Android Mobil Uygulamalar için İzin Karşılaştırma Tabanlı Kötücül Yazılım Tespiti

Recep Sinan ARSLAN, İbrahim Alper DOĞRU, Necaattin BARIŞÇI*

Gazi Üniversitesi Teknoloji Fakültesi Bilgisayar Mühendisliği Bölümü

(Geliş/Received : 21.05.2016 ; Kabul/Accepted : 20.06.2016)

ÖZ

Mobil uygulamalar izin tabanlı modelleri sayesinde kendi güvenlik ve gizlilik modellerini oluştururlar. Uygulamalar, yüklendikleri mobil araçlarda herhangi bir hassas veriye erişmek isterlerse, bu erişim için sadece ihtiyaç duydukları izinleri tanımlamalıdırlar. Ancak bazı uygulamalar, gerek duyacakları izinlerin haricinde fazladan izin talebinde bulunmakta ve bunu daha sonra yapacakları şüpheli kaynak erişimleri için kullanabilmektedirler. Bu çalışmada belirlenen yöntem ile veri setleri kullanılarak daha önceden belirlenen seviyeler doğrultusunda uygulamaların risk değerleri belirlenmektedir. Statik analiz ve kod analizi metotlarını birlikte kullanılmıştır. Kullanılan yaklaşıma göre uygulamaların istedikleri ve kullandıkları izinler belirlenmekte ve fazladan izin talebinde bulunan uygulamalar çıkarılmaktadır. Sonrasında ortaya konulan formül sayesinde her bir uygulama için şüphe değeri belirlenmekte ve bu değere göre uygulamalar kötücül veya zararsız olarak sınıflandırılmaktadır. Ortaya konulan bu yaklaşım, var olan veri setleri üzerinde uygulanarak sonuçları karşılaştırılmış ve doğruluk seviyesi belirlenmiştir. Android işletim sistemi için, geliştirilen bu yeni yöntem sayesinde kötücül yazılımların tespit edilmesi ve kullanıcılar açısından daha güvenli bir Android ortamının oluşturulması amaçlanmıştır.

Anahtar Kelimeler: Android, İzin Tabanlı, Güvenlik, Risk Değerlendirmesi.

Permisson Comparison Based Malware Detection System for Android Mobile Applications

ABSTRACT

Mobile applications create their own security and privacy models through permission based models. Applications, if they require to access any sensitive data in mobile devices that they are downloaded on, in order to do the needed system call for this access, they have to define only required permissions. However, some applications may request extra permissions which they do not need and may use these permissions for suspicious database access they do later. In this study, the aim is to determine those extra requested permissions and to use this on the security and privacy model. According to the study, through the determined methodology, risk values of applications are determined in the light of pre-determined levels within datasets. It is an approach that uses static analysis and code analysis together. According to this approach, the permissions that the applications request and use are determined separately and the applications that request extra permissions are discovered. Then, via the produced formula, suspicion value of every application is determined and applications are classified as malicious or benignant according to this value. This approach was applied on existing datasets; the results were compared and accuracy level was determined.For Android operating system, it is aimed to determine the malicious applications via this newly developed method and to create a safer Android atmosphere for users.

Keywords: Android, Permission Based, Security, Risk Assesment.

1. GİRİŞ (INTRODUCTION)

Modern iletişim teknolojilerinin gelişimi ile birlikte, akıllı telefon, tablet bilgisayar, multimedya merkezleri, ev uygulamaları gibi insanların hayatlarının içine girmiş akıllı ve kişiselleştirilmiş servisler sunan mobil araç/platformlarının ortaya çıkması hızlanmıştır [1]. Günümüzde, artık çok rahatlıkla mobil cihaz kullanımının kablolu bağlantı türlerini geçtiği söylenebilir. Mobil araç kullanımının yaygınlaşması, kullanıcıların kişisel/gizli verilerini, banka hesap numaraları gibi kritik bilgilerini mobil araçları üzerinde saklamalarındaki artışı beraberinde getirmiştir. [2][3]. Bunun gibi nedenlerle, mobil araç geliştiricileri, mobil araç işletim sistemi geliştiricileri, mobil uygulama market sahipleri, uygulama geliştiricileri ve devletler kullanıcıların mobil araçlarda karşılaşabilecekleri kötü durumları önlemek için bu alandaki açık kapıları kapatma ve önlem alma çalışmalarını yürütmektedirler [4].

Mobithinking'in 2013 yılında yayınlamış olduğu verilere göre mobil internet kullanımı kablosuz internet kullanımını geçmiştir. Bunun nedenini sadece akıllı telefonların ve altyapıların gelişimi olarak görmek eksik olacaktır. Bu noktada, akıllı telefonlar için yeni uygulamalar sunan Google Play, Apple Store gibi uygulama marketlerin varlığı göz ardı edilemez. Bu uygulama marketleri sayesinde, kullanıcılar talep ettikleri işlevleri/fonksiyonları

**Sorumlu Yazar (Corresponding Author)*
e-posta: nbarisci@gazi.edu.tr
Digital Object Identifier (DOI) : 10.2339/2017.20.1 175-189

yerine getirebilecek uygulamaları çoğu hallerde ücretsiz olarak bu uygulama marketlerinden indirebilmekte ve telefonlarında kullanabilmektedirler. Bu da kullanıcıların mobil araçları daha fazla kullanma eğilimi göstermelerine neden olmaktadır [5].

Mobil telefonlarda Nokia Symbian, Windows Phone, IOS, Blackberry OS ve Android işletim sistemi en yaygın kullanılan mobil işletim sistemleri arasındadır. Bu işletim sistemleri arasında, Android işletim sistemi açık kaynak kodlu olarak sunulan ve Linux çekirdek yapısını kullanan bir işletim sistemidir. Bu işletim sisteminin kullanılması için herhangi bir ücret talep edilmez. Sadece kullanıcılar geliştirdikleri uygulamaları satmak için markete koymak istediklerinde, geliştirici türünde bir market hesabı açmaları gereklidir. Android işletim sistemi diğer işletim sistemleri arasında açık kaynak kodlu ve ücretsiz erişime sahip olması nedeniyle en yüksek market payına ve satış miktarına sahiptir ve bu durum hızlı bir şekilde yükseliş göstermektedir [6].

Mobil araçların kullanımının yaygınlaşması, kişilerin artık daha fazla kişisel ve kritik verilerini mobil araçlarında saklıyor olmaları ve Android işletim sisteminin de temelde ücretsiz ve açık kaynak kodlu olması sebebiyle en geniş kullanım ağına sahip olması, Android tabanlı mobil araçları kötücül uygulama geliştiricileri için bir hedef haline getirmektedir. Kullanıcıların Google Play benzeri uygulama dağıtım ortamlarından istenilen her türde programın herhangi bir ekstra çalışmaya gerek duymadan, indirilip kullanabilmeleri mümkündür. Her ne kadar bunun gibi büyük ölçekli uygulama havuzları, uygulamayı kullanıcılara sunmadan bir ön güvenlik kontrolünden geçirmekte iseler de, bu platformlarda dağıtım yapılan uygulamalar için tam bir güvenlikten söz etmek mümkün değildir. "Google Bouncer" benzeri ön güvenlik kontrol sistemleri mümkün olduğunca uygulama marketi güvenliğinin sağlanması rolünü üstlenmektedirler. Ancak tam bir güvenliğin sağlanamadığını ortamlarda kullanıcılar açısından daha dikkatli olunması noktasında hassas bir duruş sergilenmesi gerekliliği açıktır [5].

Uygulama marketlerinde dağıtımı yapılan ve kullanıcılara sunulan uygulamalar incelendiğinde, birçok uygulamanın çalışması için gerek duyduğu izin sayısından daha fazla izin talebinde bulunduğu görülmüştür. Uygulamanın ihtiyaç duyduğundan daha fazla izin talebinde bulunması kullanıcılara zarar verebilecek bir nitelikte olabileceğini düşündürmektedir. Bu doğrultuda, uygulamalar ihtiyaç duyulabilecek ve özel olarak belirlenmiş bazı gereksiz izinleri de talep ederek, şüpheli bir hale gelmektedirler. Bu şekilde markette sunulan ve ihtiyaç duymayacağı izinleri de talep eden uygulamalar kullanıcılar tarafından indirilerek mobil araçlarına yüklenebilmektedir. Bu noktada bu tarz uygulamaların ayrıca takip edilmesi gerekliliği ortaya çıkmaktadır. Bunun yanında kullanıcılar mobil araçlarına uygulama marketlerinin haricinde de uygulamalar yükleyerek kullanabilmektedirler(3. Parti servisler). Bu nedenle, uygulama havuzlarında izin kullanımına yönelik kısmen sağlanabilecek güvenlik mekanizmalarının kolaylıkla atlatılabileceğini, farklı servisler yardımıyla kullanıcılara sorunlar çıkartılabileceğini göstermektedir [1].

