürmesinden dolayı küçük veri setler için tercih edilmelidir. Gürültülü veri olmasına en dirençli olan algoritmadır.

3.5.3. Random Forest

Random Forest sınıflandırma ve regression kullanılan makine öğrenmesi yöntemidir. Random Forest eğitim zamanında birden fazla karar ağacı oluşturarak verilen verilere uygun model üretmeyi amaçlamaktadır. Oluşturulan ağaçların tek tek ortalama tahminlerini almaktadır. Bu ortalama tahminlere göre en uygun olan ağacı seçmektedir.

Random Forest anlaşılması ve yorumlaması oldukça basittir. Ağacın ön işlemesi oluşturma işlemi hızlı bir şekilde gerçekleştirilir. Basit bir ağaç oluşturarak bunun üzerinden ilerleme yapar. Çok az veri seti kullanılarak başarı sonuçlar elde edebilmek mümkündür. Hem sayısal hem de sınıfsal veri türlerini işlemek için kullanılabilir. Bazı makine öğrenmesi türleri sınıfsal veriler üzerinde işlem yapılamazken random forest

ağaçları bu türler için de kullanılabilir. Hesaplama işlemleri kısa sürmektedir. Hızlı sonuç üretmektedir. Çok miktarda veri olsa dahi veriyi işleme ve sonuç üretme süresi oldukça kısadır.

Öğrenme sırasında yapılacak bir hata ile tüm ağaç farklı yönlere gidebilmektedir. Bunun için verilen veri setinin çok daha dikkatli seçilmesi gerekir. Aksi taktirde yanlış eğitilmesinden dolayı yanlış ve farklı sonuçlar üretilebilir, farklı modeller ile karşılaştırılabilir.

3.5.4. Decision Tree

Decision Tree belirli parametrelere göre sürekli olarak bölünen denetimli makine öğrenmesi türüdür. Ağacı oluşturan karar düğümleri ve yapraklar bulunur. Yapraklar kararları ve sonuçları oluşturur. Karar düğümleri verilerin bölünme yaptığı bölümlerdir. Sınıflandırma probleminde yaygın kullanılan bir yöntemdir.

Decision Tree oluştururken iki aşamadan geçer. İlk aşama Ağaç oluşturma aşamasıdır. Ağaç oluşturma aşamasında verilen örneklerle göre niteliklere bağlı yinelemeli olarak ağaç parçalanarak bölünür. Fazla niteliğe sahip veri kümesi verilir ise ağaç fazla yaprak ve karar düğümleri ekler. İkinci aşama ağaç budamadır. Ağaç oluşturma aşamasında oluşan gürültüye sahip veriler ve test aşamasında ortaya çıkan hataları tespit ederek dalları silme işlemi yapar. Bu işlem ağaçtaki başarı oranını arttırmaktadır. Erken budama ağaç büyümeden tahmin yaparak budama yapar. Geç budama ağaç tamamlandıktan sonra budama işlemi yapılır. Geç budama bütün veride işlem yaptığı için daha doğru bir yöntemdir. Ağaç oluşturma işlemi yenilemelidir. Bütün ağaç tek bir düğümle başlar. Örneklerin sayısı arttıkça yapraklara ayrılır. Aynı sınıfa ait üyeler, aynı düğüm yaprağında sonlanır ve aynı isim etiketine sahip olurlar. Aynı sınıfa ait olmayan üyeler bölünebilecek en iyi özellik nitelik seçilerek yapraklara ayrılır. Eğer verilen veriler birbirlerine aşırı bir şekilde uyum sağlıyor ise ağaç veriyi ezberleyerek yanlış sonuçlar doğurabilmektedir.

