T.C.
GEBZE TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

ANDROID ZARARLI YAZILIM TÜRLERİNİN TESPİTİ

HASSAN MAHAMAT
YÜKSEK LİSANS TEZİ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

GEBZE
2023

T.C.
GEBZE TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

ANDROID ZARARLI YAZILIM TÜRLERİNİN TESPİTİ

HASSAN MAHAMAT
YÜKSEK LİSANS TEZİ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

DANIŞMANI
PROF. DR. İBRAHİM SOĞUKPINAR

GEBZE
2023

T.R.

GEBZE TECHNICAL UNIVERSITY

GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

DETECTION OF ANDROID MALWARE VARIANTS

HASSAN MAHAMAT

A THESIS SUBMITTED FOR THE DEGREE OF

MASTER OF SCIENCE

DEPARTMENT OF COMPUTER ENGINEERING

THESIS SUPERVISOR

PROF. DR. İBRAHİM SOĞUKPINAR

GEBZE

2023

GEBZE TEKNİK ÜNİVERSİTESİ	**YÜKSEK LİSANS JÜRİ ONAY FORMU**

GTÜ Fen Bilimleri Enstitüsü Yönetim Kurulu'nun 07/02/2023 tarih ve 2023/11 sayılı kararıyla oluşturulan jüri tarafından 08/03/2023 tarihinde tez savunma sınavı yapılan Hassan MAHAMAT'ın tez çalışması Bilgisayar Mühendisliği Anabilim Dalında YÜKSEK LİSANS tezi olarak kabul edilmiştir.

JÜRİ

ÜYE (TEZ DANIŞMANI)	:Prof.Dr.İbrahim SOĞUKPINAR
ÜYE	:Doç.Dr.Mehmet GÖKTÜRK
ÜYE	:Prof.Dr.Ali Gökhan YAVUZ

ONAY

Gebze Teknik Üniversitesi Fen Bilimleri Enstitüsü Yönetim Kurulu'nun

…..…/…..…/..…..… tarih ve ……/…….. sayılı kararı.

İMZA/MÜHÜR

ÖZET

Android yavaş yavaş onu hedefleyen kötü amaçlı yazılım haline geliyor. Bu tezde, Android APK'sında Android kötü amaçlı yazılım varyantlarının tespitine odaklanılmıştır. Kötü amaçlı yazılım geliştiricileri tarafından kötü amaçlı yazılım varyantları oluşturmak ve Android APK'sından izin ve API Çağrıları çıkarmak için kullanılan gizleme tekniklerini analiz edilmiştir. Tezde, geleneksel antivirüs yazılımını karşılaştırarak android kötü amaçlı yazılım varyantı tespiti için bir yöntem önerilmiştir.

Kötü amaçlı yazılım geliştiricileri, bazı araç algılamaları veya antivirüs şirketleri tarafından tespit edilmekten kaçınmak için kötü amaçlı yazılım varyantları oluşturmak üzere gizleme tekniklerini kullanır.˙Veritabanı güncellenmezse antivirüsün bu varyantların imzasını tespit etmesi zordur. Bu nedenle, siber saldırıları önlemenin, tespit etmenin ve bunlara karşı koymanın yeni yollarını keşfetmek çok önemlidir. Bu algılama mekanizmalarında makine öğrenimi, bir uygulamanın tehlikeli olup olmadığını belirleyen sınıflandırıcılar oluşturur. Araştırmada, Android APK'sında Android kötü amaçlı yazılım tespitine odaklanılmıştır. Kötü amaçlı yazılım yazarları tarafından kötü amaçlı yazılım varyantları oluşturmak için kullanılan şaşırtma teknikleri, Android APK'sından izin ve API Çağrıları analiz edilmiştir. Yöntemleri ve geleneksel antivirüsün kötü amaçlı yazılım türevlerini algılamasının başarımları karşılaştırılmıştır.

Anahtar Kelimeler: Kötü Amaçlı Yazılım Varyantları, Android, Gizleme, Kötü Amaçlı Yazılım Analizi ve Tespiti, API Çağrıları, İzin, Makine Öğrenimi.

SUMMARY

Android is slowly becoming malware targeting it. This thesis focuses on detecting Android malware variants in Android APK. We analyzed the obfuscation techniques used by malware developers to create malware variants and extract permissions and API Calls from Android APK. In the thesis, a method for Android malware variant detection by comparing traditional antivirus software is proposed.

Malware developers use obfuscation techniques to create variants of malware to avoid detection by some tool detections or antivirus companies.˙If the database is not updated, it is difficult for the antivirus to detect the signature of these variants. That's why it's so important to discover new ways to prevent, detect, and counter cyberattacks. In these detection mechanisms, machine learning creates classifiers that determine whether an application is dangerous or not. The research focused on Android malware detection in Android APK. Obfuscation techniques used by malware authors to create malware variants, permission, and API Calls from Android APK are analyzed. The methods and performances of traditional antivirus-detecting malware variants are compared.

Key Words: Malware Variants, Android, Obfuscation, Malware Analysis and Detection, API Calls, Permission, Machine Learning.

TEŞEKKÜR

Engin bilimsel bilgilerini benimle paylaşmakla kalmayıp aynı zamanda bana hayattan büyük dersler veren hocam Prof. İbrahim SOĞUKPINAR'a derin ve içten şükranlarımı sunarım. Onun desteği, önerileri ve cesaretlendirmesi bana bu çalışmayı tamamlama dürtüsü ve isteği verdi.

Bu araştırmadaki işbirliğinden dolayı Dr. Shahid ALAM'a sıcak ve içten teşekkürlerimi sunmak isterim.

Başta Selver abla ve Sadık YILDIRIM olmak üzere arkadaşlarıma ve tanıdıklarıma teşekkür ederim.

Aileme sevgileri ve destekleri için minnettarım.