Uygulamaların ihtiyaç duyduklarından daha fazla izin talebinde bulunmaları halinde tespit edilmelerinin sağlanması için bir kaç farklı yaklaşım bulunmaktadır. Bu yöntemler sayesinde uygulamalar izlenerek; şüpheli olarak değerlendirilen ve hariçten izin talebinde bulunan bu uygulamaların kötücül olup olmadıklarının belirlenmesi amaçlanmaktadır. Bu yöntemlere kısaca bakacak olursak; birincisi ve en temel kullanılan yöntem statik analiz metodolojisidir. Buna göre uygulamalar henüz çalıştırılmadan izinleri ve kaynak kodları incelenerek analiz edilirler. Bu şekilde uygulamanın hariçten izin isteyip istemediği, istiyor ise nedeni bulunmaya çalışılır [7][8][9]. İkinci yöntem dinamik analiz metodudur. Buna göre incelenmek istenen uygulama bir benzetimlik üzerinde koşularak, hangi izinleri hangi kaynaklara erişmek için kullandığı tespit edilmeye çalışılır. Böylece, uygulamanın servis çağrı karakteristiği ve erişim stratejisi belirlenerek uygulamanın kötücül bir aktiviteye yönelip yönelmediği ortaya konulmaya çalışılır [10][11]. Üçüncü yöntem, 3. parti yazılımları kullanarak mobil araçlardaki tüm uygulamaların hangi kaynaklara eriştiklerini ve hangi izinleri kullandıklarını listelemedir. Böylece, daha önce yüklenilen veya gözden kaçmış olan bir uygulama var ise tespit edilerek kötücül aktiviteleri gözlemlenebilecektir. Dördüncü yöntem ise işletim sistemi ayrıcalık engelidir. Buna göre, işletim sistemleri uygulamalarının bazı kişisel verilere erişim sağlayıp sağlamadıkları kontrol edilebilmektedir. Örneğin, Android kullanıcılara böyle bir imkân tanımaktadır. Kullanıcılar herhangi bir marketten veya doğrudan üçüncü parti bir uygulamayı indirip kurmak istediklerinde, uygulamanın hangi izinleri talep ettikleri ve hangi kaynaklara erişmek istediklerine dair bir liste göstermektedir. Uygulamaların mobil araçlara yüklenmesi ancak kullanıcıların bu listeyi onaylamaları sonrasında mümkün olmaktadır. Böylece, bir ön güvenlik sistemi sağlanarak kullanıcılar bilgilendirilebilmektedir. Tüm bu teknikler bazı güvenlik çözümleri üretseler de, uygulamanın ihtiyaç duyduğundan daha fazla izin isteyip istemediğine dair bir sonuç çıkarmak için ortagonal metotların izlenmesi gereklidir [5].

Bu çalışmada, uygulamaların ihtiyaç duyulandan fazla izin talebinde bulunup bulunmadıkları, bulundular ise bunu kullanıp kullanmadıkları analiz edilerek bir kötücül yazılım tespit aracı geliştirilmeye çalışılmıştır. Bu yaklaşıma göre, statik analiz metodunun avantajları ile birlikte kaynak kod analizi yapılarak uygulamanın potansiyel kötücül risk durumuna ilişkin bir değerlendirme sonucu sunulması amaçlanmıştır.

2. İLGİLİ ÇALIŞMALAR(RELATED WORKS)

Bu bölümde, kullanıcıların sahip oldukları Android işletim sistemli mobil cihazları dolayısı ile karşılaştıkları güvenlik ihlalleri ve izin kullanımına yönelik kötüye kullanmaya dair daha önce yapılmış olan çalışmalar verilmiştir.

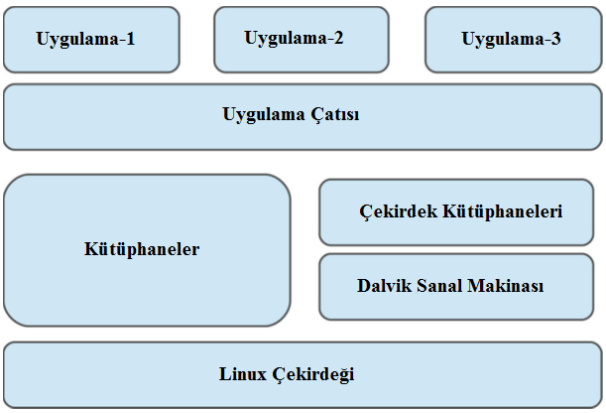
Dinamik analiz tabanlı olarak çalışmakta olan güvenlik çözümlerine ilişkin bazı çalışmalar şunlardır: TaintDroid [10] android eklentilerinin, hassas verileri üçüncü parti uygulamalar üzerinden belli merkezlere ilettiğini ortaya koymuştur. Aynı şekilde AppsPlayground [12] kötücül fonksiyonların ve gizli bilgilerin sızdırılması teşebbüslerinin tespit edilmesi için dinamik analiz altyapısını kullanan bir altyapı sunmaktadır. [13][14] Daha önce tanımlanmış politikalar ışığında kullanıcıların sahip oldukları verilerinin kontrol edilebileceği bir çalışma görüntüleyici sunmuştur. [15] 2010 yılında Enck'in çalışmasını geliştirerek, daha ince detaylarda güvenlik politikalarının tanımlanıp, uygulamaların bu detaylı politikalar dahilinde incelenebileceği bir çalışma yapmışlardır. MockDroid [16] kullanıcılara daha önce vermiş oldukları izinleri tekrar geri alabilme imkanı tanımaktadır.

Statik analiz güvenlik ve gizlilik çözümleri: [17][18] Dinamik analiz metodolojilerini tamamlayacak şekilde, statik analiz tabanlı olarak uygulamaların kullanıcı gizliliği açısından güvenli olup olmadığını tespit edebilecek çalışma yapmışlardır. AppProfiler [19] verilen uygulamaların gizlilik seviyelerini belirlemek için kullanılabilecek olayların incelenmesine imkan tanıyan bir çalışmadır. ScanDroid [20] uygulamaların manifest dosyalarından istemiş oldukları izinleri çıkarmakta ve bunların hangilerinin kullanıcı bilgilerinin sızdırılması için kullanılabileceğini değerlendirmektedir.

Yetki tabanlı çözümler: Kullanıcı verilerinin istek dışı hareketlerinin kontrol edilmesi ve engellemesi için kullanılan ve yukarıda verilmiş olan tekniklerin dışında, yetki tabanlı çözümlerde bulunmaktadır. [7][8] Uygulamaların statik analiz edilmesi ve uygulamanın gerek duyduğundan fazla izin talep edip etmediğinin tespit edilmesine yönelik çalışmadır. Her bir çözüm, izin haritasının çıkarılması ile başlamakta ve bu haritanın analiz edilerek meydana gelmesi muhtemel güvenlik risklerinin ortaya çıkarılmasını amaçlamaktadır. [21] Zincirleme olarak meydana gelebilecek tehditlerin tespit edilmesine imkan tanıyan bir tarayıcı araç geliştirmişlerdir. Bu araç mobil cihazlara diğer uygulamalar gibi yüklenmektedir. Çalıştırılması durumunda, diğer uygulamaların manifest dosyalarını analiz etmek ve muhtemel zayıf yönlerini göstermektedir. [5] benzer şekilde, uygulamaların manifest dosyasında istemiş oldukları izinleri istemekte ve bunların değerlendirmesini yapmaktadır. Uygulamaların talep ettiklerinden daha fazla izin istemeleri durumunda şüpheli olabilecekleri değerlendirmesinde bulunmaktadır.

3. ANDROİD İŞLETİM SİSTEMİ İZİN TABANLI GÜVENLİK MODELİ (ANDROID PERMISSON BASED SECURITY MODEL)

Android işletim sistemi Linux tabanlı bir model olması sebebiyle kişisel verilerin korunabilmesi, sistem kaynaklarına erişimin sınırlandırılması ve Java'dan kaynaklı problemlerin kolayca çözülmesine imkân tanımaktadır. Şekil-1'de "Android" işletim sisteminin yapısı genel hatları ile gösterilmiştir.



Şekil 1. Android Yazılımı Modüler Yapısı(Android Software Architecture) [5]

Android, her bir uygulamayı kendi kullanıcı yetkileri altında çalıştırır ve her birine eşsiz bir numara tanımlar. Bir kullanıcı altında birden fazla uygulama çalıştırmaya izin veren işletim sistemlerinden bu yönüyle ayrılır

Varsayılan olarak, uygulamalar, farklı bir kullanıcıyı veya uygulamayı etkileyecek şekilde koşturulamazlar ve normal durumlarda kendilerine tahsis edilen sınırlı bir kaynak grubuna erişmeleri mümkündür. [22] Uygulamalar tüm hassas operasyonları ile ilgili olarak manifest dosyasında gerekli izinleri tanımlamalıdırlar. Böylece kullanıcılar uygulamaları mobil araçlarına kurarlarken bu izinleri görürler ve onaylamaları halinde ilgili yazılım kurulumu gerçekleşmiş olur. Uygulamalar sadece manifest dosyasında tanımlanan izinlerin izin verdiği çerçevede kaynaklara erişebilirler. Bu şekilde, izin mekanizması sayesinde kullanıcılar için bir güvenlik katmanı oluşturulmuş olur. Örnek bir manifest dosyası Şekil-2 de gösterilmiştir.

```
<android.permission.CAMERA/>
<android.permission.WRITE_EXTERNAL_STORAGE/>
<android.permission.INTERNET/>
<android.permission.ACCESS_NETWORK_STATE />
<android.permission.READ_PHONE_STATE/>
<android.permission.READ_CONTACTS/>
<android.permission.VIBRATE/>
<android.permission.WRITE_CALENDAR/>
```

Şekil 2. Manifest Dosyası (Manifest File)[5]

Şekilde de gösterildiği gibi bir uygulama eğer mobil aracın kamera sistemine erişmek, bilgi alış verişi yapmak istiyor ise, "android.permission.camera" şeklinde izin talebini manifest dosyasında deklare etmelidir. Böylece kullanıcılar da mobil araçlarına yükleyecekleri uygulamanın kamera sistemine erişeceğini görebilecektir. Eğer kullanıcı bunu kendisi için riskli bir durum olarak görüyor ise de uygulamayı reddederek güvenliği tehdit edici bu uygulamadan kurtulmuş olabilecektir. Eğer uygulamalar gerek duydukları izinleri talep etmeden, bu sistemlere erişmek isterlerse, bu durumda Android işletim sistemi tarafından güvenlik olağandışılığı fırlatılır ve uygulamanın çalışması sonlandırılır.

