

AÇIK KAYNAKLI YAZILIM UYGULAMALARI

141180056 RENEZ ERİM ÖZTÜRK
141180061 FURKAN ŞOLPAN

FİNAL RAPORU

GAZİ ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ

BM495 BİLGİSAYAR PROJESİ I

OCAK 2018

SECURITY CHECKER

RENEZ ERİM ÖZTÜRK
FURKAN ŞOLPAN

PROJE DANIŞMANI: ÖĞR.GÖR.DR. OKTAY YILDIZ

ÖZET

Günümüzde yaygın olarak kullanılan Android telefonlarda çalışan uygulamalar arasında zararlı uygulamalar bulunmaktadır. Bu zararlı uygulamalar kullanıcının telefonunun kontrolünü ele geçirmekten, hassas bilgilerini çalmaya kadar birçok farklı tipte bulunmaktadır. Bir Android telefona uygulamalar, Google Play Store ya da farklı bir uygulama mağazası aracılığıyla yüklenebilmektedir. Bu uygulama mağazalarından kullanıcıların daha rahat bir şekilde yararlanabilmesi için Security Checker adlı uygulama geliştirilmiştir.

Bu Uygulama, zararlı ve normal uygulamaların izin listesi örneklerini çeşitli makine öğrenmesi teknikleri ile inceleyerek bir uygulamanın riskli bir uygulama olup olmadığını tespit etmektedir. Uygulama tamamen açık kaynaklı bir şekilde geliştirilmiştir. Bu Uygulama sayesinde kullanıcılar farklı Android Uygulama Mağazalarından güvenli bir şekilde yararlanabilmeleri amaçlanmıştır.

Uygulamanın kullandığı verilerin güncellenmesi amacıyla bir sunucu yapılmıştır. Sunucu İşletim Sistemi bağımsız bir şekilde açık kaynaklı Play Framework kullanılarak Java dili ile yapılmıştır. Uygulamanın gönderdiği veriler, Python dili ile yazılmış makine öğrenmesi modelini içeren tespit sistemine verilmiştir. Sistem, verinin zararlı bir uygulamaya ait olup olmadığı tespit edilerek Security Checker uygulamasına sunucu aracılığıyla yanıt vermektedir.

Bu uygulamada kullanılan izin veri seti, açık veri setlerini içeren kaggle.com sitesinden temin edilmiştir. Bu veri seti COLCOM 2016 tezinde kullanılmak üzere toplanmıştır [6]. Bu veri seti 200'ü zararlı, 400 adet örnek uygulamanın, izin listelerini içermektedir. Bu veri setinin %70'lik kısmı makine öğrenmesi modelini eğitmek için, geriye kalan %30'luk kısmı bu modeli test etmek için kullanılmıştır. İzin listeleri 330 adet farklı izin içermektedir. Bu izinlerin her biri makine öğrenmesi teknikleri için bir niteliktir. Çok fazla nitelik olması nedeniyle ilk önce bir Nitelik Seçimi yapılarak nitelik sayısı 25'e indirgenmiştir. Bu 25 adet izin makine öğrenmesi modeli üzerinde büyük etkide bulunmaktadır.

Veri seti nitelik seçimi yapılmış hali ile farklı makine öğrenmesi teknikleriyle incelenmiştir. Bu tekniklerdan NaiveBayes algoritmasından %85-90, KNN algoritmasından %90-95, RandomForest algoritmasından %90-94 doğruluk oranı elde edilmiştir. Elde edilen doğruluk oranlarına göre bu uygulamada KNN Makine Öğrenmesi Algoritması kullanılmıştır.

İÇİNDEKİLER

1. GİRİŞ 5
1.1 Projenin Amacı 5
2 Android Güvenlik Konusunda Yapılan Çalışmalar 6
2.1 Detecting Malicious Apps in Official and Alternative Android Markets 6
2.2 RiskRanker: Scalable and Accurate Zero-day Android Malware Detection 6
2.3 Crowdroid: Behavior-Based Malware Detection System for Android 6
2.4 Detecting Repackaged Smartphone Applications in Third-Party Android Marketplaces 7
2.5 Permission Based Malware Detection Analysis in Android 7
3. Kullanılacak Araç ve Yöntemler 8
3.1. Android Studio 8
3.2. Sunucu 9
3.3. Veri Seti 10
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
4. Gerçekleştirilen Uygulama 16
4.1 Sunucu Tasarımı 17
4.2 Arayüz Tasarımı 17
5. Sonuç ve Değerlendirme 22
KAYNAKLAR 24

