Mehran University Research Journal of Engineering and Technology
Vol. 40, No. 4, 847 - 859, October 2021
p-ISSN: 0254-7821, e-ISSN: 2413-7219
DOI: https://doi.org/10.22581/muet1982.2104.14

Detection and Prevention of Malware in Android Operating System

Kashif Ali Dahri[1a], Muhammad Saleem Vighio[2], Baqar Ali Zardari[1b]

RECEIVED ON 05.05.2019, ACCEPTED ON 03.03.2020

ABSTRACT

The Internet is not safe anymore, malware can be discovered anywhere on the Internet. The risk of malware has increased also due to the increasing popularity and use of Smartphones and their underlying cost-free applications. With its great market share, the Android operating system has become a prime target for malware developers. When an Android phone is injected with a malware, it may result in compromising the privacy of the user by stealing sensitive and private information like contacts, ids, passwords, photos, call records, and so on. Compared to any other Android-based application category, games are the most preferred zone for attackers, due to the high interest of users in game applications. When an end user downloads a game, which is injected with malicious code, user data is infected without bringing in the knowledge of the user. Though, there still are not sufficient protection mechanisms or guidelines stated for end user against Android malware, this study offers a novel approach to detect Android malware in order to ensure the safe usage of Android applications. The advantage of this approach is its ability to utilize Android manifest files for the detection of malware. The availability of manifest file in every Android application makes this approach applicable to all Android applications. It can also be considered as a lightweight method for malware detection, and its efficiency is experimentally confirmed by testing and comparing the results of 50 Android games samples. Experiments are carried out using the Android Package Kit (APK) tools, and based on the experiments, different kinds of malware identification and prevention guidelines have been proposed for the safe and secure usage of the Android operating system.

Keywords: Android, Malware, Game Applications, APK Tools, Manifest Files, Prevention Techniques

1. INTRODUCTION

Among recent technological innovations, Android operating system has gained much popularity because of its extensive use in Smartphone devices. Various types of applications like Google play store and other third-party applications give much freedom to Smartphone users to download and install applications of their interest. However, when it comes to Android's security, it can be considered as the most attacked platform due to its flexibility to allow its users to download and install applications which may be infected by viruses or malicious data [1-3]. The security breach of Android phone can result in stealing sensitive and private data of users which can have disastrous consequences [4, 5]. This study focuses on investigating Android malware attacks to comprehend how malware programs misuse the system's weaknesses and take advantages of loopholes which occur due to end user's lack of knowledge regarding Android applications. While earlier work to these specific issues is engrossed on developing different kinds of software approaches

[1] Department of Information Technology, Quaid-e-Awam University of Engineering, Science and Technology, Nawabshah, Sindh, Pakistan. Email: [a]Kashif10it@gmail.com (Corresponding Author), [b]bazardari34@quest.edu.pk
[2] Department of Computer Science, Quaid-e-Awam University of Engineering and Technology, Nawabshah, Sindh, Pakistan. Email: saleem.vighio@quest.edu.pk

to identify and stopover Android malware; this study focuses on looking for specific differences and patterns within Android malicious and benign applications (particularly game applications) by analyzing their requested permissions and intents, which actually give malware entrance to private resources inside a device. For this study, we take 50 game applications obtained from Google play store [6] and Virus sign [7] between the timeline of 2017-2018. Out of these 50 game applications, 25 are benign and 25 are real malwares. Malicious applications are studied to look for specific patterns inside their architecture, defined intents, and permissions usage. Based on the found patterns, the sets of permissions and intent filters (Actions) which are used by malware applications are consolidated and identified. Classifying permissions and intents that distinguish malware is the first stage of this research, after that, malware prevention guidelines are deliberated to prevent the malware attacks in the Android-based game applications.

The remaining of the paper is structured as follows: Section 2 gives a brief description of Android malware, games applications and manifest files. Section 3 presents APK tools used for decompiling game applications in order to extract manifest files, and to analyze game permissions and intent filters. Section 4 provides a comparative analysis of different malware detection approaches. Section 5 presents steps followed to carry on this study. Section 6 discusses results and gives comparative analysis of permissions requested by malicious and benign games. Following to that, section 7 presents guidelines to access and use game applications from users' security point of view. Finally, section 8 gives the conclusion of the work.

