Düzce University
Journal of Science & Technology

Research Article

Implementing a hybrid Android sandbox for malware analysis

Mert Can COŞKUNER [a,*], Dr. Murat İSKEFİYELİ [b]

[a] *Sakarya Üniversitesi, Bilgisayar Mühendisliği Bölümü, Sakarya, Türkiye*
[b] *Sakarya Üniversitesi, Bilgisayar Mühendisliği Bölümü, Sakarya, Türkiye*
** Sorumlu yazarın e-posta adresi: mert.coskuner@ogr.sakarya.edu.tr*
DOI: 10.29130/dubited.1239779

ABSTRACT

Increasing Android malware is becoming a problem for analysts in order to analyse and decide whether an application is malicious or benign. Research for this problem and prototypes is insufficient in terms of accessibility and analysis capabilities. In this paper, a hybrid Android sandbox for both dynamic analysis and static analysis is proposed while comparing malware sandboxes already used for malware analysis. As a result, a hybrid Android malware sandbox which has static and dynamic analysis capabilities is implemented.

Keywords: Android, Malware, Sandbox

Android zararlı uygulama analizi için hibrit kum havuzu geliştirilmesi

ÖZET

Artan zararlı Android uygulamalarının gerçekten zararlı olup olmadığına karar vermek için zararlı yazılım analistlerinin tipik olarak başvurduğu kum havuzları Android işletim sistemi için yetersiz kalmaktadır. Bu bağlamda yapılan akademik çalışmalar ve ortaya çıkan prototipler erişilebilirlik ve analiz yapabilme kapasitesi olarak yetersiz kalmaktadır. Bu makalede Android zararlı yazılım analizi için hibrit analiz yapabilecek bir kum havuzu önerilmiş ve zararlı yazılımların tespiti için kullanılan kum havuzlarının Android zararlı yazılımlar yönünden incelemesi yapılmıştır. Çalışma sonucunda Android zararlı yazılım analizi için kullanılabilecek statik ve dinamik analiz yeteneği olan bir Android kum havuzu geliştirilmiştir.

Anahtar Kelimeler: Android, Zararlı Yazılım, Kum Havuzu

Received: 21/01/2023, Revised: 06/04/2023, Accepted: 17/07/2023

I. GİRİŞ

Mobil telefon endüstrisi son yılların en hızlı gelişen endüstrilerinden biri olmuştur. Bu gelişmeler ışığında Android işletim sisteminin akıllı telefonlar içerisinde büyük bir pay elde etmesinin bir yan etkisi olarak Android işletim sistemi zararlı yazılım geliştiricilerinin de ilgisini çekmeye başlamıştır. Artan zararlı Android uygulamalarının gerçekten zararlı olup olmadığına karar vermek için zararlı yazılım analistlerinin tipik olarak başvurduğu kum havuzları, Android işletim sistemi için yetersiz kalmaktadır. Bu alanda L. Batyuk ve ark., A. Reina ve ark., M. Spreitzenbarth ve ark., L. K. Yan ve ark. yaptıkları çalışmalarda ve ortaya çıkardıkları prototiplerde erişilebilirlik ve analiz yapabilme kapasitesi olarak yetersiz kalmışlardır [1, 2, 3].

Zararlı yazılımlar kalıcılık, veri kaçırma, komuta kontrol sunucusu ile iletişim ve verilen komutları yerine getirme gibi yeteneklere sahip olabilir. Android zararlıları da sıradan zararlılar gibi bu özelliklere sahip olabilmektedir. Android zararlı yazılım analizi statik ve dinamik analiz olmak üzere iki yöntem ile yapılmaktadır. Bir uygulamanın zararlı olup olmadığını anlamak için en kolay yöntemlerden birisi; şüpheli uygulamayı kontrollü bir telefona yüklemek, telefonun sistemindeki ve ağ trafiğindeki değişiklikleri izlemektir. Bu analiz yöntemine dinamik analiz adı verilmektedir [4]. Dinamik analiz işlemi, dinamik analiz kum havuzu kullanılarak yapıldığında bir zararlı yazılım analistinin harcayacağı efora kıyasla daha kısa sürede şüpheli uygulama hakkında bilgi verebilir. Dinamik analiz işleminde kullanılan kontrollü ortam; sanal ortam (bilgisayar) ya da gerçek bir cihaz olabilir. Dinamik analiz yöntemi, kısa sürede analiz edilen uygulama hakkında bilgi verdiğinden dolayı sıkça kullanılmaktadır. Statik analiz yöntemi ise; şüpheli uygulamanın çalıştırılmadan incelenmesi işlemine denmektedir. Hem statik, hem dinamik analiz yeteneklerine sahip olan kum havuzlarına hibrit kum havuzu denmektedir.

Akıllı telefon sektörünün %80'ine yakınını elinde bulunduran Android işletim sistemi akıllı telefonlar için en popüler işletim sistemi olmakla beraber en büyük rakibi Apple Inc'in iOS işletim sistemidir. Fakat, iOS işletim sisteminin aksine, Android işletim sisteminin üçüncü parti mağazalardan uygulama indirip kurulmasına izin vermesi siber suçluların Android işletim sisteminde zararlı uygulama geliştirmesine ve dağıtmasına bir motivasyon sağlamaktadır. Anti-virüs şirketlerinin raporları Android işletim sistemi için geliştirilen zararlı uygulamaların artışını net bir şekilde ortaya koymaktadır: Sophos'un raporuna göre 650,000 eşsiz Android zararlısı tespit edilmiştir ve her gün 2,000 Android zararlısı bu sayıya eklenmeye devam etmektedir [5, 6].

