T.C.
SAKARYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

ANDROİD PLATFORMUNDA ZARARLI YAZILIM ANALİZİ İÇİN HİBRİT KUM HAVUZU GELİŞTİRİLMESİ

YÜKSEK LİSANS TEZİ

Mert Can COŞKUNER

Enstitü Anabilim Dalı	**:**	**BİLGİSAYAR MÜHENDİSLİĞİ**
Enstitü Bilim Dalı	**:**	**SİBER GÜVENLİK**
Tez Danışmanı	**:**	**Dr. Öğr. Üyesi Murat İSKEFİYELİ**

Haziran 2019

T.C.
SAKARYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

ANDROID PLATFORMUNDA ZARARLI YAZILIM ANALİZİ İÇİN HİBRİT KUM HAVUZU GELİŞTİRİLMESİ

YÜKSEK LİSANS TEZİ

Mert Can COŞKUNER

Enstitü Anabilim Dalı	:	BİLGİSAYAR MÜHENDİSLİĞİ
Enstitü Bilim Dalı	:	SİBER GÜVENLİK

Bu tez 11.06.2019 tarihinde aşağıdaki jüri tarafından oybirliği / ~~oyçokluğu~~ ile kabul edilmiştir.

Prof. Dr. Resul KARA	Doç. Dr. İbrahim ÖZÇELİK	Dr. Öğr. Üyesi Murat İSKEFİYELİ
Jüri Başkanı	Üye	Üye

BEYAN

Tez içindeki tüm verilerin akademik kurallar çerçevesinde tarafımdan elde edildiğini, görsel ve yazılı tüm bilgi ve sonuçların akademik ve etik kurallara uygun şekilde sunulduğunu, kullanılan verilerde herhangi bir tahrifat yapılmadığını, başkalarının eserlerinden yararlanılması durumunda bilimsel normlara uygun olarak atıfta bulunulduğunu, tezde yer alan verilerin bu üniversite veya başka bir üniversitede herhangi bir tez çalışmasında kullanılmadığını beyan ederim.

Mert Can COŞKUNER

09.05.2019

TEŞEKKÜR

Yüksek lisans eğitimim boyunca değerli bilgi ve deneyimlerinden yararlandığım, her konuda bilgi ve desteğini almaktan çekinmediğim, araştırmanın planlanmasından yazılmasına kadar tüm aşamalarında yardımlarını esirgemeyen, teşvik eden, aynı titizlikte beni yönlendiren değerli danışman hocam Dr. Murat İSKEFİYELİ'ye, beni destekleyen arkadaşlarım Emre Can KURUÇAY, Kaan PİYALE ve Metehan KARATAŞ'a ve her zaman yanımda olan Ayşenur Kübra SÜRÜCÜ'ye teşekkürlerimi sunarım.

İÇİNDEKİLER

ŞEKİLLER LİSTESİ

TABLOLAR LİSTESİ

ÖZET

Anahtar kelimeler: Android, kum havuzu, zararlı yazılım analizi

Mobil telefon endüstrisi son yılların en hızlı gelişen endüstrilerinden biri olmuştur. Bu gelişmeler ışığında Android işletim sisteminin akıllı telefonlar içerisinde büyük bir payda elde etmesinin bir yan etkisi olarak Android işletim sistemi zararlı yazılım geliştiricilerinin de ilgini 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 bağlamda yapılan akademik çalışmalar [1, 2, 3, 4] ve ortaya çıkan prototipler erişilebilirlik ve analiz yapabilme kapasitesi olarak yetersiz kalmıştır. 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 hibrit analiz yeteneklerine sahip bir android kum havuzu geliştirilmiştir.

IMPLEMENTING HYBRID ANDROID SANDBOX FOR MALWARE ANALYSIS ON ANDROID PLATFORM

SUMMARY

Keywords: Android, sandbox, malware analysis

Mobile device industry is one of the fastest growing industry in recent years. In the light of this growth, Android operating system is increasing its user base which also attracts malware developers as well. Increasing android malware is becoming a problem for analysts in order to analyse and decide whether application is malicious or benign. Researches for this problem [1, 2, 3, 4] and prototypes is not enough in terms of accessibility and analysis capabilities. In this paper, hybrid android sandbox for both dynamic analysis and static analysis is proposed while comparing malware sandboxes already used for malware analysis. As a result of this paper, hybrid android sandbox which performs static analysis and dynamic analysis as well as reports analysis results for analysts is designed.

BÖLÜM 1. GİRİŞ

Zararlı yazılımlarla mücadele etmek zararlı yazılımların hızlı yayılan ve sürekli değişen doğası gereği zor olabilir. Virüs tarayıcılar gibi pek çok güvenlik çözümü imza tabanlı zararlı kod arayarak zararlı yazılım tespiti yapmaya çalışmaktadır. Zararlı yazılım geliştiriciler de bunun bilincinde olarak kendilerini bu gibi çözümleri atlatmaya adapte etmişlerdir. Böyle hızlı ve değişken bir ortamda zararlı yazılım analistleri yazılımları sadece tersine mühendislik yoluyla manuel olarak inceleyerek geri kalmaktadırlar. Bu yüzden, zararlı yazılım analizinin otomatizasyonu önem arz etmektedir. Burada otomatizasyon ile bahsedilen analistin müdahalesi olmadan bir zararlı yazılımın davranış raporunun çıkarılabilmesidir. Çıkarılan rapor doğrultusunda analistlerin hareket etmesi manuel bir şekilde yapılacak olan tersine mühendislikten daha hızlı olacaktır. Bu doğrultuda tasarlanacak aracın efektif olabilmesi için zararlı yazılımın davranışlarını kayıt altına alması gerekmektedir, fakat bunu yaparken kayıt altına aldığı davranış sayısı az olmamalıdır. Günün sonunda, ortaya çıkan raporu inceleyecek ve karar verecek olan analiste karar verebilmesi için gerekli ve yeterli verinin üretilmesi gerekmektedir. Yeterli veri üretildiği takdirde yalancı pozitif oranı mümkün olduğunca düşürülebilir.