2. ANDROID MALWARE, GAMES, AND MANIFEST FILES

The Android is a Linux based working framework which is intended for touch screen Smartphones, and tablet PCs. It is an open source technology that enables the software to be altered and distributed freely by developers, device manufacturers, gadget makers, and wireless carriers. Android was uncovered in 2007 alongside the establishment of the Open Handset Alliance (OHA) which was made by the Google [8]. Because of the extensive use of Smartphone devices, Android operating system has gained much popularity over the past few years. Not only that, but different application developers have also attracted Smartphone users to access and use their applications to the best of their interests. This has also provided a strong opportunity to those who have malign intentions to steal users' sensitive and private data using their own developed apps [9]. Therefore, it is very much important to understand the severity of malware attacks and to take safety measures against them. Following subsection briefly describe Android malware, malware game applications, and manifest files.

2.1 Android Malware

Malware is any kind of truculent, intrusive or vexing program code which intends to utilize a contrivance without the owners' permission. Malicious projects keep an eye on users' behavior and trade-off their protection. Malware are notoriously difficult to combat as they appear and spread quickly [10-12].

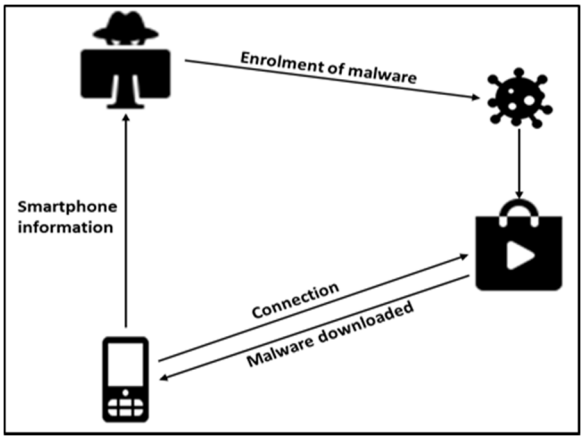

Fig. 1: Malware Attack Scenario, adapted from [11]

Usually, an attacker registers a loaded altered malicious application, which is available free of charge at the black market, and puts up the file APK (Android Package Kit) onto the famous Blogs and Social networking sites to make it possible for many people to install the altered malicious application (see, Fig. 1). As soon as users download that malicious application, their phone gets infected by the malware, and they lose control of their device [13].

2.2 Malware and Game Applications

Compared to any other category, game applications are the most preferred zone for attackers. This is due to the users' high interest in game applications. Renowned Check-Point-Blog manifests a big campaign of malware on Google play, the name of malware was Judy, which is an adware with the capability of auto click. The malware utilizes contaminated devices to create a lot of deceitful taps on promotions, and producing income for the culprits behind it. The malware application achieved a shocking spread of about 18.5 million downloads [14].

2.3 Android Manifest

The manifest file takes the form of "AndroidManifest.xml" and is included in all Android applications. The manifest file contains the essential information about Android applications, such as name and version number of an application, API level, and requested permissions (see, Fig. 2). The manifest file format is alike in both malware and benign applications, though, there are certain dissimilarities in the characteristics of numerous information items [15]. For our study, we extract two items from Android manifest files for all 50 game applications. These items include: permissions and intent filters.

- ***Permissions:*** Android applications include security features called permissions. The main purpose of the permissions is to ensure the security of an Android client. Every application needs to unequivocally ask for a few permissions from the client at the time of installation to achieve specific tasks on the device, for e.g., to get access user's sensitive information like SMS (Short Message Service), phone numbers, photos and system's features like Internet and camera [15, 16].

- ***Intent Filters:*** The intent is a notifying object which can be used to ask for an action from another application part. The <Intent–Filter> component must contain at least one <action> component. On the off chance that there are no <action> component in an intent filter, the filter doesn't acknowledge any intent items [17].