Mobil uygulamalar, konum bilgisinden çağrı geçmişine, mesajlardan maillere kadar birçok kişisel veriyi yönetmektedirler. Bu bilgiler kişilerin profilinin çıkarılması açısından kıymetli girdilerdir. Sadece kötücül uygulamalar değil, aynı zamanda zararsız uygulamalar da bu bilgilere erişmeyi istemektedirler. Örneğin Twitter uygulaması, kişilerin izni olmadan kullanıcı bilgilerini almaktadır. Uygulamalar bu bilgileri sadece almakla kalmayıp silme ve değiştirme gibi işlemler de yürütebilmektedirler. Kişisel veriler üzerinde manipülasyon işlemlerine ilişkin olarak detaylı analizlere [18][23] kaynaklarından erişilebilir.

Kötücül uygulamalar, oldukça kıymetli olan kişisel verilere erişmek için her zaman kendi izinlerini kullanmazlar. Bazı durumlarda, zararsız olan uygulamaların sahip oldukları izinleri kullanarak da bu bilgilere erişim izni elde edebilirler [24]. Böylece kullanıcı izin vermese bile kişisel bilgilerine ulaşılabilir. Bu gibi durumlar gözetildiğinde, uygulamaların ihtiyaç duyduklarından daha fazla izin talep ettikleri durumlarda, eğer kullanıcılar bu durumu onaylarlar ise uygulamalar rahatlıkla kötücül bir moda geçebilirler. Bu işlemleri gerçekleştirmek için uygulamalar genelde güncelleme yöntemini kullanırlar [21]. Fazladan talep ettikleri izinler başlangıçta herhangi bir şekilde kullanılmaz iken, bir uygulama güncellemesi sonrasında, aktif olarak kullanılır hale gelebilir. Bu durumda uygulamalar kullanıcılar için riskli bir durum oluşmasına neden olurlar.

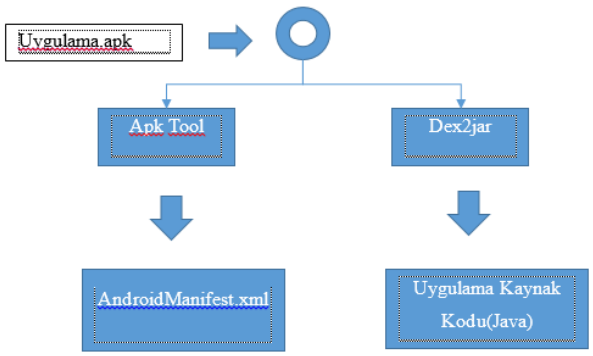
Mobil kötücül yazılımlar bu şekilde farklı yöntemler izleyerek, mobil kötücül yazılım tespit araçlarını atlatmak ve kötücül aktivitelerini hedef mobil araçlar üzerinde göstermek istemektedirler. Bu yazılımlara ilişkin üretilen güvenlik mekanizmaları da, Android'in temel felsefesini ve çalışma yapısını kullanmaktadır. Android işletim sisteminde izni önceden alınmamış hiç bir aktivite koşturulamaz [5]. Aynı zamanda, kötücül uygulama yapıları doğrudan kullanıcıdan izin isteyerek uygulama katmanı seviyesinde çalışmak yerine, bu izni almadan bir şekilde sistem kaynaklarına erişmek için farklı yöntemleri kullanabilmektedirler. Sonuçta kötücül aktiviteler bu şekilde sistemsel veya kullanıcı davranış zayıflıklarını kullanarak faaliyetlerini yerine getirmektedirler. Bu durumda, en az izin en iyidir prensibi doğru bir güvenlik kurgusu yaklaşımı olacaktır. Yapılacak çalışmada bu yaklaşım temelinde, gerektiğinden fazla izin talep eden uygulamalar özel olarak incelenecek ve güvenlik konusunda zaafları olabileceği ön yargısı ile uygulamalar güvenlik tespit yapısına tabi tutulacaklardır [5].

4. ANDROİD GÜVENLİK YAKLAŞIMI VE TEORİK ALTYAPI (ANDROID SECURITY APPROACH AND THEORETICAL FRAMEWORK)

Bu çalışmada mobil uygulamaların profillerinin çıkarılması ve analiz edilmesi için verimli ve hızlı bir model ortaya koymaya çalışılmıştır. Bunun için uygulamaların talep etmiş oldukları izinler değerlendirilmiştir. Metodolojinin temelinde uygulamaların paketlerinin açılması, izinlerinin ve kaynak kodlarının elde edilerek, uygulamaların davranışlarının tahmin edilmesi prensibi yatmaktadır. Buna göre uygulamaların bir şekilde paketlerinin açılması ve izin dosyası ile kaynak kod dosyalarının elde edilmesi gerekmektedir. Bunun için statik analiz kullanılarak ilgili dosyalara erişim sağlanmış, karşılaştırma işlemleri yapılarak izinlerin kullanılma durumları belirlenmiştir. Bu karşılaştırma sonucunda da, uygulamanın gerçekten fazladan izin talebinde bulunup bulunmadığı ve bunun ne seviyede gerçekleştiği değerlendirilerek, kötücül yazılım tespit edilmeye çalışılmıştır.

4.1. Paketlerin Açılması ve Analiz(Application Analysis And Repackaging)

Android mobil uygulamalarından doğrudan Java byte kod yerine Dalvik byte kodunu elde etmek amaçlanmaktadır. Java tabanlı uygulamalar üzerinde, Jad(Jad Java Derleyicisi), ASM gibi araçlarla kolay bir şekilde tersine mühendislik gerçekleştirebilir. Benzer şekilde Apktool aracı ile de mobil uygulamalar analiz edilebilir ve üzerinde değişiklikler gerçekleştirilebilir. Dexpler [25], Dex2jar, Androguard gibi uygulamalar da benzer şekilde kullanılabilirler. Tersine mühendislik aşamasında her bir aracın kabiliyeti uygulamadan uygulamaya değişebilmektedir. Yapılan çalışmada uygulamanın izin dosyasına sağlıklı bir şekilde erişebilmek için Apktool aracı tercih edilmiştir. Bu şekilde düzgün okunup, ayrıştırılabilir manifest dosyası elde edilebilmiştir. Uygulamanın kaynak koduna erişmek içinse, Dex2jar aracı tercih edilmiştir. Böylece basit bir işlemle uygulamanın "classes.dex" dosyası kullanılarak kaynak koda erişim sağlanmıştır. Bu işlemlere ilişkin yapı Şekil-3'te gösterilmiştir.



Şekil 2. Uygulama Tersine Mühendislik Diyagramı (Application Reverse Engineering Diagram)

Buna göre, uygulama ApkTool ve Dex2jar aracına verilir, sonucunda elde edilen değerler incelenmek üzere hazırlanmış olan Java uygulamasında işlenir.

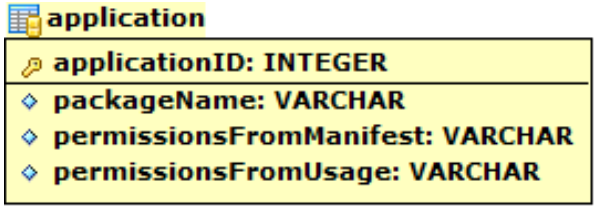
4.2. Teorik Yaklaşım(Theoretical Approach)

Uygulamaların risk tespitlerinin yapılıp değerlendirilebilmeleri için öncelikle Android işletim sistemi için kaç adet izin tipinin olduğunun belirlenmesi gereklidir. Bunun için Android resmi sitesinden toplam kaç farklı izin olduğu belirlenmiştir. Şekil-4'te bir kısmı gösterildiği gibi toplam 135 farklı izin tipi bulunmaktadır.

İzin ID	İzin ismi	İzin Tipi
1	ACCESS_CHECKIN_PROPERTIES	String
2	ACCESS_COARSE_LOCATION	String
3	ACCESS_FINE_LOCATION	String
4	ACCESS_LOCATION_EXTRA_COMMANDS	String
5	ACCESS_NETWORK_STATE	String
6	ACCESS_NOTIFICATION_POLICY	String
7	ACCESS_WIFI_STATE	String
8	ACCOUNT_MANAGER	String
9	ADD_VOICEMAIL	String
10	BATTERY_STATS	String
11	BIND_ACCESSIBILITY_SERVICE	String
12	BIND_APPWIDGET	String
13	BIND_CARRIER_MESSAGING_SERVICE	String
14	BIND_CARRIER_SERVICES	String
15	BIND_CHOOSER_TARGET_SERVICE	String
16	BIND_DEVICE_ADMIN	String
17	BIND_DREAM_SERVICE	String
18	BIND_INCALL_SERVICE	String
19	BIND_INPUT_METHOD	String
20	BIND_MIDI_DEVICE_SERVICE	String
TOPLAM 135 ADET İZİN BULUNMAKTADIR.		

Şekil 4. Android İşletim Sistemi İzin Listesi(Android Permission List)[18]

Şekil 4'te verilen izinler ışığında tüm uygulama kaynak kodu ve manifest dosyası taranarak bulunması halinde 1 bulunamaması halinde 0 olacak şekilde 1x135'lik bir uygulama matrisi elde edilmektedir. Bu matris hem manifest hem de kaynak kod için ayrı ayrı hesaplanmaktadır. Bu hesaplama işlemi tamamlandıktan sonra da veri tabanında her bir uygulama bir satıra denk gelecek şekilde Çizelge üzerine kaydedilmektedir. Şekil-5'te veri tabanı mimarisinin izin kayıt Çizelgesu gösterilmiştir.