1. GİRİŞ

Gerçekleştirilen Security Checker adlı uygulama, Foreground Service olarak çalışan yeni kurulan uygulamaları tespit eden bir servisten ve android telefonda bulunan tüm uygulamaları tarayan bir modülden oluşmaktadır. Bu uygulama tarafından ele alınan bir uygulamanın paket ismi ve izin listesi JSON formatında Play Framework ile kurulan sunucuya gönderilerek zararlı yazılım olup olmama riski öğrenilmiştir. Sunucu tarafında elde edilen veriler kayda alınmaktadır. Uygulama üzerinde zararlı olarak tespit edilmiş uygulamalar kullanıcı tarafından silinse de silinmese de bir liste içinde tutulmaktadır. Kullanıcı zararlı 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 mağazalarda yoğun miktarda zararlı yazılımlar bulunmaktadır. Bu yazılımlar 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 makine öğrenmesi teknikleri ile inceleyerek, kurulan uygulamanın zararlı yazılım 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

2.1 Detecting Malicious Apps in Official and Alternative Android Markets

ABD North Carolina State Üniversitesinde gerçekleştirilen bu çalışmada İzin temelli davranış izleri tasarısı bilinen Android Zararlı Uygulama Ailelerinin yeni örneklerinin tespiti için kullanılmıştır. Daha sonra sezgisel temelli bir filtreleme tasarısı kullanarak bilinmeyen Android Zararlı Uygulama Ailelerinin belirli miras alınmış özellikleri tespit edilmiştir. Bu iki şemayı birleştirerek DroidRanger adını verdikleri sistemi hayata geçirmişlerdir. DroidRanger 2011 yılından 204,000 uygulama kullanarak test edilmiştir. 211 zararlı uygulama bulmuştur. Bu zararlı uygulamalardan 2 tanesi sıfırıncı günde bulunmuştur [1].

2.2 RiskRanker: Scalable and Accurate Zero-day Android Malware Detection

ABD North Carolina State Üniversitesinde gerçekleştirilen bu çalışmada proaktif bir tasarı kullanılarak sıfırıncı gün android zararlı yazılımlar tespit edilmiştir. RiskRanker kademeli olarak belli bir uygulamanın tehlikeli davranışlarda bulunup bulunmadığını takip etmektedir (örnek: arkaplanda SMS göndermek). Çıktılar önceliklendirilerek azaltılan uygulama listesi elde etmekte kullanılır. 118.318 uygulama incelenerek 3.281 riskli uygulama bulundu bunlardan 718 tanesi zararlı yazılımdı. Bu zararlı yazılımların 322 tanesi sıfırıncı günde bulunmuştur [2].

2.3 Crowdroid: Behavior-Based Malware Detection System for Android

İspanya Mondragon Üniversitesinde gerçekleştirilen bu çalışmada Statik analizler efektif olmadığı vurgulanarak dinamik analiz (davranışsal analiz) kullanılmıştır. Program indirilen uygulamayı izole edilmiş bir ortamda çalıştırarak sistem çağrılarını incelemektedir [3].

2.4 Detecting Repackaged Smartphone Applications in Third-Party Android Marketplaces

ABD North Carolina State Üniversitesinde gerçekleştirilen bu çalışmada 6 popular Android Uygulama Mağazasında sistematik bir analiz yapılmıştır. Bu çalışma sırasında uygulama mağazalarında resmi android mağazalarından indirilen gerçek uygulamaların tekrar paketlenerek yüklendiği görülmüştür. DroidMOSS adlı uygulama fuzzy hashing tekniği kullanarak bu yeniden paketlenen uygulamalardaki değişiklikleri incelemiştir. Deneyler piyasadaki %5 ila %13 arasındaki uygulamaların yeniden paketlendiğini göstermiştir. Yeniden paketlenen bu uygulamalar reklam yönlendirme/çalma, backdoor, kötücül ücretlendirmeler içermektedir [4].

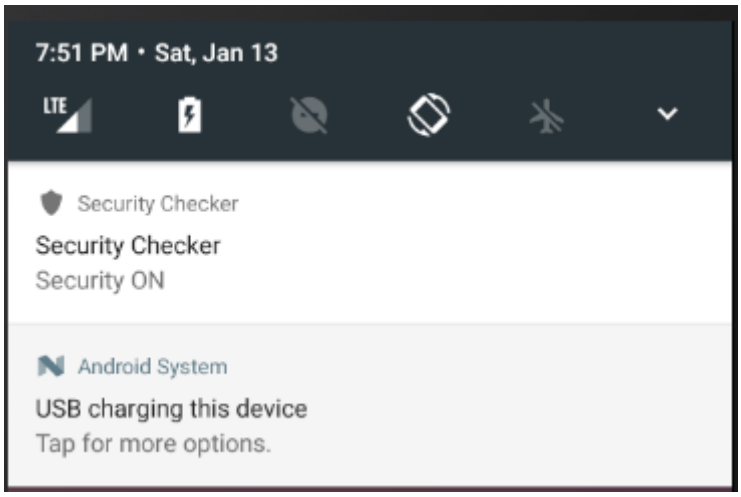
2.5 Permission Based Malware Detection Analysis in Android

