DÜMF Mühendislik Dergisi

web: http://dergipark.gov.tr/dumf

Derleme Makalesi / Review Article

Android İşletim Sisteminde Kötücül Yazılım Tespit Sistemleri

Abdullah DAĞLIOĞLU *
Gazi Üniversitesi, Bilişim Enstitüsü Bilgisayar Bilimleri, Ankara
abdullah.daglioglu@gazi.edu.tr ORCID: 0000-0002-0949-0160, Tel: (90) 555 704 07 83cihaz

İbrahim Alper DOĞRU
Gazi Üniversitesi, Teknoloji Fakültesi Bilgisayar Mühendisliği Bölümü, Ankara
iadogru@gazi.edu.tr ORCID: 0000-0001-9324-7157

Geliş: 30.04.2019, Revizyon: 22.08.2019, Kabul Tarihi: 05.09.2019

Öz

Günümüzde bilgisayarların erişilemez olduğu durumlarda mobil cihazlar bilgiye erişmek için önemli bir araç haline gelmiştir. Akıllı mobil cihazlar birçok farklı alanda farklı amaçlarla kullanılmaktadır. Sosyal ağlar gibi internete sürekli bağlılık gerektiren uygulamalarda önemli bir büyüme olduğundan bu durum internet trafiğini olumsuz etkilemiştir ve potansiyel ağ tıkanıklığına neden olmuştur. Bu popülerlik akıllı mobil cihazları kötücül yazılımların hedefi haline getirmiştir. Statista firmasının 2018'in ikinci çeyreğinde yapmış olduğu araştırmaya göre 2009 yılı ile 2018 yılları arasında dünya çapında satılan akıllı telefonların %88'i Android işletim sistemine sahip cihazlardır (Statista, 2018). Android, Google tarafından geliştirilen Linux tabanlı açık kaynak bir işletim sistemidir. Android izin tabanlı bir güvenlik mekanizmasına sahiptir. Android işletim sisteminin izin tabanlı güvenlik mekanizmasına sahip olması ve Google tarafından yeterli bir güvenlik taramasının olmaması bu işletim sistemini kötücül yazılım geliştiricilerin hedefi haline getirmiştir. Kötücül yazılımları tespit ederek kullanıcıları kötücül yazılımlardan korumak amacıyla literatürde birçok çalışma yapılmıştır. Bu çalışmada, Android yazılım mimarisi hakkında temel bilgilere yer verildikten sonra Android işletim sisteminde kötücül yazılım tespit ve koruma yöntemlerinden bahsedilip ardından literatürde yer alan çalışmalar incelenerek yapılan çalışmaların başarımları değerlendirilmiştir. İncelenen çalışmalar arasında kötücül yazılım tespitinde en yüksek başarı oranına sahip olan çalışmanın %98,32 ile Wang ve diğerleri (2015) tarafından hibrid analiz yöntemi kullanılarak geliştirilen sisteme ait olduğu gözlenmiştir.

Anahtar Kelimeler: *Android İşletim Sistemi, Android izin, kötü niyetli yazılım tespiti, uygulama güvenliği, mobil cihazlar, akıllı cihaz*

* Yazışmaların yapılacağı yazar

DOI: 10.24012/dumf.559205

Giriş

Günümüzde akıllı mobil cihazların gelişimiyle bilgiye her yerden erişmek oldukça kolay hale gelmiştir. Bu durum insanlar arasında akıllı telefon kullanım oranını oldukça arttırmıştır. Mobil cihazların artmasıyla kullanıcılar bu cihazları kişisel iletişim, veri depolama, multimedya, gerekli bilgiye ulaşma ve eğlence gibi çeşitli amaçlarla kullanmaktadırlar. Mobil cihazların bu kadar çeşitli sektörde kullanımı sebebiyle hem internet trafiği hem de mobil kötücül yazılımların sayısı artmıştır. Statista firmasının 2018'in ikinci çeyreğinde yapmış olduğu araştırmaya göre 2009 ile 2018 yılları arasında dünya çapında satılan akıllı telefonların %88'i Android işletim sistemine sahiptir (Statista, 2018). International Data Corporation (IDC) firmasının Eylül 2018 raporunda 2019 yılında akıllı telefon kullanım oranının 2018 yılına göre %3,7 artacağı belirtilmektedir (IDC, 2018). Statista istatistik firmasının 2018'de yapmış olduğu araştırmaya göre Google Play Store'deki uygulama sayısı 2.6 milyonun üzerindedir (Statista, 2018).

Android, Google tarafından geliştirilen açık kaynak kodlu ve Linux tabanlı bir mobil işletim sistemidir. F-Secure güvenlik firmasının F-Secure Siber Güvenlik Durumu 2017 raporuna göre mobil kötücül saldırıların %99'undan fazlası Android cihazları hedeflemektedir (F-Secure, 2017). Aynı raporda Android'in açık kaynak bir işletim sistemi olmasından dolayı özellikle Android cihazları hedefleyen 19 milyondan fazla kötücül yazılımın geliştirildiği belirtilmektedir (F-Secure, 2017). G Data firmasının Eylül 2018 raporuna göre 2018'in ilk çeyreğinde yapılan çalışmada 846916 yeni kötücül yazılım tespit edilmiştir (G Data, 2018). Bu araştırmaya göre 2017'nin ilk çeyreğine oranla 2018'in ilk çeyreğinde kötücül yazılımın %12 oranında arttığı gözlenmiştir (G Data, 2018). Ayrıca bu rapora göre her gün ortalama 9411 kötücül yazılım tespit edilmektedir (G Data, 2018). Android mobil kötücül yazılımlar, kullanıcılar adına yetkisiz işlem yapmak (malware) ve kullanıcıların kişisel verilerini bilgi amaçlı toplamak (grayware) üzere iki temel amaca sahiptirler. Android İşletim Sistemi'nin açık kaynak kodlu olması, Android cihazların kullanım oranının diğer cihazlara oranla fazla olması, Android'in izin tabanlı güvenlik mekanizmasına sahip olması, resmi dağıtım kanalı olan Google Play Store'de yeteri kadar güvenlik taraması olmaması ve SlideMe-PandaApp gibi üçüncü parti uygulama dağıtım kanallarından uygulamaların dağıtılabiliyor olması Android mobil kötücül yazılımların artma sebepleri olarak gösterilebilir. İzinler, uygulamaların cihazın veya kullanıcının çeşitli kaynaklarını kullanmak amacıyla ihtiyaç duyduğu onaylardır. Android 6.0 Marshmallow sürümünden önce kullanıcılar uygulamaları yüklerken uygulamaların kullandığı izinler Android tarafından kullanıcılara listelenmekteydi ve uygulamaların kurulması için kullanıcının bu izinleri onaylayarak kuruluma devam etmesi beklenmekteydi. Kullanıcı kendi cihazının güvenliğini kontrol edememekteydi. Bu durum kullanıcıya izinleri kabul edip uygulamayı kurma ya da kabul etmeyip kurulumu iptal etme seçeneklerini sunmaktaydı. Bununla birlikte Google son zamanlarda kullanıcı güvenliği için birtakım çalışmalar yapmıştır. Bunların başında Android 6.0 Marshmallow sürümüyle artık uygulamaların kullandığı izinlerin kontrolünün tamamen kullanıcının inisiyatifine bırakılması gelmektedir. Böylece kullanıcının güvenli görmediği izinleri, uygulamanın kullanımına kapatmasına rağmen uygulamayı cihazına kurabilmesi sağlanmıştır.