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    android:versionCode="1" android:versionName="1.0"
    package="com.dwwang.BlackJack">
    <uses-sdk android:minSdkVersion="9" />
    <uses-feature android:glEsVersion="20000" />
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
    <uses-permission android:name="android.permission.READ_PHONE_STATE" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
    <uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
    <uses-permission android:name="android.permission.VIBRATE" />
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <action android:name="android.intent.action.USER_PRESENT"/>
            </intent-filter>
</manifest>
```

Fig. 2: Structure of Andriod Manifest File

3. APK TOOLS

Android operating system uses APK file format for installation and distribution of its packages. APK tools refer to a group of tools used to edit, view, or alter Android applications. These tools can decode resources to almost original form and reconstruct them after creating some alterations through reverse engineering. This study utilizes two types of APK tools for reverse engineering of Android applications:

i Show Java Pro

Show Java Pro is used to decompile all 25 benign games. It extracts the source code of an Android application (including XML files and image assets) and works directly from an Android device [18].

ii Java APK De-Compiler

Java APK De-Compiler is an online de-compiler used to decompile all 25 malicious games to reduce the risk of being infected while working on the malware programs [19].

4. MALWARE DETECTION APPROACHES

The process of identifying malware can be divided into analysis, detection, classification and ensuing containment of malware. Several detection approaches are employed to identify malware according to their cases and instances, which make it possible to acknowledge the nature and activities of malware and their latest variants. Some of the recognized malware detection approaches are defined below.

4.1 Signature-Based

Signature based approach can be considered as the

most popular option for commercial software vendors to detect malware. It retrieves semantic patterns of known malware and produces an antique signature with the help of those retrieved patterns. If a program's signature matched with the signature of previous known malware, that program will be classified as a malware. Though, signature-based approach works very effectively for the malware which is already known, however, in case of detecting unknown or new malware, it gets failed which is also the main disadvantage of this approach. Furthermore, signature database is also limited; therefore, many samples of malware remain undetermined as they do not match with any signature. In order to overcome this limitation, new malware signatures need to be updated promptly into database as soon as new malware are perceived [20-22]. As an example of the use of signature-based approach, Wu *et al*. [23] proposed SP-MDM (Smartphone malware detection model) which is based on the artificial immune system likewise to the working principle of biologic immune system which guards us from several infections caused by viruses. In accordance of the model, static and dynamic signatures of malware were separately extracted, and grounded on actual valued vector encoding. 20 benign and 20 malicious samples were utilized for testing purposes to ensure the viability of the approach.

4.2 Specification-based

