Düzce Üniversitesi Bilim ve Teknoloji Dergisi

Araştırma Makalesi

Android Kötücül Yazılım Tespit Sistemleri İncelemesi

Ömer KİRAZ[a,*], İbrahim Alper DOĞRU[b]

[a] *Bilgisayar Mühendisliği Bölümü, Mühendislik Mimarlık Fakültesi, Gazi Üniversitesi, Ankara, TÜRKİYE*
[b] *Bilgisayar Mühendisliği Bölümü, Fen Bilimleri Enstitüsü, Gazi Üniversitesi, Ankara, TÜRKİYE*
** Sorumlu yazarın e-posta adresi: omerk1818@gmail.com*

ÖZET

Akıllı telefonların hayatımıza girmesiyle birlikte akıllı telefonları kullanan kullanıcı sayısı her geçen gün artarak devam etmektedir. Akıllı telefonların fazla talep görmesindeki neden, insanların bir cihazla istedikleri işleri tek dokunuşla kolaylıkla yapabilmesidir. International Data Corporation (IDC) firmasının 2016 2. çeyrek raporuna göre; akıllı telefon pazarında Android %87.6 gibi çok yüksek bir paya sahiptir [1]. Android'in akıllı telefon kullanan kullanıcılar arasında popüler olması ile birlikte açık kaynaklı bir yapıya sahip olması ve markete uygulama yüklenirken detaylı olarak kötücül yazılım incelenmesi yapılmadığından dolayı Android platformu kötü niyetli kişilerin bir numaralı hedefi haline gelmiştir. Android market ve diğer alternatif Android marketlerde kötü niyetli uygulamaların sayısı her geçen gün artmaktadır. G Data'nın 2015 1. Çeyrek raporuna göre; kötücül yazılımların %50.3 finansal amaçlıdır [2]. Finansal amaçlı olmasındaki neden Avrupa kıtasındaki kullanıcıların %41'nin banka işlemlerini akıllı telefonlarını kullanarak yapmasıdır [2]. Bu yüzden Android marketlerde bulunan uygulamaların kötücül olup olmadığını tespit etmek için etkin kötücül yazılım tespiti yapan sistemlere ihtiyaç vardır. Bu çerçevede bu çalışmada kötücül yazılım tespit sistemleri anlatılmıştır.

Anahtar Kelimeler: *Android 1, kötücül yazılım tespiti 2, akıllı telefon 3, Android uygulama güvenliği 4, Android güvenliği 5, mobil güvenlik 6*

Android Malware Detection Systems Review

ABSTRACT

With the smartphones entering our lives, the number of smartphones continues to increase day by day. The reason why smartphones are in so demand is that people can easily do what they want. According to IDC's 2016 Q2 report, Android dominated the smartphone market with an 87.6% share [1]. The Android platform has become the number one target of malicious people because of Android has an open source and new application installation has not been analyzed in detail. Therefore, the number of Android malicious applications are also increasing every day on Google Play and alternative Android application markets.

Geliş: 06/12/2016, Düzeltme: 18/01/2017, Kabul: 30/01/2017

According to G Data's 2015 Q1 mobile malware report, 50.3% of malware is for financial purposes [2]. The reason is that 41% of Europe's users use their devices for banking transactions [2]. Hence, there is need for effective malware detection systems which are detect malicious software on Android application markets. In this paper, malicious software detection systems will be explained.

Keywords: *Android 1, malware detection 2, smartphone 3, Android application security 4, Android security 5, mobile security 6*

I. GİRİŞ

AKILLI telefonların hayatımıza girmesiyle birlikte Android marketlerdeki uygulamaların sayısıda her geçen gün ciddi bir şekilde artmaktadır. Cisco firmasının tahminlerine göre 2020 yılına kadar dünya çapındaki cihazların ve internet bağlantıların yaklaşık yüzde 50'sini akıllı telefonların(tabletler dahil) oluşturacağı tahmin edilmektedir [3]. Akıllı telefon uygulamalarının artması ve akıllı telefonların hassas görevler veya ticari işlemler için kullanılması ile gizlilik ve güvenlik birincil bir endişe haline gelmiştir [4]. Akıllı telefon kullanıcıları yükledikleri uygulamaların kötücül ya da iyicil olduğunu bilememektedirler. Mcafee firmasının '2016 için ufukta neler var?' isimli mobil tehdit raporuna göre ilk altı ayda tespit edilen kötü amaçlı yazılım sayısı 37 milyondur [5]. Kötü amaçlı uygulama sayısının bu denli yüksek olması Google Play marketin bir kere ücret istemesi, detaylı bir kötücül yazılım incelemesi yapılmaması ve Android platformunun açık kaynak kodlu olmasından kaynaklandığı düşünülmektedir. Uygulamaların güvenli olup olmadığını anlamak için başka bir uygulama ya da sisteme ihtiyaç vardır.

F-Secure firmasının 2014 1. çeyrek raporuna göre; %99 oranında yeni mobil kötücül yazılım ailesi tespit edilmiştir [6]. Kötücül yazılımlar kişisel bilgileri ele geçirmek, kazanç elde etmek, konum bilgisini kullanmak, ortamı dinlemek, telefon rehberini ele geçirmek gibi farklı hedeflere sahiptirler. Android güvenlik önlemi olarak izin modelini kullanmaktadır ve uygulamalar istedikleri izinlere göre belirli işlevleri yerine getirmektedirler. SMS alma ve gönderme, fotoğraf çekme, rehbere erişme, GPS veya telefon görüşmeleri işlevlerini yapabilmek için gerekli izin tanımlaması yapılması gereklidir. Android'in önceki sürümlerinde kullanıcılar uygulamaları yükledikten sonra izinlere müdahale edemiyorlardı. Eğer kullanıcı uygulamanın o izni kullanmasını istemiyorsa uygulamayı kullanmaya devam edecek ya da uygulamayı telefonundan kaldırması gerekiyordu. Bu durum Android'in 6.0 Marshmallow ile yeni izin modeline geçildiğinden ortadan kalkmıştır. Bu yeni izin modeli Marshmallow ve sonraki sürümlerde kullanılabilecektir ancak önceki Android sürümlerini kullanan kullanıcılar için değişen bir şey olmayacaktır. Android bu yeni izin modelinde bazı izinleri tehlikeli izin olarak tanımlamıştır ve bunlar Tablo 1'de gösterilmiştir.

Tablo 1. *Tehlikeli İzin ve İzin Grupları [7]*

İZİN GRUBU	**İZİNLER**
TAKVİM	• READ_CALENDAR • WRITE_CALENDAR
KAMERA	• CAMERA
REHBER	• READ_CONTACTS • WRITE_CONTACTS • GET_ACCOUNTS
KONUM	• ACCESS_FINE_LOCATION • ACCESS_COARSE_LOCATION

MİKROFON	• RECORD_AUDIO
TELEFON	• READ_PHONE_STATE • CALL_PHONE • READ_CALL_LOG • WRITE_CALL_LOG • ADD_VOICEMAIL • USE_SIP • PROCESS_OUTGOING_CALLS
SENSÖRLER	• BODY_SENSORS
İZİN GRUBU	**İZİNLER**
SMS	• SEND_SMS • RECEIVE_SMS • READ_SMS • RECEIVE_WAP_PUSH • RECEIVE_MMS
DEPOLAMA	• READ_EXTERNAL_STORAGE • WRITE_EXTERNAL_STORAGE

Tablo 1 incelendiği zaman takvim, kamera, rehber, konum, mikrofon, telefon, sensörler, sms ve depolama ile ilgili izinler Google tarafından tehlikeli izinler olarak tanımlanmıştır. Android Marshmallow sürümü ve sonraki sürümlerde Tablo 1'de tanımlanan izinleri kullanan uygulamalar bu izinleri kullanacakları zaman kullanıcılardan bu izni kullanmak için onay alması gerekmektedir. Ayrıca Android Marshmallow sürümü ile kullanıcılar uygulamaların kullanmasını istemedikleri izinleri iptal edebileceklerdir.

Android mobil güvenliği ile ilgili birçok araştırma yapılmıştır ve yapılmaya da devam edilmektedir. Bu makalede yapılan çalışmalar hakkında bilgiler verilmiştir. Konular makalede şu sıraya göre ele alınmıştır; 2. bölümde yöntem, 3. bölümde bu yöntemlerin karşılaştırılması, 4. bölümde yapılan çalışmaların karşılaştırılması ve 5. bölümde sonuçlar ve değerlendirmelere yer verilmiştir.