İÇİNDEKİLER

SİMGELER VE KISALTMALAR DİZİNİ

Simgeler ve Kısaltmalar		Açıklamalar
GP	:	Gerçek Pozitif
GN	:	Gerçek Negatif
YP	:	Yanlış Pozitif
YN	:	Yanlış Negatif
Doğruluk	:	(GP+GN)/(GP+YP+GN+YN)
Duyarlık	:	GP/GP+YN
Kesinlik	:	GP/GP+YP
RO	:	Rastgele Orman (Random Forest)
SVM	:	Destek Vektör Makineleri(Support Vector Machine)
ML	:	Makine öğrenme(Machine Learning)
GPO	:	Gerçek Pozitif Oranı
YPO	:	Yanlış Pozitif Oranı
CM	:	Karışıklı Matrisi (Confusion matrix)

ŞEKİLLER DİZİNİ

TABLOLAR DİZİNİ

1. GİRİŞ

Android işletim sistemi için kötü amaçlı yazılım tespiti, bilgisayar güvenliği alanına katkıda bulunur. Android, mobil cihazlarda en çok kullanılan işletim sistemidir. McAfee tehdit raporuna [18] göre, Google oyunlarında bulunan kötü amaçlı yazılım ailelerinin sayısı 2017'de %30 arttı. Android, kötü amaçlı yazılım saldırıları tarafından en çok hedeflenen olmaya devam ediyor. Kötü amaçlı yazılım oluşturucular, kötü amaçlı yazılım varyantları oluşturmak için gizleme tekniklerini kullanır; bazı araçların algılamasıyla algılamadan kaçınmak için .˙Veritabanı güncellenmiyorsa antivirüsün bu varyantların imzasını algılaması zordur. Günümüzde kötü amaçlı yazılım oluşturma araçları, herhangi bir geliştiriciye mevcut kötü amaçlı yazılımın önceki aynı kod kaynağına dayalı olarak kötü amaçlı yazılım varyantları oluşturması için kolay bir olanak sağlamıştır. Antivirüs, kötü amaçlı yazılımlarını belirlemek ve imzalarını oluşturmak için kötü amaçlı yazılım varyantlarını manuel olarak analiz etti. Artan kötü amaçlı yazılım varyantlarının sayısı ve manuel analiz tespiti tarafından kullanılan tespit modellerinden kaçınmak için otomatik olarak oluşturulan otomatik araçlar ile. Yeni tehditlere yanıt verme sürecinin önünde bir engel haline geliyor. Antivirüs, çoğunlukla kötü amaçlı yazılım tespiti için statik bir imzaya dayandığından. Kötü amaçlı örnekler, kötü amaçlı yazılım kaynağının kod kaynağını değiştirmek için şaşırtma teknikleri kullanarak bu tespitten kurtulabilir. Bu nedenle, manuel analizi hızlandırmak için android kötü amaçlı yazılım varyantlarının tespiti kullanılabilir. Kötü amaçlı yazılım [12] türevleri aynı kodu, belki aynı semantik veya sözdizimiyle kullanırken veya talimat dizileri benzer olabilir.

Tez çalışmasında, geleneksel antivirüs yazılımını karşılaştırarak android kötü amaçlı yazılım varyantı tespiti için bir yöntem önerilmiştir. Bu durumda makine öğrenimi, bir uygulamanın tehlikeli olup olmadığını belirleyen sınıflandırıcılar oluşturur. Araştırmada, Android APK'sında Android kötü amaçlı yazılım tespitine odaklanılmıştır. Kötü amaçlı yazılım yazarları tarafından kötü amaçlı yazılım varyantları oluşturmak için kullanılan şaşırtma tekniklerini analiz ediyoruz. Android APK'sından izin ve API Çağrılarını analiz ediyoruz. Yöntemleri ve geleneksel antivirüsün kötü amaçlı yazılım türevlerini algılamasının başarımları karşılaştırılmıştır.

1.1. Tezin Katkısı

Bu tezde, geleneksel antivirüs yazılımını karşılaştırarak android kötü amaçlı yazılım varyantı tespitini öneriyoruz.

Geleneksel antivirüs (Kaspersky, McCafe..) imzaya bağlıydı ve dosyaları indirmek için kullanılan araçlar, yeni tehditleri algılayabilen bir yazılımdı. Virüs imzalarının temel kırılganlığı, yeni virüslerin saptanmasındadır.

Bir antivirüs firması antivirüs yazılımı bulduğu için kötü amaçlı yazılım araştırıcıları tarafından incelenir. Kötü amaçlı tür açığa çıktığında, dosyanın imzası çıkarılır ve virüsten koruma yazılımının imza veritabanına eklenir.

İmzaya dayalı teknik, kötü amaçlı yazılım salgını içinde çok başarılı olsa da, kötü amaçlı yazılım yazarları, kod bölümlerini şifreleyebilen veya imzalara karşılık gelmesi için kendisini kamufle edecek şekilde değiştirebilen virüsler tasarladılar.

Geleneksel antivirüs, tespit edilecek virüsün imzasına dayanır. Antivirüs veritabanı güncellenmezse, antivirüsün virüsleri algılaması zor olacaktır.

Bu nedenle, geleneksel antivirüsün yeni kötü amaçlı yazılımı tespit edemediği genellikle onaylanan imzanın özelliğidir.

Mobil kötü amaçlı yazılım saldırılarını tespit etmek için yeni yöntemlere ve tekniklere ihtiyacımız var.

Yöntemimiz çağrıları ve izni birleştiriyor, analiz ediyoruz. Bunu ve algılamayı onaylamak için kullanılan makine öğrenimi araçları karşılaştırılmıştır.

1.2. Tez Taslağı

Bu tezin amacı, geleneksel antivirüs yazılımını karşılaştırarak android kötü amaçlı yazılım varyantı tespitini incelemektir.
Bu tezin geri kalanı şu şekilde organize edilmiştir:

- Bölüm 1: Giriş
- Bölüm 2: Temel bilgi ve ilgili çalışma
- Bölüm 3: Metodoloji
- Bölüm 4: Sonuçlar ve Değerlendirme

Bölüm 2'de temel bilgili ve ilgili çalışmaları, Bölüm 3'te metodolojiyi, Bölüm 4'te Deneysel Değerlendirmeyi ve özellik birleştirme ile yöntemin gelişimini sunuyoruz ve Bölüm 5’te nihayet tezi sonuçlandırıyor.

2. TEMEL BİLGİ VE İLGİLİ ÇALIŞMA

2.1. İlgili Çalışma

2.1.1. API Çağrılarını Kullanarak Kötü Amaçlı Yazılım Analizi

API çağrıları ile analiz, Android izinleri ile en çok kullanılan özellikler arasında yer almaktadır. İster statik ister dinamik olsunlar, bir uygulamanın davranışını modellemeyi ve üstlenebileceği eylemleri karakterize etmeyi mümkün kılarlar. Örneğin DroidMat [1], çeşitli uygulama bileşenlerinden gelen API çağrılarını izler. DroidAPIMiner [2] ayrıca yazarlar tarafından kritik kabul edilen API çağrılarına odaklanır ve çağrı sırasında makine öğrenimi tekniklerini kullanarak bir algılama modeli eğitmek için kullanılan bağımsız değişkenleri içerir. DroidMiner [3], API işlevleri ile hassas Android kaynakları arasındaki iletişime ve SVM veya Random Forest gibi farklı sınıflandırma algoritmalarına dayalı olarak CBG (Component Behavior Graphs) adı verilen davranış grafikleri oluşturur. DroidSIFT [4], bir Naive Bayes sınıflandırıcısını eğitmek için API bağımlılık grafiklerini kullanan semantik tabanlı bir sınıflandırıcıdır.

