Android Zararlı Yazılımlarını Tespit Etme, İmza Oluşturma ve Sınıflandırma

Ömer Faruk Acar[1]

***Özet*— Mobil cihazların hızlı gelişimi ve yaygınlaşması ile birlikte bu cihazların ve üzerinde çalışan uygulamaların güvenliği bir sorun haline gelmiştir. Mobil işletim sistemlerinden özellikle Android işletim sisteminde çok yüksek oranda güvenlik tehditleri tespit edilmiştir [1][2].Doğası gereği mobil ortamlardaki zararlı yazılımların tespit edilmesi için kullanılacak yöntemlerde bugüne kadar geliştirilen yöntemler maalesef başarısız olmaktadır. Bu bildiri kapsamında Android işletim sistemi için geliştirilmiş zararlı yazılım tespit etme altyapısı, imza algoritması, diğer zararlı yazılımlarla ilişkilendirme yöntemlerinden ve test çalışmalarından bahsedilmektedir.**

***Anahtar Sözcükler*— Android, Zararlı Yazılım, İmza Algoritmaları**

Abstract—Security concern of applications for mobile platforms become one of the hot topics with the rapid evolution and wide usage of mobile devices. A notable amount of security threats have been identified on the mobile operating systems, more specifically on Android operating system, recently. [1][2] Developed methods that was used for the detection of malwares for other platforms tend to fail in the mobile environment, because of the nature and structure of malicious software. To mitigate this problem, in this paper, a new malware detection infrastructure, developed for the Android operating system, signature algorithms, correlation with other malware families and evaluation of proposed system are discussed.

***Index Terms*— Android, Malware, Signature Algorithms**

I. GİRİŞ

Mobil platformlardaki tehlikeler diğer platformlarda olduğu gibi her geçen gün artmakta ve geliştirilen zararlı yazılımlarda gittikçe karmaşıklaşmaktadır. Yeni nesil Zararlı yazılımlar, statik ve dinamik analize olanak vermemek için farklı teknikler kullanmakta böylece tespit edilmesi ve analiz edilmesi zorlaşmaktadır.

Çoğunlukla temiz bir uygulamaya eklenerek paketlenen ve geçerli uygulama marketi dışındaki marketlerden yayılan bu zararlı yazılımlar kendilerini karmaşıklaştırma (obfuscation) ve şifreleme(encryption) gibi yöntemleri kullanarak çok iyi gizleyebilmektedir. Çalışma öncesinde yapılan literatür taramalarında mevcut güvenlik çözümlerinin çalışma prensipleri araştırılmıştır. Mevcut mobil güvenlik çözümlerinde mobil cihazlardaki pil, işlem gücü, bellek gibi kısıtlardan dolayı sadece imza tabanlı kontrollerin gerçekleştirildiği tespit edilmiştir.

Mevcut uygulama marketine bakıldığında zararlı yazılımları(malware) tespit etmek için kullanılan birçok güvenlik uygulaması olduğu görülür fakat yapılan çalışmalarda bu uygulamaların başarı oranlarının düşük olduğu ve hatalı tespit (false positive) oranlarının yüksek olduğu gözlemlenmiştir[3].

Bilgisayarlarda olduğu gibi Android dünyasında da bir zararlı yazılımı tespit etmek için güvenlik uygulamaları zararlı yazılımdan çıkardıkları imzaları (signature) karşılaştırmak suretiyle tarama gerçekleştirmektedir. Bu noktada iki problem ortaya çıkmaktadır.

Birincisi, bir zararlı yazılım imzası oluşturulduktan sonra eğer zararlı yazılıma, kod karmaşıklaştırma (obfuscation), tekrar paketleme (repackage), dosya/paket adlarını değiştirme(Rename), kod akışını değiştirme(reorder code), araya anlamsız kodlar yerleştirme(insert junk code), karakter katarlarını şifreleme(encrypt strings) gibi işlemler uygulandıktan sonra güvenlik uygulaması tarafından bu zararlı yazılım tespit edebilir mi?

İkincisi, sadece imza karşılaştırması yaparak dinamik analiz gerçekleştirmeden mevcut ve henüz tespit edilememiş(zero day) zararlı yazılımlar yakalanabilir mi?