II. YÖNTEM

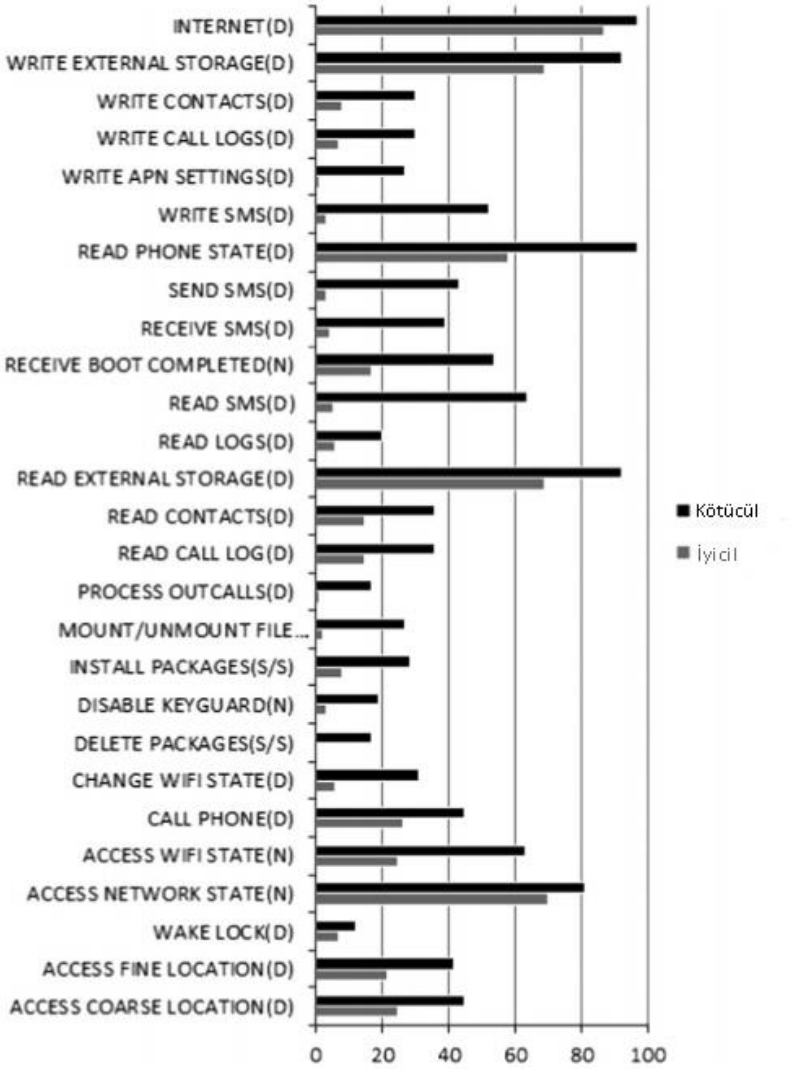
Bu bölümde Android kötücül yazılım tespit yöntemleri ile ilgili literatürde yer alan çalışmalar ele alınmıştır. Kötücül yazılım sistemleri statik, dinamik, hibrid ve Android marketteki uygulamaların bilgilerini kullanılarak yapılan analiz yöntemleri olmak üzere dört başlık altında incelenmiştir.

A. Statik Analiz Yöntemleri I

Android uygulamanın akıllı telefona yüklenmeden önce yapılan analiz yöntemidir. Android uygulamalar akıllı cihazlara apk dosyası ile yüklenmektedir. Apk dosyasının içinde AndroidManifest.xml dosyası ve java kod dosyalarının sıkıştırılmış hali olan dex dosyası bulunmaktadır. Statik analiz yapabilmek için tersine mühendislik yapılması gereklidir. Tersine mühendislik için hazır araçlar ya da her hangi bir sıkıştırma uygulaması kullanılmaktadır. Apk dosyası açıldıktan sonra classes.dex dosyası dex2jar programında dex2jar.bat komutu girilerek jar dosyasına dönüştürülmektedir. Bu oluşturulan jar dosyası jd-gui uygulaması kullanılarak java dosyalarının

içerikleri görüntülenebilir ve kötücül kod parçacıkları bulanabilir. Tersine mühendislik yöntemi her zaman yapılamamaktadır, çünkü tersine mühendislik yapmayı engellemek için kod karıştırma yöntemi kullanılmaktadır. Statik analiz yöntemi geleneksel imza temeli olarak bilinen sistemlerin sınırlamalarından ve gittikçe artan kötücül yazılım çeşitlerinden dolayı kötücül yazılım tespitinde tek başına kullanılması yetersiz kalmaktadır [8]. Feizollah ve diğ. [9] tarafından gerçekleştirilen çalışmaya göre 2015 yılına kadar yapılan 100 çalışmanın 45 tanesi statik analiz yöntemini kullanmıştır. Bunlardan %36'sı izinleri, %29'u java kodlarını ve diğer geri kalan %35 ise AndroidManifest.xml dosyasındaki diğer özellikleri(intent, ağ adresleri, dizeleri ve donanım bileşenleri) kullanarak statik analiz yapmıştır.

Kötücül ve iyicil uygulamalarda sık olarak kullanılan izinlerin bulunma yüzdeleri Şekil 1'de gösterilmiştir.

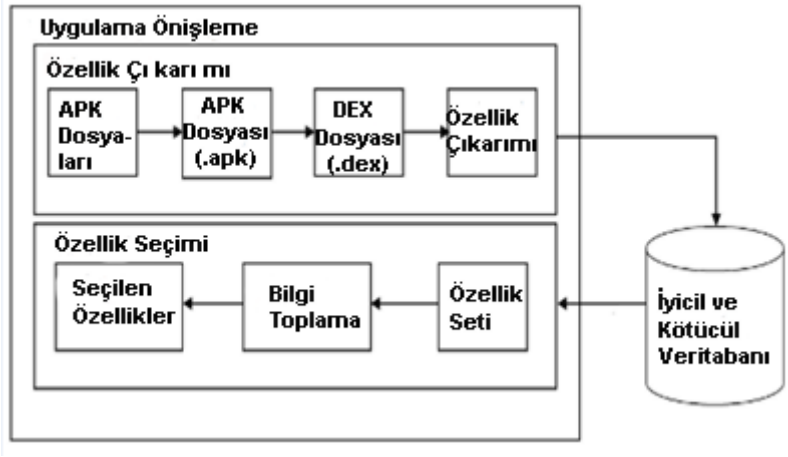


Şekil 1. *Kötücül ve iyi huylu yazılımların kullandıkları izinlerin yüzdeleri [10]*

Şekil 1 incelendiğinde; kötücül yazılım içeren uygulamalarda INTERNET, READ PHONE STATE, WRITE EXTERNAL STORAGE, READ EXTERNAL STORAGE ve ACCESS NETWORK STATE izinlerinin kullanıldığı görülmüştür. İyicil yazılım olan uygulamalarda ise INTERNET, ACCESS

NETWORK STATE, READ EXTERNAL STORAGE ve WRITE EXTERNAL STORAGE izinlerinin kullanıldığı görülmüştür. Uygulamaların kullandıkları izinler genellikle geliştirilen uygulamanın bulunduğu kategoriye göre değişmektedir. Örneğin; gelir elde etmek isteyen kötücül yazılımlarda SEND SMS izni kullanılır çünkü ücretli servislere mesaj atarak gelir elde edilmeye çalışılmaktadır.

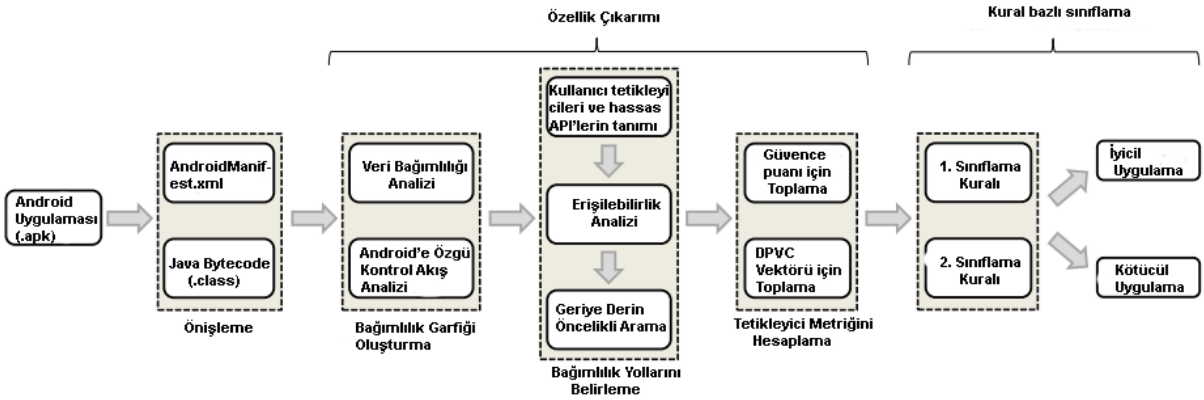
Mayuri Magdum ve diğ. [11] tarafından gerçekleştirilen çalışmada iki aşamadan oluşan bir statik analiz sistemi önerilmiştir. İlk aşamada özellik seçimi, ikinci aşamada ise Lojistik regresyon ve Karar Ağacı algoritması kullanarak sınıflandırmaya dayalı bir sistem geliştirilmiştir ve bu sistemin tüm prosedürleri Şekil 2'de gösterilmiştir.