Android mobil kötücül yazılım türünün çok çeşitli olmasından dolayı kullanıcılara uygulamaların güvenilirliği hakkında bilgi vermek ve uygulamaları kötücül yazılımlardan korumak için kötücül yazılım tespitine ihtiyaç duyulmuştur. Literatürde bu amaçla çeşitli analiz teknikleri bulunmaktadır. Bu makalede literatürde yapılan çalışmalar hakkında bilgi verilecektir. Bu makalenin 2. bölümünde Android yazılım mimarisi hakkında temel

bilgilere, 3. bölümde Android Kötücül Yazılım Tespit ve Koruma Yöntemlerine, 4. bölümde yapılan çalışmaların yeteneklerinin karşılaştırılmasına, 5. ve son bölümde ise sonuçlar ve değerlendirmelere yer verilmiştir.

Android Yazılım Mimarisi

Android'in yazılım mimarisi 5 katmandan meydana gelir (Android System Architecture, 2011). Android sistem mimarisi Şekil 1'de gösterilmiştir. Bu katmanları açıklamak gerekirse:

Linux Çekirdek (Linux Kernel)
Donanıma ait tüm bilgiler ile Android uygulamaların çalışmasını sağlayan ses, klavye, WI-FI, güç denetimi, işlem ve hafıza denetimi gibi sürücülerin bulunduğu en alt katmandır.

Kütüphaneler (Libraries)
Çekirdeğin üstünde bulunan ve genellikle C++ ve C dilleri ile yazılmış SQLite, SSL, FreeType gibi kütüphanelerin yanı sıra sistem kütüphanelerini içeren katmandır.

Android Çalışma Zamanı (Android Runtime)
Android çalışma zamanı, Android mimarisinin üçüncü bölümüdür ve alttan üçüncü bölümde bulunmaktadır. Bu bölüm Dalvik Sanal Makinesi denilen anahtar bir bileşen sağlamaktadır. Bunun yanı sıra bu katmanda Linux'un çekirdek kütüphaneleri bulunmaktadır.

a) Çekirdek Kütüphaneleri (Core Libraries)
Bunlar Java SE ve Java ME'den farklı kütüphanelerdir. Java'ya gerekli olan çekirdek API'ler, veri yapılar, hizmetler, dosya erişimler, ağ erişimler ve grafik bileşenlerinin yer aldığı kütüphanelerdir.

b) Dalvik Sanal Makinesi (Dalvik Virtual Machine - DVM)
Android işletim sisteminin en önemli bileşenidir ve mobil cihazlar için optimize edilmiş Android sanal makinedir. Modern JVM olarak da bilinir. Yüksek performans ve kusursuz bellek yönetimi sağlamaktadır. Bir cihaz üzerinde birden fazla sanal makine oluşturulabilmektedir. Dalvik Sanal Makinesi cihazdaki düşük seviyeli işlemleri düzene sokmak ve performansı arttırmak için Linux işletim sisteminin çekirdeğinden faydalanmaktadır. Gömülü (embedded) sistemler için tasarlanan bu sistemde, Java derleyicisi ile .class dosyalarına dönüştürülen .java dosyaları Dex derleyicisi ile .dex dosyalarına dönüştürür. Dönüştürülen bu .dex dosyalar Dalvik Sanal Makinesi'nde çalıştırılmaktadır.

Uygulama Çatısı (Application Framework)
Uygulama çatısı katmanı uygulamalara java classları şeklinde yüksek seviyeli servisler sağlamaktadır. Bu servisler;

a) Aktivite Yöneticisi (Activity Manager)
Projenin sahip olduğu aktivitelerin bütün yönlerini ve yaşam döngüsünü kontrol etmektedir.

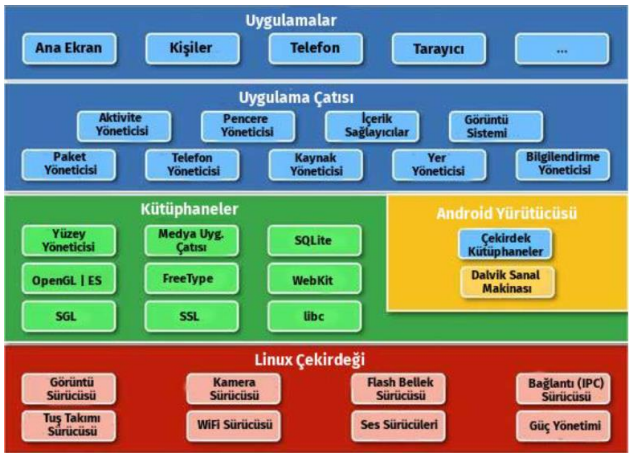
b) Görünümler (Views)
Aktivitelerin kullanması gereken arayüzlerin tasarlanması için kullanılmaktadır.

c) Uyarı Yöneticisi (Notification Manager)
Uygulamada kullanılan bildirimler ve uyarıların doğru ve zamanında çalışmasını kontrol etmektedir.

d) İçerik Sağlayıcılar (Content Providers)
Uygulamaların başka uygulamalar ile veri alışverişini sağlamaktadır.

e) Kaynak Yöneticisi (Resource Manager)
Kod kullanılmayan gömülü kaynaklara erişimi sağlamaktadır. Bu kaynaklar grafik ayarı ve kullanıcı arayüzü düzenleri olabilir.

