
GAZİ ÜNİVERSİTESİ
Gazi
1926

AĞ TRAFİĞİ VE İZİNLERE DAYALI HİBRİT ANDROID KÖTÜCÜL YAZILIM ANALİZİ

Ceren ASLANALP DİNÇER

YÜKSEK LİSANS TEZİ
ADLİ BİLİŞİM ANA BİLİM DALI

GAZİ ÜNİVERSİTESİ
BİLİŞİM ENSTİTÜSÜ

HAZİRAN 2020

Ceren ASLANALP DİNÇER tarafından hazırlanan "AĞ TRAFİĞİ VE İZİNLERE DAYALI HİBRİT ANDROID KÖTÜCÜL YAZILIM ANALİZİ" adlı tez çalışması aşağıdaki jüri tarafından OY BİRLİĞİ ile Gazi Üniversitesi Adli Bilişim Ana Bilim Dalında YÜKSEK LİSANS TEZİ olarak kabul edilmiştir.

Danışman: Doç. Dr. İbrahim ALPER DOĞRU

Bilgisayar Mühendisliği, Gazi Üniversitesi

Bu tezin, kapsam ve kalite olarak Yüksek Lisans Tezi olduğunu onaylıyorum.

...……

Başkan: Prof. Dr. M. Ali AKCAYOL

Bilgisayar Mühendisliği, Gazi Üniversitesi

Bu tezin, kapsam ve kalite olarak Yüksek Lisans Tezi olduğunu onaylıyorum.

……...

Üye: Dr. Öğr. Üyesi A. Talha KABAKUŞ

Bilgisayar Mühendisliği, Düzce Üniversitesi

Bu tezin, kapsam ve kalite olarak Yüksek Lisans Tezi olduğunu onaylıyorum. ……...

Tez Savunma Tarihi: 25/06/2020

Jüri tarafından kabul edilen bu tezin Yüksek Lisans Tezi olması için gerekli şartları yerine getirdiğini onaylıyorum.

……

Doç. Dr. Aslıhan TÜFEKCİ

Bilişim Enstitüsü Müdürü

ETİK BEYAN

Gazi Üniversitesi Bilişim Enstitüsü Tez Yazım Kurallarına uygun olarak hazırladığım bu tez çalışmasında;

- Tez içinde sunduğum verileri, bilgileri ve dokümanları akademik ve etik kurallar çerçevesinde elde ettiğimi,
- Tüm bilgi, belge, değerlendirme ve sonuçları bilimsel etik ve ahlak kurallarına uygun olarak sunduğumu,
- Tez çalışmasında yararlandığım eserlerin tümüne uygun atıfta bulunarak kaynak gösterdiğimi,
- Kullanılan verilerde herhangi bir değişiklik yapmadığımı,
- Bu tezde sunduğum çalışmanın özgün olduğunu,

bildirir, aksi bir durumda aleyhime doğabilecek tüm hak kayıplarını kabullendiğimi beyan ederim.

Ceren ASLANALP DİNÇER

25/06/2020

AĞ TRAFİĞİ VE İZİNLERE DAYALI HİBRİT ANDROID KÖTÜCÜL YAZILIM ANALİZİ

(Yüksek Lisans Tezi)

Ceren ASLANALP DİNÇER

GAZİ ÜNİVERSİTESİ

BİLİŞİM ENSTİTÜSÜ

HAZİRAN 2020

ÖZET

Hızla gelişmekte olan mobil teknolojiler ile mobil cihazlar hem iş hem günlük hayatta yüksek oranda kullanılmaktadır. Kendi cihazını getir iş modeli ile beraber bu cihazlar iş ve kamu kurumları ağlarına bağlanarak, beraberinde kötücül yazılımların (malware) tüm risklerini organizasyona taşımaktadır. Kötücül davranış, bilgiye ve cihaza yetkisiz erişim dolayısıyla hem kurum hem kişiye karşı ciddi ölçüde tehdit oluşturmaya başlamıştır. Android, açık kaynak çekirdek politikasına sahip olması sebebiyle kötücül tehditlere çok daha fazla açık bir platformdur. Bu kötücül yazılımları tespit edip önlem almak için tespit mekanizmaları geliştirilmektedir. Bu çalışmada izin tabanlı ve paket analizini birlikte kullanan hibrit bir yaklaşım modeli uygulanmıştır. İzin tabanlı model ile %96,62 doğruluk oranı elde edilirken karma yaklaşım ile %97,26 doğruluk oranı elde edilmiştir.

Bilim Kodu	:	92401
Anahtar Kelimeler	:	Android, kötücül yazılım, kötücül yazılım tespiti, dinamik yaklaşım, statik yaklaşım, imza tabanlı yaklaşım, hibrit yaklaşım
Sayfa Adedi	:	43
Danışman	:	Doç. Dr. İbrahim Alper DOĞRU

NETWORK TRAFFIC AND PERMISSION-BASED HYBRID ANDROID MALWARE ANALYSIS

(M. Sc. Thesis)

Ceren ASLANALP DİNÇER

GAZİ UNIVERSITY

INFORMATICS INSTITUTE

June 2020

ABSTRACT

Due to the rapid development of mobile technologies, mobile devices are highly used in both business and daily life. Together with the Bring Your Own Device business model, these devices connect to the networks of business and public institutions and carry all the risks of malware to the organization. Malicious behavior has become a serious threat to both the organization and the person due to unauthorized access to information and equipment. Android is a much more open platform for malicious threats due to its open-source core policy. Detection mechanisms are being developed to detect and take action against these malware. In this study, a hybrid approach model using permission-based and package analysis were applied. 96.62% accuracy rate was obtained with the permission-based model and a 97.26% accuracy rate was obtained with the hybrid approach.

Science Code	:	92401
Key Words	:	Android, Android malware, static analysis, dynamic analysis, hybrid approach, tcpdump, VirusTotal
Page Number	:	43
Supervisor	:	Assoc. Prof. Dr. Ibrahim Alper DOĞRU

TEŞEKKÜR

Çalışmalarım boyunca değerli tecrübe, yardım ve katkılarıyla beni yönlendiren, teşvik eden ve bu süreçte daima güven veren çok değerli danışman hocam Sayın Doç. Dr. İbrahim Alper Doğru'ya teşekkürü borç bilirim. Manevi destekleriyle beni hiçbir zaman yalnız bırakmayan eşim, sıra arkadaşım ve meslektaşım Egemen'e, kardeşim İrem'e, Songül Ablama, arkadaşlarım Gürkan Coşkun ve Ayça Arslanhan'a sonsuz teşekkür ederim.

Babama, anneme ve babaanneme tüm emekleri ve öğrettikleri için sonsuz bir özlemle teşekkür ederim.

İÇİNDEKİLER

Sayfa

ÇİZELGELERİN LİSTESİ

Çizelge **Sayfa**

ŞEKİLLERİN LİSTESİ

Şekil **Sayfa**

KISALTMALAR

Bu çalışmada kullanılmış kısaltmalar, açıklamaları ile birlikte aşağıda sunulmuştur.

Kısaltmalar	Açıklamalar
AAPT	Android varlık paketleme aracı
API	Uygulama programlama ara yüzü
APK	Android paket seti
AMD	Android Malware Dataset
DVM	Destek vektör makinesi
KNN	K-en yakın komşu
URL	Tekdüzen kaynak konuşlandırıcıları
SMS	Kısa mesaj servisi
OTX	Open Threat Exchange
ARFF	Öznitelik ilişkisi dosya biçimi

1. GİRİŞ

Hızla gelişmekte olan mobil teknolojiler ile birlikte mobil telefonların kullanımı ve kullanım amaçlarının çeşitlendirilmesine yönelik olarak yeni uygulamalar geliştirilmekte ve mevcut olanlar güncellenmektedir. Bu uygulamalar genellikle resmi olarak çevrimiçi mağazalardan dağıtılmaktadır. Apple'ın geliştirdiği IOS platformunun resmi uygulama mağazası App Store ve Google'ın geliştirdiği Android platformunun resmi uygulama mağazası Google Play Store adıyla hizmet vermektedir. Bu mağazalar kullanıcıların yeni uygulamaları keşfetmesi ve yüklemesi için uygulama geliştiricilerine uygun bir mekân sağlamaktadır. Mobil uygulamalar sayesinde bir kuruluşun çalışanı internet erişimi olan herhangi bir noktadan işlerini yürütebilmektedir. Örneğin, e-posta hesabına erişebilmektedir ya da elektronik belge yönetimi sisteminin mobil uygulaması aracılığıyla evrak süreçlerini yönetebilmektedir. Kendi Cihazını Getir iş modeli ise günden güne kabul görmeye başlamıştır [1]. App Store" IOS platformu ve "Google Play Store" Android platformuna uygulama yüklemek için kullanılan resmi mağazalardır. Bu mağazalar kullanıcıların yeni uygulamaları keşfetmesi ve yüklemesi için uygulama geliştiricilerine uygun bir mekân sağlamaktadır. Ayrıca, özellikle Android platformlarına yönelik 3. parti uygulama marketleri bulunmaktadır [1].

Android, ayda 2 milyardan fazla cihazda aktif edilmesiyle birlikte mobil işletim sistemi marketinde lider konuma gelmiştir. IDC1 tarafından yayınlanan bir rapora göre, Android, dünyanın her yerindeki akıllı telefonların %85 oranında kullanımla küresel akıllı telefon pazarında egemendir. Google, Play Store'da bulunan mevcut tüm uygulamaları dinamik analiz yöntemiyle tarayarak kötü amaçlı uygulamaları tespit amacıyla Bouncer'ı kullanmaktadır. Bununla birlikte, Google I / O 2017 etkinliği sırasında Google Play Protect'i duyurmuştur. Google Play Protect, Google Play Store uygulamasıyla birlikte verilen her zaman açık bir hizmettir. Google Play Protect, uygulamaların güvenlik açısından güvenli kalmasını sağlamak için kurulumdan sonra bile uygulamaları otomatik olarak tarar [2].

Android'i hedef alan kötücül yazılım konusu hem kurumsal hem bireysel kullanıcılar için kritik bir problemdir. Zararlı uygulamaların sayısı her geçen gün dramatik bir biçimde atmaktadır. Mobil cihazları hedefleyen kötü amaçlı yazılımın temel amacı, cihazda depolanan kullanıcı verilerine ve bankacılık gibi hassas finansal işlemlerde kullanılan kullanıcı bilgilerine erişmektir. Mobil kötü amaçlı yazılım, virüslü dosya ekleri, Bluetooth

ile alınan dosyalar, SMS ve MMS içerik bağlantılarında kötü amaçlı kodlara bağlantılar gibi farklı şekillerde yayılabilir [3]. G DATA'nın 2016 birinci yarıyıl mobil kötücül yazılım raporuna göre; şirketin güvenlik uzmanları tarafından 2016'nın ilk yarısında 1.723.265 adet yeni Android kötücül yazılım örneği tespit edilmiştir. Ek olarak, Android kullanıcılarının sadece %13'ünün resmi Play Store'u kullandığı ve Android 6.0 sürümünde olduğu; %30'undan fazlasının ise eski Android sürümlerden biri olan Kitkat (4.4)'ı hala kullandığı tespit edilmiştir [4].

Trend Micro'nun 2018 yılı Mobil Tehdit Manzarası raporuna göre, Android geliştiricileri, 2018 yılında, güvenlik risklerini daha da azaltmak için çeşitli mekanizmalar uygulamaya koymuştur. StrongBox Keymaster bunlardan bir tanesidir. Bu mekanizma cihazların çatı güvenlik açıklarından yararlanan istismarlara karşı korumayı amaçlamaktadır [5]. Android 9 veya daha üstünü çalıştıran ve destekleyen cihazlar, bir donanım güvenlik ünitesinde bulunan Keymaster HAL'in bir uygulaması olan bu mekanizmaya sahip olabilmektedir [6]. Raporda, eklenen özelliklere rağmen Android'deki eski ve bilinen güvenlik açıkları ve istismarların hala süregelen problemler olduğu ifade edilmiştir. Örneğin, mobil bankacılık kötü amaçlı yazılımının % 98 oranında arttığı belirtilmiştir. Siber suçlular, 2 milyardan fazla olduğu tahmin edilen mobil bankacılık uygulaması kullanıcısını para kazanabilecekleri bir veri kaynağı olarak görmektedir. Geçen yıl Trend Micro MARS tarafından temin edilen 214.323 özgün mobil bankacılık truva atı örneğinin 2017'deki mobil bankacılık truva atı sayısının neredeyse iki katı olduğu belirtilmiştir [5].

Kaspersky tarafından ilk kez Mart 2018'de tespit edilen ve 2018 yılının sonlarında uluslararası bir tehdit oluşturmaya başlayan bir Android mobil bankacılık truva atı uygulaması Riltok'un arkasındaki siber suçlular popüler ücretsiz reklam hizmetlerinin isimlerini ve simgelerini taklit ederek çeşitli maskeleme ve dağıtım yöntemlerini kullanmışlardır. Kullanıcı, öncelikle popüler bir ücretsiz reklam hizmetini taklit eden sahte bir web sitesinin kötü amaçlı bağlantısını içeren bir SMS almaktadır. Orada, truva atının maskelendiği mobil uygulamanın yeni bir sürümünü indirmeleri istenmektedir. Uygulamanın yüklenmesi için, cihaz ayarlarından bilinmeyen kaynaklardan uygulamaların yüklenmesine izin vermesi gerekmektedir. Kurulum sırasında, Riltok, kullanıcıdan sahte bir uyarı göstererek AccessibilityService sınıfındaki özel özellikleri kullanmak için izin istemektedir. Kullanıcı isteği görmezden gelir veya reddederse, pencere açılmaya devam edecektir. İstenilen hakları elde ettikten sonra, truva atı cihaz ekranından kaybolmadan önce

kendini varsayılan SMS uygulaması olarak belirlemektedir (bağımsız olarak AccessibilityService'de Evet'e tıklayarak). Artık yüklenen ve kullanıcıdan gerekli izinleri alan Riltok, komut ve komuta sunucusuyla bağlantıya geçmektedir. Daha sonraki sürümlerde uygulama başladığında truva atı ek olarak tarayıcıda, kullanıcının giriş bilgilerini ve banka kartı bilgilerini girmesine izin vermek için ücretsiz reklam hizmetini taklit eden bir kimlik avı sitesi açmaktadır ve bilgileri siber suçlulara iletmektedir [7].

Popüler bir PDF yaratma uygulaması olan CamScanner Google Play'e göre 100 milyondan fazla kez kurulmuştur. Geliştiricileri, dijitalleştirilmiş belgeleri taramak ve yönetmek için bir çözüm olarak konumlandırmaktadır. Kaspersky uzmanları uygulamayı analiz etmiş ve kötücül bir bileşeni içeren bir kütüphaneyi uygulama kodunda tespit etmişlerdir. Bu kötü amaçlı yazılımın eklenmesinin nedeninin uygulama geliştiricilerin sahtekâr bir reklamcıyla olan ortaklıklarının olabileceğini ifade etmişlerdir. Bulgular Google şirketine bildirilmiş ve uygulama derhal Google Play'den kaldırılmıştır. Bu zararlı bileşenin işlevleri, kötü amaçlı yazılımın ana görevini yerine getirmektedir: kötü amaçlı sunuculardan bir bileşen yüklemek ve başlatmak. Sonuç olarak, bileşenin sahipleri virüslü cihazı kendi çıkarları doğrultusunda kullanabilmekte, kullanıcıya izinsiz reklamlar göstermekte ve abone ücreti ödeterek mobil hesaplarından para çalabilmektedirler [8].