Şekil 2. *Mayuri Magdum ve arkadaşlarının geliştirmiş oldukları sistem [11]*

Şekil 2 incelendiğinde; geliştirilen sistem apk dosyasını kullanarak özellikleri çıkarmaktadır ve çıkarılan özellikler veritabanına kayıt edilmektedir. Veritabanına kayıt edilen özellik setleri kullanılarak bilgi elde edilmektedir. Elde edilen bilgilere göre uygulamanın zararlı olup olmadığının tespiti yapılmaktadır. Geliştirilen sistem, zararlı ve zararsız yazılımları eğitmek için uygulamaları eğitim sistemine göndermektedir. Eğitim aşaması bittikten sonra analiz sonuçları imza veritabanına kaydedilmektedir.

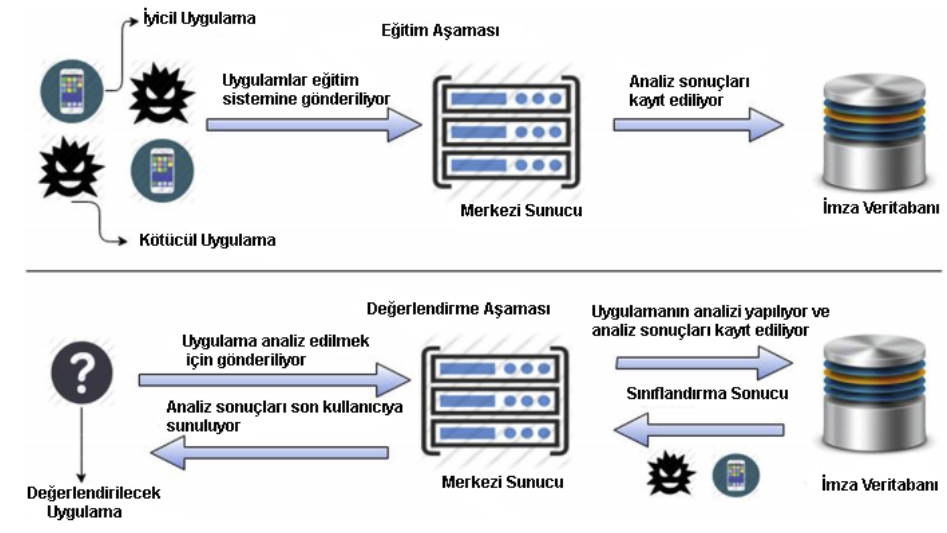
Elish ve diğ. [12] tarafından gerçekleştirilen çalışmada kötü amaçlı Android uygulamaları tespit eden bir sınıflandırma yaklaşımı geliştirilmiştir. Geliştirilen sistem kötü niyetli kalıpları saptamak yerine meşru ve arzu edilen davranış kalıplarını saptamaktadır. Yani kullanılan sınıflama uygulamanın iyi huylu özelliklere sahip olup olmadığını göstermektedir ve geliştirilen yaklaşımın iş akışı Şekil 3'de gösterilmiştir.



Şekil 3. *Elish ve arkadaşlarının geliştirmiş oldukları sistemin iş akışı [12]*

Şekil 3 incelendiğinde; Soot aracı kullanılarak apk dosyasından AndroidManifest.xml ve Java Bytecode elde edilmektedir. Veri bağımlılığı grafiği oluşturulması için veri akış analizi kullanılmaktadır. Veri bağımlılığı grafiği oluşturulduktan sonra kullanıcı tetikleyici ve hassas API çağrı çiftleri arasındaki yollar belirlenmektedir. Bağımlılık yollarını bulmak ve her işlem için bir TriggerMetric hesaplamak için geriye derin öncelikli arama (Backward Depth-First Search) yapılmaktadır. Sınıflandırma kararları güvence puanı ve geçerli çağrı yüzdelerinin dağılımı vektörüne (Distribution of the Percentages of Valid Call) göre verilmektedir. Sınıflandırma kararları doğrultusunda uygulamanın iyicil ya da kötücül olduğunun sonucu verilmektedir.

Kabakuş ve diğ. [13] tarafından gerçekleştirilen çalışmada izin tabanlı Android kötücül yazılım tespiti yapan APK Auditor isminde bir sistem geliştirilmiştir. APK Auditor, Android uygulamaları iyicil veya kötücül olarak karakterize etmek ve sınıflandırmak için statik analiz yöntemini kullanmıştır ve geliştirilen sistemin mimarisi Şekil 4'de gösterilmiştir.



Şekil 4. *APK Auditor sistem mimarisi [13]*

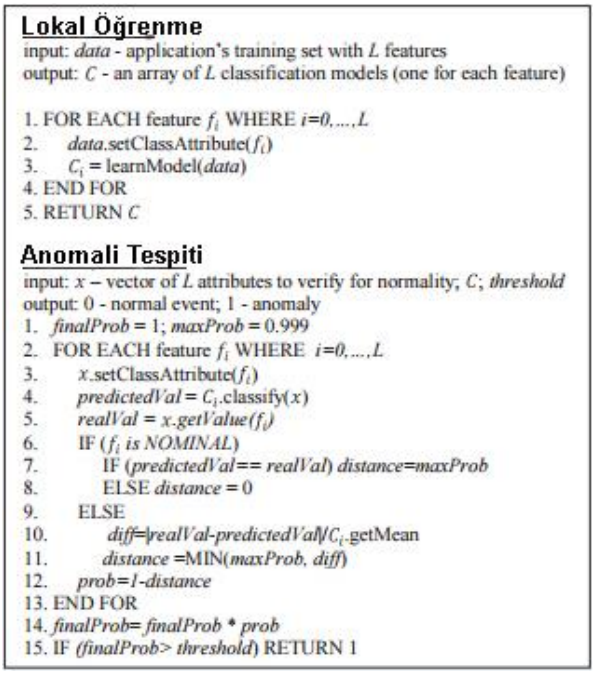
Şekil 4 incelendiğinde; APK Auditor; Android istemcisi, imza veritabanı ve merkezi sunucu olmak üzere üç ana bileşenden oluşmaktadır. Android istemcisinin görevi sisteme uygulama yüklemek ve

web servisi aracılığıyla merkezi sunucu ile iletişime geçmektir. Web servisinden alınan analiz sonuçları Android telefon veya web tabanlı portal vasıtasıyla kullanıcılara sunulmaktadır. İmza veritabanı, analiz edilen Android uygulamaların sonuçlarını saklayan bir ilişkisel veritabanı yönetim sistemidir. Merkezi sunucunun görevi Android istemcisi ve imza veritabanı ile iletişim kurmak ve analiz sürecini yürütmektir.

B. Dinamik Analiz Yöntemleri II

Android uygulamanın cihaza yüklendikten sonra kötücül yazılım tespitinin yapıldığı analiz yöntemidir. Dinamik analiz yöntemi uygulamaların mobil cihazlara yüklendikten sonraki davranışlarını incelemektedir ve bunlar genellikle uygulamanın işletim sisteminde veya ağdaki davranışlarını içermektedir [9]. Yapılan araştırmalarda genellikle dinamik analiz yöntemleri olarak uygulamaların kullandıkları sistem çağrıları ve uygulamaların kullandıkları ağ trafikleri kullanılmıştır [14]. Feizollah ve diğ. [9] tarafından gerçekleştirilen çalışmaya göre 2015 yılına kadar yapılan 100 çalışmanın 42 tanesinde dinamik analiz yöntemini kullanmıştır. Bunlardan 22 tanesi sistem çağrılarını, 10 tanesi ağ trafiğini ve diğer kalan 10 tanesi diğer özellikleri kullanmıştır.