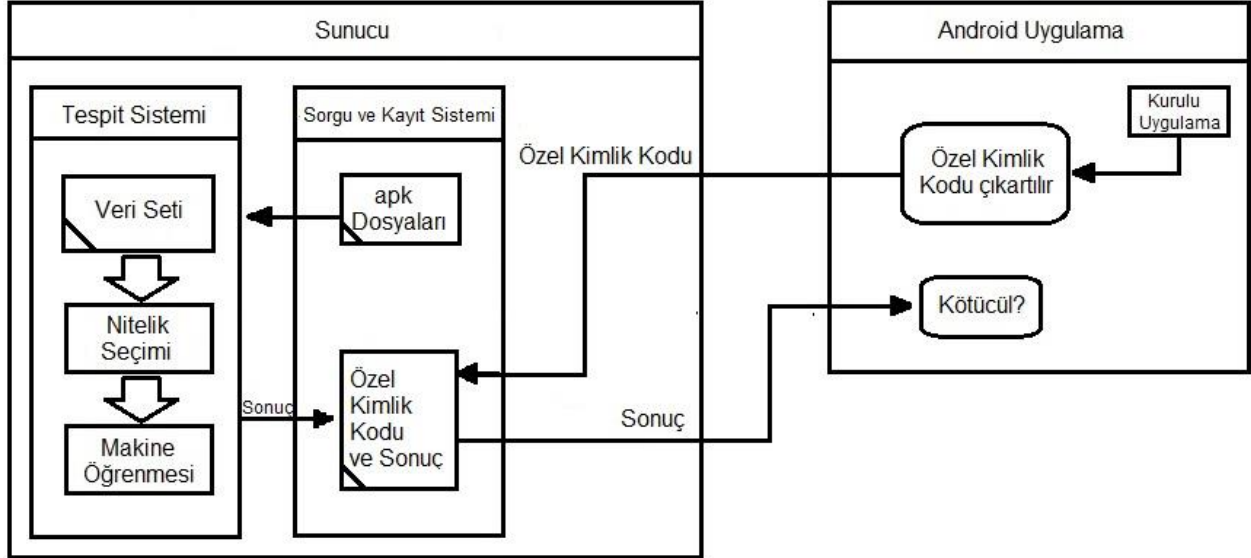
3.6. Özel Kimlik Kod

Android cihazlarda mağazalardan indirilebilecek birçok uygulama bulunmaktadır. Bu uygulamaları birbirinden ayırmak için uygulamaların sahip olduğu özel kimlik kodu üretilmektedir. Bu özel kimlik kod her uygulama için farklı değerden oluşur. Özel kimlik kodu üretmek için uygulama sertifikasından ve paket isminden yararlanılır.

Android cihazlarda uygulamaların özel kimlik kodunu üretmek için cihaz içerisindeki paketlenmiş uygulamaların listesi çıkartılır. Bu paketlenmiş uygulamalardan özel kimlik kodu elde edilmek istenilen uygulamanın imzalanmış olan X509 sertifikası elde edilir. X509 sertifikası üzerinden uygulamanın sahip olduğu sertifika kodu çıkartılır. Sertifika kodu üzerine uygulamanın paket ismi eklenir ve base64 encode haline getirilir. Oluşturulan base64 kod zararlı yazılım olup olmadığını öğrenilmek üzere sunucuya gönderilir.

Sunucu tarafında uygulamaların özel kimlik kodunu üretmek için uygulamanın APK dosyası decode edilir ve içerisinde uygulamaya ait sertifikanın bulunduğu RSA uzantılı dosya bulunur. Bu dosya openssl kullanılarak açılır ve uygulamanın sahip olduğu sertifikanın base64 encode edilmiş haline ulaşılır. Sertifika kodunun üzerine uygulamanın paket ismi eklenerek base64 haline getirilir. Uygulamanın sahip olduğu özel kimlik kodu oluşturulur. Sunucuda uygulamanın izin listesi ve apicall sayıları kullanılarak yapılan analiz sonucu, özel kimlik kodu ile birlikte kaydedilir. Sunucu Android cihazlardan gelen özel kimlik kodunu, kayıtlı olan özel kimlik kodları arasında arar. Kayıtlı olan analiz sonucunu (kötücül yazılım olup olmadığı bilgisini) android cihaza döndürür.