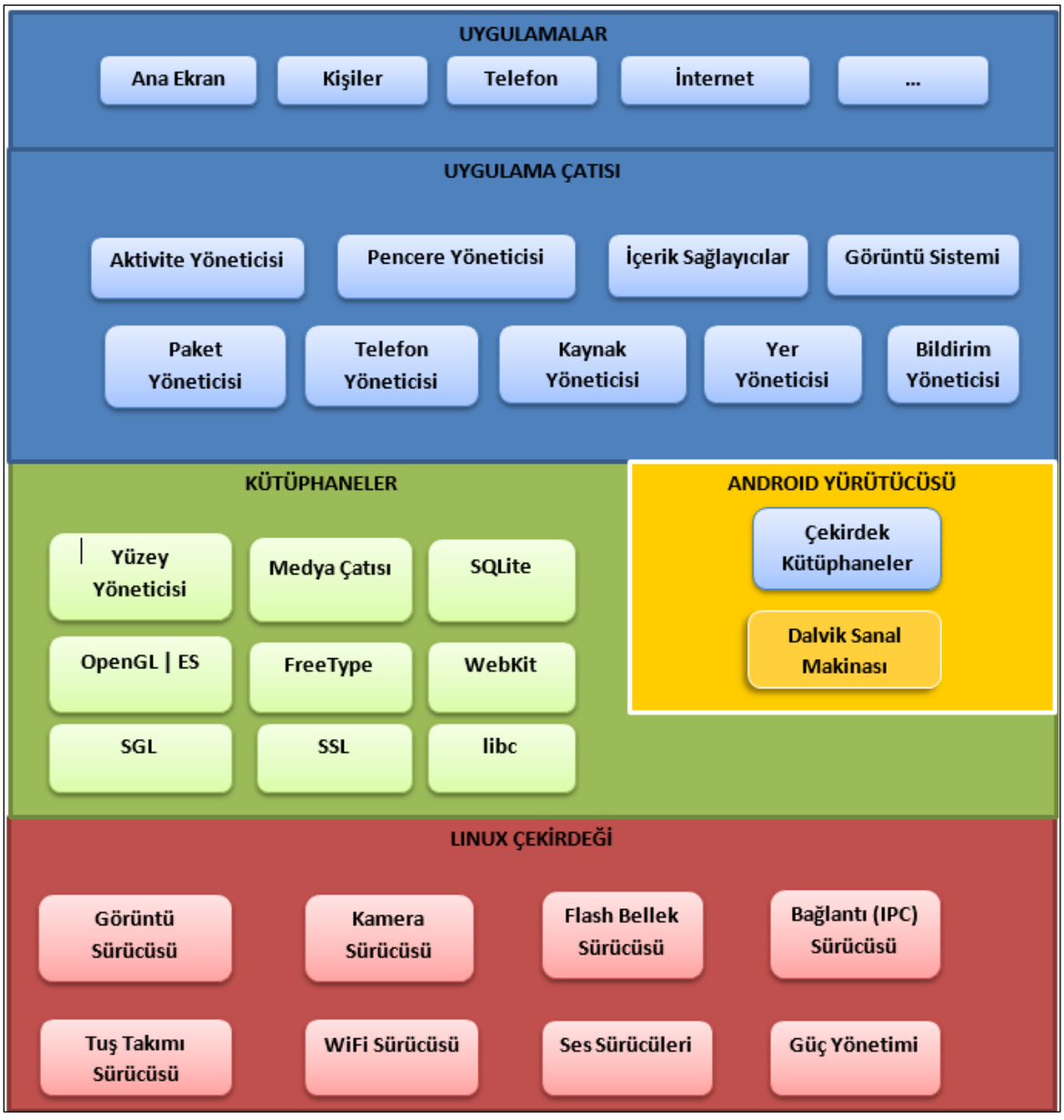
Android işletim sisteminde kötü amaçlı yazılım tespiti için genellikle iki yaklaşım bulunmaktadır: statik analiz ve dinamik analiz. Her iki yaklaşımın da kendi avantajları ve dezavantajları bulunmaktadır. Bunun yanında her ikisini de kullanıldığı karma (hibrit) yaklaşımlar mevcuttur. Bu çalışmanın temel amacı, Android uygulamalarından oluşan veri setleri üzerinden statik ve dinamik özelliklerini ortaya çıkarmak için uygulamalar yapmaktır.

Bu çalışmada, ikinci bölümde Android işletim sisteminin mimarisi anlatılmıştır. Üçüncü bölümde kötücül yazılım tespiti için önerilen farklı yaklaşımlar sunulmuştur. Dördüncü bölümde önerilen yaklaşımların çatısının oluşturulmasına yönelik öznitelik incelemeleri yapılmıştır. Beşinci bölümde ise çalışmanın sonuçlarına yer verilmiştir.

2. ANDROID İŞLETİM SİSTEMİ

Android işletim sisteminin mimari yapısı, Şekil 1'de gösterilmiş olup katmanlı yapıdan oluşan yazılım parçaları yığınıdır [9].

Android, mobil cihazlar için özelleştirilmiş Linux tabanlı mobil işletim sistemidir. Katmanlı bir sistem mimarisine sahip olan Android'in katmanları ve bileşenleri şekil 2.1'de gösterilmiştir.



Şekil 2.1. Android işletim sistemi mimarisi

Katmanların en altında Linux çekirdeği bulunmaktadır. Bu katman işlem yönetimi, bellek yönetimi, cihaz yönetimi, kamera, tuş takımı, görüntü vb. temel sistem fonksiyonlarını sağlar [9].

Linux çekirdeğinin üstünde bir kütüphane kümesi bulunur. Bunun yanında iyi bilinen libc kütüphanesi, depolama ve uygulama verisi paylaşımı için elverişli SQLite veri tabanı, ses ve video oynatmak için kullanılan kütüphaneler, internet güvenliğinden sorumlu SSL vb. bu katmanda bulunmaktadır [9].

Android Yürütücüsü, mimarinin 3. bölümünde bulunmaktadır ve Android için tasarlanmış ve optimize edilmiş bir Java Sanal Makinası olan Dalvik Sanal Makinası'nı sunar. DVM'nin sağlanması ile her bir Android uygulaması kendi sürecinde ve kendi DVM'si üzerinde çalışır. Uygulama Çatısı, Android uygulamalarına yüksek seviye Java servislerini sağlar. Uygulama geliştiriciler bu servislerden faydalanırlar [9].

Android uygulamaları en tepedeki katmanda bulunmaktadır ve uygulamalar sadece bu katmana yüklenmek üzere yazılır [9]. Android uygulamaları 4 tip bileşenden oluşur: aktiviteler, servisler, yayın alıcıları ve içerik sağlayıcılar. Bu uygulama bileşenleri uygulama kodunda sınıflar olarak uygulanır ve AndroidManifest'in içinde bildirilir. Android yazılımı uygulamalarla bu bileşenler aracılığıyla etkileşimde bulunur. Android uygulamaları Android Application Programming Interface (API) kullanılarak geliştirilmiş son kullanıcı ile etkileşimde bulunan ".apk" uzantılı sıkıştırılmış dosyalardır. Android'de, uygulamalar için Java bayt kodunun çalıştırılmasının yerine, Java bayt kodundan türetilen Dalvik baytkodunu çalıştırır. Dalvik'te çoklu sınıf dosyaları yerine tüm sınıflar tek bir .dex uzantılı dosya içine beraber paketlenir. Android uygulama paketleri classes.dex dosyasında bulunan uygulama baytkodlarını, ana kod kütüphanelerini, uygulama kaynaklarını ve AndroidManifest'i içeren jar uzantılı dosyalardır. AndroidManifest.xml uygulama paket adını, uygulama izinlerini vb. tanımları içeren XML dosyasıdır. İnsan tarafından okunabilir XML formatında yazılır ve uygulama inşası sırasında ikili XML dosyasına dönüştürülür [10].

3. KÖTÜCÜL YAZILIM TESPİTİ YAKLAŞIMLARI

Android kötücül yazılımlarının her geçen gün artmasıyla birlikte tespit yöntemleri de araştırılmakta ve geliştirilmektedir. Geleneksel kötücül yazılımdan koruma ürünleri imza temelli yöntemler kullanır ve tespit için önceden kötücül yazılımın imzasını bilmesi gerekir. Bu yüzden, bilinmeyen bir kötü amaçlı yazılım imzası imza veri tabanına eklendikten sonra tespit edilebilmektedir ve kullanıcılar yazılım imza listelerini güncel tutmalıdır. Android kötücül yazılımlarının erken tespiti için uzman sistemlere ihtiyaç duyulmaktadır. Makine öğrenmesi ve veri madenciliği teknikleri, Android kötücül yazılımların erken ve etkili tespiti ve analizi için giderek daha fazla kullanılmaktadır. Makine öğrenme yöntemleri, önceki örneklerin yardımıyla ve gelecekteki örnekler hakkında öngörülerde bulunma yoluyla akıllı sistemlerin geliştirilmesini sağlar. Veri madenciliği yöntemleri, otomasyonu yönlendirmek ve değerli verileri toplamak için makine öğrenme yöntemleriyle kullanılabilmektedir.

3.1. Statik Analiz Yaklaşımı

Bu yaklaşım kötücül yazılım tespitinin uygulamalar cihaza yüklenmeden yapılmasını sağlar. Böylelikle mobil cihaz uygulamanın kötücül işlevinden etkilenmemektedir [11]. Bu yaklaşım, uygulama çalıştırılmadan onun kötücül karakteristiklerini ve kötü kod yapılarını tespit eden hızlı ve kaynak tüketimi bakımından pahalı olmayan bir yaklaşımdır [12]. Manifest dosyası, Java kodu, dizeler, vb. ".apk" dosyasında bulunan statik özellikler kötücül yazılım tespiti için analiz edilir [13].

DroidAnalyzer, Android uygulamalarının olası zafiyetlerini ve en yüksek ayrıcalığa sahip root kullanıcısının istismarını araştıran bir statik analiz aracıdır. Çalışmada uygulama izinleri, riskli API'ler ve root kullanıcısın istismarının varlığını gösteren anahtar kelimeler incelenmiştir. Riskli izinler ve anahtar kelimeler öncelikle veri seti üzerinde tespit edilmiş, daha sonra hedef uygulamalar incelenmiştir. Uygulamaların MD5 özet değerleri ve anahtar kelimelere atanan şüphe seviyelerinin karşılaştırmasına dayalı bir algoritma kullanılmıştır. Başlangıçta root kullanıcısının istismarını içermeyen ve çalışma zamanında indirilen güncelleme saldırılarına ya da anahtar kelime tabanlı teknikleri yenebilen tekniklere karşı etkisiz kalabildiği ifade edilmiştir [14].

Xing Liu ve Jiqiang Liu, uygulama yükleme esnasında talep edilen izinleri ve kullanılan izinleri incelemeye dayalı iki katmanlı bir tespit şeması önermişlerdir. İstenen izinleri elde etmek için Manifest dosyasını, kullanılan izinleri elde etmek için Dex dosyasının analizini gerçekleştirmişlerdir. Daha sonra her istenen izin için ilgili API çağrıları, içerik sağlayıcılar ve ve Intent action dizilerine bakılmıştır. İki katmanlı modelde ilk katman iki aşamayı, ikinci katman üçüncü aşamayı içermektedir. İlk aşamada, istenen izinler özelliği ve C4.5'un WEKA uygulaması olan J48 sınıflandırıcı kullanılarak uygulamalar sınıflandırılmıştır. İkinci aşamada aynı setteki uygulamalar istenen izin çiftleri ve J48 sınıflandırıcı kullanılarak sınıflandırılmıştır. Eğer bir uygulama ilk iki aşamada zararsız olarak sınıflandırıldıysa o zararsız olarak kabul edilmektedir ya da kötücül olarak sınıflandırıldıysa kötücül olarak kabul edilmektedir. Eğer ilk iki aşamada farklı kümelerde sınıflandırıldıysa, üçüncü aşamada, başka bir kümeye konmakta ve kullanılan izinler çifti ve J48 sınıflandırıcı kullanılarak sınıflandırılmaktadır. Önerilen yaklaşım literatürdeki Kirin ve RCP+RPCP gibi diğer metotlarla kıyaslanmıştır. Önerilen metodun %98,6 doğruluk oranı ile diğer metotlara göre daha iyi sınıflandırma performansı olduğu görülmüştür [15].

Yerima ve arkadaşları, Android kötücül yazılımlarının proaktif olarak tespit edilmesi için üç farklı Bayes sınıflandırma yaklaşımını analiz etmiştir. Statik özellikler olarak izinler, kod özellikleri ve izin ve kod özelliklerinin karışımı kullanılmıştır. Bilgi Kazanç skoruna göre sıralanan ve puanlanan özellikler seçilmiştir ve karşılaştırmalı analiz için kullanılmıştır. Sınıflandırmaların sonucunda, yaklaşımın doğruluk oranı izinler kullanılarak 0,9, kod özellikleri kullanılarak 0,92 ve kod ve izin özelliklerinin karışımı kullanılarak 0,93 olarak kaydedilmiştir [16].

Shabtai ve diğerleri, Android oyun ve araçlarını sınıflandırmak için kod ve xml özelliklerinden türetilen statik özellikleri kullanarak farklı özellik seçim yöntemleriyle makine öğrenme tekniklerini uygulamışlardır. Farklı sayıdaki özelliklere, özellik seçim yöntemleri olarak Ki-kare, Fisher Skoru ve Bilgi Kazanımı yöntemleri uygulanmış ve karşılaştırma için farklı sınıflandırıcılar kullanılmıştır. Boostted Bayesian Networks ve Bilgi Kazanımı bileşimi ile 0,918 doğruluk oranı elde edilmiştir [17].

Kabakus ve diğerleri APK Auditor adlı bir izin tabanlı Android kötücül yazılım tespit sistemi önermiştir. Sistem üç bileşenden oluşmaktadır: Son kullanıcıların uygulamaları analiz etmesi için bir Android istemcisi, uygulama ve analiz sonuçları hakkında bilgi depolayan bir

imza veri tabanı ve tüm analiz işleminden sorumlu merkezi bir sunucu. Öğrenmeye dayalı analiz ve kötü amaçlı yazılım tespiti için lojistik regresyon ve kötü amaçlı yazılım eşiği puanı kullanılmıştır. Sistem uygulama örneklerini 0, 88 doğruluk oranıyla tespit ettiği ifade edilmiştir [18].