Shabtai ve diğ. [8] tarafından gerçekleştirilen çalışmada uygulamaların ağ desenlerini çıkartılarak kötücül yazılımların tespit edilebileceği dinamik analiz yöntemi geliştirilmiştir. Geliştirilen sistemin genel mimarisi Android cihaza kurulu bir istemci bileşeni ve bir sunucu bileşeninden oluşmaktadır. İstemci bileşeninin sorumluluğu; cihaza önceden yüklenmiş ve çalışan uygulamaları izlemek, kullanıcıya özgü yerel modelleri öğrenmek ve gözlemlenen "normal" davranışları algılamaktır. İşbirlikçi öğrenim sunucusu (Collaborative Learning Server) çeşitli mobil cihazlar tarafından rapor edilen verileri toplamaktan ve her uygulama için kullanıcıların ortak trafik modellerini temsil eden modelleri oluşturmaktan sorumludur. Çalışmada lokal öğrenme kullanılmıştır. Lokal öğrenmenin ve Anomali tespitinin sözde kodu Şekil 5'de gösterilmiştir.



Lokal Öğrenme

input: *data* - application's training set with L features
output: C - an array of L classification models (one for each feature)

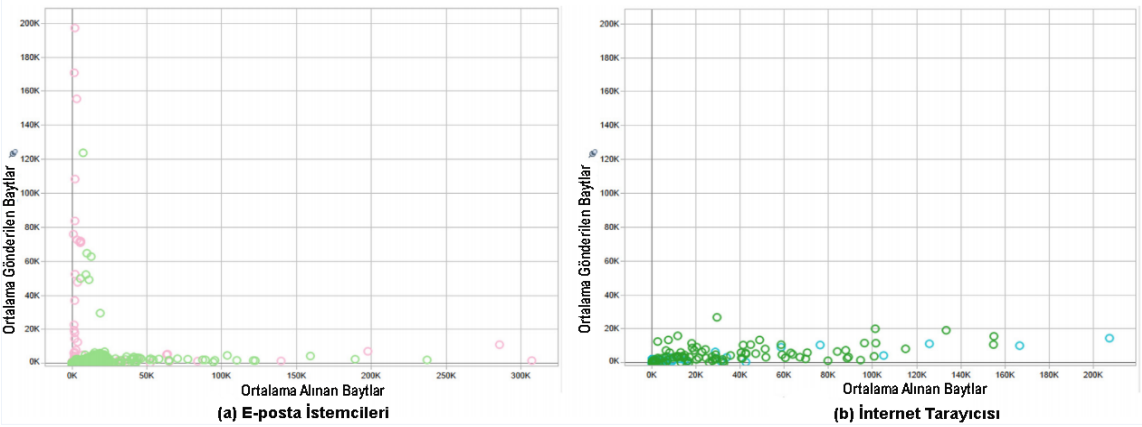
1. FOR EACH feature f_i WHERE $i=0,\ldots,L$
2. *data*.setClassAttribute(f_i)
3. C_i = learnModel(*data*)
4. END FOR
5. RETURN C

Anomali Tespiti

input: x – vector of L attributes to verify for normality; C; *threshold*
output: 0 - normal event; 1 - anomaly
1. *finalProb* = 1; *maxProb* = 0.999
2. FOR EACH feature f_i WHERE $i=0,\ldots,L$
3. x.setClassAttribute(f_i)
4. *predictedVal* = C_i.classify(x)
5. *realVal* = x.*getValue*(f_i)
6. IF (f_i *is NOMINAL*)
7. IF (*predictedVal*== *realVal*) *distance*=*maxProb*
8. ELSE *distance* = 0
9. ELSE
10. *diff*=|*realVal*-*predictedVal*|/C_i.getMean
11. *distance* =MIN(*maxProb*, *diff*)
12. *prob*=*1*-*distance*
13. END FOR
14. *finalProb*= *finalProb* * *prob*
15. IF (*finalProb*> *threshold*) RETURN 1

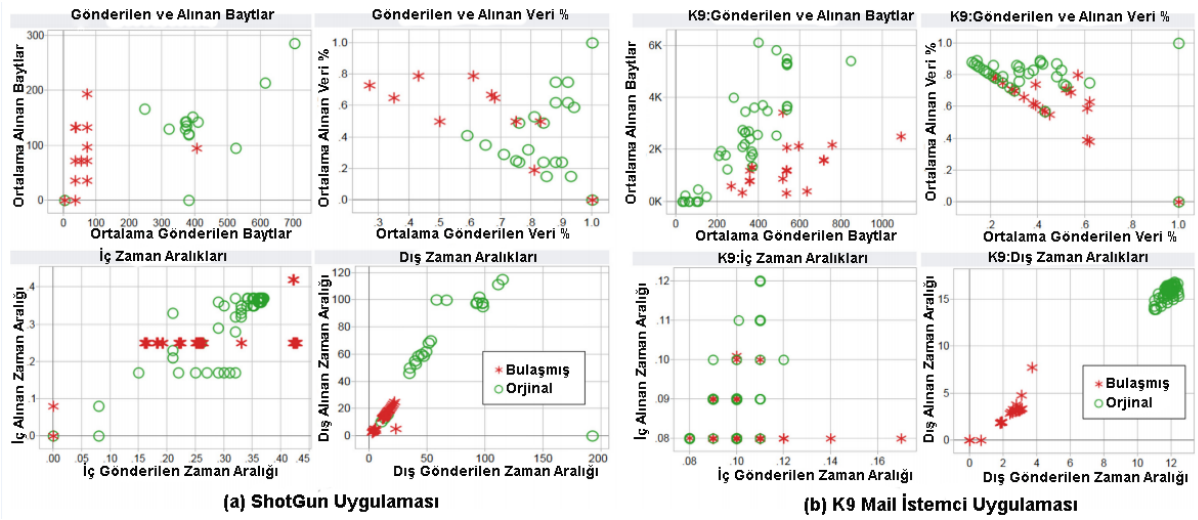
Şekil 5. *Lokal öğrenmenin ve Anomali tespitinin sözde kodu [8]*

Şekil 5 incelendiğinde lokal öğrenme olarak karar ağaç (Decision Tree) algoritması kullanılmıştır. Anomali tespitinde uygulamaların çevrimiçi ağ davranışlarının ve normal desen sapmalarının tespiti yapılmaktadır. Yapılan çalışmada aynı tipteki farklı uygulamaların benzer internet ağ trafik desenine sahip olduğu savunulmuştur. Buna göre Şekil 6 (a)' da gmail ve Android'in kendi email uygulamalarının internet ağ trafiği, Şekil 6 (b)' de ise mozilla firefox ve Android'in kendi internet tarayıcı uygulamasının internet ağ trafiği gösterilmiştir ve farklı uygulamalar farklı renkte gösterilmiştir [8].



Şekil 6. *Aynı tip ama farklı uygulamaların internet ağ trafik deseni (farklı uygulamalar farklı renkte gösterilmiştir) [8]*

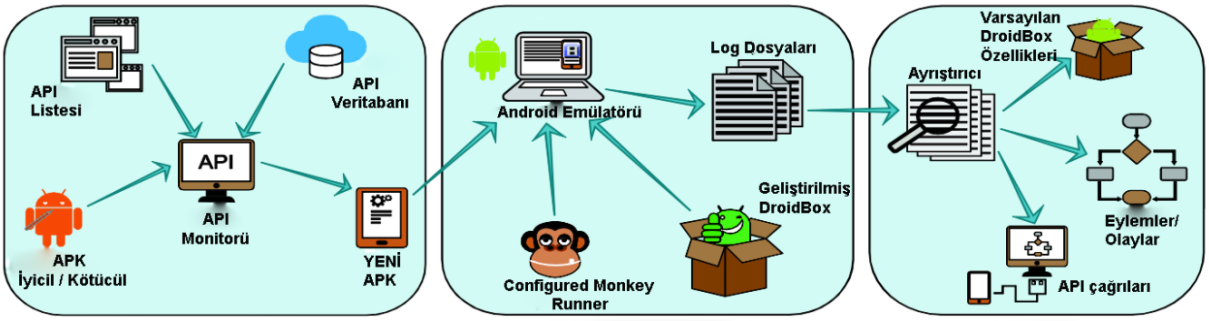
Şekil 6 incelendiği zaman farklı ama aynı işleri yapan uygulamaların ağ davranış trafiklerinin benzer oldukları görülmektedir. Şekil 7 (a)' da ShotGun uygulamasının Geinimi Truva virüsü bulaşmış ve bulaşmamış internet ağ trafik desenleri ve Şekil 7 (b)' de K-9 Mail uygulamasının kendini güncelleyen kötücül yazılım bulaşmış ve bulaşmamış internet ağ trafik desenleri gösterilmiştir [8].



