Süleyman Demirel Üniversitesi
Fen Bilimleri Enstitüsü Dergisi
Cilt 22, Sayı 2, 1087-1094, 2018

Süleyman Demirel University
Journal of Natural and Applied Sciences
Volume 22, Issue 2, 1087-1094, 2018

DOI: 10.19113/sdufbed.20066

Makine Öğrenmesi Algoritmalarıyla Android Kötücül Yazılım Uygulamalarının Tespiti

Abdurahman AYDIN*[1], İbrahim Alper DOĞRU[2], Murat DÖRTERLER[2]

[1]Gazi Üniversitesi, Fen Bilimleri Enstitüsü, Bilgi Güvenliği Mühendisliği Anabilim Dalı, 06500, Ankara
[2]Gazi Üniversitesi, Teknoloji Fakültesi, Bilgisayar Mühendisliği Bölümü, 06500, Ankara

(Alınış / Received: 10.10.2017, Kabul / Accepted: 21.02.2018, Online Yayınlanma / Published Online: 08.04.2018)

Anahtar Kelimeler
Android kötücül yazılımlar,
Makine öğrenmesi,
Statik analiz

Özet: Son yıllarda akıllı mobil cihazlar hayatımızı ciddi anlamda kolaylaştırmış ve hız kazandırmıştır. Android işletim sistemi (İS) bu cihazlar arasında en yüksek kullanım oranına sahiptir. Yaygın kullanım, yetersiz güvenlik mekanizmaları ve kullanıcıların bilinç düzeyi bu İS'ni saldırganların hedefi haline getirmektedir. Android İS'nin güvenlik mekanizmasını temelini izin tabanlı güvenlik modeli oluşturmaktadır. Uygulamalar, kullanıcı tarafından verilen izinlere bağlı olarak işlevlerini yerine getirebilmektedir. Ancak kullanıcı farkındalığı, talep edilen izinlerin suistimale açık olup olmadığı hususunda yeterli seviyede değildir. Bu sebeple bu uygulamalarda kötücül içerik tespiti için ek yöntemlere ihtiyaç duyulmaktadır. Bu çalışmada, kötücül yazılım uygulamalarının tespiti amacıyla makine öğrenmesi algoritmaları kullanılarak izin tabanlı bir yöntem önerilmiştir. Önerilen yöntem destek vektör makinesi, rastgele orman, Naïve Bayes ve K en yakın komşu makine öğrenmesi algoritmalarıyla ayrı ayrı denenmiş ve başarımları kıyaslanmıştır. Rastgele orman algoritması %95,65 doğruluk oranıyla en yüksek başarımı sergilemiştir.

The Detection of Android Malware Applications with Machine Learning Algorithms

Keywords
Android malwares,
Machine learning,
Static analysis

Abstract: Smart mobile devices have made our life easier and faster in recent years. Android Operating System (OS) has the highest usage rate among these devices. Widespread usage, inadequate security mechanisms and the level of consciousness of users make this OS the target of attackers. The security mechanism of the Android OS is based on permission-based security model. Applications can perform their functions depended on the permissions granted by the user. Yet, the awareness of the users is not at the sufficient level to ensure whether the requested permissions are open to abuse. For this reason, additional methods are needed to determine the malicious content in the applications. In this study, a permission-based method was proposed for detecting malicious software applications by using machine learning algorithms. The proposed method was tried by the support vector machine, random forest, Naïve Bayes and K nearest neighbor machine learning algorithms and their performances were compared. The random forest algorithm showed the highest performance with an accuracy rate of 95.65%.

1. Giriş

Son yıllarda akıllı mobil cihazların göstermiş olduğu hızlı gelişim sayesinde bilgiye her yerden istenilen her zaman erişim oldukça kolaylaşmıştır. Bu nedenle bu cihazların kullanım oranı hızla artmıştır. Akıllı telefonlar ve tablet bilgisayarlar kullanılarak insan hayatını kolaylaştırıcı birçok işlemi hızlı ve kolay bir şekilde gerçekleştirebilmektedir. Cisco firmasının hazırladığı rapora göre 2020 yılına kadar dünya çapındaki cihazların ve internet bağlantıların yaklaşık %50'sini mobil cihazların oluşturacağı tahmin edilmektedir [1]. Akıllı mobil cihazların artan kullanım oranları saldırgan ve diğer kötü niyetli kişilerin de ilgisini çekmektedir. International Data Corporation Media Center'in 2017'in birinci çeyreği için yapmış olduğu araştırmaya göre uygulama dağıtım kanalları önceki yılın aynı çeyreğine oranla %1,6 daha büyümüştür. Aynı araştırmaya göre Android İS %85 oranıyla dünya çapında en çok kullanılan mobil İS'dir [2].