Şekil 5. Veri Tabanı Uygulama Kayıt Çizelgesu(Database Application Registry Table)

Şekil-5'te gösterildiği gibi, 1x135'lik matris bir metin haline getirilerek veri tabanına kaydedilir. Böylece daha sonra yapılacak genel değerlendirmede her bir uygulama sırası ile "applicationID"leri yardımıyla bu Çizelgedan çekilerek kullanılabilecektir. Bu yapıda kötücül ve zararsız uygulamalar için aynı Çizelgedan iki adet oluşturulması mantığı tercih edilmiştir. Böylece kötücül ve zararsız uygulamalar şeklinde 2 ayrı veri seti elde edilmiş olacaktır.

4.3. **Risk Tespiti ve Değerlendirme** (Risk Identification and Evaluation)

Uygulamalar ortaya konulan teorik yaklaşıma göre elde edilen veriler ışığında bir değerlendirmeye tabi tutulurlar. Bu yaklaşıma göre; manifest dosyasında uygulamaların talep etmiş oldukları izinler(M) ve bunların toplam sayısı(MP), Java kaynak kodunda sistem çağrısı yaptıkları izinler(U) ile izinlerin toplam sayısı(UP) olarak tanımlanacak olursa, aşağıda verilmiş olan Java kodu kullanılarak değerlendirilirler

```
for (int j = 0; j < MP.length; j++) {
                if (MP[j].equals("0") && UP[j].equals("1")) {
                    warncount++;
                } else if ((MP[j].equals("1") && UP[j].equals("0"))) {
                    suspiciouscount++;
                    List<MyArray> liste = new ArrayList<>();
                    liste = showUsagePermissions();
                    String retrieve = ret.getPermissionNameGivenID(j);
                    for (int n = 0; n < liste.size(); n++) {
                        String listename = liste.get(n).getName();
                        if (retrieve.equals(listename)) {
                            suspiciousvalue += liste.get(n).count;
                        }
                    }

                } else {
                    normalcount++;
                }
            }
```

Şekil 6. İzin Tabanlı Kötücül Yazılım Değerlendirme Java Kodu(Permission Based Malware Evaluation Java Code)

Şekil 6'da verilmiş olan bir kısım Java kodunun da yaptığı gibi uygulamanın manifest dosyasında talep etmiş olduğu izinler ile bu izinlerden kaç tanesinin uygulama içerisinde kullanıldığı, fazla/eksik kullanımın söz konusu olup olmadığı belirlenmektedir. Bu değerlendirme için Şekil-6'da verilen Java kodu Çizelge-1'deki gibi bir değerlendirme yapmaktadır.

Çizelge 1. Değerlendirme Matrisi (Evaluation Matrix)

Mİ (Manifest Dosyası İzinleri	Kİ (Kullanılan İzinler)	SONUÇ
1	1	NORMAL
0	0	NORMAL
1	0	ŞÜPHELİ
0	1	İKAZ

Bu Çizelgeya göre, Şekil-5'te verilmiş olan izinler uygulama manifest dosyasında aranarak, bulunması halinde 1, bulunmaması halinde 0 olacak şekilde değerlendirilerek

veri tabanına kaydedilir. Sonrasında uygulamanın bu izinleri kullanıp kullanmadığı tek tek kontrol edilmektedir. Bunun için de kendi kendini çağıran bir fonksiyon yardımıyla uygulamanın var olan tüm dizin yapısı çıkarılmaktadır. Sonrasında metin arama metodu kullanılarak bu izni Java kaynak kodunun herhangi bir yerinde çağırıp çağırmadığı belirlenmektedir. Buna göre 0 veya 1 olacak şekilde değerlendirilmesi yapılmaktadır. Çizelge-1'deki matrise göre bu iki değer karşılaştırılarak her bir uygulama için şüpheli değer sayısı ortaya çıkarılmaktadır. Bu değer bir uygulamanın kötücül olup olmadığını değerlendirmemiz için kritik bir sonuçtur.

Bu değerlendirmeler sonucunda elde edilen şüpheli durumların sayısı bir uygulamanın çalışması için gerekli izinden fazlasını talep ettiği ve onun bir kötücül yazılım olabileceği riskinin bulunduğu gösterir. Ancak bazı kötücül karakteristiğe sahip olmadığı düşünülebilecek uygulamalar için de benzer durumlarla karşılaşılmaktadır. Bunun uygulama geliştiricilerin hatalarından kaynaklanabileceği düşünülmektedir. Bunun gibi hatalı durumları ortadan kaldırıp daha doğru kötücül yazılım tespiti yapmak için bu yönteme bir çarpan değeri eklenmiştir. Bu çarpan değeri Şekil-6'da örnek olarak gösterildiği gibi bir iznin kötücül veri seti içerisinde kullanılma sayısından elde edilmiştir. Gerçekte fazladan/gereksiz yere talep edilen bu izinlerin kötücül yazılımlar içerisinde kullanılma frekansı dikkate alınmaktadır. Yaklaşıma göre, bir uygulamanın fazladan talep etmiş olduğu izin kötücül uygulamalar arasında yoğunlukla kullanılan bir izin ise bu durum o uygulamanın gerçekten kötücül olma riskini artıran bir unsur olmaktadır.

Çizelge 2. Frekans Çizelgesu(Frequency Table)

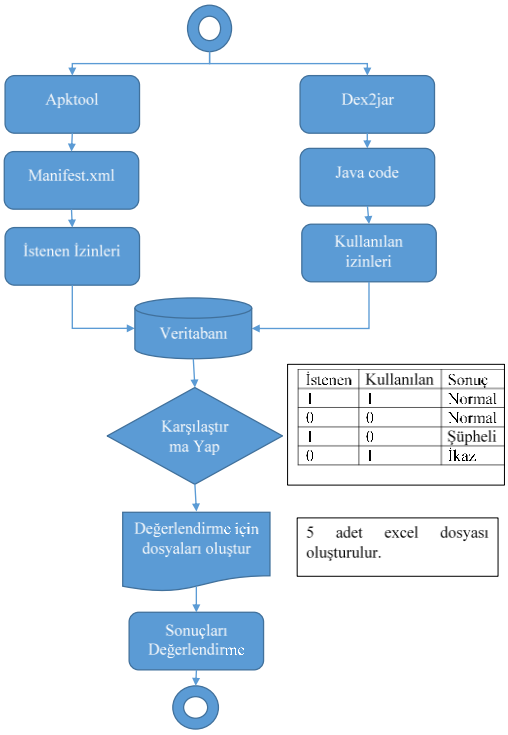
İstenilen izinler	**Kullanılma Sayısı**
INTERNET	49
READ_PHONE_STATE	46
WRITE_EXTERNAL_STORAGE	41
ACCESS_NETWORK_STATE	38
ACCESS_WIFI_STATE	36

Çizelge-2'de örnek olarak verildiği gibi kötücül uygulamalar için "INTERNET" izninin kullanım frekansı oldukça yüksektir. Eğer bir uygulama bu izni isteyip, kullanmamış ise oldukça riskli bir gruba girmiş olacaktır. Aşağıda verilen formüle göre bu değerlendirme yapılarak her bir uygulama için şüphe değeri elde edilmektedir.

$$Şüphe\ Değeri = \sum_{i=1}^{n} ŞüpheFrekans_i,\ n = şüpheli\ izin\ sayısı \quad (1)$$

Her bir uygulama için şüpheli izin sayısına göre (1) numaralı yaklaşım doğrultusunda şüphe değeri hesaplanmaktadır. Bu hesaplama sonucunda oluşan şüphe değeri eğer zararsız uygulama veri seti şüphe değeri ortalamasından yüksek ise o uygulama kötücül riskli olarak belirlenmektedir. Bu şekilde uygulamaların kötücül olup olmadıkları belirlenebilmektedir.

Ortaya konulmuş olan metodolojik yaklaşımın genel akış diyagramı Şekil-7'de gösterildiği gibidir



Şekil 7. Akış Diyagramı (Flow Chart)

Bu diyagrama özetlenecek olursa; öncelikle uygulama için Dex2jar ve Apktool araçları ile tersine mühendislik işlemi gerçekleştirilir. Sonrasında "Manifest" ve "Java Kaynak Kod" elde edilir. Bu iki elde edilen paket hazırlanan yazılım sayesinde ayrıştırılarak, 135 adet izin ışığında veri tabanına kaydedilirler. Bu kayıt sonrasında, uygulamanın istemiş olduğu ve kullandığı izinler şekilde verilen Çizelge ışığında karşılaştırılır ve sonuçta şüpheli görülen izinler listesi çıkarılır. Bu çıkarılan liste ile frekans değeri çarpımı sonucunda elde edilen değer incelenmek üzere ayrıca Excel dosyalarına aktarılır ve sonuçlar değerlendirilir. Bu değerlendirme sonucunda, zararsız uygulamalar için belirlenmiş olan ortalamaların üzerinde şüphe değerine sahip uygulamalar kötücül olarak belirlenir. Tüm bu işlemler hazırlanan Java tabanlı yazılım sayesinde gerçekleştirilmektedir.

5. UYGULAMA (Application Details)

Çalışmanın dördüncü bölümünde teorik altyapısından bahsedilmiştir. Bu altyapı doğrultusunda aşağıda verilen kabuller ve veri setleri ışığında uygulama hazırlanmış ve sonuçları elde edilmiştir. Buna göre;

Değerlendirme Ölçütleri:

1- Uygulamaların değerlendirilmesi için Android işletim sistemi resmi web sitesinde paylaşılan 135 adet farklı izin referans izin listesi olarak alınmıştır.
2- Yeni uygulamaların değerlendirilmesi için 50 adet kötücül ve 25 adet zararsız uygulama seçilmiştir.
3- Kötücül uygulamalar Drebin veri setinden [26], Zararsız uygulamalar ise sadece banka mobil uygulamaları arasından seçilmiştir.
4- Uygulamaların Java kaynak kodlarını elde etmek için Dex2jar programı tercih edilmiştir.
5- Uygulamaların "AndroidManifest.xml" dosyalarını elde etmek için Apktool programı tercih edilmiştir.
6- Tüm bu işlemleri gerçekleştirmek üzere Java programlama dili ve Mysql veritabanı yönetim sistemi Apache ile birlikte kullanılmıştır.