The specification-based approach uses the binding rule-set to identify malware. Under this strategy, a program which violates predefined binding rule-set is classified as malware. The main restraint of this approach is its difficult implementation to precisely specify the behavior of the program or system [20] [24]. Based on defined approach, Dini *et al.* [25] presented PICARD (Probabilistic Contract on Android) a framework for the detection of repackaged applications through probabilistic contracts on Android systems. As per application's contract statement, PICARD exploits the contract to validate that application is not repackaged by malware. The motivation behind the PICARD method is that actions, sequences or misbehaviors of applications even at small scale are not fragmented of the defined contract. Henceforth, that misbehavior will be recognized and ultimately stopped before it is able to damage the device.

4.3 Behavior-based

The behavior-based approach relies on the actions of software to define whether it is malware or not. It consists of many applications and offers essential resources and material required to detect malware on the Android operating system. Every application has its own precise feature and purpose within the system and therefore the alliance of all these tools creates the behavior-based malware detection system [20-21]. Based on this approach, Sheen *et al*. [26] presented MCDF (Multi-feature collaborative decision fusion) method to employ various item sets extracted from a group of benign and malicious applications. After that, an analysis is done on different item sets to find the most discerning set of items for malware detection. Based on the experiments, a comparative analysis of the proposed model with other ensemble learning techniques has also been presented.

The analysis methods for malware detection approaches can be categorized into two main types: static and dynamic.

- **Static Analysis:** liable for examining application's code by reverse engineering to seek out specific signs or patterns related to malwares. Typically, a static analysis can actually be performed without executing the particular application because this form of analysis is responsible for decompression and dissembling of an APK file without even executing it. This technique is rapid and does not require high system resources.
- **Dynamic Analysis:** works by monitoring system calls with the trace tool to analyse the behavior of Android programs. All input traces produced by the user will be collected using the data collecting software as the crowd sourcing and data collector script. In this form of analysis, the user will be responsible for the installation, execution and generation of input data for the Android software to get an output log file related to an application behavior. Because of the necessity for dynamic monitoring within the process of running the program, the degree of automation and period of

time required are comparatively high. Furthermore, it additionally has to ensure detection before the entrance of malicious code within the system to avoid harm to the system. This also means that, dynamic detection technique requires additional resources [20, 27].