Türkiye ODTÜ'nde gerçekleştirilen bu uygulamada en iyi sonuçları veren nitelik seçim teknikleri ve sınıflama algoritmaları analiz edilmiştir. Sınıflandırma algoritmaları olarak Batesian, decision tree ve SVM kullanılmıştır. Kümeleme analiz yöntemi kullanarak izinler incelenmiş ve bulgular sunulmuştur [5].

3. Kullanılacak Araç ve Yöntemler

3.1. Android Studio

Gerçekleştirilen uygulama Android Studio üzerinde geliştirilmiştir. 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 izin listeleri çıkarılmıştır. Çıkarılan izin listesi elimize bulunan veri seti formatına göre işlenmiştir. Veri setinde android uygulamalarda bulunan tüm izinler bulunmaktadır. Eğer uygulama bu izinlerden içeriyorsa “1” içermiyorsa “0” olarak işlenmektedir. Çıkartılan yeni veri sunucuya JSON formatında gönderilmiştir. Veri AsyncTask kullanarak gönderilmiştir bu uygulama arayüzünün sunucudan cevap beklenirken kitlenmemesini sağlamaktadır.

Bu uygulama zararlı olma potansiyeline sahip uygulamaları kullanıcıya bildirerek silmek için izin istemektedir. Kullanıcı zararlı uygulamayı silmese de riskli uygulama Blacklist isimli listeye eklenmektedir. Kullanıcı zararlı 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 İşletim Sistemi bağımsız bir şekilde Java dilinde Play Framework kullanarak geliştirilmiştir. Play Framework scala ile yazılmış açık kaynaklı bir frameworktür. Play Framework localhost:9000 üzerinde bir yerel sunucu kurmaktadır. Virtual Box üzerinde kurulu olan sunucuya bilgisayarın 90 portunu yönlendirerek uygulama testlerini gerçekleştirdik. Testler sırasında Android emülatör Windows üzerinde çalıştırılmıştır ve bilgisayarın 90 portuna istekler atılmıştır.

Bu sunucuda Python üzerinde yazılmış olan tespit sistemimiz çalışmaktadır. Android uygulamamızdan gönderilen verileri, elindeki KNN makine öğrenmesi Modeline göre tahmin ederek, Uygulamamızın gönderdiği verinin zararlı (1) veya zararlı olmayan (0) uygulama olduğunu geri bildirmektedir.

Elde edilen yeni veriler günlük kayıt dosyaları şeklinde sunucuda saklanmaktadır. Uygulamadan yanlış tespit olarak bildirilen uygulamalarda ayrı bir kayıt dosyasında saklanmaktadır. Gelecek günlerde bu kayıt dosyaları kullanılarak veri seti güncellenebilecektir.

3.3. Veri Seti

Tespit sistemimizde kullanılmak üzere 400 adet örnek içeren kaggle' a yüklenmiş olan veri seti indirilmiştir. Veri seti COLCOM 2016 tezinde kullanılmak üzere toplanmıştır [6]. Bu veri seti 200 adet zararlı, 300 adet zararlı olmayan uygulamanın sahip olduğu nitelikleri içermektedir. Biz bu veri setinin %70'ini modelimizi eğitmek, %30'unu test etmek için kullandık.

3.4. Nitelik Seçimi

Kaggle sitesinde bulunan android izin tabanlı veri setimizde 330 adet nitelik bulunmaktadır. Bütün bu özellikleri incelediğimiz zaman 330 adet nitelik çok fazla özellik içermektedir. Bu özelliklerin bir kısmı makine öğrenmesinin zararlı uygulama tespitinde önemli rol oynarken bazı özellikler tespit etmekte etkisi olmamaktadır. Nitelik sayısını azaltarak önemli olan nitelikler ön plana çıkmış olup zararlı uygulama tespit edilmesinde daha doğru sonuçlar vermektedir. Nitelik sayısı seçiminde belirli bir eşik değeri hesaplanmıştır. Bu eşik değerinin üstünde olan nitelikler değerlendirmeye alınmıştır. Bu eşik değerinin hesaplanmasında Şekil 3.4.1'deki formül kullanılmıştır [7].