2.1.2. İzni Kullanarak Kötü Amaçlı Yazılım Analizi

Meta bilgi analizi Birçok yöntem, yazılım algılama mekanizmalarını yalnızca geliştiriciler tarafından sağlanan meta bilgilere dayandırır. AndroidManifest.xml dosyasından çıkarılan bir dizi özellik ile makine öğrenimi algoritmalarını eğiten ve android uygulamasının açıklama metninde bir metin çıkarma işlemi gerçekleştiren bir sistem olan ADROIT [5] durumu böyledir. Manilyzer [6], makine öğrenimi algoritmalarını eğitmek için Android'in AndroidManifest.xml dosyasından toplanabilecek bilgileri kullanmaya odaklanan başka bir örnektir. Sistem izinleri, literatürde yaygın olarak kullanılan meta bilgilerdir. Aslında bunlar, bir Android uygulamasının AndroidManifest.xml dosyasında yer alan ve uygulamanın neleri yapıp neleri yapamayacağını gösterdiği için önemli olan bildirimlerdir. Kolayca okunabilirler. İzinler, Android kötü amaçlı yazılım tespitini gerçekleştirmek için en alakalı özelliklerden biridir. Örneğin, Android sistemlerine yapılan izin kaynaklı

saldırılar güvenlik ihlalleridir ve en kritik konular arasındadır. A. Sadeghi ve ark. [7], Android için geçici bir izin analizi ve uygulama çerçevesi olan TERMINATOR'u önermektedir. Geçici mantık modeli doğrulamasından yararlanan tarayıcı, uygulama durumlarının dinamik yetkilendirmesinden kaynaklanan tehditleri tanımlar.

Çok azı statik program analizini kullanarak kötü amaçlı yazılım tespitini arar. Uygulamaları statik olarak inceleyen ve kodlarını parçalara ayıran önerilen tüm teknikler.

Umme ve al [8], çeşitli makine öğrenimi modellerinden yararlanan birkaç Android uygulamasını analiz ediyor. Farklı özellikler almak ve çeşitli sınıflandırıcılar uygulamak .

[9]araştırmacılar, bir Android cihazında kötü amaçlı yazılım tespiti için çok düzeyli bir izin çıkarma çerçevesi önerdiler. Çerçeve, izinleri analiz ederek kötü amaçlı uygulamaları etiketlemek için bir izin çıkarma yaklaşımı kullanır ve performans ölçümlerini optimize ederken çok sayıda uygulamayı yönetme yeteneğine sahiptir.

[10] Bu araştırmanın amacı, tespit etmek için statik analizin etkinliğini incelemektir.
İzne dayalı özellikleri kullanarak Android kötü amaçlı yazılımı. Bu çalışma, Android kötü amaçlı yazılım tespitini değerlendirmek için kullanılan farklı sınıflandırıcı kümeleriyle makine öğrenimini önermektedir.

[11] Manifest İzinlerini Kullanan Makine Öğrenimi Tabanlı Android Kötü Amaçlı Yazılım Tespiti araştırması, uygulamaları kötü amaçlı veya zararsız olarak sınıflandırmak için Android bildirim dosyası izinlerinden seçilen özelliklerle birlikte dört farklı makine öğrenimi algoritmasının etkinliğini araştırır.

2.1.3. Diğer teknikleri kullanan kötü amaçlı yazılım analizi

[12] tarafından sunulan bir teknik, Android kötü amaçlı yazılımını algılamak için neredeyse ıskalanan bir klon detektörü olan NiCad [16]'yı kullanır. Bu teknik, kötü amaçlı yazılım kodunun klonu olarak algılanan uygulama kodunun alt kümesine dayalı olarak Android uygulamalarında kötü amaçlı yazılım sınıflarının algılanmasına olanak tanır. Teknik, yalnızca java kaynak kodu düzeyinde çalıştığı için Android'in yerel kaynak kodunu algılayamaz. Nicad [16], klonları saptamak için en uzun ortak alt dizi

algoritmasını kullanır. Nicad tarafından kullanılan teknik, aynı kodun algılanmasını optimize etmek için satır bazında karşılaştırmaya dayanmaktadır. Ancak, kodun anlamı ve sözdizimi değiştirilirse, kötü amaçlı yazılım türevlerini verimli bir şekilde algılayamaz. [13]'te yazarlar, Android Kötü Amaçlı Yazılım Varyantlarını tespit etmek için DroidClone'u sundular. DroidClone, sorunun [16]'da bulunan kısmını çözdü çünkü Droidclone hem bayt kodunu hem de yerel kodu algılayabilir ve ayrıca kötü amaçlı yazılım varyantlarını da algılayabilir. DroidClone, benzer kod bölümleriyle çalışır ve Android APK kötü amaçlı yazılımını algılamak için Kötü Amaçlı Yazılım Analizi ve Ara Dil (MAIL) kullanır. İlk olarak Droidclone, Android kötü amaçlı yazılım varyantlarını bulmak için kod klonları kullanır. İkincisi, Droidclone, MAIL'in yardımıyla Android APK için platformlar arası imzalar kullanır ve yerel kod düzeyinde çalışarak bayt koduna veya yerel koda gömülü kötü amaçlı yazılımları algılamasına olanak tanır. [14]'te Bileşen tabanlı bir Kontrol Akış Grafiği (CBCFG) önerilmiştir. CBCFG, Android API'lerinin bir grafiğidir. Bir CBCFG'nin düğümü bir API'dir ve uç, bunların kontrol akışı öncelik ilişkisini temsil eder. CBCFG, Android yeniden paketlenmiş uygulamasında ve kötü amaçlı yazılım türevlerinde kodun yeniden kullanımını algılamak için kullanılır. Sahte API çağrıları, gizleme API çağrıları ve satır içi API'ler kullanılarak değiştirilen kod bu teknik tarafından algılanmayabilir. API çağrılarının kontrol akışı kalıpları değiştirilirse, kötü amaçlı yazılımları tespit etme görevi daha zor hale gelir. [15]'te yazarlar, piggyback olan Android kötü amaçlı yazılımını tespit etmek için imzalar oluşturmak üzere bir API çağrı grafiği kullanan DroidLegacy'yi sundular. Bu teknik, yukarıda tartışılanla aynı sınırlamalara sahiptir [15].