Uygulamalar (Applications)
Android uygulama çatısının en üst kısmında yer alan katmandır. Sınıflar ve servisleri kullanan rehber, ayarlar, oyunlar vb. ana uygulamaların tümünün bulunduğu katmandır. Ayrıca bu katman yerel kütüphaneleri ve Linux çekirdeğini kullanmaktadır.



Şekil 1.Android sistem mimarisi (Android System Architecture, 2011)

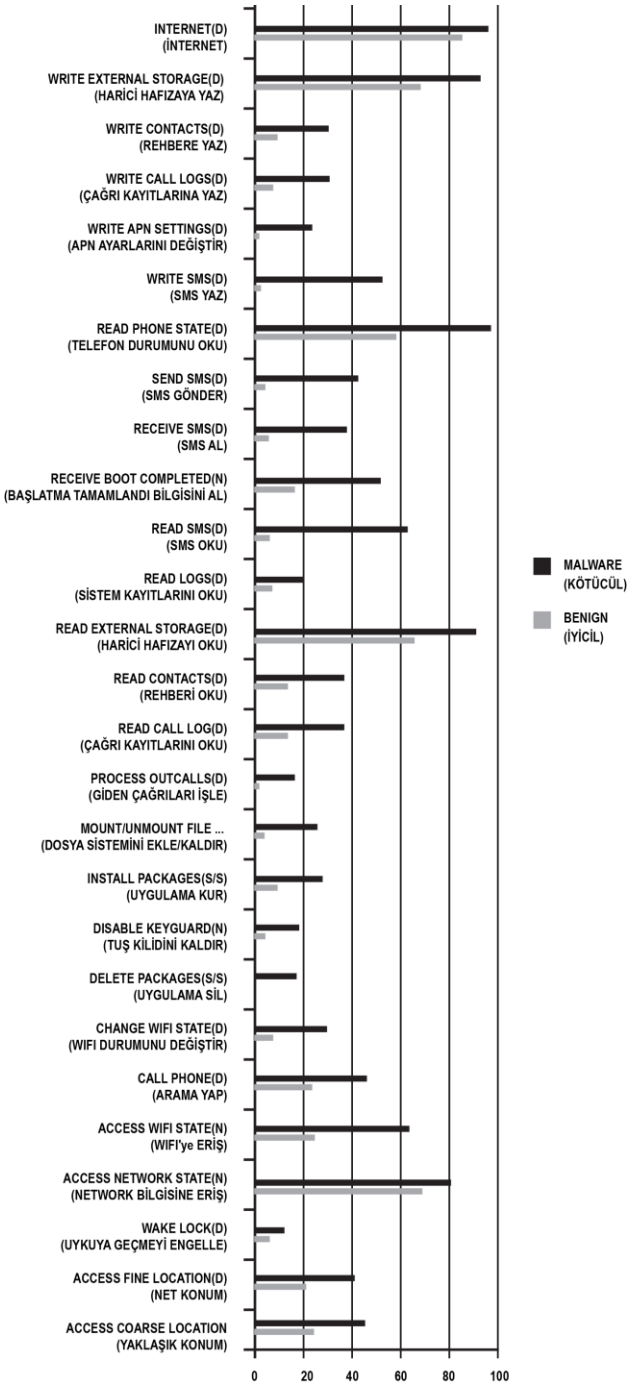
Android Kötü Niyetli Yazılımların Tespiti ve Koruma Yöntemleri

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

Statik Analiz Yöntemi

Statik analiz, yazılımın çalışmasına gerek kalmadan uygulamanın apk'sı üzerinden analiz yapılan test tekniklerinden biridir. Drebin, statik analiz yöntemi ve makine öğrenmesi algoritmalarını birlikte kullanarak kötücül yazılım tespit etmeyi amaçlayan bir araçtır (Arp vd., 2014). Uygulamanın kaynak kodunu ve AndroidManifest.xml dosyasını kullanarak uygulama tarafından kullanılan izinler, internet adresleri ve API çağrıları birleşik uzay vektörüne gömülerek kötücül yazılım tespiti için kullanılmaktadır. Android ApkTool, tersine mühendislik yöntemleri ile Android apk dosyalarını analiz ederek tekrar derlenmesini sağlamaktadır (Apktool, 2010). ApkAnalyser, statik analizi kullanan kötücül yazılım tespit araçlarından biridir (Sony Mobile Communications, 2012). Androguard, tersine mühendislik yöntemleri ve statik analiz kullanılarak kötücül yazılım tespit etmeyi amaçlayan bir araçtır (Dattani, 2015). Androguard, Linux/Windows/OSX platformlarını destekleyen python tabanlı araçtır. Dexter, statik analiz tekniğini kullanarak kötücül yazılım tespit etmeyi amaçlayan araçtır (Westervelt, 2013). Felt ve diğerleri (2011) tarafından yapılan çalışmada, çok fazla izin kullanan Android uygulamaları keşfetmeye yarayan Stowaway adında bir sistem geliştirilmiştir. Stowaway, mobil uygulamaların kullandığı Uygulama Programlama Arayüzleri (Application Programming Interface - API) çağrı kümesini bulup, çağrıları izinler ile eşleştirmeyi sağlamaktadır. Stowaway'i oluşturan iki kısım vardır: Bunlardan ilki uygulamaların kullandıkları Uygulama Programlama Arayüzleri'ni tespit etmektedir. İkincisi ise, izinleri eşleştirip Uygulama Programlama Arayüzleri çağrılarının kullandığı izinlerin belirlenmesini sağlamaktır. Wu ve diğerleri (2012) tarafından yapılan çalışmada, 238 kötü niyetli yazılım ile 1500 iyi niyetli yazılımdan faydalanılarak uygulamaların türlerinin tespiti amaçlanmıştır. Uygulamaların AndroidManifest.xml dosyalarındaki izinler, aktiviteler, servisler, alıcılar ve API çağrıları özellik olarak kullanılmıştır. Shaowu Liu ve diğerleri (2013) tarafından yapılan çalışmada, uygulamaları sınıflandırmak için veri kümesi olarak hem iyi niyetli hem de kötü niyetli uygulamalardan yararlanılmıştır. Shaowu Liu ve diğerleri kötücül yazılım için, ilk olarak Zhou ve Jiang tarafından 2012'de yayınlanan ve üçüncü parti uygulama dağıtım marketlerinden elde edilen karşılaştırmalı kötücül uygulama veri kümesini kullanmışlardır. İyi niyetli uygulamalar için ise üçüncü parti uygulama dağıtım marketleri olan SlideMe ve Pandaapp'den elde ettikleri kendi veri kümelerini kullanmışlardır. Bu veri kümesi oluşturulurken, uygulamaların indirilme sayıları ve kullanıcıların uygulamalara verdiği puanlar baz alınarak derecesi en yüksek olanlar veri kümesine dahil edilmiştir. Bu belirlemede 43 tane antivirüs programı kullanılarak hepsinden olumlu geçenler iyi niyetli uygulama veri kümesine dahil edilmiştir. Son dönemdeki çalışmaların aksine yalnızca

