Erciyes University Journal of the Institute of Science and Technology
Erciyes Üniversitesi Fen Bilimleri Enstitüsü Dergisi
ISSN 1012-2354
Cilt (Volume): 31 Sayı (Issue): 1 , Mart/March-2015
http://fbe.erciyes.edu.tr/

Android kötücül yazılım tespit ve koruma sistemleri

***Abdullah Talha KABAKUŞ[1], İbrahim Alper DOĞRU[2], Aydın ÇETİN[3]**
[1] *Abant İzzet Baysal Üniversitesi, Bilgi İşlem Daire Başkanlığı*
[2,3] *Gazi Üniversitesi, Teknoloji Fakültesi, Bilgisayar Mühendisliği*

Anahtar Kelimeler: Android, kötücül yazılım tespiti, akıllı telefon, mobil güvenlik, mobil uygulama güvenliği

ÖZET

Açık kaynak kodlu ve Linux tabanlı bir mobil işletim sistemi olan Android, son raporlara göre dünyada en çok kullanılan mobil işletim sistemidir. Bu popülarite ve açık kaynak kodlu yapı sonucunda Android, kötücül saldırıların ve saldırganların hedefi haline gelmiştir. Cisco 2014 güvenlik raporuna göre mobil kötücül yazılımların %99'u Android işletim sistemini hedef almaktadır. Android uygulamaları genellikle resmi uygulama marketi olan Play Store'dan temin edilmektedir. Play Store, çeşitli geliştiriciler tarafından yüklenen uygulamaları güvenlik taramasına tabi tutmadan yayınlamaktadır. Bunun yanı sıra kullanıcı bir uygulamayı yükleme girişiminde bulunduğu zaman Android, bu uygulamalar yüklenirken kullanıcılara uygulamanın talep ettiği izinleri sunmakta ve sonrasındaki tüm sorumluluğu kullanıcıya bırakmaktadır. Yapılan çalışmalarda Android kullanıcıların büyük bir çoğunluğunun bu izinlerden habersiz olduğu veya bu izinlerin ne tür etkilerinin olduğunu bilmediği ortaya çıkmaktadır. Bu sebeple, uygulamaların kötücül bir içerik içerip içermediğine dair bir güvenlik taraması yapılmasına ve kullanıcıların bilgilendirilmesine ihtiyaç duyulmaktadır. Bu ihtiyacı karşılamak üzere literatürde çeşitli yaklaşımlar ve yöntemler yer almaktadır. Bu çalışmada literatürdeki çeşitli Android kötücül yazılım tespit/koruma sistemleri statik, dinamil ve imza tabanlı analiz yaklaşımları ile kriptolu veri iletişimi ile koruma olmak üzere dört başlık altında incelenmiştir. Kullanılan yöntemlerin manifest incelemesi, API çağrı izlemesi, imza veritabanı, güvenli veri alışverişi ve makine öğrenmesi özellikleri karşılaştırmalı olarak sunulmuştur.

Android malware detection and protection systems

Key Words: Android, malware detection, smartphone, mobile security, mobile application security

ABSTRACT

According to recent reports, Android, an open source and Linux based operating system, is the most used mobile operating system in all over the world. Due to this popularity and being an open source software, Android has become the main target of malwares and malware developers. Cisco Security Report 2014 highlights that Android is the target of 99% of all mobile malwares. Android applications are commonly installed through the official application market – Play Store. Play Store publishes applications uploaded by different developers without putting a security test. Additionally, when a user attempts to install an application, Android displays the permissions that it demands to users and then shifts all responsibility on them. Most studies indicate that majority of Android users are not aware of these permissions or they do not understand consequences of these permissions. Hence, it is needed to analyze applications to highlight whether they contain malicious content or not and to inform users about it. In order to demand this necessity, there are various approaches and methods in the literature. In this study, four approaches; static analysis, dynamic analysis, siganature based analysis and data encryption on different Android malware detection/protection systems are investigated, and their feature comparisons based on manifest files, API call tracing, signature database and machine learning are presented.

*Sorumlu Yazar (Corresponding author) e-posta: talha.kabakus@ibu.edu.tr

1. Giriş

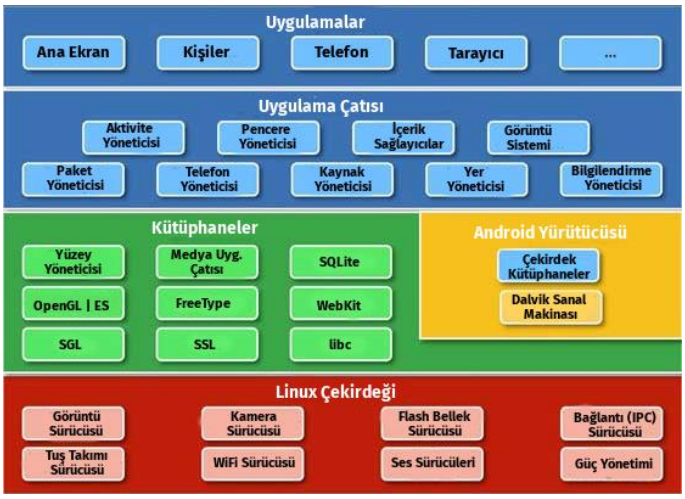
Akıllı telefon marketinde Android açık kaynak kodlu yapısı sebebiyle önemli bir pazar payı elde etmiştir. Google, 2013'ün son çeyreğinde 1 milyardan fazla Android kullanıcısının olduğunu duyurmuştur . Strategy Analytics firmasının 2013'ün son çeyreğinde yaptığı araştırmaya göre Android, akıllı telefon market dağılımı oranını bir önceki seneye göre 204.4 milyon kullanıcı sayısıyla %75'den %81.3'e çıkartmıştır [1]. Google, 2013 Temmuz ayında katıldığı bir etkinlikte Android'in resmi uygulama marketi olan Play Store 'da sunulan uygulama sayısının 1 milyonu geçtiğini duyurmuştur [2].