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 ilgini 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. Zararlı yazılım analiz yöntemleri statik ve dinamik analiz olmak üzere iki yöntem ile yapılmaktadır. Bir uygulamanın zararlı olup olmadığını anlamak için kullanılabilecek yöntemlerden birisi; şüpheli uygulamayı kontrollü bir telefona yüklemek, telefonun sistemindeki ve ağ trafiğindeki değişiklikleri izlemektir. Bu analiz yöntemine dinamik analiz adı verilmektedir [25]. Dinamik analiz işlemi, dinamik analiz kum havuzu kullanılarak yapıldığında bir zararlı yazılım analistinin yapacağına göre kısa sürede zararlı yazılım hakkında bilgi elde edebilir. 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 [5] yakınını elinde bulunduran Android işletim sistemi akıllı telefonlar için en popüler işletim sistemi olmakla beraber tek rakibi Apple'ın iOS işletim sistemidir. Bu popülerlikten doğal olarak siber suçlular da haberdardır. iOS işletim sisteminin aksine, Android işletim sisteminin üçüncü parti mağazalardan uygulama indirip kurulmasına izin vermesi de siber suçluların Android işletim sistemine zararlı uygulama yazarak 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 toplanmıştır ve her gün 2,000 Android zararlısı bu sayıya eklenmeye devam etmektedir [6].

Google da siber suçluların bu artan ilgisini görerek Şubat 2012 tarihinde Bouncer'ı [7] tanıtmıştır. 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 [17]. Fakat, 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 ne kadar uzun süre tespit edilmeden faaliyetlerini devam ettirebileceğinin pek çok örneği mevcuttur [18, 19].

Google'ın 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 kullanıcıya genel olarak rapor olarak sunulmakta olup uygulamanın zararlı olup olmadığına dair bir bilgilendirme içermektedir. Bu sistemlerin pek çoğu dinamik analiz sonuçlarına takviye olması anlamında statik analiz de yaparak hibrit bir yaklaşımla analiz yapmaktadır. İki analizi de bir arada barındıran kum havuzlarının yaptığı analiz şekline ise hibrit analiz denmektedir. 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 [8, 9, 15].

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. Tez çalışmamda, 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 yapabilme özelliği kazandırılmıştır. Bu çalışma ile aşağıdaki katkıların yapılması amaçlanmaktadır:

1. Android uygulamalarının hibrit analizini yapabilen bir kum havuzu geliştirilmesi.
2. Dinamik analiz ile uygulama davranışlarının yakalanması.

BÖLÜM 2. ZARARLI YAZILIM ANALİZİ

Bu başlıkta Android zararlı yazılımlarının tespiti için kullanılan tekniklerin anlaşılabilmesi için güncel mobil zararlı yazılım tehditlerinden bahsedilmiştir. Tehditlerin anlatımında mobil 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. [19]. Bu zararlı yazılım bulaştığı hedeflerde ücretli servislere SMS mesajları atarak gelir elde etme amacı gütmekteydi. İlk Android zararlı yazılım FakePlayer'ın çıkmasının üzerinden geçen bu zamanda mobil zararlı yazılımlar giderek karmaşıklaşmaya başlamıştır. Mobil zararlı yazılım geliştiricilerinin geliştirdikleri zararlı yazılımların karmaşıklığını giderek arttırmalarının arkasındaki motivasyon finansal kazançtır. Zararlı yazılım geliştiricileri, zararlı yazılımlarının bulaştıkları cihazlardan bankacılık bilgilerini çalabilir, FakePlayer örneğinde olduğu gibi ücretli servislerle iletişim kurabilir ya da kişisel veriler çalarak uzak sunucuya yükleyebilir. Daha da ileri giderek 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 yaparak iki faktörlü doğrulama mekanizmalarını atlatacak kapasitelere sahip olabilir.

Kurbanlarına zararlı yazılımları yükletebilmek için zararlı yazılım geliştiricileri çeşitli yöntemler kullanmaktadırlar. Gerçek uygulamaların içerisine zararlı aktivite yapan kod parçaları gömerek zararlı uygulama üretilebildiği gibi, Google Uygulama Mağazasında yer alan gerçek uygulamaları isim ve logo gibi özellikleri ile taklit ederek zararlı amaçlar güden uygulamalar da üretilebilir. Zararlı yazılım

geliştiricileri genellikle üçüncü parti uygulama mağazalarını tercih etseler de, Google uygulama mağazası da zararlı yazılım geliştiricileri için bir dağıtım merkezi görevi görmektedir [16, 18]. 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 [9, 20].

Zararlı yazılım geliştiricileri aynı zamanda uygulamalar içerisinde yer alan reklamlar aracılığıyla da kullanıcılarına zararlı yazılımlar indirtebilmektedir [15]. Başka bir yöntem olarak zararlı yazılım geliştiriciler Google uygulama mağazasına koydukları ve görünürde zararlı bir aktivite yapmayan bir uygulamadan internet, device admin ve paket yükleme gibi yetkiler aracılığıyla zararlı aktiviteler gösterecek olan asıl uygulamayı indirtebilmektedir.

Bilinen ve güncel Android zararlı uygulamaları, Koodous ve VirusTotal platformlarından indirilebilmektedir [23, 22].

2.1. Android Zararlı Yazılım Analizi

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. Bu bölümde Android zararlı yazılım analiz süreçleri ve analiz araçları tartışılacaktır.

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 uygulama kontrollü bir ortamda çalışırken 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 [12]. 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. Statik analiz desteğiyle yapılan dinamik analize hibrit analiz denmektedir.

Aşağıda farklı statik ve dinamik analiz yöntemleri ve araçları hakkında bilgiler verilecektir. Anlatılacak bilgiler, tasarlanan Android kum havuzunun tanıtılmasında yardımcı olacaktır.