AndroidManifest.xml dosyasında belirtilen gerekli izinlere (required permissions) odaklanılmamış, uygulama indirildikten sonra yer alan ve akıllı telefonun işletim sistemi tarafından kullanılan izinler (used permissions) de ele alınmıştır. İyicil ve kötücül veri kümelerini başlangıçta analiz etmek için hiyerarşik Biclustering methodu kullanılmıştır. Kötücül uygulamaları iyicil olanlardan ayırmak için kullanılabilecek ilginç izin kümelerini tanımlamak için Contrast Permission Pattern Mining algoritması önerilmiştir. Önerilen Contrast Permission Pattern Mining algoritması kötücül yazılım tespitinde, hem uygulama için kullanılması gerekli olan izinleri hem de uygulama için gerekli olmadığı halde kullanılan izinlerin birlikte değerlendirilmesi gerektiğini kanıtlamaktadır. Quang Do ve diğerleri (2014) tarafından yapılan çalışmada, gizli veri sızıntılarını önlemek için muhalif bir model önerilip, gizlice sızdırılan verilerin önlenme teknikleri gösterilmiştir. Mobil veri sızıntısını engelleme tekniği (Mobile Data Exfiltration Technique - MDET) çeşitli ortamlarda sızıntıları engellemek ve Android cihazlarda ikili (binary) verileri ayıklayarak kod enjekte etme (code injection) metotlarını desteklemek amacıyla tasarlanmıştır. Bu işlem ikili kod seçme ve analiz (binary code selection ve analysis), SMALI kod enjekte etme ve düzenleme (code injection ve modification) ve ikili yeniden derleme (binary recompilation) olmak üzere üç ana aşamadan oluşmaktadır. İspat olarak, standart mobil cihazlar kullanılarak SMS ve işitilemez ses aktarımı gibi yöntemlerle veri sızıntısını önleme yöntemlerinin uygulanabilirliğinin gösterimi sağlanmıştır. Shina Sheen ve diğerleri (2014) tarafından yapılan çalışmada, kötücül yazılım analizi için Android tabanlı zararlı yazılım ve çok özellikli işbirlikçi karar füzyon (MCDF) yapısı kullanılarak ölçeklenebilir bir tespit mekanizması göz önüne alınmıştır. Statik analiz yöntemi kullanılarak AndroidManifest.xml dosyası analiz edilip uygulamaların kullandığı izinler ve API çağrıları elde edilmiştir. İzin ve API çağrıları tabanlı özellikler kullanılarak, bir sınırlayıcı topluluk oluşturulup bunların kararları birleştirilerek daha iyi bir metot belirlenmeye çalışılmıştır. Zararlı ve zararsız yazılımlardaki en sık kullanılan izinlerin bazıları oluşum yüzdesi olarak Şekil 2'de gösterilmektedir.



Şekil 2.Zararlı ve zararsız yazılımlarda en sık kullanılan izinlerin oluşum yüzdesi (Shina Sheen ve diğerleri (2014))

Kang ve diğerleri (2015) tarafından yapılan çalışmada, kötücül yazılım tespiti için uygulama geliştiricilerinin bilgilerini kullanan statik analiz