Google LLC şirketi siber suçluların bu artan ilgisini görerek Şubat 2012 tarihinde Bouncer'ı tanıtmıştır [7]. Bouncer servisi Google Uygulama Mağazasına yüklenen uygulamaların zararlı olup olmadığına karar vermektedir. Bouncer servisinin devreye girmesiyle Google Uygulama Mağazasında yer alan zararlıların sayısı %40'a yakın bir düşüş göstermiştir. Yine de, Bouncer'ın tanıtılması Google Uygulama Mağazasında yer alan zararlı uygulamaların varlığını tamamiyle bitirmemiştir. Bunun en büyük örneği 2017 yılında Check Point tarafından Google Uygulama Mağazasında tespit edilen, bir milyonun üzerinde indirmesi bulunan ve bir sıfırıncı gün (zero-day) açığı kullanan zararlı fidye uygulamasıdır [8]. Ayrıca, Android ekosistemi içerisinde uygulama indirilebilen tek mağaza Google Uygulama Mağazası değildir. Uygulamalar torrent, hosting servisleri veya kendi depolarını oluşturan alternatif uygulama mağazalarından indirilebilmektedir. Alternatif yollarla yüklenen zararlı uygulamaların uzun süre tespit edilmeden faaliyetlerini devam ettirebildiği pek çok örnek mevcuttur [9, 10].

Google LLC'nin Bouncer'ı tanıtmasından sonra Bläsing ve ark. AASandbox adını verdiği Android uygulamaları için dinamik analiz platformunu tanıtmıştır [1]. Bu platformun tanıtılmasından sonra yeni dinamik analiz yapan kum havuzları akademide ve sektörde tanıtılmaya başlanmıştır. Windows çalıştırılabilir dosyaları için geliştirilen dinamik kum havuzları gibi Android kum havuzları da girdi

olarak verilen uygulamayı kontrollü bir ortamda çalıştırıp davranışını incelemektedir. İnceleme sonuçları zararlı yazılım analistine rapor olarak sunulmakta olup, uygulamanın zararlı olup olmadığına dair analistin karar vermesine yardımcı olmak adına amacıyla bir bilgilendirme niteliği taşımaktadır. Bu sistemlerin pek çoğu dinamik analiz sonuçlarına takviye olması anlamında statik analiz de yaparak hibrit bir yaklaşım izlemektedir. Fakat, Android zararlı uygulama analizi için kullanılan bu teknikler üzerine yapılan I. Burguera ve ark., M. C. Grace ve ark., Y. Zhou ve ark. tarafından yapılan araştırmalar kapsamlı bir teknik çözüm vermemektedir [11, 12, 13].

Android zararlı uygulamalarının her geçen gün artmasıyla analiz süreçlerinin maliyeti ve zamanı artmış ve bu artış etkili bir şekilde hibrit analiz yapabilen bir kum havuzu ihtiyacı doğurmuştur fakat Android platformunda dinamik analiz yapabilen kum havuzu sayısı kısıtlıdır. Bu çalışmada, Android platformunda kısıtlı olan hibrit kum havuzu ihtiyacını karşılayabilmek adına açık kaynak bir kum havuzu olan Cuckoo Sandbox kum havuzuna Android platformunda statik ve dinamik analiz gerçekleştirebilme özelliği kazandırılmıştır. Bu çalışma aşağıdaki katkıları içermektedir:

Android uygulamalarının hibrit analizini yapabilen bir kum havuzu geliştirilmesi.
Statik analiz ile bilinen güvenlik ihlali göstergelerinin tespit edilmesi.
Dinamik analiz ile zararlı uygulamardaki davranışların tespit edilmesi.

Makalenin ikinci bölümünde Android zararlı uygulamalarının tespiti için kullanılan tekniklerin anlaşılabilmesi için Android zararlı uygulama tehditleri açıklanmıştır, üçüncü bölümünde zararlı yazılım analizinde aktif olarak kullanılan kum havuzlarının Android zararlı yazılım analiz yetenekleri incelenmiştir ve hibrit kum havuzunun sistem tasarımı açıklanmıştır, sonuç bölümünde ise geliştirilen hibrit kum havuzu ile alınan sonuçlar değerlendirilmiştir.

II. ANDROİD ZARARLI YAZILIM ANALİZİ

Bu başlıkta Android zararlı yazılımlarının tespiti için kullanılan tekniklerin anlaşılabilmesi için Android zararlı yazılım tehditlerinden bahsedilmiştir. Tehditlerin anlatımında Android zararlı yazılım geliştiricilerinin motivasyonu, zararlı uygulamaları dağıtım şekilleri ve zararlı uygulama veri setlerinin bulunabileceği ortamlardan bahsedilecektir.

Ağustos 2010 yılında AndroidOS.FakePlayer ismiyle ilk Android zararlı yazılımı keşfedilmiştir [10]. AndroidOS.FakePlayer bulaştığı hedeflerde ücretli servislere SMS mesajları atarak gelir elde etme amacı gütmektedir. AndroidOS.FakePlayer'ın çıkmasının üzerinden geçen bu zamanda Android zararlı yazılımlar giderek karmaşıklaşmaya başlamıştır. Android zararlı yazılım geliştiricilerinin geliştirdikleri zararlı yazılımların karmaşıklığını giderek arttırmalarının arkasındaki motivasyon finansal kazançtır. Zararlı yazılım geliştiricileri, zararlı yazılımlarının bulaştıkları cihazlardan bankacılık bilgilerini çalabilir, AndroidOS.FakePlayer örneğinde olduğu gibi ücretli servislerle iletişim kurabilir, kişisel veriler çalarak uzak sunucuya yükleyebilir ya da bulaştığı cihazı komuta kontrol sunucusuna bağlayabilir ve bir botnetin parçası haline getirebilir. Mobil bankacılık zararlı yazılımları arka planda bir servis başlatarak SMS mesajlarını çalma yoluyla veya ekran enjeksiyonu gibi yöntemler ile sosyal mühendislik aracılığıyla iki faktörlü doğrulama mekanizmalarını atlatacak kapasitelere sahip olabilir.