Bu değerlendirme ölçütleri ışığında elde edilen sonuçlar Çizelge-3'te sunulduğu gibidir.

ortaya çıkmaktadır. Örneğin uygulamaların "INTERNET" izni talep etme sayısı 49 iken, bu iznin kullanılma oranı 23'tür. Bu bariz fark bu iznin birçok uygulama tarafından talep edilmekte ancak kullanılmamakta olduğunu göstermektedir. Kötücül uygulamaların aslında "INTERNET" iznini talep etmekte ve kullanmakta hevesli oldukları ancak bunların büyük bir kısmının da bu izni daha sonra kullanmak üzere istedikleri ancak kullanmadıkları gözlemlenmiştir. Bu durum kötücül uygulamalar açısından, oldukça şüphe uyandırıcı bir mesaj vermektedir.

Benzer şekilde, 25 adet zararsız veri setinin incelenmesi sonucu elde edilen izin kullanılma listesi Çizelge-4'te gösterilmiştir. Çizelge incelendiğinde, bazı izinler için istenme ve kullanılma durumları açısından farklar oluştuğu gözlemlenmiş olsa da bu farklar sadece bir kaç izin için oldukça azdır. Bu durumda da, bu karşılaştırma sonucunda elde edilecek fark değerleri hem kötücül uygulama veri setleri için hem de zararsız veri setleri için kullanıla-

Çizelge 3. Uygulama Sonucu Elde Edilen Frekans Değerleri (Kötücül Veri seti) (Frequecy Values Obtained After Application(Malware Data Set))

Uygulama Manifest Dosyası İzin Talep Frekansı		**Uygulama Kaynak Kodu İzin Talep Frekansı**	
İstenilen izinler	**Kullanılma Sayısı**	**Kullanılan İzinler**	**Kullanılma Sayısı**
INTERNET	49	ACCESS_FINE_LOCATION	27
READ_PHONE_STATE	46	ACCESS_COARSE_LOCATION	26
WRITE_EXTERNAL_STORAGE	41	INTERNET	23
ACCESS_NETWORK_STATE	38	VIBRATE	19
ACCESS_WIFI_STATE	36	ACCESS_NETWORK_STATE	18
WAKE_LOCK	29	INSTALL_SHORTCUT	16
ACCESS_COARSE_LOCATION	26	READ_PHONE_STATE	15
ACCESS_FINE_LOCATION	25	CAMERA	13
RECEIVE_BOOT_COMPLETED	22	DUMP	12
INSTALL_SHORTCUT	18	WRITE_EXTERNAL_STORAGE	10
VIBRATE	18	GET_ACCOUNTS	9
SEND_SMS	17	UNINSTALL_SHORTCUT	8
UNINSTALL_SHORTCUT	13	READ_CONTACTS	8
GET_ACCOUNTS	12	ACCESS_WIFI_STATE	8
READ_SMS	12	CALL_PHONE	6
READ_CONTACTS	10	SEND_SMS	5
TOPLAM(135 ADET)		TOPLAM(135 ADET)	

Çizelge-3'te, incelenen 50 adet kötücül olduğu varsayılan uygulama için elde edilen izin frekans değerleri gösterilmektedir. Çizelge incelendiğinde, aslında uygulamanın teorisinde anlatılan tüm durumlar sayısal değer olarak bilir bir şüphe değeri elde etmemize imkân tanımaktadır. Ortaya çıkan bu izin kullanım farkları sonucunda incelenen 50 adet kötücül uygulama için, elde edilen sonuçlar Çizelgesunun bir bölümü Çizelge-5'te verilmiştir

- **Şüpheli İzinler:** Uygulamada istenilip kaynak kodda kullanılmayan izinlerin toplam sayısıdır.

Çizelge 4. Uygulama Sonucu Elde Edilen Frekans Değerleri (Zararsız Yazılım Veri seti) (Frequecy Values Obtained After Application(Benign Data Set))

Uygulama Manifest Dosyası İzin Talep Frekansı		**Uygulama Kaynak Kodu İzin Talep Frekansı**	
İstenilen izinler	Kullanılma Sayısı	Kullanılan İzinler	Kullanılma Sayısı
INTERNET	24	INTERNET	15
ACCESS_NETWORK_STATE	20	WAKE_LOCK	15
ACCESS_FINE_LOCATION	17	NFC	14
WRITE_EXTERNAL_STORAGE	16	VIBRATE	11
ACCESS_COARSE_LOCATION	15	BLUETOOTH	10
WAKE_LOCK	12	READ_PHONE_STATE	9
READ_PHONE_STATE	11	WRITE_EXTERNAL_STORAGE	9
GET_ACCOUNTS	10	ACCESS_NETWORK_STATE	8
ACCESS_WIFI_STATE	10	ACCESS_WIFI_STATE	7
READ_CONTACTS	10	CAMERA	6
VIBRATE	9	GET_ACCOUNTS	6
CALL_PHONE	9	ACCESS_COARSE_LOCATION	4
CAMERA	9	DUMP	4
RECEIVE_SMS	8	SEND_SMS	4
TOPLAM(135 ADET)		TOPLAM(135 ADET)	

Çizelge 4. 10 Adet Kötücül Uygulama İçin İzinleri Gösterir Durum Çizelgesu(Status Table Showing Permissions for 10 Malware Application**)**

ID	**İstenilen İzinler**	**Kullanılan izinler**	**Normal izinler**	**Şüpheli izinler**	**İkaz**
1	2	0	133	2	0
2	12	5	126	8	1
3	11	0	124	11	0
4	11	4	124	9	2
5	10	0	125	10	0
6	10	5	126	7	2
7	14	13	126	5	4
8	5	13	123	2	10
9	12	7	130	5	0
10	8	7	124	6	5
TOPLAM UYGULAMA SAYISI KADAR SONUÇ BULUNMAKTADIR.					

İstenen İzinler: Uygulamanın manifest dosyasında istemiş olduğu izinlerin toplam sayısıdır.

- **Kullanılan İzinler:** Uygulamanın kaynak kodunda çağrı yaptığı izinlerin toplam sayısıdır.
- **Normal İzinler:** Manifest ve kaynak kodda isteyip kullanıldığı veya istemeyip kullanmadığı izinlerin toplam sayısıdır.
- **İkaz:** Manifest dosyasında istenilmeyip, kaynak kodda çağrı yapılan izinler toplamıdır.

Çizelge-5 incelendiğinde, her bir uygulama için 5 farklı izin sınıflandırması yapıldığı görülmektedir. Bu karşılaştırma Çizelge-1'de verilen sistematiğe göre yapılmaktadır. Bu değerlendirme sonucunda her bir uygulamanın şüpheli izin sayıları çıkarılmaktadır. 10 adet uygulama

için sunulan listede görüleceği üzere şüpheli izni hiç olmayan bir uygulama yoktur. Her bir uygulamanın belirlenen risk

matrisine göre hesaplanacak bir şüphe değeri olacaktır. Bu gözlem sonucunda, uygulamalar için bir risk değeri(şüphe değeri) hesaplanarak sınıflandırma yapılabileceği düşünülmüştür. Aynı şekilde zararsız uygulamalar için Çizelge-6'da sunulan örnek gösterge Çizelgesunda, şüpheli izin sayısı 0 olan birçok uygulama olduğu ve birçoğunun da oldukça düşük sayıda şüpheli izne sahip oldukları görülecektir. Bu genel durum dahi bize bu noktada kötücül ve zararsız uygulamalar arasında bu noktada ciddi farklar olduğunu göstermektedir.

Çizelge 5. 10 Adet Zararsız Uygulama İçin İzinleri Gösterir Durum Çizelgesu(Status Table Showing Permissions for 10 Benign Application)

ID	İstenilen İzinler	Kullanılan izinler	Normal izinler	Şüpheli izinler	İkaz
1	1	0	134	1	0
2	0	0	135	0	0
3	10	9	124	6	5
4	2	0	133	2	0
5	2	1	132	2	1
6	12	9	124	7	4
7	1	0	134	1	0
8	11	29	113	2	20
9	5	4	130	3	2

Çizelge 6. Kötücül Uygulamalar için Şüphe Değeri Çizelgesu(Suspicious Value Table for Malware Applications)

ID	İstenilen İzinler	Kullanılan izinler	Normal izinler	Şüpheli izinler	İkaz	Şüphe Değeri	Ortalama
1	2	0	133	2	0	16	
2	12	5	126	8	1	49	
3	11	0	124	11	0	78	
4	11	4	124	9	2	21	
5	10	0	125	10	0	41	
6	10	5	126	7	2	38	
7	14	13	126	5	4	48	
8	5	13	123	2	10	2	
9	12	7	130	5	0	3	
10	8	7	124	6	5	38	
11	15	7	127	8	0	54	
12	13	2	124	11	0	33	
13	11	0	124	11	0	78	
14	11	8	126	6	3	38	
15	6	2	131	4	0	2	
16	8	7	132	2	1	0	
17	5	0	130	5	0	21	
18	13	0	122	13	0	40	
19	17	10	122	10	3	55	
20	7	9	123	5	7	2	

Çizelge 7. Kötücül Uygulamalar için Şüphe Değeri Çizelgesu(Suspicious Value Table for Malware Applications) (Devamı)

ID	İstenilen İzinler	Kullanılan izinler	Normal izinler	Şüpheli izinler	İkaz	Şüphe Değeri	Ortalama
21	11	1	125	10	0	62	
22	9	7	131	3	1	34	
23	6	3	128	5	2	21	
24	8	4	131	4	0	16	
25	13	7	125	8	2	67	
25-50							
					Ortalama Değer	1654	33.03

Böylece uygulamalar ile ilgili genel bir değerlendirme Çizelgesu oluşturulmuştur. Çizelgelar değerlendirilerek (1) numaralı formül ışığında her bir uygulama için, kötücül ve zararsız uygulamalar için ayrı ayrı olarak, aşağıda verilmiş olan Çizelge-7 ve Çizelge-8 elde edilmiştir. Buna göre Çizelge-5 ve Çizelge-6'dan farklı olarak şüphe değeri her bir uygulama için hesaplanmıştır.