Cisco 'nun 2014 güvenlik raporuna göre, 2013 yılında yapılan mobil kötücül saldırıların %99'u Android'i hedeflemektedir [3]. Benzer şekilde Sophos firmasının 2014 yılında hazırladığı Mobil Güvenlik Tehdit Raporu da son bir yıl içerisinde Android kötücül yazılım sayısında 6 kat artış olduğunu ortaya koymaktadır [4]. Android tabanlı kötücül yazılımlar, kullanıcıların kişisel bilgilerini toplamak (grayware) ve yetkisiz işlem yapmak (malware) gibi kötücül hedeflere sahiptir. Play Store'daki kötücül yazılım oranı yapılan çalışmalara göre son yıllarda ciddi bir artış göstermektedir [5,6]. Bu artışın başlıca sebepleri olarak Android'in açık kaynak kodlu yapısı ve uyguladığı pasif koruma yöntemi gösterilebilir. Android, izin tabanlı güvenlik mekanizmasına sahiptir [7,8]. İzinler, uygulamaların telefonun Short Message Service (SMS) gönderimi, yer bildirimi gibi çeşitli kaynaklarını kullanabilmesi için ihtiyaç duyduğu onaylardır. Kullanıcılar uygulamaları yüklerken Android uygulamanın kullanıcından talep ettiği izinleri listelemektedir. Bu izinler uygulamanın sahip olacağı kabiliyetleri kullanıcıya bildirmek için kullanılmaktadır [9]. Bu aşamadan sonra kullanıcı kendi değerlendirmesine yüklemeye devam edebilmekte veya yüklemeyi iptal edebilmektedir. Üstelik bu izinler, yükleme işlemi tamamlandıktan sonra kullanıcıya bir daha sunulmamaktadır [10]. Yükleme esnasında talep edilmeyen izinler, uygulama tarafından kullanılması durumunda uygulama başarısız olmaktadır [11]. Felt ve arkadaşları tarafından yapılan çalışmada kullanıcıların yalnızca %17'sinin bu izinleri dikkate aldığı ve %42'sinin ise izinler hakkında bilgisinin olmadığı ortaya çıkmıştır [10]. Yapılan diğer çalışmalarda da Android izin bildirimlerinin kullanıcılar tarafından yok sayıldığı ortaya çıkmaktadır [8], [12-15].

Literatürde bu eksiklikleri hedef alan çeşitli Android kötücül yazılım tespit ve koruma sistemleri mevcuttur. Bu sistemler, statik analiz yaklaşımı, dinamik analiz yaklaşımı, imza tabanlı analiz ve koruma ve kriptolu veri iletişimi ile koruma olmak üzere 4 başlık altında gruplandırılmıştır. Makale şu şekilde yapılandırılmıştır: 2. bölümde izin mekanizmasının daha iyi anlaşılabilmesi için Android sistem mimarisi hakkında bilgi sunulmuştur. 3. bölümde statik analiz yaklaşımı, dinamik analiz yaklaşımı, imza tabanlı analiz ve koruma ve kriptolu veri iletişimi ile kötücül yazılım tespit ve koruma mekanizmaları ele alınmıştır. 4. bölümde bu yöntemler karşılaştırılmalı olarak sunulmuştur. 5. bölümde ise sonuçlar ve değerlendirmelere yer verilerek geliştirilebilecek olası koruma yöntemleri önerilmiştir.

2. Android sistem mimarisi

Android, katmanlı bir sistem mimarisine sahiptir. Bu sistem mimarisine ait katmanlar ve katmanların içerdikleri bileşenler Şekil 1'de sunulmuştur.



Şekil 1. Android sistem mimarisi [16]

Android, en alt katmanında telefon donanımı ile iletişim halinde olan özelleştirilmiş Linux işletim sistemine (Linux Kernel) sahiptir. Orta katmanda Java tabanlı kütüphaneler ve uygulama çatısı (Application Framework) bulunmaktadır. En üst katman olan uygulama katmanında (Applications) ise Android Application Programming Interface (API) kullanılarak geliştirilmiş son kullanıcı ile etkileşimde bulunulan yazılımlar bulunmaktadır. Uygulamalar tarafından talep edilen SMS gönderimi, yer bildirimi, harici belleğe okuma/yazma gibi izinler, uygulama çatısı aracılığıyla yetkilendirilmekte ve sistem kaynaklarıyla haberleştirilmektedir. Örnek olarak, Android üzerinde yer bildiriminin gerçekleştirilmesi için ACCESS_COARSE_LOCATION veya ACCESS_FINE_LOCATION izinlerinden en az birisinin uygumalar tarafından talep edilmesi gerekir. Kullanıcıların bu izin/izinleri onaylanması durumunda orta katmandaki uygulama çatısı bileşenlerinden olan Yer Yönetimi (Location Manager) servisi, yer erişim bilgisini sağlamaktadır.

Android uygulamaları apk uzantılı paket dosyaları şeklinde sunulmaktadır. Bu paket dosya içerisinde uygulamaya ait tüm kaynaklar (kaynak kodlar, resim dosyaları, sabit değer tanımlamaları gibi) bulunmaktadır. Her Android uygulaması, uygulamanın temel bilgi ve izinlerini içeren manifest dosyası (AndroidManifest.xml) barındırmaktadır [17]. Bu dosya içerisinde uygulamanın adı, paket ismi, versiyonu, uygulamanın sahip olduğu aktiviteler (Activity), servisler (Service), ve alıcılar (Receiver), uygulamanın talep ettiği izinler (Permission) bulunmaktadır. Android uygulamaları Java programlama dili kullanılarak geliştirilmektedir. Ancak Dalvik Virtual Machine (DVM) adı verilen bir sanal makinada yürütülmektedir.