Kurbanlarına zararlı yazılımları yüklebilmek için zararlı yazılım geliştiricileri çeşitli yöntemler kullanmaktadırlar. Gerçek uygulamaların içerisine zararlı aktivite yapan kod parçaları yerleştirerek zararlı uygulama üretilebildiği gibi, Google Uygulama Mağazasında yer alan gerçek uygulamaları isim ve logo gibi unsurlarını taklit ederek zararlı amaçlar güden uygulamalar da üretilebilir. Zararlı yazılım geliştiricileri üçüncü parti uygulama mağazalarını tercih etseler de, Google Uygulama Mağazası da zararlı yazılım geliştiricileri için bir dağıtım merkezi görevi görmektedir [14, 9]. Bu konuda yapılan çalışmalarda Zhou ve ark. altı Android uygulama mağazasında yaptığı çalışmada tekrar paketlenmiş

uygulamaları incelediğinde bu uygulamaların sadece zararlı yazılım barındırmadığını, aynı zamanda reklam geliri elde etmek için çeşitli kütüphaneler de barındırdığını tespit etmiştir [15].

Zararlı yazılım geliştiricileri uygulamalar içerisinde yer alan reklamlar aracılığıyla da kullanıcılarına zararlı yazılımlar indirtebilmektedir. Bu yöntem için Google Uygulama Mağazasına yükledikleri ve görünürde zararlı bir aktivite yapmayan bir uygulama kullanarak internet, device admin ve paket yükleme gibi yetkiler aracılığıyla zararlı aktiviteler gösterecek olan asıl uygulamayı indirtebilmektedirler. Bilinen ve güncel Android zararlı uygulamaları, Koodous ve VirusTotal platformlarından indirilebilmektedir [16, 17, 18].

Android işletim sisteminde çalışan zararlı uygulamaların analizi ve tespiti x86 zararlı yazılım analizinin temellerine dayanmaktadır. 2008 yılında Android telefonların çıkmaya başlamasıyla birlikte uygulama analizi için Android işletim sistemine yönelik pek çok araç geliştirilmiştir. Analiz araçları uygulamaların işlevini analiz etmek için kullanıldığı gibi, bir uygulamanın zararlı olup olmadığına karar vermek için kullanılabilen araçlar da bulunmaktadır. Statik analiz teknikleri Android uygulama paketinden (APK) uygulamaya dair özelliklerin çıkarılmasında kullanılmaktadır. Dinamik analiz teknikleri ise uygulamanın kontrollü bir ortamda çalıştırılarak uygulamanın gösterdiği davranışın analiz edilmesinde kullanılmaktadır.

Statik analiz sonucunda elde edilen bilgiler uygulamanın dinamik analizi için hayati öneme sahip olabilir. Buna örnek olarak uygulamada yer alabilecek olan root tespiti ve emulator tespiti teknikleri gösterilebilir [19]. Statik analizde keşfedilen tespit yöntemleri doğrultusunda ilgili tespit teknikleri atlatılarak daha sağlıklı dinamik analiz yapılması sağlanabilmektedir. Ayrıca, uygulamanın gösterebileceği ve statik analiz yoluyla keşfedilmiş gizli davranışlar da bu sayede dinamik analiz ile incelenebilmektedir. Aşağıda farklı statik ve dinamik analiz yöntemleri ve araçları hakkında bilgiler verilecektir. Anlatılacak bilgiler, tasarlanan Android kum havuzunun tanıtılmasında yardımcı olacaktır.

A. STATİK ANALİZ ARAÇLARI

Statik analiz araçları aşağıdaki kategorilerden birinde yer alabilir:

Decompilerlar: Dalvik byte kodu seviyesinde decompile ya da disassembly yapan araçlar.

Metadata çıkarıcılar: Uygulamanın AndroidManifest dosyasından bilgi çıkaran ve istenen izinler, aktiviteler, servisler ve yayın alıcılar hakkında bilgi sunan araçlar. Bu araçlar tarafından çıkarılan bilgiler dinamik analiz süreçlerinde kullanılabilmektedir.

Weaving: Uygulamada byte kodu seviyesinde değişiklik yapan araçlar.

Statik analiz için kullanılan araçlardan birisi Androguard aracıdır [20]. Dalvik byte kodunu Java kaynak koduna çevirebilir. İki APK dosyası verildiğinde, benzerlik oranını çıkarabilir. Benzerlik oranı çıkarma işlemi tekrar paketlenmiş uygulamalar ya da bilinen zararlı uygulamaların tespiti için kullanılmaktadır. Ayrıca, Androguard aracında yer alan bileşenler kullanılarak AndroidManifest dosyası içerisindeki bilgiler çıkarılabilir.

APKtool aracı, Android uygulamalarını tersine mühendislik yöntemleriyle incelemek için kullanılan bir araçtır. Android paketlerini neredeyse orijinal haliyle geri dönüştürme yeteneğine sahiptir. APKtool aracı ile bir Android uygulaması açılabilir, yeni özellikler eklenebilir ve tekrar paketlenebilir. Bu özelliğiyle byte kodu seviyesinde değişiklik yapmaya olanak tanımaktadır. Weaving yöntemi Joe Sandbox tarafından statik olarak kullanılmaktadır. Radare2 açık kaynak bir tersine mühendislik aracıdır. Disassemble, debug, analiz ve Android binary dosyaları manipüle etme özelliklerine sahiptir.