Şekil 7. *ShotGun ve K-9 Mail istemci uygulamalarının farklı versiyonlarındaki kötücül yazılım bulaşmış ve bulaşmamış hallerinin internet ağ trafik desenleri [8]*

Şekil 7 incelendiğinde, ShotGun ve K-9 mail uygulamalarının farklı versiyonlarındaki ağ trafik desenleri farklı olduğu görülmektedir ve bu farklılık da bu uygulamaların kötücül olduğunu göstermektedir.

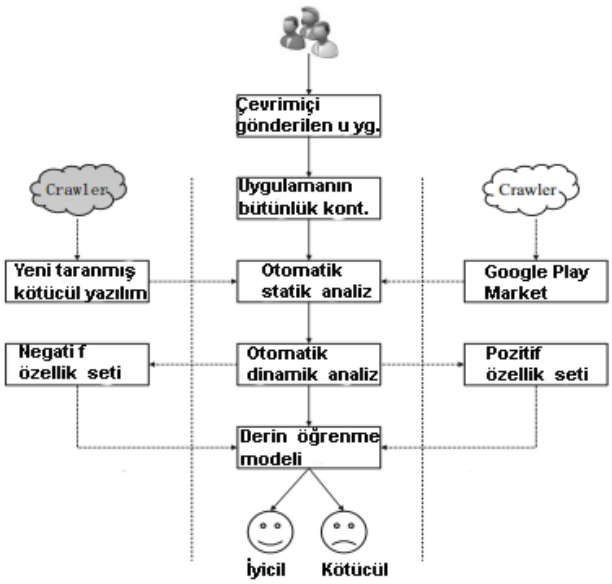
Mohammed K. Alzaylaee ve diğ. [15] tarafından gerçekleştirilen çalışmada DynaLog isminde dinamik analiz ile Android uygulamaların kötücül olup olmadığının tespitini yapan bir yapı geliştirilmiştir ve geliştirilen yapının mimarisi Şekil 8'de gösterilmiştir.



Şekil 8. *DynaLog mimarisi [15]*

DynaLog birçok bileşenden oluşmaktadır. Android uygulamaları çalıştırmak ve test etmek için DroidBox altyapısı kullanmaktadır. API çağrılarını izleyebilmek, loglamak ve çıkarmak için APK enstrümantasyon modülünü kullanmaktadır. Kod kapsamını iyileştirmek için bir uygulama içinde mevcut olan tüm aktiviteleri ve servisleri çağırma yeteneğine ve log ayrıştırma ve işleme özelliklerine sahiptir.

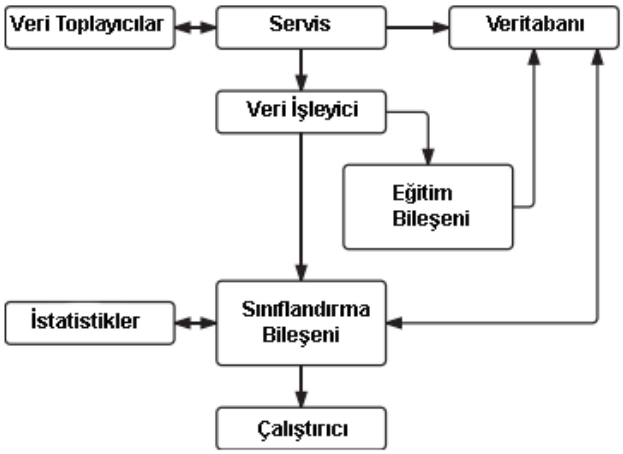
Zhenlong Yuan ve diğ. [16] tarafından gerçekleştirilen çalışmada otomatik olarak uygulamanın kötücül olup olmadığının tespitini yapan çevrimiçi derin öğrenme tabanlı DroidDetector isminde bir sistem geliştirilmiştir ve geliştirilen sistemin mimarisi Şekil 9'da gösterilmiştir.



Şekil 9. *DroidDetector mimarisi [16]*

DroidDetector ile apk dosyası sisteme gönderildiğinde; DroidDetector uygulamanın bütünlüğünü, doğruluğunu ve meşru bir Android uygulaması olup olmadığını kontrol etmektedir. DroidDetector uygulamanın kullandığı izinleri ve hassas API'leri elde etmek için statik analiz yöntemini kullanmaktadır. Ayrıca DroidDetector, DroidBox sandbox'ı kullanarak test edilmek istenilen uygulamayı yükler ve belirli bir süre boyunca uygulamayı çalıştırarak dinamik analiz yapmaktadır. Son olarak derin öğrenme modeli kullanılarak uygulamanın kötücül olup olmadığının tespiti yapılmaktadır.

Gheorghe ve diğ. [17] tarafından gerçekleştirilen çalışmada iyi bilinen ve sıfır gün zararlı yazılımları tespit etmek için ADMS adı verilen davranış temelli dinamik bir sistem sunulmuştur ve geliştirilen ADMS sisteminin mimarisi Şekil 10'da gösterilmiştir.



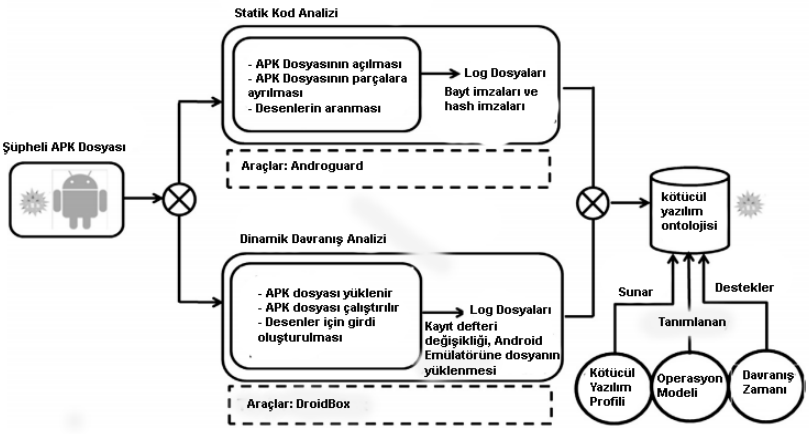
Şekil 10. *ADMS mimarisi [17]*

Şekil 10 incelendiğinde; ADMS mimarisi veri toplayıcıları, servis, veri işleyici, eğitim, sınıflandırma, istatistik çıkarıcı ve çalıştırıcı temel bileşenlerinden oluşmaktadır. Veri toplayıcılarının görevi belirli bir uygulama ile ilgili davranışsal verileri toplamaktır. Hizmet bileşeni, diğer bileşenler arasında merkezi bir yönetim ve iletişim noktası görevi görmektedir. Veritabanı bileşeninin görevi davranışsal verileri, istatistikleri ve sonuçları depolamaktır. Veri işlemcisi bileşeninin görevi davranışsal verileri veritabanından almak ve toplamaktır. Eğitim bileşeninin görevi uygulama davranışını öğrenmek için bir araya toplanmış verileri işlemektir. Sınıflandırma bileşeninin görevi bir uygulamanın kötü amaçlı olup olmadığına karar vermek için bir araya toplanmış verileri işlemektir.

C. Hibrid Analiz Yöntemleri III

Statik ve dinamik analiz yöntemlerinin aynı anda kullanıldığı kötücül yazılım tespit yöntemidir. Feizollah ve diğ. [9] tarafından gerçekleştirilen çalışmada 2015 yılına kadar yapılan 100 çalışmanın 10 tanesi hibrid analiz yöntemini kullanmıştır.

Wang ve diğ. [18] tarafından gerçekleştirilen çalışmada statik ve dinamik analiz yöntemlerini kullanan kötücül yazılım tespit sistemi geliştirilmiştir ve geliştirilen sistemin çalışma mimarisi Şekil 11'de gösterilmiştir.