Belirtilen bu sorunlara çözüm olarak yapılan araştırmalarda statik analiz ve dinamik analiz yöntemleri önerildiği [3][4] fakat bu yaklaşımların ve çözümlerin tüm durumlar için toptan bir çözüm sunmadığı görülmüştür. Mevcut çoğu çözüm normal şartlar altında başarılı olmakta fakat yeni nesil zararlı yazılımların atlatma teknikleri kullanmasıyla başarısız olmaktadır. Bazı çalışmalarda doğrudan zararlı yazılım tespit etmek yerine davranışlar ve seçilen özelliklere göre belirli skorlamalara gidildiği ve sonuçta bu skora göre sınıflandırmalar yapıldığı belirtilmektedir.[10] Bu bildiride anlatılan çözümde tüm yaklaşımlar katmanlı bir yapıda belirli bir sırayla yapılarak etkin ve başarılı bir çözüm elde edilmeye çalışılmıştır.

Yapılan çalışma kapsamında bu soruların cevabını verebilecek en iyi analiz ortamı oluşturulması ve bahsi geçen yöntemlere dayanıklı bir kimliklendirme için imza algoritması çalışmaları gerçekleştirilmiştir. Bildiri kapsamında bu çalışmanın ayrıntıları ve sonuçları değerlendirilecektir.

[1] Ömer Faruk ACAR, HAVELSAN A.Ş., Ankara; e-posta: oacar@bgt.havelsan.com.tr

II. ZARARLI YAZILIM TESPİT YÖNTEMLERİ

Zararlı yazılım tespiti konusunda çok fazla sayıda çalışma bulmak mümkündür. Farklı yaklaşımlar denenerek en az hata ile bilinen ve bilinmeyen zararlı yazılımlar tespit edilmeye çalışılmaktadır. Bunun için Statik ve Dinamik analiz yöntemleri kullanılmaktadır.

Çalışma sırasında güvenlik çözümlerinin başarı oranını ölçmek maksadıyla mevcut zararlı yazılımlardan seçilmiş örnekler ve geliştirilen zararlı yazılım Virustotal[5] ortamı üzerinde bulunan antivirüs uygulamalarına tarattırılmış ve tespit oranları Tablo 1'de gösterilmiştir. Zararlı yazılım seçiminde zararlı yazılım türü (adware, RAT, Spyware v.s.) ve farklı zararlı kabiliyetler (SMS gönderme, uzaktan komut satırına erişim, reklam gösterme v.s.) açısından çeşitlilik sağlayacak örnekler tercih edilmiştir.

TABLO 1
ZARARLI YAZILIM YAKALANMA ORANLARI

	Antivirüs Başarı Oranı		
Malware İsmi	Tespit Edebilen	Tespit Edemeyen	Tespit Edilme Oranı %
Chuli	19	28	40,43
CI4	30	17	63,83
Fakemart	22	25	46,81
Plankton	8	39	17,02
İkno	19	28	40,43
Lena	30	17	63,83
PuppyWidget	10	37	21,28
RootSmart	30	17	63,83
Leadbolt	10	37	21,28
SSUCL	25	22	53,19
Stels	25	22	53,19
SystemSecurity	31	16	65,96
Örnek Zararlı Yazılım	0	47	0
		Ortalama :	42,39

Yapılan testlerde tek başına hiçbir metodun tam olarak başarılı olduğunu söylenemez. Fakat ortaya birden fazla yöntemden oluşan karma model oluşturulduğunda (çalışmada bu model, katmanlı analiz olarak adlandırıldı) her katmanda yapılan analiz diğer katmanlardaki üretilen sonuçlarla değerlendirildiğinden hem yanlış tespit(false positive) oranı düşmekte hem de toplamda bir örnek kümesi içindeki zararlı yazılım veya zararlı yazılım olmasa bile bazı zararlı davranışlar gerçekleştiren uygulamaların tespit oranı çok yüksek olmaktadır.

A. İmza Tabanlı

Statik analiz yöntemlerinden biri olan imza çıkarma yöntemi mobil güvenlik alanında ilk kullanılmaya başlayan tespit yöntemidir. Bu yöntem, zararlı yazılıma ait hangi özelliklerin imza çıkarmada kullanılacağına ve bu imzayı oluşturmada nasıl bir algoritma kullanılacağına göre çeşitlenmektedir. Genellikle kod içerisinden belirgin bazı kısımlar alınarak bunlardan zararlı yazılıma özgü bir imza çıkartılmaktadır. [7][8] Android uygulamaları için örnek verilecek olursa, kullandığı izinler(permissions), kod içerisinde geçen bilgiler (String values), kullanılan sistem çağrıları (system call) örnek olarak verilebilir.