2.1.1. Statik analiz araçları

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 en yaygın araçlardan birisi Androguard aracıdır [24]. Dalvik byte kodunu Java kaynak koduna decompile edebilir. 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 genellikle .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, paralı bir android decompiler aracıdır. Dalvik byte kodunu direkt olarak Java kaynak koduna çevirebilmekte ve aynı zamanda APK dosyasının içeriğini disassemble etmekte, böylece kaynak dosyaları, sertifika, manifest dosyası gibi bileşenleri de geri getirebilmektedir.

2.1.2. Dinamik analiz araçları

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: Çalıştırılan sistem çağrılarının takibini yapan VMI, strace ya da kernel modülleri gibi araçlar. Native kodların kısmen takip edilmesine olanak sağlar.

Sanal makina seviyesinde gözlem: VMI tabanlı framework yapıları emülasyon yapılan ortamda olan olaylara müdahale etmektedir. Dalvik VMI tabanlı sistemler

Android API çağrılarını Dalvik VM seviyesinde gözlemler. Qemu VMI tabanlı sistemler emulator seviyesinde native kod analizi yapabilmek için geliştirilmiştir.

Metot takibi: Dalvik VM 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 [1], yüklenebilir kernel modülleri ile sistem çağrılarını izleyerek zararlı uygulama tespiti yapmaya çalışmaktadır.

TaintDroid [25] popüler bir değişiklik takip frameworküdür. Dalvik VM üzerine yazılmıştır ve uygulamaları hassas bilgi ifşasına karşı gözlemlemektedir.

Web arayüzü üzerinden erişilebilen Joe Sandbox Mobile APK Analyzer, ForeSafe, VisualThreat gibi dinamik analiz platformları dokümantasyon olarak az bilgi verdiği ve analiz için ne kullandıkları bilinmediği için bu platformlarda kullanılan yaklaşımlar hakkında bir bilgi bulunmamaktadır.

2.1.3. Diğer analiz araçları

Online servis olarak çalışan ve android uygulama analizi için kullanılan araçlar bu kategoride yer almaktadır. VirusTotal [23] ve AndroTotal [26] anti-virüs tarama servisleri bu kategoriye girmektedir. VirusTotal statik olarak antivirüs tabanlı tarama yaparak bir uygulamanın zararlı olup olmadığına karar vermektedir. AndroTotal de VirusTotal platformuna benzer olarak şüpheli uygulamayı çalıştırarak mobil zararlı yazılım tespit eden antivirüslere karşı test ettirmektedir.

BÖLÜM 3. HİBRİT KUM HAVUZU SİSTEM MODELLEMESİ

Sistem tasarımı gerçekleştirilmeden önce zararlı yazılım analizinde aktif olarak kullanılan kum havuzlarının Android zararlı yazılım analiz yetenekleri incelenmiştir. İnceleme için lastline, FireEye, VxStream ve Joe Sandbox platformları kullanılmıştır. Ele alınan bu platformların inceleme sonuçları Tablo 3.1. içerisinde görülebilir.

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.

Tablo 3.1. Kum havuzlarının android analizi yönünden karşılaştırmaları

Kum havuzları	Statik analiz yeteneği	Dinamik analiz yeteneği
Lastline	Var	Yok
Fireeye	Yok	Yok
VxStream	Var	Yok
Joe Sandbox	Var	Var

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

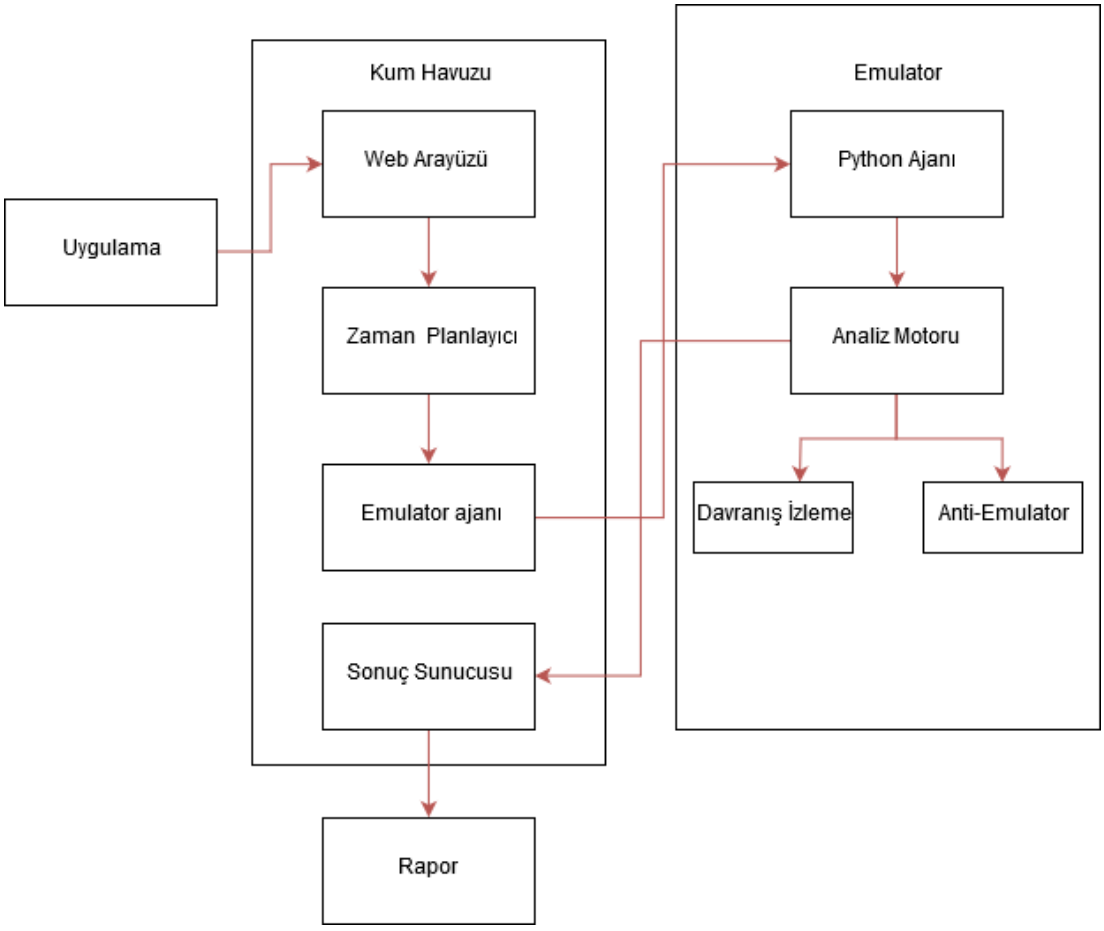
Kum havuzu tarafından incelenen her Android uygulaması aşağıdaki adımlardan geçmektedir:

Statik Analiz: Uygulamanın byte kodundan ve manifest bilgilerinden çıkarılan bilgilerin analiz edilmesi işlemidir.