Zhao Xiaoyan ve arkadaşları, izin tabanlı hafif bir statik kötücül yazılım tespit sistemi önermişlerdir. Bu çalışmada Destek Vektör Makinesi algoritmasına dayanan bir kötücül yazılım tespit çatısı geliştirilmiştir. Çatının ön işleme modülünde her Android uygulama paketi açılarak AndroidManifest.xml dosyasından uygulamaların izin listesi alınmıştır. Tüm bu izin vektörleri orijinal özellik setini oluşturmuştur. Her uygulama için izin listesi Android sistem izinleriyle karşılaştırılmıştır. Özellik seçme modülünde temel bileşeni çıkartmak için ön işleme modülünden alınan orijinal özellik seti PCA algoritmasıyla işlenmiştir. Özellik çıkartma algoritmasının sonucunda çıkan matris özellik seti Destek Vektör Makinesini eğitmek için kullanılabilmektedir. Sınıflandırma modülünde, eğitim aşaması boyunca kötücül yazılım örnekleri ve zararsız uygulama örneklerinin özellik vektörlerinden oluşan bir eğitim seti sınıflandırıcıya sağlanmıştır. Tespit aşamasında, eğitilmiş sınıflandırıcı bilinmeyen bir uygulamanın girdi özellik vektörlerini sınıflandırabilmektedir. Özellik Veri Seti örneklerden çıkartılan özellikleri depolama ve güncelleme ile sorumludur. Destek Vektör Makinesi algoritmasının etkisini ölçmek için bu sınıflandırıcı BayesNet, NaiveBayes, NBTree, CART, RandomTree, J48 sınıflandırıcılarıyla karşılaştırılmıştır. Destek vektör makinesinin 0,908 oranıyla daha iyi sonuçlar aldığı görülmüştür [19].

Sanz ve diğerleri kötücül Android uygulamalarını tespit etmek için PUMA adlı izin tabanlı bir yöntem önermişlerdir. Zararsız ve kötücül uygulamalardan çıkarılan izinlerle çeşitli makine öğrenme algoritmaları kullanmışlardır ve Random Forest algoritması ile en iyi doğruluk oranı olarak 0,8641 oranını elde etmişlerdir [20].

Yerima ve arkadaşları statik özellikler kullanılarak paralel makine öğrenmesi sınıflandırıcılar vasıtasıyla kötücül yazılımın erken tespiti için bir metot önermişlerdir. Öğrenme aşaması için API ilişkili özellikler, uygulama izinleri, standart OS ve Android çatı komutları kullanılmıştır. Çalışmada kullanılan algoritmalar şunlardır: Karar Ağacı (ağaç-tabanlı), Simple Logistic (fonksiyon-tabanlı), Naive Bayes (olasılıksal), PART (kural-tabanlı), ve RIDOR (kural-tabanlı). Bileşik model için dayanak sonuçları elde etmek için deneylerin ilk aşaması her bir aday sınıflandırma algoritmasıyla yapılmıştır. Naive Bayes

sınıflandırıcının en düşük doğruluk oranına sahip olduğu görülmüştür. PART'ın doğruluk ve hata değerlerine bakarak en iyi performansı gösterdiği kanıtlanmıştır. Deneylerin ikinci aşamasında her bir sınıflandırıcıdan elde edilmiş sınıflandırma kararlarının paralel birleşimini içeren kombine sınıflandırma yaklaşımı incelenmiştir. En iyi doğruluk oranlarının olasılıkların çarpımı ile elde edildiği görülmüştür. Tüm eşleşmelerdeki sonuçlarının bireysel sınıflandırıcılardan daha iyi olduğu görülmüştür [21].

Suarez-Tangil ve arkadaşları, Dendroid adında, metin madenciliği ve bilgi alma tekniklerine dayalı bir yaklaşım önermişlerdir. Bu çalışmada, akıllı telefon kötücül yazılım örnekleri ve kötücül yazılım ailelerinin yazılım bileşenlerinde bulunan kod yapılarına dayalı metin madenciliği yaklaşımlarının kullanımları araştırılmıştır. Veri setinde bulunan her kötücül yazılım örneğinin kod parçaları işlenmiştir. Vektör Uzay Modeli uyarlanarak veri setindeki tüm aileler üzerinde uygulanmıştır ve böylece her aileden aile özellik vektörü elde edilebilmiştir. Ayrıca bilinmeyen örneklerin, bilinen aileler içine sınıflandırmadaki uygunluğu araştırılmıştır. Daha sonra, kötücül yazılım aileleri için evrimsel ilişkiyi göstermek amacıyla hiyerarşik kümeleme kullanımı üzerinde çalışılmıştır. Bu, aileler arasındaki ilişkileri, ortak soyların varlığını, belli kod özelliklerinin yaygınlığını analiz etmek için bir aracı analizci olmuştur. Çalışmada yapılan deney sonuçlarında yaklaşımın önemli ölçüde kesin olduğunu ve büyük boyutlardaki kötücül yazılım veri tabanları ile verimli biçimde uğraşabildiği ifade edilmiştir [22].

Kayabaşı ve Doğru, zararsız ve kötücül veri setleri üzerinden farklı makine öğrenmesi algoritmalarını değerlendirmiştir. Uygulama kodları tersine mühendislik yöntemi ile dönüştürülmüştür. Uygulamaların API çağrıları ve istenen izinler çıkartılarak zararsız ve kötücül uygulamalar için özellik kümesi belirlenmiştir. Çalışmada, sınıflandırma işlemini uygulamada başarılı olanlar seçilerek 4 adet makine öğrenmesi algoritması kullanılmıştır: KNN, Naive Bayes, ID3 ve J48. Çalışmada yapılan değerlendirmelerde en güvenilir algoritmanın KNN ve en zayıf algoritmanın ise Naive Bayes olduğu belirtilmiştir [23].

Arslan ve arkadaşları gereksiz izin talebinde bulunarak şüpheli kaynak erişimi yapabilecek Android uygulamalarını tespit etmek için, uygulamaların izin ve kod yapılarına dayalı bir yöntem önermişlerdir. Her uygulamanın Manifest dosyasında tanımlanmış (istenen) izinler Android sistem izinleri ile karşılaştırılmış, mevcutsa 1 değilse 0 değeri kullanılarak kötücül ve zararsız veri setleri için birer çizelge oluşturulmuştur. Daha sonra metin arama metodu

kullanılarak her bir iznin kaynak kodunda çağırılıp çağırılmadığı 0 veya 1 değeri kullanılarak belirlenmiştir. Her bir uygulama için, talep edilmiş fakat kullanılmamış tüm izinlerin kötücül uygulamalar içerisinde kullanılma sayısı toplanarak şüphe değeri hesaplanmıştır. Bu değer zararsız uygulama veri setinin şüphe değeri ortalamasından yüksek ise uygulama kötücül olarak belirlenmiştir. Çalışmanın sonucunda şüphe değeri yaklaşımının belli bir seviyede ayırt edici değerler elde etmeye yardımcı olduğu ifade edilmiştir. Ayrıca bu yaklaşımın tek başına yeterli olmadığı ve dinamik analiz yaklaşımı ile birlikte kullanılarak daha etkili ve doğru sonuçlar üreteceği belirtilmiştir [24].

3.2. Dinamik Analiz Yaklaşımı

Statik analiz yaklaşımından farklı olarak, mobil uygulama analizleri yürütülmesi esnasında yapılmaktadır. Statik analiz yaklaşımıyla ortaya çıkartılamayacak karışıklıktaki eksiklik veya açıklıklar ortaya çıkartılabilmektedir [11]. Uygulama, sanal bir makine veya emülatör gibi yalıtılmış bir ortamda yürütülür ve uygulamanın dinamik davranışı analiz amacıyla izlenir. Dinamik analiz, donanım kaynaklarının yüksek kullanımı nedeniyle pahalı bir yaklaşımdır. [12]. Sistem çağrıları, ağ trafiği, sistem bileşenleri, kullanıcı etkileşimleri vb. özellikler analiz edilmektedir [13].

Burguera ve diğerleri Android platformunda kötücül yazılım tespiti yapmak için Crowdroid adlı bir yaklaşım önermişlerdir. Bu uygulama Linux sistem çağrılarını gözlemlemekle ve onları ön işlenmiş bir şekilde merkezi bir sunucuya göndermektedir. Kitle kaynak kullanımı yöntemiyle kullanıcılardan kullandıkları her uygulama için kişisel olmayan ama davranışsal veriyi elde etmektedir. Daha sonra uzak sunucuda, kullanılan her uygulama için davranış veri seti oluşturmak amacıyla, veri ayrıştırılmakta ve sistem çağrı vektörü oluşturmaktadır. Son olarak, her veri seti kümeleme algoritması kullanılarak kümelenir. Bu şekilde zararsız uygulamalar ve kötücül uygulamalar arasındaki çok küçük sistem çağrı örüntüleri ayırt edilebilmiştir. Sistem çağrıları Strace aracı ile elde edilmektedir. Yapılan testlerin sonucunda, önerilen yöntemin kötücül uygulama tespiti için uygulanabilir bir yöntem olduğu belirtilmiştir. [25].

Lu ve diğerleri, Android cihazında uygulama davranışlarını gözlemleyen ve makine öğrenmesi tekniklerini uygulayarak uygulamaların zararsız ya da kötücül olduğunu tespit eden bir teknoloji önermişlerdir. Yetkisi olmadan mesaj gönderme, silme ya da mesaj

önleme, kullanıcının haberi olmadan uygulama yükleme, indirme ya da silme, kullanıcının SD kart içeriğini ele geçirme, mobil terminal hakkında bilgi edinme, kullanıcının arama geçmişi, telefon rehberi, konumu vb. kişisel bilgilerini elde etme, kullanıcı farkında olmadan kötü niyetli internet kullanımı davranışlarını analiz etmişlerdir. Naive Bayes sınıflandırmayı uygulamadan önce, etkinliğini arttırmak için özellikler Ki-Kare yöntemiyle filtrelenmiştir. Daha sonra Ki-Kare yöntemi kullanılarak ve kullanılmadan aynı veri seti üzerinde Naive Bayes sınıflandırma test edilmiştir. Kombine metodun daha düşük yanlış pozitif ve daha yüksek doğruluk oranı olduğu görülmüştür [26].

Zhou ve Jiang tarafından Android kötücül yazılım tespitine yönelik önerilen dinamik analiz yaklaşımında 49 adet kötücül yazılım ailesinden elde edilen 1260 adet Android kötücül yazılımı örneği içeren bir veri seti kullanılmıştır ve örneklerin yükleme, aktivite ve taşınan veri davranışları analiz edilmiştir. Bu çalışmanın sonucu olarak literatürde sıklıkla kullanılan Android Malware Genome Project Veri Seti oluşturulmuştur [27].

Alzaylaee ve diğerleri, DynaLog adında, Android uygulamalarını otomatik olarak analiz eden dinamik analiz tabanlı bir yapı önermişlerdir. Bu yapıda, emülator tabanlı güvenli sanal ortam analizi bileşeni, uygulamanın bir Android emülatöründe çalıştırmasını ve DroidBox aracıyla bazı üst düzey davranışlar ve özelliklerin çıkarılmasını sağlamıştır. APK orkestrasyon bileşeninde APIMonitor aracı kullanılarak DroidBox ile ayrıştırılamayan ve günlüğü tutulamayan API çağrıları gözlemlenmiştir. Orkestrasyon süreci, dex dosyasına tersine mühendislik işlemleri yapmayı ve uygulama çalışırken emülatör günlüğüne bir API çağrısı sınıfının veya metodunun varlığını izlemek için kullanılabilecek imzaları eklemeyi içermektedir. Uygulama çalışırken emülatör günlüğüne bir API çağrı sınıfının veya yönteminin varlığını izlemek için kullanılabilecek mühendislik ve imza ekleme. Davranış / özellik günlüğü kaydetme ve çıkarma bileşeninde, izlenen davranışlara veya API çağrı imzalarına karşılık gelen belirli günlük girişlerini çıkarma işlemlerini yapmaktadır. Tetikleyici / eğitim bileşeninde, MonkeyRunner aracı ile uygulamalara rastgele olaylar gönderilerek mevcut olan tüm aktiviteler ve servislerin çalıştırılması sağlanmaktadır. Kütük ayırma ve işleme bileşeninde, bu otomatik yapı ile analiz edilen her uygulama için çıkarılan özellikler okunabilir çıktı raporlarına dönüştürülmektedir. Önerilen yapıyı değerlendirmek için, kötücül uygulamalar Malgenome Projesi'nden, zararsız uygulamalar ise MCAfee laboratuvarında incelenmiş örneklerden alınarak test edilmiştir. Hem zararsız hem de kötücül yazılım örnekleri, DynaLog'un genişletilmiş özellikleri üzerinde test edildiğinde,

kötücül yazılım örnekleri ile PHONE_STATE ve BOOT_COMPLETED olayları daha sık gözlemlenmiştir. Kötücül yazılım örneklerinin %60'ı BOOT_COMPLETED olayını dinlerken, test edilen zararsız örneklerin yalnızca %15'inde bu durum gözlenmiştir. PHONE_STATE zararsız örneklerin neredeyse yarısı tarafından kullanılırken, kötü amaçlı yazılım örneklerinin %90'ından fazlası bu olayı kaydetmiştir. 905 adet kötücül yazılım örneğinde APK'lar PHONE_STATE olayını kaydetmiştir. Kötücül yazılım örneklerinde en çok gözlemlenen öznitelik olduğu ifade edilmiştir. Servicestart'ın ikinci sırayı aldığı ve 840 APK tarafından çağrıldığı belirtilmiştir. Değerlendirmeler sonucunda, önerilen yapının karmaşık Android kötücül yazılımlarının toplu tespitini yapabilecek yetenekte olduğu ifade edilmiştir [28].

Jaiswal ve diğerleri kötü niyetli ve meşru oyunlar arasında ayrım yapmak için sistem çağrısı analizine dayalı dinamik bir Android oyun kötü amaçlı yazılım algılama sistemi önermişlerdir. İyi huylu ve kopya kötü amaçlı oyunların sistem çağrısı modellerini analiz ederken, hem iyi huylu hem de kötü amaçlı yazılım uygulamaları tarafından, tespit edilmesi zor olan bazı sistem çağrıları olduğunu, ancak oyunu iyi huylu veya kötü niyetli olarak sınıflandırmada yardımcı olabilecek farklı sistem çağrısı sıklığı sergilediklerini gözlemlemişlerdir. Kopya uygulamaların kullanıcıdan yüksek düzeyde izin isteğini ve verilen izni kullanarak, daha fazla kaynak tüketmeye veya gereksiz işlemleri gerçekleştirmeye izin vererek sistemi tehlikeye attığını belirtmişlerdir. Önerilen sistem, uygulama davranışını tespit etmek için sistem çağrısı olaylarını kaydeder ve parametrelerin ve metriklerin bir birleşimini (örneğin, tetiklenen sistem çağrıları türü, sistem çağrıları sıklığı ve bunların düzeni, istenen izinlerle birlikte) kullanmaktadır. Herhangi bir yeni uygulama yüklendiğinde, izleme uygulaması uygulamayı algılamakta ve uygulama kimliği, sistem çağrısı adı, sıklığı ve çağrı süresini içeren bir sistem çağrısı özeti oluşturmak için strace komutu çağırmaktadır. Algılama sistemi, bir Memu öykünücüsüne yüklenen ve sistem çağrısı olaylarını kötü niyetli ve iyi niyetli oyunlardan yakalamak için süper kullanıcı haklarıyla sürekli olarak arka planda çalışan bir izleme uygulaması içermektedir. Bir uygulama yüklendiğinde, izleme uygulaması onu algılayıp "yetki alanı" veri tabanına bir uygulama paketi adı eklemekte ve veri tabanına örnek uygulamanın paket adıyla yeni bir uygulama eklemektedir. Sistem bir veri seti ile eğitildikten sonra, herhangi bir yeni (iyi huylu veya kötü amaçlı yazılım) oyun yüklendiğinde, izleme uygulaması onu algılamakta, sistem çağrısı olayını analiz etmekte ve uygulama davranışına karar vermektedir. Algılama sistemi çok sayıda deney yapılarak eğitilmiş, farklı ayarlarda ve farklı Android sürümlerinde

20 normal ve 40 kötü niyetli oyunun sistem çağrısı analizini gerçekleştirilmiştir. Normal oyunların sistem çağrılarını analiz etmek için Google Play mağazasından 20 ücretsiz oyun uygulaması indirilmiştir. Bu oyunlar daha sonra çalışma zamanı boyunca sistem çağrıları düzenini ve sıklıklarını kaydetmek için sistemde yürütülmüştür. Deneyler sırasında clock_gettime, epoll_wait, futex, getpid, getuid32, ioct ve read gibi bazı sistem çağrılarının diğer sistem çağrılarına göre daha sık gerçekleştiği görülmüştür. Kötü amaçlı yazılımlar VirusTotal veri ambarından alınmıştır. Bu örnekler sistemde çalıştırıldığında, hem sistem çağrısı olaylarında hem de sıklıklarında bazı önemli farklılıklar kaydedilmiştir. Deneyler sırasında, brk, bind, fchown32, fdatasync, setsockopt, getsocket, gettimeofday gibi sistem çağrılarının çoğunlukla kötü amaçlı yazılım oyunlarında bulunduğu belirtilmiştir. Ayrıca, epoll_wait, getuid, getpid, clock_gettime sistem çağrılarının yüksek sıklıkta olduğu görülmüştür [29].