4. Gerçekleştirilen Uygulama



Şekil 4.1. Veri akış diyagramı

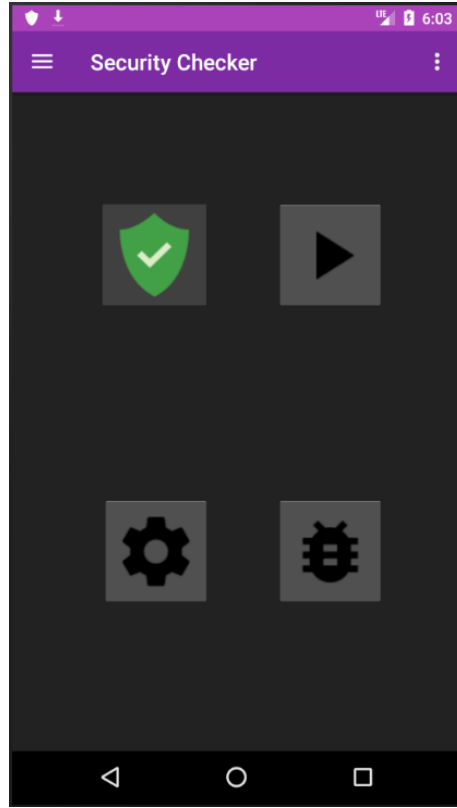
Şekil 4.1.' de gerçekleştirilen uygulamanın veri akış diyagramı verilmiştir. Diyagramda görüldüğü üzere kurulu olan bir uygulamadan özel kimlik kodu çıkartılarak sunucuya gönderilmektedir. Sunucuda daha önceden indirilerek Tespit Sistemine sokulmuş apk dosyalarının özel kimlik kodu ve sonuçları tutulmaktadır. Uygulamadan gelen bir isteğe karşı özel kimlik koduna karşılık gelen sonuç uygulamaya gönderilir.

4.1. Arayüz Tasarımı

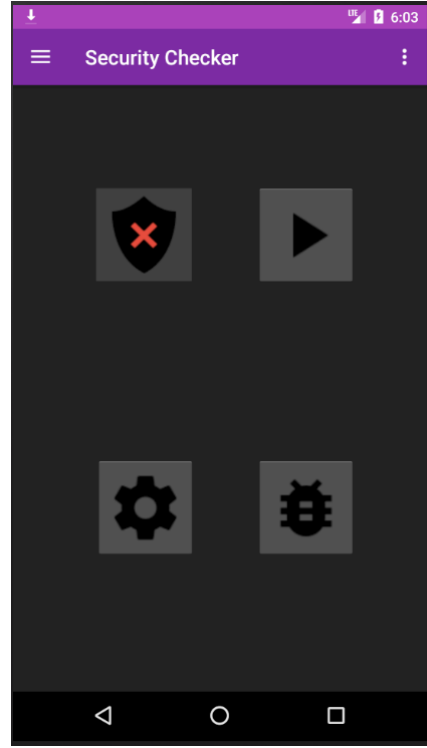
Kullanıcı arayüzü tasarlanırken pratik olması ön plana konulmuştur. Sade olan uygulama sayfaları yaratılmıştır. Şekil 4.1.1.'de uygulamanın ana sayfası görülmektedir. Uygulamanın kullanım kolaylığı sağlaması amacıyla bir NavigationDrawer da vardır. Şekil 4.1.3.'te NavigationDrawer görülmektedir. Şekil 4.1.4.'te ayarlar sayfası görülmektedir.

Yeni kurulan uygulamaları taramak için konulan ForegroundService' i Şekil 3.1.1.'de çalışır durumda görülmektedir. Bu servis Şekil 4.1.1. ve 4.1.2.'de görüldüğü şekilde bir buton ile açılıp kapanmaktadır. Servisi açmak ve kapamak sadece bir tuşla mümkünken foreground service olduğu için uygulama kapalıyken de çalışmaya devam etmektedir.

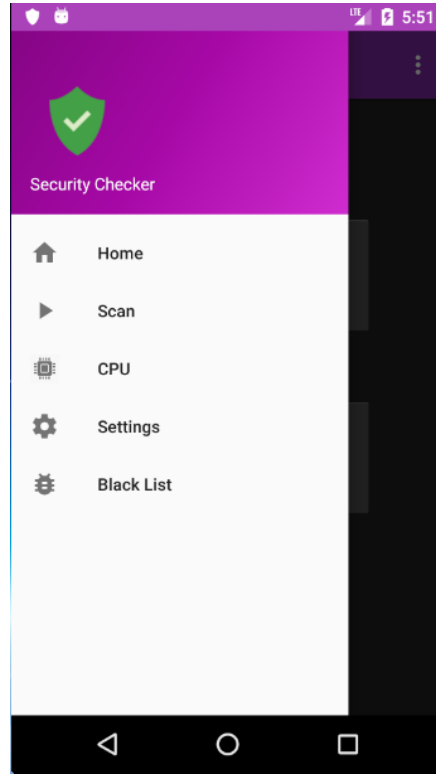
Şekil 4.1.5.'da kötücül bir uygulama bulunduğunda kullanıcıya verilen uyarı mesajı görülmektedir. Kullanıcı isterse bu uyarıyı görmezden gelebilir. Her halükârda Şekil 4.1.6.'de görülen Blacklist adlı listeye bu bulunan uygulamalar eklenmektedir. Şekil 4.1.7.'de yanlış bir tespitin nasıl sunucuya bildirildiği gösterilmiştir.