tabanlı bir çalışma sunulmuştur. Yapılan çalışmada kötücül uygulamaların seri numaraları ile geliştiricilerin uygulamalarının seri numaraları karşılaştırılarak uygulamaların güvenlik durumunun tespit edilmesi amaçlanmıştır. Anwar ve diğerleri (2016) tarafından yapılan çalışmada, mobil botnet algılamayı amaçlayan statik analiz tabanlı bir sistem önerilmektedir. Geliştirilen sistem izinler, alıcılar ve arka plan servisleri gibi özellikleri kullanarak uygulamaları sınıflandırmaktadır. DroidOL, kötü amaçlı yazılımları tespit etmeyi amaçlayan makine öğrenmesi tabanlı çevrimiçi hizmet sunan uygulamadır (Narayanan vd., 2016). Bu sistem üç aşamadan geçilerek geliştirilmiştir. İlk aşamada, statik analiz yöntemi kullanılarak işlemler arası kontrol akış grafikleri oluşturulmuştur. İkinci aşamada, Weisfeiler-Lehman çekirdeğinden yararlanılarak işlemler arası kontrol akış grafiklerinin alt grafikleri oluşturulmuştur. Son olarak ise ilk iki aşamada çıkartılan özellikler kullanılarak kötücül yazılım tespit edebilmek için çevrimiçi bir pasif agresif sınıflandırıcı eğitilmektedir. AndroDialysis çalışmasında (Feizollah vd., 2016), uygulamanın AndroidManifest.xml dosyasındaki niyetlerin analizi yapılarak kötücül yazılım tespit edilmesi amaçlanmaktadır. Statik analiz yöntemi ile yapılan çalışmada kullanılan niyetlerin izinlere göre daha etkili olduğu savunulmuş olmasına rağmen niyetlerin kötücül yazılım tespitinde izinlere kıyasla daha başarısız olacağı vurgulanmıştır. Sokolova ve diğerleri (2017) tarafından yapılan çalışmada, grafik tabanlı izin örüntüleri kullanılarak Android uygulamaların sınıflandırılması ve anormal davranışların tespit edilebilmesi amaçlanmaktadır. Yapılan çalışmada kategorize edilen uygulamaların davranışları analiz edilerek, beklenen davranışlar ile karşılaştırılmaktadır. Ek olarak kategorik uygulamalar ve kullanılan izinler grafik analiz metrikleri kullanılarak elde edilmiştir. Elde edilen modeller, kategorik olarak sınıflandırılan uygulamaların performansına göre değerlendirilmektedir. Wu ve diğerleri (2017) tarafından yapılan çalışmada, izin sızıntılarını tespit eden statik analiz tabanlı bir yaklaşım önerilmektedir. Yapılan çalışmada geliştiriciler tarafından Android uygulamalara eklenen ama kullanılmayan izinlerin tespiti yapılarak kötücül uygulamaların bu izinleri kullanmasını engellemek amaçlanmıştır. Yapılan çalışmada 550 uygulama kullanılmıştır. Deneysel sonuçlar, önerilen yöntemin başarı oranının %68 olduğunu göstermektedir. Park ve diğerleri (2018) tarafından yapılan çalışmada, Android kötücül yazılım tespiti için API (Application Programming Interface) ve izin tabanlı sınıflandırma yöntemi önerilmektedir. Önerilen yöntemde uygulamalar iyicil, kötücül ve şüpheli olmak üzere üç kategoriye ayrılmaktadır. API (Application Programming Interface) ve izin tabanlı sınıflandırma sistemi YARA adı verilen kurala göre oluşturulmaktadır. Her uygulamanın nitelikleri AndroidManifest.xml ve classes.dex dosyasından çıkartılarak YARA kuralı ile eşleştirilmektedir. Şahin ve diğerleri (2018) tarafından yapılan çalışmada, Android kötücül yazılım tespiti için diğer çalışmaların aksine izin tabanlı statik analiz yöntemini kullanarak izin ağırlıklı yaklaşım önerilmektedir. Önerilen yöntemde her bir izne birbirinden farklı puanlar verildikten sonra K-En Yakın Komşu (KNN) ve Naive Bayes (NB) algoritmaları uygulanarak önerilen yöntem önceki çalışmalar ile karşılaştırılmaktadır. Deneysel sonuçlar, önerilen yöntemin başarı oranının %96,64 olduğunu göstermektedir. Arslan ve diğerleri (2019) tarafından yapılan çalışmada, uygulamanın çalışması için ihtiyaç duymadığı halde kullanıcıdan talep edilen izinlerin şüpheli faaliyetler için kullanılabileceği savunulmaktadır. Bu şüphe uyandıran yedek izinleri tespit edebilmek için statik ve kod analizi yöntemi kullanılmaktadır. Deneysel sonuçlar, önerilen yöntemin başarı oranının %91,65 olduğunu göstermektedir. Zhu ve diğerleri (2018) tarafından yapılan çalışmada, izinler, hassas API (Application Programming Interface)'ler, monitör edilebilir sistem eventleri ve izin oranları statik analiz yöntemi ile kullanılarak bir Android uygulamanın kötücül olup olmadığının tespit edilebilmesi amaçlanmaktadır. Yapılan çalışmada önerilen yöntemin performansını doğrulamak için 2130 uygulamadan oluşan veri seti kullanılmaktadır.

Deneysel sonuçlar, önerilen yöntemin doğruluğunun %88,26 olduğunu göstermektedir. Doğru ve Kiraz (2018) tarafından yapılan çalışmada, statik analiz yöntemi kullanılarak web tabanlı Android kötücül yazılım tespit sistemi geliştirilmiştir. Geliştirilen sistem AndroidManifest.xml dosyasında bulunan izinleri kullanarak uygulamanın iyicil ya da kötücül olup olmadığı hakkında bilgi vermektedir. Yapılan çalışmada birbirinden farklı dört hesaplama yöntemi kullanılmıştır. Yapılan hesaplamalardan VirusTotal ile birlikte iyicil-kötücül veri setlerinde kullanılan izinlerle yapılan hesaplama yöntemi %97,73 oranıyla en yüksek başarıyı göstermiştir.

Dinamik Analiz Yöntemi

Dinamik analiz tekniği, çalışan bir uygulamanın sergilediği davranışları ve API çağrılarını izleyerek kötücül yazılım tespit edebilmeyi sağlayan yöntemdir. Kötücül yazılımı tespit etmek için dinamik yöntem kullanırken bir zaman kısıtı konulmamalıdır. Örneğin, 1-5 gün arası verildiğinde, saldırgan güncellemeyi 6. gün yaptığı zaman dinamik yöntem kötücül yazılımı tespit edemez. Ayrıca, sofistike bir saldırgan statik ve dinamik analiz çalışmalarını zorlaştırmak için kandırma ve şifreleme kullanan server-side polimorfizmi kullanabilir. Dinamik analiz yaklaşımı diğer analiz yaklaşımlarına göre daha geniş bir bakış açısına sahip olmasına rağmen daha maliyetli olduğu için daha az tercih edilen bir yöntemdir. Burguera ve diğerleri (2011) tarafından yapılan çalışmada, uygulama üzerinde normal olmayan hareketleri tespit eden Crowdroid adında bir sistem geliştirilmiştir. Crowdroid, Strace komutu ile sistem çağrılarını derleyerek uygulamaları iyi niyetli veya kötü niyetli olarak sınıflandırmaktadır. Zhou ve Jiang (2012) tarafından yapılan çalışmada, veri kümesi olarak 49 farklı türden oluşan 1260 adet kötü niyetli yazılım kullanılmıştır. Dinamik analiz tekniği kullanılarak kötücül yazılımların yüklenme aşaması, aktiviteler ve aktiviteler arasındaki veri alışverişi gibi çeşitli davranış biçimleri incelenmiştir. Hooker, otomatik olarak Android uygulamaların dinamik analizini yapar ve açık kaynak kodludur (Bossert ve Kirchner, 2014). Decafplatform (Dynamic Executable Code Analysis Framework Platform) (Sycurelab, 2016), dinamik analiz metodunu kullanarak anormal davranışları gözlemleyen DECAF Binary Analiz Platformu'dur. DroidScope ise Decafplatform'unun bir eklentisidir. TaintDroid, Enck ve diğerleri (2010) tarafından geliştirilen açık kaynak kodlu dinamik analiz aracıdır. Bu araç, uygulama içerisindeki veri sızmalarına odaklanarak uygulamaların hassas bilgi gereksinimlerini nasıl kullandığını takip etmektedir. Droidbox, 2011 yılında Patrik Lantz tarafından geliştirilen, dinamik analizi kullanarak uygulamanın davranışlarını analiz eden araçtır (Lantz, P., 2011). Shaptai ve diğerleri (2014) tarafından yapılan çalışmada mobil uygulamaların network davranışları incelenmiştir. Yarı denetimli makine öğrenme yöntemleri, uygulamanın normal davranış kalıplarını öğrenmek ve uygulamanın beklenen davranış sapmalarını tespit etmek için kullanılmıştır. Bu çalışmada kendini güncelleyebilen kötücül yazılımların tespit edilebilmesi amaçlanmıştır. Yapılan çalışmada aynı türden uygulamaların ağ davranışlarının benzerlik göstereceği vurgulanmıştır. Yapılan testler bu düşünceyi doğrulamaktadır. Jang ve diğerleri (2016) tarafından yapılan Andro-Profiler çalışmasında, sistem çağrıları dahil olmak üzere entegre sistem günlüklerinden çıkarılan davranış profiller kullanılarak kötücül yazılımlar sınıflandırılmaktadır. Andro-Profiler, entegre sistem günlüklerini oluşturmak için sanal cihazda kötü niyetli bir uygulamayı çalıştırır ve entegre sistem günlüklerini analiz ederek insan tarafından okunabilen davranış profillerini oluşturmaktadır. Garg ve diğerleri (2016) tarafından yapılan çalışmada, kötücül uygulamaları ağ üzerinde gözlemci bir göz ile ağ etkinliğine bakarak tespit etmeye çalışan bir sistem geliştirilmiştir. Ayrıca geliştirilen model ağ izlerini kullanarak kötücül uygulamaları tespit edebilme, işletim sistemlerinin farklı sürümleriyle çalışabilme, bilinmeyen uygulamaları tespit edebilme ve şifrelenmiş verilerle virüs bulaşmış uygulamaları tespit edebilme yeteneklerine sahiptir. Chang ve diğerleri (2016) tarafından yapılan çalışmada, makine öğrenmesi kullanılarak davranış bazlı