Aktaş ve Sen tarafından dinamik analize dayalı bir tespit yöntemi önerilmiştir. Kötü amaçlı uygulamaların çalışma sırasındaki davranışlarını inceleyerek, bunları iyi huylu uygulamalardan ayırmak için özellikler çıkartılmış ve makine öğrenme teknikleri kullanılarak bir algılama sistemi geliştirilmiştir. Droidbox'ta uygulamalar 15 dakika boyunca çalıştırılmıştır. DroidBox, veri akışını kayıt altına alabilmek için TaintDroid bilgi akış takip sistemini kullanmaktadır. Bu davranışı kayıt altına alabilmek için Droidbox'ın Android imajının kaynak kodundaki "registerReceiver" fonksiyonuna Taint kaydı eklenerek, yeni bir imaj oluşturulmuştur. Monkey DroidBox'a entegre edilmiştir ve yazılan bir betik ile tüm uygulamalar otomatik olarak emülatöre yüklenmiş ve 15 dakika boyunca çalıştıktan sonra emülatör sıfırlanmıştır. Uygulamalar çalıştırıldıktan sonra yapılan analizler sonucu, zararsız yazılımları zararlı yazılımlardan ayırt edebilecek 20 öznitelik çıkarılmıştır. Öznitelikler çıkarıldıktan sonra, saldırı ve zararlı yazılım tespitinde sıkça kullanılan sınıflandırma algoritmaları uygulanmıştır: Random Forest, J48 (C4.5), IBK (KNN), Karar Tablosu, SVM. En yüksek başarı oranı RandomForest algoritmasıyla elde edilmiştir. Bu algoritma, %91.13 doğruluk oranına sahiptir. Yanlış tespit oranı, bir tespit sistemi için oldukça yüksektir. Bu nedenle, Google Play'den indirilen ve zararsız olarak ele alınan, fakat elde edilen model ile zararlı olarak etiketlenen uygulamalardan 10 tanesi (%6,5) manuel olarak analiz edilmiştir. Bunların veri sızdırma, /proc/ dizini altındaki meminfo, cpuinfo gibi verileri okuma gibi zararlı aktiviteler gösterdiği görülmüştür. bu çalışmada Google Play'deki en popüler uygulamaların alındığı ve çoğu anti-virüs sisteminin bu uygulamaları beyaz listeler ile ele aldığı ifade edilmiştir. Bunun nedeninin, bu uygulamalardan bazılarının gri

uygulama olarak adlandırılan şüpheli davranışlar gösterdiği, fakat kötü niyetli olmayan uygulamalar sınıfına girmesinden kaynaklandığı belirtilmiştir. Güncelleme saldırısı yaptığı bilinen zararlı davranışlarını çalışma anında indiren ve çalıştıran uygulamalar için çeşitli ailelerden (DroidKungFu, Triada, Shedun, Smsreg, Rootnik, Clicker, Igexin) 50 örnekli bir test kümesi oluşturulmuş, daha sonra bu küme Random Forest algoritması ile oluşturulan model ile test edilmiştir. Bu veri kümesindeki aileler, eğitim aşamasında kullanılmamıştır. Test sonucu, bu zararlı yazılımların %88'inin başarı ile tespit edildiği sonucuna varılmıştır. Son olarak, özniteliklerin ayırt ediciliğe katkı oranlarını incelemek için Weka aracındaki bilgi kazanımı yöntemi ile yapılan değerlendirmelerde en çok katkıyı şifre çözme, anahtar oluşturma ve toplam şifreleme sayısı özniteliklerinin sağladığı gözlemlenmiştir [30].

3.3. Karma Yaklaşım

Bu yaklaşımda hem dinamik hem de statik analiz yöntemleri kullanılarak hibrit bir model oluşturulmaktadır. Statik analizin üstesinden gelemediği ancak dinamik analizde yürütülme esnasında ortaya çıkan kod karmaşıklaştırma ve saklama teknikleri hibrit çalışmaların avantajlarından birisidir. Bunun yanında dinamik analizin getirdiği yüksek zaman ve kaynak tüketimi kısıtlamaları bu yaklaşımda da geçerlidir.

Wang ve diğerleri aykırılık ve suiistimal tespitini bütünleştiren, karma bir mobil kötücül yazılım tespit sistemi önermişlerdir. Bu sistemin bileşenleri şunlardır: suiistimal bulucu statik ve dinamik analiz yöntemleriyle bilinen kötücül yazılım ya da yeni biçimlerini tespit etmekteyken, aykırılık bulucu dinamik analiz sonuçları kullanarak yeni ve bilinmeyen sıfır gün kötücül yazılımlarını tespit edebilmektedir. Karma analiz için gereken yüksek ölçekli hesaplama kaynaklarından dolayı hem suiistimal bulucu hem de aykırılık bulucunun bulut ortamı gibi bir uzak ortamda konuşlandırılması gerektiği ifade edilmiştir. Suiistimal tespitinde statik ve dinamik özellikler çıkartılmıştır. Daha sonra bu statik ve dinamik özellik dizgileri 0 ve 1 ile değerleri ile vektörlere dönüştürülmüştür. Çıkartılan özelliklerden Chi2 puanlama fonksiyonu ile özellik seçimi yapılmıştır. Daha sonra bir linearSVC sınıflandırıcı bu özellik vektörleri ile eğitilmiştir. Bilinmeyen bir uygulama sisteme verildiğinde onun özellik vektörü bu sınıflandırıcı tarafından işlenmektedir ve kötücül yazılım olup olmadığına karar verilmektedir. Eğer uygulama zararsız olarak tespit edildiyse aykırılık bulucu tetiklenmektedir. Değilse bilinen bir kötücül yazılım ailesine sınıflandırılmaktadır. Tek-sınıf Destek Vektör Makinesi sınıflandırıcı ile zararsız uygulamalar kullanılarak inşa edilmiştir.

Yeni uygulama, bu eğitilen sınıflandırıcı tarafından sıfır-gün ya da zararsız uygulama olarak etiketlenmektedir [31].

Yang ve diğerlerinin önerdiği karma yaklaşımda, etkili mobil kötücül yazılım analizi için statik analize yönelik madencilik algoritması ile dinamik analize yönelik bir kusur analizi uygulanmıştır. Bu yaklaşımda, statik analiz ile öncelikle olası saldırıları mevcut saldırı desenlerine göre belirlenmekte ve kritik yolları tespit etmektedir. Dinamik analiz ile ise tespit edilen yollar takip edilerek mevcut saldırı örnekleriyle uyumluluğu incelenmekte ve saldırı olasılığını tespit etmek için uygulama, sınırlı ve odaklanmış bir kapsamda çalıştırılmaktadır. Statik analiz aşamasında, Apriori algoritması temel alınarak geliştirilmiş DApriori algoritması çalıştırılmakta ve elde edilen küme daha önce oluşturulmuş kötücül kütüphanesiyle karşılaştırılmaktadır. Benzerliğe göre karar verilmektedir. Çalışmada, zararlı kütüphanesi oluşturmak için popüler zararlı uygulamalar seçilmiş ve örnekler Androguard ve Contagio Mobile'dan toplanmıştır. Daha sonra 12 adet mobil uygulama üzerinde DApriori çalıştırılmıştır: WhatsApp, Facebook, Zitmo, Godwon, GoldDream, Pincer, Gazon, SMSGoogle, Tele, Samsapo, FakenotifyB and Mouabad. Uygulamaların tamamının şüpheli olarak tanımlanması statik analizin bir kısıtlaması olarak ifade edilmiştir. Dinamik analiz aşamasında: öncelikle uygulama temsili bir ortamda çalıştırılmaktadır. Daha sonra, bir uygulama davranışları listesi seçilerek izlenmektedir. Aktiviteler bir günlük dosyasına yazılmakta ve kötücül niyet için analiz edilmektedir. Çalışmada statik analiz aşamasına göre 3 adet davranış izlenmek üzere seçilmiştir: sistem çalıştırılabilir yolu /system/xbin gibi kritik dizin yoluna erişim, http sunucular için alan adı erişimi, yeterli açıklama olmadan ücretlendirme. Çalışmada dinamik kusur analizi için Droidbox aracı kullanılmıştır. Bu araç yürütme sırasında hedefi ve gönderilen veriyi takip etmek için hassas API'leri izlemektedir. Çalışmada dinamik analiz Godwon uygulaması üzerinde çalıştırılmıştır ve günlük dosyası ve davranış grafiklerine göre uygulamanın zararlı olduğu sonucuna varılmıştır [32].

Kabakuş ve Doğru, hem statik hem dinamik özelliklere dayanan bir tespit yöntemi önermişlerdir. Statik analiz aşamasında izinler ve API çağrıları eşleşmeleri analiz edilmiştir. Dinamik analiz aşamasında ise GPS durumu, under/proc/UID/net/xt_qtaguid/stats altındaki günlük dosyasındaki bağlantı sayısı, veri indirme ve yükleme bakımından ağ kullanımı analiz edilmiştir. Kötücül uygulamaların, yürütmeleri sırasında mobil veri bağlantısını devre dışı bırakma eğiliminde olduğu ifade edilmiştir. İnternet üzerinden paylaşılan veri boyutu değerlendirilerek, zararsız uygulamalara göre kötücül uygulamalar için daha sınırlı olduğu

tespit edilmiştir. Zararsız uygulamaların, kötücül uygulamalara kıyasla daha fazla işlevsellik sağlamak amacıyla istenen izinler bakımından daha karmaşık olduğu belirtilmiştir. READ_SMS ve WRITE_SMS izinlerinin yalnızca kötücül uygulamalar tarafından kullanıldığından, bu izinler kötü amaçlı yazılım algılama tarafından etkin bir şekilde kullanılabilir olduğu belirtilmiştir. Ayrıcalıklı izinlerin, kötücül uygulamalarda, zararsız uygulamalara kıyasla daha yaygın olduğu tespit edilmiştir [33].

Shi ve diğerleri, mobil uygulamada güvenlik tehdidi ve saldırısını tespit etmek için statik ve dinamik analizi bir araya getiren karma bir yaklaşım önermişlerdir. Uyumlu olmayan kodlama stillerinin analizi ile kötü amaçlı yazılımları tespit etmek ve Android API'lerin maruz kaldığı güvenlik açığından yararlanarak saldırı modellerini toplamak için bir karma veri yolu izleme yöntemi uygulanmıştır. Yaklaşımda, kritik test yolu üzerinde veri durumlarının birleşimini ve yazılımın kritik test yolu üzerinde yürütülmesi incelenmiştir. Öncelikle Android API ve mevcut saldırı modellerine dayalı olası kritik yolu belirlemek için statik analiz gerçekleştirilmiştir, daha sonra programı sınırlı ve odaklanmış bir kapsamda yürütmek için bu yolu izleyen ve saldırı olasılığını tespit eden dinamik analizi gerçekleştirilmiştir ve tespit edilen yolun mevcut saldırı düzenleriyle uygunluğunun kontrol edilmiştir. Dinamik inceleme, mobil cihazlarda herhangi bir gerçek kritik ve korumalı veri kaynağına erişmeden, web tarayıcısı çerezleri gibi gizli veri sızıntısı türüne göre saldırı senaryolarının tipini raporlamaktadır [34].

Arshad ve diğerleri, dinamik analizi cihazda, statik analizi ise uzak bir sunucuda gerçekleştiren karma bir model olan SAMADroid'i önermişlerdir. Bu modelin iki ana bileşeni bulunmaktadır: dinamik analiz süreçlerinin gerçekleştiği bir son kullanıcı uygulaması ve hem statik analiz süreçlerini hem de makine öğrenmesi algoritmalarının eğitim ve test süreçlerinin gerçekleştiği bir uzak sunucu. Son kullanıcı uygulaması strace aracı ile dosya operasyonları ve ağ erişimi ile ilgili sistem çağrılarının oluşma sıklığını gözlemleyerek bir sistem çağrısı günlük dosyası oluşturmakta ve uzak sunucuya göndermektedir. Statik analiz aşamasında, uzak sunucu uygulama tanımlayıcısını almakta ve daha önce sınıflandırılıp sınıflandırılmadığını veri tabanından sorgulamaktadır. Daha önce sınıflandırıldıysa son kullanıcıya bir rapor gönderir. Eğer sınıflandırılmadıysa, ilgili uygulamayı indirerek statik analizi gerçekleştirmektedir. Donanım bileşenleri, İstenen izinler, Uygulama bileşenleri, Filtrelenmiş niyetler, Kısıtlı API çağrıları, Şüpheli API çağrısı. Altı adet statik özellik kümesinin tümü, kötü niyetli ve kötü amaçlı olmayan

uygulamalardan çıkartılarak vektör uzayına yerleştirilmektedir. Statik ve dinamik ikili özellik vektörlerini eğitmek için SVM, RF, DT ve NB algoritmaları kullanılmış ve belirtilen algoritmalar Drebin Veri Seti kullanılarak değerlendirilmiştir. Random Forest %99.07 doğruluk oranını, %0.03 olan göz ardı edilebilir bir yanlış pozitif oranı ile elde etmiştir. SVM'ye kıyasla en yüksek doğruluğu sağlamıştır, ancak düşük doğru pozitif oranına sahiptir. Dinamik analiz kısmında sistem çağrısı sıklığı vektörleri oluşturulmuştur. Veri seti çağrıları için Random Forest, Karar ağacı ve Naive Bayes uygulanmıştır. SVM ve Random Forest algoritmalarının, en düşük yanlış pozitif oranla en yüksek doğruluğu sağladığı ifade edilmiştir. Bunun yanında, mobil cihazdaki performans ve kaynak tüketimleri gözlemlenmiştir. Cihazda SAMADroid çalıştırıldıktan sonra kıyaslama değerlerinin azaldığı, SAMADroid'in diğer algılama sistemlerine kıyasla düşük kaynakları, yani belleği ve gücü tükettiğini ve kaynak kısıtlı Android cihazları için daha uygun olduğu ifade edilmiştir [35].