Android platformuna yönelik statik analiz ile alakalı diğer araçlar .class dosyasına yöneliktir. Android uygulamaları içerisinde yer alan .class dosyaları hali hazırda kullanılan Java araçları ile

işlenebilmektedir. Dex2jar aracı bir APK dosyasını .jar dosyasına çevirebildiği gibi, aynı işlemin tersini de yapabilmektedir. JEB Decompiler, lisanslı bir Android decompiler aracıdır. Dalvik byte kodunu direkt olarak Java kaynak koduna çevirebildiği gibi kaynak dosyaları, sertifika, manifest dosyası gibi bileşenleri de geri getirebilmektedir.

B. DİNAMİK ANALİZ ARAÇLARI

Dinamik analiz araçları şüpheli uygulamaları kontrollü bir ortamda çalışırken izleme ve izlenen uygulamaların davranışlarını çıkarma özelliklerine sahip araçlardır. Dinamik analiz araçları aşağıdaki teknikleri kullanarak bir uygulamanın davranışlarını izleyebilir:

Sistem seviyesinde izleme: Kullanılan sistem çağrılarının takibini yapan strace ya da kernel modülleri gibi araçlar sistem seviyesinde izleme için kullanılır. Native kodların kısmen takip edilmesine olanak sağlar.

Sanal makina seviyesinde gözlem: VMI tabanlı framework yapıları emülasyon yapılan ortamda olan olaylara müdahale etmektedir. Dalvik VM (DVM) tabanlı sistemler Android API çağrılarını DVM seviyesinde gözlemler. Qemu VMI tabanlı sistemler emulator seviyesinde native kod analizi yapabilmek için geliştirilmiştir.

Metot takibi: DVM içerisinde çalışan Java metotlarını takip eden araçlar.

Değişiklik takibi: Dinamik analiz sırasında şüpheli uygulamanın değiştirdiği ya da eriştiği kullanıcı bilgilerini tespit etmeye yarayan araçlar.

İlk dinamik Android analiz frameworklerinden birisi olan AASandbox, yüklenebilir kernel modülleri ile sistem çağrılarını izleyerek zararlı uygulama tespiti yapmaya çalışmaktadır [1]. TaintDroid popüler bir değişiklik takip frameworküdür [4]. DVM üzerine yazılmıştır ve uygulamaları hassas bilgi ifşasına karşı gözlemlemektedir. Web arayüzü üzerinden erişilebilen Joe Sandbox Mobile APK Analyzer, ForeSafe, VisualThreat gibi dinamik analiz platformları analiz için kullandıkları teknikleri açıklamadıkları için bu platformlarda kullanılan yaklaşımların analiz edilebilmesi için yeterli bir bilgi bulunmamaktadır.

C. DİĞER ANALİZ ARAÇLARI

Online servis olarak çalışan ve Android uygulama analizi için kullanılan araçlar bu kategoride yer almaktadır. VirusTotal ve AndroTotal anti-virüs tarama servisleri bu kategoriye girmektedir [18, 21]. VirusTotal statik olarak antivirüs tabanlı tarama yaparak bir uygulamanın zararlı olup olmadığına karar vermektedir. AndroTotal şüpheli uygulamayı çalıştırarak mobil zararlı yazılım tespit eden antivirüslere karşı test ettirmektedir.

III. HİBRİT KUM HAVUZU VE SİSTEM MODELLEMESİ

Sistem tasarımı gerçekleştirilmeden önce Android zararlı yazılım analizinde kullanılan kum havuzlarının analiz yetenekleri incelenmiştir. İnceleme için Lastline, FireEye, VxStream ve Joe Sandbox platformları kullanılmıştır. Lastline, FireEye ve VxStream kum havuzları Android zararlı yazılımları için sadece statik analiz yapmaktadır. Joe Sandbox kum havuzu Android zararlı yazılımları için statik ve dinamik analiz yeteneklerine sahiptir. Ele alınan bu platformların inceleme sonuçları Tablo 1 içerisinde görülebilir.

Kum Havuzları	Statik Analiz Yeteneği	Dinamik Analiz Yeteneği
Lastline	Var	Yok
FireEye	Var	Yok
VxStream	Var	Yok
Joe Sandbox	Var	Var