Şekil 2'de Android uygulamalarının yazılımsal yığın yapısı sunulmuştur. Java sınıfları derlendikten sonra her bir sınıf için bir .class dosyası oluşturulur ve Java Virtual Machine (JVM) byte kodu oluşturulur. Dalvik dx derleyicisi ise bu kodları tekrar derler ve tüm sınıfları içeren tek bir .dex dosyası oluşturur. Bu dosya DVM üzerinde yürütülmektedir. Her uygulama kendi DVM'i üzerinde çalıştırılmaktadır.

Vidas ve arkadaşları [20], kaynak kod üzerinde uygulama izin analizi yapmak için oldukça yaygın Java programlama tümleşik geliştirme ortamı (IDE) olan Eclipse üzerine bir eklenti geliştirmişlerdir. Ancak bu araç, uygulama izinlerinin kullanım amacını sorgulamadığından olası kötücül içerikleri öne çıkarmamaktadır.

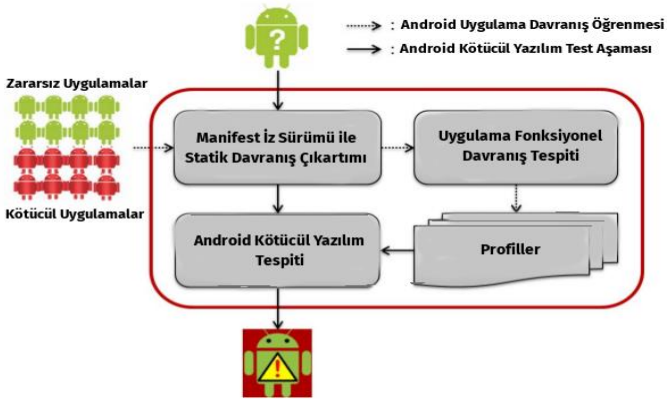
Şekil 2. Android uygulamalarının yazılım yığın yapısı [18]

3. Android Kötücül Yazılım Tespit Ve Koruma Mekanizmaları

Android kötücül yazılım çeşit ve sayısı arttıkça koruma yöntemleri de paralelinde artmakta ve çeşitlenmektedir. Literatürde yer alan kötücül yazılım tespit ve koruma sistemleri statik (durgun) analiz yaklaşımı, dinamik analiz yaklaşımı, imza tabanlı analiz ve koruma ve kriptolu veri iletişimi olmak üzere 4 başlık altında gruplandırılmıştır.

3.1. Statik analiz yaklaşımı

Statik analiz yaklaşımı, uygulamaların kötücül yazılım tespit ve korumalarını cihazlara yüklenmeden uygulamaların sundukları veriler aracılığıyla yapılmasını sağlamaktadır. Bu yaklaşımın en önemli avantajı, kötücül yazılımların cihaza yüklenmeden tespit ve korumasını sağlamasıdır. Böylelikle cihaz, kötücül yazılım içeriğinden etkilememektedir.



Şekil 3. *DroidMat* mimarisi [19]

DroidMat [19], manifest dosyası ve ilgili izinlere yönelik API çağrıları üzerinden Android kötücül yazılım tespiti yapmayı sağlayan bir araçtır. Şekil 3'de *DroidMat*'ın mimari yapısı sunulmuştur. Şekilde sunulduğu üzere manifest dosyası aracılığıyla statik davranış analiz çıkartımı yapılmaktadır. Statik davranış analizleri elde edildikten sonra uygulama fonksiyonel davranış analizi safhasına geçilmektedir. Bu analizler elde edikten sonra ise K-means algoritması ile kötücül yazılım modelleri oluşturulmaktadır. Oluşacak küme sayısı Singular Value Decomposition (SVD) algoritması ile belirlenmektedir. Son olarak k=1 olmak üzere kNN (k Nearest Neighbours) algoritması ile üzerinde çalışılan uygulamanın kötücül olup olmadığı tespit edilmektedir.

Drebin [21] statik analiz ve makine öğrenmesi yaklaşımlarını birleştirerek Android kötücül yazılım tespitinde bulunmaya çalışan bir araçtır. *Drebin*, uygulamanın kaynak kodunu ve manifest dosyasını kullanarak uygulamaya ait izinler, API çağrıları ve ağ adresleri gibi çeşitli özellikleri toplamaktadır. Bu özellikler, bir birleşik uzay vektörüne gömülerek çeşitli desenler aracılığıyla Android kötücül yazılım tespiti için kullanılmaktadır.

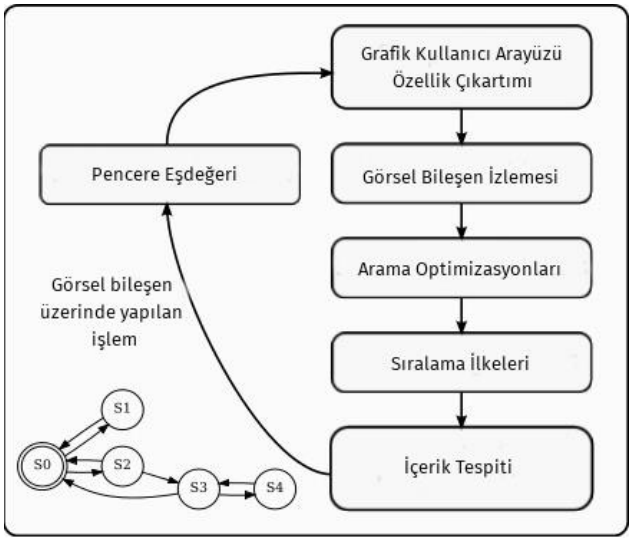
Felt ve arkadaşları [22] *Stowaway* adında gereğinden fazla yetki talep eden Android uygulamalarını tespit eden bir sistem öne sürmüşlerdir. *Stowaway*, bir uygulamanın kullandığı API çağrı setini belirlemekte ve bu çağrıları izinlerle eşleştirmektedir. *Stowaway* iki kısımdan oluşmaktadır: Birinci kısım, uygulamanın kullandığı API çağrılarını belirlemektedir. İkinci kısım ise izinleri eşleştirerek her bir API çağrısı için gerekli olan izinlerin tespit edilmesini sağlamaktadır. Böylelikle gereğinden fazla yetki/izin talep eden uygulamalar bu sistem aracılığıyla belirlenmektedir. Ayrıca çalışmada "en yaygın geliştirici hataları" şu şekilde belirtilmiştir: (1) Yanlış izin ismi kullanımı, (2) vekil kullanımı, (3) ilgili metotların kullanımı esnasındaki koruma seviyesi ihlali, (4) kullanımdan kaldırılmış (deprecated) izinlerin yer alması, (5) imza (Signature) ve/veya sistem (System) izinlerinin kullanılması, (6) test esnasında yapılan değişikliklerin unutulması ve (7) kopyala-yapıştır yöntemiyle gereksiz izin talebi.