Alshahrani ve diğerleri cihazdaki Android zararlı uygulamalarını algılayan kullanıcı dostu bir uygulama olan DDefender'ı önermişleridir. Bu uygulama kullanıcının cihazından özellikler çıkarmak için statik ve dinamik analiz tekniklerini kullanan ve ardından sunucuda kötü amaçlı uygulamaları tespit etmek için derin öğrenme algoritması uygulayan kapsamlı bir çözümdür. İlk önce, sistem çağrılarını, sistem bilgilerini, ağ trafiklerini ve denetlenen uygulamanın istenen izinlerini çıkarmak için dinamik analiz kullanılmaktadır. Ardından, denetlenen uygulamadan uygulamanın manifest dosyasından önemli özellikleri çıkarmak için statik analiz kullanılmaktadır. Sinir ağları ve 1007 özellikten oluşan geniş bir özellik setini kullanılarak, 4208 uygulama (2104 iyi huylu uygulama ve 2104 kötü amaçlı uygulama) ile değerlendirildiği ve %95'e kadar doğruluk oranı elde edildiği belirtilmiştir [36].

4. ARAÇLAR VE YÖNTEM

Bu bölümde statik ve dinamik analiz yöntemleri ile Android uygulamalarının zararsız ya da kötücül olarak sınıflandırılmasına ilişkin kullanılan araçlara ve yöntemlere yer verilmiştir.

4.1. Veri Seti

Bu çalışmada, Android'in verimli ve karşılaştırmalı analizi için farklı makine öğrenmesi modellerinin öğrenme ve test aşamaları, 2190 adet örneğin farklı öznitelik kombinasyonları ile oluşturulan 4 adet veri setine uygulanmıştır. 1120 adet kötücül yazılım örneği AMD veri setinden elde edilirken, 1070 adet zararsız uygulama örneğinin 325 âdeti Arslan ve diğerlerinin izin tabanlı çalışmasında oluşturmuş oldukları veri setinden [37] ve geri kalanı Google Play Store'dan APKPure ve Apk Downloader web sitelerinin sağladığı web taraması metodundan yararlanılarak elde edilmiştir. AMD veri seti mevcut anti-virüs tarama sonuçları ve otomasyon teknikleri kullanılarak 24650 adet kötücül yazılım örneği içeren büyük bir veri setini, 71 kötü amaçlı yazılım ailesine ait 135 türe ayrılması ile oluşturulmuştur [38].

4.2. Sınıflandırıcılar

Tüm sınıflandırma modellerinin oluşturulması ve modellerin test edilmesi işlemleri Weka aracı kullanılarak yapılmıştır. Weka, GNU Genel Kamu Lisansı altında yayınlanan açık kaynaklı bir yazılımdır. Veri madenciliği görevlerini içeren makine öğrenmesi algoritmaları topluluğudur. Veri hazırlama, sınıflandırma, regresyon, kümeleme, birleşme kuralları madenciliği ve görselleştirme için araçlar içerir [39].

Karar ağacı, örnekleri özellik değerlerine göre sıralar ve sınıflandırır. Bir karar ağacında, her düğüm sınıflandırılacak bir örneğin bir özelliğini temsil eder ve her dal düğümün üstlenebileceği bir değeri temsil eder. Karar ağaçlarının önemli avantajlarından biri, karar ağacının neden bir örneği belirli bir sınıfa ait olarak sınıflandırdığını anlamak kolay olmasıdır [40]. Random Forest algoritması, veri kümesinin çeşitli alt örneklerinde bazı karar ağacı sınıflandırıcılarına uyan ve öngörme doğruluğunu geliştirmek ve aşırı uyumu denetlemek için ortalama kullanan bir meta tahmincisidir. Alt örneklem büyüklüğü her

zaman orijinal girdi örneklem büyüklüğüyle aynıdır, ancak önyükleme = doğru (varsayılan) ise örnekler değiştirme ile alınır [41].

Naive Bayes sınıflandırıcı, koşullu olasılığın Bayes Kuralına dayanan sınıflandırma tekniklerinden biridir. Bağımsız saf Bayes modeli, olasılıkları tahmin etmeye ve karşılaştırmaya dayanır; daha büyük olasılık, asıl etiketin daha büyük ihtimalin sınıf etiketi değeri olma ihtimalinin daha yüksek olduğunu gösterir. Naive Bayes sınıflandırıcısının en önemli avantajlarından biri, eğitim aşaması için kısa hesaplama süresine ihtiyaç duymasıdır. $X = \{x_1, x_2, \ldots x_n\}$, sınıflandırılmamış bir desen içeren bir giriş vektörünü gösterir. $P(C_i|X)$ X verildiğinde C_i hipotezinin kabul edilebilir (doğru) olma olasılığıdır. $P(X)$, örnek giriş vektörünü gözlemleme olasılığı, $P(X|C_i)$, C_i hipotezi doğru olarak verildiğinde X'i gözlemleme olasılığıdır [39]. Buna göre Eş.4.1 şu şekildedir:

$$P(C_i|X) = \frac{P(X|C_i)P(C_i)}{P(X)} \tag{4.1}$$

Destek Vektör Makinesi, literatürdeki en yeni denetimli makine öğrenme tekniğidir. DVM, iki veri sınıfını birbirinden ayıran hiper düzlemin her iki tarafındaki "marj" terimini içerir. Ayırıcı hiper düzlem ile her iki tarafındaki örnekler arasında en büyük mesafenin marjı maksimize ederek yaratılması, beklenen genelleme hatasının üst sınırını azaltır. Destek vektör makinelerinin en basit modeli, doğrusal olarak ayrılabilir girişler için uygulanan modudur. Burada, iki sınıf vardır ve bir sınıflandırıcı etiket -1 ile temsil edilir ve diğer sınıflandırıcı etiket +1 ile temsil edilir. İki boyutlu bir problem için m-boyutlu x girişleri doğrusal olarak ayrılabilirse, Eş.4.2 bu sınıfı ayırt etmek için kullanılır.

$$f(X) = w^T x + b \tag{4.2}$$

İki sınıf arasındaki destek vektörlerini en üst düzeye çıkarmak için, Eş.4.3'deki ilk ifade en aza indirilmeli ve her eğitim örneğine tüm örneklerin doğru sınıflandırılması için ikinci eşitsizlik sağlanmalıdır [42].

$$\frac{1}{2} \| w \|^2 \; minimum$$

$$y_i(w^T x_i + b) \geq 1 \;\; \forall \; eğitim\ örneği \tag{4.3}$$

Bir sınıflandırma modeli (veya sınıflandırıcı), örneklerin öngörülen sınıflara eşleşmesidir. Bazı sınıflandırma modelleri, sınıf üyeliğini tahmin etmek için farklı eşiklerin uygulanabileceği sürekli bir çıktı üretir. Diğer modeller, örneğin yalnızca öngörülen sınıfını gösteren ayrı bir sınıf etiketi üretir. Bir sınıflandırıcı ve bir örnek verildiğinde, dört olası sonuç vardır. Örnek pozitif ise ve pozitif olarak sınıflandırılmışsa, doğru pozitif olarak sayılır; eğer negatif olarak sınıflandırılmışsa, yanlış bir negatif olarak sayılır. Örnek negatifse ve negatif olarak sınıflandırılmışsa, doğru bir negatif olarak sayılır; eğer pozitif olarak sınıflandırılmışsa, yanlış bir pozitif olarak sayılır. Bir sınıflandırıcı ve bir dizi örnek (test seti) göz önüne alındığında, örneklerin sırasını temsil eden ikiye iki karışıklık matrisi oluşturulabilir. Bu matris, birçok ortak ölçümün temelini oluşturur. Gerçek sınıf ile öngörülen sınıf arasında ayrım yapmak için, bir model tarafından üretilen sınıf öngörüleri için {m, b} etiketleri kullanıldığında Şekil 4.1'de gösterilen karışıklık matrisi ortaya çıkmaktadır. Ana diyagonal boyunca sayılar alınan doğru kararları temsil eder ve bu diyagonalin sayıları çeşitli sınıflar arasındaki hataları (karışıklığı) gösterir [43].

		Gerçek sınıf	
		m	b
Hipotez sınıfı	m	Doğru Pozitifler	Yanlış Pozitifler
	b	Yanlış Negatifler	Doğru Negatifler

Şekil 4.1. Karışıklık matrisi

Oluşturulan sınıflandırıcı modellerinin performansını karşılaştırmak amacıyla şu ölçütler kullanılmıştır:

Genel Doğruluk (ACC): Pozitif ve negatif örnekler dâhil olmak üzere doğru sınıflandırılmış örneklerin yüzdesi Eş.4.4'de gösterilmiştir.

$$ACC = \frac{(TP+TN)}{(TP+TN+FP+FN)} \tag{4.4}$$

Doğru Pozitif Oranı (TPR): Doğru sınıflandırılmış pozitif örneklerin yüzdesi Eş. 4.5'de gösterilmiştir.

$$TPR = \frac{TP}{(TP+FN)} \tag{4.5}$$

Doğru Negatif Oranı (TNR): Doğru sınıflandırılmış pozitif örneklerin yüzdesi Eş. 4.6'te gösterilmiştir.

$$TNR = \frac{TN}{(TN+FP)} \tag{4.6}$$

Kesinlik (P): Sınıf olarak doğru şekilde sınıflandırılan örneklerin, sınıf olarak sınıflandırılan toplam örneklere oranı Eş. 4.7'te gösterilmiştir.

$$P = \frac{TP}{(TP+FP)} \tag{4.7}$$

F-Ölçütü (F-M): P ve TPR'nin harmonik ortalaması Eş. 4.8'te gösterilmiştir.

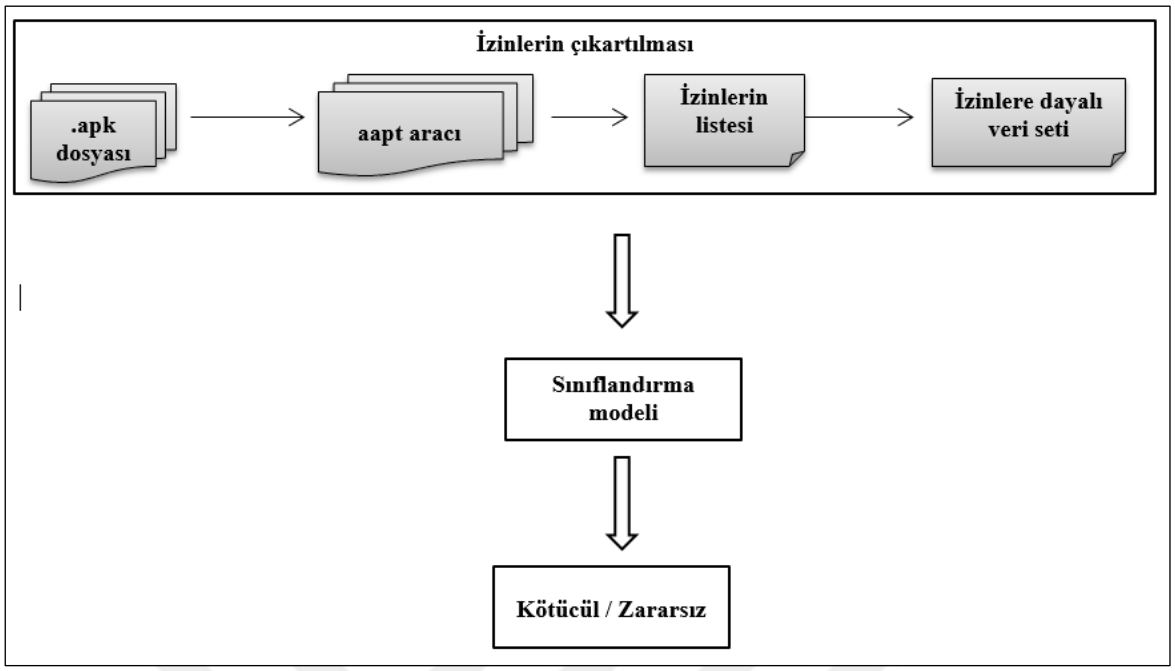
$$F - M = \frac{(2 \times P \times TPR)}{(P+TPR)} \tag{4.8}$$

4.3. Bulgular ve Tartışma

Bu bölümde öncelikle statik ve dinamik özniteliklerin çıkartılma yöntemlerine yer verilmiştir. Statik analiz yaklaşımı ve dinamik analiz yaklaşımı ile beraber oluşturulan hibrit modelin işleyişi anlatılmış ve modellerin yukarıda anlatılan kıstaslara göre test sonuçları değerlendirilmiştir.

4.3.1. Statik analiz yaklaşımı

Çalışmanın statik analiz aşamasında örneklerden alınan izinler kullanılarak bir sınıflandırma yapılmıştır. Statik analiz yönteminin işleyişi Şekil 4.2'de sunulmuştur.



Şekil 4.2. Statik analiz yöntemi

İzinler, Android işletim sisteminin en bilinen güvenlik özelliklerinden biridir. İzinler normal ve tehlikeli olarak iki koruma seviyesine ayrılmıştır. Normal izinler kullanıcının gizliliği için çok az risk taşır. Bir uygulamanın bildirim dosyasında normal bir izin bildirilirse, sistem tarafından otomatik olarak verilir. Diğer yandan, tehlikeli izinler kullanıcının depolanan verilerini veya diğer uygulamaların çalışmasını etkileme konusunda önemli bir potansiyele sahiptir. Bir Android cihaz, API seviyesi 22 veya daha düşük olan Android 5.1 OS veya daha düşük bir sürümü kullanıyorsa, sistem uygulamayı kullanıcı yüklediğinde tehlikeli izinleri ister. Bir Android cihaz, API seviyesi 23 veya daha üstünü destekleyen Android 6.0 OS veya daha üstünü kullanıyorsa, uygulama kullanıcının çalışma zamanında tehlikeli izinler almasını gerektirir ve uygulama, izinleri alıp almadıklarını her çalıştırdığında bu izinleri kontrol etmelidir. Kullanıcı izinleri istediği zaman iptal edebilir [44].