Tablo 1. *Kum havuzlarının Android analizi yönünden karşılaştırmaları*

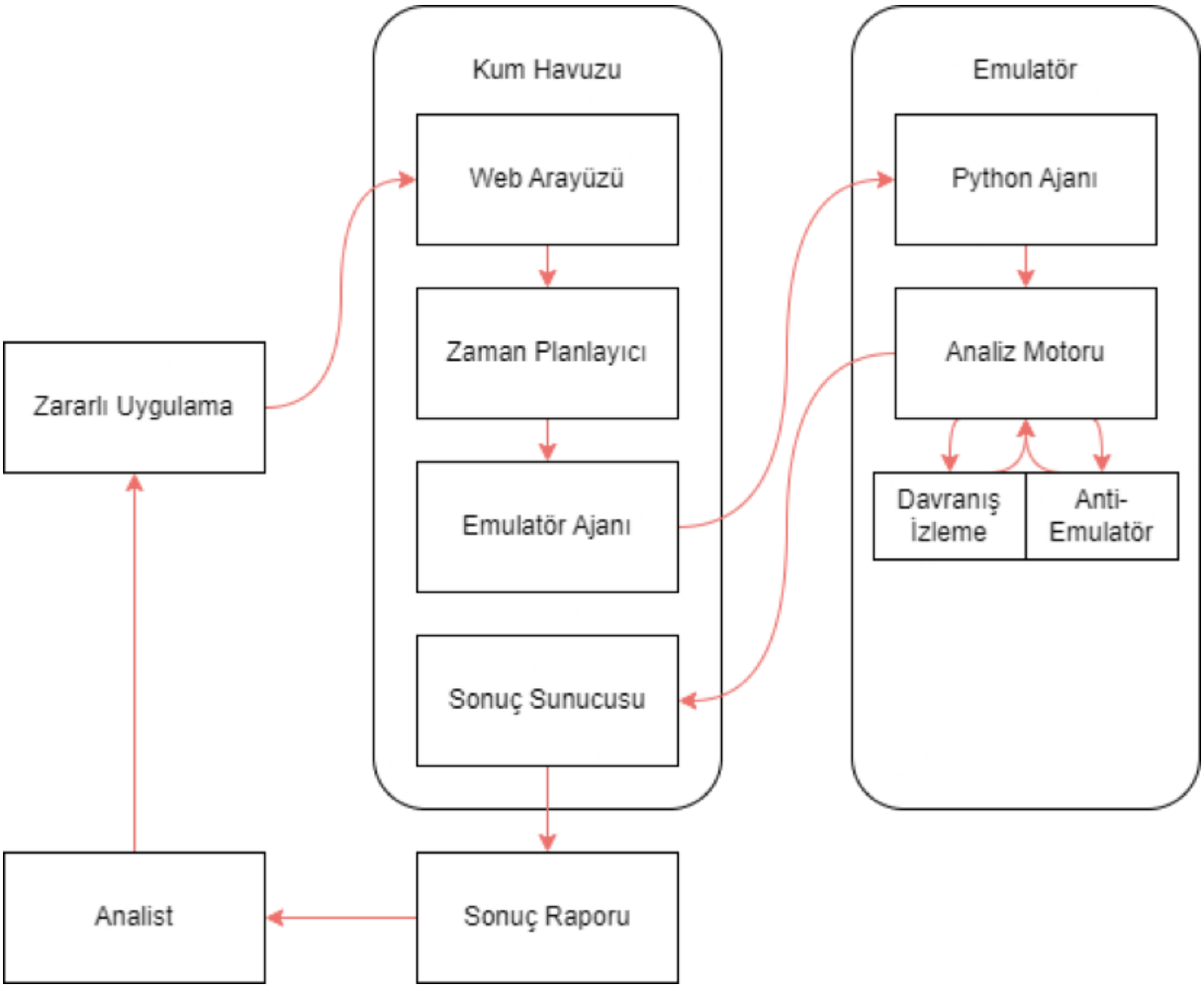
Tasarlanan kum havuzu hem statik hem dinamik analiz yeteneklerine sahiptir. Bu bölümde kum havuzunun statik ve dinamik analiz yeteneklerine dair detaylar ve bu yeteneklerin kum havuzunun bütününe nasıl etki ettiği anlatılacaktır.

Kum havuzu tarafından incelenen her Android uygulaması aşağıdaki adımlardan geçmektedir:

Statik Analiz: Uygulamanın byte kodundan ve manifest bilgilerinden (AndroidManifest.xml) çıkarılan bilgilerin analiz edilmesi işlemidir.

Dinamik Analiz: Uygulamanın Android ortamında çalıştırılması ve davranışlarının DVM seviyesinde takip edilmesi işlemidir.

Raporlama: Analiz sürecinin bitmesinden sonra elde edilen verilerin YARA kural veritabanı ve zararlı uygulama imza veritabanı içerisinde yer alan verilerle eşleştirme işlemidir. Bu işlem sonucunda tespit edilen YARA kural ve imza bilgileri ile statik analiz ve dinamik analiz detaylarını içeren bir rapor üretilmektedir. Analiz sürecinin şeması Şekil 1 içerisinde görülebilir.



Şekil 1. *Sistem analiz şeması*

A. STATİK ANALİZ

Android uygulamaları Android Uygulama Paketi (APK) olarak paketlenmiş bir dosyadır. İçerisinde manifest dosyası barındırır. Bu manifest dosyası, uygulamanın kurulumu ve çalıştırılması için gerekli bir dosyadır. Statik analizin ilk adımı olarak APK'nın açılması ve Java metotlarının, AndroidManifest içerisinden uygulama tarafından istenen izinler, servisler, yayın alıcılar, aktiviteler, paket adı ve SDK versiyonu gibi uygulama alt bilgilerinin ve string değerlerinin çıkarılması yer almaktadır. AndroidManifest içerisinde bulunan servis, yayın alıcı ve aktivite parçalarının tanımları aşağıdaki gibidir:

Aktiviteler: Aktiviteler kullanıcılarla etkileşim için sunulan ekranlara verilen isimdir. Bir aktivite çalışmak için AndroidManifest içerisinde tanımlı olmalıdır. Aktiviteler kullanıcılara sunulan ekranlardaki etkileşimleri ve bu etkileşimlerin sonucunun ne olacağını belirlemektedir. Kum havuzu dinamik analiz sırasında çalıştırılan aktivitelerin faaliyetlerini kayıt altına almaktadır.

Servisler: Android platformunda arka planda çalışan süreçlerdir. Aktivitelerden farklı olarak bir arayüz barındırmamaktadır. Arayüzleri bulunmadığı için zararlı yazılım geliştiricilerinin yaygın olarak kullandığı bileşenlerdendir. Zararlı yazılım geliştiricileri tarafından komuta kontrol sunucusu iletişimi, kişisel veri kaçırma ya da SMS mesajı kaçırmak için kullanılmaktadır. Uygulama tarafından kullanılan tüm servisler AndroidManifest içerisinde tanımlı olmalıdır. Kum havuzu dinamik analiz sırasında çalışan servis faaliyetlerini kayıt altına almaktadır.

Yayın Alıcılar: Yayın alıcılar sistem içerisinden ya da farklı uygulamalar yayımlanan aktiviteleri almak için kullanılmaktadır. Android işletim sistemi açıldıktan sonra işletim sisteminin yayınladığı BOOT_COMPLETED aktivitesi ya da bir SMS geldiğinde haberdar olmak için kullanılan SMS_RECEIVED aktivitesi örnek yayınlardandır. Yayın alıcılar diğer parçalardan farklı olarak AndroidManifest içerisinde tanımlanmak zorunda değildir. Kum havuzu dinamik analiz sırasında çalıştırılan yayın alıcılar ve faaliyetlerini kayıt altına almaktadır.