Bu verilerden çıkartılan bir veri kümesi (data set) oluşturulduktan sonra seçilen algoritmaya göre hesaplatılarak uygulamayı tanımlayacak bir değer elde edilmektedir. Genellikle imza çıkarma işlemi için özet (hash) algoritmaları kullanılmaktadır[18] fakat bu algoritmalar uygulamadaki en ufak bir değişiklikte çok farklı sonuçlar çıkardığı için imza çıkarma işleminde başarılı olamamıştır. Bu problemden dolayı bulanık özet(fuzzy hash) algoritmaları kullanılmaya başlanmış [9] ve iki imza değeri arasında ilişki kurulabilmiştir. Fakat bu da yeterli olmamıştır. Bir imza parçasından diğer imzaları hesaplayabilme, küçük boyutlu veriler içinde ilgileşim (correlation) hesaplamaları yapma ihtiyacı doğduğu için farklı algoritmalar üretilmelidir. Bu çalışma kapsamında kullanılan karakter katarları üzerinde arama yapmayı sağlayan algoritmalar ve bu algoritma içerisinde ötelemeli özet(Rolling hash) algoritmaları çözüm olarak sunulmuştur. İmza algoritmalarında en önemli kural her bir örnek için tekilliğin(unique) sağlanmış olmasıdır. Aksi takdirde algoritma başarısız demektir.

B. İlişki (Korelasyon) Hesaplama

Yapı olarak imza tabanlı yönteme benzese de imza tabanlı yöntemdeki gibi birebir eşleştirerek kontrol etme yerine bütünü daha küçük parçalara ayırıp istatiksel olarak parçaların birbirine benzerliği hesaplanmaktadır. Bu yöntemde bazen makine öğrenmesi, yapay sinir ağları gibi yeni yaklaşımlarda kullanılmaktadır. Burada önemli olan yüzde yüz bir eşleştirme değil eldeki bilinen zararlı yazılım verilerine göre incelenen uygulamanın bu verilerden ne kadarını içerdiği veya hangi kısımların benzer özelliklerden oluştuğunun tespit edilmesidir.

İlişki hesaplama yöntemi kullanılırken en fazla sınıflandırma algoritmaları kullanılmaktadır. Bu algoritmaların Android uygulamalarından çıkartılan özelliklere göre benzerliği hesaplamada ne kadar başarılı olduğu bildiri içerisinde ayrı bir başlık altında toplanmıştır.

C. Davranışsal Kalıplar

Bahsedilen iki yöntemde de ağırlıklı olarak statik bir inceleme yapılırken bu yöntemde ise bir uygulamanın çalışma sırasındaki davranışları hedef alınmaktadır. Tablo 2'de bahsedilen özellikler izlenerek davranışlardan bir tasarım kalıbı oluşturulmaya çalışılmakta ve böylece benzer davranışları gösteren uygulamalar gruplandırılabilmektedir.

TABLO 2 KULLANILAN ÖZELLİKLER

Uygulama İzinleri	-	Permissions
İşlemci Komutları	-	CPU Instructions
İşletim Sistemi Çağrıları	-	System Call
Ağ Aktiviteleri	-	Network Activity(Wi-Fi,3G)
İşlemci Kullanımı	-	CPU Usage
Hafıza Kullanımı	-	Memory Pattern
Pil Tüketimi	-	Battery Usage
Sensor Kullanımı		Sensor Usage

Bu yöntemde karşılan en büyük sorun uygulamanın her zaman aynı davranışı sergilememesidir. Mesela, sadece SMS ile aktif hale gelen bir zararlı yazılım izlendiğinde herhangi bir davranış sergilemeyeceği için tasarım kalıbını oluşturacak yeterli veri elde edilemeyecektir. Diğer yöntemlerden avantajlı olduğu kısımlar ise uygulamada yapılan kod karmaşıklaştırma(obfuscation), şifreleme(encryption) gibi işlemler davranış şekliğini değiştirmediği için analiz başaralı olmaktadır.[6] Ayrıca, çalışma esnasında kullanıcı davranışları simüle edilerek komuta kontrol sunucusu (C&C) ile yapılan haberleşme, sonradan kod indirip çalıştırma (Dynamic code loading and payload execution) gibi işlemler bu yöntemde tespit edilebilmektedir.[15][16][17]