3.2. Dinamik analiz yaklaşımı

Dinamik analiz yaklaşımının statik analiz yaklaşımından en önemli farkı, uygulama analizlerinin yürütme zamanında (runtime) yapılmasıdır. Dinamik analiz yaklaşımının en önemli avantajı, statik analiz aracılığıyla ortaya çıkartılamayacak karmaşıklıktaki eksiklik veya açıkların ortaya çıkartılmasını sağlamasıdır [23].

Şekil 4. *Crowdroid* Android kötücül yazılım tespit mekanizması [24]

Burguera ve arkadaşları [24] *Crowdroid* isimli Android uygulamaları üzerinde anormal davranış tespiti gerçekleştiren bir uygulama çatısı sunmaktadır. *Crowdroid*, Linux tabanlı bir araç olan Strace komutunu kullanarak Android çekirdek (kernel) sistem çağrılarını derleyerek Android uygulamalarını "zararsız" veya "kötücül" olarak sınıflandırmaktadır. Şekil 4'de *Crowdroid*'in Android kötücül yazılım tespit mekanizması sunulmuştur. Temel olarak bu mekanizma, veri (cihaz bilgileri, yüklü uygulama listesi ve izleme kayıtları) toplanması, veri işlenmesi, kötücül yazılım analizi ve tespiti aşamalarından



oluşmaktadır.

Şekil 5. AppsPlayground akıllı yürütme modülü ön izlemesi [25]

Rastogi ve arkadaşları [25] geliştirdikleri *AppsPlayground* isimli bir araç ile akıllı telefon uygulamalarının otomatik dinamik analizini gerçekleştirmektedir. *AppsPlayground*, efektif analiz ortamı sunabilmek için çekirdek katmanı izlemesi, API çağrı izleri, olay tetikleme, akıllı yürütme gibi çeşitli tespit, tetkik ve maskeleme tekniklerini kullanmaktadır. Şekil 5'de *AppsPlayground* akıllı yürütme modülünün ön izlemesi sunulmuştur.

Zhou ve Jiang [26] Android kötücül yazılım tespitine yönelik sistematik bir yaklaşım sunmaktadır. Çalışmada 49 farklı kötücül yazılım ailesinden elde edilen 1260 Android kötücül yazılımı içeren veri seti kullanılmıştır. Zhou ve Jiang, kötücül yazılımları yükleme, aktivite ve taşınan veri gibi çeşitli davranış biçimlerine göre karakterize etmektedir.

RiskRanker [27], uygulamaların üst seviye yetkiler edinme, arka planda SMS gönderilmesi gibi tehlikeli davranışlar sergileyip sergilemediğini analiz eden özdevimli bir sistemdir. *DroidMOSS* [28], uygulama marketlerdeki uygulamalara kötücül içerik eklenerek tekrar paketlenmesi yoluyla yapılan saldırıları tespit etmeyi amaçlayan bir sistemdir.

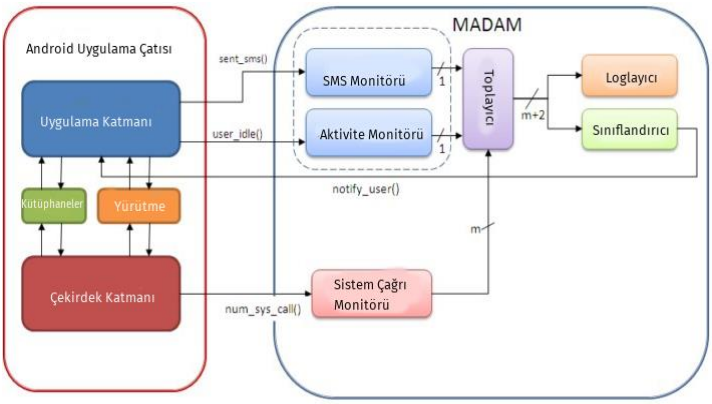
Paranoid Android [29] Android uygulamalarına yönelik güvenlik taramalarını, akıllı telefonların sanal ortamda birebir kopyaları üzerinde gerçekleştirmektedir. Sunucuların kullanılması çoklu tespit tekniklerinin eş zamanlı olarak uygulanabilme fırsatı vermektedir. Ayrıca Paranoid Android, uygulamalara yönelik olarak yaptığı virüs taramasını mobil cihazlar yerine sunucularda yapması sonucunda yazılımsal maliyeti düşürmektedir.

CopperDroid [30], düşük seviye işletim sistemine özel ve yüksek seviye Android'e özel uygulama davranışlarını QEMU adı verilen emülatör üzerinde karakterize ederek dinamik davranış analizi yapmaktadır. Bu uygulama davranış analizleri, sistem çağrı talepleri üzerinden gözlemlenmektedir. Böylelikle *CopperDroid* Java programlarından, Java Native Interface (JNI) üzerinden veya öz (native) kod yürütmelerinden davranış analizi yapabilmektedir.

