ANDROID KÖTÜCÜL YAZILIM TESPİTİ YAKLAŞIMLARI

Ceren Aslanalp Dinçer, İbrahim Alper Doğru
Teknoloji Fakültesi Bilgisayar Mühendisliği Bölümü
Gazi Üniversitesi, Ankara, Türkiye
ceren.aslanalp@gmail.com, iadogru@gazi.edu.tr

ÖZET

Mobil cihazlar, mobil uygulamaların işlevselliklerinin gelişmesiyle birlikte hem iş hem günlük hayatta vazgeçilmez cihazlar olmaya başlamıştır. Kendi cihazını getir iş modeli ile beraber bu cihazlar iş ve kamu kurumlarının ağlarına bağlanarak, beraberinde kötücül yazılımların tüm risklerini organizasyona taşımaktadır. Kötücül davranış, bilgiye ve cihaza yetkisiz erişim dolayısıyla hem kurum hem kişiye karşı ciddi ölçüde tehdit oluşturmaya başlamıştır. Android, açık kaynak çekirdek politikası sebebiyle bu tehditlere çok daha fazla açık bir platformdur. Bu kötücül yazılımları tespit edip önlem almak için tespit mekanizmaları geliştirilmekte, buna karşılık olarak kötücül yazılım geliştiricileri dönüşüm gibi güçlü tekniklerle bu tespit tekniklerinden kaçmayı amaçlamaktadır. Bu çalışmada, Android kötücül yazılım tespiti yaklaşımları sunan farklı çalışmalar incelenmiştir ve bu çalışmalar çeşitli ölçütler bakımından karşılaştırılmıştır.

Anahtar Kelimeler: Android malware, kötücül yazılım, kötücül yazılım tespiti, dinamik yaklaşım, statik yaklaşım, imza tabanlı yaklaşım, hibrit yaklaşım

ANDROID MALWARE DETECTION APPROACHES

ABSTRACT

With the development of the functionality of mobile applications, mobile devices have become essential in both business and daily life. Bring your own device business model permits the employees to connect their own devices to the networks of business and public institutions and carry all the risks of malware together with the organization. Malicious behavior via unprivileged access to information and device has been a serious threat to both organizations and individuals. Android has open source kernel policy, so it is more vulnerable to these threats. Detection approaches have been developed for the mitigation and detection of malware by time. In response to this, malware developers aim to hide from these detection techniques by developing complicated techniques like obfuscation. In this study, various studies presenting Android malware detection approaches have been reviewed and compared in terms of various criteria.

Keywords: Android malware, malware detection, dynamic approach, static approach, signature-based approach, hybrid approach

I. GİRİŞ (INTRODUCTION)

Günümüzde mobil sistemlerin gelişmesi ve yaygınlaşması ile birlikte geçmişte kullanılan geleneksel cep telefonları artık yerini akıllı telefonlara bırakmıştır. Akıllı telefon marketinin dünya çapındaki hızlı büyümesinin sebeplerinden biri de akıllı telefonların iş amacıyla kullanımının yaygınlaşmasıdır. Kendi Cihazını Getir iş modeli günden güne kabul görmeye başlamıştır. Akıllı telefonların en farklı özelliklerinden biri genellikle 'apps' olarak anılan üçüncü parti uygulama programlarının kullanıcılar tarafından telefona yüklenebilmesi ve çalıştırılabilmesidir. Bu uygulamalar genellikle resmi olarak çevrimiçi mağazalardan dağıtılmaktadır. Apple Store IOS platformu ve Google Play Store Android platformu için kullanılmaktadır. Bu mağazalar kullanıcıların yeni uygulamaları keşfetmesi ve yüklemesi için uygulama geliştiricilerine uygun bir mekân sağlamaktadır [1].

Android'in açık kaynak çekirdek politikası sebebiyle kötücül yazılım geliştiricileri bu mobil platform hakkında daha derin bilgi sahibi olabilmektedir. Google Market'in stratejisi dolayısıyla üçüncü parti

uygulamaların geliştirilmesi teşvik edilmektedir [1]. 3. parti uygulama marketleri ise kapsamlı bir güvenlik taraması olmadan uygulama dağıtılmasına olanak sunmaktadır [2]. Android kötücül yazılım konusu hem kurumsal hem bireysel kullanıcılar için artan bir problem haline gelmektedir. Android işletim sistemini hedef alan zararlı uygulamaların sayısı dramatik bir biçimde atmaktadır. G DATA şirketinin 2016 birinci yarı mobil kötücül yazılım raporuna göre; G DATA güvenlik uzmanları 2016'nın ilk yarısında 1.723.265 adet yeni Android kötücül yazılım örneği tespit etmiştir. Ek olarak, Android kullanıcılarının sadece %13'ünün resmi Play Store'u kullandığı ve Android 6.0 sürümünde olduğu; %30'undan fazlasının ise hala eski Android sürümlerden biri olan Kitkat (4.4)'ı kullandığı ifade edilmiştir [3].

Android kötücül yazılıma karşı önlem almak için mevcut endüstri yaklaşımı telefona virüs tarayıcısı yüklemektir. Bu virüs tarayıcılar tipik olarak Dalvik sanal makinesinde çalışır ve bilinen kötücül yazılım imzalarıyla yüklenen uygulamaları karşılaştırır. Bu "kara liste" tekniği kötücül yazılım dağıtıcıları tarafından istismar edilebilen bir zayıflıktır. Bir sıfır-gün kötücül uygulaması kritik sistem dosyalarını değiştirerek ayrıcalık yükseltebilmekte ve telefonun davranışını değiştirebilmektedir. Bu sebeple virüs tarama motorları bu saldırıları gözden kaçırmaktadır [4].

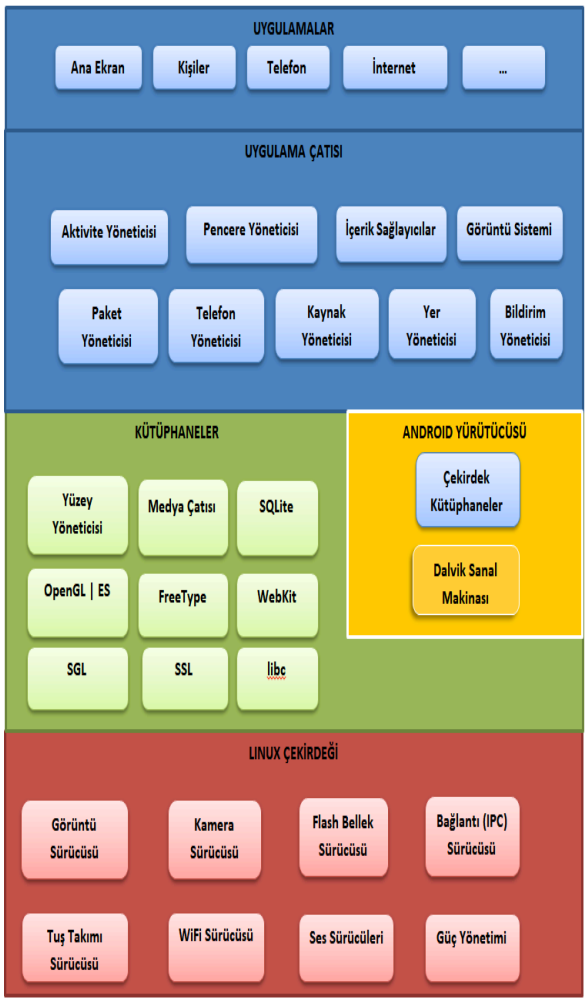
Akıllı telefonlar çoğunlukla kurum sınırları dışında da aktif olduğu için ağ seviyesindeki izleme araçları da etkili olmamaktadır. Bu yüzden, cihazların zararlı uygulamalara karşı kendilerini izleyebileceği sağlam yöntemler gereklidir [4]. Bu kötücül yazılımları tespit edip önlem almak için tespit mekanizmaları geliştirilmekte, buna karşılık olarak kötücül yazılım yazarları güçlü tekniklerle bu tespit tekniklerinden kaçmayı amaçlamaktadır.