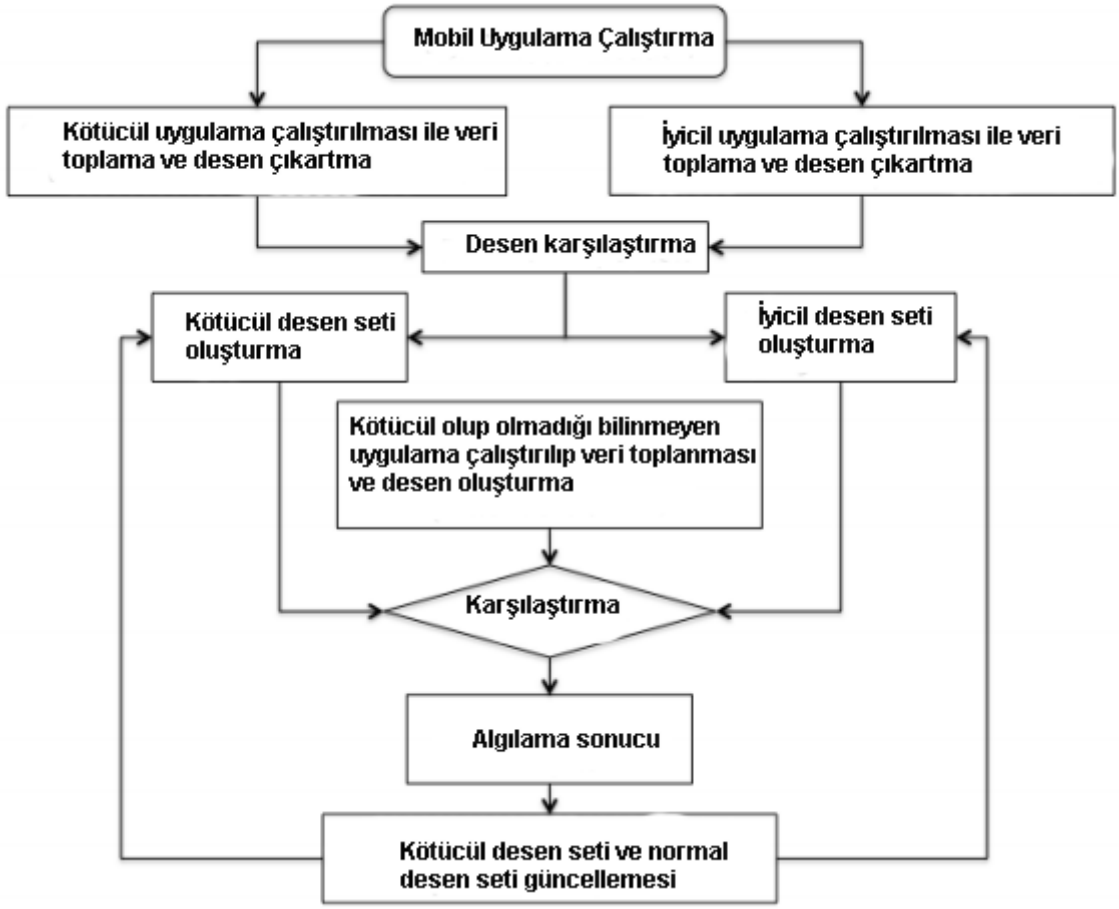
Şekil 11. *Statik ve dinamik analizini kullanan kötücül yazılım tespit sistemi [18]*

Şekil 11 incelendiğinde; statik kod analizi yapmak için Androguard aracı kullanılmaktadır. Statik analiz ile özel kaynak kodları ve binary stringler incelenmektedir. Karşılaştırma sonuçları log dosyasında farklılıkların karşılaştırılması için saklanmaktadır. DroidBox sandbox aracı kullanılarak dinamik davranış analizi yapılmaktadır. Dinamik analiz ile uygulamanın çalışma sırasındaki ağ, dosya ve disk davranışları izlenilmektedir.

Blasing ve diğ. [19] tarafından gerçekleştirilen çalışmada AASandbox adında statik ve dinamik analiz yapan bir sistem geliştirilmiştir. AASandbox statik analiz yöntemi ile apk dosyasında bulunan izinleri ve java kodu incelenmektedir. Daha sonra apk dosyası akıllı telefona yüklenir ve sistem çağrı logları dinamik analiz yöntemi ile elde edilmektedir. AASandbox sisteminin bulut servis desteği bulunmaktadır, yani; Android Market uygulaması gibi kullanıcılar istedikleri uygulamaları yükleyip analiz yapabilmektedir.

Wei ve diğ. [20] tarafından gerçekleştirilen çalışmada ProfileDroid adında bir hibrid analiz sistemi geliştirilmiştir. ProfileDroid statik, kullanıcı etkileşimi, işletim sistemi ve ağ olmak üzere 4 farklı katmandan oluşmaktadır. Her bir katman görüntüleme ve profil oluşturma olmak üzere iki bölümden oluşmaktadır. Her katman için izleme bileşeni uygulamanın çalıştığı Android cihazında çalışmaktadır. Elde edilen bilgi bağlı bilgisayarda çalışan profil oluşturma bölümünden beslenmektedir. Statik katmanında, apktool aracı kullanılarak apk dosyasında bulunan AndroidManifest.xml ve java kodları incelenmektedir. Kullanıcı etkileşimi katmanında, kullanıcının oluşturmuş olduğu eylemlere odaklanılmıştır. Bu katmanda verileri toplamak için adb de bulunan logcat ve getevent araçları kullanılmaktadır. İşletim sistemi katmanında, uygulamaların oluşturdukları sistem çağrıları toplanılmaktadır. Ağ katmanında, veri paketleri ağ trafiğini analiz etmek için loglanmaktadır.

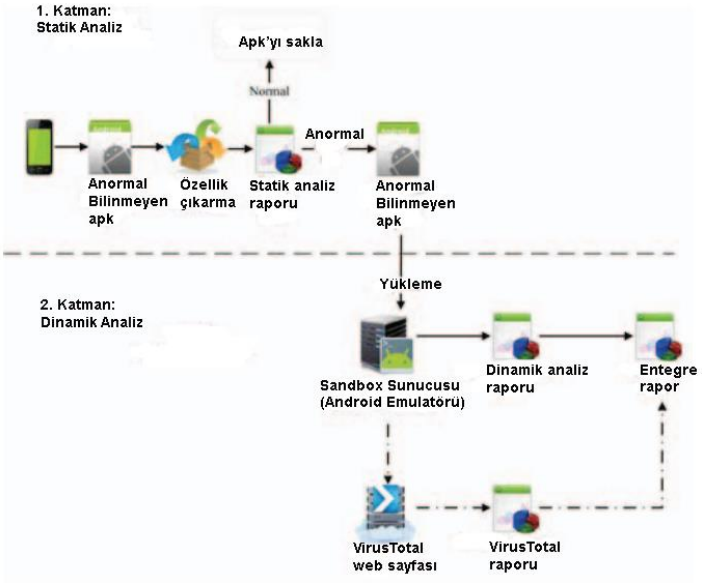
Fei Tong ve diğ. [21] tarafından gerçekleştirilen çalışmada mobil kötücül yazılım tespiti için dinamik analizi ve statik analizi benimseyen yeni bir hibrid yaklaşım sistemi önerilmiştir ve önerilen sistemin akış diyagramı Şekil 12' de gösterilmiştir.



Şekil 12. *Fei Tong ve arkadaşlarının önerdiği akış diyagramı [21]*

Fei Tong ve diğ. [21] tarafından gerçekleştirilen çalışmada kötücül ve iyicil uygulamaların çalışma zamanında oluşturduğu sistem çağrılarını toplamak için dinamik analiz yöntemi kullanılmıştır. Ayrı sistem çağrıları ve farklı çağrı derinliğindeki ardışık sistem çağrıları hakkında toplanan verilerden desenler çıkarılmaktadır. Kötü niyetli ve iyi huylu uygulamaların sahip olduğu kalıplar karşılaştırılmaktadır. Ayrı sistem çağrıları ve farklı çağrı derinliğindeki ardışık sistem çağrıları kötü amaçlı, anormal veya normal olabilecek davranışları yansıtmaktadır. İyicil veya kötücül olduğu bilinmeyen bir uygulamayı tespit etmek için uygulamanın çalışma zamanındaki ayrı sistem çağrıları ve farklı çağrı derinliğindeki ardışık sistem çağrıları toplanmaktadır. Toplanan sistem çağrıları kullanılarak hedef kalıplar çıkarılır ve normal desenle ve kötü niyetli desenle kıyaslama yapılarak uygulamanın iyicil veya kötücül olduğu değerlendirilmektedir.

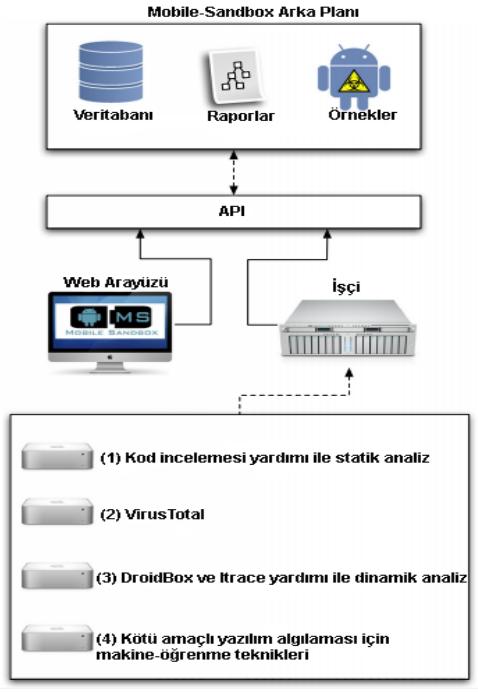
Ming-Yang Su ve diğ. [22] tarafından gerçekleştirilen çalışmada kötü amaçlı yazılım tespit sistemi için bir hibrid yaklaşımı geliştirilmiştir ve Şekil 13' de geliştirilen sistemin mimarisi gösterilmiştir.