In this paper, a behavior-based Android malicious detection approach is proposed by using static analysis. The malware detection method is grounded on reverse engineering. Static analysis is a well explored approach and used in past by several researchers having the alike goal to ours like malware detection with their own methods, selected characteristics, items, or also as a sub process to support their research and other purposes. However, all the methods, characteristics, extracted items and the experimental Android application samples completely differ compared to our research. In this paper, we used static analysis to extract items like permissions and intent filters from Android manifest files, after that, a comparative analysis of specific characteristics of 25 benign and 25 malicious Android applications has also been presented. In order to determine whether the sample to be detected is a malicious or not, we have matched and compared the specific characteristics of the known malicious applications, and based on that, have provided end user guidelines for prevention of malware and safe usage of Android operating system. This method is fast, cost effective, uses very low system resource, and has easy user accessibility as compared to other detection approaches and methods as shown in Table 1 and Table 2.

5. METHODOLOGY

As mentioned earlier, this study focuses on detection and prevention of malware in Android game applications. For this purpose, we have selected 50 game applications acquired from Google play store [6] and Virus sign [7] between the timeline of 2017-2018. As shown in Fig. 3. Methodology of the research is distributed in various phases which are self-explanatory. These phases include: selection of benign and malicious games, scanning and installation of selected games, selection and usage of APK tools, de-compilation, and extraction of items from manifest files, comparison of results, declaration of hazardous permissions and intent filters, and design and demonstration of guidelines for the end users.

Table 1: Comparison of Different Malware Detection Approaches

S/No	Detection Approach		Advantages	Disadvantages
1	Signature-Based		1) Fast 2) Works well against the technique of attaching a worm to normal traffic	1) Only detect known malware 2) Less effective in identifying lethal exercises because of the quickly changing nature of portable malware 3) Malware obfuscation during the program's run-time 4) Requires frequent updates
2	Specification-based		1) Good in coping with code obfuscation encryption 2) Lower false-positives	1) Very difficult to accurately specify the behaviour of the system or program. 2) Defined specification can vary depending the nature of the malware
3	Behaviour-based	Static	1) Fast 2) Cost effective 3) Resource usage is very low	1) Need frequent updates to modify specific characteristics to improve detection.
		Dynamic	1) Good in coping with code obfuscation encryption 2) Dynamic monitoring of the running program	1) The degree of automation and real-time requirements are comparatively high 2) Requires comparatively more resources to detect malicious code before it gets inside the system

Table 2: Comparative Analysis of proposed and reviewed Malware Detection approaches

Case Study	Detection approach	Data analysis method	Dataset	Advantages	Disadvantages
SP-MDM [23]	Signature-based	Hybrid	Android malware database XVNA	1) Merging static and dynamic malware analysis 2) Utilization of clone and mutation method	1) Only detect known malware 2) Resource usage high because of the usage of both static and dynamic analysis 3) No comparison with any other classification approach 4) Testing dataset quantity is low only 40 Android application collected
PICARD [25]	Specification-based	Hybrid	Collected from Several websites	1) Merging static and dynamic malware analysis 2) Applications checked on run time	1) Only repackaged application can be detected on Android systems 2) Resource usage high because of the usage of both static and dynamic analysis 3) Contract presented to perceive behaviour of applications need to updated as malware and Android operating system evolves