3.3. İmza tabanlı analiz ve koruma

İmza tabanlı analiz ve koruma sistemlerinde analiz edilen uygulamalar, imza veritabanında saklanarak sürekli öğrenmeye dayalı bir yaklaşım sergilenmektedir. Bu yaklaşım genelde merkezi bir sunucu ve imza veritabanı kullanılarak gerçekleştirilmektedir. Merkezi sunucu analiz ve koruma süreçlerinin gerçekleştirilmesinde görev alırken, veritabanı sunucusu elde edilen analizlerin saklanmasını ve sonraki analizlerde tekrar kullanılmasını sağlamaktadır.

Guido ve arkadaşları [31] akıllı telefonda yürütülmek ve merkezi bir sunucuya Wi-Fi üzerinden bilgi aktarmak için *Tractor Beam* adı verilen bir servis geliştirmişler. Bu servis ile aralıklarla bit değişimlerini ve ofset değerlerini merkezi sunucuya Wi-Fi üzerinden gönderilmektedir. *Tractor Beam*, blok cihazlar yerine sadece bit değişimlerini gösteren Secure Hash Algorithm-256 (SHA-256) hash'ler saklandığından disk alanı açısından da sıkıntı oluşturmamaktadır. Merkezi sunucu, *Tractor Beam* aracılığıyla gönderilen bit değişimlerini ve ofset değerlerini normalize edilmiş ilişkisel veritabanında saklamaktadır. Merkezi sunucu, sistem imajları oluşturmakta ve bu imajları analiz uygulama çatısı aracılığıyla etmektedir. Analiz uygulama çatısı, detektörler ve loglayıcılardan oluşmaktadır. Detektörler, kötücül yazılım tespit tekniklerini barındırırken loglayıcılar kötücül olabilecek tüm aktiviteleri kayıt altına almaktadır.

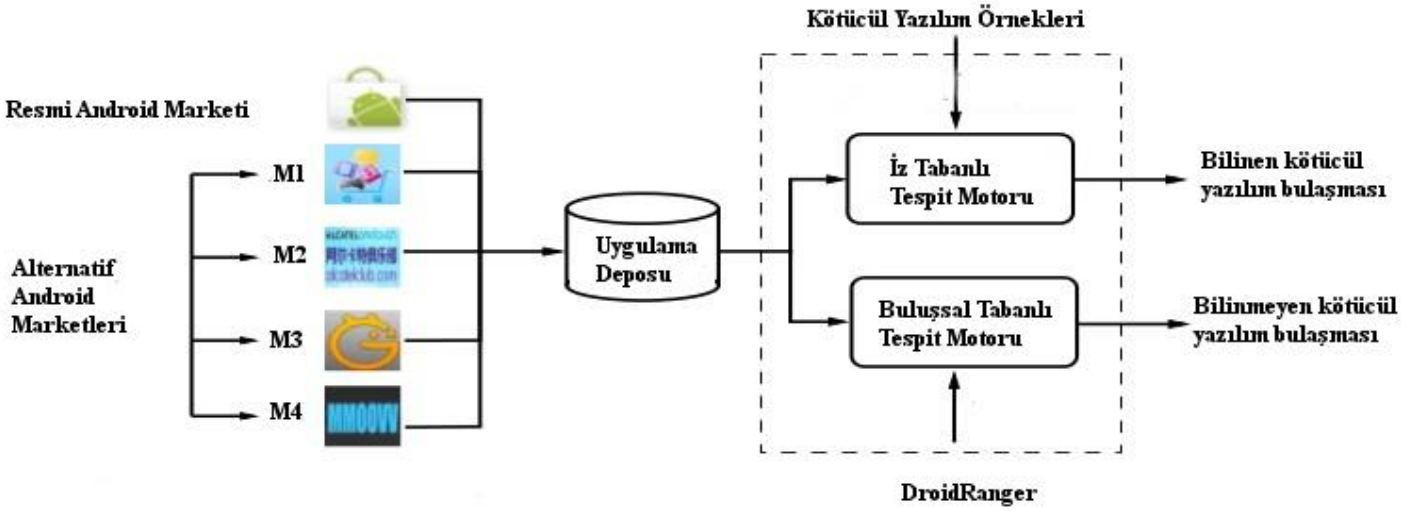


Şekil 6. MADAM fonksiyon blokları [32]

MADAM [32] Android kötücül yazılımlar için çok seviyeli aykırılık tespit aracıdır. MADAM, kernel (çekirdek) ve uygulama seviyelerinde kötücül yazılım tespiti gerçekleştirmektedir. Sistem, kernel seviyesinde kullanıcı aktivitelerini, dosya ve bellek erişimlerini, gelen-giden veri trafiğini, enerji tüketimini ve sensör durumunu izlemeyi sağlayan sistem çağrılarından faydalanmaktadır. Uygulama seviyesi tespitleri ise çıkartılan özellikler kullanıcının boşta olup olmadığını, gönderilen-alınan SMS sayısını ve Bluetooth/Wi-Fi analizlerini belirlemekte kullanılmaktadır. Şekil 6'da MADAM fonksiyonel blokları sunulmuştur. Tablo 1'de ise MADAM seviye özellikleri sunulmuştur.

Tablo 1. *MADAM* seviye özellikleri

Seviye	Özellik
Kernel	Sistem çağrıları Yürütülen işlemler Boş RAM CPU kullanımı
Kullanıcı/Uygulamalar	Boş kullanım/aktif Klavye işlemleri Aranan numaralar Gönderilen/alınan SMS'ler Bluetooth/Wi-Fi analizleri

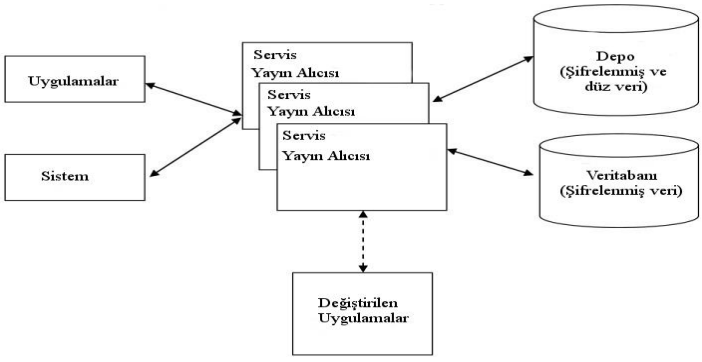


Şekil 7. DroidRanger yazılım mimarisi [6]

DroidRanger [6] sunduğu izin tabanlı davranış iz sürümü şeması ile bilinen kötücül yazılım ailelerinin yeni örneklerinin tespit edilmesini amaçlamaktadır. Sonrasında ise buluşsal yöntemler tabanlı filtreleme şeması uygulanarak bilinmeyen kötücül yazılım ailelerinin kalıtsal davranış biçimleri ortaya çıkartılmaktadır. Şekil 7'de *DroidRanger*'ın yazılım mimarisi sunulmuştur.