B. DİNAMİK ANALİZ

Android işletim sistemi akıllı telefon ve tabletler için tasarlandığından yaygın olarak ARM tabanlı cihazlarda çalışmaktadır. Dinamik analiz ortamının gerçek bir telefonu en yakın şekilde simüle etmesi için kum havuzunun dinamik analiz ortamı da ARM platformu ile hazırlanmıştır. Emülasyon ortamı olarak Android uygulamalarını çalışabileceği ve davranışlarının gözlenebileceği QEMU tabanlı bir ortam kullanılmıştır. Tasarlanan ortam ile Android işletim sistemi içerisindeki uygulamaların çalıştığı VM olan Dalvik VM yakından gözlemlenmiş ve incelenen uygulama tarafından çalıştırılan tüm aktiviteler kayıt altına alınmıştır. Dinamik analizin Dalvik VM seviyesinde yapılması dosya sistemi, arama ve SMS gibi aktivitelerin de gözlemlenebilmesine olanak sağlamıştır. Kapsamlı bir analiz için toplanan bu bilgiler yeterli değildir. Bu yüzden kum havuzu dinamik analiz sırasında zararlı davranışları detaylandırmak için aşağıda detaylandırılan ek bilgiler de toplamaktadır.

B. 1. Hassas Bilgi Kaçırma Tespiti

Hassas bilgi çalınması davranışına karşı telefondan çıkarılmaya çalışılan hassas bilgiler tespit edilmektedir. Hassas bilgi tespitinin yapılması kum havuzuna kabul edilebilir bir ekstra bir yük bindirmektedir. Fakat bu özellik sayesinde kum havuzu ağ üzerinden ya da SMS üzerinden telefondan çıkarılan bilgileri tespit edebilmektedir.

B. 2. Ağ Trafiğinin Yakalanması

Ağ trafiğinin yakalanması modern zararlı yazılımların analizi için en önemli faktörlerden biridir. Zararlı yazılımın komuta kontrol sunucusu ile iletişiminin yakalanması buna bir örnektir. Yapılan çalışmalara göre x86 zararlı yazılımlarının %98'inden fazlası TCP/IP bağlantısı kurmaktadır [22]. Ağ trafiği yaratan uygulamaların çoğunlukta olması tek başına bu metriği barındıran bir uygulamanın zararlı yazılım olarak nitelendirilmesini engellemektedir. Bu yüzden, analiz sırasında telefon üzerinde üretilen tüm ağ trafiği yakalanmaktadır. Bunun sebebi incelenen uygulamanın internet izni olmasa bile tarayıcı gibi başka uygulamalar üzerinden veri kaçırabileceği ya da iletişim kurabileceği ihtimalidir.

B. 3. Metot Takibi

Çalıştırılan Java metotları, komutlar, bu metot ve komutlarda kullanılan parametreler kayıt altına alınmaktadır. Uygulama tarafından çalıştırılan metot ve komutların listesi analiz sonucu üretilen raporda yer almaktadır.

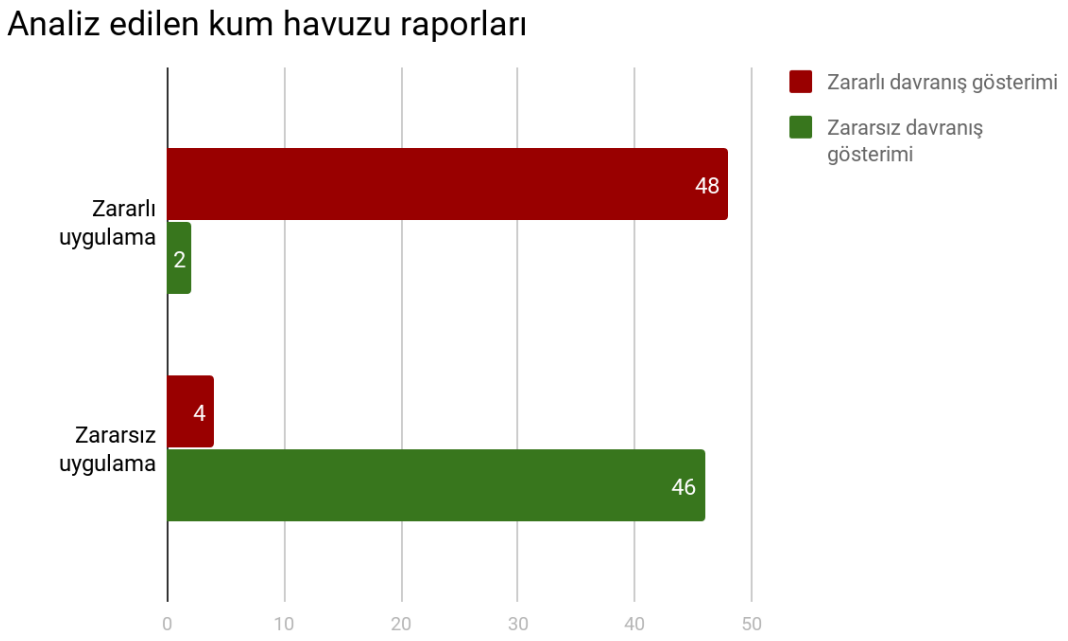
B. 4. Native Kütüphane Tespiti

Android uygulamaları Java Native Interface (JNI) kullanarak sistem seviyesi kütüphanelerin kullanılması ile yazılan native kodları da çalıştırabilmektedir. Bu özellik 3D grafik gibi performans tabanlı işlemler için düşünülmüş bir özelliktir. Eğer incelenen uygulama tarafından kullanılıyorsa, yüklenen native dosyalar kayıt altına alınmaktadır.