Bu çalışmada, ikinci bölümde Android'in mimarisi anlatılmıştır. Üçüncü bölümde kötücül yazılımlara yer verilmiştir. Dördüncü bölümde kötücül yazılım (malware) tespiti için önerilen farklı yaklaşımlar incelenmiştir. Beşinci bölümde incelenen sistemlerin karşılaştırması yapılmış ve altıncı bölümde çalışmanın sonuçlarına yer verilmiştir.

II. ANDROID MİMARİSİ (ANDROID ARCHITECTURE)

Android işletim sistemi, Şekil 1'de gösterildiği gibi katmanlardan oluşan bir yazılım bileşenleri yığınıdır [5].

Katmanların en altı Linux 3.6'dır. Bu katman işlem yönetimi, bellek yönetimi, cihaz yönetimi, kamera, tuş takımı, görüntü vb. temel sistem fonksiyonlarını sağlar. Linux çekirdeğinin üstünde çatı (framework) kütüphaneleri ve veri tabanı erişimi, grafik çizimi gibi işlevlere olanak sağlayan bir kütüphane kümesi bulunur. Android Yürütücüsü, kilit bir bileşen olan Dalvik Sanal Makinası'nı sağlar. Bu bileşen, her Android uygulamasının kendi sürecinde ve kendi Dalvik Sanal Makinası üzerinde çalışmasını sağlar. Uygulama Çatısı Android uygulamalarına servisleri sağlar. Uygulama geliştiriciler bu servislerden faydalanırlar. Uygulamalar sadece uygulama katmanına yüklenmek üzere yazılır [5].



Şekil 1. Android mimarisi [5]

Android uygulamaları ".apk" uzantılı sıkıştırılmış dosyalardır. Android Application Programming Interface (API) kullanılarak geliştirilmektedir ve 4 tip bileşenden oluşur: aktiviteler, servisler, yayın alıcıları ve içerik sağlayıcılar. Android yazılımı uygulamalarla bu bileşenler aracılığıyla etkileşimde bulunur. Uygulama paketlerinde çoklu sınıf dosyaları yerine tüm sınıflar tek bir .dex uzantılı dosya içine paketlenir. Android uygulama paketleri uygulama bayt kodunu, yerel kod kütüphanelerini, uygulama kaynaklarını ve AndroidManifest'i içeren jar dosyalarıdır. AndroidManifest uygulama paket adını, uygulama izinleri vb. bilgileri içeren XML dosyasıdır. İnsan tarafından okunabilir XML formatında yazılır ve uygulama inşası sırasında binary XML'e dönüştürülür [6].

III. MOBİL KÖTÜCÜL YAZILIM (MOBILE MALWARE)

Bir kötücül yazılım akıllı telefona yüklendikten sonra cihazdaki veriye erişmeye çalışacak, normal işlevlerine müdahale edecek ya da uzaktan erişim açmak gibi telefonu daha savunmasız hale getirecektir [1]. Cabir, 2004 yılında tespit edilmiş Dünya'nın ilk mobil solucanıdır. Nokia 60 serisine zarar vermek için tasarlanmıştır. Saldırısı sonucunda ekranda 'Cabire' yazısı çıkmaktadır. Daha sonra solucan kendi kendini kopyalayarak telefonun bluetooth bağlantısını kullanarak yakınındaki diğer cihazlara bulaşmaktadır [7]. Genel olarak, çok çeşitli saldırı tipi kötücül yazılım aracılığıyla başlatılabilir. Bunların bazıları şunlardır:

- Casus Yazılım: Akıllı telefonlardan gizlice kullanıcı bilgilerini toplayan kötücül yazılımlardır [1].
- Gözetim Saldırıları: Kullanıcının telefonundaki GPS, mikrofon, kamera gibi sensörler aracılığıyla takip edilmesidir [1].
- Çevirici (Diallerware) Saldırıları: Kullanıcılar farkında olmadan yüksek oranlı arama ya da SMS servisleri ile yüksek ücretlere maruz bırakılabilirler [1].
- Finansal Kötücül Yazılım: Kredi kartı bilgilerini toplayan bir keylogger ya da gerçek bankacılık uygulamasını taklit eden bir uygulama olabilir [1].
- Botnetler: Uzaktan yönetilebilen kötücül yazılım bulaşmış zombi cihazlar topluluğudur. Organize bir biçimde saldırı başlatırlar [1].
- Solucan: Kendi kendini çoğaltır ve ağ üzerinde kullanıcı müdahalesi olmadan bir cihazdan başka cihaza bulaşabilir [1].
- Fidyecilik: Şifreleme ile kullanıcı verisi şifrelenmekte ya da ekranı kilitlemektedir ve şifre çözme anahtarı saldırganın elinde tutulmaktadır. Saldırgan, bir miktar para karşılığı veriyi ya da cihazı serbest bırakacağını iddia etmektedir.[3].

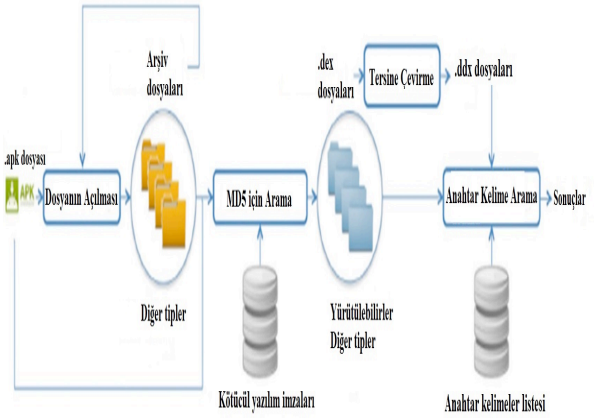
IV. VERİ SETLERİ ÜZERİNDEN KÖTÜCÜL YAZILIM TESPİTİ YAKLAŞIMLARI (MALWARE DETECTION APPROACHES ON DATA SETS)

Android kötücül yazılımlarının her geçen gün artmasıyla birlikte tespit yöntemleri de araştırılmakta ve geliştirilmektedir. Yapılan çalışmada bu yöntemlerden statik, dinamik, hibrit ve imza tabanlı çalışmalar incelenmiştir.

A. Statik Analiz Yaklaşımı

Bu yaklaşım, kötücül yazılım tespitinin uygulamalar cihaza yüklenmeden yapılmasını sağlar. Böylelikle mobil cihaz uygulamanın kötücül işlevinden etkilenmemektedir. Bu yaklaşım, uygulama çalıştırılmadan onun kötücül karakteristiklerini ve kötü kod parçalarını tespit eden hızlı ve pahalı olmayan bir yaklaşımdır [8–9].

DroidAnalyzer, Android uygulamalarının potansiyel zafiyetlerini ve root ayrıcalığı istismarı varlığını tanımlayan bir statik analiz aracıdır. Çalışmada uygulama izinleri, riskli API'ler ve root ayrıcalığı istismarının varlığını gösteren anahtar kelimeler incelenmiştir. Riskli izinler ve anahtar kelimeler öncelikle veri seti üzerinde tespit edilmiş, daha sonra hedef uygulamalar incelenmiştir. Uygulamaların MD5 kriptografik özet değerleri ve anahtar kelimelere atanan şüphe seviyelerinin karşılaştırmasına dayalı bir algoritma kullanılmıştır [2]. Şekil 2'de aracın mimarisi gösterilmiştir.