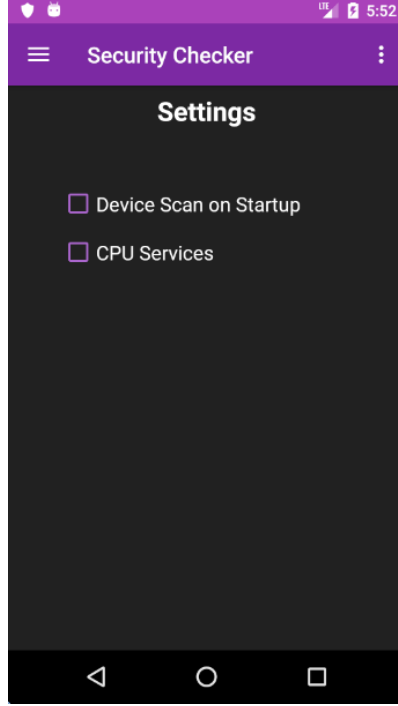
Şekil 4.1.1. Uygulama Ana Sayfası ve servisin açık olduğu durum



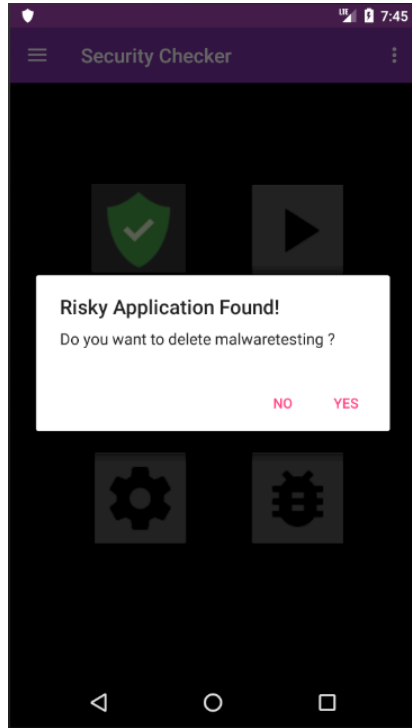
Şekil 4.1.2. Uygulama Ana Sayfası Servis kapalı olduğu durum



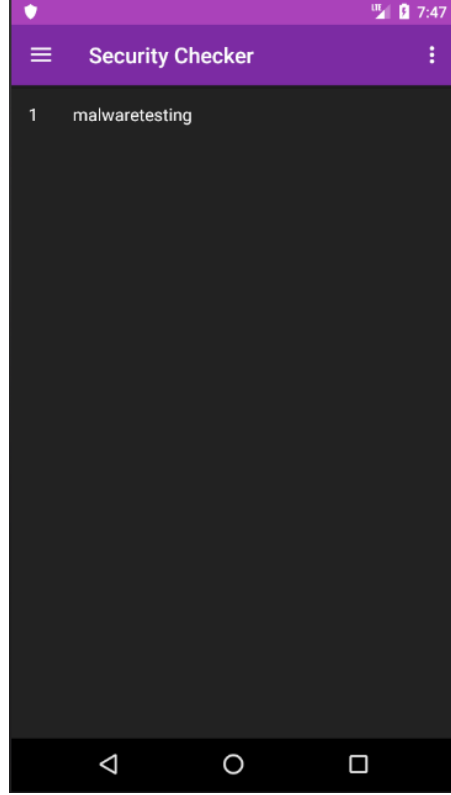
Şekil 4.1.3. NavigationDrawer görüntüsü



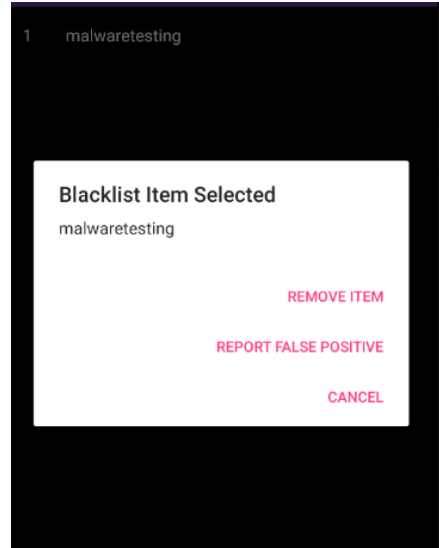
Şekil 4.1.4. Ayarlar sayfasının görünümü