2.2. Arka Plan

2.2.1. Gizleme Tekniği

Android uygulamaları, burada APK dosyası olarak kabul edilen pakete göre yeniden gruplandırılmıştır ve tersine mühendislik için bir araç olan APKTOOL ile birlikte kullanılır. Orijinalleri kaynak koda dönüştürebilir, derleyebilir ve daha az değişiklik yaptıktan sonra bunları oluşturabilir. Bu araç, python koduyla birlikte kötü amaçlı yazılım varyantları oluşturmaya olanak tanır, yani 132 kötü amaçlı yazılım

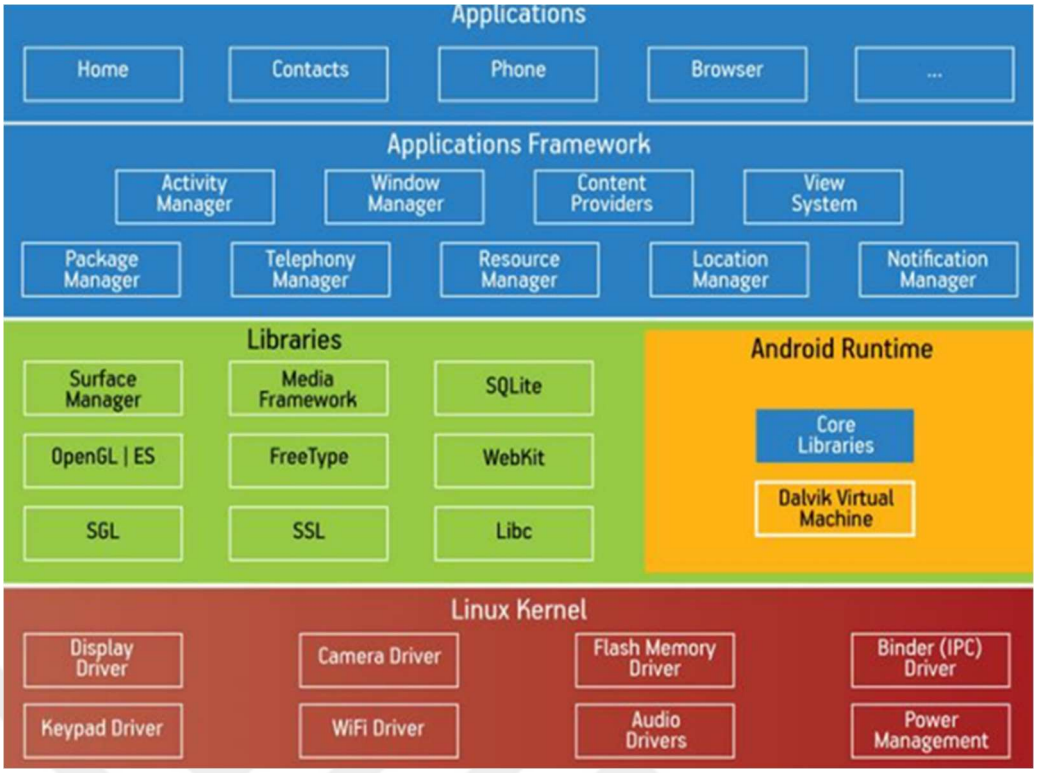
örneğiyle, kötü amaçlı yazılım varyantları oluşturmak için şaşırtma teknikleri kullandık. Bunun için NOP (işlem ekleme yok), CND (Çağrı yönlendirme), RDI (hata ayıklama bilgilerini kaldırma) ve RVD (Siparişi Tersine Çevirme) gibi bazı karartmalar oluşturuyoruz. Bazılarında 132 varyant oluşturduk ve VirusTotal.com algılama web sitesiyle test ettik. NOP kullandığımda, APK dosyasının her dosyası için (kötü amaçlı yazılımdan oluşan APK dosyası) 86 NOP algılama oranı buluyoruz, tüm dosyalar oluşturuluyor, 100'ün sonucu 33 örnekle eğitiliyor.

2.2.2. Android Mimarisi

Akıllı telefon gereksinimlerini desteklemek için bileşenlerden oluşan bir yazılım yığınıdır. Android uygulamaları APK'yı bir uzantı olarak arşivler, APK bileşeni: izin içeren XML dosyası, class.dex dosyası API içerir, yürütülebilir dosya olarak tanımlanır, kaynaklar dizinleri res içerir, kitaplık adının bir dizini lib olarak ve bir dizin adını meta olarak içerir kaynak ve sertifika içerir.

Android mimarisinin ana bileşenleri şunlardır:

- 1.Uygulamalar
- 2.Android Çerçevesi
- 3.Android Çalışma Zamanı
- 4.Kütüphaneler
- 5.Linux çekirdeği



Şekil 2.2 : Android Mimarisi.

2.2.3. Güvenlik Sorunu

İnternette kötü amaçlı yazılım ve virüslerin varlığı sürekli ve değişkendir. Bilgisayar korsanları, herhangi bir sayıda hedef için sürekli olarak yeni yazılım yolları yaratıyor.

- Kişisel bilgileri çalmak
- Banka veya kredi çalmak
- Hizmet Reddi (DDOS) saldırıları

3. METODOLOJİ

3.1. İzin Tespiti

Android uygulamaları, geleneksel ZIP sıkıştırmasına dayanan bir Android Paket Kiti (APK) formatı oluşturur. Kötü amaçlı yazılım oluşturucular, yeni kötü amaçlı yazılım oluşturmak için APK'nın ters mühendisliği kullanır. APK arşivinde AndroidManifest.xml dosyası da bulunur. Şekil 3.1, izin listesini gösterir.

İzin, uygulamaların hassas ortamlara veya android akıllı telefonlardaki donanım özelliklerine erişmesine izin veren özel bir ayrıcalıktır.

22 İzin	
erişim_wifi_durumu	okuma_günlüğü
kamera	telefon_durumunu_oku
ağ_durumunu_değiştir	okuma_sms
wifi_durumunu_değiştir	önyükleme_tamamlandı
tuş_kilidini _devre_dışı_bırak	yeniden_paketler
görev_almak	sms_gönder
paketleri_kur	duvar_kağıdı_ayarla
arama_kaydını_oku	sistem_uyarı_pencereleri
kişileri_oku	apn_ayarını_yaz
harici_depolamayı_oku	kişileri_yaz
geçmiş_imlerini_oku	yazma_ayarı

Şekil 3.1: İzin.

3.2. Uygulama Programı Tespiti (API)

Bir uygulama programı arabirimi (API), yazılım programının başka bir yazılım programı veya harici bir sistemle etkileşime girmesini sağlayan bir dizi işlev, komut ve protokole sahip bir koddur.

API çağrıları, örneğin; beklenen davranışını modellemeyi mümkün kılar. Kötü amaçlı yazılım örnekleri arasında bulunan API çağrıları.

Şekil 3.2'de gösterilen sınıfa göre gruplandırılır.