$$\mathrm{Var}[X] = p(1-p)$$

Şekil 3.4.1 Nitelik seçiminde kullanılan eşik değeri hesaplama formülü

Nitelik seçiminde %90 oranının altında değişkenlik göstermeyenleri eleme yapılmıştır. p olasılık 0.9 olarak hesapladığımızda eşik değerimiz 0.9 (1-0.9) =0.09'dur. Eşik değeri altında bulunan nitelikler elenmiştir. Toplam 330 nitelik bulunan veri setimiz eleme yaptığımızda 25 nitelik sayısına indirilmiştir. Seçilen nitelikler Tablo 3.4.1'de görüldüğü gibidir.

Feature Name	Threshold
android.permission.ACCESS_COARSE_LOCATION	0.23
android.permission.ACCESS_FINE_LOCATION	0.23
android.permission.ACCESS_NETWORK_STATE	0.57
android.permission.ACCESS_WIFI_STATE	0.41
android.permission.CALL_PHONE	0.15
android.permission.CHANGE_WIFI_STATE	0.22
android.permission.DISABLE_KEYGUARD	0.10
android.permission.GET_TASKS	0.12
android.permission.INSTALL_PACKAGES	0.12
android.permission.INTERNET	0.75
android.permission.READ_CONTACTS	0.21
android.permission.READ_EXTERNAL_STORAGE	0.11
android.permission.READ_LOGS	0.11
android.permission.READ_PHONE_STATE	0.53
android.permission.READ_SMS	0.32
android.permission.RECEIVE_BOOT_COMPLETED	0.33
android.permission.RECEIVE_SMS	0.17
android.permission.RESTART_PACKAGES	0.11
android.permission.SEND_SMS	0.18
android.permission.VIBRATE	0.22
android.permission.WAKE_LOCK	0.25
android.permission.WRITE_APN_SETTINGS	0.13
android.permission.WRITE_CONTACTS	0.13
android.permission.WRITE_EXTERNAL_STORAGE	0.53
android.permission.WRITE_SMS	0.26

Tablo 3.4.1 Nitelik Seçimi sonrası elde edilen 25 nitelik

3.5. Makine Öğrenmesi

Zararlı yazılım tespitinde Statik Analiz yöntemi kullanılmıştır [8]. Android telefonda kurulu olan uygulamaların 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 ve RandomForest algoritmaları kullanılmıştır [9]. Bu veri setindeki 400 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. Kullanılan üç farklı makine öğrenmesi işlemini scikit-learn kullanılarak makine öğrenmesi sonuçları karşılaştırılmıştır. Daha sonra Nitelik Seçimi yapılarak aynı makine algoritmaları eski sonuçlar ile karşılaştırılmıştır. Naive Bayes algoritmasında %2-3 başarı artışı, KNN algoritmasında %2-4 başarı artışı, Random Forest algoritmasında %1-2 başarı artışı gözlenmiştir.

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 [10]. 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

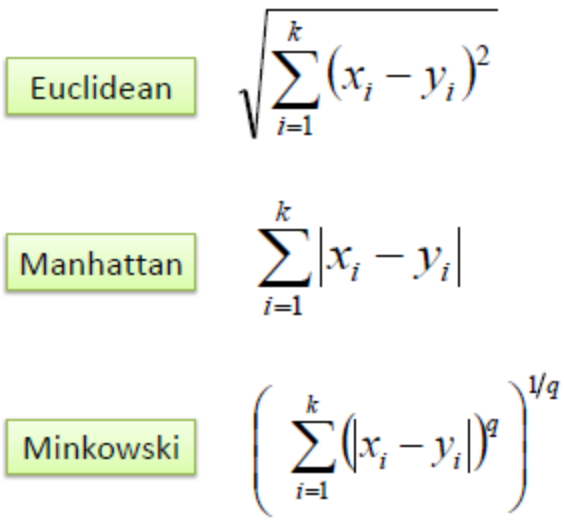
A ve B'nin önsel olasılığı bayes teoremine öznellik katar. A olayı hakkında bir bilgi edinmeden A olasılığı hakkında bilgi sahibi olunabilir. Bunun yanında P(*B*|*A*) ardıl olasılıktır, veriler toplanmaya başladıktan sonra A'nın gerçekleşmiş olduğu durumlarda B'nin gerçekleşme olasılığı hakkında tahmin 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 [11].

Elimizde bulunan veri setimizde NaiveBayes algoritması uyguladığımızda %85-90 oranında başarı oranı vermektedir. Nitelik Seçimiyle nitelik sayısı azalttıldığı zaman %87-%91 oranında başarı vermektedir.