*İlgili yazar: abdurahman.aydin@gazi.edu.tr

Android İS cep telefonları ve tablet bilgisayarların yanı sıra televizyonlarda, otomobillerde, kameralarda, ev otomasyon sistemlerinde ve giyilebilir teknolojilerde de yoğun olarak kullanılmaktadır. Statista'nın paylaşmış olduğu istatistiklere göre, Android'in resmi uygulama marketi olan Play Store'daki uygulama sayısı 3 milyonu aşmıştır [3]. Sensor Tower'ın 2016'nın son çeyreği için yayınlamış olduğu rapora göre dünya çapında Play Store'dan 2015 yılının son çeyreğinde 10,5 milyar, 2016 yılının son çeyreğinde ise 12,7 milyar uygulama indirilmiştir [4]. Bu durum resmi marketler aracılığıyla indirilen uygulama sayısının gün geçtikçe arttığını göstermektedir. Mcafee firmasının mobil tehdit raporuna göre ilk altı ayda tespit edilen kötü amaçlı yazılım sayısı 37 milyon civarındadır [5]. Play Store üzerindeki kötücül yazılım uygulaması (KYU - Malicious software application) sayısının çok fazla olmasının sebebi, Android İS'nin açık kaynak kodlu olması ve markete yüklenen uygulamaların güvenlik taramalarından yeterince geçirilmemesidir. Ayrıca siber güvenliğin önem kazandığı çağımızda, akıllı cihazlara yönelik tehditlerin tespit edilmesi ve önlenmesi amaçlı kapsamlı çalışmaların yapılması gerekmektedir.

G Data firmasının yayınlamış olduğu rapora göre her yıl ortaya çıkan yeni Android KYU örneği bir önceki yıla göre önemli miktarda artmaktadır. Bu sayının, 2016 yılı için 4.250.000'e ulaştığı tahmin edilmektedir [6]. KYU'lar genellikle kişisel verilerin elde edilmesi, ortam dinlemesi ve şifrelerin çalınması gibi amaçlarla kullanılmaktadırlar. Android İS'nin güvenlik mekanizmasının temelinde izin modeli bulunmaktadır. Bu modelde uygulamalar, kullanmış oldukları izinler aracılığıyla görevlerini yerine getirebilmektedirler. Uygulamalar işlevine göre, rehbere erişim, iletileri okuma-yazma, galeriye erişim, konum bilgisine erişim gibi çeşitli izinleri talep etmektedirler. Android 6.0'dan önceki sürümlerde uygulamanın yüklenebilmesi için kullanıcının talep edilen izinleri onaylaması gerekmekteydi. Android 6.0 sürümü ile birlikte uygulama yükledikten sonra da kullanıcıların bu izinlere müdahale edebilmesine imkân sağlanmıştır. Böylece kullanıcı uygulama tarafından kullanılan izinleri onaylamadan uygulamayı kurabilmekte ya da kullanım aşamasında da bu izinleri kaldırabilmektedir. Ancak bu düzenleme kullanıcıların farkındalığının düşük olması nedeniyle istenilen faydayı sağlamamaktadır. Nitekim Felt ve arkadaşları tarafından yapılan çalışma, kullanıcıların yalnızca %17'sinin bu izinlere dikkat ettiğini, %42'sinin ise izinler hakkında bilgisinin olmadığını ortaya çıkarmıştır [7]. Dolayısıyla Android cihaz güvenliğinin sağlanması açısından KYU'ların tespiti kadar kullanıcıların bilinçlendirilmesi de önem teşkil etmektedir. Tablo 1'de Android geliştiricileri tarafından belirlenen tehlikeli izinler ve izin grupları yer almaktadır.

Tablo 1. Tehlikeli İzin ve İzin Grupları [8]

İZİN GRUBU	İZİNLER
CALENDAR (TAKVİM)	READ_CALENDAR (TAKVİMİ OKU) WRITE_CALENDAR (TAKVİME YAZ)
CAMERA (KAMERA)	CAMERA (KAMERA)
CONTACTS (REHBER)	READ_CONTACTS (REHBERİ OKU) WRITE_CONTACTS (REHBERE YAZ) GET_ACCOUNTS (REHBERİ EDİN)
LOCATION (KONUM)	ACCESS_FINE_LOCATION (NET KONUM) ACCESS_COARSE_LOCATION (YAKLAŞIK KONUM)
MICROPHONE (MİKROFON)	RECORD_AUDIO (SES KAYDET)
PHONE (TELEFON)	READ_PHONE_STATE (TELEFON DURUMUNU OKU) CALL_PHONE (ÇAĞRI YAP) READ_CALL_LOG (ÇAĞRI KAYITLARINI OKU) WRITE_CALL_LOG (ÇAĞRI KAYITLARINA YAZ) ADD_VOICEMAIL (SESLİ MESAJ EKLE) USE_SIP (SIP-OTURUM BAŞLATMA PROTOKOLÜNÜ KULLAN) PROCESS_OUTGOING_CALLS (GİDEN ÇAĞRILARI İŞLE)
SENSORS (SENSÖRLER)	BODY_SENSORS (SENSÖRLERİ KULLAN)
SMS (KISA MESAJ SERVİSİ)	SEND_SMS (SMS GÖNDER) RECEIVE_SMS (SMS AL) READ_SMS (SMS OKU) RECEIVE_WAP_PUSH (WAP MESAJLARINI OKU) RECEIVE_MMS (MMS AL)
STORAGE (DEPOLAMA)	READ_EXTERNAL_STORAGE (HARİCİ HAFIZAYI OKU) WRITE_EXTERNAL_STORAGE (HARİCİ HAFIZAYA YAZ)

Takvim, kamera, rehber, konum, mikrofon, telefon, sensörler, SMS ve depolama ile ilgili izinler tehlikeli izinler olarak tanımlanmıştır. Android 6.0 ve sonraki sürümlerde, uygulamaların Tablo 1'de tanımlanan bir izini kullanabilmesi için daha öncesinde kullanıcının onay vermiş olması gerekmektedir [9].