3.4. Kriptolu veri iletişimi ile koruma yaklaşımı

Kriptolu veri iletişimi ile koruma yaklaşımı, veri iletişiminin güvenli bir şekilde sağlanmasını hedef almaktadır. Bu sayede kötücül yazılımlar tarafından sıklıkla kullanılan oluşabilecek güvenlik açıklarının önlenmesi hedeflenmektedir.



Şekil 8. Pocatilu tarafından öne sürülün kriptolu sistem mimarisi [18]

Pocatilu [18] öne sürdüğü yaklaşımda kullanıcıların SMS bilgileri, elektronik mailleri, dosyaları gibi hassas verilerini şifreleyerek veritabanında saklamaktadır. Kullanıcı sistemden veya uygulamalardan mesaj aldığı zaman bir kesme devreye girmekte ve mesaj içeriğini şifreleyerek veritabanında saklamaktadır. Saklanan veriye başka bir uygulama ihtiyaç duyduğu zaman SQLite veritabanından veriler çekilmekte ve şifreleri çözülerek ilgili uygulamalara iletilmektedir. Şekil 8'de öne sürülen sistem mimarisi sunulmuştur.

Android uygulama programlama arayüzünde bulunan javax.crypto paketi , simetrik anahtarlama (AES, DES), açık anahtarlı şifreleme (RSA, DH) ve mesaj öz fonksiyonlarına (message digest) yönelik sınıfları sunmaktadır. Şekil 9'da DES algoritması kullanılarak karakter dizilerinin şifrelenmesini ve geri çözümlenmesini sağlayan örnek Java sınıfı sunulmuştur.

Vidas ve Christin [33], *AppIntegrity* isimli yazılım geliştirici ile kullanıcı arasında kriptografik doğrulamaya yardımcı olan bir araç sunmaktadır. Bu araç, uygulama marketlerinde sunulan doğrulama mekanizmalarının güçlendirilmesini amaçlamaktadır. Bu sayede popüler bir kötücül yazılım yayılma şekli olan uygulama tekrar paketleme (application repackaging) yönteminin zorlaştırıldığı düşünülmektedir.

```
import javax.crypto.*;
import org.kobjects.base64.Base64;

class SecCD
{
 public static String decripteazaDES(String sir, SecretKey cheie)
 {
       String sirDecriptat = null;

       try
       {
        Cipher cifDecriptare = Cipher.getInstance("DES/ECB/PKCS5Padding");
        cifDecriptare.init(Cipher.DECRYPT_MODE, cheie);
        sirDecriptat = new String(cifDecriptare.doFinal(Base64.decode(sir)));
       }
       catch(Exception ex)
       { Log.e("PDM1", ex.getMessage()); }
       return sirDecriptat;
}

 public static String cripteazaDES(String sir, SecretKey cheie)
 {
       String sirCriptat = null;

       try
       {
        Cipher cifCriptare = Cipher.getInstance("DES/ECB/PKCS5Padding");
        cifCriptare.init(Cipher.ENCRYPT_MODE, cheie);
        sirCriptat = Base64.encode(cifCriptare.doFinal(sir.getBytes("UTF8")));
       }
       catch(Exception ex)
       { Log.e("PDM1", ex.getMessage()); }
       return sirCriptat;
 }
}
```

Şekil 9. Karakter dizilerinin DES algoritması kullanılarak şifrelemesini ve geri çözümlemesini sağlayan örnek Java

Khalil ve arkadaşları [34] mevcut kimlik yönetim sistemlerinin mobil bulut bilişim için açıklarından yola çıkarak *"Birleştirilmiş Kimlik Yönetim Sistemi (Consolidated Identity Management System)"* adında yeni bir mimari öneri sürmüşlerdir. Bu yeni mimarinin mevcut kimlik yönetim sistemlerinin de tespit edilen sunucu açıkları, mobil cihaz açıkları ve ağ trafik dinlemesi açıklarını giderdiği düşünülmektedir.

Play Store'da sunulan uygulamalara ait apk dosyaları temin edilebilmekte ve modifiye edilerek kötücül kod yerleştirilebilmektedir. Sonrasında bu uygulamalar farklı bir isimle yine market üzerinden veya erişime açık olan başka bir web sayfası aracılığıyla sunulmaktadır. Tüm bu çalışmalar haricinde Android uygulamaları üzerinde tekrar derleme ve modifiye işlemleri için kullanılan belli başlı araçlar aşağıdaki gibi sıralanabilir:

•APKTool – apk dosyalarının derlenmesi ve tekrar derleme işlemleri için kullanılmaktadır.
•smali – dex dosyaları için çevirici ve geri çevirici görevi görmektedir.
•dex2jar, jad – dex dosyalarından jar dosyaları elde edilmesini sağlamaktadır.
•JD-GUI – Derlenmiş Java sınıflarının (class dosyaları) okunabilir Java kaynak kodlarına dönüştürülmesini sağlayan arayüz programıdır.

4. Sistem karşılaştırmaları

Kötücül yazılımların çok çeşitli olmasından ötürü tek bir koruma ve tespit yöntemi, tüm kötücül yazılımların tespiti ve koruması için yeterli olmamaktadır. 3. bölümde sunulan her bir yaklaşım kendine özgü bir kötücül yazılım koruma ve tespit yöntemi barındırmaktadır. Bu yaklaşımların kötücül yazılım tespit ve koruma özellikleri karşılaştırmalı olarak Tablo 2'de sunulmuştur.