III. İMZA ÇIKARMA

İyi bir imza algoritması geliştirmek zararlı yazılım analizinde en çok zorlanılan aşamalardan birisidir çünkü geliştirilen imza, tüm atlatma tekniklerinden etkilenmemeli, performanslı olmalı ve anlamlı sayılardan(bitlerden) oluşmalı ki sınıf seviyesinde veya metot seviyesinde karşılaştırmalar yapılabilsin. Bildiri kapsamında yapılan çalışma sırasında da mevcut özet algoritmaları (hash) ve karakter katarları(String) üzerinden arama yapabilen, özet çıkartabilen algoritmalar araştırıldı. Çalışmalar sırasında farklı algoritmalar test edildi. Bilişim sistemlerinde imza oluşturmak için kullanılabilecek gelişmiş Elliptic Curve, Elgamal İmza Şeması, NSA tarafından geliştirilmiş Dijital İmza Algoritması gibi algoritmalar zararlı yazılımın oluşturulan farklı varyantları üzerinde farklı imzalar çıkardığı için başarısız olmuştur. Bu durum sonrasında genellikle adli analiz çalışmalarında kullanılan bulanık özet algoritması (fuzzy hash) uyarlanmaya çalışılmıştır. Bu algoritma da, büyük veri setlerinden oluşturulan imzalarda benzerlik oranı vermesi yönüyle başarılıydı fakat küçük boyutlu veri kümelerinde (metot ve sınıf numarası) hesaplama yapmak için uygun değildi. Sonunda test edilen algoritmalar içerisinde Karp-Rabin algoritmasının mevcut durum için uygun olduğu gözlemlendi. Testlerde çok küçük veri setleriyle bile imza oluşturulabiliyor, performanslı çalışıyordu tek sorun farklı sayıdaki ve farklı uzunluktaki değerleri karşılaştırmak gerekiyordu. Bunun içinde bir özet algoritması olarak "Rolling Hash" algoritmaları incelenerek özelleştirilmiş bir algoritma kullanıldı.

Özetle, Rabin-Karp algoritmasında her bir sistem çağrısının(system call) tablodaki karşılık gelen değerini girdi olarak alıp bunu öncede belirlinmiş bir asal sayı ile metot içerisinde kaçıncı sıradaysa bu sistem çağrısı asal sayınının bu sıra sayısı kadar üssü alınıp çarpılması ve elde edilen değerlerin toplanması ile hesaplanmaktadır. Formülüze edilecek olursa, hesaplanacak özet değeri H olsun. Girdi olarak verilen metot içerisindeki sistem çağrılarının tablodaki karşılık değerleri de $c_1 + \cdots + c_k$ şeklinde bir seri olsun. Önceden belirlenmiş asal sayı da a olsun.

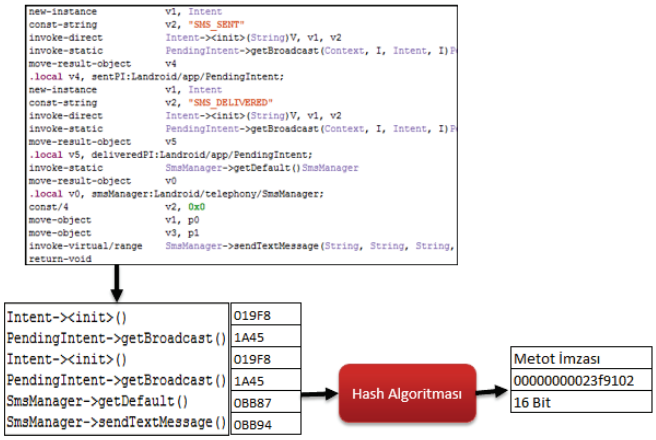
$$H = c_1 a^{k-1} + c_2 a^{k-2} + c_3 a^{k-3} + \cdots + c_k a^0 \quad (1)$$

Bu formül Java dilinde gerçekleştirilmiştir. Burada önemli olan hesaplanan özet değerin önceki ve sonraki hesaplanan değer ile doğrudan ilişkili olması ve yeni bir sistem çağrısı eklendiğinde ya da çıkartıldığında bunun rahatlıkla hesaplanabilmesidir.

```
final static BigInteger primenumber = new BigInteger("11");
public BigInteger hashcalculator(String deger)
{
    int tmplength=deger.length();
    tmpcarpan = BigInteger.ONE;
    smallhash=BigInteger.ZERO;
    for (int j=0; j < tmplength ; ++j) {
        int karakter=deger.charAt(j);
        String unicodevalue=String.valueOf(karakter);
        BigInteger unicodeBigInteger=new BigInteger(unicodevalue);
        smallhash=smallhash.add( \
        unicodeBigInteger.multiply(primenumber.pow(j)));
    }
    return smallhash;
}
```

Şekil 1. Rolling Hash Algoritması

Büyük resime bakıldığında Şekil 2'de görüldüğü üzere bir metot içindeki sistem çağrıları tespit edilip tüm çağrıların belirli bir numara verildiği sistem çağrıları tablosundaki değeri bulunup sırayla özet algoritmasına girdi olarak verilmektedir. Çıkan değer o metoda ait imza değeridir.