Sınıf	Tanım	Yöntem örneği
Content Resolver	İçerik sağlayıcılarına erişim sağlar.	Insert(),query()
Context	Sınıflar, kaynaklar, varlıklar vb. Gibi evrensel verilerin alınmasına yardımcı olur.	openFileOutput(), getApplicationInfo()
Intents	Uygulamanın diğer etkinlikleri, etkinlikleri ve hizmetleri başlatmasını sağlar. Ayrıca uygulamaların cihazın donanımıyla etkileşime girmesine izin verir.	setDataAndType(), addFlags()
ActivityManager	Cihazın diğer çalışan aktiviteleriyle ilişkilendirmeye erişim sağlar.	getRunningServices(), getMemoryInfo()
PackageManager	Hedef sistemdeki yüklü paketlerle ilgili bilgi almaya yardımcı olur	getInstalledPackages()
SmsManager	SMS işlemlerini yönetmeye yardımcı olur	sendTextMessage()
TelephonyManager	Bilgi ile ilgili sistemin telefon hizmetlerinin getirilmesine yardımcı olur.	getDeviceId(), getLine1Number()
Connectivity Manager, NetworkingInfo, WifiManage	Ağla ilgili bilgileri sağlar	GetNetworkInfo(), isConnected(), getWifiState()
HttpURLConnection	Bu sınıf, uzak sunucularla bir bağlantı kurmamızı ve ayrıca veri göndermemizi ve almamızı sağlar.	SetRequestMethod(), getInputStream()

Şekil 3.2: API ÇAĞRILARI Kategorileri.

3.3. Kullanılan Konsept ML Algoritması

- Adaptive Boosting olarak da adlandırılan AdaBoost, Makine Öğreniminde Topluluk Yöntemi olarak kullanılan bir tekniktir.
- Destek Vektör Makinesi ve Rastgele Orman, makine öğreniminde en çok kullanılan algoritmalar oldukları için bölüm denemelerinde kullanılmıştır. Hem sınıflandırma hem de regresyon zorlukları için kullanılabilir.

Kötü amaçlı yazılım tespiti için kategori sınıflandırması verilir. Bu nedenle RO, dosdoğruluk ve kesinlik sağlar. Çünkü RO, karar ağacının bir kümesidir.

- N Katlı Çapraz Doğrulama

Sunulan model kötü amaçlı ve zararsız olmak üzere iki sınıfa ayrılmıştır. Özellik, bölüm 4.1'de kötü amaçlı ve iyi huylu olarak belirtildiği gibi bir dizi satır API'si ve izin ve sütunlar sundu.

Çapraz doğrulama, makine öğrenimi modelinizin görünmeyen verilerin sonucunu ne kadar iyi tahmin edebileceğini belirlemek için makine öğreniminde kullanılan bir değerlendirme yöntemidir. Kötü amaçlı yazılımların tespiti için N-katlama yaklaşımı kullanılır.

Kötü amaçlı yazılım tespitine verimli bir yaklaşım oluşturmak için bu yedi ölçüt kullanıldı:

- GP,GN,YP,YN .
- DOĞRULUK, DUYARLIK, KESİNLİK.

4. SONUÇLAR VE DEĞERLENDIRME

Bizim için veri kümesi, [20] MalGenomeProject, [21]'deki iyi huylu ve kötü amaçlı yazılım özütüne ve ayrıca DroidKungFu4'ün veri kümesine göre yeniden gruplandırılır.

Android uygulaması, burada APK dosyası olarak kabul edilen paket tarafından yeniden gruplandırılır ve tersine mühendislik için bir araç olan Apktool ile kullanılır. Orijinalleri kaynak koda dönüştürebilir, derleyebilir ve daha az değişiklik yaptıktan sonra bunları oluşturabilir,Şekil.4.1, gösterdiği gibi Python koduyla birleştirilmiş bu araçlar, kötü amaçlı yazılım varyantlarının oluşturulmasına izin verir. Tablo 4.1, her gizlemenin ortalamasını ve 33 kötü amaçlı yazılım örneğini ve 132 varyantı gösterir.kötü amaçlı yazılım varyantları oluşturmak için şaşırtma teknikleri kullandık. Bunun için NOP (işlem ekleme yok), CND (Çağrı yönlendirme), RDI (hata ayıklama bilgilerini kaldırma) ve RVD (Reversing Order) gibi bazı gizlemeler oluşturuyoruz.

Şekil 4.2'ü gösterecek şekilde. Antivirüs, 4 şaşırtma ile kötü amaçlı yazılım varyantlarını tespit ediyor, varyantları oluşturduk ve VirusTotal.com tespit web sitesiyle test ettik, her defasında adware ve Kaspersky %84.875, Drweb %84.115 ve Mcafee %22.0375 nops tespit oranı bulduk. APK dosyası.

NOP kullandığımda, tüm dosyalar 33 örnekle oluşturulur ve eğitilir, bu da %100 kötü amaçlı yazılım varyantlarının oluşturulmasına, bunun iyi doğruluğuna ve etkinliğine neden olur. RDI'da da durum aynıdır. 132değişken oluşturulduğunda bunları Virustotal'da analiz ettik ve herhangi bir antivirüs ticari Mcafee tespit edilmedi.

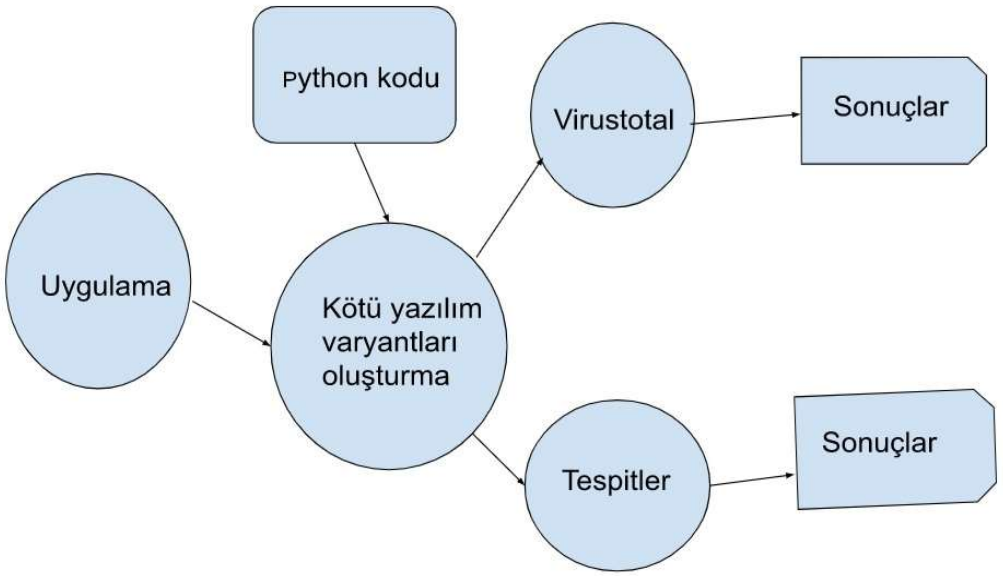
Ancak DR ve Kaspersky'nin tespit oranı %86 gibi yüksek bir tespit oranına sahip. Bu nedenle diğer dosyaların değişken oluşturma olasılığı yoktur.