MCDF [26]	Behaviour-based	Static	Google play and android Malware services	1) Dropping concepts for increasing feature assortment 2) Usage of multi feature attributes	1) Run time overhead 2) Excessive complexity 3) Feature sets updates can be required in future due to the evolution of malware and Android operating system
Proposed Method (Detection of malware via Android Manifest)	Behaviour-based	Static	Google play and virus sign	1) Fast 2) Low cost 3) Low resource usage 4) End-user accessibility 5) Malware samples are new (2017-18)	1) Extracted items can be required to change or update in future as malware and Android operating system evolve to ensure this method's effectiveness

Table 3: Collected Benign Game Applications

S/No	Game	Version
1	100 Doors Remix	1.2.0
2	Balance 3D	2.5.8
3	Beach Buggy Racing	1.2.16
4	Boney The Fish Arcade	1.08
5	Cat ball Bounce	1.05
6	City Racing 3D	3.3.133
7	Death Pipe	2.2.1
8	Dodger	1.3
9	Don't be Hunted	1.0.0
10	Fighter II	1.4
11	Free Fall	1.2
12	Froggy Jump	1.66
13	Iron Ball Fall down	2.0
14	Kitten Jump	3.0
15	Racers VS Cops	1.27
16	Racing Fever	1.5.18
17	Racing Moto	1.2.12
18	Raging Thunder	1.0.17
19	Rolling Maze	1.0.0.0
20	Scoops	3.0
21	Sonic Dash 2	1.7.8
22	Teeter Pro	1.9.3
23	Traffic Racer	2.4
24	Tunnel Rush	1.11
25	Zombie Highway	1.4.3

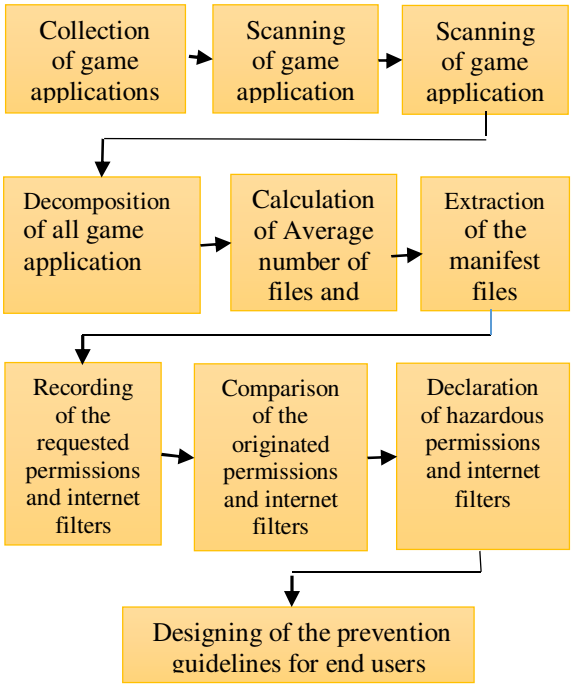

Fig. 3: Methodology

The Table 3 and Table 4 show 50 Android games which have been selected for this study. Out of these 50 games, 25 are benign and 25 are malicious games. Benign games are collected from Google play store [6], whereas, malicious games have been collected from Virus sign database [7].

All 50 game applications were scanned on Virus total [28], before installation on Smartphone. Fig. 4 and Fig. 5 show the scanned results of two game applications out of all 50 games samples. After the installation, all applications are decompiled to calculate their average number of files and size as shown in Table 5 and Table 6 below. The manifest files are extracted in order to analyze the permissions and intent filters. The analysis of permissions is discussed in the sections to follow.

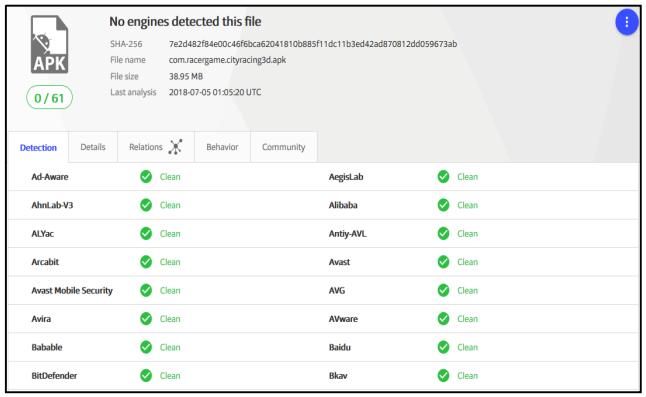

Fig. 4: Scan Results of Benign Games Samples

Table 4: Collected Malware Game Applications		
S/No	Game	Version
1	3D Archery	1.0.5
2	Baby Puzzle Fun	141030
3	Black Jack	1.0
4	Bubble Shooter	1.0307
5	Candy Crush Jelly Saga	4.05
6	Crazy Tank	1.0
7	Demonic Runner	2.0
8	Dragon Hunter	1.0.1
9	Friendly Cars	1.0
10	Fruit Pair Up	2.2.6
11	Godus	0.0.29
12	Maze Origin	1.0.17
13	Mi Zhuan	2.2.8
14	Mobi Army	2.3.1
15	Montezuma 2	1.2.25
16	New Gold Miner	1.0.45
17	Red Alert 2	1.0.8
18	SAO MD	1.10.0
19	Shaky Tower	1.141
20	Snail Battles	1.0.3
21	Sonic 3	4.0.3
22	Spider-Man	1.1.9
23	Stick Moto Race	1.22
24	Touch grind BMX	1.22
25	World Cup Penalty Shootout	1.0.15

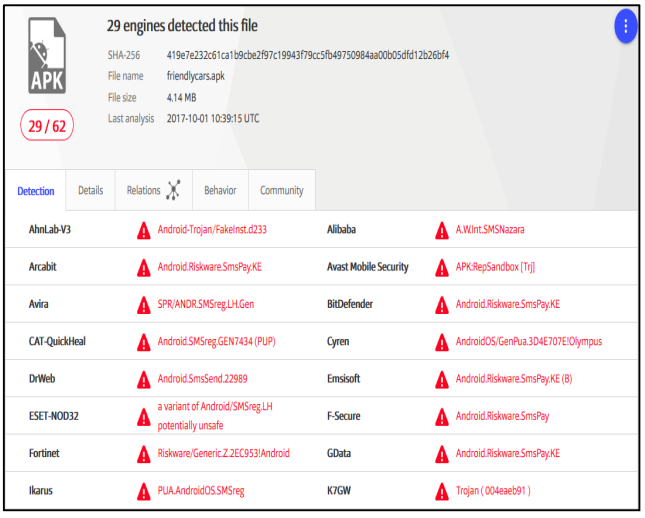

Fig. 5: Scan Results of Malware Games Samples

From Table 5 and 6, it can be clearly observed that the cost of decompiling whole APKs is much higher as compared to manifest files. Furthermore, working on manifest files is a very lightweight process for malware detection.

Table 5: Average file size in KBs		
Game Applications	Decompiled APKs	Manifest File
25 Benign Applications	10880	5
25 Malware Applications	21480	11

Table 6: Average Number of Files		
Game Applications	Decompiled APKs	Manifest File
25 Benign Game Applications	1007	1
25 Malware Game Applications	1567	1

6. RESULTS