Çalışmada, her bir uygulamanın manifest dosyasından izinleri ayıklamak ve veri setini oluşturmak amacıyla Python dili kullanılarak çeşitli betikler geliştirilmiştir. Bir statik analiz özniteliği olan izinlerin uygulamalardan çıkartılması amacıyla, uygulama arşiv dosyası hiç açılmadan AAPT (Android Asset Packaging Tool) kullanılarak izinler ve paket bilgileri bir metin dosyasına aktarılmıştır. Örneğin, e633706db5bc0b015fad9951b0fd4d1e.apk uygulama paketi için Şekil 4.3'te aapt çıktısı örneği gösterilmiştir.

```
C:\>aapt dump permissions e633706db5bc0b015fad9951b0fd4d1e.apk
package: pjbang.yigou
uses-permission: name='android.permission.INTERNET'
uses-permission: name='android.permission.READ_PHONE_STATE'
uses-permission: name='android.permission.WRITE_EXTERNAL_STORAGE'
uses-permission: name='android.permission.ACCESS_NETWORK_STATE'
uses-permission:
name='com.android.launcher.permission.INSTALL_SHORTCUT'
uses-permission:
name='com.android.browser.permission.READ_HISTORY_BOOKMARKS'
uses-permission:
name='com.android.browser.permission.WRITE_HISTORY_BOOKMARKS'
```

Şekil 4.3. aapt çıktısı örneği

Her bir uygulama için izin metin dosyaları oluşturulmuştur. Hedef API seviyesi olarak API 26 belirlenmiş ve bu Android API'sine ait tüm izinlerle uygulamanın izinleri karşılaştırılarak bir veri seti dosyası oluşturulmuştur. Bu API seviyesinde toplam 146 adet sistem izni bulunmaktadır [45]. Veri setindeki zararsız uygulamalar 2016-2020 yılları arasında oluşturulmuştur ve AMD veri setinin 2010-2016 yılları arasında ortaya çıkan kötücül yazılım ailelerinden oluştuğu belirtilmiştir. Uygulama ve API seviyesi uyumsuzluğu olasılığı göz önünde bulundurularak en yakın ve en üst seviye olarak 2017 yılında sunulan Oreo 8.0'ın API seviyesi seçilmiştir. Eğer ilgili izin, uygulamanın manifest dosyasında tanımlanmışsa 1, değilse 0 değeri girilmiştir. Son olarak eğer uygulama kötücül veri setine ait ise 'm', zararsız veri setinin bir örneği ise 'b' sınıf etiketi satırın sonuna eklenerek sıradaki uygulamaya geçilmiştir. Bu yöntemle öncelikle txt uzantılı bir metin belgesine tüm uygulama izinleri işlenerek veri seti oluşturulmuştur. Şekil 4.4'te örnek bir kod bloku gösterilmiştir. Daha sonra WEKA'nın kullandığı ARFF dosya formatı kurallarına uygun olarak ilişki satırında veri setine bir isim verilmiş, özellikler değişken tipi ile birlikte alt alta sıralanmıştır. Son olarak python betikleri ile metin belgesinde oluşturulmuş veri seti eklenmiştir ve .arff uzantısı ile kaydedilerek WEKA'da okunabilir hale gelmiştir. Şekil 4.5'te dosya yapısı gösterilmiştir.

```
api26_izinleri=["ACCESS_CHECKIN_PROPERTIES", "ACCESS_COARSE_LOCATION",...]
veriseti_dosyası = open("veriseti.txt", "a")
apk_izinleri = open("apk.txt", "a")
for izin in api26_izinleri:
    if izin in apk_izinleri:
        veriseti_dosyası.write("1,")
    else:
        veriseti_dosyası.write("0,")
    veriseti_dosyası.write("m")
veriseti_dosyası.close()
```

Şekil 4.4. Veri setinin oluşturulması

```
@relation androidd
@attribute ACCESS_CHECKIN_PROPERTIES numeric
@attribute ACCESS_COARSE_LOCATION numeric
.
.
.
@attribute class {m,b}

@data
0,0,1,0,1,0,0,0,0,0,0,…,m
.
.
.
0,0,0,0,1,0,1,0,0,0,1,...,b
```

Şekil 4.5. ARFF dosya yapısı

Elde edilen veri seti üzerinden çeşitli makine öğrenmesi algoritmaları ile bir model oluşturulmuştur. Çalışmada 3 farklı algoritma ile sınıflandırma yapılmıştır: Karar ağacı, Naive Bayes ve Destek Vektör Makinesi.

Oluşturulan makine öğrenmesi modellerinin başarısını gözlemlemek için 10-kez çapraz geçerleme uygulanarak modeller test edilmiştir. Yapılan deneyler sonucunda hem doğruluk hem de doğru pozitif oranı ölçütleri bakımından en başarılı model Random Forest sınıflandırılmasıdır. Kötücül yazılım tespitinde doğruluk ölçütünün modelin genel performansını belirlemesiyle birlikte, doğru pozitif oranı m sınıfı için kötücül uygulamanın tespit başarısını göstermektir. Cihaz güvenliği bakımından zararsız bir uygulamanın engellenmesi, kötücül bir uygulamanın zararsız olarak sınıflandırıp izin verilmesinden daha güvenli olduğu düşünülmektedir. Random Forest sınıflandırma sonuçları Çizelge 4.1'de sunulmuştur.

Çizelge 4.1. Random forest sınıflandırma sonuçları

ACC(%)	TPR(%)	FPR(%)	P(%)	F-M(%)	Sınıf
96,621	97,8	4,6	95,7	96,7	m
	95,4	2,2	97,6	96,5	b

Naive Bayes sınıflandırma sonuçları Çizelge 4.2'de sunulmuştur.

Çizelge 4.2. Naive bayes sınıflandırma sonuçları

ACC(%)	TPR(%)	FPR(%)	P(%)	F-M(%)	Sınıf
83,0594	75,1	8,6	90,1	81,9	m
	91,4	24,9	77,8	84,1	b

Destek Vektör Makinesi sınıflandırma sonuçları Çizelge 4.3'te sunulmuştur.

Çizelge 4.3. Destek vektör makinesi sınıflandırma sonuçları

ACC(%)	TPR(%)	FPR(%)	P(%)	F-M(%)	Sınıf
94,8402	96	6,4	94,1	95	m
	93,6	4	95,7	94,7	b

Çizelge 4.4'te kötücül veri setinde en çok sayıda istenen ilk 15 izin, zararsız veri setinde en çok sayıda istenen ilk 15 izin Çizelge 4.5'te gösterilmiştir.

Her iki veri seti için de en çok sayıda istenen izinler incelendiğinde, neredeyse tüm kötücül uygulamaların ve zararsız uygulamaların tamamının INTERNET iznini istediği görülmektedir. Günümüzde, uygulamalar kötücül ya da zararsız olması fark etmeksizin işlevini yerine getirebilmesi için internet bağlantısına ihtiyaç duymaktadır. Telefonların geleneksel tasarımından çıkıp uygulamalarla daha fazla işlev kazanmasını başlatan gelişme mobil internet hizmetinin sağlanması ve kullanımının yaygınlaşması olmuştur. Dolayısıyla, bu izin göz ardı edilmektedir.

Kötücül veri setinde READ_PHONE_STATE izni hemen ikinci sırada yer almaktadır. INTERNET izni göz ardı edildiğinde aslında en çok istenen izin olarak değerlendirilebilmektedir. Buna karşılık zararsız veri setinde sonlara doğru yer almaktadır. Bu iznin işlevi cihazın telefon numarası, geçerli hücresel şebeke bilgileri, devam eden aramaların durumu ve cihaza kayıtlı tüm PhoneAccount'ların bir listesi dâhil olmak üzere

telefon durumuna salt okunur erişim sağlar [46]. Bu izin PHONE izin grubuna ait bir izin olmakla birlikte koruma seviyesi tehlikeli olarak tanımlanmıştır. SMS izin grubunda olan ve yine tehlikeli seviyede tanımlanmış olan izinler de kötücül veri setinde dikkat çekmektedir. SMS mesajları gönderme izni veren SEND_SMS, uygulamaya SMS mesajlarını okuma izni veren READ_SMS ve uygulamaya SMS mesajları alma izni veren RECEIVE_SMS izinleri kötücül veri setinde ilk 15'te yer alırken zararsız veri setinde bu izinler en çok istenen izinler arasında gözlemlenmemektedir.

Çizelge 4.4. Kötücül veri setinde en çok sayıda istenen ilk 15 izin

İzin	Toplam
INTERNET	1109
READ_PHONE_STATE	1099
ACCESS_NETWORK_STATE	1041
WRITE_EXTERNAL_STORAGE	804
RECEIVE_BOOT_COMPLETED	725
ACCESS_WIFI_STATE	706
GET_TASKS	529
WAKE_LOCK	513
ACCESS_COARSE_LOCATION	480
VIBRATE	473
ACCESS_FINE_LOCATION	450
READ_CONTACTS	440
READ_SMS	431
RECEIVE_SMS	411
SEND_SMS	401

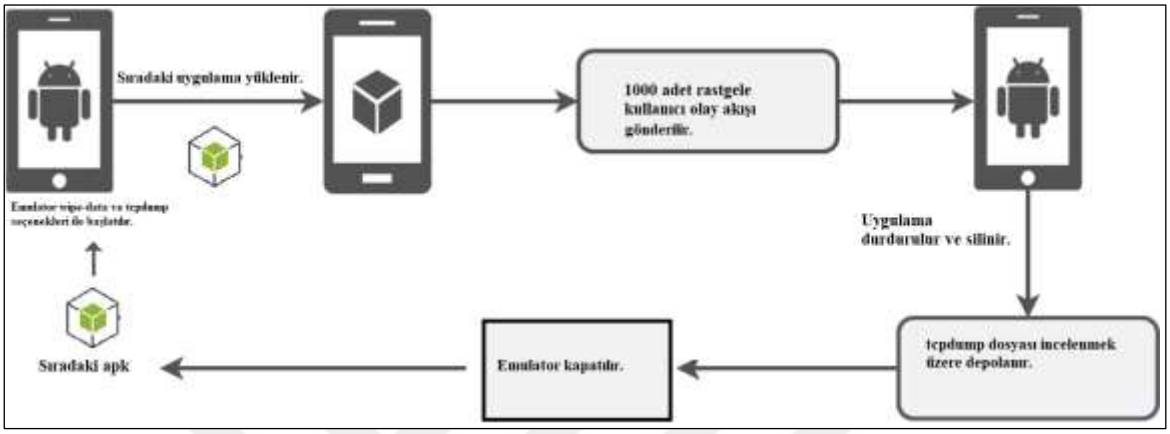
Çizelge 4.5. Zararsız veri setinde en çok sayıda istenen ilk 15 izin

İzin	Toplam
INTERNET	1060
ACCESS_NETWORK_STATE	1026
WAKE_LOCK	822
WRITE_EXTERNAL_STORAGE	804
VIBRATE	558
ACCESS_WIFI_STATE	480
READ_EXTERNAL_STORAGE	434
ACCESS_FINE_LOCATION	407
GET_ACCOUNTS	403
RECEIVE_BOOT_COMPLETED	397
ACCESS_COARSE_LOCATION	392
READ_PHONE_STATE	361
CAMERA	318
READ_CONTACTS	163
WRITE_SETTINGS	151

4.3.2. Hibrit analiz yaklaşımı

Deneysel çalışmanın bu bölümünde veri setindeki her bir uygulama Android Studio'nun bir bileşeni olan Android Emulator kullanılarak izole bir ortamda gözlenmiştir. Örneklerin ağ paket bilgilerini alabilmek amacıyla öykünücü, komut satırından 'tcpdump' seçeneği ile başlatılmıştır. Tcpdump detaylı ağ gözlemi yapabilen bir komut-satırı aracıdır [47]. Aynı zamanda, emulatörün her açıldığında ilk oluşturulduğundaki haline dönmesi ve verinin sıfırlanması amacıyla 'wipe-data' seçeneği de birlikte kullanılmıştır. Statik analizde

kullanılan izin öznitelikleriyle birlikte hibrit bir sınıflandırma yapmak amacıyla öncelikle dinamik öznitelikler çıkartılmıştır. Dinamik özniteliklerin elde edilmesi için her bir uygulama izole emülatöre yüklenerek .pcap uzantılı paket dosyası oluşturulmuştur. İşlem adımları Şekil 4.6'da sunulmuştur.



Şekil 4.6. Ağ paket dosyalarının oluşturulması

Yukarıdaki işlem adımlarını otomatik olarak uygulayabilmek için çeşitli Python betikleri yazılmıştır. MonkeyRunner aracılığıyla öykünücü cihaz uzaktan yönlendirilmiş ve uygulamalar otomatik olarak yüklenmiştir. Daha sonra, Monkey aracı ile rastgele komutlar gönderilerek analiz dosyasına ağ paketleri yazılmıştır. Tcpdump aracı ile pcap uzantılı, trafik paket analizlerinin bulunduğu dosyalar oluşturulmaktadır ve çeşitli araçlarla bu dosyaların içindeki her bir paket bilgisi incelenebilmektedir. Tcpdump'ın yanında, en yaygın kullanılan açık kaynaklı paket yoklama ve ağ izleme araçlarından bir diğeri de Wireshark'tır [48]. Bu çalışmada süreci otomatikleştirmek adına Wireshark'ın komut-satırı aracı olan 'tshark' ile betikler içinden her bir uygulamanın iletişim kurduğu IP adresleri filtrelenmiştir [49]. Bu işlem, uygulamanın iletişim kurduğu hedeflerle arasındaki trafik hakkında bilgi sahibi olmayı sağlamıştır ve olası anormalliklerin araştırılmasında kullanılmıştır. VirusTotal'de otomatikleştirilmiş bir şekilde sorgulama yapabilmek amacıyla geliştirilen betik ile her bir uygulamanın iletişim kurduğu her bir IP adresi sorgulanmış ve istek yanıtı JSON formatında alınmıştır. VirusTotal, 70'in üzerinde anti virüs tarayıcısı ve URL ya da etki alanı kara liste

hizmetleri içeren öğeleri tarayarak bulguları raporlamaya yarayan bir araçtır. Örnek bir sorgu cevabı Şekil 4.7'de gösterilmiştir.

{'https_certificate_date': 1573102205,
'asn': 16509, 'undetected_downloaded_samples': [],
'whois_timestamp': 1572868214,
'detected_downloaded_samples': [],
'resolutions': [{'last_resolved': '2019-08-21 15:32:17', 'hostname':
'adtag.ad.smaato.net'},
{'last_resolved': '2019-09-26 15:34:42', 'hostname': 'api.ad.smaato.net'},
{'last_resolved': '2019-08-01 21:52:45', 'hostname': 'sdk-
android.ad.smaato.net'},
{'last_resolved': '2019-08-06 17:30:39', 'hostname': 'soma.smaato.com'},
{'last_resolved': '2019-08-01 10:36:41', 'hostname': 'soma.smaato.net'}],
'detected_communicating_samples': [{'date': '2019-10-24 16:44:04', 'positives': 24,
'total': 71,

Şekil 4.7. Örnek VirusTotal sorgu çıktısı

Yukarıdaki örnekte VirusTotal sorgulaması sonucunda 24 pozitif tespit edildiği görülmektedir. İncelenen ağ trafiğinin ait olduğu uygulama kötücül bir hedefle iletişim kurmuştur.

VirusTotal IP adresi raporlarının içeriği VirusTotal destek sayfasında şu şekilde belirtilmiştir: "Dosya ve URL raporlarının aksine, ağ konumu görünümleri, söz konusu kaynağa ilişkin ortak kararlarını kaydetmez. Bunun yerine, bu raporlar, VirusTotal'in söz konusu kaynak için gördüğü yakın zamanda gerçekleştirdiği tüm etkinlikleri ve bununla ilgili bağlamsal bilgileri yoğunlaştırır. Bu detaylar şunları içerir:

- IP adresleri için Özerk Sistem ve yer ülkesi.

- Pasif DNS çoğaltma bilgileri: VirusTotal'ın çalışılan öğe için zaman içinde gördüğü tüm IP etki alanı adı eşlemeleri. Bu çözümler, tarama için VirusTotal'a gönderilen URL'lerle bağlantı kurulduğunda, sanal alanlardaki dosyaları çalıştırırken, üçüncü taraflarla ortaklıklar vb. Aracılığıyla gerçekleştirilir.

- Whois'ler aranır: kayıtlı kullanıcılar veya alan adı, IP adres bloğu veya özerk bir sistem gibi bir İnternet kaynağının atamaları, ancak daha geniş bir bilgi yelpazesi için de kullanılır.

- Gözlenen alt alanlar: VirusTotal'da depolanan başka bir alan altında hiyerarşik olarak görülen alanlar.

- Kardeşlik alanları: çalışılan alanla aynı hiyerarşik seviyedeki alanlar.