Şekil 2. DriodAnalyzer mimarisi [2]

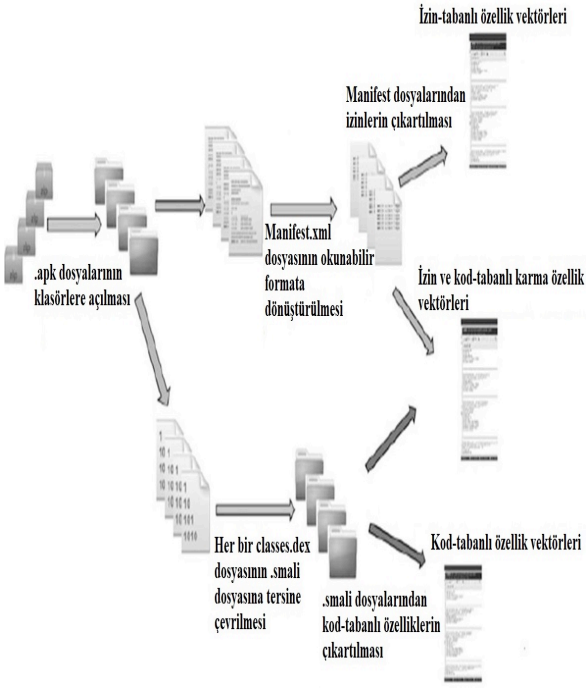
Xing Liu ve Jiqiang Liu izin tabanlı iki katmanlı bir tespit şeması önermişlerdir. İlk katman ilk iki aşamayı, ikinci katman üçüncü aşamayı içermektedir. Uygulama yükleme esnasında talep edilen izinler 'istenen izinler' olarak ve uygulama çalışması sırasında kullanılan izinler 'kullanılan izinler' adlandırılmıştır. Sistem tasarımında zararsız ve kötücül veri setindeki her uygulama için: istenen izinler açılan APK dosyasından alınan AndroidManifest.xml dosyası normal bir xml dosyasına çevrilerek ve daha sonra Python'daki xml.dom.minidom paketiyle ayrıştırılarak çıkartılmış ve boolean değerlere çevrilmiştir. Kullanılan izinleri elde etmek için Dex dosyasının analiz edilmesi gerekmektedir. Bunun için APK dosyası apktools ile geri derlenerek uygulamanın istenen izinleri ve smali kodu elde edilmiştir.

Daha sonra her istenen izin için ilgili API çağrıları, içerik sağlayıcı URI'ları, ve intent action dizgilerine bakılmıştır. İlk aşamada istenen izinler ve C4.5'un WEKA uygulaması olan J48 sınıflandırıcı kullanılarak uygulamalar sınıflandırılmıştır. İkinci aşamada aynı kümedeki uygulamalar istenen izin çiftleri ve J48 sınıflandırıcı kullanılarak sınıflandırılmıştır. Eğer bir

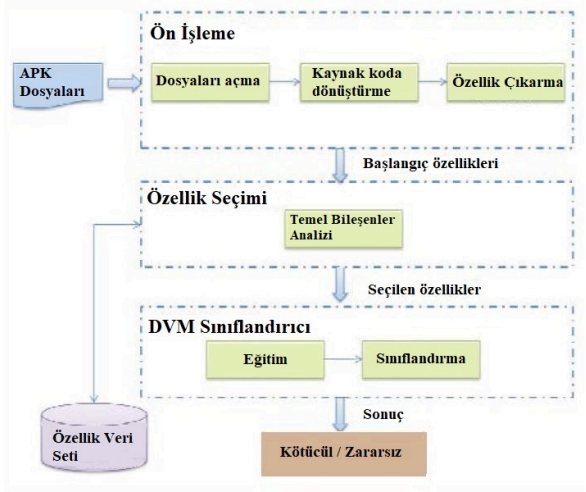
uygulama ilk iki aşamada zararsız olarak sınıflandırıldıysa o zararsız olarak kabul edilmektedir ya da kötücül olarak sınıflandırıldıysa kötücül olarak kabul edilmektedir. Eğer ilk iki aşamada farklı kümelerde sınıflandırıldıysa, üçüncü aşamada, başka bir kümeye konmakta ve kullanılan izinler çifti ve J48 sınıflandırıcı kullanılarak sınıflandırılmaktadır [10].

Yerima ve arkadaşları proaktif Android kötücül yazılım tespiti için statik analiz uygulayan Bayes sınıflandırması tabanlı makine öğrenmesi yaklaşımları önermiştir. Bu yaklaşımlarda izin tabanlı özellikler, kod tabanlı özellikler ve her ikisinden oluşan karma özellikli modeller incelenmiştir. İzin-tabanlı Bayes sınıflandırıcıda APK Analyser her uygulamanın manifest dosyasından izinleri çıkarmış ve izin detektörünü kullanarak standart Android izinleriyle eşleşme yapmıştır. Bir izin tespit edildiğinde onun sayısı yükseltilmiş ve kaydedilmiştir. Her izin için kaydedilen toplam Bilgi Kazancı'nı kullanan izin seçimi fonksiyonu tarafından derecelendirme ve en ilgili izinleri seçme için kullanılmıştır. Kod tabanlı Bayes sınıflandırıcıda, Bir miktar kod tabanlı özellik bir özellik detektörü setinin eşleşme kriteri olarak ayrıştırılmış ve APK analyser'da uygulanmıştır. Detektörler .smali dosyalarını ve varsa ek olarak dış kütüphaneleri, asset dosyalarını ve kaynak klasörlerini inceleyip ayrıştırmıştır. Bunlar daha sonra Bilgi Kazancı kriteriyle en ilgili özellikler seçilerek azaltılan büyük bir özellik seti sağlamıştır. Özellik seçimi aşamasından sonra en yüksek dereceli kod tabanlı özellik, eğitimde kullanılan her uygulamayı karakterize eden girdi özellik vektörlerini oluşturmak için kullanılmıştır. Derecelendirilmiş izinler ve kod tabanlı özelliklerden karma sınıflandırıcıda, özellik seçimi fonksiyonu aynı anda izinleri ve kodları derecelendirmek için kullanılmıştır. Daha sonra her ikisinden de en yüksek derecelendirilmiş olanlar Bayesian sınıflandırma modeli için girdi özellik vektörü olarak seçilmiştir. Şekil 3'te modelleri oluşturmak üzere otomatikleştirilmiş tersine mühendislik ve veri madenciliği için inşa edilen APK Analyser'ın yapısı gösterilmiştir. Yapılan sınıflandırma performansı değerlendirmeleri sonucunda bileşik tabanlı ve kod özelliği tabanlı modellerin izin tabanlı modellere göre daha iyi bir seçim olduğu görülmüştür [11].

Zhao Xiaoyan ve arkadaşları, izin tabanlı hafif bir statik kötücül yazılım tespit sistemi önermişlerdir. Şekil 4'te çatı modülleri gösterilmektedir. Ön İşleme Modülünde, AndroidManifest.xml dosyasından APK'ların izin listesi alınmıştır ve başlangıç özellik seti oluşturulmuştur. Manifest dosyasını okunabilir XML dosyasına çevirebilmek için AXMLPrinter2.Jar kullanılmıştır. Her uygulama için izin listesi Android sistem izinlerinin toplam setiyle karşılaştırılmıştır ve özellik değerleri 0 ya da 1 olarak kaydedilmiştir. Özellik Seçme Modülünde, orijinal özellik setinden Temel Bileşenler Analizi algoritmasıyla temel bileşen çıkartılmıştır. Destek Vektör Makinesi (DVM) Sınıflandırma Modülünde, eğitim aşamasında, kötücül ve zararsız uygulama örneklerinin özellik vektörleriyle sınıflandırıcı eğitilmiştir. Tespit aşamasında, sınıflandırıcı bilinmeyen bir APK'yı sınıflandırmaktadır. Özellik Veri Seti, örneklerden çıkartılan özellikleri depolama ve güncelleme ile sorumludur. [12].