Şekil 13. *Ming-Yang Su ve arkadaşlarının önerdiği sistem mimarisi [22]*

Şekil 13'e göre önerilen sistem; ilk önce statik analiz yöntemi ile uygulamaların kullandığı izinler, native izinler, intentler ve çağrı fonksiyonları elde edilmektedir. Statik analiz sonucunda uygulamada şüpheli bir durum tespit edilirse dinamik analiz yapılmak için sandbox sunucuya gönderilmektedir. Geliştirilen sistemin dinamik analiz yapmak için sandbox sunucuya ve uygulamaları çalıştırmak için izole edilmiş Android sanal makinesine ihtiyacı vardır.

Spreitzenbarth ve diğ. [23] tarafından gerçekleştirilen çalışmada Android uygulamaları otomatik olarak analiz edecek şekilde tasarlanmış bir sistem olan Mobile-Sandbox geliştirilmiştir ve geliştirilen Mobile-Sandbox'ın bileşenleri Şekil 14'de gösterilmiştir.



Şekil 14. *Mobile-Sandbox bileşenleri [23]*

Mobile-Sandbox statik analiz yapmak için bir kaç farklı modeli kullanmaktadır. İlk olarak analiz edilen uygulamanın md5 hash kodunu VirusTotal veritabanındaki tüm hash'ler ile karşılaştırılır ve eğer hash bulunursa uygulamayı kötü niyetli olarak sınıflandıran araçların sayısı uygulamayı analiz eden anti-virüs araçlarının sayısına bölünerek "bulunma oranı" hesaplanmaktadır. İkincil olarak unzip aracı ile uygulama bileşenleri elde edilmektedir ve classes.dex dosyasında bulunan Dalvik bit kodu daha iyi ayırt etmeyi sağlamak için smali'ye çevrilmektedir. Potansiyel olarak tehlikeli işlevlerin ve yöntemlerin bulunması için smali kodu araştırılmaktadır. Mobile-Sandbox dinamik analiz için altyapı olarak TaintDroid ve DroidBox'u kullanmaktadır. Makine öğrenmesi olarak kötü niyetli ve iyi huylu örnekleri içeren geniş bir veri kümesi üzerinde bir sınıflandırma modelini öğrenmek için doğrusal destek vektörü kullanılmaktadır.

D. Android Markette Bulunan Bilgiler Kullanılarak Yapılan Analiz Yöntemleri IV

Bu analiz yönteminde Android marketlerde bulunan geliştiriciler ve uygulamalar ile ilgili bilgiler kullanılarak kötücül yazılım tespiti yapılmaktadır. Android marketlerde bulunan bilgilere uygulama açıklaması, uygulamanın istediği izinler, uygulamanın kategorisi, en son güncelleme tarihi, verilen oylar ve değerlendirmeler örnek olarak gösterilebilir [9]. Kötücül yazılım tespiti üzerine yapılan araştırmaların çok azında bu yöntem kullanılmıştır.

WHYPER [24] çalışması bu kötücül yazılım tespit sistemine örnek olarak gösterilebilir. WHYPER çalışması doğal dil işleme (Natural Language Processing) tekniğini kullanarak uygulamaların neden bu izinleri(adres defteri, takvim ve ses kaydı izinleri) kullandığını açıklamaktadır. Özellikle, bu bilgileri almak için uygulama açıklamalarını kullanmaktadır. Bu yüzden WHYPER iskeleti uygulama pazarı ve son kullanıcılar arasında çalışmaktadır.

MAST [25] çalışması Android markette bulunan bilgileri kullanarak kötücül yazılım tespiti yapan bir çalışmadır. MAST çalışmasında kötücül yazılım tespit işlemi dört aşamadan oluşmaktadır. Birinci aşamada ilgi çekici güvenlik özelliklerini tanımlayan uygulama öznitelikleri tanımlanmaktadır. Özellik tanımlamada markete ait özel bilgi olan kategoriler, kullanıcı puanları ve açıklamaları yerine izinler, intentler, native kod ve zip dosyaları kullanılmaktadır. İkinci aşamada Çoklu Uyum Analizi (Multiple Correspondence Analysis) sorularını oluşturmak için ilgili özellik kümeleri birleştirilmektedir. Üçüncü aşamada şüpheli davranışın göstergelerini üretmek için birden çok anket üzerinde Çoklu Uyum Analizi çalıştırılmaktadır. Dördüncü aşamada şüpheli uygulamaların doğru bir MAST sıralamasını oluşturmak için bu göstergeler birleştirilmektedir.

Teufl ve diğ. [26] tarafından gerçekleştirilen çalışmada Android Market gibi akıllı telefon uygulama mağazalarında bulunan uygulama metaverisi kullanılarak kötü amaçlı yazılım tespiti yapan bir sistem geliştirilmiştir. Kullanıcıların uygulama yüklerken görebilecekleri verilere uygulamanın açıklaması, kullandığı izinler, oylamalar veya geliştirici ile ilgili bilgiler örnek olarak verilebilir. Bu verileri analiz etmek için karmaşık bilgi keşif süreçleri ve zayıf istatistiksel yöntemler kullanılmaktadır.

III. BULGULAR ve TARTIŞMA

Yukarıda verilen analiz yöntemlerinin birbirlerine göre avantajlı ve dezavantajlı yönleri bulunmaktadır. Bu dört analiz yönteminin avantajlı ve dezavantajlı yönlerinin karşılaştırılması Tablo 2 de gösterilmiştir.

Tablo 2. *Analiz Yöntemlerinin Avantajları ve Dezavantajları [11]*

Analiz Yöntemleri	**Avantajları**	**Dezavantajları**
Statik Analiz	Hızlıdır ve pahalı değildir Ayıklamak kolaydır	Kod karıştırma(Obfuscation)
Dinamik Analiz	Statik analizden daha kapsamlıdır	Kod kapsama(Code coverage) Ayıklamak zordur Rootlu cihazlara ihtiyaç vardır
Hibrid Analiz	En kapsamlı analizdir	Karmaşıktır Dinamik ve Statik özelliklere (feature) sahip olmalıdır
Android Market Bilgileri Kullanarak Yapılan Analiz	Hızlıdır ve pahalı değildir Ayıklamak kolaydır	Araştırmalar da çok sık kullanılmamaktadır

Tablo 2 incelendiğinde; statik analizde ayıklamak kolaydır ama kod karıştırmaya (obfuscation) karşı çözüm yoktur. Dinamik analiz statik analize göre daha kapsamlıdır ama ayıklamak zordur. Hibrid analiz en kapsamlısıdır ama ayıklama işlemi karmaşıktır. Android market bilgileri kullanılarak yapılan analizde ayıklamak kolaydır ama araştırmacılar tarafından fazla kullanılmamaktadır.

Android kötücül yazılımlar üzerine yapılan araştırmaların karşılaştırılması Tablo 3'de gösterilmiştir.

