
GAZİ ÜNİVERSİTESİ
Gazi
1926

ÖĞRENMEYE DAYALI İSTEMCİ ve SUNUCU TABANLI ANDROID KÖTÜCÜL YAZILIM TESPİT SİSTEMİ

Abdullah DAĞLIOĞLU

YÜKSEK LİSANS TEZİ
BİLGİSAYAR BİLİMLERİ ANA BİLİM DALI

GAZİ ÜNİVERSİTESİ

BİLİŞİM ENSTİTÜSÜ

HAZİRAN 2020

Abdullah DAĞLIOĞLU tarafından hazırlanan "ÖĞRENMEYE DAYALI İSTEMCİ ve SUNUCU TABANLI ANDROID KÖTÜCÜL YAZILIM TESPİT SİSTEMİ" adlı tez çalışması aşağıdaki jüri tarafından OY BİRLİĞİ ile Gazi Üniversitesi Bilgisayar Bilimleri Anabilim 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:

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.

Abdullah DAĞLIOĞLU

ÖĞRENMEYE DAYALI İSTEMCİ VE SUNUCU TABANLI ANDROID KÖTÜCÜL YAZILIM TESPİT SİSTEMİ

(Yüksek Lisans Tezi)

Abdullah DAĞLIOĞLU

GAZİ ÜNİVERSİTESİ

BİLİŞİM ENSTİTÜSÜ

Haziran 2020

ÖZET

Günümüzde bilgisayarların erişilemez olduğu durumlarda mobil cihazlar bilgiye erişmek için önemli bir araç haline gelmiştir. Akıllı mobil cihazlar birçok farklı alanda farklı amaçlarla kullanılmaktadır. Android, dünya genelinde en çok tercih edilen mobil işletim sistemidir. Bu popülerlik akıllı mobil cihazları kötücül yazılımların hedefi haline getirmiştir. Android, Google tarafından geliştirilen Linux tabanlı ve açık kaynak kodlu bir işletim sistemidir. Android izin tabanlı güvenlik mekanizmasına sahiptir. Yapılan birçok araştırma, kullanıcıların izinler hakkında yeterli bilgiye sahip olmadığını ve bu nedenle mobil cihazlarını tehlikeye attıklarını göstermiştir. Bu problemi gidermek için araştırmacılar yoğun çaba harcamaktadır. Bu tezde, bahsi geçen problemi gidermek için kullanıcılara kullandıkları uygulamaların güvenlik durumunu analiz edip bildiren bir sistem geliştirilmiştir. Geliştirilen sistem, öğrenmeye dayalı kötücül yazılım tespit etmeyi sağlayan istemci ve sunucu tabanlı kullanıcı dostu bir platformdur. Bu çalışma, iyicil veya kötücül uygulamaların kullandıkları izinleri makine öğrenme algoritması ile analiz edip analiz sonucunu kullanıcıya bildirmektedir. Geliştirilen sistem ile kullanıcıların telefonlarındaki uygulamaların türü ve kullandığı izinler hakkında bilgilendirilmesi amaçlanmaktadır. Test sonuçları incelendiğinde en başarılı sonuçların %91,76 doğruluk, %91,31 hassasiyet ve %93,73 özgüllük oranlarıyla Gradyan Arttırıcı Sınıflandırma Algoritması ile elde edildiği gözlenmiştir.

BilimKodu	:	92403
AnahtarKelimeler	:	Android işletim sistemi, kötü niyetli yazılım tespiti, uygulama güvenliği, mobil güvenlik, makine öğrenmesi
SayfaAdedi	:	55
Danışman	:	Doç. Dr. İbrahim Alper DOĞRU

LEARNING ORIENTED CLIENT AND SERVER-BASED ANDROID MALWARE DETECTION SYSTEM

(M. Sc. Thesis)

Abdullah DAĞLIOĞLU

GAZİ UNIVERSITY

INFORMATICS INSTITUTE

JUNE 2020

ABSTRACT

Today, when computers are inaccessible, mobile devices have become an important tool to access information. Smart mobile devices are used in many different areas for different purposes. Android is the most preferred mobile operating system across the world. This popularity has turned smart mobile devices into the target of malware. Android is a Linux-based and open source operating system developed by Google. Android has a permission-based security mechanism. Many studies have shown that users do not have enough information about permissions and therefore endanger their mobile devices. Researchers are endeavoring to solve this problem. In order to solve the problem mentioned in this study, a system has been developed to analyze and report the security status of the applications they use. The developed system, Learning Oriented Client and Server-Based Android Malware Detection System, is a client and server-based user-friendly platform that enables the detection of Android malware using the machine learning algorithms. This study analyzes the permits used by good or malicious applications with the machine learning algorithm and reports the result of the analysis to the user. It is intended to inform users about the type of applications and the permissions they use by means of the developed system. When the test results were examined, it was observed that the most successful results were obtained with the Gradient Boosting Classifier Algorithm with 91.76% accuracy, 91.31% sensitivity and 93.73% specificity rates.

ScienceCode	:	92403
KeyWords	:	Android operating system, malware detection systems, application security, mobile security, machine learning
PageNumber	:	55
Supervisor	:	Assoc. Prof. Dr. İbrahim Alper DOĞRU

TEŞEKKÜR

Tez çalışmalarım sırasında kıymetli vaktini ve mesleki tecrübesini beni yönlendirmek için harcayan değerli tez danışmanım Doç. Dr. İbrahim Alper DOĞRU'ya, eğitim hayatım boyunca her zaman bana maddi manevi destek olan babam Hasan DAĞLIOĞLU, annem Sevim DAĞLIOĞLU ve kardeşlerim Yunus Emre DAĞLIOĞLU ile Arzu DAĞLIOĞLU'na, yüksek lisans eğitimim boyunca merakla çalışmalarımı takip edip daima yanımda olduklarını hissettiren değerli eşim Duygu DAĞLIOĞLU ve sevgili oğlum Hasan Toprak DAĞLIOĞLU'na teşekkürü kendime borç bilirim.

İÇİNDEKİLER

Sayfa

ÖZET iv

ABSTRACT v

TEŞEKKÜR vi

İÇİNDEKİLER vii

ÇİZELGELERİN LİSTESİ ixx

ŞEKİLLERİN LİSTESİ x

KISALTMALAR xii

1. GİRİŞ 1

2. ANDROID İŞLETİM SİSTEMİ 5

2.1. Linux Çekirdek (Linux Kernel) 5

2.2. Kütüphaneler (Libraries) 5

2.3. Android Çalışma Zamanı (Android Runtime) 6

2.3.1. Çekirdek kütüphaneleri (Core libraries) 6

2.3.2. Dalvik sanal makinesi (Dalvik virtual machine - DVM) 6

2.4. Uygulama Çatısı (Application Framework) 6

2.4.1. Aktivite yöneticisi (Activity manager) 6

2.4.2. Görünümler (Views) 6

2.4.3. Uyarı yöneticisi (Notification manager) 7

2.4.4. İçerik sağlayıcılar (Content providers) 7

2.4.5. Kaynak yöneticisi (Resource manager) 7

2.5. Uygulamalar (Applications) 7

3. ANDROID KÖTÜ NİYETLİ YAZILIMLARIN TESPİTİ VE KORUMA YÖNTEMLERİ 9

3.1. Statik Analiz Yöntemi 9

Sayfa

3.1.1. Literatürdeki statik analiz yöntemleri 10

3.2. Dinamik Analiz Yöntemi 20

3.3. Hibrit Analiz Yöntemi 22

3.4. Literatürdeki Çalışmaların Yeteneklerinin Karşılaştırılması 25

4. KULLANILAN YÖNTEM VE MATERYAL 27

4.1. Android İstemci Uygulaması 28

4.2. Vekil Sunucu 300

4.3. Merkez Sunucu 31

4.3.1. Veri ön işleme 32

4.3.2. Model eğitimi ve seçimi 32

4.3.3. Model değerlendirmesi 37

4.4. Veritabanı Sunucusu 44

5. TARTIŞMA 45

6. SONUÇ VE ÖNERİLER 47

KAYNAKLAR 49

ÖZGEÇMİŞ 555

ÇİZELGELERİN LİSTESİ

Çizelge **Sayfa**

Çizelge 3.1. Literatürdeki çalışmaların yetenek karşılaştırılması 26

Çizelge 4.1. VirusTotal'e göre oluşan veri seti özeti 27

Çizelge 4.2. Doğruluk skorları 37

Çizelge 4.3. Karışıklık matrisi 38

Çizelge 4.4. K En Yakın Komşu Algoritması ile eğitilen modelin karışıklık matrisi çıktısı 38

Çizelge 4.5. Destek Vektör Makinesi Algoritması ile eğitilen modelin karışıklık matrisi çıktısı 38

Çizelge 4.6. Rastgele Orman Algoritması ile eğitilen modelin karışıklık matrisi çıktısı 39

Çizelge 4.7. Gradyan Arttırıcı Sınıflandırma Algoritması ile eğitilen modelin karışıklık matrisi çıktısı 39

Çizelge 4.8. K En Yakın Komşu Algoritması ile eğitilen modelin sınıflandırma raporu çıktısı 40

Çizelge 4.9. Destek Vektör Makinesi Algoritması ile eğitilen modelin sınıflandırma raporu çıktısı 40

Çizelge 4.10. Rastgele Orman Algoritması ile eğitilen modelin sınıflandırma raporu çıktısı 40

Çizelge 4.11. Gradyan Arttırıcı Sınıflandırma Algoritması ile eğitilen modelin sınıflandırma raporu çıktısı 40

Çizelge 4.12. Kullanılan makine öğrenme algoritmasıyla alıcı işlem karakteristiği eğrisi alanlarının karşılaştırılması 43

Çizelge 5.1. Kötücül yazılımların en çok kullandığı 10 izin ve bu izinlerin güvenlik durumları 45

Çizelge 5.2. İyicil yazılımların en çok kullandığı 10 izin ve bu izinlerin güvenlik durumları 45

Çizelge 5.3. Elde edilen sonuçların karşılaştırılması 46

ŞEKİLLERİN LİSTESİ

Şekil **Sayfa**

Şekil 2.1. Android işletim sistemi mimarisi..... 5

Şekil 3.1. Şüpheli API örnekleri 12

Şekil 3.2. Ayrıcalık arttırıcı komutlar 13

Şekil 3.3. Kötücül yazılımların farklı sektörlerde kullandığı izinler 13

Şekil 3.4. Resmi ve resmi olmayan marketlerdeki uygulamaların kullandığı izinler..... 14

Şekil 3.5. Resmi ve resmi olmayan marketlerdeki uygulamaların kullandığı komutlar. 14

Şekil 3.6. Zararlı ve zararsız yazılımlardaki en sık kullanılan izinlerin oluşum yüzdesi 16

Şekil 3.7. Tespit yöntemi 24

Şekil 3.8. Bilinmeyen uygulamaların farklı gruplarının algılama doğruluğu oranları .. 25

Şekil 4.1. Geliştirilen sistemin mimarisi 28

Şekil 4.2. İstemci uygulamasının ana ekran görüntüsü 29

Şekil 4.3. Analizi yapılan uygulamanın analiz sonuçlarını gösteren ekran görüntüsü .. 30

Şekil 4.4. Makine öğrenimi algoritmalarının uygulanma adımları..... 31

Şekil 4.5. K En Yakın Komşu Algoritması kod parçası 33

Şekil 4.6. Destek Vektör Makinesi Algoritması kod parçası..... 34

Şekil 4.7. Rastgele Orman Algoritması kod parçası 35

Şekil 4.8. Gradyan Arttırıcı Sınıflandırma Algoritması kod parçası 36

Şekil 4.9. K En Yakın Komşu Algoritması ile eğitilen modelin alıcı işlem karakteristikleri eğrisi 41

Şekil 4.10. Destek Vektör Makinesi Algoritması ile eğitilen modelin alıcı işlem karakteristikleri eğrisi 42

Şekil 4.11. Rastgele Orman Algoritması ile eğitilen modelin alıcı işlem karakteristikleri eğrisi 42

Şekil **Sayfa**

Şekil 4.12. Gradyan Arttırıcı Sınıflandırma Algoritması ile eğitilen modelin alıcı işlem karakteristikleri eğrisi..... 43

Şekil 4.13. Geliştirilen sistemin veritabanı E-R diyagramı 44

KISALTMALAR

Bu çalışmada kullanılmış kısaltmalar, açıklamaları ile birlikte aşağıda sunulmuştur.

Kısaltmalar	Açıklamalar
API	Uygulama programlama arayüzleri (Application programming interface)
APK	Android paketi (Android package)
CPU	Merkezi işlem birimi (Central process unit)
DEX	Dalvik yürütülebilir (Dalvik executable)
DVM	Dalvik sanal makinesi (Dalvik virtual machine)
DECAFPLATFORM	Dinamik yürütülebilir kod analizi çerçeve platformu (Dynamic executable code analysis framework platform)
ER	Varlık ilişkisi (Entity relation)
FN	Yanlış olumsuz (False negative)
FP	Yanlış olumlu (False positive)
FPR	Yanlış olumlu oranı (False positive rate)
HTTP	Hiper metin transfer protokolü (Hypertext transfer protocol)
JVM	Java sanal makinesi (Java virtual machine)
k-NN	K en yakın komşu (K nearest neighbor)
MCDF	İşbirlikçi karar füzyon yapısı (Multifeature collaborative decision fusion)
NB	Naif bayes (Naive bayes)
RAM	Rastgele erişimli hafıza (Random access memory)
ROC	Alıcı işlem karakteristik eğrisi (Receiver operation characteristic)
SMS	Kısa mesaj servisi (Short message service)
SSL	Güvenli giriş katmanı (Secure sockets layer)
SVM	Destek vektör makinesi (Support vector machine)
TN	Doğru olumsuz (True positive)
TPR	Doğru olumlu oranı (True positive rate)
TP	Doğru olumlu (True positive)

1. GİRİŞ

Günümüzde akıllı mobil cihazların gelişimiyle bilgiye her yerden erişmek oldukça kolay hale gelmiştir. Bu durum insanlar arasında akıllı telefon kullanım oranını oldukça arttırmıştır. Mobil cihazların artmasıyla kullanıcılar bu cihazları kişisel iletişim, veri depolama, multimedya, gerekli bilgiye ulaşma ve eğlence gibi çeşitli amaçlarla kullanmaktadırlar. Statista firmasının 2018'in ikinci çeyreğinde yapmış olduğu araştırmaya göre 2009 yılı ile 2018 yılları arasında dünya çapında satılan akıllı telefonların %88'i Android işletim sistemine sahip cihazlardır [1]. International Data Corporation (IDC) firmasının Eylül 2018 raporuna göre 2019 yılında akıllı telefon kullanım oranının 2018 yılına göre %3,7 artacağı belirtilmektedir [2]. Aynı raporda bu artışın sebebi olarak tüketicilerin büyük ekranlı cihaz kullanma isteğinin artıyor olması gösterilmektedir [2]. Statista, istatistik firmasının 2018'de yapmış olduğu araştırmaya göre Google Play Store'daki uygulama sayısı 2,6 milyonun üzerindedir [3]. Mobil cihazların kullanım oranının artması ve Android işletim sisteminin en çok kullanılan işletim sistemi olması bu platformu kötü niyetli yazılım geliştiricilerinin hedefi haline getirmiştir.

Android, Google tarafından geliştirilen açık kaynak kodlu ve Linux tabanlı bir mobil işletim sistemidir. F-Secure güvenlik firmasının F-Secure Siber Güvenlik Durumu 2017 raporuna göre mobil kötücül saldırıların %99'undan fazlası Android cihazları hedeflemektedir [4]. Ayrıca aynı raporda Android'in açık kaynak bir işletim sistemi olmasından dolayı özellikle Android cihazları hedefleyen 19 milyondan fazla kötücül yazılımın geliştirildiği belirtilmektedir [4]. G Data firmasının Eylül 2018 raporuna göre 2018'in ilk çeyreğinde yapılan çalışmada 846916 adet yeni kötücül yazılım tespit edilmiştir [5]. Ayrıca bu araştırmaya göre 2017'nin ilk çeyreğine oranla 2018'in ilk çeyreğinde kötücül yazılımın %12 oranında arttığı gözlenmiştir [5]. Ayrıca bu rapora göre her gün ortalama 9411 adet kötücül yazılım tespit edilmektedir [5]. Android mobil kötücül yazılımlar, kullanıcılar adına yetkisiz işlem yapmak (malware) ve kullanıcıların kişisel verilerini bilgi amaçlı toplamak (grayware) üzere iki temel amaca sahiptirler. Android mobil kötücül yazılımların artma sebepleri olarak Android platformunun açık kaynak kodlu olması, resmi uygulama dağıtım kanalı olan Google Play Store'da yeteri kadar güvenlik taraması olmaması ve uygulamaların resmi uygulama dağıtım kanalı dışında da dağıtılabiliyor olması gösterilebilir. İzinler, uygulamaların cihaz veya kullanıcının çeşitli kaynaklarını kullanmak amacıyla ihtiyaç