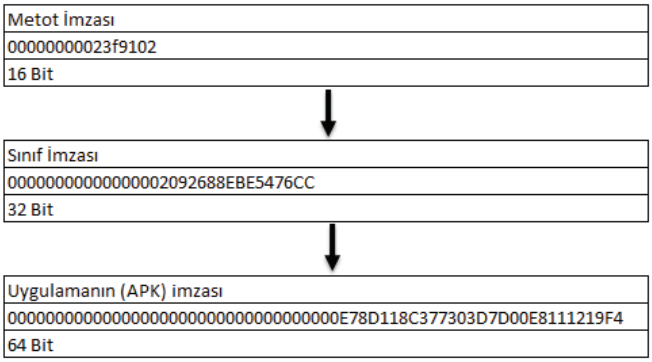
Şekil 2. Metottan imza çıkarma adımları

Yine aynı şekilde tüm metotlardan elde edilen özetler küçükten büyüğe sıralanarak (kod bulanıklaştırma ile sıranın değiştirilmesi yönteminden etkilenmemesi için) özet algoritmasına verilmekte ve sınıf imzası hesaplanmaktadır.Bu sırada yaygın kullanılan kütüphaneler gibi bazı kodlar temizlenerek imzanın asıl uygulamaya ait değerleri içermesi sağlanmaktadır.

IV. SINIFLANDIRMA ALGORİTMALARININ BAŞARISI

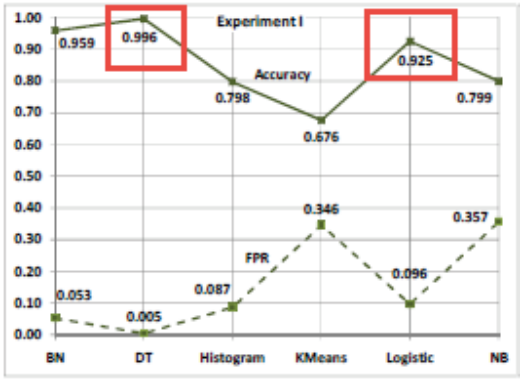
Sınıflandırma algoritmaları, istenilen özellik ya da özelliklere göre birbirine benzeyen veri setlerini gruplamaya yarayan algoritmalardır. Bu çalışma kapsamında sınıflandırma algoritmaları başarılı bir imza işleminden sonra eldeki ham

veriden bilgiye dönüşmüş kümeyi sınıflandırmak için kullanılmıştır. Mevcut zararlı yazılımlardan elde edilmiş imzalar sınıflandırma algoritmaları kullanarak yeni bir uygulama incelendiğinde ne kadar zararlı kod bulunduruyor bunu tespit etmek için kullanılmaktadır [13][14]. Bu amaçla farklı sınıflandırma algoritmalarının performansları değerlendirilerek en az yanlış sonuç üreten algoritma veya birden fazla algoritma seçilerek analiz ortamına dâhil edilmiştir.



Şekil 3. Metot, Sınıf ve Uygulamaya ait imzalar

Bu konuda daha önce yapılmış bir çalışma olan Asaf Shabtai ve Yuvol Elovici'ye ait makalede [6] "Bayesian Networks", "Decision Tree", "Logistic Regression", "Naive Bayes", "Histogram" ve "KMeans" algoritmalarını test etmiş ve en iyi sonuçları Decision Tree ve Logistice Regression algoritmalarından elde etmişlerdir.



Şekil 4. Mobil Uygulama Özelliklerinin Sınıflandırılması

Bu çalışma referans alınarak geliştirilen analiz ortamında "Decision Tree" ve "Logistic Regression" algoritmalarının kullanılması kararlaştırılmıştır.

V. ANDROİD GÜVENLİK VE DENETLEME ARACI-ASAT

Mevcut güvenlik çözümlerinin sadece cihaz üzerinde veritabanlarındaki imzaları doğrudan karşılaştırma yaparak zararlı yazılımları tespit ettiklerinin ve başarı oranlarının düşük olduğunun tespit edilmesi üzerine böyle bir çalışma yapılma ihtiyacı ortaya çıkmıştır.[10] Çalışma kapsamında gerçekleştirilen yapı, sunucu tarafında çalışacak analiz ortamı ve mobil cihaz üzerinde çalışacak güvenlik uygulamasından oluşmaktadır.