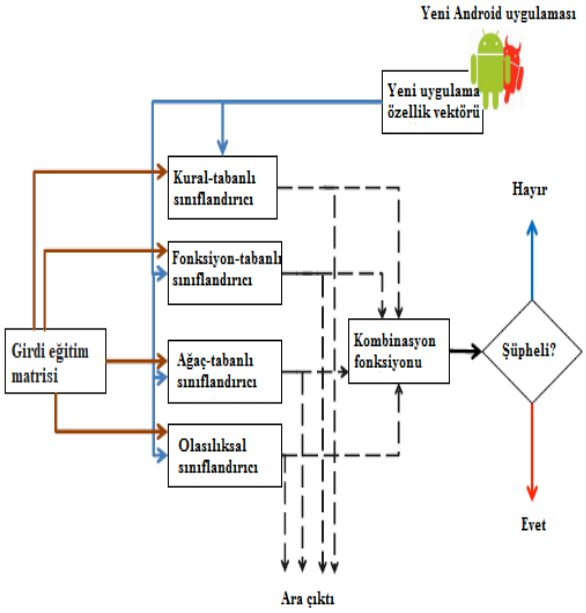
Şekil 3. APK Analyser tarafından yapılan otomatikleştirilmiş tersine mühendislik ve veri madenciliği [11]



Şekil 4. Kötücül yazılım tespit çatısı [12]

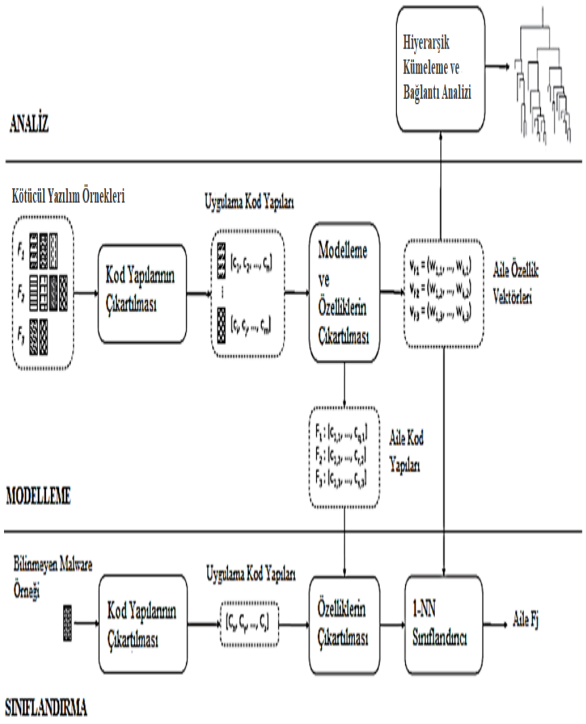
Yerima ve arkadaşları statik özellikler kullanılarak paralel makine öğrenmesi sınıflandırıcıları vasıtasıyla kötücül yazılımın erken tespiti için bir metot önermişlerdir. Öğrenme aşaması için API ilişkin özellikler, uygulama izinleri, standart işletim sistemi ve Android çatı komutları kullanılmıştır. Çalışmada kullanılan algoritmalar şunlardır: Karar Ağacı (ağaç-

tabanlı), Simple Logistic (fonksiyon-tabanlı), Naïve Bayes (olasılıksal), PART (kural-tabanlı), ve RIDOR (kural-tabanlı). Deneylerin ilk kısmı her bir aday sınıflandırma algoritmasıyla yapılmıştır. Naive Bayes sınıflandırıcının en düşük, PART'ın en yüksek doğruluk oranına sahip olduğu görülmüştür. Deneylerin ikinci kısmında her ayrı sınıflandırıcıdan elde edilmiş sınıflandırma kararlarının; olasılıklarının ortalaması, olasılıklarının çarpımı, maksimum olasılık, çoğunluk oylamasının sonucuna dayanan kombine sınıflandırma yaklaşımı incelenmiştir. En iyi doğruluk oranının olasılıkların çarpımı şemasından geldiği görülmüştür. Şemanın tüm performans sonuçlarının tek sınıflandırıcılardan daha iyi olduğu görülmüştür [13]. Şekil 5'te bu yaklaşımın mimarisi gösterilmiştir.



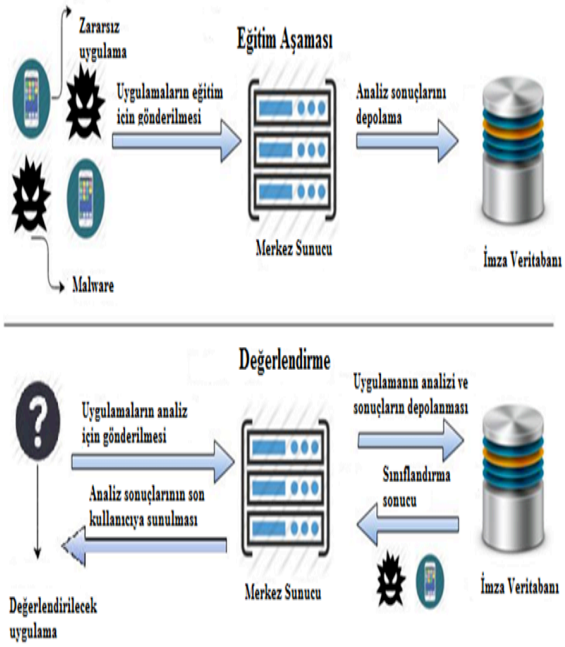
Şekil 5. Bileşik paralel sınıflandırıcı yaklaşımı [13]

Suarez-Tangil ve arkadaşları metin madenciliği ve bilgi alma tekniklerine dayalı bir sistem olan Dendroid'i önermişlerdir. Bu çalışmada, akıllı telefon kötücül yazılım örnekleri ve ailelerinin onların yazılım bileşenlerinde mevcut olan kod yapılarına dayanarak otomatik olarak analiz edilmesi için metin madenciliği yaklaşımlarının kullanımları araştırılmıştır. Dendroid'in ana yapı blokları Şekil 6'da sunulmuştur. Veri setinde bulunan her kötücül yazılım örneği için öncelikle Dalvik komutları tersine çevrilmiştir. Daha sonra Androguard kullanılarak kötücül uygulamaların kod parçaları çıkartılmış ve yapıları işlenmiştir. Vektör Uzay Modeli adapte edilerek her aileden aile özellik vektörü elde edilebilmesi için veri setindeki tüm aileler üzerinde uygulanmıştır ve bilinmeyen örnekleri bilinen aileler içine sınıflandırmadaki uygunluğu araştırılmıştır. Daha sonra, kötücül yazılım aileleri için filogenetik ağaçlar olarak anlaşılabilecek dendogramları türetmek için hiyerarşik kümeleme kullanımı üzerinde çalışılmıştır. Bu, aileler arasındaki ilişkileri, ortak soyların varlığını, belli kod özelliklerinin yaygınlığını analiz etmek için bir aracı analist olmuştur. Çalışmada yapılan deney sonuçlarında yaklaşımın önemli ölçüde kesin olduğunu ve kötücül yazılım örneklerinin büyük veri tabanları ile verimli biçimde baş ettiğini gösterdiği ifade edilmiştir [14].



Şekil 6. Dendroid'in yapısı [14]

Kabakuş ve arkadaşları izin tabanlı statik analiz yapan öğrenmeye dayalı bir sistem olan APK Auditor'ü önermişlerdir. Sistem mimarisi Şekil 7'de gösterilmiştir. Mobil cihaza yüklenen istemci uygulaması hem yereldeki uygulamaları hem de Play Store'da uygulamaları taramaktadır ve kullanıcılar uygulama izinleri ve güvenlik seviyeleri hakkında bilgi alabilmektedir. Uygulamanın talep ettiği izinler, hafıza yönetimi ve istemcinin koruma seviyesi istatistiği ara yüzde görsel olarak sunulmaktadır. Merkez sunucunun iki ana işlevi vardır: Play Store'dan düzenli olarak en popüler uygulamaların sisteme yüklenmesi, incelenmesi ve kötücül yazılım puanı hesaplanması ve Android uygulamalarının anlık olarak kötücül içerik kontrolünden geçirilmesi. Merkez sunucuda uygulamalar incelenirken öğrenmeye dayalı bir mekanizma sunulmuştur. Öğrenme aşamasında uygulamanın karakteristiği çıkarılmakta ve veri tabanında depolanmaktadır. Eğitim veri seti kullanılarak uygulamaları sınıflandırmak için kötücül yazılım skor eşik değeri ve lojistik regresyon kullanılmıştır. Değerlendirme aşamasında ise incelenmek istenen uygulama öğrenme sonucunda üretilen profillerden hangisine yakınsa o gruba dâhil edilmektedir. İncelenen uygulamalar, inceleme sonuçları, izinler, servisler ve alıcılar ilişkisel bir veri tabanı halinde depolanmaktadır [15].