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 işlemdir [12].

KNN uzaklık hesaplarken 3 farklı uzaklık formülü kullanmaktadır. Bunlar Öklid, Manhattan ve Gauss teoremidir [13]. Şekil 3.5.2.1. de uzaklık hesaplama formülleri verilmiştir.



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.

Elimizde bulunan veri setimizde KNN algoritması %90-93 oranında başarı oranı vermektedir. Nitelik Seçimiyle nitelik sayısı azalttıldığı zaman %92-%96 oranında başarı vermektedir.

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

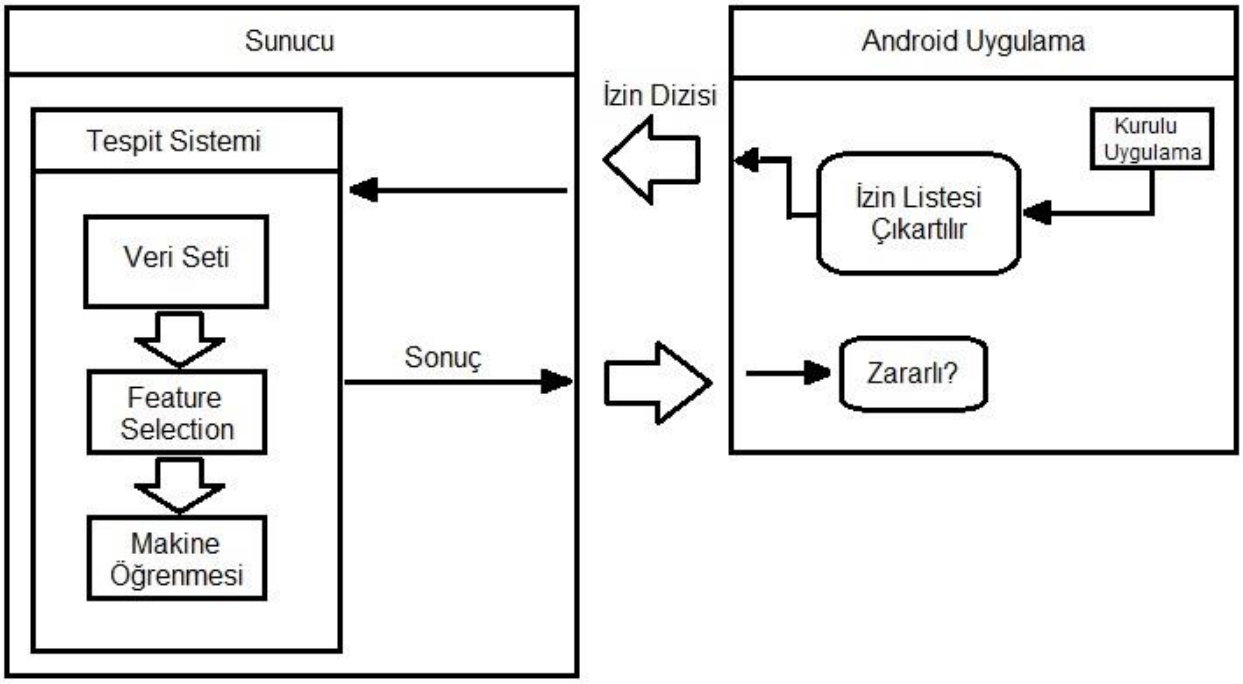
tahminlerini almaktadır. Bu ortalama tahminlere göre en uygun olan ağacı seçmektedir [14].

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 kullanılabilmektedir. 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 karşılaşılabilir [15].

Elimizde bulunan veri setimizde RandomForest algoritması %90-93 oranında başarı oranı vermektedir. Nitelik Seçimiyle nitelik sayısı azaltıldığı zaman %91-%94 oranında başarı vermektedir.