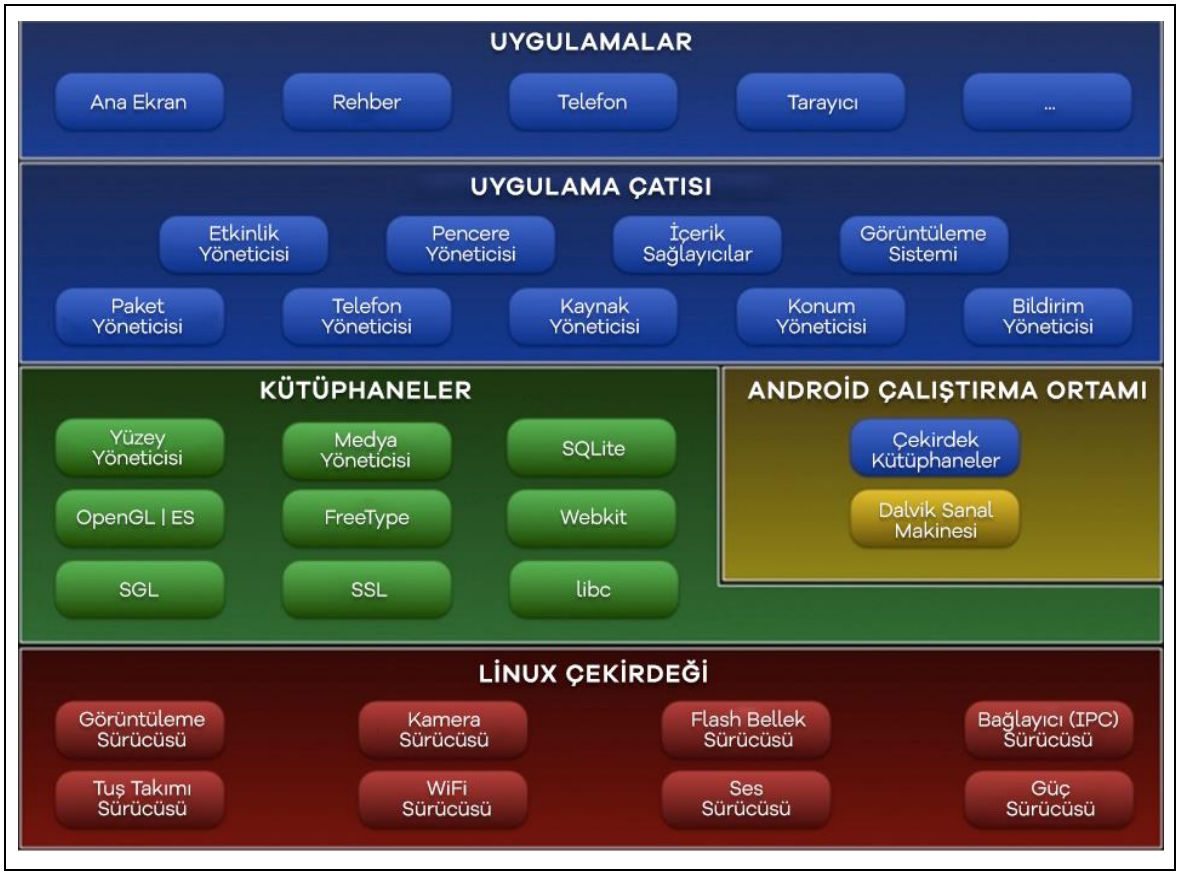
duyduğu onaylardır. Yapılan birçok araştırma kullanıcıların bu izinler hakkında yeterli bir bilgiye sahip olmadığını göstermektedir. Felt ve arkadaşları yaptıkları çalışmada kullanıcıların %42'sinin izinler hakkında herhangi bir bilgiye sahip olmadıklarını gözlemlemişlerdir [6]. Aynı çalışmada izinleri dikkate alan kullanıcı yüzdesinin ise sadece %17 olduğu saptanmıştır [6]. Android kötücül yazılımların artmasını engellemek ve Android platformunu daha güvenilir bir hale getirmek için Google tarafından yoğun çaba gösterilmektedir. Google ilk olarak izin tabanlı güvenlik mekanizmasını iyileştirerek Android 6.0 Marshmallow sürümünü geliştirmiştir. Android 6.0 Marshmallow sürümü ile tehlikeli olarak tariflenen izinlerin kullanılmadan önce kullanıcının onayının alınması amaçlanmıştır. Google ikinci çalışma olarakta Google I/O 2017'de uygulamaların çalışma anında analizinin yapılabilmesi için Play Store uygulamasına gömülü olarak çalışacak Google Play Protect'i tanıtmıştır. Yapılan araştırmalara göre Google tarafından alınan birtakım önlemlere rağmen hala her 10 saniyede yeni bir kötücül yazılım ile karşılaşılmakta ve bu hızla 2018'in sonunda 3,4 milyon yeni Android kötücül yazılımın olacağı tahmin edilmektedir [6]. Bu doğrultuda hala kullanıcıları kullandıkları uygulamalar hakkında bilgilendiren ve kötücül yazılımları tespit eden sistemler geliştirmeye ihtiyaç duyulmaktadır.

Bu tezde, kullanıcıları kullandıkları uygulamaların güvenilirliği hakkında bilinçlendirmek için makine öğrenme algoritmalarından yararlanılarak istemci ve sunucu mimarisinde kötücül yazılım tespiti sağlayan bir sistem geliştirilmiştir. Geliştirilen istemci uygulamasında cihazda kullanılan üçüncü parti uygulamalar listelenmekle birlikte uygulamaların kullandığı izinler ve bu uygulamaların iyicil mi kötücül mü olduğu hakkında son kullanıcıya bilgiler verilmektedir. Bu sistem ile uygulamaların AndroidManifest.xml dosyalarında tanımlanan izinler kullanılarak Gradyan Arttırıcı Sınıflandırma (Gradient Boosting Classifier) Algoritması, Rastgele Orman (Random Forest) Algoritması, Destek Vektör Makinesi (Support Vektor Machine) Algoritması ve K En Yakın Komşu (K-Nearest NeighBor) Algoritması ile uygulamaların karekteristiği çıkarılıp son kullanıcıya uygulamanın iyicil mi kötücül mü olduğu belirtilmektedir. Geliştirilen sistem dört bileşenden oluşmaktadır: (1) Android cihazlarda kullanılacak Android istemci uygulaması, (2) istemci uygulamasından aldığı istekleri merkez ve veritabanı sunucuları ile iletişim kurarak gelen isteklere cevap verecek vekil sunucu, (3) uygulamaların analizlerinin yapıldığı merkez sunucu, (4) analizi yapılan uygulamaların bilgilerinin tutulduğu veritabanı sunucusu.

Tez çalışmasının başlıkları şu şekilde planlanmıştır: 2. bölümde Android işletim sisteminin yazılım mimarisi incelenmiştir. 3. bölümde literatürdeki Android kötücül yazılımların tespiti için yapılan çalışmalar ve kullanılan yöntemler sunulmuştur. 4. bölümde kullanılan yöntem ve materyaller sunulmuştur. 5. bölümde elde edilen bulgular tartışılmıştır ve son olarak 6. bölümde sonuç ve öneriler sunulmuştur.

2. ANDROID İŞLETİM SİSTEMİ

Android'in yazılım mimarisi Şekil 2.1'de de gösterildiği gibi Linux çekirdek, kütüphaneler, Android çalışma zamanı, uygulama çatısı ve uygulamalar olmak üzere 5 katmandan meydana gelmektedir.



Şekil 2.1. Android işletim sistemi mimarisi [7]

2.1. Linux Çekirdek (Linux Kernel)

Donanıma ait tüm bilgiler ile android uygulamaların çalışmasını sağlayan ses, klavye, WI-FI, güç denetimi, işlem ve hafıza denetimi gibi sürücülerin bulunduğu en alt katmandır [7].

2.2. Kütüphaneler (Libraries)

Çekirdeğin üstünde bulunan ve genellikle C++ ve C dilleri ile yazılmış SQLite, SSL, FreeType gibi kütüphanelerin yanı sıra sistem kütüphanelerini içeren katmandır [7].

2.3. Android Çalışma Zamanı (Android Runtime)

Bu katman Android mimarisinin üçüncü bölümüdür ve alttan üçüncü bölümde bulunmaktadır. Bu bölüm Dalvik Sanal Makinesi denilen anahtar bir bileşen sağlamaktadır. Bunun yanı sıra bu katmanda Linux'un çekirdek kütüphaneleri bulunur [7].

2.3.1. Çekirdek kütüphaneleri (Core libraries)

Bunlar Java SE ve Java ME'den farklı kütüphanelerdir. Java'ya gerekli olan çekirdek API'ler, veri yapılar, hizmetler, dosya erişimler, ağ erişimler ve grafik bileşenlerininin yer aldığı kütüphanelerdir [7].

2.3.2. Dalvik sanal makinesi (Dalvik virtual machine - DVM)

Android İşletim Sisteminin en önemli bileşenidir ve mobil cihazlar için optimize edilmiş android sanal makinelerdir. Modern JVM olarakta bilinir yüksek performans ve kusursuz memory yönetimi sağlar. Bir cihaz üzerinde birden fazla sanal makine oluşturulabilir. Dalvik Sanal Makinesi cihazdaki düşün seviyeli işlemleri düzene sokmak ve performansı arttırmak için Linux işletim sisteminin çekirdeği kullanılmaktadır. Gömülü (embedded) sistemler için tasarlanan bu sistemde, Java derleyicisi ile .class dosyalarına dönüştürülen .java dosyaları Dex derleyicisi ile .dex dosyalarına dönüştürür. Dönüştürülen bu .dex dosyalar Dalvik Sanal Makinesi'nde çalıştırılır [7].

2.4. Uygulama Çatısı (Application Framework)

Uygulama çatısı katmanı uygulamalara java classları şeklinde yüksek seviyeli servisler sağlamaktadır [7].

2.4.1. Aktivite yöneticisi (Activity manager)

Projenin sahip olduğu aktivitelerin bütün yönlerini ve yaşam döngüsünü kontrol eder, sistemde çalışan tüm aktiviteler ile etkileşim halindedir [7].

2.4.2. Görünümler (Views)

Aktivitelerin kullanması gereken arayüzlerin tasarlanması için kullanılmaktadır. Listeler, tarayıcılar ve düğmeleri barındıran zengin ve genişletilebilir bir görüntüleme sistemidir [7].

2.4.3. Uyarı yöneticisi (Notification manager)

Uygulamada kullanılan bildirimler ve uyarıların doğru ve zamanında çalışmasını kontrol eder [7].

2.4.4. İçerik sağlayıcılar (Content providers)

Rehber uygulamaları gibi başka uygulamaların verilerine erişmini sağlar veya kendi verilerini başka uygulamara aktarmayı [7].

2.4.5. Kaynak yöneticisi (Resource manager)

Kod kullanılmayan gömülü kaynaklara erişimi sağlamaktadır. Bu kaynaklar grafik ayarı ve kullanıcı arayüzü düzenleri olabilir [7].

2.5. Uygulamalar (Applications)

Android uygulama çatısının en üst kısmında yer alan katmandır. Sınıflar ve servisleri kullanan rehber, ayarlar, oyunlar vb. gibi ana uygulamaların tümünün bulunduğu katmandır. Ayrıca bu katman yerel kütüphaneleri ve Linux çekirdeğini kullanır [7].

3. ANDROID KÖTÜ NİYETLİ YAZILIMLARIN TESPİTİ VE KORUMA YÖNTEMLERİ

Android işletim sisteminin kullanımının artmasıyla Android mobil kötücül yazılımlar da artmıştır. Bu artış, mobil kötücül yazılım çeşitliliğini de arttırmıştır. Mobil kötücül yazılım çeşitliliği araştırmacıları ve kullanıcıları kötücül yazılım tespit ve korunma yöntemleri araştırmasına yöneltmiştir. Bu bölümde mobil kötücül yazılım tespitinde kullanılan teknikler ve koruma yöntemleri incelenmiştir.

3.1. Statik Analiz Yöntemi

Statik analiz yazılımın çalışmasına gerek kalmadan uygulamanın apk'sı üzerinden analiz yapılan test tekniklerinden biridir. Statik analiz araçları kötücül davranış sergileyen komutları veya uygulama içindeki statik API'leri bulmak için kullanılmaktadır. Statik analiz yöntemlerinden bazıları şu şekilde sıralanabilir [8]:

- Kaynak kod soruşturma (Source code inspection): Kaynak dosyanın analiz edilmesi işlemidir.

- İmza Eşleştirme (Signature matching): Kötücül yazılım tespitinde kullanılan en yaygın yöntemlerden biridir. Dinamik analize göre daha az maliyetlidir.

- Konfigürasyon Dosyası Soruşturma (Inspecting configuration file): Android uygulamalar AndroidManifest.xml dosyası barındırmaktadır. Uygulama hakkındaki tüm temel bilgiler bu dosyada tanımlanmaktadır. Bu yöntem ile manifest dosyası incelenerek tespit yapılmaktadır.

- Ortak dizi çıkartma (Common string extraction): Bu yöntem ile uygulama içinde kullanılan internet adresleri, sabit değişkenler, kullanılan kütüphaneler, kullanıcı kimlik bilgileri gibi önemli dizi seti oluşturularak kötücül yazılım tespitinde kullanılır.

3.1.1. Literatürdeki statik analiz yöntemleri

Android işletim sisteminde kötü niyetli yazılımları tespit etmek için literatürde yapılan çalışmalar aşağıdaki başlıklarda anlatılmıştır.

Drebin

Drebin [9], statik analiz yöntemi ve makine öğrenmesi algoritmalarını birlikte kullanarak kötücül yazılım tespit etmeyi amaçlayan bir araçtır. Uygulamanın kaynak kodu ve AndroidManifest.xml dosyasını kullanarak uygulama tarafından kullanılan izinler, internet adresleri ve API çağrıları birleşik uzay vektörüne gömülerek kötücül yazılım tespiti için kullanılmaktadır.

Android ApkTool

Android ApkTool [10], tersine mühendislik yöntemleri ile Android apk dosyalarını analiz ederek tekrar derlenmesini sağlamaktadır.

ApkAnalyser

ApkAnalyser [11], statik analizi kullanan kötücül yazılım tespit araçlarından biridir. Apk statik, dinamik ve hibrid kod analizi uygulanarak, Joe Sandbox Mobile ile analiz edilir.

Androguard

Androguard [12], tersine mühendislik yöntemleri ve statik analiz kullanılarak kötücül yazılım tespit etmeyi amaçlayan bir araçtır. Linux/Windows/OSX platformlarını destekleyen python tabanlı araçtır.

Felt ve arkadaşlarının çalışması

Felt ve arkadaşlarının çalışmasında [6], çok fazla izin kullanan Android uygulamaları belirlemeye yarayan Stowaway adında bir sistem önerilmektedir. Stowaway, mobil uygulamaların kullandığı Uygulama Programlama Arayüzleri çağrı kümesini bulup,

çağrıları izinler ile eşleştirmesini sağlamaktadır. Stowaway'i oluşturan iki kısım vardır: Bunlardan ilki uygulamaların kullandıkları Uygulama Programlama Arayüzleri'ni tespit etmektedir. İkincisi ise, izinleri eşleştirip Uygulama Programlama Arayüzleri çağrılarının kullandığı izinlerin belirlenmesini sağlamaktır. Yapılan çalışmada geliştiriciler tarafından yapılan en yaygın hatalar da şu şekilde belirtilmiştir:

1. İzin isimlerinin yanlış kullanılması,
2. ilgili metotların kullanımı esnasındaki koruma seviyesi ihlali,
3. vekil kullanımı,
4. artık mevcut olmayan izinleri kullanmak,
5. sistem veya imza izinlerini kullanmak,
6. test için oluşturulan değişiklikleri unutmak,
7. kullanılmayan izinlerin uygulamaya internetten kopyalanıp yapıştırılması.

Shaowu Liu ve arkadaşlarının çalışması

Shaowu Liu ve arkadaşlarının [13] 2013'te yaptığı çalışmada, kötücül ve iyicil olmak üzere veri kümesi kullanılmıştır. Kötücül yazılım için, Zhou ve Jiang tarafından 2012 yılında yayınlanan ve üçüncü parti uygulama dağıtım marketlerinden elde edilen karşılaştırmalı kötücül uygulama veri kümesi kullanılmıştır. İyicil uygulamalar için ise üçüncü parti uygulama dağıtım marketleri olan SlideMe ve Pandaapp'den elde ettikleri kendi veri kümelerini kullanmışlardır. Bu veri kümesi oluşturulurken, uygulamaların indirilme sayıları ve kullanıcıların uygulamalara verdiği puanlar baz alınarak derecesi en yüksek olanlar veri kümesine dahil edilmiştir. Bu belirlemede 43 tane antivirüs programı kullanılarak hepsinden olumlu geçenler iyi niyetli uygulama veri kümesine dahil edilmiştir. Son dönemdeki çalışmaların aksine yalnızca AndroidManifest.xml dosyasında belirtilen gerekli izinlere (required permissions) odaklanılmamış, uygulama indirildikten sonra yer alan ve akıllı telefonun işletim sistemi tarafından kullanılan izinler (used permissions)'de ele alınmıştır. İyi ve kötücül veri kümelerini başlangıçta analiz etmek için hiyerarşik Biclustering methodu kullanılmıştır. Kötücül uygulamaları iyicil olanlardan ayırmak için kullanılabilecek ilginç izin kümelerini tanımlamak için Contrast Permission Pattern Mining algoritması önerilmiştir. Önerilen Contrast Permission Pattern Mining algoritması kötücül yazılım tespitinde hem gerekli hem de kullanılan izinlerin birlikte kullanılması gerektiğini kanıtlamaktadır.

Seung-Hyun Seo ve arkadaşlarının çalışması

Seung-Hyun Seo ve arkadaşlarının [14] 2013'te yaptığı çalışmada, Homeland Security'e yönelik mobil saldırı senaryoları gösterilip Android uygulamalardaki potansiyel güvenlik açıklarını analiz eden statik analiz aracı olan DroidAnalyzer incelenmiştir. DroidAnalyzer root yetkisinin mevcudiyetini, ona ilişkin kötü yazılımın tipik anahtar kelimelerini ve benzersiz özelliklerini kullanarak belirleme fonksiyonuna sahiptir. Mobil güvenlik için mevcut olan önlem ve kuralları geliştirerek Homeland Security'e karşı olan mobil malware tehditleri azaltacak yeni bir metod önerilmiştir. Repackacing, Malvertizing, Browser Attacks, Update Attack, Drive-by Download gibi yöntemlerle akıllı telefonların işletim sistemleri ele geçirilerek edinilen bilgilerle ulaşılması planlanan amaçlar incelenmiştir. DroidAnalyzer kullanılarak elde edilen 1260 kötücül yazılım örneğinde kullanılan şüpheli API'ler Şekil 3.1'de gösterilmiştir.

Şüpheli API örnekleri

No.	API	Tanım
1	getLine1Number()	Telefon numarası sızıntısı
2	getSimSerialNumber()	USIM numarası sızıntısı
3	getSubscriberId()	IMSI sızıntısı
4	getDeviceId()	IMEI sızıntısı
5	getNETWORKCountryIso()	Ülke kodu sızıntısı
6	getLastKnownLocation() getLatitude().getLogitude()	Konum sızıntısı
7	Socket(),openConnection() getOutputStream() getInputStream() URL.openStream()	İnternet bağlantısı
8	sendTextMessage() sendDataMessage()	SMS gönderme MMS gönderme
9	getPackageManager() installPackage()	Uygulama indirme
10	getContentResolver().delete()	SMS ya da dosya silme
11	getAssets()	Varlıklar klasöründen bir metin dosyası okuma

Şekil 3.1. Şüpheli API örnekleri [14]

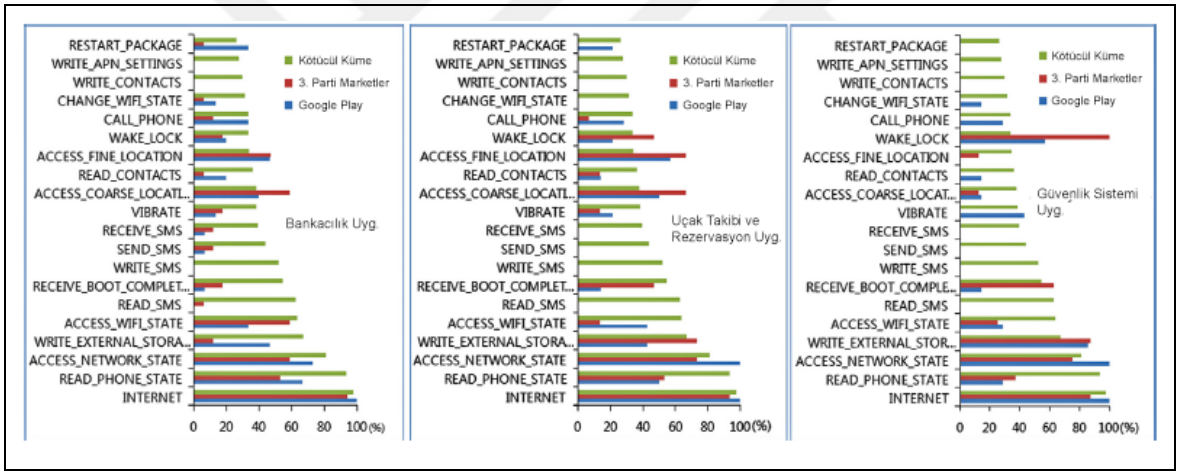
Bunlardan 1204 tane örnek Şekil 3.2'de gösterilen ayrıcalık artırıcı (privilege escalation) bir işleve sahipken, 1172'si uzaktan kontrol sağlamaya, 1132 örnek ise SMS veya çağrı yoluyla finansal kazanç sağlamaya ve 744'ü kişisel bilgileri çalmaya yönelik olduğu gözlenmiştir.