Çizelge 8. Zararsız Uygulamalar için Şüphe Değeri Çizelgesu(Suspicious Value Table for Benign Applications)

ID	İstenilen İzinler	Kullanılan izinler	Normal izinler	Şüpheli izinler	İkaz	Şüphe Değeri	Ortalama
1	14	9	124	8	3	9	
2	0	0	135	0	0	0	
3	10	9	124	6	5	5	
4	2	0	133	2	0	2	
5	2	1	132	2	1	0	
6	12	9	124	7	4	11	
7	14	9	124	8	3	9	
8	14	8	123	9	3	7	
9	13	10	124	7	4	7	
10	12	7	126	7	2	5	
11	11	10	120	8	7	17	
12	4	0	131	4	0	4	
13	11	0	124	11	0	23	
14	28	26	109	14	12	9	
15	11	29	113	2	20	1	
16	5	4	130	3	2	1	
17	9	17	127	0	8	0	
18	14	5	124	10	1	11	
19	8	5	128	5	2	6	
20	1	0	134	1	0	0	
21	6	0	129	6	0	4	
22	11	6	126	7	2	21	
23	1	0	134	1	0	0	
24	5	5	125	5	5	7	
					Ortalama Değer	159	6.26

Çizelge-7 ve Çizelge-8 bu bölümde anlatılan tüm teorik altyapının ve modelin var olan kötücül ve zararsız uygulama veri setlerine uygulanması sonucu elde edilen değerlerdir. Çizelgeda yeşil olarak işaretlenmiş olan hücreler, yüksek şüphe değerine sahip uygulamaları göstermektedir. Genel bir bakış ile kötücül Çizelgesundaki yeşil hücrelerin çokluğu ve aksi yönde zararsız uygulama Çizelgesundaki yeşil hücre sayısının azlığı kolayca gözlemlenebilecektir.

Ayrıca Şekil-8'de görüleceği üzere sonuçlar detaylı olarak incelendiğinde, 50 adet kötücül uygulama için toplam 1654 şüphe değeri ve ortalama 33.08 gibi bir ortalama şüphe değeri elde edilir iken, 25 adet zararsız uygulama için 159 toplam şüphe değeri ve ortalama 6.26 şüphe değeri elde edilmektedir.

Kötücül veri seti için ortalama olarak 33.08 ve zararsız veri seti için elde edilen 6.26'lık ortalama şüphe değeri arasında ciddi bir fark olduğu görülmektedir. Bu bize aslında (1) numaralı formüle göre elde edilecek şüphe değerinin, uygulamaların kötücül olup olmadığının tespit edilmesi noktasında iyi bir fikir vereceğini göstermektedir.

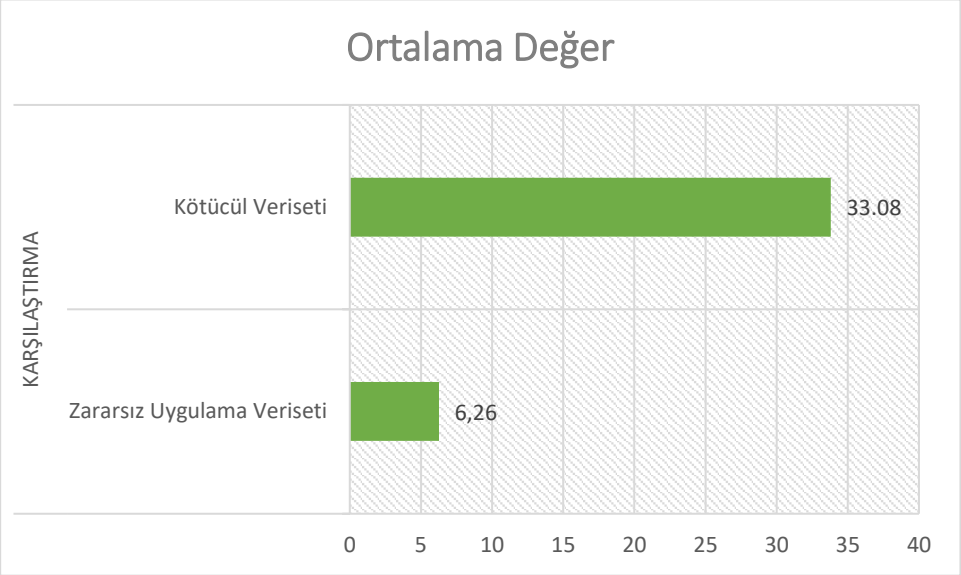
6. TARTIŞMA(DISCUSSION)

Bu çalışmada önerilmiş olan modelin kendi içerisinde tutarlı sonuçlar ürettiği, kötücül uygulamalar ile zararsız uygulamalar için elde edilen şüphe değeri yaklaşımının belli bir seviyede ayırt edici değerler elde etmemize yardımcı olduğu beşinci bölümde anlatılmıştır.

Bu bölümde Android 6.0 Marshmallow ile birlikte gelen yeni izin mekanizması karşısında modelin nasıl davranışlar sergileyeceği incelenecektir. Ayrıca, ikinci bölümde verilmiş olan bu konudaki benzer çalışmalar ile geliştirilmiş olan modelin performans değerlerinin karşılaştırılması yapılacaktır.

6.1. Android 6.0 Marshmallow ile Geliştirilen Modelin Karşılaştırılması (Comparison Between Android 6.0 Marshmallow and Developed Model)

Android işletim sisteminin son jenerasyon ürünü olan 6.0 (API Level 23) Marshmallow olarak adlandırılmıştır. Bu yeni ürün Kasım 2015 gibi çıkmış ve bir dizi güncelleme bugüne kadar yayınlamıştır [27]. Bu version "material design" olarak adlandırılan yeni sistemin ayak seslerini duyurmaktadır. Bu sisteme göre Android klasik temasından vazgeçerek, her bir uygulama için kendine özgü bir dünya hazırlamaya imkân tanıyarak, uygulamalar arasındaki geçişte yepyeni bir düzen ve şema ile karşılaşılmasını sağlamaktadır [27]. Bu yeni işletim sisteminde başta yukarıda verilmiş olan "material design" yapısı olmak üzere, bu makalede bahsedilen izin mekanizmasına ilişkin bazı değişikliklere gidilmiştir. 6.0 öncesi versiyonlarda, kullanıcıların doğrudan Android resmi marketinden veya üçüncü parti web servisleri üzerinden uygulama yüklemek istediklerinde, uygulamaların kurulması esnasında talep ettikleri tüm izinler kabul etmeleri zorunluluğu bulunmaktaydı. Bu izinleri kabul etmemeleri halinde hiç bir şekilde bu uygulamaları kullanabilmeleri mümkün olamamaktaydı. Kullanıcıların, geliştirilen bu



Şekil 8. Şüphe Değeri Karşılaştırması(Suspicious Value Comparison)

Uygulama markete veya 3.parti bir uygulama paylaşım organizasyonuna her yeni yüklenen uygulamanın şüphe değerinin belirlenmesi ve bunun kullanıcılar ile paylaşılması, uygulamayı yüklemeyi planlayan kullanıcılar açısından bir fikir verebilecektir. Hatta bazı durumlarda çok yüksek şüphe değerleri elde edilmesi halinde, uygulamalar markete yüklenmeden ciddi bir şekilde sorgulaması yapılabilecektir.

Bu sonuç göstermektedir ki, elde edilen metodoloji kendi içerinde kötücül ve zararsız uygulamalar için yol gösterici sonuçlar üretmektedir ve kendi içinde tutarlı bir yapıdır. Diğer taraftan, uygulamanın statik analiz kullanarak kötücül yazılım tespiti yapan araçlar ile karşılaştırılması çalışmanın altıncı bölümünde ayrıca değerlendirilmiştir.

yeni izin mekanizması sayesinde izinleri kurulum sonrası, uygulamaları kullanmaları esnasında kabul/ret edebilmeleri mümkün olabilmektedir. Kullanıcıların güvenlikle ilgili farkındalıklarını geliştirecek bir mekanizma kurulmuştur [27].

Bu çalışmada geliştirilmiş olan model statik analiz tabanlı olarak kod analizi yapan bir yapıda çalışmaktadır. Statik analiz metodu kullanıcıların sahip oldukları uygulamaları çalışmaları esnasında analiz etmek yerine doğrudan geliştirilmiş olan kod üzerinden analiz yapmaktadır [28]. Bu sebeple, Android işletim sisteminin izin yönetim mekanizması ile doğrudan ilişkili değildir. Android 6.0 ile gelen bu yeni izin mekanizmasında kullanıcılar için daha güvenli olan bir izin kabul/ret yapısı kurulmuştur. Bu kurguda, izinlerin fazladan talep edilip edilmedi göz önünde bulundurulmamaktadır.

Yapılan bu çalışmada, aslolan uygulamanın isteyip kullanmadığı izinlerin sayısı ve frekansıdır. Kullanıcıların bu izinleri onaylayıp onaylamadığından etkilenmemektedir. Yeni sistem içinde uygulama yüklendikten sonra kod analizini gerçekleştirip uygulamanın kötücül olup olmadığı belirlenecektir.