Şekil 7. APK Auditor mimarisi [15]

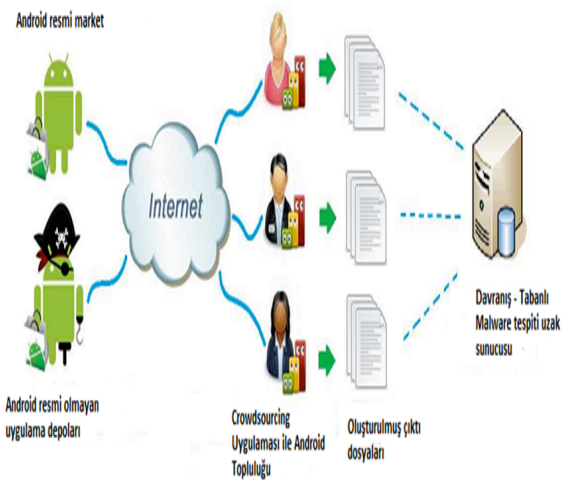
Arslan ve arkadaşları fazladan izin talebinde bulunarak şüpheli kaynak erişimi yapabilecek Android uygulamalarını tespit etmek için, veri setleri kullanarak daha önceden belirlenmiş seviyeler doğrultusunda uygulamaların izin ve kod yapılarını inceleyerek risk değerlerini belirleyen bir yöntem önermişlerdir. Uygulamalar apktool ile açılarak Manifest dosyası okunabilir formata dönüştürülmüştür. Uygulamaların kaynak koduna erişmek için Dex2jar aracı kullanılmıştır. Manifest dosyasında istenen izinler 135 adet Android izni ile karşılaştırılmış, varsa 1 yoksa 0 değeri kullanılarak kötücül ve zararsız veri setleri için birer çizelge oluşturulmuştur. Daha sonra metin arama metodu kullanılarak her bir iznin kaynak kodunda çağırılıp çağırılmadığı 0 veya 1 değeri kullanılarak belirlenmiştir. Her bir uygulama için, talep edilmiş fakat kullanılmamış tüm izinlerin kötücül uygulamalar içerisinde kullanılma sayısı toplanarak şüphe değeri hesaplanmıştır. Bu değer zararsız uygulama veri setinin şüphe değeri ortalamasından yüksek ise uygulama kötücül olarak belirlenmiştir. Çalışmanın uygulama kısmında kötücül veri seti Drebin veri setinden ve zararsız veri seti bankacılık uygulamalarından alınarak oluşturulmuştur. Çalışma sonucunda kötücül ve zararsız uygulamalar için hesaplanan şüphe değeri yaklaşımının belli bir seviyede ayırt edici değerler elde etmeye yardımcı olduğu ifade edilmiştir. Ayrıca bu yaklaşımın tek başına yeterli olmadığı ve dinamik analiz yaklaşımı ile birlikte kullanılarak daha etkili ve doğru sonuçlar üreteceği belirtilmiştir [16].

Kayabaşı ve Doğru, oluşturdukları zararsız ve kötücül veri setleri üzerinden mobil uygulamaların sınıflandırılmasında kullanılan çeşitli makine öğrenmesi algoritmalarını güvenirlilik bakımından değerlendirmiştir. Zararsız uygulamalar Google Play Store'dan Android Market API kullanılarak indirilmiştir ve zararsız veri seti oluşturulmuştur. Kötücül uygulama veri seti için ise Malgenome Projesi olarak da bilinen Android Malware Genome Project'ten uygulamalar alınmıştır. Uygulamalar apktool aracını kullanan bir program kodlanarak tersine derlenmiştir. Uygulamaların API çağrıları, bu çağrıların paket düzeyindeki bilgileri ve başlangıç modelini oluşturmak için istenen izinler çıkartılarak zararsız ve kötücül uygulamalar için özellik kümesi belirlenmiştir. Zararsız ve kötücül sınıflara ait veri setleri kullanılarak, bir makine öğrenmesi algoritmaları aracı olan Weka ile sistemin eğitim aşaması gerçekleştirilmiştir. Çalışmada, sınıflandırma işlemini uygulamada başarılı olanlar seçilerek 4 adet makine öğrenmesi algoritması kullanılmıştır: KNN, Naive Bayes, ID3 ve J48. Çalışmada yapılan değerlendirmelerde en güvenilir algoritmanın KNN ve en zayıf algoritmanın ise Naive Bayes olduğu belirtilmiştir [17].

B. Dinamik Analiz Yaklaşımı

Statik analiz yaklaşımından farklı olarak, mobil uygulama analizleri yürütme esnasında yapılır. Statik analiz yaklaşımıyla ortaya çıkartılamayacak karışıklıktaki eksiklik veya açıklıklar ortaya çıkartılabilmektedir [8].

Burguera ve arkadaşları Android Platformunda kötücül yazılım tespiti yapmak için yeterli kaynak ve mekanizmaları sağlayan bileşenlere sahip bir çatı olan Crowdroid'i önermişlerdir. Crowdroid'in yapısı Şekil 8'de sunulmuştur. Bu uygulama Linux çekirdek sistem çağrılarını gözlemlemekle ve onları ön işlenmiş bir şekilde merkezi bir sunucuya göndermekle yükümlüdür. Crowdsourcing felsefesine göre kullanıcılar kullandıkları her uygulama için kişisel olmayan ama davranışsal veriyi göndererek yardımcı olacaklardır. Daha sonra uzak sunucu veriyi ayrıştırmakla ve uygulamalardaki her kullanıcı etkileşimi için sistem çağrı vektörü oluşturma ile yükümlü olacaktır. Böylece, kullanılan her uygulama için davranış verisi veri seti oluşmuş olacaktır. Son olarak, her veri seti bir bölücü kümeleme algoritması kullanılarak kümelenir. Bu şekilde meşru uygulamalar ile kötücül uygulamalar –aynı ada ve tanımlayıcıya sahip olsalar bile- arasındaki çok küçük sistem çağrı örüntüleri ayırt edilebilmiştir. Normallik modelini oluşturmak ve Android uygulamalarındaki anormal davranışları tespit etmek için bir önceki aşamada elde edilen vektörler analiz edilir ve kümelenir. Crowdroid sistem çağrılarını toplamak için Linux'ta bulunan Strace aracını kullanmaktadır. Crowdroid'in yapısı Şekil 8'de gösterilmiştir. Yapılan testlerin sonucunda, sistem çağrılarını gözlemlemenin kötücül yazılım tespiti için uygulanabilir bir yöntem olduğu belirtilmiştir. [18].



Şekil 8. Crowdroid'in yapısı [18]

Lu ve arkadaşları, Android cihazında uygulama davranışlarını gözlemleyen ve makine öğrenmesi tekniklerini uygulayarak uygulamaların zararsız ya da kötücül olduğunu tespit eden bir teknoloji önermişlerdir. Analiz ettikleri davranışlar şunlardır: 1)Kullanıcılardan yetkisi olmadan mesaj gönderme ya da silme, 2) Kullanıcının telefonunun kötücül olarak ele geçirildiğini fark etmemesi için mesaj önleme 3)Kullanıcının haberi olmadan uygulama yükleme, indirme ya da silme, 4)Kullanıcının SD kart içeriğini ele geçirme, 5)ICCID, IMEI, IMSI ya da MSISDN gibi mobil terminal hakkında bilgi edinme, 6)Kullanıcının arama geçmişi, telefon rehberi, konumu vb. kişisel bilgilerini elde etme, 7)Kullanıcı farkında olmadan internet bağlantısı yaparak kötücül kullanım yapma. Kötücül davranışı tetikleyen aktiviteleri gözlemlemek için aktiviteler ve yayın alıcıları incelenmiştir. Ek olarak kötücül davranışın yapılabilmesi için gereken izinler de AndroidManifest.xml'den alınmıştır. Naive Bayes sınıflandırmayı uygulamadan önce, etkinliğini arttırmak için özellikler Chi-Square metotla filtrelenmiştir. Kötücül yazılım davranışının amacına ulaşması için bir seri eylemi sırasıyla gerçekleştirmesi gerekmektedir. Örneğin, kullanıcının rehberini elde etmek (davranış) istiyorsa, öce onu okumalı, daha sonra internet ya da mesaj yoluyla bilgiyi göndermelidir. Dolayısıyla, davranışlar özellikler ve eylemler kümeleri ile ifade edilmiş ve Chi-Square metotla filtrelenmiştir. Chi-Square metodu kullanılarak ve kullanılmadan aynı veri seti üzerinde Naive Bayes sınıflandırma test edilmiştir. Kombine metodun daha düşük yanlış pozitif ve daha yüksek doğruluk oranı olduğu görülmüştür [19].