Bu çalışma, Android KYU'ların tespiti amacıyla statik analiz yöntemlerinden faydalanılmıştır. Bu kapsamda destek vektör makinesi (DVM- Support Vector Machine), rastgele orman (RO – Random Forest), Naïve Bayes (NB), K en yakın komşu (K-EYK - K Nearest Neighbors) makine öğrenmesi algoritmaları kullanılmıştır. Bu algoritmalar, literatürdeki pek çok ilgili çalışmada kullanılmış ve başarılı sonuçlar sağlamışlardır.

DVM [10], sınıflandırma ve eğri uydurma (regresyon) analizi için denetimli bir makine öğrenmesi yöntemidir. DVM'nin temel amacı, veriyi iki parçaya ayıran en iyi hiperdüzlemi bulmaktır. DVM modeli boşlukta noktalarla temsil edilen örneklerden oluşmaktadır. Farklı kategorilerdeki örnekler hiperdüzlem aracılığıyla bölünmektedirler. Bu hiperdüzlem daima iki bölge arasındaki aralığı maksimize etmektedir. Bu aralık iki bölgenin örnekleri arasındaki maksimum uzaklıkla tanımlanmakta ve ayrı bölgelerde yer alan en yakın örnekler arasındaki mesafeye göre hesaplanmaktadır. Aralık belirlendikten sonra test örnekleri de aynı boşluk üzerinde haritalanmaktadır. Sonuç olarak test örneklerinin hiperdüzlemin hangi tarafında yer aldığına göre, ilgili örneğin ait olduğu kategori tahmin edilmektedir.

RO algoritması [11], bir karar ağacı sınıflandırma yöntemidir. Karar ağacı sınıflandırmada bir ağaç yapısı oluşturulmaktadır. Bu ağacın her bir ara düğümü ile öznitelikler sınanmakta ve sınama sonucu dallarla ifade edilmektedir. Ağacın yaprakları ise sınıfları temsil etmektedir. RO algoritmasında ise özetle, sınıflandırma yapılırken birden fazla karar ağacı kullanılarak sınıflandırma başarısı arttırılmaya çalışılmaktadır.

NB sınıflandırıcısı [12], veri madenciliğinde kullanılan Bayesian teoremine dayalı, temelde olasılığa dayalı basit bir sınıflandırma yöntemidir. NB sınıflandırıcısında öznitelikler arasında herhangi bir bağımlılık olmadığı kabul edilmektedir. Test kümesinden bir örnek verildiğinde, eğitim kümesindeki veriler üzerinde olasılık hesaplamaları yapılmaktadır. Eğitim kümesi baz alınarak yapılan hesaplama sonucunda, verilen örneğin olasılığı en büyük olan kategoriye ya da sınıfa ait olduğu belirlenmektedir.

K-EYK algoritması [13], verilerin dağılımından bağımsız olarak sınıflandırma ve eğri uydurma için kullanılan istatistiksel bir yöntemdir. K-EYK, öğrenme kümesinde yer alan örneklerle test kümesinde yer alan örnekler arasındaki uzaklığı ölçümleyerek test örneğinin sınıfını belirlemektedir. Algoritmanın adında da yer alan K değeri test örneğine en yakın kaç adet komşu örneğin ele alınacağını göstermektedir ve kullanıcı tarafından belirlenmektedir. Test örneğine en yakın K adet komşu örnek Öklid bağıntıları vb. yöntemler yardımıyla tespit edilmektedir.

Bu çalışmada literatürde yer alan çalışmalardan farklı olarak, yüzlerce Android izni arasından yalnızca tehlikeli izinler kullanılmıştır. Android 6.0 (API seviyesi 23) sürümüyle birlikte Android izinleri koruma seviyelerine göre sınıflandırılmıştır. Bu çalışma için en önemlileri normal ve tehlikeli sınıfındaki izinlerdir. Normal sınıfındaki izinlerde kullanıcı verilerinin gizliliği ve diğer uygulamaların işlemleriyle ilgili riskler en düşük seviyededir. Normal sınıfına ait izinler için sistem uygulamayı yükleme anında otomatik olarak onay vermekte ve kullanıcıdan izin talep etmemektedir. Tehlikeli sınıfa giren izinlerde ise kişisel verilere, kritik kaynaklara ve diğer uygulamaların süreçlerine erişim ve müdahale olasılığı en yüksek seviyededir. Tehlikeli izinlere kullanıcı tarafından onay verilmediği takdirde uygulamanın ilgili özellikleri etkinleştirilmemektedir.

Bu çalışmada Android Malware Genome Projesinden indirilmiş 105 adet KYU, Google Play Store üzerinden indirilmiş 124 adet normal uygulama kullanılmıştır. Çalışma kapsamında kullanılan Android uygulamalarının sınıflandırma yöntemleri için uygun olduğu tespit edilmiştir.

Çalışmanın ikinci bölümde literatürde bulanan çalışmalardan bahsedilmiştir. Üçüncü bölümünde kullanılan materyal ve metotlar sunulmuştur. Dördüncü bölümünde deneysel çalışma sonucu elde edilen bulgular paylaşılmıştır. Beşinci bölümünde ise çalışmadan elde edilen sonuçlara değinilmiştir.