Kum havuzunun kurulum ortamı, ağ kurulumu, veritabanı kurulumu gibi sistem unsurları diğer kum havuzları ile benzerdir. Zararlı yazılımların analiz ortamına zarar vermemesi için DoS saldırılarına, e-mail ya da SMS spamları yollamasına ve ağ içerisinde yayılmalarına karşı önlemler içermektedir. Bu önlemler x86 zararlı yazılım analizlerinde karşılaşılan vakalardan yola çıkılarak alınmıştır [23, 24].

IV. SONUÇ

Kum havuzunun hibrit analiz yeteneklerinin değerlendirilmesi için 50 zararlı ve 50 zararsız uygulama seçilmiştir. 50 zararsız uygulamanın hepsi Google Uygulama Mağazasından rastgele seçilmiştir. Zararlı yazılımlar ise VirusTotal ve Koodous platformlarından antivirüsler tarafından zararlı olarak nitelendirilen örnekler içerisinden rastgele seçilmiştir. Analizler sırasında herhangi bir davranış göstermeyen uygulamalar test örnekleri arasından çıkarılmıştır ve yerine yeni örnekler seçilmiştir. Örnekler rastgele seçilerek zararlı yazılım çeşitliliği mümkün olduğunca üst seviyede tutulmaya çalışılmıştır.



Şekil 2. *Test sonuçları*

Test örnekleri ile yapılan çalışmalar sonucunda kum havuzunun hibrit analiz sonucu üretilen raporun analistin bir uygulamanın zararlı olup olmadığına karar vermesi için yeterli olduğu görülmüştür. Bazı zararlı uygulamaların sahip olduğu gelişmiş anti teknikler, bu zararlı uygulamaların kum havuzu analizini atlatmasına olanak sağlamıştır. Aynı zamanda, bazı gerçek uygulamaların sahip olduğu zararlı benzeri davranışlar ilgili uygulamanın zararlı olarak nitelendirilmesine sebebiyet vermiştir.

Ayrıca, ilgili sistem tasarlanmadan önce yapılan akademik taramalarda tasarlanan kum havuzuna benzer çalışmalar incelenmiştir. Gilbert ve ark. zararlı aktivitelerin tespiti konusunda yaptığı çalışmada bağımlılık grafiklerine yer vermiştir [16]. DroidScope VMI üzerinden dinamik analiz yapan bir sistem önermiştir [25]. Java objelerinin analiz için tekrar oluşturulma işlemi gibi hassas işlemler barındırdığından Google LLC'nin her güncellemesinde DroidScope içerisinde büyük adaptasyonlar yapılması gerektiğinden kullanılması makul değildir. I. Burguera ve ark. tarafından geliştirilen CrowdDroid davranış tabanlı dinamik analiz sistemi önermiştir fakat Android işletim sisteminin özelliklerini yeteri kadar kapsayamamıştır [11]. Hibrit Android kum havuzu dinamik analiz ile davranış tespiti yapmaktadır ve davranış imza veritabanı kullanarak bilinen zararlı davranışları tespit etmektedir. İncelenen çalışmalar hibrit zararlı yazılım analizi yapabilecek bir kum havuzu ortamının oluşturulmasında yetersiz kalmaktadır. Geliştirilen sistem, bu bilgiler ışığında hibrit analiz yetenekleri olan bir kum havuzu elde etme amacıyla geliştirilmiştir.

Hibrit Android kum havuzunun amacı statik ve dinamik analiz yaparak raporlamak ve zararlı yazılım analistlerinin Android uygulama örnekleri içerisinden hangilerinin zararlı olabileceğine ve detaylı incelemeye gerek duyduğuna dair verecekleri kararı desteklemektir.

Otomatize bir analiz ortamının en büyük problemi sanal cihaz tabanlı olmasından kaynaklı olarak anti teknikleridir. Zararlı uygulamaların kullandığı anti-VM ve anti-sandbox teknikleri gelişmekte ve değişmektedir. Kum havuzu bünyesinde alınan önlemler karmaşık zararlı yazılımları kandırmakta yetersiz kalabilmektedir. Yapılan çalışmalar x86 zararlı yazılımlarda anti tekniklerinin yaygın olarak kullanılmadığını gösterse dahi Android tabanlı kum havuzları için bu varsayımın doğru olduğunu kanıtlayan bir çalışma bulunmamaktadır [19, 16].

Google Uygulama Mağazası için tanıtılan Bouncer teknolojisi göz önünde bulundurulduğunda Android zararlı uygulama geliştiricilerinin analiz ortamları tespiti üzerinde çaba harcayacakları tahmin edilebilir. Yapılan testlerde Şekil 2'de görüldüğü üzere kum havuzu %96 oranında başarıma, %8 yalancı pozitif ve %4 yalancı negatif oranına sahiptir. Başarım oranının %100 olmama sebebi yukarıda bahsedilen karmaşık zararlı uygulamaların kullandığı teknikler olarak değerlendirilmektedir. Yine de, kum havuzunun incelenecek uygulama sayısının fazla olduğu durumlarda analistlere sağladığı rapor, analistlerin hangi uygulamanın detaylı inceleme gerektirdiğine karar vermesi konusunda kolaylık ve hız kazandırmaktadır.