Tablo 2. Kötücül yazılım tespit ve koruma sistemlerinin özellik karşılaştırması

Özellik	Drebin	Crowdroid	MADAM	DroidMat	Julia
Manifest incelemesi	Var	Yok	Var	Var	Var
API çağrı izlemesi	Var	Var	Var	Var	Var
İmza veritabanı	Yok	Var	Var	Var	Yok
Güvenli veri alışverişi	Yok	Yok	Yok	Yok	Yok
Makine öğrenmesi	Var	Var	Var	Var	Yok

Tablo 2'de görüleceği üzere incelenen sistemlerin hiç birisi tanımlanan 5 özelliğin tamamına sahip değildir. Bu sistemler içerisinden sadece *MADAM* ve *DroidMat* benzer şekilde temel aldığımız beş özellikten dördüne sahip olmasına karşın kötücül yazılım tespit ve koruma sistemlerinin hiç birisinin güvenli veri alışverişi sağla(ya)mamaktadır. Diğer yandan, bu sistemlerin tamamı API çağrı izlemesi tespit ve koruma özelliğine sahip oldukları görülmektedir. Ayrıca sistemlerin 4/5 inin tespit ve koruma sistemlerinde makine öğrenmesini kullandıkları görülmektedir.

5. Sonuç ve değerlendirmeler

Android, dünya genelinde en çok kullanılan mobil işletim sistemi olma özelliğini devam ettirmektedir. Android'in resmi uygulama marketi olan Play Store'a her gün binlerce yeni uygulama yüklenmektedir. Bu popülarite beraberinde Android'i kötücül yazılımların/geliştiricilerin hedefi haline getirmiştir. Yapılan araştırmalara göre mobil tabanlı kötücül saldırıların %99'u Android tabanlıdır. Android'in kötücül yazılımlar tarafından hedef haline gelmesinde tespit edilen temel etkenler şöyle sıralanabilir: (1) Android resmi uygulama marketi olan Play Store, yüklenen uygulamalara yönelik pasif koruma sergilenmesi. (2) Uygulamaların talep ettiği izinlerin onaylanmasının son kullanıcıya bırakılması ve son kullanıcılarının büyük bir çoğunluğunun bu izinlerden habersiz olması. (3) Açık kaynak kodlu yapı. (4) Popüler programlama dillerinden biri olan Java desteği.

Bu çalışmada, çeşitli açılardan Android kötücül yazılım tespit ve korumasında bulunan sistemler karşılaştırmalı olarak incelenmiştir. İnceleme sonucunda sistemlerin güvenli veri alışverişi konusunda herhangi bir özellik geliştirmedikleri ancak, kötücül yazılım tespit ve koruma konusunda makine öğrenmesi yöntemlerini kullanma eğiliminde oldukları veya makine öğrenmesini tercih ettikleri görülmektedir.

Bu sistemler genelde uygulamanın sisteme yüklenmesinden sonra çalışmakta, kurulum öncesi tespit ve koruma mekanizması kullanmamaktadır. Bu nedenle, Android uygulamaları sisteme yüklenmeden önce kötücül yazılım tespitinde bulunarak kullanıcıları yönlendirecek daha kapsamlı kötücül yazılım tespit ve koruma sistemlerine ihtiyaç duyulmaktadır. Geliştirilecek bu sistemler aracılığıyla güvenlik açıklarının temelini oluşturan izinlerin, uygulama kaynak kodu üzerinden kullanım amacının analiz edilmesi önem arz etmektedir. Bu şekilde geliştirilebilecek bir uygulama ile talep edilen izinlerin kötücül bir amaç taşıyıp taşımadığı anlaşılabilir ve kullanıcının veya sistemin gerekli önlemleri alması sağlanabilir.

Kaynaklar

1. S. Bicheno, “Android Captures Record 81 Percent Share of Global Smartphone Shipments in Q3 2013,” Strategy Analytics, 2013. [Web].
2. D. Rowinski, “Google Play Hits One Million Android Apps,” ReadWrite, 2013. [Web]. Erişim Adresi: http://readwrite.com/2013/07/24/google-play-hits-one-million-android-apps. [Erişim Tarihi: 10.10.2014].
1. “Cisco 2014 Annual Security Report,” Cisco, 2014. [Web]. Erişim Adresi: https://www.cisco.com/web/offer/gist_ty2_asset/Cisco_2014
2. _ASR.pdf
3. “Mobile Security Threat Report,” Sophos, 2014. [Web]. Erişim Adresi: http://www.sophos.com/en-us/threat-center/mobile-security-threat-report.aspx?utm_source=Non-campaign&utm_medium=Cross-link&utm_campaign=CL-CorpBlog. [Erişim Tarihi: 21.09.2014].
4. P. Felt, M. Finifter, E. Chin, S. Hanna, and D. Wagner, “A survey of mobile malware in the wild,” in SPSM ’11 Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices, 2011, pp. 3–14.
5. Y. Zhou, Z. Wang, W. Zhou, and X. Jiang, “Hey, You, Get Off of My Market: Detecting Malicious Apps in Official and Alternative Android Markets,” in Proceedings of the 19th Annual Network and Distributed System Security Symposium (NDSS), 2012.
6. Barrera, P. C. Van Oorschot, and A. Somayaji, “A Methodology for Empirical Analysis of Permission-Based Security Models and its Application to Android Categories and Subject Descriptors,” in Proceedings of 17th ACM Conference on Computer and Communications Security, 2010, pp. 73–84.
7. P. Felt, K. Greenwood, and D. Wagner, “The effectiveness of application permissions,” in Proceeding of the WebApps’11 Proceedings of the 2nd USENIX conference on Web application development, 2011, p. 7.
8. “Android Security Overview,” Google. [Web]. Available: http://source.android.com/devices/tech/security/. [Erişim Tarihi: 03.04.2014].
9. P. Felt, E. Ha, S. Egelman, A. Haney, E. Chin, and D. Wagner, “Android permissions: User Attention, Comprehension, and Behavior,” in Proceedings of the Eighth Symposium on Usable Privacy and Security - SOUPS ’12, 2012, p. 1.
10. Shabtai, Y. Fledel, U. Kanonov, Y. Elovici, S. Dolev, and C. Glezer, “Google Android: A Comprehensive Security Assessment,” IEEE Secur. Priv. Mag., vol. 8, 2010.
11. W. Enck, M. Ongtang, and P. Mcdaniel, “On Lightweight Mobile Phone Application Certification,” in ACM conference on Computer and communications security, 2009, pp. 235–245.
12. W. Enck, M. Ongtang, and P. McDaniel, “On lightweight mobile phone application certification,” in Proceedings of the 16th ACM conference on Computer and communications security - CCS ’09, 2009, pp. 235–245.
13. P. G. Kelley, S. Consolvo, L. F. Cranor, J. Jung, N. Sadeh, and D. Wetherall, “A Conundrum of Permissions: Installing Applications on an Android Smartphone,” in Financial Cryptography and Data Security, vol. 7398, J. Blyth, S. Dietrich, and L. J. Camp, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 68–79.
14. J. King, A. Lampinen, and A. Smolen, “Privacy: is there an app for that?,” in Proceedings of the Seventh Symposium on Usable Privacy and Security - SOUPS ’11, 2011, p. 1.
15. “Android Architecture,” tutorialspoint, 2014. [Web]. Erişim Adresi: http://www.tutorialspoint.com/android/android_architecture.htm. [Erişim Tarihi: 10.10.2014].
16. S. Mansfield-Devine, “Android architecture: Attacking the weak points,” Netw. Secur., vol. 2012, pp. 5–12, 2012.