2. İlgili Çalışmalar

Literatürde yapılan çalışmalara göre, Android İS'ne sahip cihazlardaki KYU'ların tespiti için statik analiz, dinamik analiz ve hibrit analiz yöntemleri kullanılmaktawdır. Statik analiz yöntemi, Android uygulamalarının akıllı mobil cihaza yüklenmeden analiz edilmesini sağlamaktadır. Dinamik analizde ise uygulama, çalışma zamanında incelenmektedir. Dinamik analizle uygulamanın akıllı mobil cihazlara yüklendikten sonraki durumu incelenerek uygulamanın İS ve ağ üzerindeki etkileri izlenmektedir. Bu nedenle dinamik analizle, statik analiz yöntemiyle ortaya çıkarılamayacak karışıklıktaki KYU'lar ortaya çıkarılabilmektedir.

Statik ve dinamik analiz yöntemlerinin bir arada kullanıldığı KYU tespit yöntemine ise hibrit analiz adı verilmiştir. Statik analiz yönteminin diğer yöntemlere göre avantajı, KYU'nun kullanıcıya herhangi bir zarar veremeden engellenmesini sağlamasıdır. Ayrıca diğer Android KYU analiz yöntemlerine göre hızlı sonuç verme ve düşük maliyete sahip olma noktalarında da avantajlıdır.

Statik analize ilişkin literatürde yer alan güncel çalışmalar incelendiğinde, Xing Liu vd. [14] tarafından yapılan çalışmada, uygulama yüklenirken talep edilen izinler "istenilen izin", uygulama çalışırken kullanılan izinler ise "kullanılan izin" olarak adlandırılmıştır. Bu izinler esas alınarak makine öğrenmesi teknikleri yardımıyla KYU tespiti gerçekleştirilmiştir. Shuang Liang vd. [15] tarafından yapılan çalışmada Android KYU tespiti için izin kombinasyonu tabanlı bir yöntem sunulmuştur. Çevrimdışı çalışan Droid Detective aracı geliştirilerek

çoğunlukla kötücül uygulamaların talep ettiği izin kombinasyonları elde edilmiştir. Droid Detective aracılığıyla bu izin kombinasyonlarına göre kural kümeleri üretilmiş ve bunlara karşılık gelen kötücül davranışlar arasındaki bağlantı gösterilmiştir. Zhao Xiaoyan vd. [16] tarafından gerçekleştirilen çalışmada, izinler manifest dosyasından çıkarıldıktan sonra, öznitelik seçimi için Temel Bileşen Analizi (Principal Component Analysis) algoritması kullanılarak sınıflandırıcının yapacağı matematiksel işlemler azaltılmıştır. Toplanan verinin normal ya da kötücül olduğunu tespit için DVM kullanılmıştır. DVM, eğitim örneklerinin sınırlı olması durumunda bağımsız test kümeleri için daha düşük hata oranıyla sonuca götürmektedir. Wen Liu [17] tarafından yapılan çalışmada KYU'ların farklı amaçlara sahip olduğundan hareket edilmiştir. Tüm KYU türlerinin özelliklerini tespit etmek için yalnızca tek bir sınıflandırıcı kullanılmasının yeterli olmayacağı vurgulanarak Çoklu Sınıflandırıcı Sistemi (Multiple Classifier System) ile çözüm önermişlerdir. Bu çalışmada casus, SMS gönderme ve cihaz kontrol yazılımları olmak üzere üç türdeki KYU'lara odaklanılmıştır. Doğruluğu arttırmak amacıyla her bir sınıflandırıcı için öznitelik seçimi olarak bilgi kazancı (information gain) metodu uygulanmıştır. Wei Wang vd. [18] tarafından yapılan çalışmada, KYU'ların tespitinin yanı sıra Android uygulamaların da sınıflandırılması gerekliliği vurgulanmıştır. Uygulama marketlerinin genellikle uygulamaları, geliştiricisinin belirttiği kategoriye göre ya da tanıtımında verilen bilgiye göre kategorize ettiği ifade edilmiştir. Bu durumun da tespit sistemlerine yakalanmamak için KYU geliştirici tarafından kolayca manipüle edilebildiği belirtilmiştir. Çalışmada, APK dosyalarındaki özniteliklerden faydalanılarak uygulamanın davranışı karakterize edilmektedir. Tespit ve sınıflandırmanın verimliliğini arttırmak için DVM kullanılarak öznitelikler önemine göre sınıflandırılmıştır. Ardından DVM, RO, NB, K-EYK ile sınıflandırma ve regresyon ağaçları (SRA - Classification and Regression Tree) yöntemlerinden oluşan sınıflandırıcı topluluğu kullanılarak KYU'lar tespit edilmiştir. Ayrıca, normal uygulamaların ise hangi kategoride olduğu belirlenmiştir.

Dinamik analize ilişkin literatürde yer alan çalışmalar incelendiğinde, Zhenlong Yuan vd. [19] tarafından yapılan çalışmada DroidDetector isimli çevrimiçi çalışan derin öğrenme tabanlı bir sistem geliştirilmiştir [20]. DroidDetector, .APK dosyasını inceleyerek uygulamanın doğruluğunu, bütünlüğünü ve meşruluğunu kontrol etmektedir. DroidDetector statik analiz yöntemiyle uygulama tarafından kullanılan izinleri ve hassas uygulama programlama arayüzlerini (UPA, Application Programming Interface) arka planda incelemektedir. Dinamik analiz aşamasında ise DroidDetector, DroidBox sandbox yardımıyla test edilen uygulamayı yüklemekte ve bir müddet uygulamayı çalıştırarak dinamik analiz gerçekleştirmektedir. Statik analiz ve dinamik analiz yöntemleriyle elde edilen veriler kullanılarak uygulamanın kötücül olup olmadığı derin öğrenme modeli kullanılarak tespit edilmektedir. Kurniawan vd. [21] tarafından yapılan çalışmada, Android platformu içerisinde çalışabilen bir uygulama olan Logger geliştirilmiştir. Logger her dakika için kullanılan internet trafiğini, kullanılan batarya yüzdesini ve batarya sıcaklığını tespit ederek merkezi sunucuya göndermektedir. Merkezi sunucu, imza veri tabanındaki verilere göre uygulamanın kötücül olup olmadığını belirleyerek istemciye yanıt vermektedir. Öznitelik kümesi olarak toplanan bilgiler Naive Bayes, J48 karar ağacı ve Random Forest algoritmalarıyla açık kaynak kodlu bir veri madenciliği uygulaması olan WEKA ile değerlendirilmiştir.