A. Katmanlı Analiz Ortamı

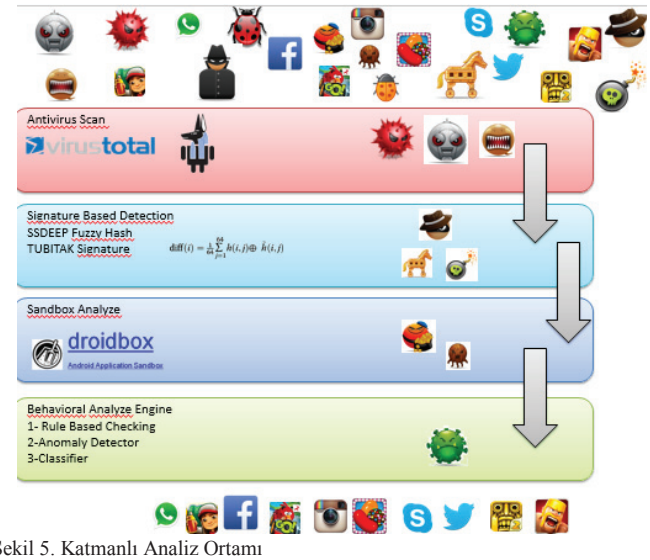
İlk aşamada verilerin toplanabilmesi için tüm marketlerle entegre olup uygulamalara ait "APK" dosyalarının ve uygulama bilgilerinin indirilmesi gerekmektedir.[11][12] Bunun için "Regular Expression" ve "Xpath" sorgularını destekleyen dinamik olarak marketlerin tanımlanabildiği "Scrapy" altyapısını kullanan bir ortam tasarlanmıştır. Ayrıca bu ortam içerisinde dinamik web arayüzleri geliştirilebilecek "Django" altyapısı kullanılarak bir web arayüzü geliştirilmiştir.

İndirilen uygulamalar "Androguard" kütüphanesi kullanılarak apk uzantılı dosyalar açılıp alt bileşenlerine ayrıştırılarak tüm üst veri(meta data) bilgileri elde edilmiş ayrıca uygulamaya ait sertifika, paket ismi, kullanılan izinler, içerdiği sınıflar v.s. veritabanına kaydedilmektedir.

Katmanlı analiz ortamında analiz işlemi dört farklı katmanda gerçekleşmektedir.

1. Antivirüs Taraması
2. İmza Algoritmaları ile Kontrol
3. Dinamik Analiz Ortamı (Sandbox)
4. Davranışsal Kalıplar ile Kontrol

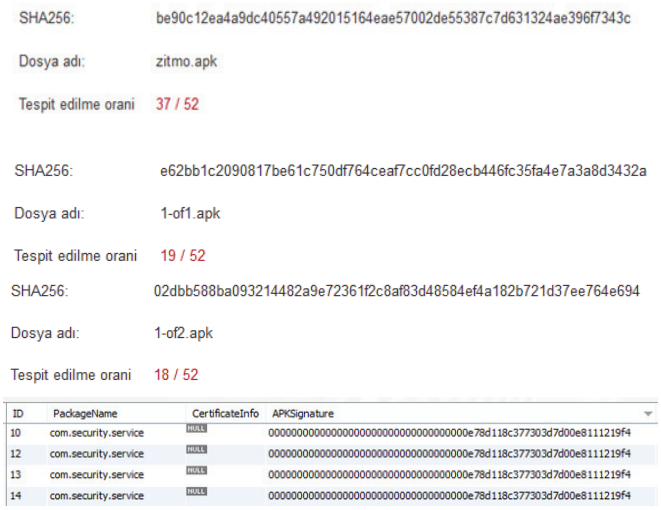
Bu dört katmanlı analiz sırasında bir uygulama sonuç olarak temiz, şüpheli, zararlı yazılım şeklinde bir gruba düşmektedir. Temiz ve zararlı yazılım grubuna giren uygulamalara herhangi bir işlem uygulanmazken şüpheli bulunan uygulamalar manuel analiz için araştırmacıya gönderilmektedir. Manuel analiz sonrasında grubuna karar verilerek analiz ortamının güncellenmesi sağlanmaktadır. Bu sayede sürekli canlı ve kendini güncelleyen bir yarı otomatik analiz ortamı geliştirilmiştir.



Şekil 5. Katmanlı Analiz Ortamı

Katmanlı analiz ortamının diğer çalışmalara göre en başarılı olduğu kısım geliştirilen imza algoritmasıyla yapılan tespit başarı oranıdır. Bunu test etmek için mevcut bir zararlı yazılım "DroidChameleon" [3] isimli uygulama değiştirme altyapısı ile değiştirilmiş ve mevcut antivirüslere tarattırılmıştır. Diğer

antivirüsler değişiklik sonrası bu zararlı yazılımı tespit edemezken geliştirilen imza algoritması aynı imzayı oluşturmaya devam ettiği için tespit edebilmiştir.