- URL'ler: Alan veya IP adresinde incelenen en son URL'ler. Bu bölümde yansıtılan tarihin, URL ile iletişim kurulduğu tarih değil, kaynak için sahip olduğumuz son raporun tarihi olduğunu, bu durumun alma tarihinden daha yeni veya daha eski olabileceğini unutmayın.

- İndirilen dosyalar: İncelenen alanda veya IP adresinde oturan URL'lerden en son alınan dosyalar. Bu bölümde kaydedilen tarihin, dosyanın indirildiği tarih değil, kaynak için elimizdeki son raporun tarihi olduğunu unutmayın.

- Dosyaların iletilmesi: Sanal alan içerisinde sanal bir ortamda yürütülmesiyle, ele alınan IP adresi veya etki alanı ile bir tür iletişim kurduğu görülmüş olan son dosyalar. Bu bölümde kaydedilen tarihin, iletişimin gerçekleştiği tarih değil, kaynak için sahip olduğumuz son raporun tarihi olduğunu unutmayın.

- Başvuran dosyalar: VirusTotal, hizmete sunulan dosyalarda bulunan dizeleri inceleyecek ve etki alanlarını ve IP adreslerini tanımlamak için bunlara belirli düzenli ifadeler uygulayacaktır. Bu bölüm, söz konusu etki alanına veya IP adresine başvuran dosyaları kaydeder. Bu bölümde kaydedilen tarihin, ilişkiye neden olan dosyanın

> gönderildiği tarih değil, kaynak için elimizdeki son raporun tarihi olduğunu unutmayın [50].”

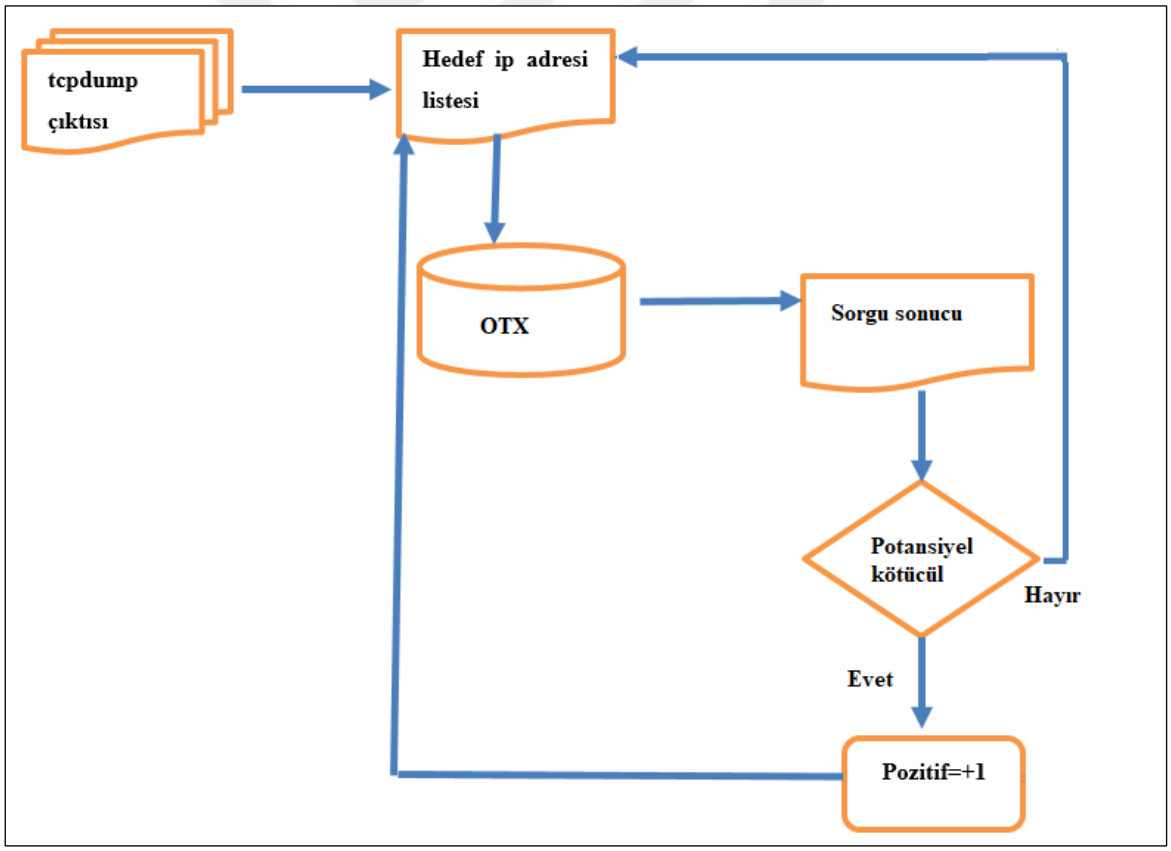
VirusTotal’ın sorgu sonuçlarına ek olarak, AlienVault tarafından sağlanan OTX Tehdit İstihbaratı veri tabanından da IP adresleri sorgulanmıştır. Open Threat Exchange, katılımcıların en son tehditler hakkında bilgi sahibi olmalarını, ortamlarında gözlenen ihmallerin araştırma göstergelerini öğrenmelerini, belirledikleri tehditleri paylaşmalarını ve ortamlarını korumak için güvenlik altyapılarını en son göstergelerle otomatik olarak güncellemelerini sağlayan açık bir topluluktur. OTX Direct Connect, tehdit göstergelerini otomatik olarak Open Threat Exchange portalından sorgulamak için bir mekanizma sağlamaktadır [5]. Bu kaynaktan her bir uygulamanın iletişim kurduğu IP adreslerini sorgulamak için OTX DirectConnect API aracılığıyla DirectConnect Python SDK kullanılarak bir betik geliştirilmiştir. Dowgin kötücül uygulama ailesine ait bir uygulamanın

dinamik analizi sonrasında çıkartılan IP adresinin gösterge sonucu Şekil 4.8'de gösterilmiştir.

```
4c27d3a08a869babf503da272a701eb0.apk
Sorgulanan ip adresi: 59.188.232.88
Identified as potentially malicious
['In pulse: SMS Phishing Android']
```

Şekil 4.8. Örnek OTX sorgu çıktısı

Bu yöntemle her bir uygulamanın iletişim kurduğu her IP adresi sorgulanmış ve potansiyel kötücül tespitlerin toplam sayısı ilgili uygulamanın toplam pozitif tespit değeri olarak hesaplanmıştır. Şekil 4.9'da hesaplama adımları gösterilmiştir.

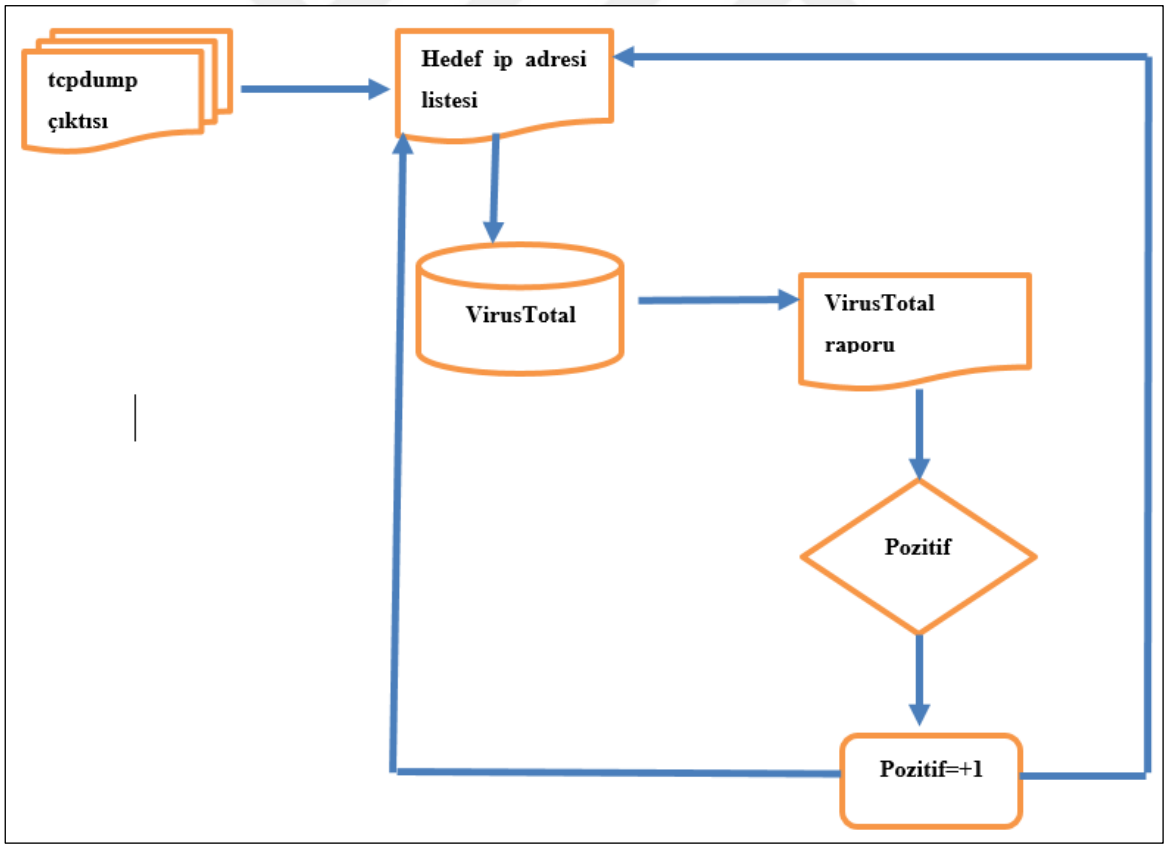


Şekil 4.9. OTX sorgu sonucu analizi

Dinamik analiz yöntemi ile ağ iletişimi incelenebilmekte ve sonrasında uygulamanın iletişim kurduğu IP adreslerinin çeşitli araçlarla taranabilmesine olanak sağlamaktadır. Bunun yanında uygulanmanın yüklenmesi, rastgele kullanıcı akışı gönderilmesi, uygulamanın

durdurulması, silinmesi ve cihazın yeniden başlatılması adımları otomatikleştirilse de işleme zamanı ve kaynak kullanımı ihtiyacı yüksektir. Örneğin, zararsız veri setindeki 116 adet uygulamanın statik ve dinamik analizleri yapılarak ön işlenmesi ve özelliklerin çıkartılması yaklaşık 2 saati aşan bir sürede tamamlanmıştır. Bu süreçte python betik çıktısındaki 'emulator: INFO: boot time' bilgisi kullanılarak her bir uygulama için ön yüklemenin tamamlanma süresinin ortalaması 35386 ms olarak hesaplanmıştır.

Deneysel çalışmanın 2. kısmında VirusTotal'den ve OTX'ten gelen sorgu sonuçlarından her bir uygulama için toplam pozitif tespit değeri bulunmuştur. Bu toplama VirusTotal'den alınan tüm IP adresi raporlarındaki tespit edilen tüm etkinlikler (detected_communicating_samples, detected_urls, detected_referrer_samples, detected_downloaded_samples) dâhil edilmiştir. Toplam pozitif değerinin hesaplanması Şekil 4.10'da gösterilmiştir.



Şekil 4.10. VirusTotal sorgu sonucu analizi

Her bir uygulama için toplam pozitif değeri hesaplandıktan sonra izin öznitelikleriyle birlikte toplam pozitif değerleri de öznitelik olarak seçilmiş ve elde edilen 3 adet veri seti

üzerinden modeller eğitilerek sınıflandırılma modeli oluşturulmuştur. Veri seti öznitelikleri sırasıyla izinler ve OTX kaynağından oluşturulan dinamik özellikler, izinler ve VirusTotal kaynağından oluşturulan dinamik özellikler ve izinler ile hem OTX hem VirusTotal kaynağından oluşturulan dinamik özellikler. Oluşturulan modeller 10 kez çapraz geçerleme ile test edilmiştir. Test sonuçları Çizelge 4.6'da gösterilmiştir.

Çizelge 4.6. İzinler ve OTX öznitelikleri ile sınıflandırma sonuçları

Naive Bayes					
ACC(%)	TPR(%)	FPR(%)	P(%)	F-M(%)	Sınıf
83,9726	77	8,7	90,3	83,1	m
	91,3	23	79,1	84,8	b
DVM					
ACC(%)	TPR(%)	FPR(%)	P(%)	F-M(%)	Sınıf
95,2055	96,4	6,1	94,3	95,4	m
	93,9	3,6	96,2	95	b
RF					
ACC(%)	TPR(%)	FPR(%)	P(%)	F-M(%)	Sınıf
96,9406	97,6	3,7	96,5	97	m
	96,3	2,4	97,4	96,9	b

İzinler ve VirusTotal kaynağından oluşturulan dinamik özellikler kullanılarak oluşturulan veri seti ile sınıflandırıcı modellerinin sonuçları Çizelge 4.7'de gösterilmiştir.

Çizelge 4.7. İzinler ve VirusTotal öznitelikleri ile sınıflandırma sonuçları

Naive Bayes					
ACC(%)	TPR(%)	FPR(%)	P(%)	F-M(%)	Sınıf
86,621	80,2	6,6	92,7	86	m
	93,4	19,8	81,8	87,2	b
DVM					
ACC(%)	TPR(%)	FPR(%)	P(%)	F-M(%)	Sınıf
95,4795	96,3	5,4	94,9	95,6	m
	94,6	3,7	96,1	95,3	b
RF					
ACC(%)	TPR(%)	FPR(%)	P(%)	F-M(%)	Sınıf
96,8037	97,1	3,6	96,6	96,9	m
	96,4	2,9	97	96,7	b

İzinler, VirusTotal ve OTX kaynağından oluşturulan dinamik özellikler kullanılarak oluşturulan veri seti ile sınıflandırıcı modellerinin sonuçları Çizelge 4.8'de gösterilmiştir.

Naive Bayes sınıflandırıcının ACC ölçütü bakımından izin tabanlı ve hibrit yaklaşımın çok ciddi fark yaratmamasının yanında TPR bakımından izinler, OTX ve VirusTotal öznitelikleri ile en yüksek sonucu verdiği görülmüştür. Sınıflandırıcı zararsız uygulamaları tespit etmekte başarılıdır fakat pozitif sınıf olan kötücül uygulama tespitinde yetersiz kalmıştır. Destek vektör makinesi sınıflandırıcısı, farklı kombinasyonlarla oluşturulmuş veri setlerinde ACC ve TPR bakımından birbirine çok yakın performans gösterirken en düşük FPR oranını izinler, OTX ve VirusTotal öznitelikleri ile aldığı görülmüştür. Random Forest sınıflandırıcı 4 farklı öznitelik kombinasyonuna sahip oluşturulmuş veri setlerinin her birinde en yüksek ve birbirine çok yakın oranlarda performans gösteren sınıflandırıcı olmuştur. Ağ trafiği izlenerek elde edilen IP adresi bilgilerinin 2 farklı istihbarat kaynağından sorgulanması ve sonuçların öznitelik olarak kullanılmasının modeli iyileştirdiği görülmüştür.