Ayrıcalık yükseltme işleminde kullanılan komut örnekleri

Numara	Komut	Tanım
1	su	Geçerli kullanıcıyı kök olarak değiştirme
2	mount -o remount	Salt okunur bir dosya sistemini yazılabilir yap
3	sh	Bir kabuk komutu çalıştır
4	insmod	Yüklenebilir bir çekirdek modülü yükle
5	reboot	Cihazı yeniden başlat
6	zergRush, m7, fre3vo, psneuter, wpthis, exploid, rageagainstthecage, motofail, GingerBreak	Kök istismar anahtar kelimeleri

Şekil 3.2. Ayrıcalık arttırıcı komutlar [14]

Farklı sektörleri amaçlayan kötücül yazılımların kullandığı izinler Şekil 3.3'te gösterilmiştir.



Şekil 3.3. Kötücül yazılımların farklı sektörlerde kullandığı izinler [14]

Resmi ve resmi olmayan uygulama dağıtım kanallarında aynı sektörde kullanılan zararlı ve zararsız uygulamaların kullandığı izinler Şekil 3.4'te gösterilmektedir.

Uyg.	Resmi market	Resmi olmayan market
KFH bank	INTERNET	READ_SOCIAL_STREAM, WRITE_SOCIAL_STREAM, USE_CREDENTIALS, ACCESS_FINE_LOCATION, INTERNET, WAKE_LOCK WRITE_EXTERNAL_STORAGE READ_HISTORY_BOOKMARKS, SET_WALLPAPER, VIBRATE RECEIVE_BOOT_COMPLETED, RESTART_PACKAGES, ACCESS_NETWORK_STATE, ACCESS_WIFI_STATE, CHANGE_WIFI_STATE, ACCESS_COARSE_LOCATION, WRITE_HISTORY_BOOKMARKS, READ_LOGS
S bank	INTERNET WRITE_EXTERNAL_STORAGE	INTERNET, WAKE_LOCK ACCESS_NETWORK_STATE ACCESS_COARSE_LOCATION ACCESS_FINE_LOCATION ACCESS_LOCATION_EXTRA_READ_PHONE_STATE RECEIVE_BOOT_COMPLETED
axis bank	INTERNET ACCESS_NETWORK_STATE	INTERNET, SEND_SMS READ_PHONE_STATE GET_TASKS

Şekil 3.4. Resmi ve resmi olmayan marketlerdeki uygulamaların kullandığı izinler [14]

Resmi ve resmi olmayan uygulama dağıtım kanallarında aynı sektörde kullanılan zararlı ve zararsız uygulamaların kullandığı komutlar Şekil 3.5'te gösterilmektedir.

Komut	Numara	Tanım
chmod	578	İzni değişir
insmod	16	Modülü çekirdeğe yerleştir.
su	32	Anlık kullanıcıyı süper kullanıcı olarak değiştir
mount	90	Bir dosya sistemi ekle
sh	100	Varsayılan kabuğu çağır
chown	46	Dosyanın veya dizinin sahibini değiştir
killall	31	Tüm işlemleri öldür
reboot	22	Sistemi yeniden başlatın
hosts	28	Verilen alan adının IP adresini bul
getprop	347	Kullanılabilir sistem özelliğini al
mkdir	16	Dizin oluştur
ln	1193	Belirtilen hedefe dosya veya dizin adlı bir bağlantı oluştur
ps	1199	İşlem durumunu raporla

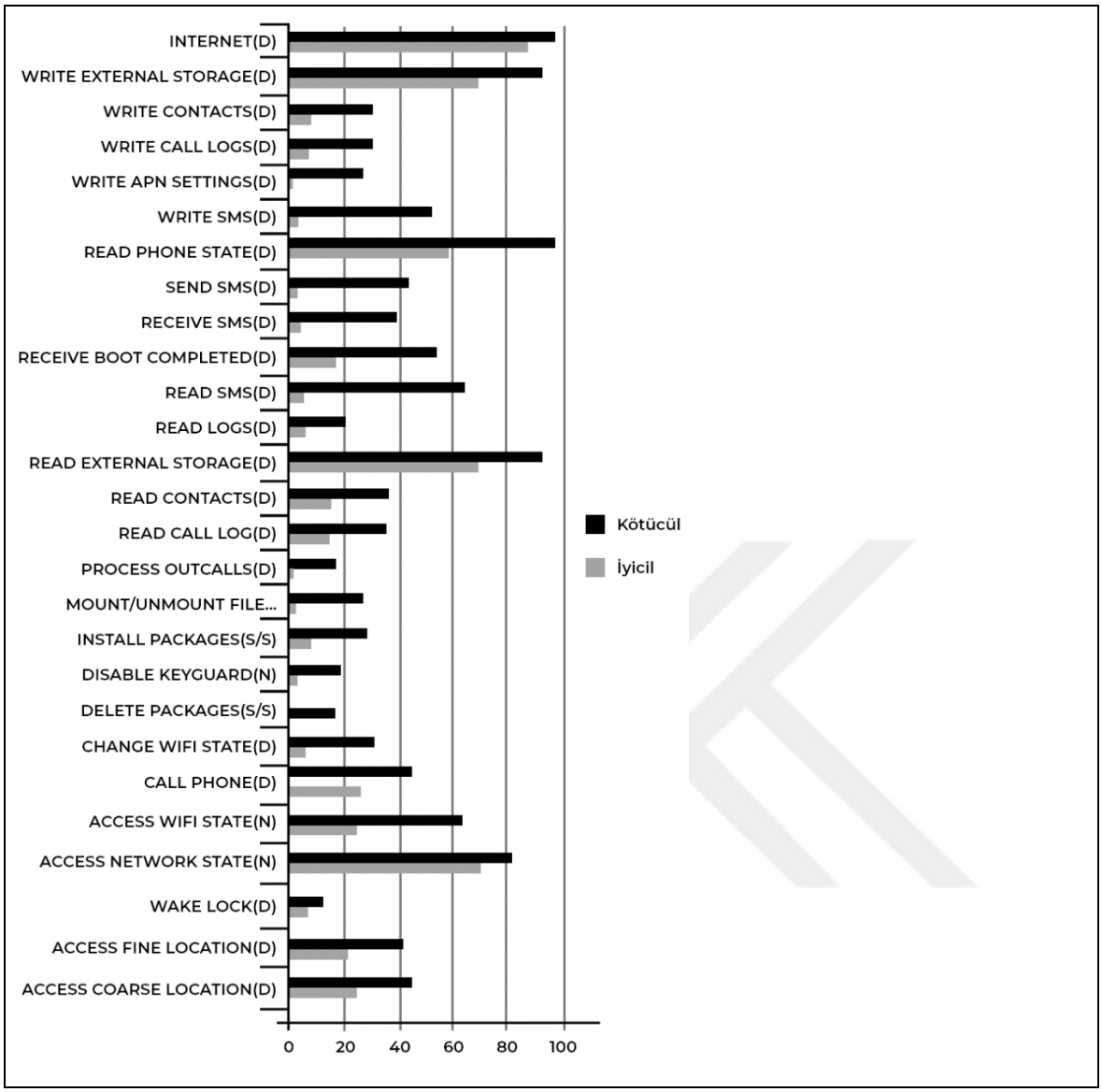
Şekil 3.5. Resmi ve resmi olmayan marketlerdeki uygulamaların kullandığı komutlar [14]

Quang Do ve arkadaşlarının çalışması

Quang Do ve arkadaşlarının [15], 2014'te yaptığı çalışmada, gizli veri sızıntılarını önlemek için muhalif bir model önerilip, gizlice sızdırılan verilerin önlenme teknikleri gösterilmiştir. Mobil veri sızıntısını engelleme tekniği (Mobile Data Exfiltration Technique - MDET) çeşitli ortamlarda sızıntıları engellemek ve Android cihazlarda ikili (binary) verileri ayıklayarak kod enjekte etme (code injection) metotlarını desteklemek amacıyla tasarlanmıştır. Bu işlem ikili kod seçme ve analiz (binary code selection ve analysis), SMALI kod enjekte etme ve düzenleme (code injection ve modification) ve ikili yeniden derleme (binary recompilation) olmak üzere üç ana aşamadan oluşmaktadır. İspat olarak, standart mobil cihazlar kullanılarak SMS ve işitilemez ses aktarımı gibi yöntemlerle veri sızıntısını önleme yöntemlerinin uygulanabilirliğinin gösterimi sağlanmıştır.

Shina Sheen ve arkadaşlarının çalışması

Shina Sheen ve arkadaşlarının [16] 2014'te yaptığı çalışmada, kötücül yazılım analizi için Android tabanlı zararlı yazılım ve çok özellikli işbirlikçi karar füzyon (Multifeature Collaborative Decision Fusion - MCDF) yapısı kullanılarak ölçeklenebilir bir tespit mekanizması göz önüne alınmıştır. Statik analiz kullanılarak AndroidManifest.xml dosyası analiz edilerek uygulamaların kullandığı izinler ve API çağrıları elde edilmiştir. İzin ve API çağrıları tabanlı özellikler kullanılarak, bir sınırlayıcı topluluk oluşturulup ve bunların kararları birleştirilerek daha iyi bir metot belirlenmeye çalışılmıştır. Zararlı ve zararsız yazılımlardaki en sık kullanılan izinlerin bazıları oluşum yüzdesi olarak Şekil 3.6'da gösterilmektedir.



Şekil 3.6. Zararlı ve zararsız yazılımlardaki en sık kullanılan izinlerin oluşum yüzdesi [16]

Şekil 3.6'da da görüldüğü gibi sistem eğitim aşamasında sistem hem iyicil hem de kötücül Android uygulamaları eğitim veri setinde ele almaktadır ve otomatik olarak kötü amaçlı davranış kalıplarını araştırmaktadır. Eğitim aşamasında, bir sınıflandırıcı ve işlenmiş veri akışıyla ilgili API listesi genelleştirilmiştir. Tanımlama aşamasında, sistem bilinmeyen bir Android uygulamayı girdi olarak alır, özellik vektörlerini çıkarır ve uygulamanın kötü amaçlı olup olmadığına karar vermektedir. Sistem hassas veri aktarım yollarını keşfetmek için işlenmiş API listesini kullanarak kötücül uygulamada statik veri akış analizi yapmaktadır.

Ping Wang ve arkadaşlarının çalışması

Ping Wang ve arkadaşlarının [17] 2015'te yaptığı çalışmada imza tabanlı analiz, semantik analiz ve makine öğrenmesi kullanılarak Android kötücül yazılım tespiti için çok küçük bir hata olabileceği vurgulanarak destek vektör modeli sunulmuştur. Bu çalışmada davranışsal imzalara dayalı SVM sınıflandırıcı eğitilerek otomatik malware algılama sistemi geliştirilmiştir. Sınıflandırmadaki doğruluk problemlerini çözmek için 60 tane gerçek kötücül yazılım aileleriyle çapraz doğrulama şeması ile ilişkili SVM kullanılmıştır. Deneysel sonuçlara göre, test verilerinin boyutu arttıkça sınıflandırma hatası azalmıştır. Farklı boyuttaki kötü amaçlı yazılım örnekleri için kötücül yazılım tahminindeki doğruluk 100 örnek için %98,7'ye kadar çıkmaktadır. Ayrıca belirsiz mobil kötü amaçlı yazılımlara karşı SVC genel algılama doğruluğunun %85'ten fazla olduğu gözlenmiştir.

APK auditor

APK Auditor [18], uygulama izinlerini kullanarak kötücül yazılım tespiti yapan izin tabanlı bir sistemdir. Uygulamaları iyicil veya kötücül olarak gruplandırmak ve sınıflandırmak için statik analiz yöntemini kullanmaktadır.

Kang ve arkadaşlarının çalışması

Kang ve arkadaşları [19], kötücül yazılım tespiti için uygulama geliştiricilerinin bilgilerini kullanan statik analiz tabanlı bir çalışma sunmuşlardır. Yapılan çalışmadan kötücül uygulamaların seri numaraları ile geliştiricilerin uygulamalarının seri numaraları karşılaştırılarak uygulamaların güvenlik durumunu tespit etmeyi amaçlamışlardır.

Anwar ve arkadaşlarının çalışması

Anwar ve arkadaşları çalışmalarında [20], mobil botnet algılamayı amaçlayan statik analiz tabanlı bir sistem önermektedirler. Geliştirilen sistem uygulama izinleri, alıcıları ve arka plan servisleri gibi özellikleri kullanarak uygulamaları sınıflandırmaktadır.

DroidOL

DroidOL [21], kötü amaçlı yazılımları tespit etmeyi amaçlayan makine öğrenmesi tabanlı çevrimiçi hizmet sunan uygulamadır. Geliştirilen sistem üç aşamadan geçilerek geliştirilmiştir. İlk aşamada, static analiz yöntemi kullanılarak işlemler arası control akış grafikleri oluşturulmuştur. İkinci aşamada, Weisfeiler-Lehman çekirdeğinden yararlanarak işlemler arası kontrol akış grafiklerinin alt grafikleri oluşturulmuştur. Son olarak ise çevrimiçi bir pasif agresif sınıflandırıcısı kullanılarak ilk iki aşamada çıkartılan özellikler yardımıyla kötücül yazılım tespit edebilmek için eğitilmektedir.

AndroDialysis

AndroDialysis çalışmasında [22], uygulamanın AndroidManifest.xml dosyasındaki niyetlerin analizi yapılarak kötücül yazılım tespit etmeyi amaçlamaktadır. Statik analiz yöntemi ile yapılan çalışmada kullanılan niyetlerin izinlere göre daha etkili olduğu savunulmuş olmasına rağmen niyetlerin kötücül yazılım tespitinde izinlere kıyasla daha başarısız olacağı vurgulanmıştır.

Sokolova ve arkadaşlarının çalışması

Sokolova ve arkadaşları çalışmalarında [23], grafik tabanlı izin örüntüleri ile Android uygulama sınıflandırmayı ve anormal davranışları tespit etmeyi amaçlamışlardır. Kategorize edilen uygulamaların davranışları analiz edilerek, beklenen davranışlar ile karşılaştırılmaktadır. Ek olarak kategorik uygulamalar ve kullanılan izinler grafik analiz metrikleri kullanılarak elde edilmiştir. Elde edilen modeller, kategorik olarak sınıflandırılan uygulamaların performansına göre değerlendirilmektedir.

Songyang Wu ve arkadaşlarının çalışması

Songyang Wu ve arkadaşları çalışmalarında [24], izin sızıntılarını tespit eden statik analiz tabanlı bir yaklaşım önermektedirler. Yapılan çalışmada geliştiriciler tarafından Android uygulamalara eklenen ama kullanılmayan izinlerin tespitini yaparak kötücül uygulamaların bu izinleri kullanmasını engellemeyi amaçlamışlardır. Yapılan çalışmada 550 uygulama kullanılmıştır. Sonuçlar, önerilen yöntemin başarı oranının %68 olduğunu göstermektedir.

Jungsoo Park ve arkadaşlarının çalışması

Jungsoo Park ve arkadaşları çalışmalarında [25], Android kötücül yazılım tespiti için API ve izin tabanlı sınıflandırma yöntemini önermektedirler. Önerilen yöntemde uygulamalar iyicil, kötücül ve şüpheli olmak üzere üç kategoriye ayrılmaktadır. API ve izin tabanlı sınıflandırma sistemi YARA adı verilen kurala göre oluşturulmaktadır. Her uygulamanın nitelikleri AndroidManifest.xml ve classes.dex dosyasından çıkartılarak YARA kuralı ile eşleştirilmektedir.

Durmuş Özkan Şahin ve arkadaşlarının çalışması

Durmuş Özkan Şahin ve arkadaşları çalışmalarında [26], Android kötücül yazılım tespiti için diğer çalışmaların aksine izin tabanlı statik analiz yöntemini kullanarak izin ağırlıklı yaklaşımı önermektedirler. Önerilen yöntemde her bir izne birbirinden farklı puanlar verildikten sonra k-NN ve NB algoritmaları uygulanarak önerilen yöntem önceki çalışmalar ile karşılaştırılmaktadır. Deneysel sonuçlar, önerilen yöntemin başarı oranının %96,64 olduğunu göstermektedir.

Arslan ve arkadaşlarının çalışması

Arslan ve arkadaşları çalışmalarında [27], uygulamanın ihtiyaç duymadığı halde istenilen izinlerin şüpheli faaliyetler için kullanılabileceğini savunmaktadırlar. Bu şüphe uyandıran yedek izinleri tespit etmek için statik ve kod analizi yöntemini kullanmaktadırlar. Deneysel sonuçlar, önerilen yöntemin başarı oranının %91,65 olduğunu göstermektedir.

Zhu ve arkadaşlarının çalışması

Zhu ve arkadaşları çalışmalarında [28], izinler, hassas API'ler, monitör edilebilir sistem eventleri ve izin oranlarını statik analiz yöntemi ile kullanarak bir Android uygulamanın kötücül olup olmadığını tespit etmeyi amaçlamaktadırlar. Yapılan çalışmada önerilen yöntemin performansını doğrulamak için 2130 uygulamadan oluşan veri seti kullanılmaktadır. Deneysel sonuçlar, önerilen yöntemin doğruluğunun %88,26 olduğunu göstermektedir.

İbrahim Alper Doğru ve Ömer Kiraz'ın çalışması

İbrahim Alper Doğru ve Ömer Kiraz çalışmalarında [29], statik analiz yöntemini kullanarak web tabanlı Android kötücül yazılım tespit sistemi geliştirmişlerdir. Geliştirilen sistem AndroidManifest.xml dosyasında bulunan izinleri kullanarak uygulamanın iyicil mi kötücül mü olduğu hakkında bilgi vermektedir. Yapılan çalışmada birbirinden farklı dört hesaplama yöntemi kullanılmışlardır. Yapılan hesaplamalardan VirusTotal ile birlikte iyicil-kötücül veri setlerindeki kullanılan izinlerle yapılan hesaplama yöntemi %97,73 oranıyla en yüksek başarıyı göstermiştir.

3.2. Dinamik Analiz Yöntemi

Dinamik analiz yöntemi çalışan bir uygulamanın sergilediği davranışları izleyerek analiz yapmayı sağlayan yöntemdir. Dinamik analiz yöntemi kullanılırken doğru bir analiz sonucu elde etmek için herhangi bir zaman kısıtı olmamalıdır. Örneğin, 1-5 gün arası verildiğinde, saldırgan güncellemeyi 6. gün yaptığı zaman dinamik yöntem kötücül yazılımı tespit edemez. Dinamik analiz yaklaşımı diğer analiz yaklaşımlarına göre daha geniş bir bakış açısına sahip olmasına rağmen daha maliyetli olduğu için daha az tercih edilen bir yöntemdir [30].