4. Gerçekleştirilen Uygulama

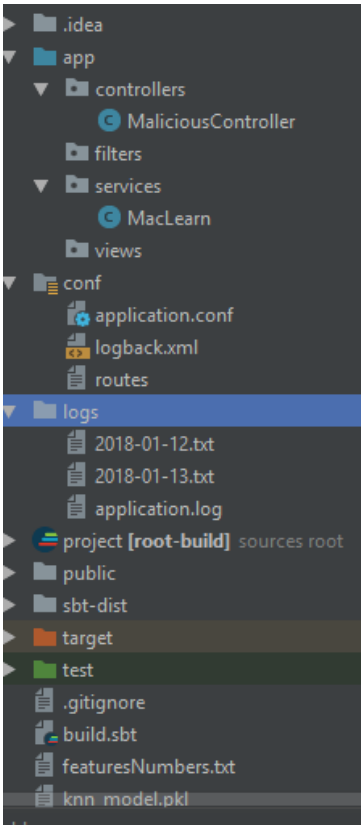


Şekil 4.1 Veri akış diyagramı

Şekil 4.1’ de gerçekleştirilen uygulamanın veri akış diyagramı verilmiştir. Diyagramda görüldüğü üzere kurulu olan bir uygulamadan izin listesi çıkartılarak sunucuya gönderilmektedir. Sunucuda daha önceden çalışır vaziyette olan Tespit Sisteminde sunucudan alınan izin listesi hazır olan makine öğrenmesi modeline göre tahmin edilerek sonuç uygulamaya gönderilmektedir.

4.1 Sunucu Tasarımı

Play Framework ile kurulan sunucuda basit bir post isteği alınarak kalan işlemler Java ve Python arasındaki veri alışverişi üzerinden ilerlemektedir. Şekil 4.1.1 de sunucunun sınıf hiyerarşisi görülmektedir. Günlük tutulan kayıtlar ve yanlış tespit kayıtları burada bulunmaktadır.



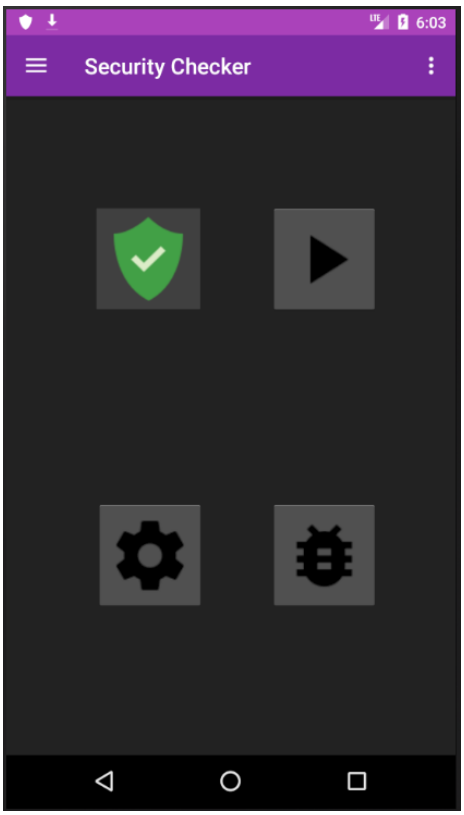
Şekil 4.1.1 Play Framework Sınıf Hiyerarşisi

4.2 Arayüz Tasarımı

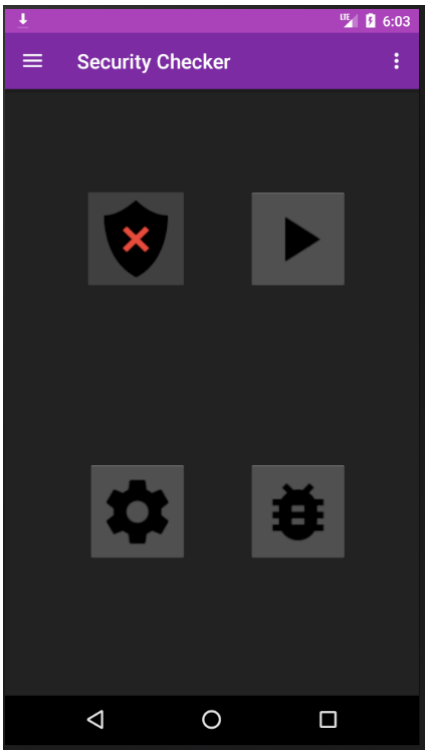
Kullanıcı arayüzü tasarlanırken pratik olması ön plana konulmuştur. Sade olan uygulama sayfaları yaratılmıştır. Şekil 4.2.1' de uygulamanın ana sayfası görülmektedir. Uygulamanın kullanım kolaylığı sağlaması amacıyla bir NavigationDrawer da vardır. Şekil 4.2.3' te NavigationDrawer görülmektedir. Şekil 4.2.4' te ayarlar sayfası görülmektedir.

Yeni kurulan uygulamaları taramak için konulan ForegroundService' i Şekil 3.1.1' de çalışır durumda görülmektedir. Bu servis Şekil 4.2.1 ve 4.2.2 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.