Mükemmel bir emülasyon ortamı geliştirilmedikçe zararlı uygulama geliştiricileri için ortam tespit ihtimali her zaman bulunacaktır. Bu yüzden, ortam tespitini zorlaştırmak için adımlar atarak zararlı uygulama geliştiricisinin işini mümkün olduğunca zorlaştırmak bu konuda atılabilecek en iyi adımdır. Bu makalede Android işletim sistemi için geliştirilen zararlı uygulamaların statik ve dinamik analizini yapabilen hibrit bir kum havuzu sunulmuştur. Sunulan sonuçlar kum havuzunun analiste herhangi bir uygulamanın zararlı olup olmadığına ve detaylı incelemeye gerek olduğunu tespit etme konusunda yol gösterdiğini göstermektedir.

V. REFERANSLAR

[1] T. Bläsing, L. Batyuk, A. -D. Schmidt, S. A. Camtepe and S. Albayrak, "An Android Application Sandbox system for suspicious software detection," 2010 5th International Conference on Malicious and Unwanted Software, Nancy, France, 2010, pp. 55-62, doi: 10.1109/MALWARE.2010.5665792.

[2] Reina, Alessandro, Aristide Fattori and Lorenzo Cavallaro, "A System Call-Centric Analysis and Stimulation Technique to Automatically Reconstruct Android Malware Behaviors.", 2013.

[3] Spreitzenbarth, Michael, Felix C. Freiling, Florian Echtler, Thomas Schreck and Johannes Hoffmann, "Mobile-sandbox: having a deeper look into android applications.", ACM Symposium on Applied Computing, 2013.

[4] Enck, William & Gilbert, Peter & Chun, Byung-Gon & Cox, Landon & Jung, Jaeyeon & McDaniel, Patrick & Sheth, Anmol, TaintDroid: An Information-Flow Tracking System for Realtime Privacy Monitoring on Smartphones, Communications of the ACM, 2010, pp. 57, doi: 10.1145/2494522.

[5] IDC, "Android and iOS Continue to Dominate the Worldwide Smartphone Market with Android Shipments Just Shy of 800 Million in 2013," http://www.idc.com/getdoc.jsp?containerId=prUS24676414 (2023.07.09).

[6] V. Svajcer, "Sophos Mobile Security Threat Report," http://www.sophos.com/en-us/medialibrary/PDFs/other/sophos-mobile-security-threat-report.ashx (2023.07.09).

[7] H. Lockheimer, “Android and Security,” http://googlemobile.blogspot.com/2012/02/Android-and-security.html (2023.07.09).

[8] Lookout, Pegasus for Android (April 2017).

[9] Google, An investigation of chrysaor malware on Android (2023.07.09).

[10] D. Maslennikov, “First SMS Trojan for Android,” https://www.securelist.com/en/blog/2254/First SMS Trojan for Android, August 2010.

[11] Burguera, Iker & Zurutuza, Urko & Nadjm-Tehrani, Simin, Crowdroid: Behavior-Based Malware Detection System for Android, SPSM '11, 2011, pp. 15-26, doi: 10.1145/2046614.2046619.

[12] Grace, Michael & Zhou, Wu & Jiang, Xuxian & Sadeghi, Ahmad-Reza, Unsafe Exposure Analysis of Mobile In-App Advertisements ABSTRACT, WiSec'12 - Proceedings of the 5th ACM Conference on Security and Privacy in Wireless and Mobile Networks, 2012, doi: 10.1145/2185448.2185464.

[13] Zhou, Yajin & Wang, Zhi & Zhou, Wu & Jiang, Xuxian, Hey, You, Get Off of My Market: Detecting Malicious Apps in Official and Alternative Android Markets, Proceedings of the 19th Network and Distributed System Security Symposium NDSS 2012, 2012.

[14] CheckPoint, Charger malware calls and raises the risk on google play (2023.07.09).

[15] Zhou, Wu, Yajin Zhou, Xuxian Jiang and Peng Ning, “Detecting repackaged smartphone applications in third-party android marketplaces.”, Conference on Data and Application Security and Privacy, 2012.

[16] Gilbert, Peter & Chun, Byung-Gon & Cox, Landon & Jung, Jaeyeon, Vision: Automated security validation of mobile apps at app markets, Proceedings of the Second International Workshop on Mobile Cloud Computing and Services, 2011, doi: 10.1145/1999732.1999740.

[17] “Koodous”, https://koodous.com (2023.07.09).

[18] “VirusTotal”, https://virustotal.com (2023.07.09).

[19] D. Shi, X. Tang and Z. Ye, "Detecting environment-sensitive malware based on taint analysis," 2017 8th IEEE International Conference on Software Engineering and Service Science (ICSESS), Beijing, China, 2017, pp. 322-327, doi: 10.1109/ICSESS.2017.8342924.

[20] “Androguard”, https://github.com/androguard/androguard (2023.07.09).

[21] Maggi, Federico, Andrea Valdi and Stefano Zanero, “AndroTotal: a flexible, scalable toolbox and service for testing mobile malware detectors.”, Security and Privacy in Smartphones and Mobile Devices, 2013.

[22] Kapratwar, Ankita & Di Troia, Fabio & Stamp, Mark, Static and Dynamic Analysis of Android Malware, 2017, pp. 653-662, doi: 10.5220/0006256706530662.

[23] Bayer, Ulrich & Kruegel, Christopher & Kirda, Engin, TTAnalyze: A Tool for Analyzing Malware, 2006.

[24] Xu Chen, J. Andersen, Z. M. Mao, M. Bailey and J. Nazario, "Towards an understanding of anti-virtualization and anti-debugging behavior in modern malware," 2008 IEEE International Conference

on Dependable Systems and Networks With FTCS and DCC (DSN), Anchorage, AK, USA, 2008, pp. 177-186, doi: 10.1109/DSN.2008.4630086.

[25] Kwong, Lok & Yin, Heng, DroidScope: Seamlessly Reconstructing the OS and Dalvik Semantic Views for Dynamic Android Malware Analysis, 2012, Proceedings of the 21st USENIX Security Symposium.