Hibrit analize ilişkin Lindorfer vd. [22] tarafından yapılan çalışmada, Android uygulamalarının risk değerlendirmesini yapabilmek için büyük ölçekli Android KYU analiz kum havuzu (sandbox) olan ANDRUBIS [23] ile birlikte çalışan MARVIN isimli bir sistem önerilmiştir. MARVIN makine öğrenmesi teknikleri yardımıyla hem statik hem de dinamik analizle çıkarılan yaklaşık 490.000 özniteliğe göre KYU tespiti gerçekleştirmektedir. MARVIN uygulaması aracılığıyla kullanıcılar mobil cihazları üzerindeki uygulamaları göndererek, bu uygulamaların ne derecede kötücül olduğunu gösteren KYU derecesini almaktadırlar. MARVIN ile yapılan statik analizlerde öznitelik olarak izinler, UPA çağrılarına dayalı izinler, yansıma UPA'ları, kriptografik UPA'lar kullanılmıştır. Dinamik analizlerde ise dosya işlemleri, ağ işlemleri, telefon olayları, veri sızıntıları, dinamik olarak yüklenen kodlar, dinamik olarak kaydedilen yayın alıcıları öznitelik olarak kullanılmıştır.

Botha vd. [24] tarafından bilgisayarlarda kullanılan antivirüs yazılımları analiz edilmiş ve bu yazılımların akıllı mobil cihazlara uygulanabilirliği değerlendirilmiştir. Antivirüs yazılımları yapıları gereği kaynak tüketimi fazladır. Diğer taraftan akıllı cihazların kaynakları ise sınırlıdır. Bu nedenle bu yazılımların mobil platformlara uygulanması etkin bir çözüm önerisi olarak değerlendirilememektedir.

3. Materyal ve Metot

Çalışmanın bu bölümünde, çalışmanın amacı ve öneminin yanı sıra sistem mimarisinin temelini oluşturan APK önişleme aşamasına ve kullanılan veri kümesine ilişkin bilgilere yer verilmiştir.

3. 1. Çalışmanın amacı

Android API 23 sürümüyle birlikte izin koruma seviyeleri tanımlanmıştır. Sistem normal izin taleplerini doğrudan onaylarken, tehlikeli izin talepleri için kullanıcı onayını şart koşmaktadır. Ancak kullanıcı farkındalığının çok düşük olması

nedeniyle, KYU tespit sistemlerine ihtiyaç devam etmektedir. Bu çalışma ile yeni izin sistemiyle uyumlu etkin bir KYU tespit sisteminin geliştirilmesini amaçlanmıştır.

3.2. Çalışmanın önemi

Daha önce yapılan KYU tespit çalışmalarında sayıları üç yüzden fazla olan izinlerin önemli bir kısmı öznitelik olarak kullanılmıştır. Android API 23 izin seviyeleri KYU tespiti çalışmaları için yeni bir zemin oluşturmuştur. Bu çalışmada, yeni izin sisteminde tehlikeli olarak nitelendirilmiş izinlerden faydalanılarak KYU tespiti yapılmıştır. Farklı makine öğrenme algoritmalarının başarımları sınanmıştır. Neticede az sayıda öznitelik kullanılarak yüksek doğruluk oranı elde edilmiştir.