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

Raporlama: Ana analiz sürecinin bitmesinden sonra elde edilen statik ve dinamik analiz sonucu 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 beraber statik analiz, dinamik analiz ve ağ analiz detaylarını içeren bir rapor üretilmektedir.

Bütün adımların da yer aldığı analiz sürecinin şeması Şekil 3.1. içerisinde görülebilir.



Şekil 3.1. Sistem analiz şeması

Kum havuzuna gönderilen şüpheli yazılımın internet ortamıyla iletişime geçmesi istenmediği takdirde sistem içerisindeki yapıda yer alan emulator internetsiz bir ortamda çalıştırılabileceği gibi, sahte bir internet simülasyonu ile de

çalıştırılabilmektedir. Sistemde yer alan internet kısıtlama ve sahte internet özellikleri sayesinde şüpheli uygulamanın analiz edildiğine dair bir şüphe uyandırılmadan analiz işlemi yapılabilmektedir.

3.1. Statik Analiz

Android uygulamaları Android Uygulama Paketi (APK) olarak paketlenmiş bir dosyadır. İçerisinde manifest dosyası (AndroidManifest.xml) barındırır. Bu manifest dosyası uygulamanın kurulumu ve çalıştırılması için gerekli bir dosyadır. Statik analizin ilk adımı olarak APK'nın açılması, AndroidManifest içerisinden uygulama alt bilgilerinin ve string değerlerinin çıkarılması yer almaktadır. Çıkarılan değerlerin yara kuralları ve imza veritabanında karşılığı olup olmadığına karşılığı var mı yok mu bakılarak bilinen bir zararlı yazılım olup olmadığının tespiti yapılır. Çıkarılan bileşenler ve tanımları aşağıdaki gibidir.

3.1.1. Aktiviteler

Aktiviteler kullanıcılarla etkileşim için sunulan ekranlara verilen isimdir. Bir aktivite çalışmak için AndroidManifest içerisinde tanımlı olmak zorundadır. Aktiviteler kullanıcılara sunulan ekranlardaki etkileşimleri ve bu etkileşimler sonucu ne olacağını belirler. Kum havuzu dinamik analiz sırasında çalıştırılan aktivitelerin faaliyetlerini kayıt altına almaktadır.

3.1.2. Servisler

Android platformunda arka planda çalışan süreçlere servis denmektedir. Aktivitelerden farklı olarak herhangi bir arayüz barındırmazlar. Arayüzleri bulunmadığı için zararlı yazılım geliştiricilerinin en çok ilgilendiği parçalardan bir tanesidir. Zararlı yazılım geliştiricileri tarafından komuta kontrol sunucusu iletişimi, kişisel veri kaçırma ya da SMS mesajları kaçırmak için kullanılmaktadır. Uygulama tarafından kullanılan tüm servisler AndroidManifest içerisinde tanımlı olmak

zorundadır. Kum havuzu dinamik analiz sırasında çalışan servis faaliyetlerini kayıt altına almaktadır.

3.1.3. Yayın alıcılar

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

Statik analiz ile APK dosyası içerisinden çıkarılan bilgiler ve açıklamaları Tablo 3.2.'de yer almaktadır.

Tablo 3.2. Statik analiz ile çıkarılan bilgiler

Çıkarılan bilgiler	Açıklama
String değerleri	APK içerisindeki tüm dosyaların içeriği
Dosya listesi	APK içerisindeki tüm dosyaların isimleri
Paket ismi	Uygulamanın paket ismi
Ana aktivite ismi	Uygulamanın çalıştırıldığında başladığı ilk aktivite ismi
Aktiviteler	Uygulama içerisinde yer alan tüm aktivitelerin listesi
Servisler	Uygulama içerisinde yer alan tüm servislerin listesi
Yayın alıcılar	Uygulama içerisinde yer alan tüm yayın alıcıların listesi
İçerik Sağlayıcılar	Uygulama içerisinde yer alan tüm içerik sağlayıcıların listesi
Native kod barındırma bilgisi	Java kodları içerisinde native kod kontrolü
Dinamik kod barındırma bilgisi	Java kodları içerisinde dinamik kod kontrolü
Yansıma kod barındırma bilgisi	Java kodları içerisinde yansıma kod kontrolü

Çıkarılan bilgiler rapor içerisinde apkinfo başlığı altında toplanarak rapora eklenmektedir.

3.2. Dinamik Analiz

Android işletim sistemi akıllı telefon ve tabletler için tasarlanmış olduğundan dolayı en yaygın olarak ARM tabanlı cihazlarda çalışmaktadır. Dinamik analiz ortamının gerçek telefona en yakın şekilde olması adına kum havuzunun dinamik analizinin yapılacağı ortam olarak ARM platformu kullanılmıştır. Emülasyon ortamı olarak Android uygulamalarını çalıştıracak ve çalıştığı sırada davranışlarının gözlenebileceği qemu tabanlı bir ortam kullanılmıştır. Android uygulamaları java tabanlı olduğundan, Android işletim sistemi içerisinde 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 çeşitli bilgileri de toplamaktadır.

3.2.1. 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.