Burguera ve arkadaşlarının çalışması

Burguera ve arkadaşlarının [31] 2011'de yaptığı çalışmada, uygulama üzerinde normal olmayan hareketleri tespit eden Crowdroid'i sunmuşlardır. Bu sistem, Strace komutunu kullanıp sistem çağrılarının derler ve bu yöntemle uygulamanın iyi niyetli veya kötü niyetli olarak sınıflandırır.

Zhou ve Jiang'ın çalışması

Zhou ve Jiang [32] tarafından 2012 yılında yapılan çalışmada, 49 tane birbirinden farklı türden oluşan kötü niyetli yazılımlardan oluşturulan 1260 tane kötü niyetli yazılım kullanılmışlardır. Zhou ve Jiang dinamik analiz tekniğini kullanarak kötücül yazılımların yüklenmesi, aktiviteler ve bunlar arasında taşınan veriler gibi çeşitli davranış biçimleri incelenmiştir.

TaintDroid

TaintDroid [33], William Enck ve arkadaşları tarafından geliştirilen açık kaynak kodlu dinamik analiz aracıdır. Uygulamaların hassas bilgi gereksinimlerini nasıl kullandığını takip eder. Uygulama içerisindeki veri sızmalarına odaklanmıştır.

Hooker

Hooker [34], Android uygulamalarının otomatik dinamik analizini yapar ve açık kaynak kodludur.

Decafplatform

Decafplatform [35], dinamik analiz metodunu kullanarak anormal davranışları gözlemleyen DECAF Binary Analiz Platformu'dur. DroidScope bir eklentisidir.

ZjDroid

ZjDroid [36], Xposed çatısına dayanan tersine mühendislik yöntemlerini kullanan dinamik analiz aracıdır ve python ile yazılmıştır.

Droidbox

Droidbox [37], 2011 yılında Patrik Lantz tarafından geliştirilen, dinamik analizi kullanarak uygulamanın davranışlarını analiz eden araçtır.

Shaptai ve arkadaşlarının çalışması

Shaptai ve arkadaşlarının 2014'te yaptığı çalışmada [38] mobil uygulamaların network davranışları incelenmiştir. Yarı denetimli makine öğrenme yöntemleri, uygulamanın normal davranış kalıplarını öğrenmek ve uygulamanın beklenen davranış sapmalarını tespit etmek için kullanılmıştır. Kendini güncelleyebilen kötücül yazılımların tespit edilmesi amaçlanmıştır. Yapılan çalışmada aynı türden uygulamaların ağ davranışlarının benzerlik göstereceği vurgulanmıştır. Yapılan testler bu düşünceyi doğrulamaktadır.

Andro-Profiler

Andro-Profiler [39] çalışmasında, sistem çağrıları dahil olmak üzere entegre sistem günlüklerinden çıkarılan davranış profillerini kullanarak kötücül yazılımları sınıflandırmaktadır. Andro-Profiler, entegre sistem günlüklerini oluşturmak için sanal cihazda kötü niyetli bir uygulamayı çalıştırır ve entegre sistem günlüklerini analiz ederek insan tarafından okunabilen davranış profillerini oluşturmaktadır.

Garg ve arkadaşlarının çalışması

Garg ve arkadaşları çalışmalarında [40], kötücül uygulamaları ağ üzerinde gözlemci bir göz ile ağ etkinliğine bakarak tespit etmeye çalışan bir sistem geliştirilmiştir. Ayrıca geliştirilen model ağ izlerini kullanarak kötücül uygulamaları tespit etmek, işletim sistemlerinin farklı sürümleriyle çalışmak, bilinmeyen uygulamaları tespit etmek ve şifrelenmiş verilerle virüs bulaşmış uygulamaları tespit etme yeteneklerine sahiptir.

Chang ve arkadaşlarının çalışması

Chang ve arkadaşları çalışmalarında [41], makine öğrenmesi kullanılarak davranış bazlı kötücül yazılım algılama sistemi önerilmektedir. Bu çalışmada DroidBox [36] yapısına ek olarak otomatik tetiklenen görünüm tanımlama programı eklenmiştir. Bu program sayesinde mobil uygulamalara anlamlı sıraya göre ulaşılmaktadır. Geliştirilen sistem; ön işleme, veri izleme, karar modeli ve veritabanı olmak üzere dört bileşenden oluşmaktadır.

3.3. Hibrit Analiz Yöntemi

Statik ve dinamik analiz yöntemlerinin birlikte kullanılarak kötücül yazılım tespit etmeyi amaçlayan analiz yöntemine hibrit analiz yöntemi denir [42].

Xiaolei Wang ve arkadaşlarının çalışması

Xiaolei Wang ve arkadaşlarının [43] 2015'te yaptığı çalışmada, anormallik tespiti ile kötüye kullanım tespitini entegre eden yeni bir hibrid mobil kötücül yazılım tespit sistemi geliştirilmiştir. Statik analiz kullanılarak uygulamanın AndroidManifest.xml dosyasındaki

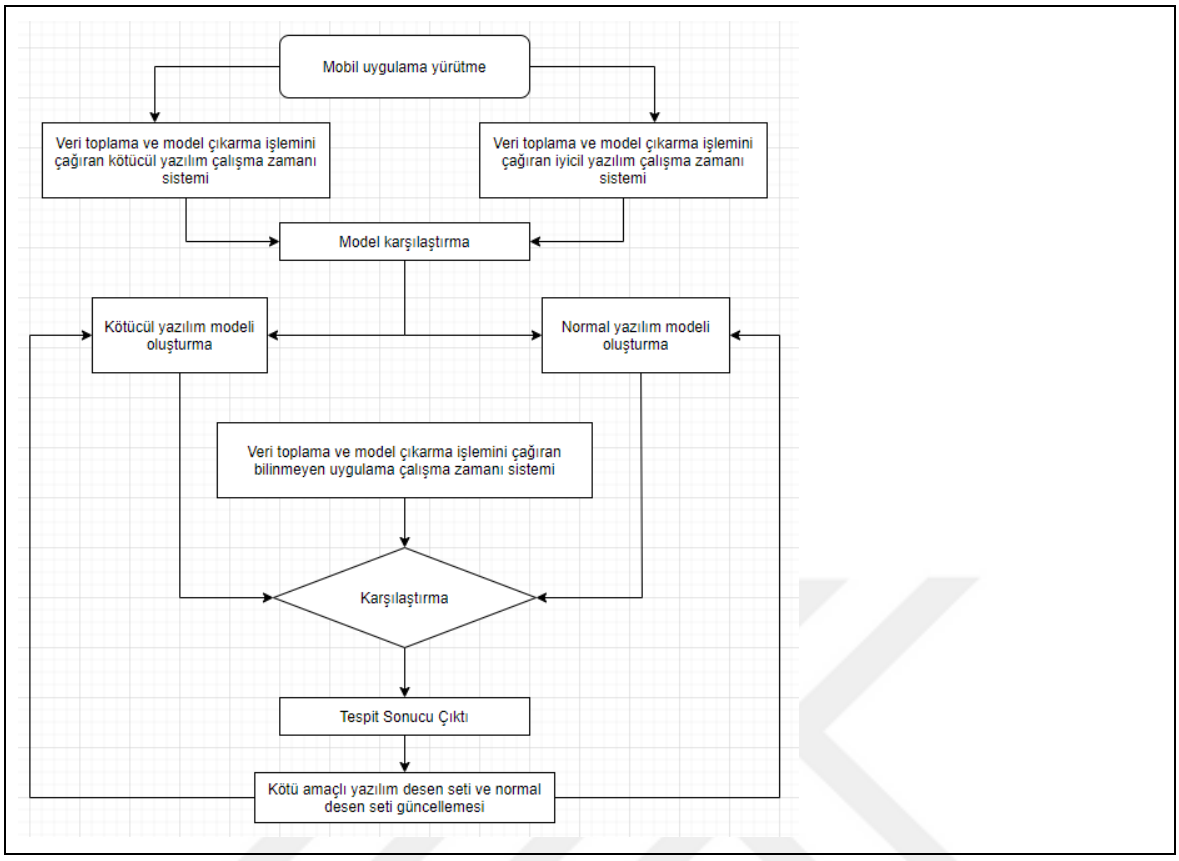
statik özellikler çıkarılmıştır. Çalışma esnasında uygulamanın dinamik özelliklerinin çıkarılması için emülatör ortamında uygulamaları çalıştıracak CuckooDroid kullanılmıştır. Eş zamanlı olarak dinamik API çağrıları belirlenmiştir. Dinamik ve statik analizle elde edilen veriler makine öğrenmesi yöntemlerini besleyerek kötücül yazılım tespiti sağlanmıştır. 5560 kötücül yazılım örneği ve 12000 iyicil yazılım örneği ile çalışma yapılmıştır. Yapılan deneyler sonucunda %98,32 oranında başarı elde edilmiştir.

Gini ve arkadaşlarının çalışması

MADAM (A Multi-level Anomaly Detector for Android Malware) [44] çok katmanlı bir tespit mekanizması sunmaktadır. Uygulamalar makine öğrenmesi algoritmaları da kullanılarak hem çekirdek seviyesinde hem de uygulama seviyesinde takip edilmektedir. API çağrıları, kullanılan RAM miktarı ve uygulamanın kullandığı CPU analiz edilmektedir. Uygulama seviyesinde ise kullanıcıların pasif olduğu durumlarda kullanılan işlemler, alınan veya gönderilen SMS'ler, indirilme veya silme işlemleri analiz edilerek uygulamanın kötücül olma durumu değerlendirilmektedir.

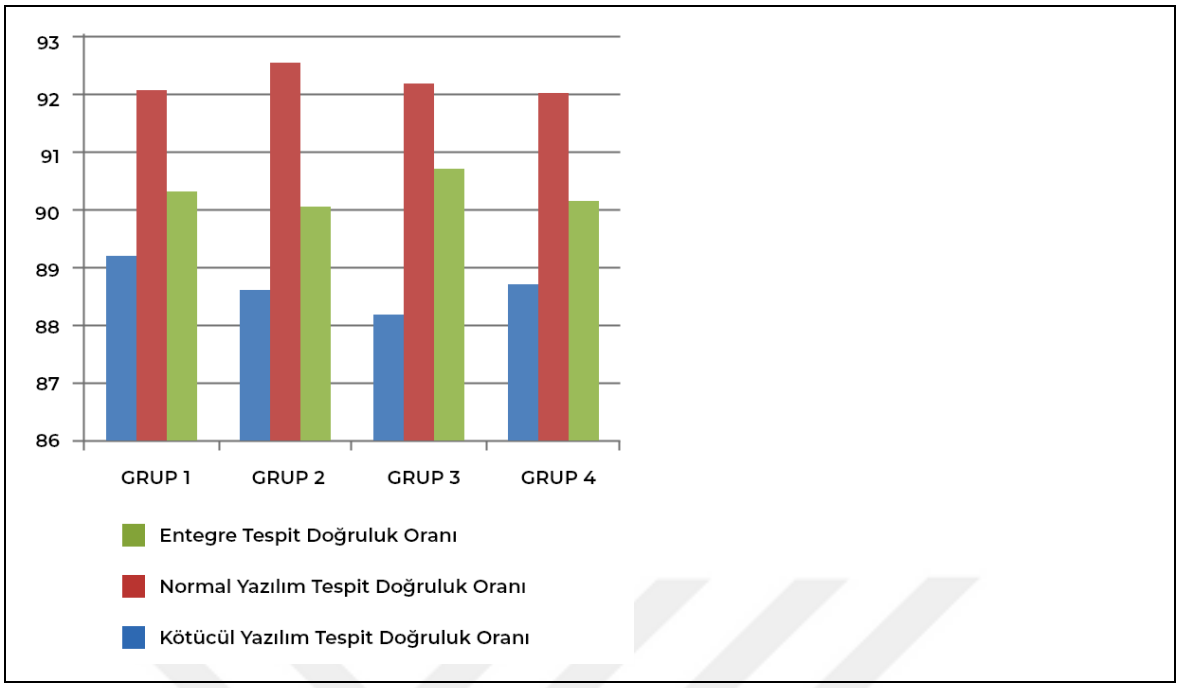
Fei Tong ve Zheng Yana'ın çalışması

Fei Tong ve Zheng Yana [45] 2016'da yaptıkları çalışmada hem statik hem de dinamik analiz tekniğini kullanılarak mobil kötücül yazılım tespiti için yeni bir hibrid model önermişlerdir. Yaptıkları çalışmada dosya ve ağ erişimi ile ilgili sistem çağrılarının kalıplarını oluşturmak için net_link teknolojisini kullanarak kötücül ve iyicil uygulama örneklerinden veri toplamışlardır. Ayrıca kötücül ve iyicil uygulamaları birbirleriyle karşılaştırarak bir kötücül kalıp bir de normal kalıp oluşturmuşlardır. Oluşturulan bu kalıplardan yararlanarak bilinmeyen bir uygulamanın türünü belirlemişlerdir. Geliştirilen sistemin akış diyagramı Şekil 3.7'de gösterilmektedir.



Şekil 3.7. Tespit yöntemi [45]

Yapılan çalışma sonunda elde ettikleri sonuçlara dayanarak kendi yöntemlerinin diğer statik veya dinamik analiz yöntemini kullanan yaklaşımlara göre daha iyi algılama oranına sahip olduğunu vurgulamışlardır. Farklı gruplardaki bilinmeyen uygulamaların algılama doğruluğu oranları Şekil 3.8'de gösterilmektedir.



Şekil 3.8. Bilinmeyen uygulamaların farklı gruplarının algılama doğruluğu oranları [45]

Abdullah Talha Kabakus ve İbrahim Alper Doğru'nun çalışması

Abdullah Talha Kabakus ve İbrahim Alper Doğru çalışmalarında [46], hibrid teknikleri kullanarak Android kötücül yazılımların derin analizini gerçekleştirmişlerdir. Yapılan çalışmada kötü amaçlı yazılımları tespit etmek için statik ve dinamik analiz tekniklerini birleştiren mad4a olarak isimlendirilen uygulama çatısı ile uygulamaların karakteristiklerini ortaya koymayı amaçlamışlardır. Yapılan çalışmanın statik analiz kısmına uygulama izinleri, dinamik analiz kısmında ise emulatörde çalıştırılan uygulamaların davranışları incelenmiştir.

3.4. Literatürdeki Çalışmaların Yeteneklerinin Karşılaştırılması

Bu çalışmada incelenen literatürdeki çalışmaların yetenek karşılaştırmaları Çizelge 3.1'de gösterilmiştir.

Çizelge 3.1. Literatürdeki çalışmaların yetenek karşılaştırılması

Yetenekler	Statik Analiz	Dinamik Analiz	İmza Tabanlı Analiz	Makine Öğrenmesi	Hibrid Analiz	Çalışma Yılı
Felt ve ark. [6]	Var	Yok	Yok	Yok	Yok	2011
Shaowu Liu ve ark. [13]	var	Yok	Yok	Yok	Yok	2014
Quang Do ve ark. [15]	Var	Yok	Yok	Yok	Yok	2014
Kang ve ark. [19]	Var	Yok	Yok	Yok	Yok	2015
Anwar ve ark. [20]	Var	Yok	Yok	Yok	Yok	2016
Songyang Wu ve ark. [24]	Var	Yok	Yok	Yok	Yok	2017
Durmuş Özkan Şahin ve ark. [26]	Var	Yok	Yok	Var	Yok	2018
Arslan ve ark. [27]	Var	Yok	Yok	Yok	Yok	2019
İbrahim Alper Doğru ve Ömer Kiraz [29]	Var	Yok	Yok	Yok	Yok	2018
Fei ve ark. [45]	Var	Var	Yok	Yok	Var	2016
Burguera ve ark. [31]	Yok	Var	Yok	Yok	Yok	2011
Zhou ve Jiang'ın [32]	Yok	Var	Yok	Yok	Yok	2012
Hooker [34]	Yok	Var	Yok	Yok	Yok	2014
Decafplatform [35]	Yok	Var	Yok	Yok	Yok	
Shaptai ve ark. [38]	Yok	Var	Yok	Var	Yok	2014
Garg ve ark. [40]	Yok	Var	Yok	Yok	Yok	2016
Chang ve ark. [41]	Yok	Var	Yok	Var	Yok	2016
Ping Wang ve Yu-Shih Wang'ın [17]	Yok	Yok	Var	Var	Yok	2014
Abdullah Talha Kabakuş ve İbrahim Alper Doğru [46]	Var	Var	Yok	Yok	Var	2018