kötücül yazılım algılama sistemi önerilmektedir. Bu çalışmada DroidBox yapısına ek olarak otomatik tetiklenen görünüm tanımlama programı geliştirilmiştir. Bu program sayesinde mobil uygulamalara anlamlı sıraya göre ulaşılmaktadır. Geliştirilen sistem; ön işleme, veri izleme, karar modeli ve veritabanı olmak üzere dört bileşenden oluşmaktadır.

Hibrid Analiz Yöntemi

Birden fazla mobil kötücül yazılım tespit yöntemini kullanarak kötücül yazılım tespit etmeyi amaçlayan tekniklerden biridir. Wang ve diğerleri (2015) tarafından yapılan çalışmada, anormallik tespiti ile kötüye kullanım tespitini entegre eden yeni bir hibrid mobil kötücül yazılım tespit sistemi geliştirilmiştir. İlk olarak AndroidManifest.xml dosyasındaki statik özelliklerden yararlanılarak statik analiz yöntemi kullanılmıştır. İkinci adımda ise dinamik analiz yönteminden faydalanılmıştır. Çalışma esnasında uygulamanın dinamik özelliklerinin çıkarılması için emülatör ortamında uygulamaları çalıştıracak CuckooDroid kullanılmıştır. Eş zamanlı olarak dinamik API çağrıları belirlenmiştir. Dinamik ve statik analizle elde edilen veriler makine öğrenmesi yöntemlerini besleyerek kötücül yazılım tespiti sağlanmıştır. 5560 kötücül yazılım örneği ve 12000 iyicil yazılım örneği ile çalışma yapılmıştır. Yapılan deneyler sonucunda %98,32 oranında başarı elde edilmiştir. Kabakuş ve Doğru (2018) tarafından yapılan çalışmada, hibrid analiz tekniği kullanılarak Android kötücül yazılımların derin analizi gerçekleştirilmiştir. Yapılan çalışmada kötü amaçlı yazılımları tespit etmek için statik ve dinamik analiz tekniklerini birleştiren mad4a olarak isimlendirilen uygulama çatısı ile uygulamaların karakteristiklerinin ortaya konulması amaçlanmıştır. Yapılan çalışmanın statik analiz kısmında uygulama izinleri, dinamik analiz kısmında ise emulatörde çalıştırılan uygulamaların davranışları incelenmiştir.

Literatürdeki Çalışmaların Yeteneklerinin Karşılaştırılması

Bu çalışmada incelenen literatürdeki bazı çalışmaların yetenek karşılaştırmaları Tablo 1'de gösterilmiştir.

Tablo 1. Literatürdeki çalışmaların yetenek karşılaştırmaları

Çalışmalar	Statik Analiz	Dinamik Analiz	Hibrid Analiz	Çalışma Yılı	Başarı Oranları	Özellikler
Wu ve diğerleri	VAR	YOK	YOK	2012	%97,87	İzinler, aktiviteler, servisler, alıcılar ve API çağrıları
Liu ve diğerleri	VAR	YOK	YOK	2013		İzinler
Arp ve diğerleri	VAR	YOK	YOK	2014	%94	İzinler ve kaynak kodlar
Sheen ve diğerleri	VAR	YOK	YOK	2014	%98	İzinler ve API çağrıları
Kang ve diğerleri	VAR	YOK	YOK	2015	%98	Uygulama seri numaraları
Sokolova ve diğerleri	VAR	YOK	YOK	2017	-	İzinler ve uygulama davranışları
Şahin ve diğerleri	VAR	YOK	YOK	2018	%96,64	İzinler

Park ve diğerleri	VAR	YOK	YOK	2018	-	İzinler ve API çağrıları
Zhu ve diğerleri	VAR	YOK	YOK	2018	%88,26	İzinler, API çağrıları, sistem eventleri
Doğru ve Kiraz	VAR	YOK	YOK	2018	%97,73	İzinler
Arslan ve diğerleri	VAR	YOK	YOK	2019	%91,65	İzinler
Burguera ve diğerleri	YOK	VAR	YOK	2011	-	Sistem çağrıları
Wu ve diğerleri	VAR	YOK	YOK	2012	%97,87	İzinler ve API çağrıları
Zhou ve Jiang	YOK	VAR	YOK	2012	%79,6	Uygulama davranışları, aktiviteler, aktiviteler arası taşınan veriler
Hooker	YOK	VAR	YOK	2014	-	Uygulama davranışları
Jang ve diğerleri	YOK	VAR	YOK	2016	%98	Sistem çağrıları ve uygulama davranışları
Wang ve diğerleri	VAR	VAR	VAR	2015	%98,32	İzinler, niyetler, API çağrıları, donanım özellikleri ve kodla ilgili bilgiler
Kabakuş ve Doğru	VAR	VAR	VAR	2018	-	İzinler ve uygulama davranışları