3.2.2. 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 [14] TCP/IP bağlantısı kurmaktadır. Ağ trafiği yaratan

uygulamaların çoğunlukta olması tek başına bu metriği barındıran bir uygulamanın zararlı yazılım olarak nitelendirilmesini engellemektedir. Bu yüzden, telefondan hassas bilgi çıkarılması takibi noktasında ağ takibi yapılsa dahi telefon üzerinde üretilen tüm ağ trafiği de yakalanmaktadır. Bunun sebebi incelenen uygulamanın internet izni olmasa dahi ya da uygulamanın kendisi ağ trafiği üretmese bile tarayıcı gibi başka uygulamalar üzerinden veri kaçırabileceği ya da iletişim kurabileceği ihtimalidir.

3.2.3. 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.

3.2.4. Native kütüphane tespiti

Varsayılan olarak Android uygulamaları Java ile yazılmaktadır ve APK olarak dağıtılmaktadır. Fakat, Android uygulamaları aynı zamanda Java Native Interface (JNI) kullanarak sistem seviyesi kütüphanelerin kullanılması ile yazılan native kodları da çalıştırabilmektedir. Bu özellik 3D grafikler gibi performans tabanlı işlemler için düşünülmüş bir özelliktir. Eğer varsa, yüklenen native dosyaların hangileri olduğu kayıt altına alınmaktadır.

Kum havuzunun kurulum ortamı, ağ kurulumu, veritabanı kurulumu gibi sistemsel etmenleri diğer kum havuzları ile karşılaştırılabilir benzerliktedir. Zararlı yazılımların analiz ortamına zarar vermemesi için DoS saldırılarına, e-mail ya da SMS spamları yollamasına ve ağ içerisinde yayılmalarına karşı önlemler alınmıştır. Bu önlemler x86 zararlı yazılım analizlerinde karşılaşılan vakalardan yola çıkılarak alınmıştır [10, 11].

Dinamik analiz için yukarıda belirtilen tespitlerin yapılabilmesi, çalışan uygulamanın hooklanarak davranışlarının kayıt altına alınması ile yapılmıştır. Kayıt altına alınan bilgiler Tablo 3.3.'de görülebilir.

Tablo 3.3. Dinamik analiz ile çıkarılan bilgiler

Kayıt altına alınan davranışlar	Açıklama
Kripto anahtarlar	Uygulamanın kripto işlemler için kullandığı şifreleme ve deşifreleme anahtar bilgileri
Yansıma çağrılar	Uygulamanın çalıştırdığı yansıma çağrılar
Çağrı yapılan sistem özellikleri	Uygulamanın mobil cihazdan topladığı sistem özellikleri
Başlatılan aktiviteler	Uygulama tarafından başlatılan aktiviteler
Erişilen dosyalar	Uygulama tarafından erişilen dosyalar
Mobil cihazın parmak izini çıkarmak için kullanılan veriler	Uygulamanın mobil cihazı tanımak için oluşturduğu eşsiz değeri hesaplamak için kullandığı mobil cihaz bilgileri
Başlatılan yayın alıcılar	Uygulama tarafından başlatılan yayın alıcılar
SharedPrefences aktiviteleri	Uygulamanın SharedPrefences altına yazdığı ve okuduğu değerler
Sorgulanan içerik sağlayıcılar ve sorgulanan değerler	Uygulamanın sorgulama yaptığı içerik sağlayıcılar ve sorgu stringi
Encode/Decode edilen base64 değerleri	Uygulama tarafından yapılan base64 işlemleri
Çalıştırılan komutlar ve çıktıları	Uygulama tarafından çalıştırılan komutlar ve komut çıktıları
Veri sızıntısı	Uygulama tarafından sızdırılan veriler
SMS bilgileri	Uygulama tarafından gönderilen ve uygulamanın eriştiği SMS mesajları
Öldürülen süreçler	Uygulamanın çalışmaya başladıktan sonra öldürmeye çalıştığı süreçler
Yüklenen dex dosyaları	Uygulamanın çalışmaya başladıktan sonra yüklediği dex dosyaları
Çağrı bilgileri	Uygulama tarafından yapılan ve uygulamanın eriştiği çağrı bilgileri
Erişilen hesap bilgileri	Uygulama tarafından erişilen hesap bilgileri

Tablo 3.3. (Devamı)

Kurulan HTTP bağlantıları	Uygulama tarafından kurulan HTTP bağlantıları
Çalışan süreçlerin sorgulanması	Uygulama tarafından varlığı sorgulanan süreçler

Çıkarılan bilgiler rapor içerisinde droidmon başlığı altında toplanarak rapora eklenmektedir.

Dinamik analizi yapılan uygulamanın ağ trafiği tcpdump aracılığıyla toplanmaktadır. Toplanan trafik verileri pcap formatında dışarı aktarılarak analiz bitiminde rapor eki olarak sunulur. Ayrıca toplanan ağ trafiğinden bilgiler rapor içerisinde network başlığı altında gösterilir. Bu başlık altında DNS sorgulamaları, bağlantı kurulan IRC, IP ve domain bilgileri yer almaktadır.

3.3. Raporlama

Statik analiz ve dinamik analiz sırasında toplanan bilgiler raporlama modülü içerisinde bütünleşik bir yapı haline getirilmektedir. Analist, bir şüpheli yazılımı analiz için kum havuzuna gönderdikten sonra şüpheli yazılımın geçirdiği tüm statik ve dinamik analiz süreçlerinin çıktıları raporda yer almaktadır. Raporda yer alan tüm alanlar Tablo 3.4.'de görülmektedir.