Çizelge 4.8. İzinler, OTX ve VirusTotal öznitelikleri ile sınıflandırma sonuçları

Naive Bayes					
ACC(%)	TPR(%)	FPR(%)	P(%)	F-M(%)	Sınıf
86,895	80,8	6,7	92,6	86,3	m
	93,3	19,2	82,3	86,9	b
DVM					
ACC(%)	TPR(%)	FPR(%)	P(%)	F-M(%)	Sınıf
95,5708	96,4	5,3	95	95,7	m
	94,7	3,6	96,2	95,4	b
RF					
ACC(%)	TPR(%)	FPR(%)	P(%)	F-M(%)	Sınıf
97,2603	97,8	3,3	96,9	97,3	m
	96,7	2,2	97,6	97,2	b

5. SONUÇ

Bu çalışmada literatürde yaygın olarak kullanılan Android kötücül yazılım tespit sistemleri incelenmiş ve bir uygulaması geliştirilmiştir. Yapılan deneylerde izin tabanlı statik yaklaşımın yüksek doğruluk ve doğru pozitif oranına sahip olduğu görülmüştür. İzinler derlenen veri setinde ciddi ölçüde ayırt edici olmuştur. Bunun yanında, karmaşık saldırıları tespit etmek için dinamik analiz yaklaşımına ihtiyaç duyulmaktadır. Örneğin INTERNET izni hem kötücül uygulamalarda hem de zararsız uygulamalarda yaygın olarak istenmektedir. Bu noktada cihazın internet trafiğini incelemek daha ayırt edici sonuçlar verebilmektedir. Bu çalışmada ağ paket analizi yapılmış ve hedef IP adresleri her bir uygulama için ağ paketleri dosyalarından filtrelenmiş ve hem OTX hem de VirusTotal API aracılığıyla IP adresi raporu oluşturulmuştur. Daha sonra izin öznitelikleri ile her bir uygulamanın IP adres raporundaki toplam pozitif sayısı öznitelik olarak birlikte kullanılmış ve 3 adet hibrit model oluşturulmuştur. Oluşturulan modeller test edildiğinde izin tabanlı modele göre daha yüksek doğruluk ve doğru pozitif oranına sahip olduğu görülmüştür.

Çalışmada uygulanan IP adresi analizi yöntemi VirusTotal ve OTX aracının tarama yeteneklerine bağlıdır. Bundan dolayı birden fazla araç ile tarama yapılması etkin bir kötücül yazılım tespitine katkı sağlayacaktır. Bunun yanında dinamik analizin yüksek hesaplama maliyeti ve hesaplama süresi dinamik özniteliklerin kullanıldığı modellerde bir kısıtlamadır.

Android kullanıcıları uygulamaları resmi mağazadan indirmeli ve her bir uygulamanın talep ettiği izinleri hassasiyetle incelemelidir. Uygulamaların istediği izinler şüpheli bir biçimde uygulamanın işlevi ile bağdaşmadığı düşünülüyorsa uygulamayı yüklememe ya da ilgili izinleri kabul etmeme tercihi kişiyi kötücül hedef olmaktan büyük ölçüde koruyabilecek bir son kullanıcı yöntemidir.

Kötücül uygulamaların her geçen yıl arttığı ve çeşitlendiği güvenlik şirketlerinin raporlarında vurgulanmaktadır. Daha etkin tespit sistemlerinin geliştirilmesi halen bir ihtiyaçtır. Bu bağlamda, gelecekteki çalışmalarda ağ analizi özelliklerini URL metini, sunucu adı vb. veri ile çeşitlendirerek ve ağ analizine ek olarak sistem çağrıları, bellek kullanımı gibi diğer dinamik unsurları da özniteliklere ekleyerek model çok yönlü ve daha kapsamlı bir tespit sistemi olarak iyileştirilebilir. Bu çalışmada uygulanmış geleneksel makine öğrenmesi yaklaşımlarının yanında, Android kötücül yazılım analizine yönelik

çalışmalarda yüksek eğilim olarak uygulanmaya başlanmış derin öğrenme yöntemleri ile çalışmanın entegre edilerek daha etkili ve daha güçlü bir modelin öne sürülebileceği düşünülmektedir.

KAYNAKLAR

1. He, D., Chan, S., and Guizani, M. (2015). Mobile application security: malware threats and defenses. *IEEE Wireless Communications*, *22*(1), 138-144.

2. Utku, A., ve Doğru, İ. A. (2017). Permission based detection system for android malware. *Journal of the Faculty of Engineering and Architecture of Gazi University*, *32*(4), 1015-1024.

3. Kabakus, A. T., ve Doğru, İ. A. (2018). An in-depth analysis of Android malware using hybrid techniques. *Digital Investigation*, *24*, 25-33.

4. Dinçer, C. A., Doğru, İ. A. Android Kötücül Yazılım Tespiti Yaklaşımları. *Uluslararası Bilgi Güvenliği Mühendisliği Dergisi*, *3*(2), 48-58.

5. Internet: 2018 Mobile Threat Landscape. URL: https://www.trendmicro.com/vinfo/au/security/research-and-analysis/threat-reports/roundup/2018-mobile-threat-landscape, Son Erişim Tarihi: 7.11.2019.

6. Internet: Android keystore system. URL: https://developer.android.com/training/articles/keystore#HardwareSecurityModule, Son Erişim Tarihi: 7.11.2019.

7. Internet: Riltok mobile Trojan: A banker with global reach. URL: https://securelist.com/mobile-banker-riltok/91374/, Son Erişim Tarihi: 3.12.2019.

8. Internet: An advertising dropper in Google Play. URL: https://securelist.com/dropper-in-google-play/92496/, Son Erişim Tarihi: 3.12.2019.

9. Internet: Android Architecture. URL: http://www.tutorialspoint.com/android/android_architecture.htm, Son Erişim Tarihi: 7.11.2019.

10. Rastogi, V., Chen, Y., and Jiang, X. (2013, May). Droidchameleon: evaluating android anti-malware against transformation attacks. *In Proceedings of the 8th ACM SIGSAC symposium on Information, computer and communications security* (pp. 329-334). ACM.

11. Kabakus, A. T., ve Doğru, İ. A., ve Çetin, A. (2015). Android kötücül yazılım tespit ve koruma sistemleri. *Erciyes Üniversitesi Fen Bilimleri Enstitüsü Fen Bilimleri Dergisi*, *31*(1), 9-16.

12. Chandramohan, M., and Tan, H. B. K. (2012). Detection of mobile malware in the wild. *Computer*, (9), 65-71.

13. Feizollah, A., Anuar, N. B., Salleh, R., and Wahab, A. W. A. (2015). A review on feature selection in mobile malware detection. *Digital investigation*, *13*, 22-37.

14. Seo, S. H., Gupta, A., Sallam, A. M., Bertino, E., ve Yim, K. (2014). Detecting mobile malware threats to homeland security through static analysis. *Journal of Network and Computer Applications*, *38*, 43-53.

15. Liu, X., and Liu, J. (2014, April). A two-layered permission-based android malware detection scheme. *In 2014 2nd IEEE International Conference on Mobile Cloud Computing, Services, and Engineering*, 142-148.

16. Yerima, S. Y., Sezer, S., and McWilliams, G. (2014). Analysis of Bayesian classification-based approaches for Android malware detection. *IET Information Security*, *8*(1), 25-36.

17. Shabtai, A., Fledel, Y., and Elovici, Y. (2010, December). Automated static code analysis for classifying android applications using machine learning. *In 2010 International Conference on Computational Intelligence and Security*, 329-333.

18. Kabakus, A. T., Dogru, I. A., and Cetin, A. (2015). APK Auditor: Permission-based Android malware detection system. *Digital Investigation*, *13*, 1-14.

19. Xiaoyan, Z., Juan, F., and Xiujuan, W. (2014). Android malware detection based on permissions. *In International Conference on Information and Communications Technologies (ICT 2014)*, 1-5.

20. Sanz, B., Santos, I., Laorden, C., Ugarte-Pedrero, X., Bringas, P. G., and Álvarez, G. (2013). Puma: Permission usage to detect malware in android. In *International Joint Conference CISIS'12-ICEUTE 12-SOCO 12 Special Sessions*, Berlin, 289-298.

21. Yerima, S. Y., Sezer, S., and Muttik, I. (2014, September). Android malware detection using parallel machine learning classifiers. In *2014 Eighth International Conference on Next Generation Mobile Apps, Services and Technologies*, 37-42.

22. Suarez-Tangil, G., Tapiador, J. E., Peris-Lopez, P., and Blasco, J. (2014). Dendroid: A text mining approach to analyzing and classifying code structures in android malware families. *Expert Systems with Applications*, *41*(4), 1104-1117.

23. Kayabaşı, G., ve Doğru, I. A. (2016). Mobil Uygulamaların Sınıflandırmasında Kullanılan Makine Öğrenmesi Algoritmalarının Güvenirlilik Tespiti. *9. Bilgi Güvenliği ve Kriptoloji Konferansı Bildiri Kitabı*, 191-195.

24. Arslan, R. S., Doğru, İ. A., ve Barışçı, N. (2017). Android Mobil Uygulamalar için İzin Karşılaştırma Tabanlı Kötücül Yazılım Tespiti. *Politeknik Dergisi*, *20*(1), 175-189.

25. Burguera, I., Zurutuza, U., and Nadjm-Tehrani, S. (2011). Crowdroid: behavior-based malware detection system for android. In *Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices*, 15-26.

26. Lu, Y., Zulie, P., Jingju, L., and Yi, S. (2013). Android malware detection technology based on improved Bayesian classification. In *2013 third international conference on instrumentation, measurement, computer, communication and control*, 1338-1341.

27. Zhou, Y., and Jiang, X. (2012). Dissecting android malware: Characterization and evolution. In *2012 IEEE symposium on security and privacy*. 95-109).

28. Alzaylaee, M. K., Yerima, S. Y., and Sezer, S. (2016). DynaLog: An automated dynamic analysis framework for characterizing android applications. In *2016*

International Conference On Cyber Security And Protection Of Digital Services (Cyber Security), 1-8.

29. Jaiswal, M., Malik, Y., and Jaafar, F. (2018). Android gaming malware detection using system call analysis. In *2018 6th International Symposium on Digital Forensic and Security (ISDFS)*, 1-5.

30. Aktaş, K., and Sen, S. (2018). Android malware detection based on runtime behaviour. In 2018 26th Signal Processing and Communications Applications Conference (SIU), 1-4).

31. Wang, X., Yang, Y., Zeng, Y., Tang, C., Shi, J., and Xu, K. (2015). A novel hybrid mobile malware detection system integrating anomaly detection with misuse detection. In *Proceedings of the 6th International Workshop on Mobile Cloud Computing and Services*. 15-22.

32. Yang, T., Qian, K., Li, L., Lo, D., and Tao, L. (2016, November). Static mining and dynamic taint for mobile security threats analysis. In *2016 IEEE International Conference on Smart Cloud (SmartCloud)*, 234-240.

33. Kabakus, A. T., ve Dogru, I. A. (2018). An in-depth analysis of Android malware using hybrid techniques. *Digital Investigation*, *24*, 25-33.

34. Shi, Y., You, W., Qian, K., Bhattacharya, P., and Qian, Y. (2016). A hybrid analysis for mobile security threat detection. In *2016 IEEE 7th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*, 1-7.

35. Arshad, S., Shah, M. A., Wahid, A., Mehmood, A., Song, H., ve Yu, H. (2018). Samadroid: a novel 3-level hybrid malware detection model for android operating system. *IEEE Access*, *6*, 4321-4339.

36. Alshahrani, H., Mansourt, H., Thorn, S., Alshehri, A., Alzahrani, A., ve Fu, H. (2018). DDefender: Android application threat detection using static and dynamic analysis. In *2018 IEEE International Conference on Consumer Electronics (ICCE)*, 1-6.

37. Arslan, R. S., Doğru, İ. A., ve Barışçı, N. (2019). Permission-Based Malware Detection System for Android Using Machine Learning Techniques. *International Journal of Software Engineering and Knowledge Engineering*, 29(01), 43-61.

38. Wei, F., Li, Y., Roy, S., Ou, X., and Zhou, W. (2017, July). Deep ground truth analysis of current android malware. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*. 252-276.

39. Witten, I. H., Frank, E., and Hall, M. A. (2005). Practical machine learning tools and techniques. *Morgan Kaufmann*, 578.

40. Kotsiantis, S. B., Zaharakis, I., and Pintelas, P. (2007). Supervised machine learning: A review of classification techniques. *Emerging artificial intelligence applications in computer engineering*, *160*, 3-24.

41. Internet: sklearn.ensemble.RandomForestClassifier. URL: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html, Son Erişim Tarihi: 7.11.2019.

42. Cortes, C., and Vapnik, V. (1995). Support-vector networks. *Machine learning*, *20*(3), 273-297.

43. Fawcett, T. (2006). An introduction to ROC analysis. *Pattern recognition letters*, *27*(8), 861-874.

44. Internet: Android Developers, Requesting Permissions, URL: https://developer.android.com/guide/topics/permissions/requesting.html. Son Erişim Tarihi: 7.11.2019.

45. Internet: platform_frameworks_base, AndroidManifest.xml, URL: https://github.com/aosp-mirror/platform_frameworks_base/blob/master/core/res/AndroidManifest.xml#L1. Son Erişim Tarihi: 7.11.2019.

46. Internet: Manifest.permission. URL: https://developer.android.com/reference/android/Manifest.permission, Son Erişim Tarihi: 7.12.2019.

47. Joseph, D. A., Paxson, V., and Kim, S. (2006). tcpdump Tutorial. *University of California, EE122 Fall*.

48. Goyal, P., and Goyal, A. (2017). Comparative study of two most popular packet sniffing tools-Tcpdump and Wireshark. In *2017 9th International Conference on Computational Intelligence and Communication Networks (CICN)*. 77-81.

49. Internet: tshark - Dump and analyze network traffic .URL: https://www.wireshark.org/docs/man-pages/tshark.html, Son Erişim Tarihi: 18.02.2020.

50. Internet: Domain and IP address reports. URL: https://support.virustotal.com/hc/en-us/articles/115002719069-Reports#h_c095fe17-40ce-4a5d-a561-6df598cf34d6, Son Erişim Tarihi: 7.12.2019.

51. Internet: The Python SDK for AlienVault OTX. URL: https://github.com/AlienVault-OTX/OTX-Python-SDK

ÖZGEÇMİŞ

Kişisel Bilgiler

Soyadı, adı : ASLANALP DİNÇER, Ceren

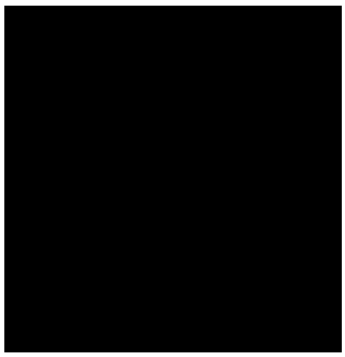
Uyruğu : T.C.

Doğum tarihi ve yeri :

Medeni hali :

Telefon :

E-posta :



Eğitim

Derece	Eğitim Birimi	Mezuniyet Tarihi
Yüksek lisans	Gazi Üniversitesi / Adli Bilişim	2020
Lisans	Atılım Üniversitesi / Bilgisayar Mühendisliği	2009
Lise	M.E.V. Özel Ankara Fen Lisesi	2003

İş Deneyimi

Yıl	Yer	Görev
2011-Halen	Çankaya Belediyesi	Mühendis

Yabancı Dil

İngilizce.

Yayınlar

1. Dinçer, C. A. ve Doğru, İ. A. Android Kötücül Yazılım Tespiti Yaklaşımları. *Uluslararası Bilgi Güvenliği Mühendisliği Dergisi*, *3*(2), 48-58.

GAZİLİ OLMAK AYRICALIKTIR…