All discovered malware were identified and their symptoms or effects on device were recorded after their installation. The total extracted permissions and intent filters of benign and malicious games are recorded and compared. Based on the comparison, a set of dangerous permissions and intent filters within Android games category have been declared which must be avoided by end users to reduce the risk of being infected from malware.

6.1 Malware Identification

As defined below, total five types of malware are discovered and identified during the experiments.

1. **Trojan:** Trojan is a kind of malware which keeps running in the background of a device with the ability to hide itself from the user. It inaudibly waits for instructions from its creator, and these instructions can be anything from hijacking personal information to sending that information at any particular destination. In the Android operating system, it usually hides its existence by not generating an icon and disguising itself with a generic name in the list of applications.

   Symptoms: In most cases, users possibly notice a slowdown in device's performance because Trojan consumes device's resources in the background.

2. **Adware:** This is the most well-known and unequaled prominent Android malware that a Smartphone gets contaminated with. Having adware on the phone can be a baffling thing, as the user experiences constant popups and advertisements on the screen of a Smartphone. Additionally, on the off chance that any of the advertisements is clicked, at that point another noxious program will be downloaded or some undesirable application will be installed on user's Smartphone. Adware produces revenue for its creator by automatically showing online advertisements on the user's device.

   Symptoms: Users start seeing different types of ads, like, uncertain miracle weight loss agendas, offers to get rich within few days, and fake virus warnings that invite users to click on. Also, user might experience frequent opening of new tabs on their own, change in browser home page or even a pass on to a NSFW (Not safe for work) websites.

3. **SMS Agent:** It is also a kind of Trojan, which spreads itself with the help of SMS, e-mail and other kind of messages circulating on social media or the web. The messages which are being sent may contain pernicious links which when the user on the receiving end clicks get infected by that malware. Google has added cautioning functionality for these types of short code messages in latest versions of the Android. This additional protection helps to warn a user in the shape of a popup notice before sending the SMS or following any kind of links from the text messages.

   Symptoms: Users may experience sudden charges on their mobile phone bill or decrease in their mobile credit. Users may also be notified that they have signed up for multiple premium services or websites which they did not presume.

4. **Riskware:** Riskware is the name specified to valid applications and programs that can cause harm if they are oppressed by malicious users in order to modify, copy, or delete data, and disrupt the device's performance. Riskware is not designed preferably as malicious program, but though, they do have some functions that can be served for malicious purposes.

   Symptoms: Users may encounter device's strange or lagging behavior, unwanted background processes, an increase in spam e-mail etc. Additional symptoms may depend on the severity of an installed program.