6.2. Bu Konudaki Benzer Çalışmalar ile Performans Değerleri Karşılaştırması(Compare Performance Values with Similar Studies)

Android kötücül yazılım tespit araçlarına ilişkin olarak statik analiz metodolojisini kullanan birçok yöntem bulunmaktadır ve bu konudaki detaylı çalışmalar ikinci bölümde verilmiştir. Bu bölümde geliştirilmiş olan android kötücül yazılım tespit aracı ile literatürde bulunan kötücül yazılım tespit araçlarından statik analiz yöntemi kullananların özellikleri Çizelge-9'da verilmiştir

Çizelge 9. Android Kötücül Yazılım Araçları Karşılaştırması(Comparison Between Android Mobile Software Tools)

Referans No	Analiz Tekniği	Algoritma	Özellikler	Veriseti	Değerlendirme	Başarı Oranı(%)
[29]	Statik	1-NN	Kod Parça Grameri	Android Genom Project	Sınırlandırma Doğruluğu	94.26
[30]	Statik	Naive Bayes	İzinler, Kod Tabanlı Özellikler	Android Genom Project, 1000 zararsız uygulama	Hata oranı, doğruluk, Gerçek Negatif Oranı, Gerçek Pozitif Oranı, Yalancı Negatiflik Oranı, Yalancı Pozitiflik Oranı, Hassaslık	93
[31]	Statik	-	İzin Kombinasyonları	Android Genom Project, 741 zararsız uygulama	Tespit Oranı	88 zararsız, 96 kötücül
[32]	Statik	Karar Ağacı, Naive Bayes	API ile ilgili özellikler ve izinler	McAfee iç kaynaklar	Gerçek Negatif Oranı, Gerçek Pozitif Oranı, Yalancı Negatiflik Oranı, Yalancı Pozitiflik Oranı, Sınıflandırma Oranı	97
[33]	Statik	Naive Bayes	API çağrıları, ikili izinler ve linux komutları	Android Genom Project, 1000 zararsız uygulama	Gerçek Negatif Oranı, Gerçek Pozitif Oranı, Yalancı Negatiflik Oranı, Yalancı Pozitiflik Oranı, Sınıflandırma Oranı	92.1
[34]	Statik	Karar Ağacı	İstenilen ve Kullanılan izin çiftleri	AppChina, Android Genom Project	Gerçek Negatif Oranı, Gerçek Pozitif Oranı, Hassaslık, Sınıflandırma Doğruluğu	98.6
[4]	Statik	Birincil Parça Analizi, Destek Vektör Makineleri	İzinler	Android Genom Project, Android Resmi Marketi	Gerçek Pozitif Oranı, Sınırlandırma Doğruluğu	90
[35]	Statik	Destek Vektör Makineleri	İzinler	Contagio Mobil ve Anzhi Marketi	Sınırlandırma Doğruluğu	75.78 - 85.11
[36]	Statik	Grup Sınıflandırma	API çağrıları ve ikili izinler	Android Genom Project	Hassaslık, F-ölçüm	98.8
[37]	Statik	VF2 algoritması	Android, Özel parçalar	Android Genom Project	Tespit Oranı	86.36
[38]	Statik	Lojistik Regresyon	İzinler	Android Genom Project, Drebin	Doğruluk, Çeşitlilik	88 (doğruluk), 92.5 (çeşitlilik)

Çizelge-9 incelendiğinde, kötücül yazılım tespit araçlarının algoritma olarak Karar ağacı, Naive Bayes teoremleri ile Destek Vektör Makinelerini yaygın olarak kullandıkları görülmüştür. Özellik olarak genelde izin tabanlı olarak çalışmış ve izin çiftlerinin karşılaştırılması yoluna gidilmiştir. Geliştirilen araçların test edilmesinde yaygın olarak kullanılan Android Genom Project ve Drebin verisetleri tercih edilmiştir. Değerlendirme ölçütü kötücül yazılımlar için doğru şekilde tespit olarak belirlenmiştir. Ve sonuçta, bütün bu seçimler sonucunda geliştiren modellerin verisetleri üzerinde çalıştırılması sonucunda elde edilen doğruluk oranları belirlenmiştir. Genel olarak incelenecek olursa, iki tanesi haricinde tamamı sadece kötücül verisetlerinde çalıştırılmıştır. Kötücül verisetleri

için doğruluk oranı %98.8 ile %85.11 arasında değişmekle birlikte, zararsız yazılım veri setleri için %75.78 ile %88 oranları elde edilmiştir.

Çalışmanın, uygulama bölümünde 50 adet kötücül ve zararsız yazılım için test sonuçlarını gösterir sonuçlar sunulmuştur. Buna göre kötücül uygulamalardan oluşan veriseti için incelenen toplam 50 uygulama için %80'lik bir doğruluk oranı elde edilmiştir. Bu oran, Çizelge-9'da sunulmuş olan uygulamalara göre düşük bir değer gibi görünebilmektedir. Ancak, geliştirilmiş olan model kötücül uygulamaların kullandıkları izin frekanslarını baz alarak sonuçlar ürettiği için uygulama sayısındaki artış ile birlikte modelin daha doğru sonuçlar elde etmesi kaçınılmazdır. Çünkü 50 adet uygulama için sonuçlar incelendiğinde, 5 adet kötücül uygulama için 0 şüphe değeri elde edilirken, 3 adet uygulama içinde 2 şüphe değeri elde edilmiştir ve bu sebeple geliştirilen modelin doğruluğunu düşüren uygulamalar olmuşlardır. Ancak 0 veya 0'a çok yakın çıkan değerler elenmesi gereken uygulamalardır [39]. Uygulama sayısının belli bir sayıyı aşması durumunda, aslında şüpheli izinlere sahip olmasına rağmen frekans değerlerinin 0 olması sebebiyle sonuçta 0 şüphe değerine sahip olmaları bu uygulamaların zararsız olduğunu göstermeyecektir. Bu uygulamaların elenmesi halinde de %97.62'lik bir doğruluk oranı elde edilmiş olacaktır. Bu da yaklaşık olarak Çizelgeda verilmiş olan 11 adet uygulama arasında ilk sıralarda yer alacak kadar doğrulukta sonuçlar üreten bir model olduğunu gösterecektir. Diğer taraftan, zararsız uygulamalar üzerinde de test edilen iki adet araç için %75.78 ve %88'lik doğruluk oranları elde edilmiştir. Bu çalışmada geliştirilmiş olan model, %80'lik bir doğruluk oranı ile sonuçlar üretmiştir. Bu oran daha önceki çalışmalara göre ortalama bir sonuçtur.

Bu bölümde daha önceki çalışmaların verisetleri üzerinde test edilmesi sonucu elde edilen doğruluk oranları ile bu çalışmada geliştirilmiş olan modelin doğruluk oranları arasında bir kıyaslama yapılmıştır. Ancak bu kıyaslamanın salt değerler üzerinden yapılması doğru bir yaklaşım da olmayacaktır. Çünkü bu araçların kullandığı verisetleri aşağı yukarı aynı olmakla birlikte, seçilen örnek uygulamaların sayısı 1000'ler seviyesindedir ve homojen bir dağılım gösterdiği söylenemez [5]. Bu sebeple, aynı uygulamalar üzerinde test edilmeyen araçların sonuçlarının değerlendirilmesi sağlıklı olmayacaktır. Ayrıca, kötücül verisetlerinin dağıtımının sınırlı olması sebebiyle tüm araçlar aynı verisetleri üzerinde test edilmişlerdir. Ancak zararsız yazılımlar için böyle bir veriseti dağıtımı bulunmamaktadır. Bu sebeple araçların hangi zararsız olduğu düşünülen yazılımlar üzerinde test edileceği belli değildir. Dolayısı ile rastgele resmi marketlerden indirilen uygulamaların zararsız olarak kabul edilmesi ve test edilerek bunların araçların doğruluğu gösterir sonuçlar olarak alınması gerçekçi karşılaştırmalar yapılmasına engel bir durumdur.

Sonuçta, bazı problemli yanları olsa, kötücül ve zararsız yazılım verisetleri üzerinde yapılan testlerde diğer araçlara göre kötücül verisetlerinde en iyi araçlara yakın bir değer elde edilebilirken, zararsız yazılım verisetlerinde ortalama doğrulukta sonuçlar üretilmiştir. Ancak, test edilen uygulama sayısının artışı ile birlikte daha doğru sonuçlar üretilebileceği düşünülmektedir.

7. SONUÇ (CONCLUSION)

Kullanıcılar arasında kullanılma oranı hızla artış gösteren Android işletim sistemi, yazılımcıların bu alana yönelmelerine neden olmaktadır. Bu yöneliş kullanıcılar açısından birçok faydalı uygulamanın marketlerde paylaşılmasını beraberinde getirmiştir. Kullanıcılar ihtiyaç duydukları birçok uygulamayı ücretsiz bir şekilde resmi veya 3.parti uygulama marketlerinden indirebilmektedirler. Bu şekilde ücretsiz olarak paylaşılan birçok uygulama beraberinde kötücül uygulamaların dağıtımını da getirmektedir. Böylece içerisinde kötücül ve zararsız uygulamaları barındıran bir dağıtım pratiği ortaya çıkmaktadır.

Uygulamaların "AndroidManifest.xml" dosyasında talep ettikleri izinler ile bunların bazıları kullanmadıkları izinler arasındaki fark gereksiz izinler olarak tanımlanabilir. Bu durumda, uygulamalar için kullanıcıların hassas ve kişisel verilere erişme de kullandıkları yeni bir kötücül yaklaşımı beraberinde getirmektedir. Kullanıcılar uygulama kurulurken, uygulamanın istedikleri izinlere güvenerek uygulamayı mobil araçlarına kurmaktadırlar. Ancak bu şekildeki gereksiz izinler barındıran uygulamalar, daha sonra gelebilecek güncelleme vb. şekildeki kod parçaları ile bu gereksiz izinler kullanılarak kötücül faaliyet yapabilmektedirler.