Şekil 4.1.5. Kötücül uygulama bulundu uyarı mesajı



Şekil 4.1.6. Blacklist Listesinin görünümü

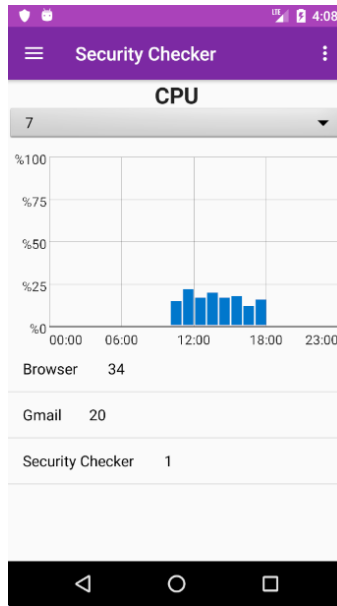


Şekil. 4.1.7 Sunucuya bildirilen yanlış tespit

4.2. CPU Tüketimi Modülü

Android cihazda her uygulama belirli bir CPU tüketimi kullanmaktadır. Yaptıkları işlem yoğunluğuna göre bu CPU tüketimi değişmektedir. Bazı uygulamalar ön planda CPU tüketimi yaparken bazı uygulamaların arka plan servisleri CPU tüketimi yapmaktadır. Kullanıcı farkında olmadan arka plan servisi olarak ve işlemciyi devamlı olarak kullanan bazı kötücül yazılımlar bulunmaktadır. Bu uygulamaları tespit etmek için uygulamamızda her uygulamanın kullandığı CPU oranı kayıt edilmektedir. Uygulamamız servis olarak çalışmaktadır. Bu servis kullanıcı tarafından açılıp kapatılabilmektedir. Her on saniyede bir cihaz üzerindeki CPU tüketimleri alınmaktadır. Bir saatlik tüketim tamamlandığında bir saat boyunca çalışan tüm uygulamalar cihazın kayıt dosyasına kayıt edilmektedir.

Elde edilen saatlik veriler kullanılarak cihazda CPU tüketimin fazla olduğu saatler analiz edilmektedir. Kullanıcı seçim yapığı aralıklara göre son 7,15,30 günde olan ortalama CPU tüketim oranının saatlere göre çizilmiş grafiğini görüntüleyebilmektedir. Aynı zamanda seçilen bu periyotlarda çalışan uygulamaların ismi ve CPU kullanım oranlarının sıralaması görüntülenebilmektedir (Şekil 4.2.1.'de görüldüğü gibi).



Şekil. 4.2.1. Yedi Günlük Ortalama CPU Kullanımı Grafiğı

5. Sonuç ve Değerlendirme

Veri setinden apicall ve izin niteliklerini bulundurmayan örnekler ve yinelenen örnekler atıldıktan sonra veri setinde 11490 normal, 3848 kötücül uygulama olmak üzere toplam 15338 uygulama örneği kalmıştır. Bu veri setinin %70'i train, %30'u test için kullanılmıştır. Scikit-learn kütüphanesinin f_classif skorlarına göre en iyi 79 nitelik seçilerek NaiveBayes, Knn, RandomForest, DecisionTree gibi makine öğrenmesi yöntemleri karşılaştırılmıştır. Nitelik seçimi öncesi ve sonrası doğruluk oranları Tablo 5.1.'de verilmiştir. Doğruluk oranlarına bakılarak 79 nitelik ile RandomForest algoritması %91,22 doğruluk oranıyla tespit sistemimizde kullanılmak üzere seçilmiştir. Tablo 5.2.'de nitelik seçimi öncesi, Tablo 5.3.'de nitelik seçimi sonrası algoritmanın TP (gerçek pozitif), FP (yanlış pozitif), TN (gerçek negatif), FN (yanlış negatif) değerleri verilmiştir.