Alzaylaee ve arkadaşları Android uygulamalarını otomatik olarak analiz etmek için dinamik analiz tabanlı bir yapı olan DynaLog'u önermişlerdir. DynaLog'un bileşenleri Şekil 9'da gösterilmiştir [20].

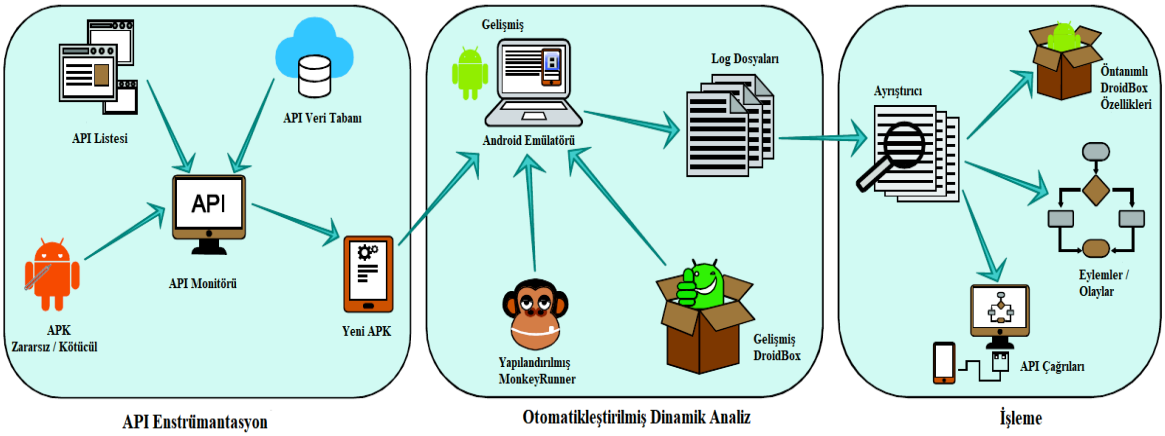
Emülator-tabanlı güvenli sanal ortamda analiz bileşeninde, uygulama bir Android emülatörü üzerinde çalıştırılarak DroidBox aracı ile bazı üst seviye davranışlar ve karakteristikler çıkartılmaktadır. APK enstrümantasyon modülünde, DroidBox tarafından ayrıştırılamayan ve loglanamayan API çağrıları izlenmiştir. Bunun için, derleyici/ayırıcı araçlarını içeren APIMonitor aracı kullanılmıştır. Enstrümantasyon işlemi, dex dosyasına tersine mühendislik yapmayı ve uygulama çalışırken emülatör logunun içine bir API çağrısı sınıfının veya metodunun varlığını izlemek için kullanılabilecek imzaları eklemeyi içermektedir. Davranış/özellik loglama ve çıkarma bileşeninde, DynaLog izlenen davranışlara ya da API çağrı imzalarına karşılık gelen belirli log girdilerini çıkarma yeteneğini sağlar. Tetikleyici/egzersiz modülünde, MonkeyRunner aracı kullanılarak uygulamalara ekran dokunuşları, kaydırma vb. rastgele olaylar gönderilmektedir ve uygulamada mevcut olan tüm aktivite ve servislerin çalışması sağlanmaktadır. Log ayrıştırma ve işleme modülünde, otomatikleştirilmiş yapı ile analiz edilen her uygulama için çıkartılan özellikler okunabilir çıktı raporlarına biçimlendirilmektedir. Önerilen yapıyı değerlendirmek için Malgenome Projesinden alınmış kötücül ve MCAfee laboratuvarında incelenmiş zararsız uygulamalar kullanılmıştır. Yapılan değerlendirmeler sonucunda önerilen yapının sofistike Android kötücül yazılımının kitlesel tespitinde kullanılabilecek yetenekte olduğu ifade edilmiştir [20]

C. İmza Tabanlı Yaklaşım

Bu tür yaklaşımda uygulamalar analiz edilerek bir imza veri tabanında saklanır. Merkezi bir sunucu analiz ve koruma süreçlerini gerçekleştirirken, veri tabanı sunucusu analizleri saklar ve sonraki analizlerde tekrar kullanılmasını sağlamaktadır [8].

Guido ve arkadaşları kurumsal olarak kullanılan Android cihazlara istenerek ya da istenmeyerek yüklenmiş zararlı uygulamaların tespiti için, Android telefonlarda yerel bir bileşenle koşan Tractor Beam adında özel bir servis önermişlerdir. Servis, periyodik olarak her blok cihaz için değişen bit dizilerinin ofsetlerini, bir SHA256 kriptografik özeti yerel bir SQLite veri tabanında saklanmış bir önceki ölçüm ile karşılaştırarak tespit eder. Akıllı telefon merkez sunucunun dinlediği servisle Wi-Fi üzerinden iletişim kurabilir durumda olduğunda Tractor Beam güvenli yetkilendirilmiş bağlantıları başlatır, depolama ve analiz için her değişmiş bit dizisini telefon blok cihazından okur ve kopyalar. Tractor Beam'in yerel depolama ihtiyacı küçüktür çünkü blok cihaz bit dizileri değil sadece onun SHA256 kriptografik özetleri tutulur [4]. Merkez sunucu HTTPS portunu dinler. Merkez sunucunun servisi gelen bit dizileri ve ofset değerleri için dinler ve bunları normalize edilmiş ilişkisel veri tabanında saklar. Merkez sunucu aynı zamanda hem temel imajı hem veri tabanındaki gelen bit dizileri ve ofset yerlerini kullanarak hedef telefonun blok cihazlarının imajlarını dosya sisteminin

geçici alanında tekrar oluşturur [4]. Analiz uygulama çatısı Detektör ve loglayıcılar olarak adlandırılmış, otomatize edilmiş adli bilişim işlemleri serisini yeniden oluşturulmuş imajlar üzerinde dinamik bir biçimde çalıştırmaktadır. Detektörlerin içinde kötücül aktiviteyi tespit edecek teknikler geliştirilmiştir ve loglayıcılar şüpheli aktiviteleri kaydeder. Tractor Beam ve bileşenlerinin sık değişmemesi gereken dosya sistemi değişikliklerini tespit ettiği, cihaz yeniden başlatıldıktan sonra kötücül yazılımın tekrar başlamasına izin veren sürekli yapıları bulduğu, yeni yüklenmiş uygulamalar içinde dosya sistemine konulan kötücül dosyaları bulduğu ifade edilmiştir [4].



Şekil 9. DynaLog'un bileşenleri [20]

D. Hibrit Analiz Yaklaşımı

Bu yaklaşımda uygulamanın sahip olduğu hem statik hem dinamik özellikler kullanılarak kötücül yazılım analizi yapılmaktadır.

Wang ve arkadaşları anomali tespiti ve suistimal tespitini entegre eden hem uygulama marketleri hem de kullanıcılara yönelik hibrit bir mobil kötücül yazılım tespit sistemi önermişlerdir [21].

Bu sistem iki ana kısımdan oluşmaktadır: suistimal detektörü statik ve dinamik analiz yaparak bilinen kötücül yazılım ya da yeni varyantlarını tespit etmek ve sınıflandırmakla sorumlu iken, dinamik analiz sonuçları kullanılarak anomali detektörü yeni ve bilinmeyen sıfır gün kötücül yazılımlarını tespit edebilmektedir. Hibrit analiz için gereken aşırı hesaplama kaynaklarından dolayı hem suistimal detektörü hem de anomali detektörünün cihaz dışında bulut gibi bir ortamda konuşlandırılması gerektiği ifade edilmiştir [21].