Bu çalışmada, bu kötücül ve zararsız yazılımların belirlenmesi için farklı bir statik tabanlı metodolojik yaklaşım ortaya konulmuştur. Bu metodolojinin temelinde, uygulamaların "AndroidManifest.xml" dosyalarında talep ettikleri izinler ile bunları ne ölçüde kaynak kodlarında kullandıklarına ilişkin bir değerlendirme yatmaktadır. Bu değerlendirme sonucunda, uygulamanın çalışması için gereksiz izin talebinde bulunulmuş ise, bu uygulamalar riskli kategoride değerlendirilerek, kendileri için bir şüphe değeri belirlenmektedir. Buna göre, ortalamanın üzerindeki değerlerine sahip uygulamalar kötücül uygulama olarak sınıflandırılmaktadır. İncelenen 50 Adet kötücül ve 25 adet zararsız uygulama için; Kötücül uygulamalarda %97.62'lık bir tespit oranı yakalanmıştır. Ancak aynı yaklaşım zararsız kabul edilen uygulamalar için yaklaşık %20'lik bir kötücül tespit oranı vermektedir. Bu da göstermektedir ki, ortaya konulan yeni yaklaşım tek başına yeterli değildir ve dinamik analiz yaklaşımı ile desteklenerek birlikte kullanılmalıdır. Böylece daha etkili ve doğru sonuçlar üretilebilecektir.

REFERANSLAR (REFERENCES)

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

[2] Leavitt N., "Mobile phones: the next frontier for hackers?", ***IEEE Xplore:Computer***, 38: 20-23, (2005).

[3] Shih, D.H., Lin, B., Chiang, H.S., Shih, M.H., "Security aspects of mobile phone virus: a critical survey", ***Industrial Management & Data Systems***, 108: 478-494, (2008).

[4] Xiaoyan Z., Juan F., Xiujuan W., "Android malware detection based on permissions", ***Information and Communications Technologies (ICT 2014), 2014 International Conference on***, Nanjing, 1-5, (2014).

[5] Geneiatakis D., Fovino I. N. , Kounelis I. ve Stirparo P., "A Permission verification approach for Android mobile applications", ***Computer & Security***, 49: 192-205, (2015).

[6] Su M.Y., Chang W.C., "Permission-based Malware Detection Mechanisms for Smart Phones", ***Information Networking(OCOIN) International Conference***, Phuket, 449-452, (2014).

[7] Bartel A., Klein J., Le Traon Y., Monperrus M., "Automatically securing permission-based software by reducing the attack surface: an application to Android", ***ASE 2012 Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering***, New York, 274-277, (2012).

[8] Felt A.P., Chin E., Hanna S., Song D., Wagner D., **"**Android permissions demystified", ***CCS '11 Proceedings of the 18th ACM conference on Computer and communications security***, New York, 627-638, (2011).

[9] Rosen S., Qian Z., Mao Z.M., **"**AppProfiler: a flexible method of exposing privacy-related behavior in android applications to end users", ***CODASPY '13 Proceedings of the third ACM conference on Data and application security and privacy***, New York, 221-232, (2013).

[10] Enck W., Gilbert P., Chun B.G., Cox L.P., Jung J., McDaniel P., Sheth A.N., "**Appsplayground**: an information-flow tracking system for realtime privacy monitoring on smartphones"***, OSDI'10 Proceedings of the 9th USENIX conference on Operating systems design and implementation***, Berkeley, 393-407, (2010).

[11] Berthome P., Fecherolle T., Guilloteau N., Lalande J.F., "Repackaging Android Applications for Auditing Access to Private Data", ***7th International Conference on Availability, Reliability and Security. IEEE Computer Society***, Prague, 388-396, (2012).

[12] Rastogi V, Chen Y, Enck W., "Appsplayground: automatic security analysis of smartphone applications", ***3rd ACM Conference on Data and Application Security and Privacy***, NewYork, 209-220, (2013).

[13] Schreckling D, Kstler J, Schaff M., "Information Security Technical Report. Kynoid: real-time enforcement of fine-grained, userdefined, and data-centric security policies for android", ***6th IFIP WG 11.2 international conference on Information Security Theory and Practice: security, privacy and trust in computing systems and ambient intelligent ecosystems,*** Berlin, 208-223, (2012).

[14] Kodeswaran P, Nandakumar V, Kapoor S, Kamaraju P, Joshi A, Mukherjea S., "Securing enterprise data on smartphones using run time information flow control", ***13th International Conference on Mobile Data Management. IEEE Computer Society***, Bengaluru, Karnataka, 300-305, (2012).

[15] Feth D, Pretschner A., "Flexible data-driven security for android.", 2012 IEEE Sixth ***International Conference on Software Security and Reliability IEEE Computer Society***, Washington, 41-50, (2012).

[16] Beresford AR, Rice A, Skehin N, Sohan R., "Mockdroid: trading privacy for application functionality on smartphones", ***12th Workshop on Mobile Computing Systems and Applications,*** NewYork, 49-54, (2011).

[17] Xiao X, Tillmann N, Fahndrich M, De Halleux J, Moskal M., "Useraware privacy control via extended static-information-flow analysis", ***27th IEEE/ACM International Conference on Automated Software Engineering***, NewYork, 80-89, (2012).

[18] Gibler C., Crussell J., Erickson J., Chen H., "AndroidLeaks: automatically detecting potential privacy leaks in android applications on a large scale", ***TRUST'12 Proceedings of the 5th international conference on Trust and Trustworthy Computing***, Berlin, 291-307, (2012).

[19] Rosen S, Qian Z, Mao ZM., "AppProfiler: a flexible method of exposing privacy-related behavior in android applications to end users", ***3rd ACM Conference on Data and Application Security and Privacy,*** NewYork, 221-232, (2013).

[20] Fuchs AP, Chaudhuri A, Foster JS., "Scandroid: automated security certification of android applications", ***Tech Rep***, (2009).

[21] Xing L., Pan X., Wang R., Yuan K., Wang X., "Upgrading your Android, elevating my malware: Privilledge escalation through Mobile OS updating", ***IEEE Symposium on Security and Privacy***, Washington, 393-408, (2014).

[22] Fang Z., Han W., Li Y., "Permission based Android security: Issues and Countermeaures", ***Computer & Security***, 43 :205-218, (2014).

[23]Stirparo P., Kounelis I., "The mobileak project: Forensics methodology for mobile application privacy assessment", ***Internet Technology and Secured Transactions: IEEE***, London, 297-303, (2012).

[24] Orthacker C., Teufl P., Kraxberger S., Lackner G., Gissing M., Marsalek A., Leibetseder J., Prevenhueber O., "Android security permissions- can we

trust them?", ***Security and Privacy in Mobile Information and Communication Systems***, 94: 40-51, (2011).

[25] Bartel A, Klein J, Le Traon Y, Monperrus M., "Dexpler: converting Android dalvik bytecode to jimple for static analysis with soot", ***ACM SIGPLAN International Workshop on State of the Art in Java Program analysis***, New York, 27-38, (2012).

[26] http://user.informatik.uni-goettingen.de/~darp/drebin/

[27]http://knowyourmobile.com/devices/android-marshmallow/23415/android-marshmallow-review-features-material-design

[28] RR Maier D., Protsenko M., Müller T., "A game of Droid and Mouse: The threat of split-personality malware on Andoid", ***Computer&Security,*** 1-14, (2015).

[29] Suarez-Tangil, G., Tapiador, J.E., Peris-L., "DENDROID: A text mining approach to analyzing and classifying code structures in Android malware families", ***Expert Systems with Applications,*** 1104-1117, (2014).

[30] Yerima, S.Y., Sezer, S., McWilliams, G., "Anaylsis of Bayesian classifcation-based approaches for Android malware detection", ***IET Information Security,*** 25-36, (2014).

[31] Liang, S., Du, X., "Permission-Combination-based Scheme for Android Mobile Malware Detection", ***2014 IEEE International Conference on Communications,*** Sydney, 2301-2306, (2014).

[32] Yerima, S.Y., Sezer, S., Muttik, I., "Android Malware Detection Using Parallel Machine Learning Classifiers", ***2014 18th International Conference on Next Generation Mobile Applications, Services and Technologies,*** Oxford, 37-42, (2014)

[33] Yerima, S.Y., Sezer, S., Muttik, I., "A New Android Malware Detection Approach Using Bayesian Classification", ***2013 IEEE 27th International Conference on Advanced Information Network and Applications,*** Barcelona, 121-128, (2013)

[34] Liu, X., Liu, J., "A Two-layerd Permission-based Android Malware Detection Scheme", ***2nd IEEE International Conference on Mobile Cloud Computing, Services and Engineering,*** Oxford, 142-148, (2014)

[35]] Liu, W., "Multiple classifier system based android malware detection", ***Internation Conference on Machine Learning and Cybernetics,*** Tianjin, 57-62, (2013)

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

[37] Shen, T., Zhongyang, Y., Xin, Z., "Detect Android Malware Variants using Component Based Topology Graph", ***IEEE 13th International Conference on Trust, Security and Privacy in Computing and Communications,*** Beijing, 406-413, (2014)

[38] Kabakus, A.T., Dogru, I.A., Aydın, C., "APK Auditor: Permission-based Android Malware Detection Systems", ***Digital Investigation,*** 1-14**,** (2015).

[39] Yılmaz, E., Koğar H., "Uç Değerle Baş Etmede Kullanılan Farklı Tekniklerin Bazı İstatistiksel Analiz Sonuçları Üzerindeki Etkisi", Journal of Education, 61-67, (2015).