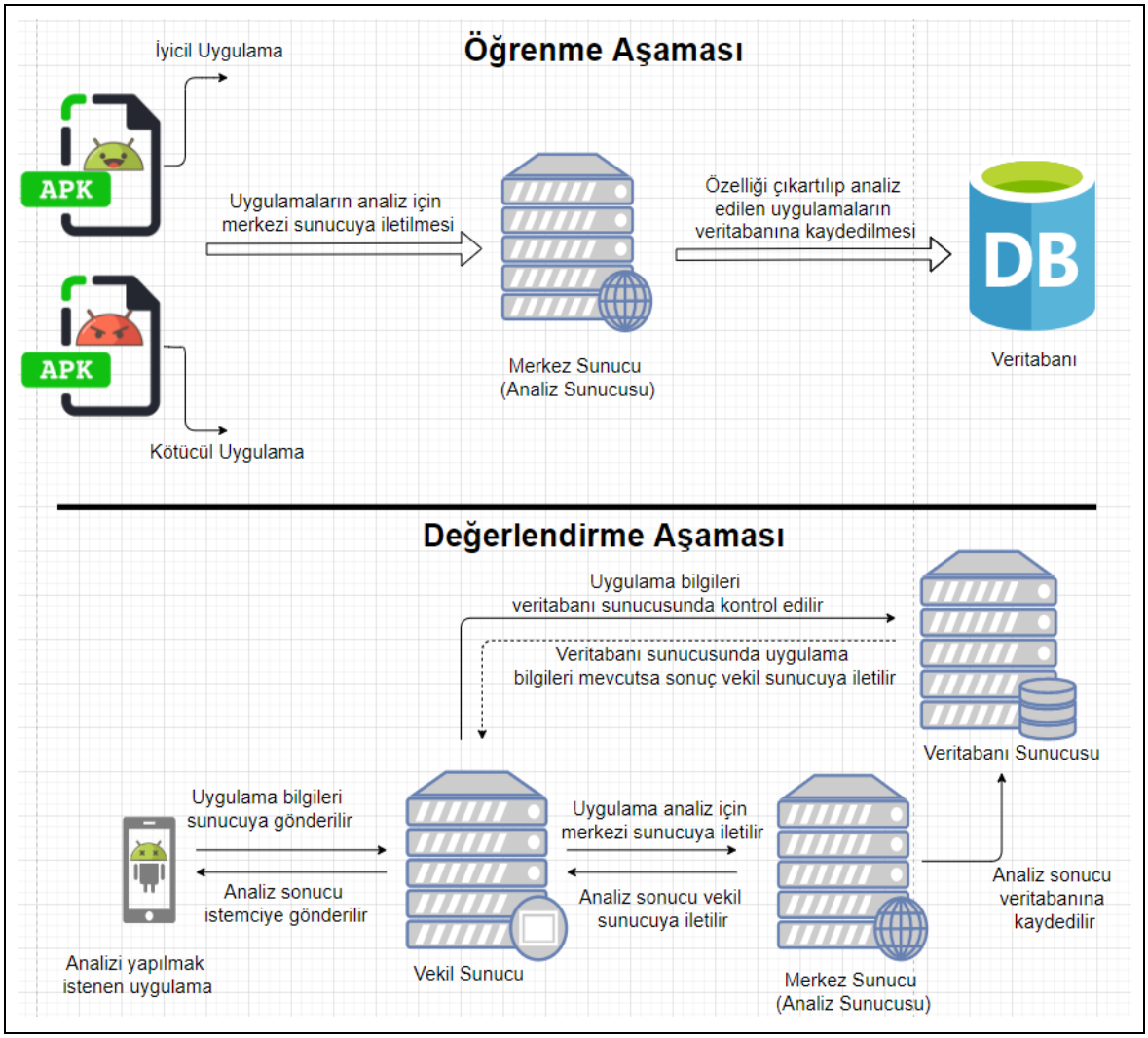
4. KULLANILAN YÖNTEM VE MATERYAL

Bu çalışmada, makine öğrenme teknikleri kullanılarak kötücül yazılım tespit etmeyi sağlayan istemci ve sunucu tabanlı kullanıcı dostu bir platform geliştirilmiştir. Geliştirilen sistem ile kullanıcıları kullandıkları uygulamaların güvenilirliği hakkında bilgilendirilmesi amaçlanmıştır. Bu doğrultuda uygulamaların AndroidManifest.xml dosyalarında tanımlanan izinler ve hedef sdk versiyonları tersine mühendislik yöntemleri ile elde edildikten sonra makine öğrenme teknikleri yardımıyla uygulamaların karekteristikleri çıkartılıp uygulamaların iyicil / kötücül olduğu belirlenmektedir. Sınıflandırma için makine öğrenme algoritmalarından Gradyan Arttırıcı Sınıflandırma, Destek Vektör Makinesi, Rastgele Orman ve K En Yakın Komşu algoritmaları kullanılmıştır. Kullanılan algoritmalar arasında en başarılı sonuçların Gradyan Arttırıcı Sınıflandırma Algoritması'ndan elde edildiği gözlenmiştir. Gradyan Arttırıcı Sınıflandırma Algoritması rekresyon ve sınıflandırma problemleri için kullanılan makine öğrenmesi algoritmalarından biridir. Yapılan çalışma, iyicil veya kötücül uygulamaların kullandıkları izinleri makine öğrenme algoritması ile analiz edip analiz sonucunu kullanıcıya bildirmektedir. Buna ek olarak uygulama kullanıcıya mobil cihazında kurulu olan uygulamaların sahip olduğu izinleri de listelemektedir. Yapılan bu çalışma ile kullanıcılara telefonlarındaki uygulamaların türü ve bu uygulamaların kullandığı izinler hakkında gerekli bilginin verilmesi amaçlanmaktadır. Kötücül uygulama veri seti için açık kaynak olarak sunulan Drebin [9] veri seti kullanılmıştır. Geliştirilen kötücül yazılım tespit sisteminin doğru sonuçlar sunması için ilk olarak kullanılan kötücül veri seti incelenmiştir. İnceleme sonucunda 5545 kötücül uygulamanın 2823 tanesinin çoklanmış veri olduğu tespit edilmiştir. Bunun sonucunda 5545 veri setinin 2722 tanesi uygun görülüp kullanılmış ve 2823 tanesi çoklandığı için analizden çıkartılmıştır. Kötücül veri seti kullanıma uygun hale getirildikten sonra iyicil veri seti hazırlanmıştır. Bu aşamada, iyicil veri seti olarak 2018 yılında A. H. Lashkari ve arkadaşları[47] tarafından oluşturulup kullanıma sunulan CICAndMal2017 isimli veri seti kullanılmıştır. Sonuç olarak kullanılan veri setinin özeti Çizelge 4.1'de gösterilmiştir.

Çizelge 4.1. VirusTotal'e göre oluşan veri seti özeti

Veri Seti Türü	Sayısı	Kaynağı
Kötücül	2722	Drebin [8]
İyicil	3406	CICAndMal2017 [45]

Geliştirilen sistem dört bileşenden oluşmaktadır: (1) Android cihazlarda kullanılacak istemci uygulaması, (2) istemci uygulaması ile merkezi ve veritabanı sunucuları arasında iletişimi sağlayacak vekil sunucu, (3) uygulamaların özelliklerinin çıkartılıp analizlerinin yapıldığı merkezi sunucu, (4) analizi yapılan uygulamaların bilgilerinin tutulduğu veritabanı sunucusu. Geliştirilen sistemin mimarisi Şekil 4.1'de gösterilmiştir.

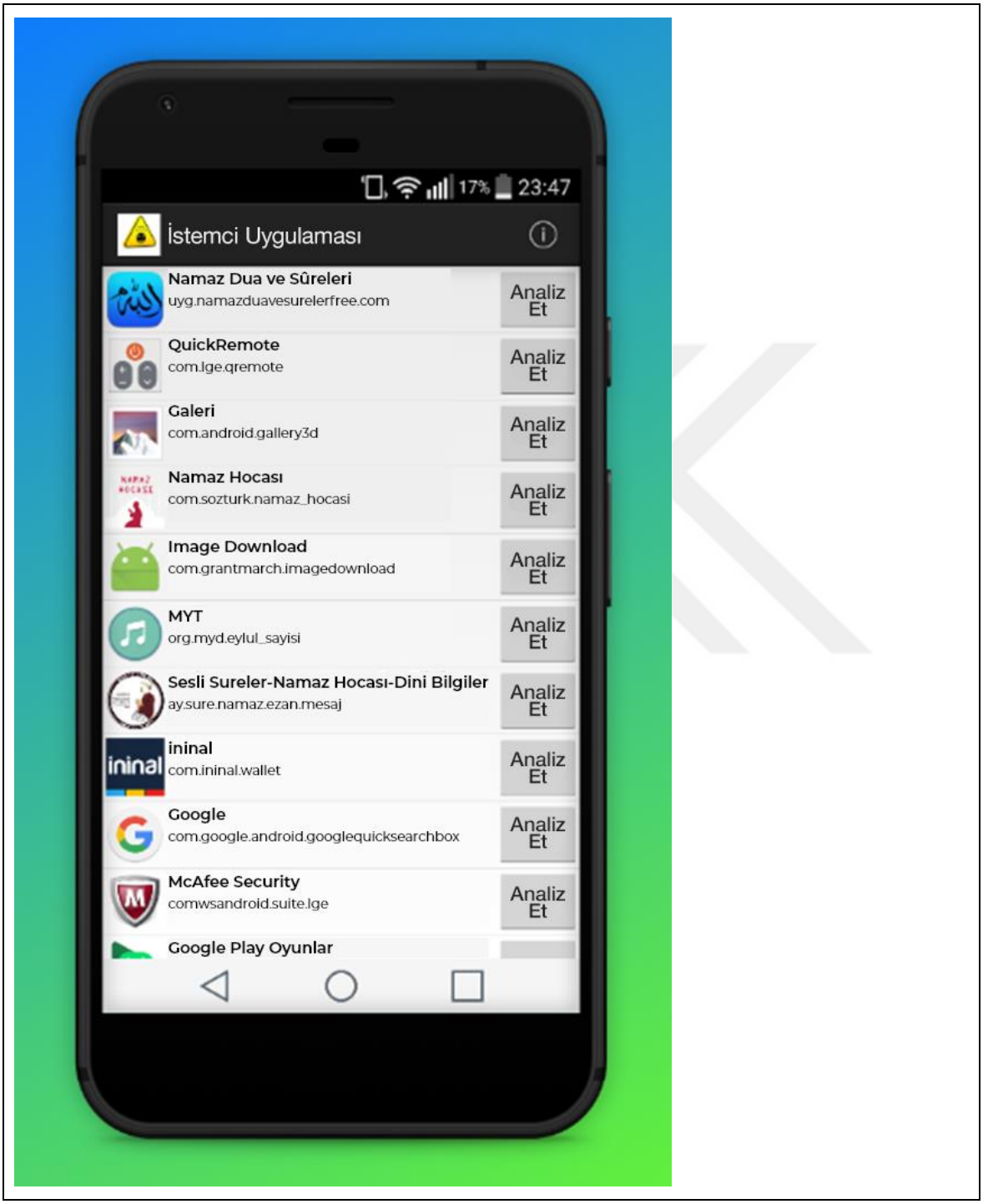


Şekil 4.1. Geliştirilen sistemin mimarisi

4.1. Android İstemci Uygulaması

Android akıllı cihazlarda yürütülecek bu uygulama ile kullanıcılara telefonlarında yüklü uygulamaların güvenlik durumu hakkında bilgi verilmesi amaçlanmaktadır. Uygulama ilk açıldığında kullanıcıya telefonunda yüklü olan uygulamalar listelenmektedir. Kullanıcı kendisine sunulan bu listeden istediği uygulamayı seçerek bu uygulama hakkında gerekli analiz sonuçlarını görebilmektedir. Şekil 4.2'de örnek bir Android akıllı cihazda kurulu

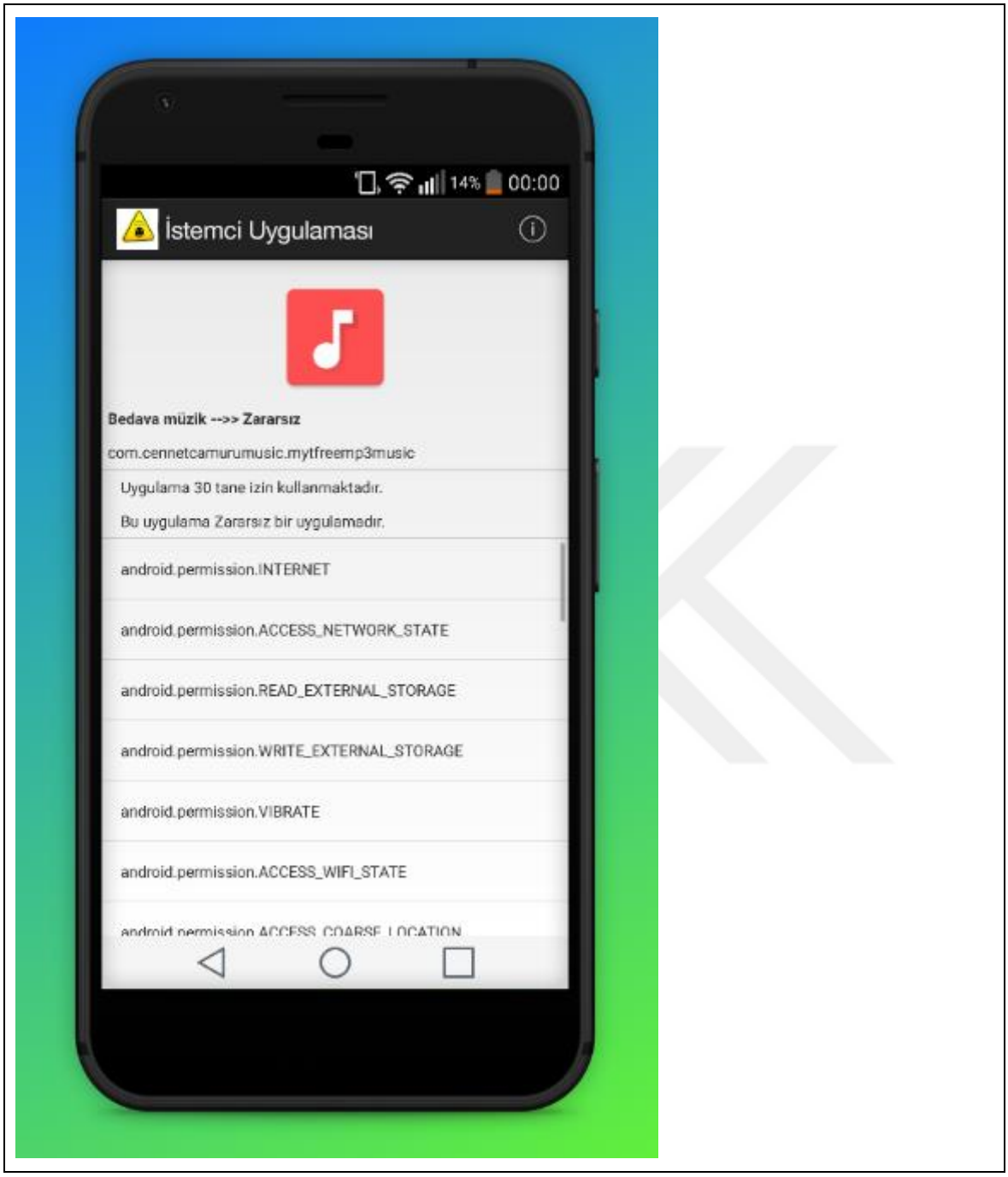
uygulamaların listesi istemci uygulamasının ana ekran kullanıcı arayüzü ile gösterilmektedir.



Şekil 4.2. İstemci uygulamasının ana ekran görüntüsü

Android cihaz kullanıcıları geliştirilen istemci uygulaması ile cihazındaki bir uygulamanın sahip olduğu izinleri ve uygulamanın güvenlik durumunu öğrenebilmektedir. Bir

uygulamanın güvenlik durumunu gösteren ekran görüntüsü Şekil 4.3'te gösterilmektedir.



Şekil 4.3. Analizi yapılan uygulamanın analiz sonuçlarını gösteren ekran görüntüsü

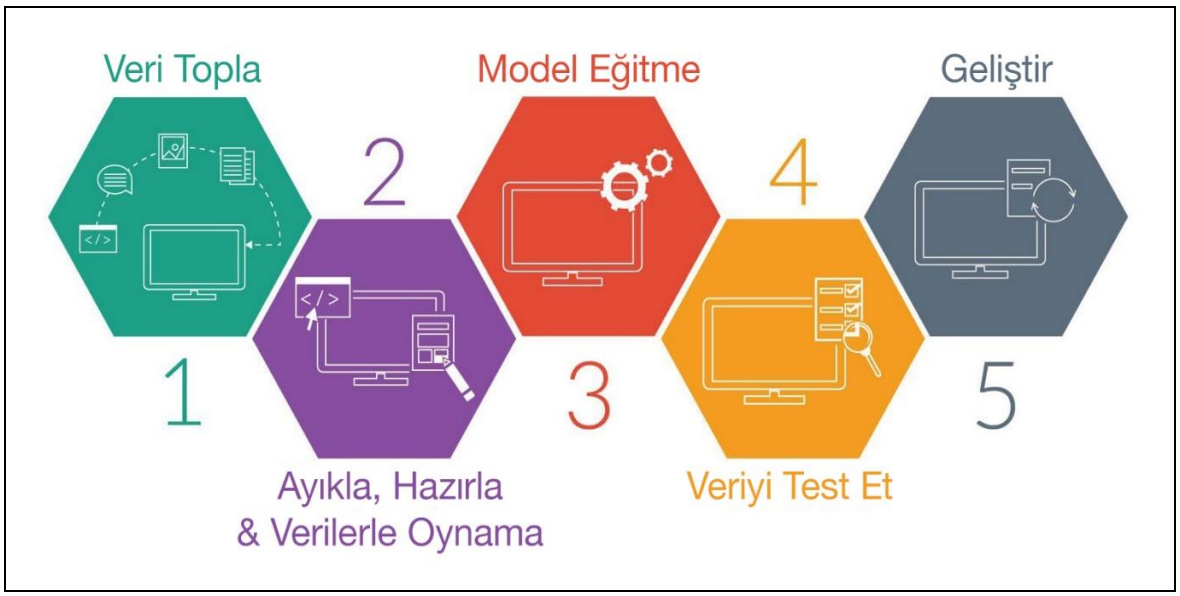
4.2. Vekil Sunucu

Vekil sunucunun ana görevi, merkez ve veritabanı sunucuları ile iletişim kurarak istemci uygulamasından aldığı isteklere cevap vermektir. Vekil sunucu üzerinde Java Spring Framework'ü kullanılarak geliştirilen RESTful servis çalışmaktadır. İstemciden alınan istek önce veritabanı sunucuna bağlanılarak analiz edilmek istenilen uygulamanın bilgisinin olup

olmadığı kontrol edilir. Analiz edilmek istenen uygulamanın bilgisi veritabanı sunucusunda mevcut ise istemciye uygulamanın veritabanında kayıtlı analiz sonuçları iletilir. Sonuç mevcut değil ise uygulamanın analiz edilmesi için merkez sunucuya bağlanır. Analizi tamamlanan uygulamanın bilgisini istemciye iletir.

4.3. Merkez Sunucu

Merkez sunucu, uygulamaların karakteristiğinin çıkarılıp analiz edildiği sunucudur. Bu sunucu da ilk olarak tersine mühendislik yöntemleri kullanılarak iyicil ve kötücül veri setlerinin karakteristikleri çıkartılıp veritabanına kaydedilmektedir. Tersine mühendislik yöntemi ile veritabanına kaydedilen özellikler, uygulamanın desteklediği minSdk ve targetSdk versiyonları ile birlikte uygulamaların kullandığı izinlerdir. İyicil ve kötücül uygulamaların özellikleri çıkartıldıktan sonra bu uygulamaların %60'ı analiz için eğitilen modellerin oluşturulmasında geri kalan %40 uygulama ise eğitilen modellerin testi için kullanılmıştır. Bu oranlar belirlenirken iyicil veri seti olarak kullanılan CICAndMal2017 isimli veri setinin oluşturulduğu A. H. Lashkari ve arkadaşlarının[47] 2018 yılında yapmış olduğu çalışmadaki oranlar baz alınmıştır. Bir makine öğrenimi görevini gerçekleştirmek için Şekil 4.4'te gösterilen 5 temel adımın uygulanması gerekmektedir [48].



Şekil 4.4. Makine öğrenimi algoritmalarının uygulanma adımları [48]

Analiz için kullanılan makine öğrenmesi algoritmaları Şekil 4.5'te gösterilen makine öğrenmesi algoritmalarının uygulanma adımlarına göre uygulanmıştır.