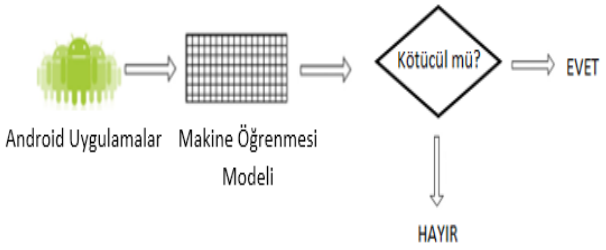
3.3. APK önişleme

Android uygulamaların akıllı cihazlara yüklenmesini sağlayan. APK dosyaları genellikle AndroidManifest.xml, classes.dex, dosyaları ile kaynak (resource) ve varlık (asset) dosyalarından oluşmaktadır. Statik analiz yapılabilmesi için. APK üzerinde tersine mühendislik yöntemleri uygulanmaktadır. APK önişleme aşamasında uygulama örnekleri kaynak koduna dönüştürülmekte ve içerisinden öznitelikler elde edilmektedir. APK dosyaları sıkıştırılmış halde bulunduklarından öncelikle açılıp sonrasında kaynak koduna dönüştürülmektedir. APK, kaynak dosyalarından birisi olan AndroidManifest.xml, Android uygulamasının yapılandırma dosyasıdır. AndroidManifest.xml, Android uygulama paketinin adını, uygulama tarafından talep edinen izinleri ve uygulamanın ihtiyaç duyduğu minimum UPA sürümü bilgilerini içermektedir. Bu çalışma kapsamında AXMLPrinter2.Jar kaynak kod dönüştürücüsü kullanılarak AndroidManifest.xml dosyası okunabilir XML haline getirilmiştir [25]. FakeNetflix kötücül yazılımına ait okunabilir XML dosyası içinde yer alan izinlerin bazıları Şekil 1'de örnek olarak verilmiştir.

```
<uses-permission
        android:name="android.permission.INTERNET"
        >
</uses-permission>
<uses-permission
        android:name="android.permission.ACCESS_NETWORK_STATE"
        >
</uses-permission>
<uses-permission
        android:name="android.permission.ACCESS_WIFI_STATE"
        >
</uses-permission>
<uses-permission
        android:name="android.permission.READ_PHONE_STATE"
        >
</uses-permission>
<uses-permission
        android:name="android.permission.WAKE_LOCK"
        >
</uses-permission>
<uses-permission
        android:name="android.permission.INJECT_EVENTS"
        >
</uses-permission>
<uses-permission
        android:name="android.permission.READ_LOGS"
        >
</uses-permission>
```

Şekil 1. FakeNetflix KYU'nun talep ettiği bazı izinler

Bu çalışmada ele alınan her bir Android uygulaması okunabilir hale getirilmiş ve AndroidManifest dosyası incelenmiştir. Bu çerçevede Android 6.0 sürümü için yayınlanmış olan toplamda 24 tehlikeli izin baz alınarak bir CSV dosyası oluşturulmuştur. Tablo 1'de gösterilen izinler, öznitelik olarak belirlenmiştir. Şekil 2'de yer alan sistem mimarisinde gösterildiği üzere, öznitelik olarak belirlenen izinlere göre oluşturulan CSV dosyası, makine öğrenmesi modelinin yapılandırılması amacıyla kullanılmıştır. Örneğin bu dosyada yer alan Facebook uygulamasına ait değerler "1,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,1,0,0,1,1,2" şeklindedir. Burada izinler ikili değer ile ifade edilmiştir. Kullanılan izinler için 1, kullanılmayan izinler için ise 0 verilmiştir. Her bir örnek için son sütunda da sınıf etiketine yer verilmiştir. Sınıf etiketi kolonunda 1 sayısı ile uygulamanın kötücül, 2 sayısıyla ise uygulamanın normal olduğu belirtilmiştir.



Şekil 2. Önerilen sistem mimarisi

3.4. Çalışmada kullanılan veri kümesi

Bu çalışma kapsamında Android Malware Genome Projesinden 105 adet KYU, Google Play Store'dan ise 124 normal uygulama toplanmıştır. 229 adet örnekten oluşturulan CSV dosyasında her bir örnek 24 niteliğe ve 1 sınıf etiketine sahiptir. 24 niteliğin her birinin tanımına ve alabileceği değerlere Tablo 2'de yer verilmiştir.

4. Bulgular

Android KYU tespiti için oluşturulan veri kümesi açık kaynak kodlu WEKA yazılımı üzerinde değerlendirilmiştir. WEKA veri madenciliği işleri için kullanılan makine öğrenmesi algoritmalarını içeren bir yazılımdır. WEKA veri önişleme, sınıflandırma, ilişki arama, kümeleme, birliktelik kuralları ve görselleştirme için çeşitli araçlar içermektedir [26]. Bu kapsamda makine öğrenmesi yöntemlerinin başarısı Doğruluk (1), TPR (True Positive Rate) (2), TNR (True Negative Rate) (3) ve F-measure (4) testlerine göre değerlendirilmiştir.

$$Doğruluk = \frac{TP+TN}{TP+TN+FP+FN} \quad (1)$$

$$TPR = \frac{TP}{TP+FN} \quad (2)$$

$$TNR = \frac{TN}{TN+FP} \quad (3)$$

$$F - measure = \frac{2.TP}{2.TP+FN+FP} \quad (4)$$

Burada, TP (True Positive), gerçekte kötücül olan Android uygulamaların kötücül olarak tespit edilme sayısını; FN (False Negative), gerçekte kötücül olan uygulamaların normal olarak tespit edilme sayısını; FP (False Positive), gerçekte normal olan uygulamaların kötücül olarak tespit edilme sayısını; TN (True Negative), gerçekte normal olan uygulamaların normal olarak tespit edilme sayısını göstermektedir.

Doğruluk ölçütü ile doğru sınıflandırılmış örnek sayısının toplam örnek sayısına olan oranı belirlenmektedir. TPR, gerçekte kötücül olan uygulamaları kötücül olarak etiketleme olasılığını, TNR ise gerçekte normal olan uygulamaları normal olarak etiketleme ihtimalini göstermektedir. TPR ve TNR değerlerinin harmonik ortalaması olan F-measure ise TPR ve TNR'nin birlikte değerlendirilmesini sağlamaktadır.

Çalışmada kullanılan 229 örnekten oluşan veri kümesi rastgele karıştırılarak %80'i öğrenme kümesi geriye kalan %20'si ise test kümesi olarak belirlenmiştir. Yani öğrenme kümesi 183 örnekten, test kümesi ise 46 örnekten oluşmaktadır.