Şekil 6. Zararlı yazılımın türevleri ve antivirüs tarama sonuçları

B. *Mobil Güvenlik ve Denetleme Uygulaması*

Mobil cihaz üzerinde çalışacak uygulamanın en temel görevi arka planda güncel olarak elde edilen akılı zararlı yazılım bilgilerinden oluşturulan veritabanı (Threat Intelligence Database) ile cihaz üzerinde yüklü uygulamalarda tarama gerçekleştirmektir. Bunun dışından cihaz üzerinde zafiyete neden olabilecek tüm açıklıkları değiştirmek ve ayarları gerçekleştirmek için sıkılaştırma modülü bulunmaktadır.

Çalışma prensibi olarak diğer güvenlik çözümleri gibi imza kontrolü gerçekleştirse de imza yaklaşımındaki dinamik çözümler başarı oranını arttırmıştır. Bir uygulamanın cihaz üzerinde analizi kaynakları çok hızlı bir şekilde tüketeceği için onun yerine uygulamanın hangi kaynaktan indirildiği, uygulama paket adı (package name), uygulama özet(hash) değeri gibi bilgiler analiz ortamına gönderilerek henüz veritabanında bulunmayan bu uygulama kaynaktan indirilip analizi gerçekleştirilip cihaza sadece sonuç ile ilgili bilgiler dönmektedir. Eğer manuel olarak bir uygulama yükleme işlemi gerçekleştirilmişse kullanıcıdan uygulamayı içeren "APK" dosyasının analiz ortamına gönderilmesi istenmektedir.

VI. SONUÇ VE GELECEK ÇALIŞMALAR

Bu çalışma, çok fazla sayıda kaynaktan veri toplayıp işlenmesi, kendine özgü imza algoritması ve birden fazla yöntemi bir arada destekleyen katmanlı analiz ortamı kullanılması yönüyle zararlı yazılıma karşı yeni bir yaklaşım sunmuştur. Farklı yöntem ve tekniklerin bir arada kullanılması, dinamik bir yapı oluşturularak yeni saldırı durumlarına göre rahatlıkla özelleştirilebilmesi başarı oranının artmasına ve hızlı bir şekilde sonuç elde edilmesine olanak sağlamaktadır.

Mevcut çözümlerin yetersiz olduğu bir durumda geliştirilen analiz ortamı oldukça başarılı sonuçlar ortaya çıkarmıştır fakat hala eksik veya zayıf olan kısımlar bulunmaktadır. Örnek olarak, yeni geliştirilen kod karmaşıklaştırma ortamı [19] "Bangcle", mevcut bir uygulamanın kodlarını JAVA katmanından alıp JNI (Java Native Interface) kullanan C++ diline çeviriyor ve bu şekilde derliyor. Bu durum şu anda analiz ortamında farklı imzalar çıkmasına neden olabiliyor. Aynı arayüzü kullanan zararlı yazılımlar henüz ekosistemde pek rastlanmasa da yakın zamanda yaygınlaşacağı tahmin edilmektedir. Bundan dolayı "Smali" dilinde gerçekleştirilen mantık "Assembly" dili içinde uygulanmalı ve test edilmelidir. Gelecek çalışma olarak, ulusal zararlı yazılım veritabanı oluşturmak ve bu veritabanını bir web sayfası/ web servisi aracılığıyla erişime açılması planlanmaktadır. Diğer bir yapılması gereken çalışma mevcut "Sandbox" dinamik analiz ortamlarının ihtiyaca göre uyarlanması yerine ulusal dinamik analiz ortamının geliştirilmesidir. Şu anda sadece kullanıcı katmanı (User Mode) dikkate alınarak tasarlanmış güvenlik çözümünün daha başarılı analiz kabiliyetleri kazanabilmesi için çekirdek katmanında (Kernel Mode) çalışacak şekilde tekrardan tasarlanması gerekmektedir. Bu katmanda çalışması, uygulamaya daha fazla kaynağı izleyip tıpkı IDS/IPS yazılımları gibi dinamik aykırılık tespit(anomaly detection) yeteneği kazandıracaktır.

Yapılan çalışmalar kapsamında halen Android işletim sisteminin güvenlik açısından yeterli özelliklere sahip olmadığı tespit edilmiş olup dinamik izin modeli, ağ erişim kontrolü gibi diğer mobil işletim sistemlerinde bulunan özelliklerin sonraki sürümlerde ekleneceği tahmin edilmektedir.