Tablo 3.4. Raporda yer alan bilgi alanları

Info	Analiz hakkında genel bilgiler
Droidmon	Dinamik analiz sırasında elde edilen bilgiler
Signatures	Analiz sonucunda elde edilen bilgilerin davranış veritabanında aratılması sonucunda elde edilen bilgiler
Target	Analiz edilen dosyanın kimlik bilgleri
Network	Analiz edilen dosyanın bağlantı bilgileri
Apkinfo	Statik analiz sırasında elde edilen bilgiler
Screenshots	Analiz sırasında elde edilen ekran görüntüleri
Strings	Analiz edilen dosyanın string bilgileri
Metadata	Analiz edilen dosyadan elde edilen pcap gibi alt bilgi dosyalarının bilgileri

Tablo 3.4.'de yer alan alanların içeriği ve üretilen raporun tamamı Şekil 3.2., Şekil 3.3. ve Şekil 3.4.'de gösterilmiştir.


```
{
    "info": {
        "added": 1545168817.422165,
        "started": 1545168817.685639,
        "duration": 117,
        "ended": 1545168935.113018,
        "owner": null,
        "score": 1.0,
        "id": 65,
        "category": "file",
        "git": {
        "monitor": "e19c4b4b529be2e90b3c5a3dfaad96f71c4fd54b",
        "package": "apk",
        "route": "internet",
        "custom": null,
        "machine": {
        "platform": null,
        "version": "2.0.5",
        "options": "procmemdump=yes,route=internet"
    },
    "droidmon": {
        "DexFile": [
        "killed_process": [],
        "crypto data": [
        "TelephonyManager_listen": [],
        "SystemProperties": [
        "reflection calls": [
        "mac_data": [],
        "findLibrary": [
        "accounts": [],
        "SharedPreferences": [],
        "registerContentObserver": [],
        "started_activities": [],
        "registered receivers": [
        "ComponentEnabledSetting": [],
        "sms": [],
        "decoded_base64": [],
        "loadDex": [
        "httpConnections": [
            {
            {
        ],
        "commands_output": [],
        "encoded_base64": [],
        "ContentResolver queries": [
        "events": [],
        "loadClass": [],
        "setMobileDataEnabled": [],
        "DexClassLoader": [],
        "findResource": [],
        "data leak": [
        "fingerprint": [
        "commands": [],
        "file accessed": [
        "crypto_keys": [
            {
        ],
```


Şekil 3.2. Örnek rapor çıktısı

```
        ],
        "PathClassLoader": [],
        "ContentValues": [],
        "handleReceiver": []
    },
    "signatures": [
        {
            "markcount": 1,
            "families": [],
            "description": "Connects to an IP address that is no longer responding to requests
            (legitimate services will remain up-and-running usually)",
            "severity": 5,
            "marks": [
                {
                    "category": "dead_host",
                    "ioc": "123.56.206.59:7878",
                    "type": "ioc",
                    "description": null
                }
            ],
            "references": [],
            "name": "dead_host"
        }
    ],
    "target": {
        "category": "file",
        "file": {
            "yara": [],
            "sha1": "9337ccd945cb320eb26b731edf1321f8d939ee85",
            "name": "E5E22B357893BC15A50DC35B702DD5FCDFEAFC6FFEC7DAA0D313C724D72EC854.apk",
            "type": "Java archive data (JAR)",
            "sha256": "e5e22b357893bc15a50dc35b702dd5fcdfeafc6ffec7daa0d313c724d72ec854",
            "urls": [],
            "crc32": "347C99E3",
            "path":
            "/root/.cuckoo/storage/binaries/e5e22b357893bc15a50dc35b702dd5fcdfeafc6ffec7daa0d313c724d
            72ec854",
            "ssdeep": null,
            "size": 1565163,
            "sha512":
            "756572eaedaaec5316f4696faf10b8fd140c8239ab523d832e653ed3d1e6c05abb96c717ec3074a149857330
            f9762028a5a9b01dbc3392e5989e9f95a72ab436",
            "md5": "40507254b8156de817f02c0ed111e99f"
        }
    },
    "network": {
        "tls": [],
        "udp": [
        "dns_servers": [
        "http": [],
        "icmp": [],
        "smtp": [],
        "tcp": [],
        "smtp_ex": [],
        "mitm": [],
        "hosts": [
            "123.56.206.59",
```


Şekil 3.3. Örnek rapor çıktısı

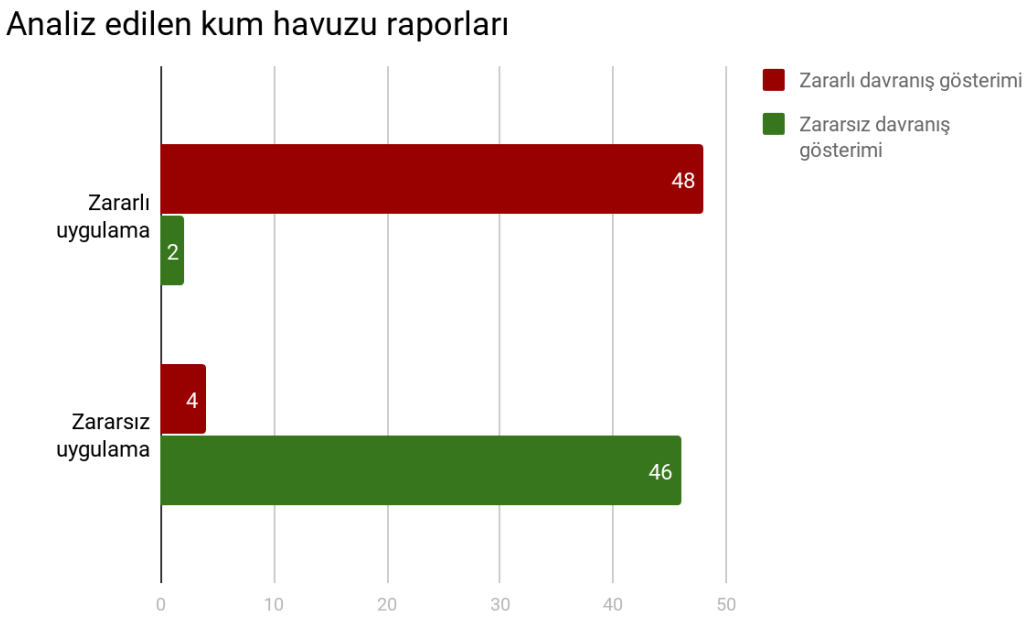
```
        "pcap_sha256": "224739eaaf181625711689aae6e50fcd7bdb3230ca9cbe65fbbb91425846e8f8",
        "dns": [
        "http_ex": [],
        "domains": [
        "dead_hosts": [
            [
                "123.56.206.59",
                7878
            ]
        ],
        "sorted_pcap_sha256": "86fb1af3ed54181439aa0c5a0e27eac8f5b4b3d3e57c229c17a3907dd101b39c",
        "irc": [],
        "https_ex": []
    },
    "apkinfo": {
        "files": [
        "static_method_calls": {
            "is_dynamic_code": true,
            "is_reflection_code": true,
            "is_native_code": true,
            "all_methods": [
        },
        "manifest": {
            "activities": [
                "com.sd.clip.activity.SDManagerActivity",
                "com.sd.clip.activity.FileManagerActivity",
                "com.sd.clip.activity.FileDeleteActivity",
                ""
            ],
            "main_activity": "com.sd.clip.activity.FileManagerActivity",
            "receivers": [
                "com.hg.mer.Nws",
                ""
            ],
            "providers": [],
            "package": "com.web.sdfile",
            "libraries": [],
            "services": [
                "com.sd.clip.urr.UploadErrorInfoService",
                "com.hg.mer.PG"
            ]
        }
    },
    "screenshots": [
    "strings": [
    ],
    "metadata": {
        "output": {
            "pcap": {
                "basename": "dump.pcap",
                "sha256": "224739eaaf181625711689aae6e50fcd7bdb3230ca9cbe65fbbb91425846e8f8",
                "dirname": ""
            }
        }
    }
}
```