Yapılan çalışmada normal ve kötücül uygulamalardan oluşan veri kümesi ve makine öğrenmesi algoritmaları kullanılarak sınıflandırma yapılmıştır. Kullanılan makine öğrenmesi algoritmaları içerisinde sınıflandırma işlemini en başarılı olarak yapan yöntem Doğruluk, TPR, TNR ve F-measure testlerine göre belirlenmiş olup elde edilen sonuçlara Tablo 3'te yer verilmiştir. Tablo 3'ten görüleceği üzere RO sınıflandırıcısının başarımı belirlenen metrikler bakımından diğerlerine göre daha yüksektir. Bu çalışma kapsamında Android kötücül yazılımların tespitinde, paylaşılan kriterlere göre en başarılı olan makine öğrenmesi algoritması, RO sınıflandırıcısıdır.

RO sınıflandırıcısının Android KYU tespiti için Tablo 4'te yer alan karışıklık matrisinden görüleceği üzere, gerçekte kötücül olan 20 uygulama, kötücül olarak sınıflandırılmıştır. Başka bir deyişle de karışıklık matrisinin true positive değeri 20'dir. Gerçekte kötücül olan hiçbir uygulama normal olarak sınıflandırılmamıştır. Yani karışıklık matrisindeki false negative değeri 0'dır. Gerçekte normal olan uygulamaların 2 tanesi kötücül olarak belirlenmiştir. Bu ise karışıklık matrisindeki false positive değerinin 2 olduğunu göstermektedir. Gerçekte normal olan 24 adet uygulama, normal olarak sınıflandırılmıştır. Bu da karışıklık matrisindeki true negative değerinin 24 olduğunu göstermektedir.

Tablo 2. Öznitelik olarak belirlenmiş tehlikeli izinler

Öznitelik	Öznitelik Tanımı	Değer Kümesi
1	READ_CALENDAR	0/1
2	WRITE_CALENDAR	0/1
3	CAMERA	0/1
4	READ_CONTACTS	0/1
5	WRITE_CONTACTS	0/1
6	GET_ACCOUNTS	0/1
7	ACCESS_FINE_LOCATION	0/1
8	ACCESS_COARSE_LOCATION	0/1
9	RECORD_AUDIO	0/1
10	READ_PHONE_STATE	0/1
11	CALL_PHONE	0/1
12	READ_CALL_LOG	0/1
13	WRITE_CALL_LOG	0/1
14	ADD_VOICEMAIL	0/1
15	USE_SIP	0/1
16	PROCESS_OUTGOING_CALLS	0/1
17	BODY_SENSORS	0/1
18	SEND_SMS	0/1
19	RECEIVE_SMS	0/1
20	READ_SMS	0/1
21	RECEIVE_WAP_PUSH	0/1
22	RECEIVE_MMS	0/1
23	READ_EXTERNAL_STORAGE	0/1
24	WRITE_EXTERNAL_STORAGE	0/1

Tablo 3. Sınıflandırma başarısı sonuçları

Kriter	***Doğruluk***	***TPR***	***TNR***	***F-meas.***
DVM	%84.78	%85	%84.6	%82.9
RO	**%95.65**	**%100**	**%92.3**	**%95.2**
NB	%89.13	%95	%84.6	%88.4
K-EYK	%80.43	%85	%76.9	%79.1

Tablo 4. Rastgele orman algoritmasının karışıklık matrisi

	Tahmin	
	Kötücül	İyicil
Gerçek		
Kötücül	20	0
İyicil	2	24

RO sınıflandırıcısının karışıklık matrisi maliyet açısından değerlendirildiğinde, sistemin gerçekte 20 adet KYU'dan 20 tanesini de doğru olarak tespit edebilmesi mutlak başarı olarak gösterilebilir. Ancak gerçekte 26 adet normal olan uygulamanın 24 tanesinin normal olarak belirlenmiş olmasının aksine 2 tanesinin kötücül olarak sınıflandırılmış olması kullanıcılar açısından memnuniyetsizlik oluşturacağından, sistemin geliştirilmesi gerektiğini göstermektedir. Özetle, makine öğrenmesinde FN

değerinin, FP değerinden düşük olması maliyet çerçevesinde tercih sebebi olsa da her iki sınıflandırma hatasının da minimize edilmesi önem arz etmektedir.

5. Tartışma ve Sonuç

Android İS açık kaynak kodlu yapısı nedeniyle birçok saldırı yöntemine karşı savunmasız kalmaktadır. Dahası talep edilen izinlerin kullanıcılar tarafından önemsenmeden kabul edildiği düşünüldüğünde, cihazlar üzerindeki KYU tespit sistemlerinin geliştirilerek etkin bir biçimde kullanılması önem kazanmaktadır. Bu amaçla, çalışma kapsamında toplamda 229 örnekten oluşan veri kümesi üzerinde çeşitli makine öğrenmesi algoritmalarıyla KYU tespiti yapılmıştır. Bu algoritmaların başarımı üzerine yapılan kıyaslamada RO sınıflandırıcısının daha başarılı olduğu görülmüştür. Elde edilen sonuçlardan Android KYU tespitinde makine öğrenmesi yöntemlerinin etkin bir çözüm olduğu ve geliştirilerek daha başarılı sonuçlar elde edilebileceği anlaşılmaktadır. Android KYU tespitinde, makine öğrenmesi algoritmaları kullanılarak yapılan sınıflandırma hatalarının minimize edilmesi, bu çalışmanın gelecekteki temel hedefleri arasında yer almaktadır.