4.3.1. Veri ön işleme

Veri ön işleme, ham verileri temiz bir veri kümesine dönüştürmek için kullanılan bir tekniktir. Başka bir deyişle, veriler farklı kaynaklardan toplandığında, analiz için uygun olmayan ham formatta toplanır. Makine öğrenim projelerinde uygulanan modelden daha iyi sonuçlar elde etmek için verilerin formatı uygun bir şekilde olmalıdır. Bazı makine öğrenme modelleri belirli bir formatta bilgiye ihtiyaç duyar. Örneğin Rastgele Orman Algoritması boş değerleri desteklemez bu nedenle Rastgele Orman Algoritması yürütülmeden önce boş değerler orijinal ham veri kümesinden ayıklanır [49]. Yapılan bu çalışmanın veri ön işleme kısmında veriler One Hot Encoding formatına dönüştürülmüştür. One Hot Encoding, veri setindeki kategorik değişkenlerin ikili (binary) veriye dönüştürülerek kullanılmasıdır [50]. Bu dönüşüm yapılırken tüm kategorik değerler ikili sayılarla eşlenir ardından tüm değerler ikili vektör olarak temsil edilir. Yapılan işlemde özelliği çıkartılan izinler Boolean olarak 1 (izin kullanılıyor) veya 0 (izin kullanılmıyor) olacak şekilde ikili vektör olarak tanımlanmıştır. Ham verinin ikili vektör olarak tanımlanıp kullanılmasının temel sebebi kategorik verilerden daha anlaşılır ve etkileyici olmasıdır. Makine öğrenme algoritmalarının birçoğu kategorik veriler ile kullanılamadığından değişkenler sayılarla ifade edilmelidir.

4.3.2. Model eğitimi ve seçimi

Yapılan çalışmada toplanan verilerin %60'ı eğitim için kullanılmıştır. Tüm modelleri eğitmek için Google tarafından sunulan scikit – learn kütüphanesi kullanılmıştır [51]. Eğitim veri kümesi üzerinde K En Yakın Komşu, Destek Vektör Makinesi, Rastgele Orman ve son olarak Gradyan Arttırıcı Sınıflandırma Algoritması ile modeller eğitilmiştir. Eğitilen modeller arasında en iyi sonucu Gradyan Arttırıcı Sınıflandırma algoritmasının verdiği gözlenmiştir. Gradyan Arttırıcı Sınıflandırma Algoritması, rekreasyon ve sınıflandırma problemleri için kullanılan bir makine öğrenme algoritmasıdır. Bu algoritmada tahminler sırayla yapılmaktadır. Önceden tahmin edilen tahminlerin hatalarından öğrenerek daha güçlü tahminler üretmektedir. Bir başka deyişle, zayıf tahmin modellerinin bir araya gelmesiyle karar ağaçlarının oluşturduğu bir model oluşturup daha güçlü tahminler üretmektedir [52].

Sınıflandırma algoritmalarının ön işleme fazında analiz edilmek üzere eğitilen uygulamalarda bulunmayan izinler 'false' olarak işaretlenmiştir. Kullanılan sınıflandırma algoritmalarından K En Yakın Komşu Algoritması'nın örnek kod parçası Şekil 4.5'te sunulmuştur.

```
In [ ]: from sqlalchemy import create_engine
        import numpy as np
        import pandas as pd
        from sklearn.neighbors import KNeighborsClassifier
        from sklearn import preprocessing
        from sklearn.model_selection import train_test_split
        from sklearn.metrics import accuracy_score
        from sklearn import metrics
        import matplotlib.pyplot as plt
        from sklearn.metrics import roc_curve
        import pickle
        from sklearn import svm

In [ ]: engine = create_engine('postgresql://postgres:qwe123**@localhost:5432/postgres')

        query ="with temp as (SELECT * FROM crosstab($$ select al.packagename::TEXT as packagename, lower(repla
        raw_data  = pd.read_sql_query(query,con=engine)

In [ ]:

In [ ]: my_data=raw_data.replace(to_replace=np.nan,value='false')

        le = preprocessing.LabelEncoder()

        for i in range(1,my_data.shape[1]):
            my_data.iloc[:,i] = le.fit_transform(my_data.iloc[:,i])


        target = my_data.iloc[:,:1]

        data =  my_data.iloc[:,2:]

In [ ]: data

        data_train, data_test, target_train, target_test = train_test_split(data,target, test_size=0.40, random

        clf = KNeighborsClassifier(n_neighbors=5)

        clf.fit(data_train,target_train.values.ravel())

        predictions = clf.predict(data_test)

In [ ]: predictions

In [ ]: print("Accuracy:")
        print (accuracy_score(target_test, predictions))

        print("Confusion Matrix")
        print(metrics.confusion_matrix(target_test, predictions))


        print("Classification Report")
        print(metrics.classification_report(target_test, predictions))

        fpr, tpr, thresholds = roc_curve(target_test, predictions, pos_label=1)

        print("ROC Curve")
        plt.plot(fpr,tpr)
        plt.plot([0, 1], [0, 1],'r--')
        plt.xlim([0.0, 1.0])
        plt.ylim([0.0, 1.05])
        plt.xlabel('1 - Özgüllük(False Positive Rate)')
        plt.ylabel('Hassasiyet(True Positive Rate)')
        plt.title('Alıcı İşlem Karakteristiği Eğrisi')
        plt.show()

        auc = np.trapz(tpr,fpr)
        print("Eğrinin alanı:",auc)



        pickle.dump(clf,open("apk_classifier.pkl","wb"))

In [ ]: my_data
```

Şekil 4.5. K En Yakın Komşu Algoritması kod parçası

Kullanılan sınıflandırma algoritmalarından Destek Vektör Makinesi Algoritması'nın örnek kod parçası Şekil 4.6'da sunulmuştur.

```
In [*]: from sqlalchemy import create_engine
        import numpy as np
        import pandas as pd
        from sklearn.ensemble import GradientBoostingClassifier
        from sklearn import preprocessing
        from sklearn.model_selection import train_test_split
        from sklearn.metrics import accuracy_score
        from sklearn import metrics
        import matplotlib.pyplot as plt
        from sklearn.metrics import roc_curve
        import pickle
        from sklearn import svm

        C:\Users\Abdullah\Anaconda3\lib\site-packages\sklearn\ensemble\weight_boosting.py:29: DeprecationWarning: numpy.core.umath_test
        s is an internal NumPy module and should not be imported. It will be removed in a future NumPy release.
          from numpy.core.umath_tests import inner1d

In [*]: engine = create_engine('postgresql://postgres:qwe123**@localhost:5432/postgres')

        query ="with temp as (SELECT * FROM crosstab($$ select al.packagename::TEXT as packagename, lower(replace(pl.constant_value, '.',
        raw_data  = pd.read_sql_query(query,con=engine)

In [ ]:

In [*]: my_data=raw_data.replace(to_replace=np.nan,value='false')

        le = preprocessing.LabelEncoder()

        for i in range(1,my_data.shape[1]):
            my_data.iloc[:,i] = le.fit_transform(my_data.iloc[:,i])


        target = my_data.iloc[:,:1]

        data =  my_data.iloc[:,2:]

In [ ]:

In [*]: data_train, data_test, target_train, target_test = train_test_split(data,target, test_size=0.40, random_state=42)

        clf = svm.SVC(C=7, kernel='sigmoid')

        clf.fit(data_train,target_train.values.ravel())

        predictions = clf.predict(data_test)

In [ ]:

In [*]: print("Accuracy:")
        print (accuracy_score(target_test, predictions))

        print("Confusion Matrix")
        print(metrics.confusion_matrix(target_test, predictions))


        print("Classification Report")
        print(metrics.classification_report(target_test, predictions))

        fpr, tpr, thresholds = roc_curve(target_test, predictions, pos_label=1)

        print("ROC Curve")
        plt.plot(fpr,tpr)
        plt.plot([0, 1], [0, 1],'r--')
        plt.xlim([0.0, 1.0])
        plt.ylim([0.0, 1.0])
        plt.xlabel('1 - Özgüllük(False Positive Rate)')
        plt.ylabel('Hassasiyet(True Positive Rate)')
        plt.title('Alıcı İşlem Karakteristiği Eğrisi')
        plt.show()

        auc = np.trapz(tpr,fpr)
        print("Eğrinin alanı:",auc)


        pickle.dump(clf,open("apk_classifier.pkl","wb"))
```

Şekil 4.6. Destek Vektör Makinesi Algoritması kod parçası

Kullanılan sınıflandırma algoritmalarından Rastgele Orman Algoritması'nın örnek kod parçası Şekil 4.7'de sunulmuştur.

In [1]:
```
from sqlalchemy import create_engine
import numpy as np
import pandas as pd
from sklearn.ensemble import RandomForestClassifier
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn import metrics
import matplotlib.pyplot as plt
from sklearn.metrics import roc_curve
import pickle
from sklearn import svm
```

```
C:\Users\Abdullah\Anaconda3\lib\site-packages\sklearn\ensemble\weight_boosting.py:29: DeprecationWarning: numpy.core.umath_test
s is an internal NumPy module and should not be imported. It will be removed in a future NumPy release.
  from numpy.core.umath_tests import inner1d
```

In [2]:
```
engine = create_engine('postgresql://postgres:qwe123**@localhost:5432/postgres')

query ="with temp as (SELECT * FROM crosstab($$ select al.packagename::TEXT as packagename, lower(replace(pl.constant_value, '.',
raw_data  = pd.read_sql_query(query,con=engine)
```

In [ ]:
```
```

In [*]:
```
my_data=raw_data.replace(to_replace=np.nan,value='false')

le = preprocessing.LabelEncoder()

for i in range(1,my_data.shape[1]):
    my_data.iloc[:,i] = le.fit_transform(my_data.iloc[:,i])


target = my_data.iloc[:,:1]

data =  my_data.iloc[:,2:]
```

In [ ]:
```
```

In [*]:
```
data_train, data_test, target_train, target_test = train_test_split(data,target, test_size=0.40, random_state=42)

clf = RandomForestClassifier(n_estimators=1000)

clf.fit(data_train,target_train.values.ravel())

predictions = clf.predict(data_test)
```

In [ ]:
```
```

In [*]:
```
print("Accuracy:")
print (accuracy_score(target_test, predictions))

print("Confusion Matrix")
print(metrics.confusion_matrix(target_test, predictions))


print("Classification Report")
print(metrics.classification_report(target_test, predictions))

fpr, tpr, thresholds = roc_curve(target_test, predictions, pos_label=1)

print("ROC Curve")
plt.plot(fpr,tpr)
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('1 - Özgüllük(False Positive Rate)')
plt.ylabel('Hassasiyet(True Positive Rate)')
plt.title('Alıcı İşlem Karakteristiği Eğrisi')
plt.show()

auc = np.trapz(tpr,fpr)
print("Eğrinin alanı:",auc)


pickle.dump(clf,open("apk_classifier.pkl","wb"))
```

Şekil 4.7. Rastgele Orman Algoritması kod parçası

Kullanılan sınıflandırma algoritmalarından Gradyan Arttırıcı Sınıflandırma Algoritması'nın örnek kod parçası Şekil 4.8'de sunulmuştur.

```
In [1]: from sqlalchemy import create_engine
        import numpy as np
        import pandas as pd
        from sklearn.ensemble import GradientBoostingClassifier
        from sklearn import preprocessing
        from sklearn.model_selection import train_test_split
        from sklearn.metrics import accuracy_score
        from sklearn import metrics
        import matplotlib.pyplot as plt
        from sklearn.metrics import roc_curve
        import pickle

        C:\Users\Abdullah\Anaconda3\lib\site-packages\sklearn\ensemble\weight_boosting.py:29: DeprecationWarning: numpy.core.umath_test
        s is an internal NumPy module and should not be imported. It will be removed in a future NumPy release.
          from numpy.core.umath_tests import inner1d

In [2]: engine = create_engine('postgresql://postgres:qwe123**@localhost:5432/postgres')

        query ="with temp as (SELECT * FROM crosstab($$ select al.packagename::TEXT as packagename, lower(replace(pl.constant_value, '.',
        raw_data  = pd.read_sql_query(query,con=engine)

In [ ]:

In [*]: my_data=raw_data.replace(to_replace=np.nan,value='false')

        le = preprocessing.LabelEncoder()

        for i in range(1,my_data.shape[1]):
            my_data.iloc[:,i] = le.fit_transform(my_data.iloc[:,i])


        target = my_data.iloc[:,:1]

        data =  my_data.iloc[:,2:]

In [ ]:

In [*]: data_train, data_test, target_train, target_test = train_test_split(data,target, test_size=0.40, random_state=42)

        clf = GradientBoostingClassifier(n_estimators=1000, learning_rate=0.01, max_depth=10, random_state=0)

        clf

        clf.fit(data_train,target_train.values.ravel())

        predictions = clf.predict(data_test)

In [ ]:

In [*]: predictionsperc = clf.predict_proba(data_test)

In [ ]:

In [*]: print("Accuracy:")
        print (accuracy_score(target_test, predictions))

        print("Confusion Matrix")
        print(metrics.confusion_matrix(target_test, predictions))


        print("Classification Report")
        print(metrics.classification_report(target_test, predictions))

        fpr, tpr, thresholds = roc_curve(target_test, predictions, pos_label=1)

        print("FPR: ", fpr)
        print("TPR: ", tpr)

        print("ROC Curve")
        plt.plot(fpr,tpr)
        plt.plot([0, 1], [0, 1],'r--')
        plt.xlim([0.0, 1.0])
        plt.ylim([0.0, 1.05])
        plt.xlabel('1 - Özgüllük(False Positive Rate)')
        plt.ylabel('Hassasiyet(True Positive Rate)')
        plt.title('Alıcı İşlem Karakteristiği Eğrisi')
        plt.show()

        auc = np.trapz(tpr,fpr)
        print("Eğrinin alanı:",auc)


        pickle.dump(clf,open("apk_classifier.pkl","wb"))
```

Şekil 4.8. Gradyan Arttırıcı Sınıflandırma Algoritması kod parçası

4.3.3. Model değerlendirmesi

Eğitim veri seti üzerinde eğitilen model, test verisi üzerinden değerlendirmeye alınmaktadır. Test edilmek üzere ayrılan %40'lık veri seti için model tahminleri üretilip, önceden bilinen gerçek sonuçlar ile karşılaştırılarak modelin başarısı ölçümlenmektedir.

Doğruluk (Accuracy)

Doğruluk, sınıflandırma modellerini değerlendirmek için bir metriktir, modelin yapmış olduğu tahminlerin % kaçının doğru olduğunu gösterir. Gayri resmi olarak doğruluk, modelimizin haklı olduğu tahminlerin bir parçasıdır. Formal olarak doğruluk değerinin formülize edilmiş hali Eş. 4.1'de gösterilmiştir.

$$Accuracy = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}} \tag{4.1}$$

İkili sınıflandırma için doğruluk, Eş. 4.2'de formülize edildiği gibi pozitif ve negatif olarak da hesaplanabilir:

$$Accuracy = \frac{TP+TN}{\text{TP+TN+FP+FN}} \tag{4.2}$$

Formüldeki TP = Gerçek Pozitif, TN = Gerçek Negatif, FP = Yanlış Pozitif ve FN = Yanlış Negatif'liği ifade eder [53].

Yapılan çalışmanın farklı makine öğrenmesi algoritmalarındaki doğruluk skorları Çizelge 4.2'de gösterilmiştir.

Çizelge 4.2. Doğruluk skorları

	k-NN	SVM	Random Forest	Gradyan Arttırıcı
Doğruluk	0.8962	0.8913	0.9175	0.9176

Karışıklık matrisi (Confusion matrix)

Bir karışıklık matrisi, gerçek değerlerin bilindiği bir test verisi kümesinde bir sınıflandırma modelinin veya sınıflandırıcının performansını tanımlamak için sıklıkla kullanılan bir tablodur [54]. Kullanılan veri seti etiketine göre kullanılan karışıklık matrisi örneği Çizelge 4.3'te gösterilmiştir.

Çizelge 4.3. Karışıklık matrisi [54]

True Positive: Gerçek: Kötücül ML model tahmini: Kötücül	False Positive: Gerçek: İyicil ML model tahmini: Kötücül
False Negative: Gerçek: Kötücül ML model tahmini: İyicil	True Negative: Gerçek: İyicil ML model tahmini: İyicil

K En Yakın Komşu Algoritması ile eğitilen modelin karışıklık matrisi çıktısı Çizelge 4.4'te gösterilmiştir.

Çizelge 4.4. K En Yakın Komşu Algoritması ile eğitilen modelin karışıklık matrisi çıktısı

		Modelin Tahminleri: x_m	
Karışıklık Matrisi		False	True
Gerçek Değerler: x	False	2212	868
	True	852	12644

Destek Vektör Makinesi Algoritması ile eğitilen modelin karışıklık matrisi çıktısı Çizelge 4.5'te gösterilmiştir.

Çizelge 4.5. Destek Vektör Makinesi Algoritması ile eğitilen modelin karışıklık matrisi çıktısı

		Modelin Tahminleri: x_m	
Karışıklık Matrisi		False	True
Gerçek Değerler: x	False	2618	462
	True	1339	12157

Rastgele Orman Algoritması ile eğitilen modelin karışıklık matrisi çıktısı Çizelge 4.6'da gösterilmiştir.

Çizelge 4.6. Rastgele Orman Algoritması ile eğitilen modelin karışıklık matrisi çıktısı

Karışıklık Matrisi		Modelin Tahminleri: x_m	
		False	True
Gerçek Değerler: x	False	2873	207
	True	1161	12335

Gradyan Arttırıcı Sınıflandırma Algoritması ile eğitilen modelin karışıklık matrisi çıktısı Çizelge 4.7'de gösterilmiştir.

Çizelge 4.7. Gradyan Arttırıcı Sınıflandırma Algoritması ile eğitilen modelin karışıklık matrisi çıktısı

Karışıklık Matrisi		Modelin Tahminleri: x_m	
		False	True
Gerçek Değerler: x	False	2887	193
	True	1173	12323

Sınıflandırma Raporu (Classification Report)

Duyarlılık (Recall): Gerçek pozitif değeri doğru tahmin etme oranıdır. Formülize edilmiş hali Eş. 4.3'te gösterilmiştir.

$$Recall = \frac{TP}{\mathrm{TP+FN}} \tag{4.3}$$

Kesinlik (Precision): Doğru tahmin edilen pozitif gözlemlerin toplam tahmini pozitif gözlemlere oranıdır. Formülize edilmiş hali Eş. 4.4'te gösterilmiştir.

$$Precision = \frac{TP}{\mathrm{TP+FP}} \tag{4.4}$$

F1-Score = Duyarlılık ve kesinliğin ağırlıklı ortalamasıdır. Formülize edilmiş hali Eş. 4.5'te gösterilmiştir.

$$F1 - Score = \frac{2*Recall*Precision}{\mathrm{Recall+Precision}} \tag{4.5}$$

K En Yakın Komşu Algoritması ile eğitilen modelin sınıflandırma raporu çıktısı Çizelge 4.8'de gösterilmiştir.

Çizelge 4.8. K En Yakın Komşu Algoritması ile eğitilen modelin sınıflandırma raporu çıktısı

	Precision	Recall	F1–Score	Support
False	0.72	0.72	0.72	3080
True	0.94	0.94	0.94	13496
avg / total	0.90	0.90	0.90	16576

Destek Vektör Makinesi Algoritması ile eğitilen modelin sınıflandırma raporu çıktısı Çizelge 4.9'da gösterilmiştir.

Çizelge 4.9. Destek Vektör Makinesi Algoritması ile eğitilen modelin sınıflandırma raporu çıktısı

	Precision	Recall	F1–Score	Support
False	0.66	0.85	0.74	3080
True	0.96	0.90	0.93	13496
avg / total	0.91	0.89	0.90	16576

Rastgele Orman Algoritması ile eğitilen modelin sınıflandırma raporu çıktısı Çizelge 4.10'da gösterilmiştir.