KAYNAKLAR

[1] F-Secure Mobile Threat Report Q12014 http://www.f-secure.com/static/doc/labs_global/Research/Mobile_Threat_Report_Q1_2014.pdf.
[2] Sophos Mobile Security Threat Report http://www.sophos.com/en-us/medialibrary/PDFs/other/sophos-mobile-security-threat-report.pdf
[3] Vaibhav Rastogi, Yan Chen, Xuxian Jiang, DroidChameleon: Evaluating Android Anti-malware against Transformation Attacks, ASIA CCS'13,May 8-10 2013, ACM 978-1-4503-1767-2/13/05
[4] G. Suarez-Tangil, J. Tapiador, P. Peris-Lopez and A. Ribagorda, 'Evolution, detection and analysis of malware for smart devices', IEEE, 2013.
[5] https://www.virustotal.com/
[6] A. Shabtai and Y. Elovici, 'Applying behavioral detection on android-based devices', Springer, pp. 235--249, 2010.
[7] J. Kornblum, 'Identifying almost identical files using context triggered piecewise hashing', Digital investigation, vol 3, pp. 91--97, 2006.
[8] S. Schleimer, D. Wilkerson and A. Aiken, 'Winnowing: local algorithms for document fingerprinting', pp. 76--85, 2003.
[9] D. Hurlbut-AccessData, 'Fuzzy Hashing for Digital Forensic Investigators', 2009.
[10] M. Grace, Y. Zhou, Q. Zhang, S. Zou and X. Jiang, 'Riskranker: scalable and accurate zero-day android malware detection', pp. 281--294, 2012.
[11] 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.', 2012.
[12] W. Zhou, Y. Zhou, X. Jiang and P. Ning, 'Detecting repackaged smartphone applications in third-party android marketplaces', pp. 317--326, 2012.
[13] A. Schmidt, R. Bye, H. Schmidt, J. Clausen, O. Kiraz, K. Yuksel, S. Camtepe and S. Albayrak, 'Static analysis of executables for collaborative malware detection on android', pp. 1--5, 2009.

[14] Y. Aafer, W. Du and H. Yin, 'DroidAPIMiner: Mining API-level features for robust malware detection in android', Springer, pp. 86--103, 2013.
[15] T. Petsas, G. Voyatzis, E. Athanasopoulos, M. Polychronakis and S. Ioannidis, 'Rage against the virtual machine: hindering dynamic analysis of Android malware', p. 5, 2014.
[16] T. Blasing, L. Batyuk, A. Schmidt, S. Camtepe and S. Albayrak, 'An android application sandbox system for suspicious software detection', pp. 55--62, 2010.
[17] I. Burguera, U. Zurutuza and S. Nadjm-Tehrani, 'Crowdroid: behavior-based malware detection system for android', pp. 15--26, 2011.
[18] H. Kuzuno and S. Tonami, 'Signature generation for sensitive information leakage in android applications', pp. 112--119, 2013.
[19] http://bangcle.com/

Ömer Faruk Acar, Ege Üniversitesi Bilgisayar Müh. Bölümünden mezun olduktan sonra Hacettepe Üniversitesi Bilişim Enstitüsünde Yüksek Lisansını tamamlamış ve doktora çalışmalarına devam etmektedir. Bilişim (IT) sektörüne web uygulamaları ve KOBİ'lere hazır paket uygulamalar geliştirerek başlamıştır. Sonrasında Türk Telekom A.Ş. bünyesinde Telekom altyapısı üzerine kendini geliştirmiş ve Entegrasyon Uzmanı olarak çalışmıştır. Farklı teknolojiler ve platformlar üzerinde çalışırken hep bir merak ile araştırdığı bu nasıl çalışır? Zafiyet barındırıyor mu? Korumalar nasıl aşılır? Sorularına cevap bulmak için güvenlik alanında uzmanlaşmaya başlamıştır. TÜBİTAK Siber Güvenlik Enstitüsü bünyesinde Veritabanı güvenliği, sızma (penetrasyon) testleri, zararlı yazılım(malware) analizi ve son iki yıldır mobil cihaz güvenliği konularında araştırmalarda bulunmuştur. TR-BOME ekibinin bir üyesi olarak Siber güvenlik olaylarına müdahale etmiştir. Türkiye'deki bankalar, devlet kurumları ve özel sektöre yazılım, danışmanlık, eğitim projeleri gerçekleştirmiştir. Bu çalışmalar sırasında kullanımı yaygınlaşan mobil cihazların nasıl tehlike oluşturduğunu gözlemlemiş ve önlem olarak neler yapılabilir soruları üzerine çalışmalarına devam etmektedir. Şu anda HAVELSAN A.Ş. Siber Güvenlik Takımında siber güvenlik uzmanı olarak görev almaktadır.