Şekil 3.4. Örnek rapor çıktısı

BÖLÜM 4. TEST SONUÇLARI VE DEĞERLENDİRME

4.1. Değerlendirme

Kum havuzunun hibrit analiz yeteneklerinin değerlendirilmesi için 50 zararlı ve 50 zararsız uygulama seçilmiştir. 50 zararsız uygulamanın hepsi Google Uygulama Mağazasından alınmıştır. Zararlı yazılımlar ise VirusTotal ve Koodous platformlarından antivirüsler tarafından zararlı olarak nitelendirilen örnekler içerisinden rastgele alınmıştır. Analizler sırasında herhangi bir davranış göstermeyen uygulamalar test örnekleri arasından çıkarılmıştır ve yerine yeni örnekler seçilmiştir. Örnekler rastgele alınarak zararlı yazılım çeşitliliği mümkün olduğunca üst seviyede tutulmaya çalışılmıştır. Çalışma sonucunda elde edilen sonuçlar Şekil 4.1.'deki gibidir.



Şekil 4.1. Test sonuçları

Test örnekleri zararlı yazılım analistlerine inceletilerek sonuçlar üretilmiştir. Yapılan çalışmalar sonucunda kum havuzunun hibrit analiz sonucu üretilen raporun analistin bir uygulamanın zararlı olup olmadığına karar vermesi noktasında yeterli olduğu görülmüştür. Bazı zararlı uygulamaların sahip olduğu gelişmiş anti-* teknikleri, bu zararlı uygulamaların kum havuzu analizini atlatmıştır. Aynı zamanda, bazı gerçek uygulamaların sahip olduğu zararlı benzeri davranışlar ilgili uygulamanın zararlı olarak nitelendirilmesine sebebiyet vermiştir. Yapılan testler sonucunda elde edilen veriler içerisindeki yalancı pozitif (false-positive) ve yalancı negatif (false negative) değerleri Şekil 4.2.'de görülmektedir.

Şekil 4.2. Yalancı pozitif ve yalancı negatif değerleri

4.2. Tartışma

Ayrıca, ilgili sistem tasarlanmadan önce yapılan akademik taramalarda tasarlanan kum havuzuna benzer çalışmalar incelenmiştir. Gilbert ve ark. [13] zararlı aktivitelerin tespiti konusunda yaptığı çalışmada bağımlılık grafiklerine yer vermiştir. DroidScope [4] VMI üzerinden dinamik analiz yapan bir sistem önermiştir. Java objelerinin analiz için tekrar oluşturulma işlemi gibi hassas işlemler barındırdığından Google'ın her güncellemesinde DroidScope içerisinde büyük adaptasyonlar

yapılması gerektiğinden kullanılması makul değildir. CrowdDroid davranış tabanlı dinamik analiz sistemi önermiştir fakat android işletim sisteminin özelliklerini yeteri kadar kapsayamamıştır [8]. 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, bilgiler ışığında hibrit analiz yetenekleri olan bir kum havuzu elde etme amacıyla geliştirilmiştir.

BÖLÜM 5. SONUÇLAR

Hibrit android kum havuzunun amacı kapsamlı bir statik ve dinamik analiz yaparak çıktıların raporlanması ile zararlı yazılım analistlerinin hızlı bir şekilde Android uygulama örnekleri içerisinden hangilerinin zararlı olabileceğinin tespitini yapabilmesidir.

Otomatize bir analiz ortamının en büyük problemi sanal cihaz tabanlı olmasından kaynaklı olarak evasion yöntemleridir. Zararlı uygulamaların kullandığı anti-VM ve anti-sandbox teknikleri her geçen gün gelişmekte ve değişmektedir. Kum havuzu bünyesinde alınan anti-evasion önlemleri her zararlı yazılımı kandırmakta yetersiz kalabilir. Yapılan çalışmalar x86 zararlı yazılımlarda evasion tekniklerinin yaygın olarak kullanılmadığını gösterse dahi [12, 13] Android tabanlı kum havuzları için bu varsayımın doğru olduğunu kanıtlayan bir çalışma bulunmamaktadır.

Google'ın uygulama mağazası için tanıttığı Bounce teknolojisi göz önünde bulundurulduğunda Android zararlı uygulama geliştiricilerinin analiz ortamları tespiti üzerinde çaba harcayacakları tahmin edilebilir. Yapılan testlerde de Şekil 4.2.'de görüldüğü üzere kum havuzu %96 oranında başarıma, %8 yalancı pozitif ve %4 yalancı negatif oranına sahiptir. Başarım oranının %100 olmama sebebi yukarıda bahsedilen sürekli gelişen zararlı uygulamalar ve kullandıkları teknikler olarak değerlendirilmektedir. Yine de, kum havuzunun incelenecek uygulamanın fazla olduğu ortamlarda analistlere sağladığı rapor, analistlerin hangi uygulamanın zararlı olup olmadığına 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 olmadığını tespit etme konusunda yol gösterdiğini göstermektedir.