Çizelge 4.10. Rastgele Orman Algoritması ile eğitilen modelin sınıflandırma raporu çıktısı

	Precision	Recall	F1–Score	Support
False	0.71	0.93	0.81	3080
True	0.98	0.91	0.95	13496
avg / total	0.93	0.92	0.92	16576

Gradyan Arttırıcı Sınıflandırma Algoritması ile eğitilen modelin sınıflandırma raporu çıktısı Çizelge 4.11'de gösterilmiştir.

Çizelge 4.11. Gradyan Arttırıcı Sınıflandırma Algoritması ile eğitilen modelin sınıflandırma raporu çıktısı

	Precision	Recall	F1–Score	Support
False	0.71	0.94	0.81	3080
True	0.98	0.91	0.95	13496
avg / total	0.93	0.92	0.92	16576

Alıcı İşlem Karakteristik Eğrisi (ROC Curve)

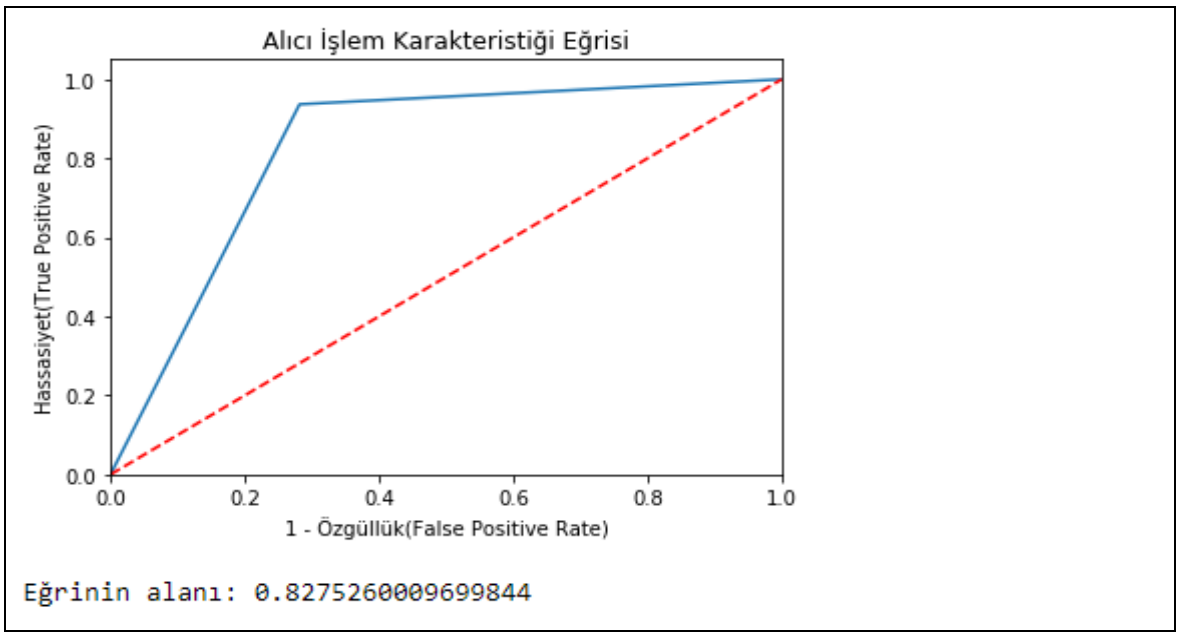
Sınıflandırma modelinin başarı performansını gösteren grafiktir [55]. True positive rate ve false positive rate olmak üzere iki metriğe odaklanmaktadır.
True Positive Rate duyarlılık ile eş göreve sahiptir. Bu eşitliğin formülize edilmiş hali Eş. 4.6'da gösterilmiştir.

$$TPR = \frac{TP}{\text{TP+FN}} \tag{4.6}$$

False Positive Rate aşağıdaki gibi tanımlanmaktadır. Bu eşitliğin formülize edilmiş hali Eş. 4.7'de gösterilmiştir.

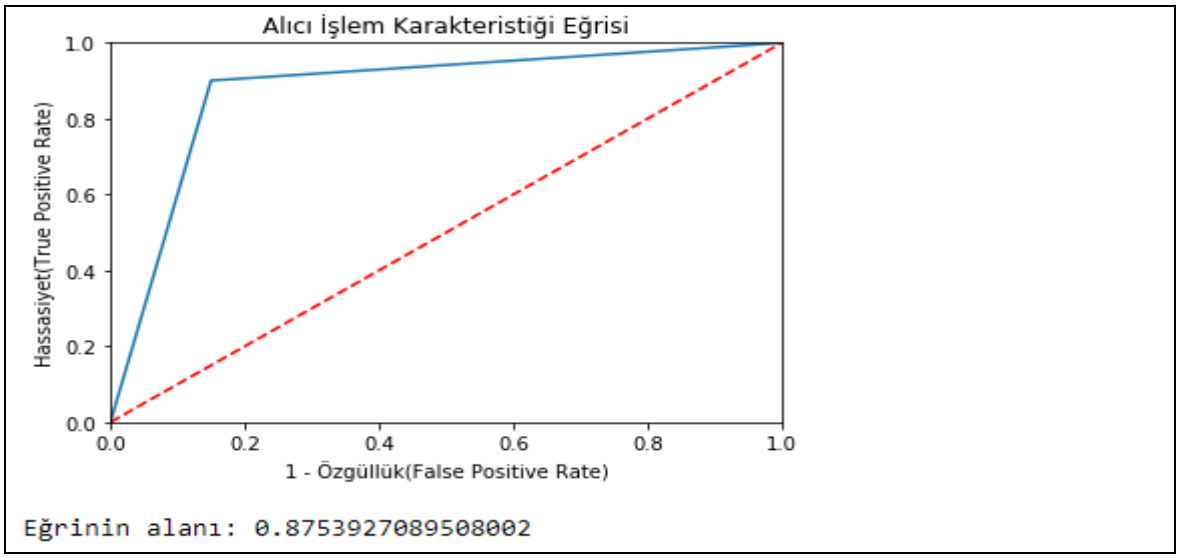
$$FPR = \frac{FP}{\text{FP+TN}} \tag{4.7}$$

K En Yakın Komşu Algoritması ile eğitilen modelin başarı performansını gösteren alıcı işlem karakteristiği eğrisi Şekil 4.9'da gösterilmiştir.



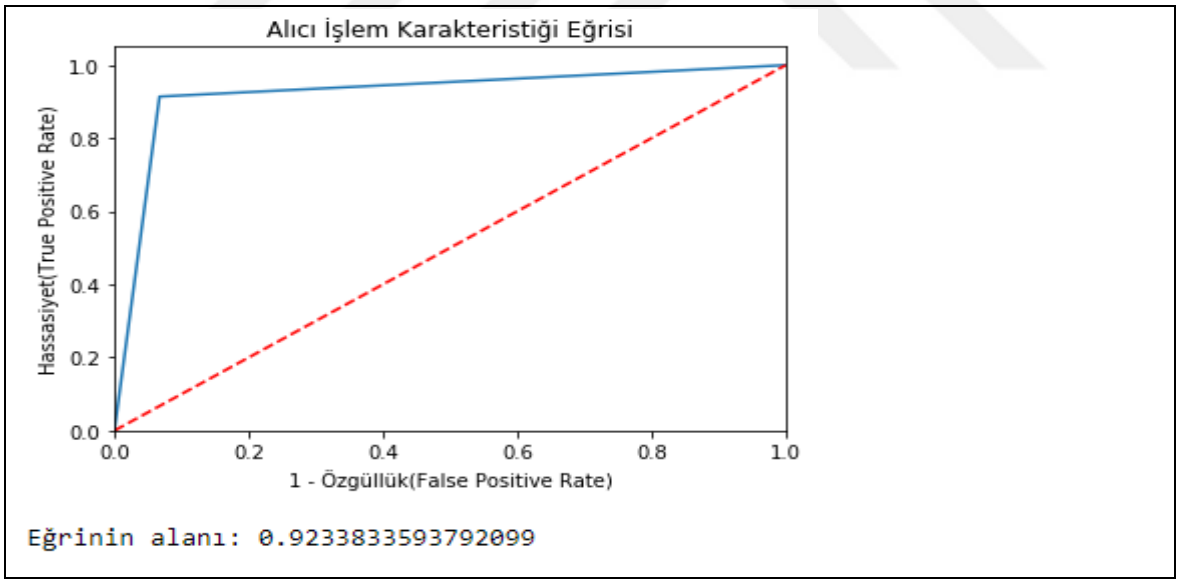
Şekil 4.9. K En Yakın Komşu Algoritması ile eğitilen modelin alıcı işlem karakteristikleri eğrisi

Destek Vektör Makinesi Algoritması ile eğitilen modelin başarı performansını gösteren alıcı işlem karakteristiği eğrisi Şekil 4.10'da gösterilmiştir.



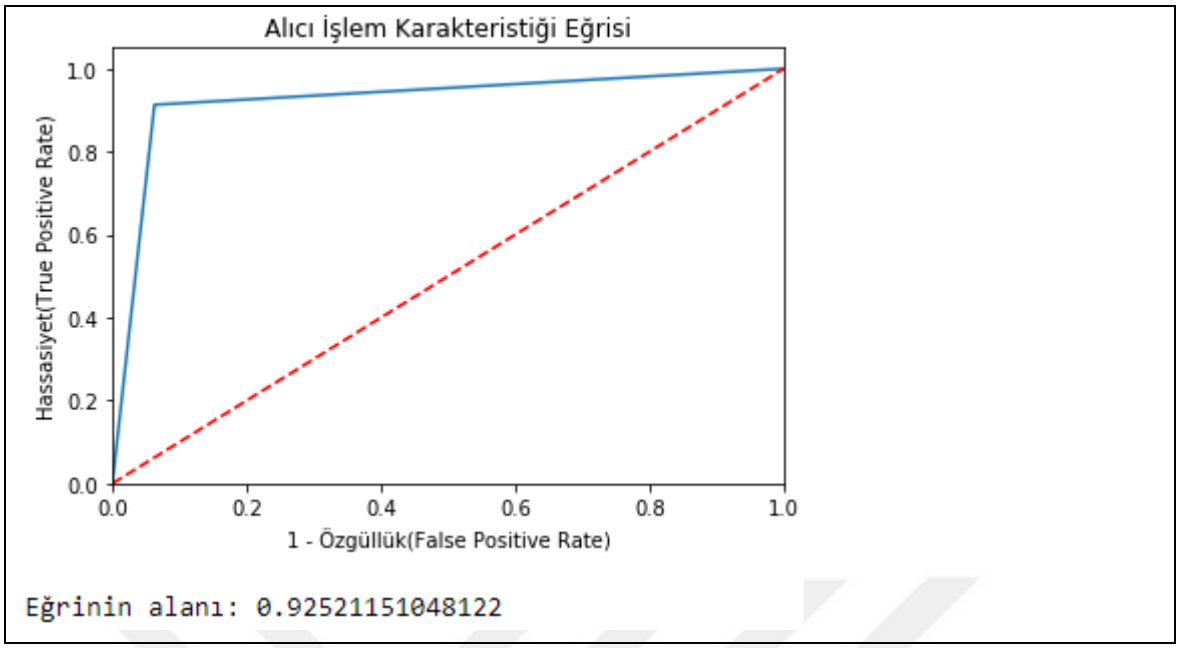
Şekil 4.10. Destek Vektör Makinesi Algoritması ile eğitilen modelin alıcı işlem karakteristikleri eğrisi

Rastgele Orman Algoritması ile eğitilen modelin başarı performansını gösteren alıcı işlem karakteristiği eğrisi Şekil 4.11'de gösterilmiştir.



Şekil 4.21. Rastgele Orman Algoritması ile eğitilen modelin alıcı işlem karakteristikleri eğrisi

Gradyan Arttırıcı Sınıflandırma Algoritması ile eğitilen modelin başarı performansını gösteren alıcı işlem karakteristiği eğrisi Şekil 4.12'de gösterilmiştir.



Şekil 4.32. Gradyan Arttırıcı Sınıflandırma Algoritması ile eğitilen modelin alıcı işlem karakteristikleri eğrisi

Bir alıcı işlem karakteristiği eğrisi incelendiğinde eğrinin altında kalan alan 1 ise mükemmel bir testi temsil etmektedir. Eğrinin altında kalan 0,5'lik bir alanı kapsıyorsa yapılan çalışmanın etkisiz bir çalışma olduğunu temsil edilmektedir. Çizelge 4.12 incelendiğinde çalışmadaki en yüksek sınıflandırma modelinin başarı performansının 0,9252 olduğu gözlenmektedir.

Çizelge 4.12. Kullanılan makine öğrenme algoritmalarının alıcı işlem karakteristiği eğrisi lanlarının karşılaştırılması

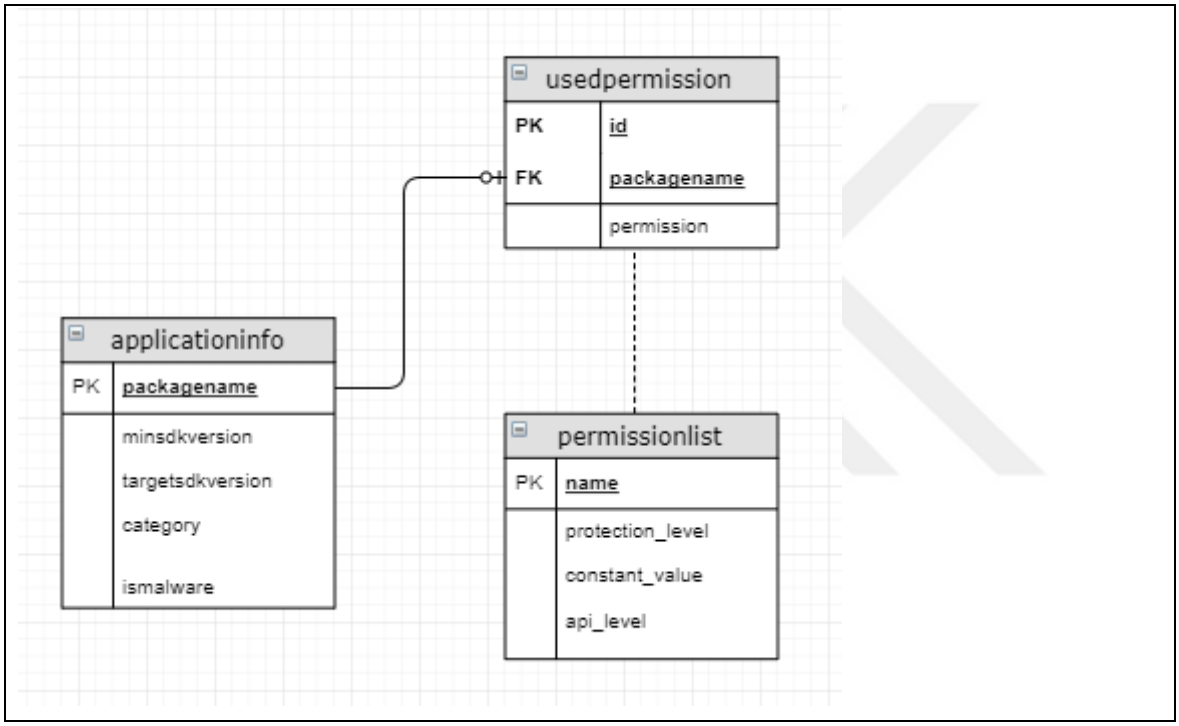
	k-NN	SVM	Random Forest	Gradyan Arttırıcı
Alıcı İşlem Karakteristiği Eğrisi Alanı	0.8275	0.8753	0.9233	0.9252

Model Dağıtımı (Model Deployment)

Eğitilen modelin Pickle kütüphanesi kullanılarak kaydedilip ardından Flask kütüphanesi kullanılarak HTTP istekler ile tahmin üretmesi sağlanmıştır. Pickle kütüphanesi herhangi bir Python nesnesini byte stringine dönüştürerek nesnenin dosyaya kaydedilmesini, arşivlenmesini veya başka bir sunucuya iletilmesini sağlayan Python modülüdür [56]. Flask kütüphanesi ise Python dili ile web uygulamaları yapmaya yarayan mini frameworktür [57].

4.4. Veritabanı Sunucusu

Merkez sunucu tarafından analiz edilen uygulamaların karakteristiğinin çıkarıldıktan sonra kaydının tutulduğu sunucudur. Tersine mühendislik yöntemi ile elde edilen verilerin veritabanına kaydedilen özellikleri, uygulamanın desteklediği minSdk ve targetSdk versiyonları ile birlikte uygulamaların kullandığı izinlerdir. Uygulamaların analizi sonucu elde edilen veriler ilişkisel olarak veritabanında tutulmaktadır. Veritabanının ER diyagramı Şekil 4.13'te gösterilmektedir.



Şekil 4.13. Geliştirilen sistemin veritabanı E-R diyagramı

Şekil 4.13'te de görüldüğü gibi veritabanında 3 tane tablo bulunmaktadır. Bu tablolar ile ilgili bilgiler aşağıda belirtilmektedir:

- applicationinfo: Uygulamanın paket adı, kategorisi, türü, minSdk ve targetSdk versiyonlarının tutulduğu tablodur
- permissionlist: Google tarafından sunulan Android işletim sisteminin sahip olduğu tüm izinlerin güvenlik seviyesi ve api seviyesine göre saklandığı tablodur
- usedpermission: Her bir uygulamanın kullandığı izinlerin paket adına göre tutulduğu tablodur.

5. TARTIŞMA

Bu çalışmada, sistemi eğitmek için kötücül veri seti olarak 2722 uygulama iyicil veri seti olarak ise 3406 uygulama analiz edilmiştir. Kullanılan kötücül veri seti literatürde sıklıkla kullanılan Drebin veri setinden temin edilmiştir. Analiz sonucunda elde edilen kötücül uygulamaların en çok kullandığı izinler ve bu izinlerin güvenlik seviyeleri Çizelge 5.1'de gösterilmiştir.

Çizelge 5.1. Kötücül yazılımların en çok kullandığı 10 izin ve bu izinlerin güvenlik durumları

İzin	Toplam	Güvenlik Seviyesi
android.permission.INTERNET	2632	Normal
android.permission.READ_PHONE_STATE	2416	Tehlikeli
android.permission.WRITE_EXTERNAL_STORAGE	1829	Tehlikeli
android.permission.ACCESS_COARSE_LOCATION	1163	Tehlikeli
android.permission.ACCESS_FINE_LOCATION	1054	Tehlikeli
android.permission.SEND_SMS	947	Tehlikeli
android.permission.ACCESS_NETWORK_STATE	2105	Normal
android.permission.RECEIVE_BOOT_COMPLETED	1458	Normal
android.permission.ACCESS_WIFI_STATE	1447	Normal
android.permission.WAKE_LOCK	1178	Normal

Analiz sonucunda elde edilen iyicil uygulamaların en çok kullandığı izinler ve bu izinlerin güvenlik seviyeleri ise Çizelge 5.2'de gösterilmiştir.