Talimat frekansına odaklanan algılama tekniğimiz, APK'nın bloke edeceği tüm dosyalarla varyantlar oluşturabilmelidir. İkinci kez algılama oranı antivirüsten(AV) daha iyi olmalıdır.

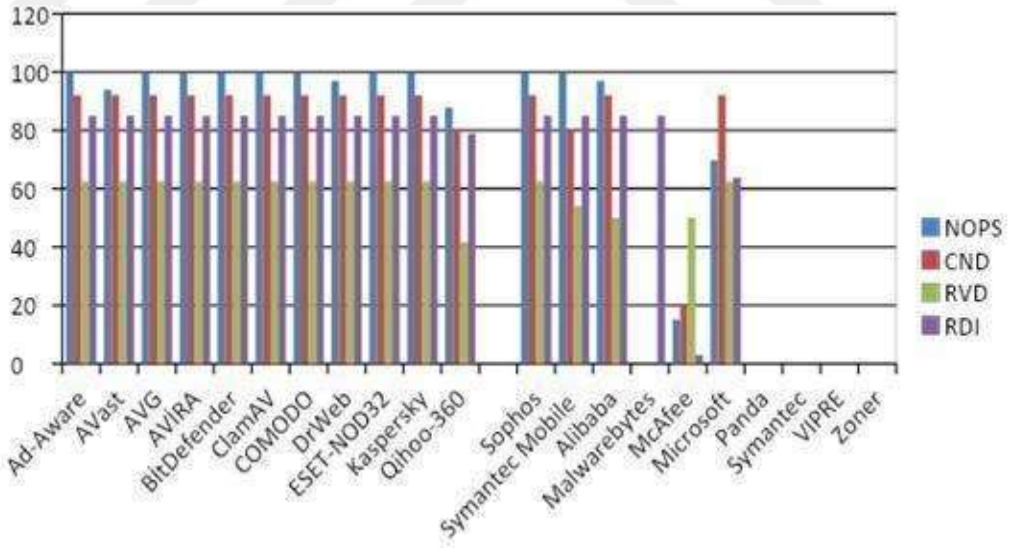
Çünkü AV imza veri tabanına dayalıdır ve erişim manuel olarak yapılır ve zaman alır. Bu nedenle, imza DB imzasında mevcut değilse, aynı önceki koda dayalı yeni kötü niyetli oluşumları tespit etmek mümkün değildir.

Tablo 4.1, NOPS'nin tespit edilen kolay karartma olduğunu tespit eden antivirüs testini göstermektedir.

Şekil 4.3'te Kaspersky algılama oranı %100 olan NOPS ve Şekil .4.4'de ortalama başarı NOP %69.55, CND %65.52, RVD %47.29 ve RDI %63.6 olarak gösterilirken.



Şekil 4.1: Kötü amaçlı yazılım varyantları oluşturma.

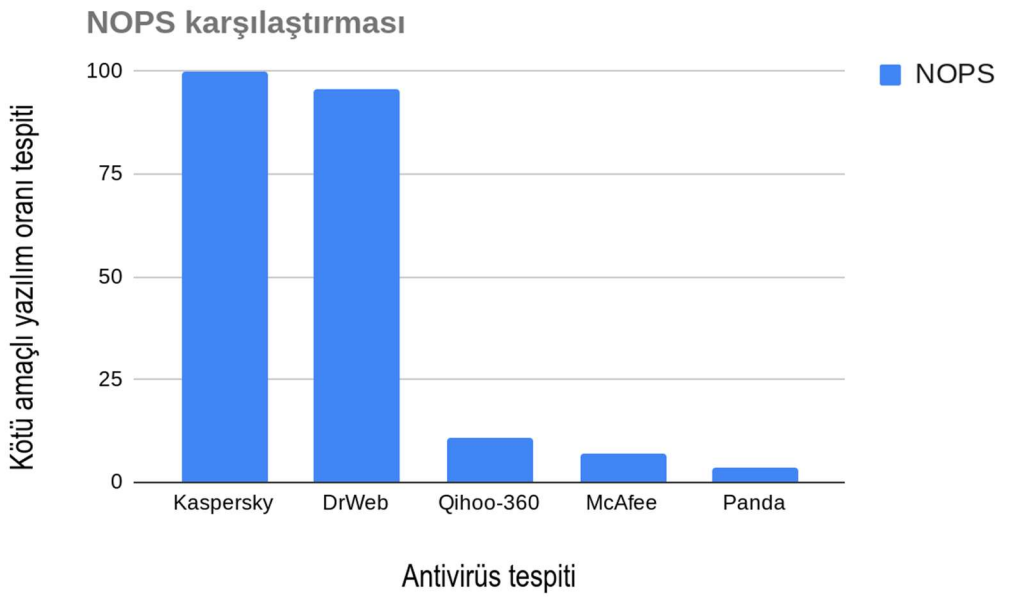


Şekil 4.2: Antivirüs, 4 gizleme ile kötü amaçlı yazılım varyantlarını algılar.

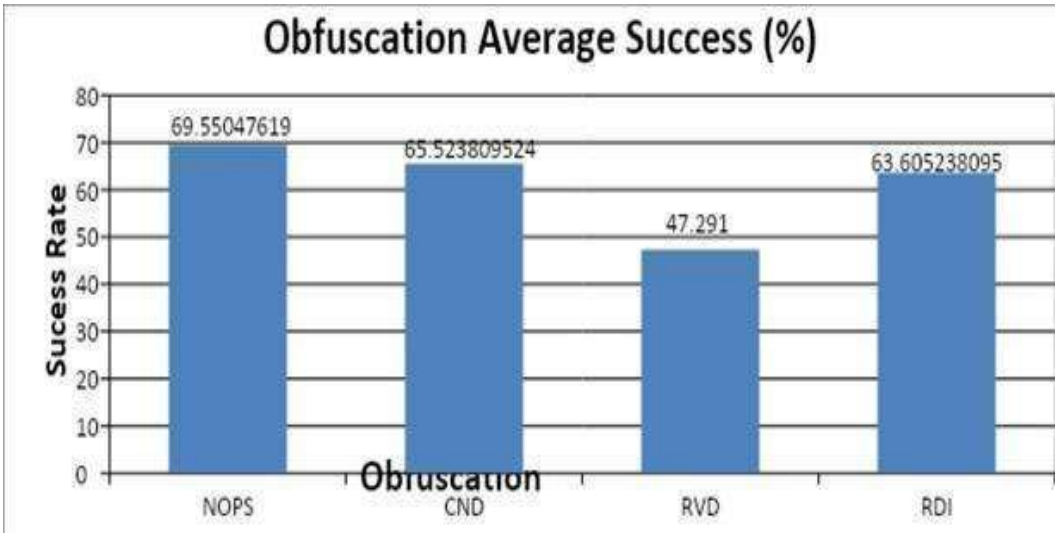
- NOPS: yerleştirme işlemi yok, algılaması ve oluşturması çok kolay bir gizleme türüdür.
- CND: Çağrı yönlendirme
- RVD: Sırayı Tersine Çevirme
- RDI: Hata Ayıklama Bilgilerini Kaldırma