Tablo 3. *Yapılan Çalışmaların Karşılaştırılması*

	Statik Analiz	**Dinamik Analiz**	**Hibrid Analiz**	**Android Market Bilgilerini Kullanan Analiz**	**Makine Öğrenmesi**	**Veriseti Sayısı**
Shabtai ve diğ.	YOK	VAR	YOK	YOK	VAR	500.000 uygulama
MCDF	VAR	YOK	YOK	YOK	VAR	1200'den fazla kötücül uygulama (Genome Project)
Mayuri Magdum ve diğ.	VAR	YOK	YOK	YOK	VAR	110 kötücül uygulama 75 iyicil uygulama
Elish ve diğ.	VAR	YOK	YOK	YOK	YOK	1433 kötücül uygulama 2684 iyicil uygulama
APK Auditor	VAR	YOK	YOK	YOK	VAR	1853 iyicil uygulama 6909 kötücül uygulama
Choi ve diğ.	VAR	YOK	YOK	VAR	VAR	1200 kötücül uygulama

	Statik Analiz	Dinamik Analiz	Hibrid Analiz	Android Market Bilgilerini Kullanan Analiz	Makine Öğrenmesi	Veriseti Sayısı
DynaLog	YOK	VAR	VAR	YOK	YOK	1 000 iyicil uygulama 1 226 kötücül Uygulama
DroidDetector	VAR	VAR	VAR	YOK	VAR	20 000 uygulama 1 760 kötücül uygulama
ADMS	VAR	VAR	YOK	YOK	VAR	400 iyicil uygulama 400 kötücül uygulama
Wang ve diğ.	VAR	VAR	VAR	YOK	VAR	60 Kötücül Uygulama, 20 İyicil Uygulama
AASandbox	VAR	VAR	VAR	YOK	YOK	150 uygulama
Fei Tong ve diğ.	VAR	VAR	VAR	YOK	YOK	2 000 uygulama 1 200 kötücül uygulama
Ming-Yang Su ve diğ.	VAR	VAR	VAR	YOK	VAR	70 kötücül uygulama 30 iyicil uygulama
Mobile-Sandbox	VAR	VAR	VAR	YOK	VAR	6 100 kötücül uygulama 69 233 uygulama
Whyper	YOK	YOK	YOK	VAR	VAR	581 uygulama
Mast	YOK	YOK	YOK	VAR	YOK	732 kötücül uygulama 14 888 uygulama
Teufl ve diğ.	YOK	YOK	YOK	VAR	VAR	440 000 uygulama

Tablo 3 incelendiğinde; 17 çalışmadan sadece 5 tanesi makine öğrenmesini kullanmamıştır. Ayrıca yapılan çalışmaların kullanmış oldukları iyicil ve kötücül verisetlerinin sayısı arasında oransal bir ilişki görülmemiştir.

IV. SONUÇ

Akıllı telefonların hayatımıza girmesiyle birlikte akıllı telefonlarda bulunan değerli bilgiler kötücül yazılım geliştiricilerinin hedefi haline gelmiştir. Android platformunun diğer mobil işletim sistemlerine göre pazar payının oldukça fazla olması ve açık kaynaklı olmasından dolayı kötücül yazılım geliştiricilerin ana hedefi haline gelmiştir. Bu yüzden Google Play'e her geçen gün farklı özellikleri ve davranışları olan kötücül yazılımları barındıran uygulamalar yüklenmektedir. Android bunların önüne geçmek için son sürümü olan Marshmallow ile yeni izin modeline geçmiştir. Bu izin modelinde, uygulamalar Android'in tehlikeli olarak tanımladığı izinleri çalışma zamanında kullanacağı zaman kullanıcıdan onay alması gerekmektedir. Fakat bu izin modeli Marshmallow ve sonraki sürümlerde kullanılabileceği için Android'in önceki sürümleri için değişen herhangi bir şey yoktur. Bu yüzden Android platformunda bu kötücül yazılımların tespiti için kötücül yazılım tespit sistemlerine ihtiyaç vardır. Literatürde statik, dinamik, hibrid ve Google market bilgileri kullanarak analiz yapan dört adet kötücül yazılım tespit sistemi vardır. Statik analizde, uygulama yüklenmeden önce apk dosyasının incelenmesi ile kötücül yazılım tespiti yapılmaktadır. Dinamik analizde, uygulama yüklendikten sonra uygulamanın davranışlarının incelenmesi ile kötücül yazılım tespitini yapılmaktadır. Hibrid analizde, hem statik hem de dinamik kötücül yazılım tespiti sistemleri birlikte kullanılarak kötücül yazılım tespiti yapılmaktadır. Google marketde bulunan bilgiler kullanılarak yapılan analizde, market de bulunan uygulamalar ile ilgili izinler, açıklamalar, oylar, değerlendirmeler ve benzeri bilgiler kullanılarak kötücül yazılım tespiti sistemi yapılmaktadır. Bahsedilen 4 farklı kötücül yazılım sistemlerinin avantajlı ve dezavantajlı yönleri vardır, fakat hiç bir sistem tam olarak tek başına yeterli değildir.

V. KAYNAKLAR

[1] Anonim, http://www.idc.com/prodserv/smartphone-os-market-share.jsp *(Erişim tarihi: 06 Aralık, 2016).*

[2] Anonim, https://public.gdatasoftware.com/Presse/Publikationen/Malware_ Reports/ G_DATA _MobileMWR_Q1_2015_US.pdf (Erişim tarihi: 06 Aralık, 2016).

[3] Anonim,http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/mobile-white-paper-c11-520862.html *(Erişim tarihi: 06 Aralık, 2016).*

[4] B. Rashidi, C. Fung, E. Bertino, *Comput. & Secur.* **65** (2017) 90-107.

[5] Anonim,http://www.mcafee.com/us/resources/reports/rp-mobile-threat-report-2016.pdf (Erişim tarihi: 20 Ocak 2017)

[6] Anonim, https://www.fsecure.com/documents/996508/1030743/Mobile_Threat_Report https://www.fsecure.com/documents/996508/1030743/Mobile_Threat_Report_Q1_2014.pdf *(Erişim tarihi: 06 Aralık, 2016)*

[7] Anonim, http://developer.Android.com/guide/topics/security/permissions.html#normal-dangerous *(Erişim tarihi: 06 Aralık, 2016).*

[8] A. Shabtai, L. Tenenboim-Chekina, D. Mimran, L. Rokach, B. Shapira, Y. Elovici *Comput. & Secur.* **43** (2014) 1-18.

[9] A. Feizollah, N. B. Anuar, R. Salleh, A. W. A. Wahab *Digit. Invest.* **13** (2015) 22-37.

[10] S. Sheen, R. Anitha, V. Natarajan *Neurocomputing* **151** (2015) 905-912.

[11] M. Magdum, S. K. Wagh *Int. J. Comput. Sci. Inf. Secur. (IJCSIS)* **14** (2016).

[12] K. O. Elish, X. Shu, D. D. Yao, B. G. Ryder, X. Jiang, *Comput. & Secur.* **49** (2015) 255–273.

[13] A. T. Kabakuş, İ. A. Dogru, A. Çetin, *Digit. Invest.* **13** (2015) 1-14

[14] J. Choi, W. Sung, C. Choi, P. Kim *Pervasive Mob. Comput.* **24** (2015) 138-149.

[15] M. K. Alzaylaee, S. Y. Yerima, S. Sezer *DynaLog: An automated dynamic analysis framework for characterizing Android applications*, **2016 International Conference on Cyber Security and Protection of Digital Services (Cyber Security 2016)**, Londra-İngiltere, (2016) 1-8.

[16] Z. Yuan, Y. Lu, Y. Xue *Tsinghua Sci. Tech.* **21** (2016) 114-123.

[17] L. Gheorghe, B. Marin, G. Gibson, L. Mogosanu, R. Deaconescu, V.-G. Voiculescu, M. Carabas, *Secur. Comm. Networks* **8** (2015) 4254-4272

[18] P. Wang, Y-S. Wang *J. Comput. Sys. Sci.* **81** (2015) 1012-1026.

[19] T. Blasing, L. Batyuk, A-D. Schmidt, S. A. Camtepe, S. Albayrak *An Android application sandbox system for suspicious software detection*, **5th International Conference on Malicious and Unwanted Software (MALWARE)**, Nancy -Fransa, (2010) 55-62.

[20] X. Wei , L. Gomez, I. Neamtiu, M. Faloutsos *Profiledroid: multi-layer profiling of Android applications*, **18th annual international conference on Mobile computing and networking**, İstanbul -Türkiye, (2012) 137-148.

[21] F. Tong, Z. Yan (2016) **DOI: 10.1016/j.jpdc.2016.10.012.**

[22] M-Y. Su, K-T. Fung, Y-H. Huang, M-Z. Kang, Y-H. Chung *Detection of Android Malware: Combined with Static Analysis and Dynamic Analysis*, **The 2016 International Conference on High Performance Computing & Simulation (HPCS 2016)**, Innsbruck-Avusturya, (2016) 1013-1018.

[23] M. Spreitzenbarth, T. SchreckFlorian, E. Arp, J. Hoffmann (2014) **DOI**: **10.1007/s10207-014-0250-0.**

[24] R. Pandita, X. Xiao, W. Yang, W. Enck, T. Xie *Whyper: towards automating risk assessment of mobile applications*, **22nd USENIX Security Symposium**, Washington, D.C.-Amerika, (2013).

[25] S. Chakradeo, B. Reaves, P. Traynor, W. Enck, *MAST: Triag e for Market-scale Mobile Malware Analysis*, **WiSec'13 (2013)**, Budapeşte-Macaristan.

[26] P. Teufl, M. Ferk, A. Fitzek, D. Hein, S. Kraxberger, C. Orthacker (2013) **DOI: 10.1002/sec.675.**