Suistimal tespitinde statik ve dinamik özellikler çıkartılmıştır. Statik analiz başlıca manifest dosyası ve tersine çevrilen dex kodlarından sağlanan özelliklere odaklanmıştır. Android Asset Packaging Tool adapte edilerek statik özellikler çıkartılmıştır. Dinamik analizde kullanılacak özellikleri çıkartmak için bir dinamik analiz çatısı olan CuckooDroid aracından yararlanılmıştır ve dosya erişimi ve operasyonları, alıcı kayıtları, çalıştırılan komutlar, içerik çözen sorgular, dinamik şüpheli aramalar, ağ operasyonları vb. olaylar izlenmiştir. Daha sonra bu statik ve dinamik özellik dizgilerini 0 ve 1 ile temsil eden numerik vektörlere dönüştürülmüştür. Çıkartılan çok sayıdaki özellik Chi2 puanlama fonksiyonu ile en yüksek puana sahip olanlar bulunarak azaltılmıştır ve özellik seçimi yapılmıştır. Daha sonra bir linearSVC sınıflandırıcı bu özellik vektörleri ile eğitilmiştir. Bilinmeyen bir uygulama sisteme verildiğinde onun özellik vektörü bu sınıflandırıcı tarafından işlenmektedir ve kötücül yazılım olup olmadığına karar verilmektedir. Eğer uygulama zararsız olarak tespit edildiyse anomali tespiti tetiklenmektedir. Değilse çoklu-aile sınıflandırıcısı kullanılarak belli bir kötücül yazılım ailesine sınıflandırılmaktadır. Anomali tespitinde dinamik özellikler kullanılmaktadır. Özellik seçimi için VarianceThreshold yaklaşımı kullanılmıştır. Anormal uygulamaları tespit etmek için, tek-sınıf DVM sınıflandırıcı zararsız uygulamalar kullanılarak inşa edilmiştir. Yeni uygulama, bu eğitilen sınıflandırıcı tarafından sıfır-gün kötücül ya da zararsız uygulama olarak etiketlenmektedir [21].

Tianda Yang ve arkadaşları etkili mobil kötücül yazılım analizi için statik bir madencilik algoritması ile dinamik bir kusur analizini kullanan hibrit bir yaklaşım önermişlerdir. Bu yaklaşımda, statik analiz ile öncelikle olası saldırı kritik yollar Android API'lerine ve mevcut saldırı desenlerine göre belirlenmekte ve dinamik analiz ile belirlenen yollar takip edilerek mevcut saldırı örnekleriyle uyumluluğu kontrol edilerek saldırı olasılığını tespit etmek için uygulama sınırlı ve odaklanmış bir kapsamda çalıştırılmaktadır. Statik analiz aşamasında; apktool kullanılarak uygulamalar açılarak kaynak dosyalarına ulaşılmaktadır. dex2jar kullanılarak .dex dosyası .class

dosyalarına dönüştürülmektedir. Bir API seti oluşturmak için jd-gui kullanılarak API'ler okunmakta ve hassas olanları seçilmektedir. Apriori algoritması temel alınarak geliştirilmiş DApriori algoritması son kümeyi oluşturmak için çalıştırılmaktadır. Elde edilen küme daha önce oluşturulmuş kötücül kütüphanesiyle karşılaştırılmaktadır ve belirlenen orandaki benzerliğe göre karar verilmektedir. Çalışmada, zararlı kütüphanesi oluşturmak için popüler zararlı uygulamalar seçilmiş ve örnekler Androguard ve Contagio Mobile'dan toplanmıştır. Daha sonra 12 adet mobil uygulama üzerinde DApriori çalıştırılmıştır: WhatsApp, Facebook, Zitmo, Godwon, GoldDream, Pincer, Gazon, SMSGoogle, Tele, Samsapo, FakenotifyB and Mouabad. Uygulamaların tamamının şüpheli olarak tanımlanması statik analizin bir kısıtlaması olarak ifade edilmiştir. Dinamik analiz aşamasında: öncelikle uygulama simüle edilmiş bir ortamda çalıştırılmaktadır, ardından izleme için bir uygulama davranışları listesi seçilmektedir. Aktiviteler bir log dosyasında saklanmaktadır ve kötücül niyet için analiz edilmektedir. Çalışmada statik analiz aşamasına göre 3 adet davranış izlenmek üzere seçilmiştir: sistem çalıştırılabilir yolu /system/xbin gibi kritik dizin yoluna erişim, http sunucular için alan adı erişimi, yeterli açıklama olmadan ücretlendirme. Çalışmada dinamik kusur analizi için Droidbox aracı kullanılmıştır. Bu araç yürütme sırasında hedefi ve gönderilen veriyi takip etmek için hassas API'leri izlemektedir. Çalışmada dinamik analiz Godwon uygulaması üzerinde çalıştırılmıştır ve log dosyası ve davranış grafiklerine göre uygulamanın zararlı olduğu sonucuna varılmıştır [22].

V. SİSTEM KARŞILAŞTIRMALARI (SYSTEM COMPARISON)

Bu bölümde incelenen çalışmaların yaklaşımları, kullandığı araçlar ve servisler, veri setleri ve kullanılan yöntem ve algoritmalar bakımından genel bir bakış sunulmuştur. Tablo 1'de bu karşılaştırmaya yer verilmiştir.

Statik analizde yaygın olarak izinler, API çağrıları ve kod özellikleri kullanılmaktayken dinamik analizde davranışlar, aktiviteler ve sistem çağrıları gözlemlemektedir. Kötücül uygulamalar için Android Malware Genome Project veri setinin yaygın olarak kullanıldığı ve Google Play Store ve 3. parti resmi olamayan marketlerden zararsız uygulamaların alındığı, zararsız olduğunun doğrulanması için Virustotal tarama servisinin kullanıldığı görülmektedir. Android Malware Genome Project (AMGP) veri seti Yajin Zhou ve Xuxian Jiang tarafından yapılan çalışma sonucunda ortaya çıkmıştır. 49 farklı kötücül yazılım ailesinden ve 1260 örnekten oluşan ilk büyük koleksiyondur. Örnekler resmi ve alternatif marketlerden toplanmıştır [23].

Sıkıştırılmış formata sahip apk uzantılı uygulama dosyalarını açmak ve izinler, kodlar gibi uygulama özelliklerini kullanabilmek için AXMLPrinter2.jar, Baksmali, apktools, dex2jar gibi çeşitli tersine mühendislik araçları kullanılmıştır. Bunun yanında dinamik özellikleri elde etmek için ve uygulama davranışlarını gözlemleyebilmek için Strace, CuckooDroid ve DroidBox araçları kullanılmıştır. Android kötücül yazılım tespitinde makine öğrenmesi algoritmalarının yaygın olarak kullanılmaya başlandığı görülmektedir. Ek olarak daha etkin sistemler geliştirmek için birden çok makine öğrenmesi algoritmasının birlikte kullanıldığı da görülmüştür.

VI. SONUÇ (CONCLUSION)

Andriod tabanlı telefonların yaygınlaşmasıyla birlikte Android için geliştirilen kötücül yazılımlar artan bir problem haline gelmiştir. Bu kötücül yazılımlar kişisel, kurumsal ve ulusal güvenlik bakımından bir tehdit oluşturmaktadır. Bu çalışmada bu kötücül yazılımlara yönelik tespit mekanizmaları incelenmiştir.

Dinamik analiz sadece yürütme esnasındaki uygulama davranışlarını izleyebilmektedir ve potansiyel zafiyetleri tespit etme yeteneğinden yoksundur. Statik analiz uygulama cihaza yüklenmeden önce potansiyel zafiyetleri tespit edebilmektedir fakat uygulama davranışlarını gözlememektedir. Hibrit yaklaşımlar hem statik hem dinamik analiz yapabilmektedir fakat yüksek kaynak kullanımına ihtiyaç duymaktadır.

Mobil cihaz kullanıcıları 3. parti marketlerden uygulama indirmekten kaçınmalıdır. Bir uygulamanın 3. parti uygulama marketlerindeki sürümü resmi marketlerdekine göre farklılıklar gösterebilmektedir ya da buradaki uygulamalar kötücül niyetle geliştirilmiş uygulamalar olabilmektedir.

Resmi uygulama marketleri uygulamaları detaylı olarak incelemeli ve etkin tespit mekanizmaları kullanarak kötücül uygulamaların bu kaynaklardan dağılmasına karşı önlem almalıdır. Uygulama izinleri güvenlik açısından önemli bir role sahiptir. Kullanıcılar uygulamaların talep ettiği izinlere karşı dikkatli olmalıdır. Kullanıcıların bir anti-kötücül yazılım ürünü kullanmaları bilinen kötücül uygulamalara karşı önlem almayı sağlamaktadır.