Tablo 4.1 :kötü yazılım türevleri örnekleri (1:evet gizlemeyi algılar ve 0:hayır)

Dosya	NOP	RVD	CND	RDI
S1	1	0	0	1
S2	1	0	0	1
S3	1	1	1	1
S4	1	1	1	1
S5	1	1	1	1
S6	1	1	1	1
S7	1	1	1	1
S8	1	1	1	1
S9	1	1	0	1
S10	1	1	1	1
S11	1	1	1	1
S12	1	1	1	1
S13	1	1	0	1
S14	1	1	1	1
S15	1	0	0	1
S16	1	1	1	1
S17	1	1	1	1
S18	1	1	1	1
S19	1	1	1	1
S20	1	1	1	1
S21	1	1	1	1
S22	1	1	1	1
S23	1	1	1	1
S24	1	1	1	1
S25	1	1	1	1
S26	1	1	1	1
S27	1	1	1	1
S28	1	0	0	1
S29	1	1	1	1
S30	1	0	1	1
S31	1	0	1	1
S32	1	0	0	1
S33	1	1	1	1



Şekil .4.3: Antivirüs, kötü amaçlı yazılım türevlerini NOP olarak algılar.



Şekil .4.4: Gizleme Tekniğinin ortalama başarısı.

4.1. Özellik Birleştirme İle Yöntemin Gelişimi

Bu bölümde, görev makine öğrenimi için önemlidir çünkü iyi model özellik çıkarma ve seçmeye aittir. Kötü amaçlı ve iyi huylu verileri ayıklamak için AndroGuard ve APKTool araçlarını kullandık. Androguard, Android dosyalarının

DEX, APK, XML, demonte edilmesi ve kaynak koda dönüştürülmesi dahil olmak üzere çoğu Android dosyasından bilgi çıkarmak için farklı amaçlar için kullanılır.

Koleksiyon için, özellik çıkarımı için zararsız ve kötü amaçlı verileri etiketledik. Ayıklanan veriler, izinler, API Çağrıları, amaçlar vb. gibi birden fazla alan içerir.

Bu kısım için, Android APK'sından çıkarılan şekil 3.1 ve Şekil3. 2'nin özelliklerini kullandık.

Açıklaması :

- Kötü amaçlı yazılım ve iyi huylu veri kümesi örnekleri toplanır.
- classs.dex'ten API çağrılarını ve manifest.XML'den ve sınıflanmış iki sınıftan izinleri çıkartılır.
- veri kümesindeki her APK için toplanan özellikler (API çağrıları ve izinler) ve bunları bir CSV dosyasında özellikler olarak depolanır. Örnek bir CSV dosyası:

A1 A2 A3 ..... Bir P1 P2 P3 ..... Pn Sınıfı

0 1 1 0 1 0 1 0 1(kötü amaçlı yazılım)
0 0 1 0 1 0 1 0 1(kötü amaçlı yazılım)

0 1 0 1 1 0 0 0 1(kötü amaçlı yazılım)

1 0 0 0 1 0 0 0 0(iyi huylu)

1 1 0 0 0 1 0 1 0(iyi huylu)

1 0 1 0 0 1 1 1 0(iyi huylu)

1'in mevcut olduğu ve 0'ın mevcut olmadığı anlamına geldiği A1, A2, ..., An API çağrılarıdır ve P1, P2, ..., Pn izinlerdir

- Özellik Birleştirme Siniflandirmasi

Kötü amaçlı yazılım tespitine verimli bir yaklaşım için yedi ölçüt kullanıldı:

- GP,GN,YP,YN .
- DOĞRULUK, DUYARLIK, KE

Tablo 4.2: Sonuç algılama sınıflandırması.

sınıflandırma	Doğruluk	Duyarlılık	kesinlik	f1_score
Destek Vektörü	94.28	93.6	93.7	93.7
Karar ağacı	92.88	93.4	91.1	92.2
KNN	94.33	92.9	94.6	93.5
rastgele orman	96.02	94.1	92.8	93.5
Adaboost	90.27	87.9	91.9	88.8

Tablo 4.1.2: API CALL ve İzin sınıflandırma tespiti sonuçlarını açıklar. Tespit oranını destek vektörü, karar ağacı ve rastgele orman ile elde ediyoruz, oysa rastgele orman 96.02 doğruluk, 94.1 Duyarlılık ve Destek Vektörü 94.28 doğruluk ve duyarlılık olarak %93.6. Rastgele orman ve Destek Vektörü verimli sınıflandırmalardır.

Tablo 4.1.2'nin sonucu, makine öğrenimini kullanarak doğru ve yanlış tahminlere dayanan karışıklık matrisini gösterir.

5. SONUÇLAR VE YORUM

Kötü amaçlı yazılım tespitine, bazı antivirüs şirketleri ve herhangi bir bilgi işlem mühendisliği veya android geliştirme, hızlı sınıflandırma ve kötü amaçlı yazılım tespiti için, makine yöntemini kullanan kötü amaçlı yazılım algılama odaklı kötü amaçlı yazılım sınıflandırma tekniği tarafından meydan okunmaktadır.

Sunulan yöntem, ilgili eser incelemesi kullanıldı ve başka bir çalışma yapılıp yapılmadığı kontrol edildi.

Android gerçekten kötü amaçlı saldırıların hedefi haline geldi, bu nedenle kötü amaçlı yazılım yazarları tarafından kötü amaçlı yazılım varyantları oluşturmak ve Android APK'sından izin ve API Çağrıları çıkarmak için kullanılan karartma teknikleri analiz edildi. Geleneksel antivirüs yazılımını karşılaştırarak android kötü amaçlı yazılım varyantı tespiti için etkili bir yöntem önerdik.