Kaynakça

[1] Anonim, 2017. Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2016–2021 White Paper. http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/mobile-white-paper-c11-520862.html (Erişim Tarihi: 03.03.2017).

[2] Anonim, 2017. IDC Smartphone OS Market Share in 2017 Q1. http://www.idc.com/promo/smartphone-market-share/os (Erişim Tarihi: 11.05.2017).

[3] Anonim, 2017. Number of available applications in the Google Play Store from December 2009 to June 2017. https://www.statista.com/statistics/266210/number-of-available-applications-in-the-google-play-store/ (Erişim Tarihi: 06.07.2017).

[4] Anonim, 2017. Sensor Tower Store Intelligence Q4 2016 Data Digest. https://s3.amazonaws.com/sensortower-itunes/Quarterly+Reports/Sensor-Tower-Q4-2016-Data-Digest.pdf?src=blog (Erişim Tarihi: 18.03.2017).

[5] Snell, B. 2017. Mobile Threat Report What's on the Horizon for 2016. https://www.infopoint-security.de/medien/rp-mobile-threat-report-20161.pdf (Erişim Tarihi: 26.03.2017).

[6] Anonim, 2016. G Data Mobile Malware Report – Threat report: H1/2016. https://file.gdatasoftware.com/web/en/documents/whitepaper/G_DATA_Mobile_Malware_Report_H1_2016_EN.pdf (Erişim Tarihi: 01.04.2017).

[7] Felt, AP., Ha, E., Egelman, S., Haney, A., Chin, E., Wagner, D. 2012. Android permissions: user attention comprehension, and behavior. In: Proceedings of the eighth symposium on usable privacy and security e SOUPS '12, (2012), 3.1-3.14.

[8] Anonim, 2017. Requesting Permissions. https://developer.android.com/guide/topics/permissions/requesting.html (Erişim Tarihi: 06.05.2017).

[9] Kiraz, Ö., Doğru, İ.A. 2017. Android Malware Detection Systems Review. Düzce Univ. Journal of Science & Technology, Vol. 5 (1) (2017), 281-298.

[10] Vapnik V.N. 2000. The Nature of Static Learing Theory. Springer, 314s.

[11] Breiman L. 2001. Random Forests. Statistics Department University of California Berkeley, (2001), 1- 33.

[12] Hanson R., Stutz J. 1991. Cheeseman P. Bayesian Classification Theory. NASA Ames Research Center Artificial Intelligence Research Branch. (1991), 1-9.

[13] Fix E., Hodges J.L. 1952. Discriminatory analysis: Nonparametric discrimination: Small sample performance. Technical Report Project 21-49-004. Report Number 11, (1952), 1-20.

[14] Liu, X., Liu, J. 2014. A Two-layered Permission-based Android Malware Detection Scheme. 2nd IEEE International Conference on Mobile Cloud Computing, Services, and Engineering, Oxford, (2014), 142-148.

[15] Liang, S., Du, X. 2014. Permission-Combination-based Scheme for Android Mobile Malware Detection. IEEE International Conference on Communications (ICC), Sydney, NSW, 2301 – 2306.

[16] Xiaoyan, Z., Juan, F., Xiujuan, W., 2014. Android malware detection based on permissions. International Conference on Information and Communications Technologies (ICT 2014), Nanjing, China, 1-5.

[17] Liu, W., 2013. Mutiple classifier system based android malware detection. International Conference on Machine Learning and Cybernetics (ICMLC), Tianjin, (2013), 57-62.

[18] Wei Wang, Yuanyuan Li, Xing Wang, Jiqiang Liu, Xiangliang Zhang, 2017. Detecting Android Malicious Apps and Categorizing Benign Apps with Ensemble of Classifiers. Future Generation Computer Systems, (2017).

[19] Yuan, Z., Lu, Y., Xue, Y. 2016. DroidDetector: Android Malware Characterization and Detection Using Deep Learning. Tsinghua Sci. Tech. 21, (2016), 114-123.

[20] Anonim, 2014. DroidDetector: A deep learning based Android malware detection engine. http://analysis.droid-sec.com (Erişim Tarihi: 21.04.2017).

[21] Kurniawan, H., Rosmansyah, Y., DabarsyahAndroid, B. 2015. Android anomaly detection system using machine learning classification. In Electrical Engineering and Informatics (ICEEI), 2015 International Conference on, (2015), 288–293.

[22] Lindorfer, M., Neugschwandtner, M. and Platzer, C. 2015. Marvin: Efficient and comprehensive mobile app classification through static and dynamic analysis. In Computer Software and Applications Conference (COMPSAC), 2015 IEEE 39th Annual, volume 2, (2015), 422–433.

[23] Weichselbaum, L., Neugschwandtner, M., Lindorfer, M., Fratantonio, Y., Van der Veen, Y., Platzer, C. 2014. Andrubis: Android malware under the magnifying glass. Vienna University of Technology, (2014), Tech. Rep. TR-ISECLAB-0414-001.

[24] Botha, R.A., Furnell, S.M., Clarke, N.L. 2009. Fromdesktop to mobile: Examining the security experience. Computer & Security 28, (2009), 130–137.

[25] Galli, Enrico. Reverse Engineering Android Applications. Diss. University of Georgia, 2012.

[26] Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., & Witten, I. H. (2009). The WEKA data mining software: an update. ACM SIGKDD explorations newsletter, 11(1), 10-18.