17. P. Pocatilu, "Android applications security," Inform. Econ., vol. 15, pp. 163–171. Retrieved from http://revistaie.ase.ro, 2011.
18. D.-J. Wu, C.-H. Mao, T.-E. Wei, H.-M. Lee, and K.-P. Wu, "DroidMat: Android Malware Detection through Manifest and API Calls Tracing," in 2012 Seventh Asia Joint Conference on Information Security, 2012, pp. 62–69.
19. T. Vidas, N. Christin, and L. F. Cranor, "Curbing Android Permission Creep," in In Proceedings of the 2011 Web 2.0 Security and Privacy Workshop (W2SP 2011), 2011.
20. D. Arp, M. Spreitzenbarth, H. Malte, H. Gascon, and K. Rieck, "Drebin: Effective and Explainable Detection of Android Malware in Your Pocket," in Symposium on Network and Distributed System Security (NDSS), 2014, pp. 23–26.
21. P. Felt, E. Chin, S. Hanna, D. Song, and D. Wagner, "Android permissions demystified," in Proceedings of the 18th ACM conference on Computer and communications security - CCS '11, 2011, p. 627.
22. "Dynamic Analysis vs. Static Analysis," Intel, 2013. [Web]. Erişim Adresi: https://software.intel.com/sites/products/documentation/doclib/
23. iss/2013/inspector/lin/ug_docs/GUID-E901AB30-1590-4706-94B1-9CD4736D8D2D.htm [Erişim Tarihi: 09.10.2014].
24. Burguera, U. Zurutuza, and S. Nadjm-Tehrani, "Crowdroid: behavior-based malware detection system for Android," Science (80-.)., pp. 15–25, 2011.
25. V. Rastogi, Y. Chen, and W. Enck, "AppsPlayground : Automatic Security Analysis of Smartphone Applications," in CODASPY '13 Proceedings of the third ACM conference on Data and application security and privacy, 2013, pp. 209–220.
26. Y. Zhou and X. Jiang, "Dissecting Android Malware: Characterization and Evolution," in 2012 IEEE Symposium on Security and Privacy, 2012, pp. 95–109.
27. M. Grace, Y. Zhou, Q. Zhang, S. Zou, and X. Jiang, "RiskRanker: Scalable and Accurate Zero-day Android Malware Detection," in Proceedings of the 10th international conference on Mobile systems, applications, and services - MobiSys '12, 2012, pp. 281–294.
28. W. Zhou, Y. Zhou, X. Jiang, and P. Ning, "Detecting repackaged smartphone applications in third-party android marketplaces," in Proceedings of the second ACM conference on Data and Application Security and Privacy - CODASKY '12, 2012, pp. 317–326.
29. G. Portokalidis, P. Homburg, K. Anagnostakis, and H. Bos, "Paranoid Android: Versatile Protection For Smartphones," in Annual Computer Security Applications Conference (ACSAC), 2010, pp. 347–356.
30. Reina, A. Fattori, and L. Cavallaro, "A System Call-Centric Analysis and Stimulation Technique to Automatically Reconstruct Android Malware Behaviors," in Proceedings of the 6th European Workshop on System Security (EuroSec), 2013.
31. M. Guido, J. Ondricek, J. Grover, D. Wilburn, T. Nguyen, and A. Hunt, "Automated identification of installed malicious Android applications," Digit. Investig., vol. 10, pp. 96–104, 2013.
32. G. Dini, F. Martinelli, A. Saracino, and D. Sgandurra, "MADAM: A Multi-level Anomaly Detector for Android Malware," in Computer Network Security, vol. 7531, I. Kotenko and V. Skormin, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 240–253.
33. T. Vidas and N. Christin, "Sweetening android lemon markets: measuring and combating malware in application marketplaces," in Proceedings of the third ACM conference on Data and application security and privacy - CODASPY '13, 2013, p. 197.
34. I. Khalil, A. Khreishah, and M. Azeem, "Consolidated Identity Management System for secure mobile cloud computing," Comput. Networks, Mar. 2014.