Yöntemimiz, geleneksel antivirüsün kötü amaçlı yazılım varyantlarını tespit etmesinin kolay olmadığını bilmemizi sağladı.Antivirüs, imza güncellemeleri veritabanını düzenli olarak tutmalıdır.API Çağrısı özelliğinin birleştirilmesi ve iznin yüksek bir tespit oranı olduğu kanıtlanmıştır. Yalnızca izin veya API Çağrısı kullanılırsa daha iyi iniş ve doğruluk sağlar.

Gelecekte, izin ve API CALL kullanımını tespit etme çalışmalarını aşağıdaki adımlarla genişleteceğiz:

Antivirüs ve bu yöntemi karşılaştırmak için veritabanını kullanan ilk adım.

İkinci adımda izin, API CALL ve antivirüs test etmek için kodun karartılması

Gelecekte, API çağrı anahtarları vb. gibi diğer özellikleri dahil ederek kullanılan yöntemin performansını daha da iyileştireceğiz. Ayrıca, bu belgede önerilen tekniği birden çok sınıflandırma için kullanacağız.

KAYNAKLAR

[1] Wu D. J., Mao C. H., Wei T. E., Lee H. M., Wu K. P., (2012), "Droidmat: Android malware detection through manifest and api calls tracing", Seventh Asia joint conference on information security, 62-69, Tokyo, Japan, 09-10 August.

[2] Aafer Y., Du. W., Yin H., (2013), "Droidapiminer: Mining api-level features for robust malware detection in android", In International conference on security and privacy in communication systems, 86-103, Sydney, NSW, Australia, 25-28 September.

[3] Yang C., Xu Z., Gu G., Yegneswaran V., Porras P., (2014), "Droidminer: Automated mining and characterization of fine-grained malicious behaviors in android applications", European symposium on research in computer security,163-182, Wroclaw, Poland, 7-11 September.

[4] Zhang M., Duan Y., Yin H., Zhao Z., (2014), "Semantics-aware android malware classification using weighted contextual api dependency graphs" , Proceedings of the 2014 ACM SIGSAC conference on computer and communications security, 1105-1116, Scottsdale, Arizona, USA, 3-7 November.

[5] Martin A., Calleja A., Menendez H. D., Tapiador J., Camacho D., (2016), "ADROIT: Android malware detection using meta-information", Symposium Series on Computational Intelligence, 1-8, Athens, Grece, 06-09 December.

[6] Feldman S., Stadther D., Wang B., (2014), " Manilyzer: automated Android malware detection through manifest analysis", 11th International Conference on Mobile Ad Hoc and Sensor Systems, 767-772, Philadelphia, PA, USA, 28-30 October.

[7] Sadeghi A., Jabbarvand R., Ghorbani N., Bagheri H., Malek S., (2018), "A temporal permission analysis and enforcement framework for android", Proceedings of the 40th International Conference on Software Engineering, 846-857, Gothenburg, Sweden, 27 May-3 June.

[8] Jannat U. S., Hasnayeen S. M., Shuhan M. K. B., Ferdous M. S., (2019), "Analysis and detection of malware in Android applications using machine learning", 2019 International Conference on Electrical, Computer and Communication Engineering (ECCE), 1-7, Cox'sBazar, Bangladesh 07-09 February.

[9] Ghasempour A., Sani N. F. M., Ovye J. A., (2020), "Permission extraction framework for Android malware detection", International Journal of Advanced Computer Science and Applications, 11(11), 1-13.

[10] Mohamad Arif J., Ab Razak M. F., Awang S., Tuan Mat S. R., Ismail N. S. N., Firdaus A., (2021), " A static analysis approach for Android permission-based malware detection systems", PloS one, 16(9), e0257968.

[11] Herron N., Glisson W. B., McDonald J. T., Benton R. K., (2021), "Machine learning-based android malware detection using manifest permissions", Proceedings of the 54th Hawaii International Conference on System Sciences, 1-10, Kauai, Hawaii, USA, 05 January.

[12] Chen J., Alalfi M. H., Dean T. R., Zou Y., "Detecting android malware using clone detection", Journal of Computer Science and Technology, 30(5), 942-956.

[13] Alam S., Riley R., Sogukpinar I., Carkaci N., (2016), " Droidclone: Detecting android malware variants by exposing code clones", 2016 Sixth International Conference on Digital Information and Communication Technology and its Applications (DICTAP), 79-84, Konya, Turkey, 21-23 July.

[14] Sun X., Zhongyang Y., Xin Z., Mao B., Xie L., (2014), "Detecting code reuse in android applications using component-based control flow graph", IFIP international information security conference, 142-155, Marrakech, Morocco, 2-4 June.

[15] Deshotels L., Notani V., Lakhotia A., (2014), "Droidlegacy: Automated familial classification of android malware", Proceedings of ACM SIGPLAN on program protection and reverse engineering workshop, 1-12, CA, San Diego, USA, 22-24 January.

[16] Alam S., Alharbi S. A., Yildirim S., (2020), "Mining nested flow of dominant APIs for detecting android malware", Computer Networks, 167(3), 107026.

[17] Farhat D., Awan M. S., (2021), " A brief survey on ransomware with the perspective of internet security threat reports", 9th International Symposium on Digital Forensics and Security (ISDFS), 1-6, Elazig, Turkey, 28-29 June.

[18] Ayoade G., Chandra S., Khan L., Hamlen K., Thuraisingham B., (2018), " Automated threat report classification over multi-source data", 4th International Conference on Collaboration and Internet Computing (CIC), 236-245, Philadelphia, PA, USA, 18-20 October 2018.

[19] Alam S., Qu Z., Riley R., Chen Y., Rastogi V., " DroidNative:Automating and optimizing detection of Android native code malware variants", computers security, 65(2), 230-246.

[20] Web 1, (2018 http://www.malgenomeproject.org/, (Erişim Tarihi: 5/08/2018).

[21] Web 2, (2018), https://www.kaggle.com/goorax/datasets, (accessed: 27/08/2018).

[22] Web 3, (2018), http://sccpu2.cse.ust.hk/elite/ downloadApks.html, accessed: 5/08/2018).

[23] Web 4, (2018), https://androguard.readthedocs.io/en/latest/,(Accessed: 05/08/2018).

ÖZGEÇMİŞ

Hassan MAHAMAT, Bangui Üniversitesi, Orta Afrika Cumhuriyeti lisans eğitimini tamamladı. Gebze Teknik Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Anabilim Dalında Yüksek Lisans yaptı.

EKLER

Ek A: Tez Çalışması Kapsamında Yapılan Yayınlar

Mahamat H., Soğukpınar İ., (2022) “Android Malware Variant Detection by Comparing Traditional Antivirus”, Uluslararası Bilgisayar Bilimi ve Mühendisliği Konferansı (UBMK), Diyarbakır, Türkiye, 14-16 Eylül.