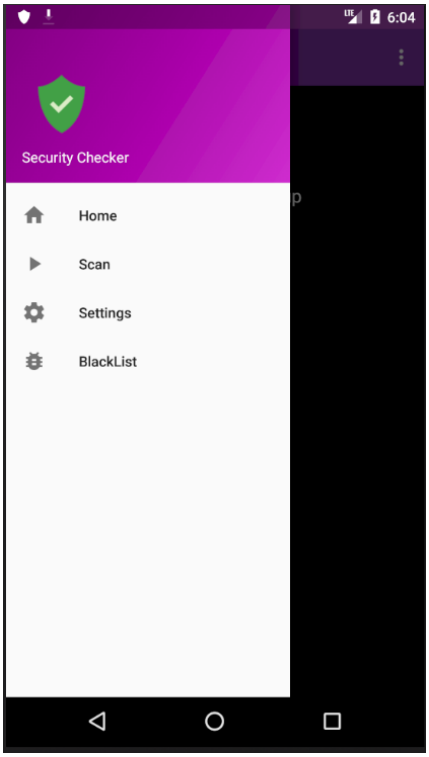
Şekil 4.2.5' da zararlı bir uygulama bulunduğunda kullanıcıya verilen uyarı mesajı görülmektedir. Kullanıcı isterse bu uyarıyı görmezden gelebilir. Her halükârda Şekil 4.2.6' de görülen Blacklist adlı listeye bu bulunan uygulamalar eklenmektedir. Şekil 4.2.7' de yanlış bir tespitin nasıl sunucuya bildirildiği gösterilmiştir.



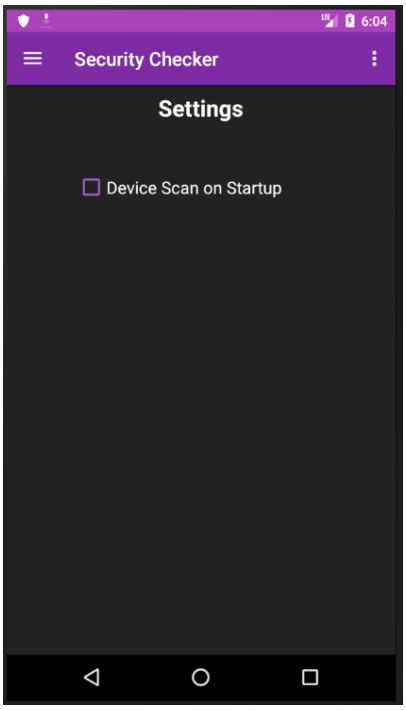
Şekil 4.2.1 Uygulama Ana Sayfası ve servisin açık olduğu durum



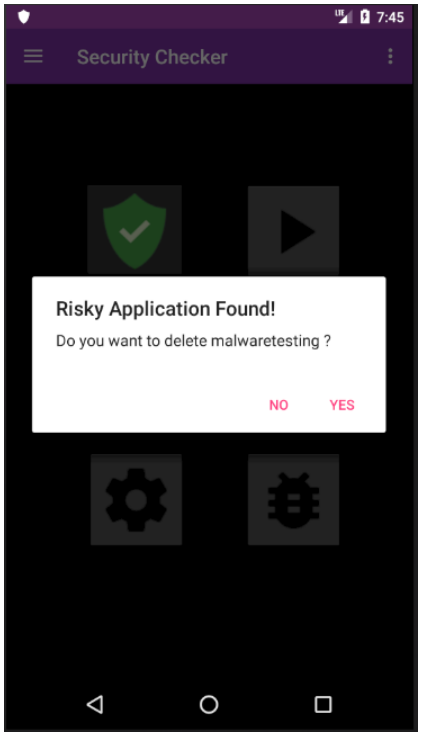
Şekil 4.2.2 Uygulama Ana Sayfası Servis kapalı olduğu durum