Mevcut önleme yaklaşımları ve tespit araçları bazı saldırıları önlemeye yardımcı olsa da kötücül yazılım davranışı hızla değişmektedir ve kötücül yazılım geliştiricileri günden güne tespit mekanizmalarından kaçabilecek yeni kötücül uygulamalar geliştirmektedir. Bu nedenle, kötücül yazılım geliştikçe daha etkin araçların geliştirilmesine ihtiyaç vardır. Bu amaçla öğrenmeye dayalı makine öğrenmesi yaklaşımları günden güne önerilen sistemlerde kullanılmaya başlanmıştır. Mobil cihazlardaki kısıtlı kaynaklar kötücül yazılım tespitine yönelik araçlar geliştirilirken dikkate alınmalıdır.

TABLO I. SİSTEM KARŞILAŞTIRMALARI

Çalışma	Yaklaşım	Araçlar / Servisler	Veri Seti Kaynağı	Yöntem / Algoritma
DroidAnalyzer [2]	Statik	Dedexer	AMGP, 3.parti marketler, Google Play Store.	Kelime ve kriptografik özet eşleşme algoritması
Liu ve Liu [10]	Statik	Apktools, xmldom.minidom	AppChina AMGP, Çin güvenlik şirketleri	Karar ağacı (C4.5)
Yerima ve arkadaşları [11]	Statik	AXML2jar, Baksmali, APKAnalyser, Virustotal	AMGP, 3.parti marketler	Naive Bayes, Bilgi Kazancı
Xiaoyan ve arkadaşları [12]	Statik	AXMLPrinter2.jar	AMGP, Google Play Store	DVM, Temel Bileşenler Analizi
Yerima ve arkadaşları [13]	Statik	APKAnalyser	MCAfee dahili kaynakları	Karar Ağacı, Simple Logistic, Naive Bayes, PART, RIDOR
Dendroid [14]	Statik	Androguard	AMGP	Vektör Uzay Modeli, 1-NN
APK Auditor [15]	Statik	APIfy ve 42matters web servisi,	Contagio mobile, Drebin, AMGP, Google Play Store	Lojistik regresyon
Arslan ve Arkadaşları [16]	Statik	Apktool, Dex2jar	Drebin, bankacılık uygulamaları	Şüphe değeri karşılaştırması
Kayabaşı ve Doğru [17]	Statik	Android Market API, apktool, Weka	Google Play Store, AMGP	KNN, Naive Bayes, ID3 ve J48
Crowdroid [18]	Dinamik	Strace	Crowdroid kullanıcıları, resmi ve resmi olmayan marketler	k-means
Lu ve arkadaşları [19]	Dinamik	Belirtilmemiş.	Belirtilmemiş.	Naive Bayes, Chi Square
DynaLog [20]	Dinamik	DroidBox, APIMonitor, MonkeyRunner	AMGP, MCAfee laboratuvarı	Davranış analizi
Guido ve arkadaşları [4]	İmza	Tractor Beam	AMGP	Bit dizilerinin karşılaştırılması
Wang ve arkadaşları [21]	Hibrit	CuckooDroid, Android Asset Packaging Tool, VirusTotal	Çin uygulama mağazaları, Drebin	LinearSVC, tek-sınıf DVM
Yang ve arkadaşları [22]	Hibrit	Apktool, dex2jar, jd-gui, DroidBox	Androguard, Contagio Mobile	DApriori, benzerlik, kusur analizi

KAYNAKLAR

[1] D. He, S. Chan, M. Guizani, "Mobile application security: malware threats and defenses," IEEE Wireless Communications, vol. 22(1), pp. 138-144, February 2015.

[2] S.- H. Seo, A. Gupta, A. M. Sallam, E. Bertino, K. Yim, "Detecting mobile malware threats to homeland security through static analysis," Journal of Network and Computer Applications, vol. 38, pp. 43-53, February 2014,

[3] G DATA Mobile Malware Report. (Visited July 2017) [Online]. Available: https://file.gdatasoftware.com/web/en/documents/whitepaper/G_DATA_Mobile_Malware_Report_H1_2016_EN.pdf.

[4] M. Guido, J. Ondricek, J. Grover, D. Wilburn, T. Nguyen, A. Hunt, "Automated identification of installed malicious Android applications," Digital Investigation, vol. 10, pp. 96-104, August 2013.

[5] Android – Architecture. (Visited July 2017) [Online]. Available:

http://www.tutorialspoint.com/android/android_architecture.htm

[6] V. Rastogi, Y. Chen, X. Jiang, "DroidChameleon: evaluating Android anti-malware against transformation attacks," In Proceedings of the 8th ACM SIGSAC symposium on Information, computer and communications security, pp. 329-334, May 2013.

[7] A. Apvrille, "The evolution of mobile malware," Computer Fraud & Security, vol. 2014(8), pp. 18-20, August 2014.

[8] A. T. Kabakuş, İ. A. Doğru, A. Çetin, "Android kötücül yazılım tespit ve koruma sistemleri," Erciyes Üniversitesi Fen Bilimleri Enstitüsü Dergisi, vol. 31(1), pp. 9-16, March 2015.

[9] M. Chandramohan, H. B. K. Tan, "Detection of mobile malware in the wild," IEEE Computer, vol. 45(9), pp. 65-71, January 2012,

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

[11] S. Y. Yerima, S. Sezer, G. Mcwilliams, "Analysis of bayesian classification-based approaches for android malware detection," IET Information Security, vol. 8(1), pp. 25-36, January 2014.

[12] Z. Xiaoyan, F. Juan, W. Xiujuan, "Android malware detection based on permissions," In 2014 International Conference on Information and Communications Technologies (ICT 2014), pp. 1-5, May 2014.

[13] S. Y. Yerima, S. Sezer, I. Muttik, "Android malware detection using parallel machine learning classifiers," In 2014 Eighth International Conference on Next Generation Mobile Apps, Services and Technologies, pp. 37-42, September 2014.

[14] G. Suarez-Tangil, J. E. Tapiador, P. Peris-Lopez, J. Blasco, "Dendroid: a text mining approach to analyzing and classifying code structures in Android malware families," Expert Systems with Applications, vol. 41(4), pp. 1104-1117, March 2014.

[15] A. T. Kabakus, I. A. Doğru, A. Cetin, "APK Auditor: Permission-based Android malware detection system," Digital Investigation, vol. 13, pp. 1-14, June 2015.

[16] R. S. Arslan, I. A. Doğru, N. Barışçı, "Android Mobil Uygulamalar için İzin Karşılaştırma Tabanlı Kötücül Yazılım Tespiti," Politeknik Dergisi, vol. 20(1), pp. 175-189, Haziran 2016.

[17] G. Kayabaşı, I. A. Dogru, "Mobil Uygulamaların Sınıflandırmasında Kullanılan Makine Öğrenmesi Algoritmalarının Güvenirlilik Tespiti," ISCTurkey 2016 Bildiriler Kitabı, pp. 191-195, Ekim 2016.

[18] I. Burguera, U. Zurutuza, S. Nadjm-Tehrani, "Crowdroid: behavior-based malware detection system for Android," In Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices, pp. 15-26, October 2011.

[19] Y. Lu, P. Zulie, L. Jingju, S. Yi, "Android malware detection technology based on improved bayesian classification," In 2013 Third International Conference on Instrumentation, Measurement, Computer, Communication and Control (IMCCC), pp. 1338-1341, September 2013.

[20] M. K. Alzaylee, S. Y. Yerima, S. Sezer, "Dynalog: an automated dynamic analysis framework for characterizing android applications," In Proceedings of the 2016 International Conference On Cyber Security And Protection Of Digital Services (Cyber Security), pp. 1-8, June 2016.

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

[22] T. Yang, K. Qian, L. Li, D. Lo, L. Tao, "Static Mining and Dynamic Taint for Mobile Security Threats Analysis," In IEEE International Conference on Smart Cloud (SmartCloud), pp. 234-240, November 2016.

[23] Y. Zhou, X. Jiang, "Dissecting android malware: characterization and evolution," In 2012 IEEE Symposium on Security and Privacy, pp. 95-109, May 2012.