Nitelik Boyutu	NaiveBayes	KNN	RandomForest	Karar Ağacı
237	%71,18	%87,22	%90,41	%85,87
79	%71,07	%87,26	%91,22	%85,37

Tablo 5.1. Nitelik Seçimi öncesi ve sonrası doğruluk oranları

Uygulama Tipi	Normal Tespit	Kötücül Tespit
Normal Uygulama (0)	3335 (TN)	112 (FP)
Kötücül Uygulama (1)	329 (FN)	826 (TP)

Tablo 5.2. Nitelik seçimi öncesi elde edilen karışıklık matrisi

Uygulama Tipi	Normal Tespit	Kötücül Tespit
Normal Uygulama (0)	3319 (TN)	128 (FP)
Kötücül Uygulama (1)	276 (FN)	879 (TP)

Tablo 5.3. Nitelik seçimi sonrası, 100 nitelik için elde edilen karışıklık matrisi

KAYNAKLAR

- [1] B. Sanz, I. Santos, X. Ugarte-Pedrero, C. Laorden, J. Nieves and P. Bringas, "Anomaly Detection Using String Analysis for Android Malware Detection", *Advances in Intelligent Systems and Computing*, pp. 469-478, 2014.
- [2] A. Saracino, D. Sgandurra, G. Dini and F. Martinelli, "MADAM: Effective and Efficient Behavior-based Android Malware Detection and Prevention", *IEEE Transactions on Dependable and Secure Computing*, vol. 15, no. 1, pp. 83-97, 2018.
- [3] T. Bläsing, L. Batyuk, A. Schmidt, S. Camtepe and S. Albayrak, "An Android Application Sandbox system for suspicious software detection", *2010 5th International Conference on Malicious and Unwanted Software*, 2010.
- [4] T. Isohara, K. Takemori and A. Kubota, "Kernel-based Behavior Analysis for Android Malware Detection", *2011 Seventh International Conference on Computational Intelligence and Security*, 2011.
- [5] S. Yerima, S. Sezer, G. McWilliams and I. Muttik, "A New Android Malware Detection Approach Using Bayesian Classification", *2013 IEEE 27th International Conference on Advanced Information Networking and Applications (AINA)*, 2013.
- [6] N. Peiravian and X. Zhu, "Machine Learning for Android Malware Detection Using Permission and API Calls", *2013 IEEE 25th International Conference on Tools with Artificial Intelligence*, 2013.
- [7] I. Burguera, U. Zurutuza and S. Nadjm-Tehrani, "Crowdroid", *Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices - SPSM '11*, 2011.
- [8] D. Arp, M. Spreitzenbarth, M. Hübner, H. Gascon and K. Rieck, "Drebin: Effective and Explainable Detection of Android Malware in Your Pocket", *Proceedings 2014 Network and Distributed System Security Symposium*, 2014.
- [9] D. Wu, C. Mao, T. Wei, H. Lee and K. Wu, "DroidMat: Android Malware Detection through Manifest and API Calls Tracing", *2012 Seventh Asia Joint Conference on Information Security*, 2012.
- [10] Yajin Zhou, Zhi Wang, Wu Zhou, Xuxian Jiang. "Hey, You, Get Off of My Market: Detecting Malicious Apps in Official and Alternative Android Markets". *Proceedings of the 19th Network and Distributed System Security Symposium NDSS*, 2012.
- [11] Uğur Pehlivan, "Permission Based Malware Detection Analysis in Android", unpublished, 2014.
- [12] G. Kuchimanchi, V. Phoha, K. Balagani and S. Gaddam, "Dimension reduction using feature extraction methods for real-time misuse detection systems", *Proceedings from the Fifth Annual IEEE SMC Information Assurance Workshop*, 2004.
- [13] W. Zhou, Y. Zhou, X. Jiang and P. Ning, "Detecting repackaged smartphone applications in third-party android marketplaces", *Proceedings of the second ACM conference on Data and Application Security and Privacy- CODASKY '12*, 2012.
- [14] M. Grace, Y. Zhou, Q. Zhang, S. Zou and X. Jiang, "RiskRanker", *Proceedings of the 10th international conference on Mobile systems, applications, and services- MobiSys '12*, 2012.
- [15] E. Davarci, B. Soysal, I. Erguler and E. Anarim, "Side channel analysis on android smartphones", *2016 24th Signal Processing and Communication Application Conference (SIU)*, 2016.
- [16] Dr. Wei Wang, "Data sets of Android apps' features extracted from APK files", Son Erişim Tarihi: 27.05.2018, URL: http://infosec.bjtu.edu.cn/wangwei/?page_id=85, 2015.