Çizelge 5.2. İyicil yazılımların en çok kullandığı 10 izin ve bu izinlerin güvenlik durumları

İzin	Toplam	Güvenlik Seviyesi
android.permission.INTERNET	3304	Normal
android.permission.ACCESS_NETWORK_STATE	3201	Tehlikeli
android.permission.WAKE_LOCK	2077	Normal
android.permission.ACCESS_WIFI_STATE	1546	Normal
android.permission.VIBRATE	1402	Normal
android.permission.READ_EXTERNAL_STORAGE	1116	Tehlikeli
android.permission.READ_PHONE_STATE	1238	Tehlikeli
android.permission.RECEIVE_BOOT_COMPLETED	961	Normal
android.permission.ACCESS_FINE_LOCATION	1018	Tehlikeli
android.permission.ACCESS_COARSE_LOCATION	968	Tehlikeli

Test sonuçları incelendiğinde en başarılı sonuçların %91,76 doğruluk, %91,31 hassasiyet ve %93,73 özgüllük oranlarıyla Gradyan Arttırıcı Sınıflandırma algoritmasıyla elde edildiği gözlenmiştir. Aynı test sonuçlarında Gradyan Arttırıcı Sınıflandırma algoritmasıyla eğitilen modelin %93 başarı oranıyla en yüksek başarı oranına sahip model olduğu görülmüştür. Geliştirilen sistemin kullanılan farklı makine öğrenmesi algoritmalarında elde edilen sonuçlarının karşılaştırılması Çizelge 5.3'te gösterilmektedir.

Çizelge 5.3. Elde edilen sonuçların karşılaştırılması

	k-NN	SVM	Random Forest	Gradyan Arttırıcı
Doğruluk	0.8962	0.8913	0.9175	0.9176
Alıcı İşlem Karakteristiği Eğrisi Alanı	0.8275	0.8753	0.9233	0.9252
Duyarlılık	0.9369	0.9008	0.9140	0.9131
Kesinlik	0.9358	0.9634	0.9835	0.9846
F1-Score	0.9364	0.9310	0.9474	0.9475
FPR	0.2818	0.15	0.0672	0.0627
Özgüllük	0.7182	0.85	0.9328	0.9373

Çizelge 5.3'te de görüldüğü gibi farklı makine öğrenmesi algoritmaları kullanıldığında farklı sonuçlar elde edilmektedir. Sonuçlar incelendiğinde en başarılı sonuçların %91,76 doğruluk, %91,31 hassasiyet ve %93,73 özgüllük oranlarıyla Gradyan Arttırıcı Sınıflandırma algoritmasıyla elde edildiği gözlenmiştir.

6. SONUÇ VE ÖNERİLER

Yapılan birçok araştırma, kullanıcıların izinler hakkında yeterli bilgiye sahip olmadığını ve bu nedenle mobil cihazlarını tehlikeye attıklarını göstermiştir. Kullanıcıların büyük birçoğunun bilinçsiz kullanıcı olması ve Android işletim sisteminin yeterli güvenlik mekanizmasına sahip olmaması Android işletim sistemini kötücül yazılım geliştiricilerin hedefi haline getirmektedir. Android işletim sisteminde kötücül yazılım tespiti için araştırmacılar yoğun çaba harcamaktadır. Yapılan araştırmaların birçoğu incelendiğinde genellikle araştırmacıların akademik çalışma amacıyla kötücül yazılım tespit sistemi geliştirdikleri görülmüştür. Bu tez çalışması ile kullanıcıların Android işletim sistemli mobil cihazlarında kullandıkları uygulamaların ne kadar güvenilir olduğunun belirlenmesi ve kullanıcıları kullandıkları uygulamalar hakkında bilinçlendirmek için bu uygulamaların kullandıkları izinler hakkında kullanıcıyı bilgilendirmek amaçlanmaktadır. Geliştirilen sunucu ve istemci tabanlı kötücül yazılım tespit etmeyi sağlayan sistem, rekresyon ve sınıflandırma problemleri için kullanılan makine öğrenmesi algoritmalarından biri olan Gradyan Arttırıcı Sınıflandırma algoritmasıyla birlikte Rastgele Orman algoritması, Destek Vektör Makinesi ve K En Yakın Komşu algoritmalarını kullanan öğrenmeye dayalı kullanıcı dostu bir platformdur. Bu sistem kötücül veya iyicil Android uygulamaların kullandıkları izinleri inceleyerek uygulamanın güvenilirliği hakkında kullanıcıyı bilgilendirmektedir. Uygulama modelleri için kötücül uygulama olarak Drebin veri setinden 2722 uygulama kullanılmıştır. İyicil veri seti olarak 2018 yılında A. H. Lashkari ve arkadaşları[47] tarafından oluşturulup kullanıma sunulan CICAndMal2017 isimli veri setinden 3406 uygulamadan yararlanılmıştır. Geliştirilen sistemin yapılan ilk testleri incelendiğinde en başarılı sonuçların %91,76 doğruluk, %91,31 hassasiyet ve %93,73 özgüllük oranlarıyla Gradyan Arttırıcı Sınıflandırma algoritmasıyla elde edildiği gözlenmiştir. Ayrıca Gradyan Arttırıcı Sınıflandırma algoritmasıyla eğitilen modelin %93 başarı oranıyla en yüksek başarı oranına sahip model olduğu görülmüştür.

Geliştirilen sistem kullanıcılara kullandıkları uygulamaların izinlerini göstererek bu izinlerin kötücül veya iyicil olma durumunu sunmaktadır. Test sonuçları incelendiğinde geliştirilen sistemin kötücül yazılım tespitinde başarılı sonuçlar elde ettiği gözlenmektedir. Başarı oranının daha da arttırılması için Android uygulamaların kullandığı izinlere ek olarak sahip olduğu diğer özelliklerinde incelenmesinin önemli olduğu düşünülmektedir. Gelecekte geliştirilecek sistemlerin, derin öğrenme algoritmalarıyla birlikte dinamik analiz yöntemini

kullanmalarının uygulamaların güvenlik durumları hakkında daha yüksek başarı oranlarına sahip sonuçlar verebileceği düşünülmektedir. Böylelikle bir uygulamanın hangi amaçla yazıldığının daha kesin olarak anlaşılabilir olacağı düşünülmektedir.

KAYNAKLAR

1. İnternet: Global market share held by the leading smartphone operating systems in sales to end users from 1st quarter 2009 to 2nd quarter 2018. Statista. URL: https://www.statista.com/statistics/266136/global-market-share-held-by-smartphone-operating-systems/, Son Erişim Tarihi: 06.04.2019.

2. İnternet: Worldwide Mobile Phone Forecast Update. *IDC*. URL: https://www.idc.com/getdoc.jsp?containerId=US44269718, Son Erişim Tarihi: 06.04.2019.

3. İnternet: Statista: Number of available applications in the Google Play Store from December 2009 to December 2018. *Statista*. URL: https://www.statista.com/statistics/266210/number-of-available-applications-in-the-google-play-store/, Son Erişim Tarihi: 06.04.2019.

4. İnternet: Another Reason 99% of Mobile Malware Targets Androids. *F-Secure.* URL: https://blog.f-secure.com/another-reason-99-percent-of-mobile-malware-targets-androids/, Son Erişim Tarihi: 06.04.2019.

5. İnternet: New malware every 10 seconds! *G Data.* URL: https://www.gdatasoftware.com/blog/2018/05/30735-new-malware-every-10-seconds, Son Erişim Tarihi: 06.04.2019.

6. Felt, A.P., Finifter, M., Chin, E., Hanna, S., Wagner, D. (2011). A survey of mobile malware in the wild. *SPSM '11 Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices.* pp. 3–14., Chicago, IL, USA

7. İnternet: Android System Architecture. URL: https://developer.android.com/guide/platform, Son Erişim Tarihi: 17.03.2020.

8. 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.

9. Arp, D., Spreitzenbarth, M., Malte, H., Gascon, H. ve Rieck, K. (2014). Drebin: Effective and explainable detection of Android malware in your pocket. *Symposium on Network and Distributed System Security (NDSS)*, 23–26.

10. İnternet: Apktool: A tool for reverse engineering Android apk file. URL: https://ibotpeaches.github.io/Apktool/, Son Erişim Tarihi: 06.04.2019.

11. İnternet: ApkAnalyser is a static, virtual analysis tool for examining and validating the development work of your Android app *ApkAnalyser* URL: https://github.com/sonyxperiadev/ApkAnalyser, Son Erişim Tarihi: 06.04.2019.

12. İnternet: Androguard [Abstract] *Dattani.* Web: http://www.technotalkative.com/part-1-reverse-engineering-using-androguard/, Son Erişim Tarihi: 06.04.2019.

13. Moonsamy, V., Rong, J. and Liu, S. (2014). Mining permission patterns for contrasting clean and malicious android applications. *Future Generation Computer Systems*, 36, 122–132.

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

15. Do, Q., Martini B. and Choo, K.-K. R. (2015). Exfiltrating data from Android devices. *Compute & Security* 48, 74-91.

16. Sheen, S., Anitha, R. and Natarajan, V. (2015). Android based malware detection using a multifeature collaborative decision fusion approach. *Neurocomputing* 151, 905-912.

17. Wang, P. and Wang, Y.-S. (2015). Malware behavioural detection and vaccine development by using a support vector model classifier. *Journal of Computer and System Sciences 81*, 1012-1026.

18. Kabakuş, A. T., Doğru, İ. A., and Çetin, A. (2015). APK Auditor: Permission-based Android malware detection system. *Digital Investigation*, 13, 1-14.

19. Kang, H., Jang, J.-W., Mohaisen, A. and Kim, H. K. (2015). *Detecting and Classifying Android Malware Using Static Analysis along with Creator Information.* International Journal of Distributed Sensor Networks.

20. Anwar, S., Zain, J. M., Inayat, Z., Karim, A., Haq, R. U., and Jabir, A. N. (2016). *A Static Approach towards Mobile Botnet Detection.* Conference on Electronic Design (ICED), Phuked, Tayland.

21. Narayanan, A., Yang, L., Chen, L., and Jinliang, L. (2016). *Adaptive and Scalable Android Malware Detection through Online Learning.* Paper presented at 2016 International Joint Conference on Neural Networks (IJCNN), Vancouver, Kanada.

22. Feizollah, A., Anuar, N. B., Salleh, R., Suarez-Tangil, G., and Furnell, S. (2016). AndroDialysis: Analysis of Android Intent Effectiveness in Malware Detection. *Computers & Security*, 65, 121-134.

23. Sokolova, K., Perez, C., and Lemercier, M. (2017). Android Application Classification and Anomaly Detection with Graph-based Permission Patterns. *Decision Support Systems*, 93, 62-76.

24. Wu, S., Zhang, Y., Jin, B. and Cao, W. (2017). *Practical static analysis of detecting intent-based permission leakage in Android application.* Paper presented at 2017 IEEE 17th International Conference on Communication Technology (ICCT), Chengdu, China.

25. Park, J., Chun, H. and Jung, S. (2018). *API and permission-based classification system for Android malware analysis.* Paper presented at 2018 International Conference on Information Networking (ICOIN), Chiang Mai, Thailand.

26. Şahin, D. Ö., Kural, O. E., Akleylek, S. and Kılıç, E. (2018). *New results on permission based static analysis for Android malware.* Paper presented at 2018 6th International Symposium on Digital Forensic and Security (ISDFS), Antalya, Turkey.

27. Arslan, R. S., Doğru, İ. A. and Barışcı, N. (2019). Permission Based Malware Detection System For Android Using Machine Learning Techniques, *International Journal of Software Engineering and Knowledge Engineering*, Vol. 29(1), pp. 63-91.

28. Zhu, Juan, H., Hong, Y., Zhu, Xuan, Z., Shi, Lei, W., Chen, Xing and Cheng, Li. (2018). DroidDet: Effective and robust detection of android malware using static analysis along with rotation forest model, *Neurocomputing*, vol. 272, pp. 638-646.

29. Doğru, İ. A. and Kiraz, Ö. (2018). *Web-Based Android Malicious Software Detection and Classification System*. Applied Sciences, Vol. 8 (9), pp. 1-19.

30. Dimjasevic, M., Atzeni, S., Ugrina, I. and Rakamaric, Z. (2016). Evaluation of Android Malware Detection Based on System Calls. *Paper presented at International Workshop on Security and Privacy Analytics (IWSPA)*, New Orleans, Amerika.

31. Burguera I., Zurutuza, U., and Nadjm-Tehrani, S. (2011). Crowdroid: behavior-based malware detection system for Android. *Science* (80-.), pp. 15–25.

32. Zhou, Y. and Jiang, X. (2012). Dissecting Android Malware. Characterization and Evolution. *2012 IEEE Symposium on Security and Privacy*, 95–109.

33. Enck, W., Gilbert, P., Chun, B.-G., Cox, L. P., Jung J., McDaniel P. and Sheth, A. N. *TaintDroid: An Information-Flow Tracking System for Realtime Privacy Monitoring on Smartphones.*

34. İnternet: Hooker - Automated Dynamic Analysis of Android Applications. URL: http://www.kitploit.com/2014/06/hooker-automated-dynamic-analysis-of.html, Son Erişim Tarihi: 06.04.2019.

35. İnternet: DECAF (short for Dynamic Executable Code Analysis Framework) is a binary analysis platform based on QEMU. URL: https://github.com/sycurelab/DECAF, Son Erişim Tarihi: 06.04.2019.

36. İnternet: Android app dynamic reverse tool based on Xposed framework. URL: https://github.com/minihand/ZjDroid, Son Erişim Tarihi: 06.04.2019.

37. Lantz, P. (2011). *An Android application sandbox for dynamic analysis*. Master Thesis, Department of Electrical and Information Technology, Sweden.

38. Shabtai, A., Tenenboim-Chekina, L., Mimran, D., Rokach, L., Shapira, B. and Elovici, Y. (2014). Mobile malware detection through analysis of deviations in application network behavior. *Computers & Security*, 43, 1-18.

39. Jang, J-W., Yun, J., Mohaisen, A., Woo, J. and Kim, H. K. (2016). Detecting and classifying method based on similarity matching of Android malware behavior with profile. *SpringerPlus*, 5(273), 1-23.

40. Garg, S., Peddoju, S. K. and Sarje, A. K. (2016). Network-based detection of Android malicious apps. *International Journal of Information Security*, 1-16.

41. Chang, W-L., Sun, H-M. and Wu, W. (2016). An Android Behavior-Based Malware Detection Method using Machine Learning. *Paper presented at 2016 IEEE International Conference on Signal Processing, Communications and Computing (ICSPCC)*, Hong Kong, China.

42. Arshad, S., Khan, A., Shah, M. A. and Ahmed, M. (2016). Android Malware Detection & Protection: *A Survey. Advanced Computer Science and Applications*, 7, 463-475.

43. Wang, X., Yang, Y., Zeng, Y. and Tang, C. (2015). A Novel Hybrid Mobile Malware Detection System Integrating Anomaly Detection With Misuse Detection. *Proceedings of the 6th International Workshop on Mobile Cloud Computing and Services*, 15-22.

44. G. Dini, F. Martinelli, A. Saracino, and D. Sgandurra (2012). MADAM: A Multi-level Anomaly Detector for Android Malware *in Computer Network Security*, vol. 7531.

45. Tong F. and Yana, Z. (2016). A hybrid approach of mobile malware detection in Android. *Journal of Parallel and Distributed Computing.*

46. Kabakuş, A. T. and Doğru, İ. A. (2018). An in-depth analysis of Android malware using hybrid techniques, *Digit. Investig.*, vol. 24, pp. 25–33.

47. Lashkari, A. H., A. Kadir, A. F., Taheri L., and Ghorbani, A. A. (2018). *Toward Developing a Systematic Approach to Generate Benchmark Android Malware Datasets and Classification*. Paper presented at 2018 2018 International Carnahan Conference on Security Technology (ICCST), Montreal, QC, Kanada.

48. İnternet: Understand The Machine Learning From Scratch For Beginners. URL: https://www.houseofbots.com/news-detail/3581-4-understand-the-machine-learning-from-scratch-for-beginners, Son Erişim Tarihi: 06.04.2019.

49. İnternet: Data Preprocessing for Machine learning in Python. URL: https://www.geeksforgeeks.org/data-preprocessing-machine-learning-python/, Son Erişim Tarihi: 06.04.2019.

50. İnternet: What is One Hot Encoding? Why And When do you have to use it? URL: https://hackernoon.com/what-is-one-hot-encoding-why-and-when-do-you-have-to-use-it-e3c6186d008f/, Son Erişim Tarihi: 06.04.2019.

51. İnternet: Scikit-learn- Machine Learning in Python. URL: https://scikit-learn.org/stable/, Son Erişim Tarihi: 06.04.2019.

52. İnternet: Gradient Tree Boosting or Gradient Boosted Regression Trees (GBRT) is a generalization of boosting to arbitrary differentiable loss functions. URL: https://scikit-learn.org/stable/modules/ensemble.html#gradient-tree-boosting, Son Erişim Tarihi: 06.04.2019.

53. İnternet: Classification: Accuracy. URL: https://developers.google.com/machine-learning/crash-course/classification/accuracy, Son Erişim Tarihi: 06.04.2019.

54. İnternet: Simple guide to confusion matrix terminology. https://www.dataschool.io/simple-guide-to-confusion-matrix-terminology/, Son Erişim Tarihi: 06.04.2019.

55. İnternet: Classification: ROC and AUC. URL: https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc/, Son Erişim Tarihi: 06.04.2019.

56. İnternet: Pickle — Python object serialization. URL: https://docs.python.org/3/library/pickle.html, Son Erişim Tarihi: 06.04.2019.

57. İnternet: Flask is a microframework for Python based on Werkzeug, Jinja 2 and good intentions. URL: http://flask.pocoo.org/, Son Erişim Tarihi: 06.04.2019.

ÖZGEÇMİŞ

Kişisel Bilgiler

Soyadı, adı : DAĞLIOĞLU, Abdullah
Uyruğu : T.C.
Doğum tarihi ve yeri : [redacted]
Medeni hali : [redacted]
Telefon : [redacted]
e-mail : [redacted]

Eğitim

Derece	Eğitim Birimi	Mezuniyet Tarihi
Yüksek lisans	Gazi Üniversitesi / Bilgisayar Bilimleri	Devam ediyor
Lisans (Yandal)	Anadolu Üniversitesi / Havacılık Elektr.	2013
Lisans	Gazi Üniversitesi / Bilg. Mühendisliği	2013
Lise	Mimar Sinan Lisesi	2006

İş Deneyimi

Yıl	Yer	Görev
2013-2020	Sebit A.Ş. (Türk Telekom)	Yazılım Mühendisi

Yabancı Dil

İngilizce

Yayınlar

Dağlıoğlu, A. ve Doğru, İ. A. (2020). *Android İşletim Sisteminde Kötücül Yazılım Tespit Sistemleri*. Dicle Üniversitesi Mühendislik Fakültesi Mühendislik Dergisi, 11(2), 499-511

Hobiler

Seyahat etmek, kitap okumak.

GAZİLİ OLMAK AYRICALIKTIR…