Şekil 4.2.3 NavigationDrawer görüntüsü



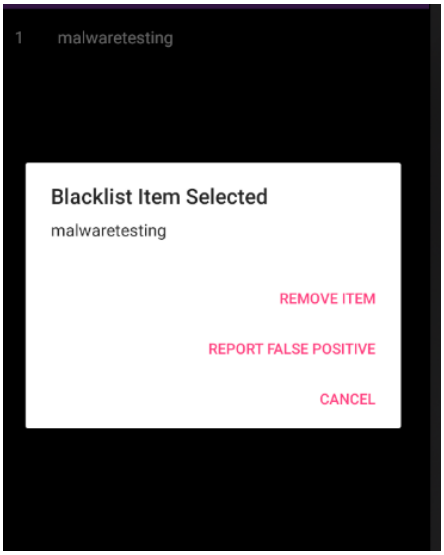
Şekil 4.2.4 Ayarlar sayfasının görünümü



Şekil 4.2.5 Zararlı uygulama bulundu uyarı mesajı



Şekil 4.2.6 Blacklist Listesinin görünümü



Şekil 4.2.7 Sunucuya bildirilen yanlış tespit

5. Sonuç ve Değerlendirme

Nitelik seçiminde %90 oranının altında değişkenlik göstermeyenleri eleme yapılmıştır. p olasılık 0.9 olarak hesapladığımızda eşik değerimiz 0.9 (1-0.9) =0.09'dur. Eşik değeri altında bulunan nitelikler elenmiştir. Toplam 330 nitelik bulunan veri seti eleme yapıldığında 25 nitelik sayısına indirilmiştir. Seçilen nitelikler Tablo 3.4.1'de görüldüğü gibidir.

Doğruluk oranlarına göre seçilerek gerçekleştirdiğimiz uygulamada KNN algoritması kullanılmıştır. Şema 5.2.1. 'de gösterilen doğruluk oranları train ve test rastgele oranlarda seçildiği için tutarsızlıklar göstermektedir. Rapora konulmak üzere tek denemede çıkan oranlar doğrudan kullanılmıştır.

Şema 5.2.1 'de Nitelik Seçimi öncesi ve sonrası elde edilmiş Doğruluk Oranları verilmiştir. NaiveBayes algoritması %89 oranından %90-91 oranına yükselmiştir. KNN algoritması %90 oranından %91-92 oranına yükselmiştir. RandomForest algoritması %90 oranından %92-93 oranına yükselmiştir. Nitelik Seçimi bize %1-2 oranında bir iyileştirme sağlamıştır.

Play Framework Virtual Boxta Linux üzerine kurulmuştur. Yerel bir sunucu olarak bu sistemi kullanılmıştır. Play Framework, Sanal makinanın 9000 portunu dinlemektedir. Bilgisayarın 90 portunu VirtualBox' a yönlendirerek daha gerçekçi bir test ortamı yaratmayı amaçladık. Android Emülatör'den gelen istekleri 90 portuna atarak testler gerçekleştirilmiştir.

	NaiveBayes	KNN	RandomForest
DoğrulukOranı (300+ feature)	0.89	0.91	0.90
DoğrulukOranı (25 Feature)	0.91	0.94	0.93

Şema 5.2.1. Selection Öncesi ve Sonrası Doğruluk Oranları

KAYNAKLAR

1- Detecting Malicious Apps in Official and Alternative Android Markets,
Url: http://wp.internetsociety.org/ndss/wp-content/uploads/sites/25/2017/09/07_5.pdf

2- RiskRanker: Scalable and Accurate Zero-day Android Malware Detection,
Url: https://dl.acm.org/citation.cfm?id=2307663

3- Crowdroid: Behavior-Based Malware Detection System for Android,
Url: https://dl.acm.org/citation.cfm?id=2046619

4- Detecting Repackaged Smartphone Applications in Third-Party Android Marketplaces
Url: https://dl.acm.org/citation.cfm?id=2133640

5-Permission Based Malware Detection Analysis in Android,
Url: http://etd.lib.metu.edu.tr/upload/12618000/index.pdf

6- Urcuqui, C., & Navarro, A. (2016, April). Machine learning classifiers for android malware analysis. In Communications and Computing (COLCOM), 2016 IEEE Colombian Conference on (pp. 1-6). IEEE.

7- Feature Selection
Url: http://scikit-learn.org/stable/modules/feature_selection.html

8- DroidMat: Android Malware Detection through Manifest and API Calls Tracing
Url: http://ieeexplore.ieee.org/document/6298136/

9- Essentials of Machine Learning Algorithms (with Python and R Codes)
Url:https://www.analyticsvidhya.com/blog/2017/09/common-machine-learning-algorithms/

10. Naïve Bayes Classifier 20/12/2017
Url: https://en.wikipedia.org/wiki/Naive_Bayes_classifier

11. Naïve Bayes 20/12/2017
Url: http://scikit-learn.org/stable/modules/naive_bayes.html

12. KNN 20/12/2017
Url: https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm

13. KNN 20/12/2017
Url: http://kenanatmaca.com/knn-k-nearest-neigborhood-algoritmasi/

14. Random Forest 20/12/2017
Url: https://en.wikipedia.org/wiki/Random_forest

15. Karar Ağacı öğrenmesi 20/12/2017
Url: http://bilgisayarkavramlari.sadievrenseker.com/2012/04/11/karar-agaci-ogrenmesi-decision-tree-learning/