Sonuçlar ve Değerlendirmeler

Günümüzde bilgisayarların yerini, kullanım kolaylığından dolayı akıllı mobil cihazların aldığı görülmektedir. Akıllı cihazların günümüzde oldukça popüler hale gelmesiyle birlikte kullanım alanı da oldukça çeşitlenmiştir. Bu çeşitlilik mobil kötücül yazılım geliştiricilerinin dikkatini çekmiştir ve bu gelişme onları çeşitli amaçlarla mobil kötücül yazılım geliştirmeye yöneltmiştir. Yapılan araştırmalara göre dünya genelinde %82,8 oranla en çok kullanılan mobil işletim sisteminin Android olduğu gözlenmiştir. Android işletim sisteminin bu denli popüler olması, Linux tabanlı açık kaynak kodlu mobil işletim sistemi olması, resmi uygulama dağıtım kanalı olan Google Play Store tarafında uygulamaların güvenilirliğini test eden etkin bir alt yapının olmamasının Android'in kötücül yazılımların öncelikli hedefi haline gelmesinde önemli etkileri olmuştur. Bu sebeplerden dolayı her gün yeni bir tür mobil kötücül yazılım tespit edilmektedir. Tüm bu gelişmeler sonucunda Google tarafından 2015 yılında Marshmallow sürümü ile yeni izin kontrol mekanizması duyurulmuştur. Bu gelişme ile birlikte uygulamaların kullanmak istediği izinler kurulum esnasında tamamen kullanıcının tercihine bırakılmaktadır. Fakat bu yöntemin de uygulamaların güvenirliğini tespit etmek için ne kadar yararlı bir yöntem olduğu merak edilmektedir. Araştırmacılar tarafından gün geçtikçe yeni çalışmalar yapılmakla birlikte daha güvenli ve doğru kötücül yazılım tespit mekanizmaları üretmeyi amaçlayan çalışmalar literatüre girmektedir. Bu çalışmada bahsedilen çalışmaların hiçbiri kötücül yazılım tespiti için %100 çözüm oluşturamasa bile her biri çeşitli yönlerle kötücül yazılımı tespit etmeyi

başarmaktadır. Bahsedilen çalışmalar arasında kötücül yazılım tespiti için en başarılı sonucun %98,32 ile Wang ve diğerleri (2015) tarafından hibrid analiz yöntemi kullanılarak geliştirilen sisteme ait olduğu gözlenmiştir. Android kötücül yazılım tespitinde yapılan hataların minimize edilerek başarı oranının arttırılması için bundan sonraki çalışmalarda çok yönlü bir hibrid model üzerinde çalışılması gerektiği düşünülmektedir. Ayrıca semantik analiz, statik-dinamik analiz, imza tabanlı analiz ve makine öğrenmesi algoritmaları birleştirilip, yeni geliştirmeler üzerinde çalışılarak kötücül yazılım tespiti sağlayan etkin bir araç üretilebileceği düşünülmektedir.

Kaynaklar

Anwar, S., Zain, J. M., Inayat, Z., Karim, A., Haq, R. U., ve Jabir, A. N., (2016). A static approach towards mobile botnet detection, 3rd International Conference on Electronic Design (ICED), Thailand.

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

Arslan R. S., Doğru İ. A ve Barışcı N., (2019). Permission based malware detection system for Android using machine learning techniques, *International Journal of Software Engineering and Knowledge Engineering*, **29**, 1, 63-91.

Burguera, I., Zurutuza, U. ve Nadjm-Tehrani, S., (2011). Crowdroid: behavior-based malware detection system for Android, *1st ACM workshop*, Security and privacy in smartphones and mobile devices, 15-25.

Chang, W.-L., Sun, H.-M. ve Wu, W. (2016). An Android behavior-based malware detection method using machine learning, *IEEE International Conference on Signal Processing*, Communications and Computing (ICSPCC), Hong Kong.

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

Doğru, İ. A. ve KİRAZ, Ö. (2018). Web-based Android malicious software detection and classification system, *Applied Sciences*, **8**, 9, 1622.

Enck, W., Gilbert, P., Chun, B.-G., Cox, L. P., Jung, J., McDaniel, P., Sheth, A. N., (2010). TaintDroid: an information-flow tracking system for realtime privacy monitoring on smartphones, *Proceedings*, 9th USENIX conference, Operating Systems Design and Implementation, 393-407, Vancouver.

Feizollah, A., Anuar, N. B., Salleh, R., Suarez-Tangil, G., ve Furnell, S., (2016). AndroDialysis: Analysis of Android intent effectiveness in malware detection, *Computers & Security*, **65**, 121-134.

Felt, A.P., Finifter, M., Chin, E., Hanna, S. ve Wagner, D., (2011). A survey of mobile malware in the wild, *Proceedings*, SPSM '11 1st ACM workshop, Security and Privacy in Smartphones and Mobile Devices, 3-14, Chicago.

Garg, S., Peddoju, S. K. ve Sarje, A. K. (2016). Network-based detection of Android malicious apps, *International Journal of Information Security*, **16**, 4, 1-16.

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

Kabakuş, A. T. ve Doğru, İ. A., (2018). An in-depth analysis of Android malware using hybrid techniques, *Digital Investigation*, **24**, 25-33.

Kang, H., Jang, J.-W., Mohaisen, A. ve Kim, H. K., (2015). Detecting and classifying Android malware using static analysis along with creator information, *International Journal of Distributed Sensor Networks*, **11**, 6.

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

Narayanan, A., Yang, L., Chen, L., ve Jinliang, L., (2016). Adaptive and scalable Android malware detection through online learning, 2016

International Joint Conference on Neural Networks (IJCNN), Vancouver.

Park, J., Chun, H. ve Jung, S., (2018). API and permission-based classification system for Android malware analysis, International Conference on Information Networking (ICOIN), Thailand.