KAYNAKLAR

[1] T. Blasing, L. Batyuk, A.-D. Schmidt, S. Camtepe, and S. Albayrak, "An Android Application Sandbox System for Suspicious Software Detection," inProceedings of the 5th International

[2] A. 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.

[3] M. Spreitzenbarth, F. Freiling, F. Echtler, T. Schreck, and J. Hoffmann, "Mobile-sandbox: Having a Deeper Look into Android Applications," in Proceedings of the 28th Annual ACM Symposium on Applied Computing (SAC), 2013.

[4] L. K. Yan and H. Yin, "Droidscope: Seamlessly reconstructing the os and dalvik semantic views for dynamic android malware analysis," in Proceedings of the 21st USENIX Security Symposium, 2012.

[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, 2014.

[6] V. Svajcer, "Sophos Mobile Security Threat Report," http://www.sophos.com/en-us/medialibrary/PDFs/other/sophos-mobile-security-threat-report.ashx, 2014.

[7] H. Lockheimer, "Android and Security," http://googlemobile.blogspot.com/2012/02/android-and-security.html, 2012.

[8] I. Burguera, U. Zurutuza, and S. Nadjm-Tehrani, "Crowdroid: Behavior-Based Malware Detection System for Android," in Proceedings of the 1st ACM Workshop on Security and Privacy in Smartphones and Mobile Devices (SPSM), 2011.

[9] M. C. Grace, W. Zhou, X. Jiang, and A.-R. Sadeghi, "Unsafe Exposure Analysis of Mobile In-App Advertisements," in Proceedings of the 5th ACM Conference on Security and Privacy in Wireless and Mobile Networks (WISEC), 2012.

[10] U. Bayer, C. Kruegel, and E. Kirda, “TTAnalyze: A Tool for Analyzing Malware,” in Proceedings of the 15th European Institute for Computer Antivirus Research (EICAR) Annual Conference, 2006.

[11] X. Chen, J. Andersen, Z. M. Mao, M. Bailey, and J. Nazario, “Towards an Understanding of Anti-Virtualization and Anti-Debugging Behavior in Modern Malware,” in Proceedings of the 38th Annual IEEE International Conference on Dependable Systems and Networks (DSN), 2008.

[12] M. Lindorfer, C. Kolbitsch, and P. Milani Comparetti, “Detecting Environment-Sensitive Malware,” in Proceedings of the 14th International Symposium on Recent Advances in Intrusion Detection (RAID), 2011.

[13] P. Gilbert, B.-G. Chun, L. P. Cox, and J. Jung, “Vision: Automated Security Validation of Mobile Apps at App Markets,” in Proceedings of the 2nd International Workshop on Mobile Cloud Computing and Services (MCS), 2011.

[14] Kapratwar, Ankita, "Static and Dynamic Analysis for Android Malware Detection" (2016). Master's Projects. 488.

[15] 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 & Distributed System Security Symposium (NDSS), 2012.

[16] CheckPoint, Charger malware calls and raises the risk on google play 2017.

[17] Lookout, Pegasus for android, https://info.lookout.com/rs/051-ESQ-475/images/lookout-pegasus-android-technical-analysis.pdf., Erişim Tarihi: 30.04.2019.

[18] Google, An investigation of chrysaor malware on android 2017.

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

[20] W. Zhou, Y. Zhou, X. Jiang, and P. Ning, “Detecting Repackaged Smartphone Applications in Third-Party Android Marketplaces,” in Proceedings of the 2nd ACM Conference on Data and Application Security and Privacy (CODASPY), 2012.

[21] A. Bibat, “GGTracker Malware Hides As Android Market,” http://www.androidauthority.com/ggtracker-malware-hides-as-android-market-17281/, 2011. Erişim Tarihi: 30.04.2019.

[22] https://koodous.com., Erişim Tarihi: 30.04.2019.

[23] https://virustotal.com., Erişim Tarihi: 30.04.2019.

[24] https://github.com/androguard/androguard., Erişim Tarihi: 30.04.2019.

[25] W. Enck, P. Gilbert, B.-G. Chunn, L. P. Cox, J. Jung, P. McDaniel, and A. N. Sheth, "TaintDroid: An Information-Flow Tracking System for Realtime Privacy Monitoring on Smartphones," in Proceedings of the 9th USENIX Symposium on Operating Systems Design and Implementation (OSDI), 2010.

[26] F. Maggi, A. Valdi, and S. Zanero, "AndroTotal: A Flexible, Scalable Toolbox and Service for Testing Mobile Malware Detectors," in Proceedings of the 3rd ACM Workshop on Security and Privacy in Smartphones and Mobile Devices (SPSM), 2013.

ÖZGEÇMİŞ

Mert Can COŞKUNER, 28.11.1993'de Samsun'da doğdu. İlk, orta ve lise eğitimini Samsun'da tamamladı. 2011 yılında Ondokuz Mayıs Lisesi'nden mezun oldu. 2011 yılında başladığı İzmir Ekonomi Üniversitesi Yazılım Mühendisliği Bölümü'nü 2016 yılında bitirdi. 2017 yılında Sakarya Üniversitesi Bilgisayar Mühendisliği Bölümü'nde yüksek lisans eğitimine başladı. 2017 yılında Garnizon Bilgi Güvenliği'nde sızma testi uzmanı olarak çalışmaya başladı akabinde 2018 yılında Savunma Teknolojileri ve Mühendislik A.Ş.'de siber güvenlik uzmanı olarak çalışmaya başladı. Halen Savunma Teknolojileri ve Mühendislik A.Ş.'de siber güvenlik uzmanı olarak görev yapmaktadır.