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

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

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

Sokolova, K., Perez, C., ve Lemercier, M., (2017). Android application classification and anomaly detection with graph-based permission patterns, *Decision Support Systems*, **93**, 62-76.

Şahin, D. Ö., Kural, O. E., Akleylek, S. ve Kiliç, E.,(2018). New results on permission based static analysis for Android malware, 6th International Symposium on Digital Forensic and Security (ISDFS), Antalya.

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

Wang, X., Yang, Y., Zeng, Y. ve Tang, C., (2015). Chuan Tang : A novel hybrid mobile malware detection system integrating anomaly detection with misuse detection, *Proceedings*, 6th International Workshop on Mobile Cloud Computing and Services, 15-22, Paris.

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

Wu, D. J., Mao, C. H., Wei, T. E., Lee, H. M., and Wu, K. P. (2012). Droidmat: Android malware detection through manifest and api calls tracing, *2012 Seventh Asia Joint Conference*, Information Security, Tokyo.

Zhou, Y. ve Jiang, X., (2012). Dissecting Android Malware: Characterization and evolution, *2012 IEEE Symposium*, Security and Privacy, 95–109, San Francisco.

Zhu, H.-J., You, Z.-H., Zhu, Z.-X., Shi, W.-L., Chen, X. ve Cheng, L., (2018). Effective and robust detection of android malware using static analysis along with rotation forest model, *Neurocomputing*, **272**, 638-646.

Statista. Global market share held by the leading smartphone operating systems in sales to end users from 1st quarter 2009 to 2nd quarter 2018. https://www.statista.com/statistics/266136/global-market-share-held-by-smartphone-operating-systems/ . Yayın tarihi 2018. Erişim tarihi Nisan 6, 2019.

IDC. Worldwide Mobile Phone Forecast Update, 2018–2022. https://www.idc.com/getdoc.jsp?containerId=US44269718 . Yayın tarihi Eylül 2018. Erişim tarihi Nisan 6, 2019.

Statista. Number of available applications in the Google Play Store from December 2009 to December 2018. https://www.statista.com/statistics/266210/number-of-available-applications-in-the-google-play-store/ . Yayın tarihi 2018. Erişim tarihi Nisan 6, 2019.

F-Secure. Another Reason 99% of Mobile Malware Targets Androids. https://blog.f-secure.com/another-reason-99-percent-of-mobile-malware-targets-androids/ . Yayın tarihi Şubat 2017. Erişim tarihi Nisan 6, 2019.

G Data. New malware every 10 seconds!. https://www.gdatasoftware.com/blog/2018/05/30

735-new-malware-every-10-seconds . Yayın tarihi Eylül 5, 2018. Erişim tarihi Nisan 6, 2019.

Android System Architecture. http://elinux.org/Android_Architecture . Yayın tarihi Haziran 13, 2011. Erişim tarihi Nisan 6, 2019.

Apktool. https://ibotpeaches.github.io/Apktool/ . Yayın tarihi Mart 1, 2010. Güncellenme tarihi Mart 3, 2019. Erişim tarihi Nisan 6, 2019.

Sony Mobile Communications. Apkanalyser. https://github.com/sonyxperiadev/ApkAnalyser . Yayın tarihi Nisan 13, 2012. Güncellenme tarihi Mart 15, 2013. Erişim tarihi Nisan 6, 2019.

Dattani. Androguard. http://www.technotalkative.com/part-1-reverse-engineering-using-androguard/ . Yayın tarihi Mart 7, 2015. Erişim tarihi Nisan 6, 2019.

Robert Westervelt. Dexter. https://www.crn.com/news/security/240150803/android-dexter-security-tool-could-bolster-mobile-defenses.htm . Yayın tarihi Mart 14, 2013. Erişim tarihi Nisan 6, 2019.

Bossert G. ve Kirchner, D. Hooker. https://github.com/AndroidHooker/hooker , Yayın tarihi Nisan 14, 2014. Güncellenme tarihi Ağustos 9, 2016. Erişim tarihi Nisan 6, 2019.

Sycurelab, Decaf-platform. https://github.com/sycurelab/DECAF , Yayın tarihi Temmuz 1, 2015. Güncellenme tarihi Şubat 21, 2016. Erişim tarihi Nisan 6, 2019.

Lantz, P., Droidbox. https://github.com/pjlantz/droidbox , Yayın tarihi Haziran 4, 2011. Güncellenme tarihi Ekim 19, 2017. Erişim tarihi Nisan 6, 2019.

Malware Detection Systems in Android Operating System

Extended abstract

Today mobile devices have become an important tool to access information when computers are inaccessible. Smart mobile devices are being used in many different areas for various purposes. Since there has been a significant growth in applications that require constant internet connection, such as social networks, this has adversely affected internet traffic and has caused potential network congestion. This popularity has turned smart mobile devices into the target of malware. According to a study conducted by Statista in the second quarter of 2018, 88% of smartphones sold worldwide between 2009 and 2018 are devices with the Android operating system (Statista, 2018).

Android is a Linux-based open source operating system developed by Google. Android has a permission-based security mechanism. The fact that Android operating system has a permission-based security mechanism and does not have an adequate security scan by Google makes it the target of malware developers. According to the report presented in 2017 by F-Secure security corporation, more than 99 percent of mobile malware attack targets Android devices (F-Secure, 2017). In another report presented in 2018 by Statista statistics corporation, it's clear that more than 2.6 millions of applications exist in Google Play Store (Statista, 2018). Another report, presented in September 2018 by G Data Corporation, states that 846.916 new malware applications are detected (G Data, 2018). Many studies have been conducted in the literature to protect users from malware by detecting them. In this study, researches on the detection of Android malware in the literature are examined and presented. We claim that the hybrid analysis method is more efficient than any other malware detection methods available in the literature. Despite a number of precautions taken by Google, a new malware application is still encountered every ten seconds and it's expected that there will be about 3.4 millions of new Android malware application by the end of 2018 (G Data, 2018).

Relying on the studies made, new malware detection methods, which will come up with more accurate results, are required for a better mobile malware detection. For an efficient malware detection, we think that, at forthcoming studies, a versatile hybrid model should be investigated based on machine learning algorithms.

Keywords: *Android Operating System, Android permission, malware detection, application security, mobile devices, smart device.*