5. **PUP (Potentially Unwanted Program):** A PUP is a software that the user might not want clogging up his/her device's system. The name "Potentially Unwanted Program" was specified by McAfee to avoid marking downloadable applications as malware. What marks a PUPs dissimilar from malware is that, the user gives approval to download it. When the user downloads an application from the Internet and avoids to read the user agreement or prescribed instructions with it, he may fail to comprehend what other unwanted programs are being installed with that particular application. In most cases these PUPs are adware or browser hijackers. These unwanted programs either get control of your browser homepage and default search engine, or shows unwanted ads not devising from the websites you are surfing, and usually employ enormous amounts of system assets which are the common reasons of clumpy operating systems.

   Symptoms: Frequent Ads displaying while surfing the web, changing of homepage and installation of unwanted applications, and tool bars without user's knowledge.

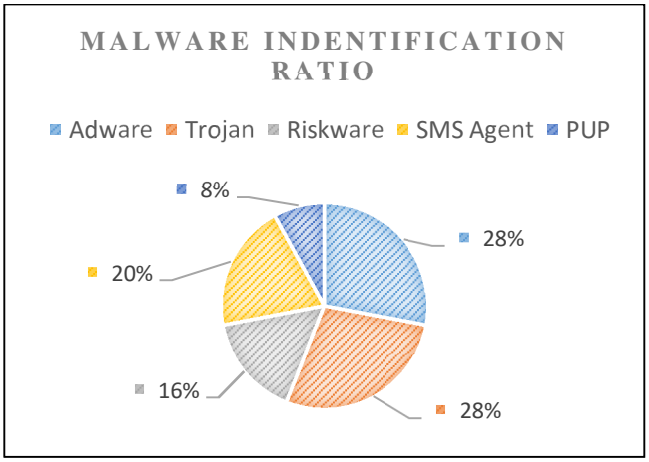

Fig. 6. Discovered Malware Types

Fig. 6 shows the collective ratio of malwares which were discovered and identified in this research. Most of the malicious games were found working as Adware and Trojans with a collective percentage of 56 percent. However, the remaining 44% of game applications are identified to be affected by SMS Agent, Riskware and PUP malwares.

6.2 Results of Benign Games

All 25 benign games were decompiled with the help of Show Java Pro APK tool. Android manifest.xml file was obtained from each decompiled game, after that, each game's permissions and intent filters were extracted from its manifest file and separately calculated and recorded. **Fig. 7** Shows the top 15 permissions requested by benign games.

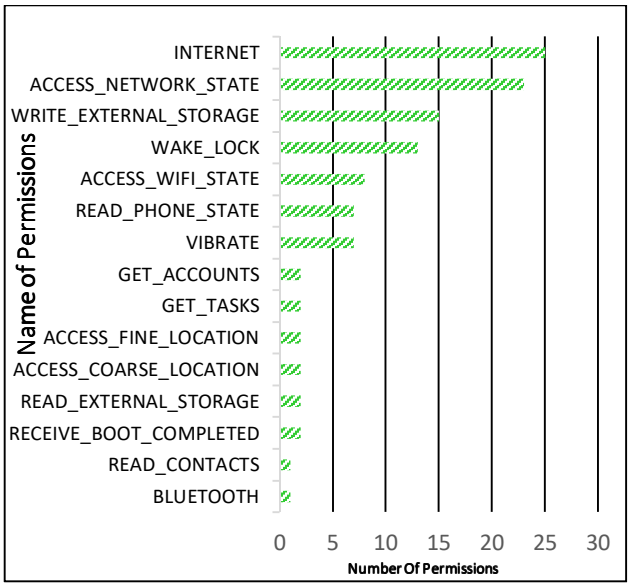

Fig. 7: Permissions Requested by Benign Game Applications

6.3 Results of Malicious Games

All 25 malicious games were decompiled with the help of online APK tool, java APK De-compiler to avoid the security issues in Smartphones. Android manifest.xml file is obtained from each decompiled game, after that, each game's permissions and intent filters were extracted from its manifest file and separately calculated and recorded.

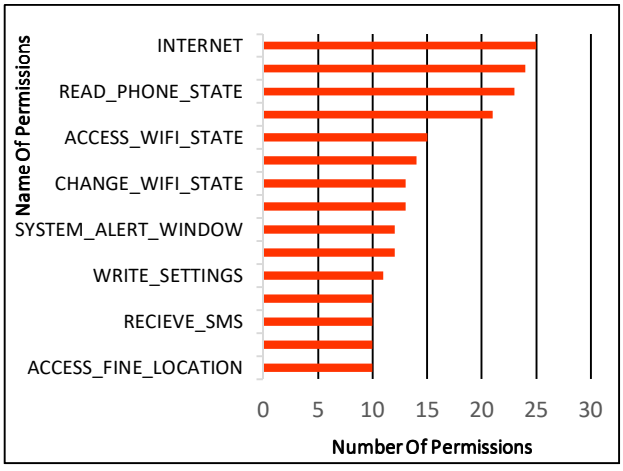

Fig. 8: Permissions Requested By Malware Game Applications

Fig. 8 shows the top 15 permissions requested by malicious games. In contrast to benign games' top 15 permissions in Fig. 8. show some new permissions like System alert window, change network state, write settings, change wi-fi state, send and receive SMS.

6.4 Comparison of Requested Permissions

Fig. 9 Shows the comparison of total permissions requested by all benign and malicious games. It can be seen that, benign games requested 112 permissions, whereas, malicious games requested 285 permissions. Here, we can clearly see that the number of permissions requested by malicious games is a way much higher than the permissions requested by the benign games.

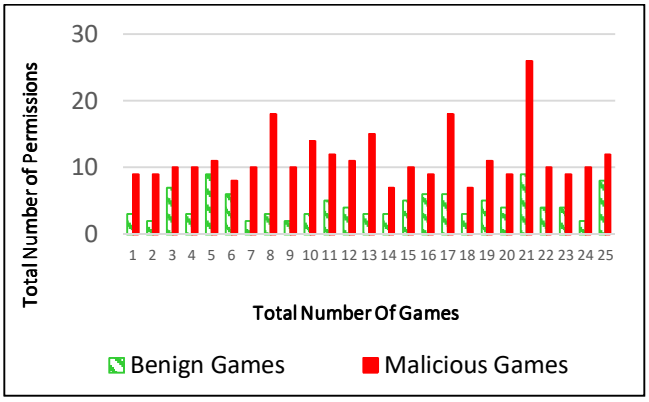

Fig. 9: Comparison of Requested Permissions

6.5 Hazardous Permission

After the comparison of benign and malicious games' permissions, below mentioned permissions are marked as dangerous, and are found and requested by malicious games only. A brief description of these permissions is given below:

a) WRITE_SETTINGS
Permits an application to read or write settings of the system.
b) CHANGE_NETWORK_STATE
Permits an application to change the connectivity state of the network.
c) CHANGE_WIFI_STATE
Permits an application to change the connectivity state of the Wi-Fi.
d) SEND_SMS
Permits an application to send the messages through SMS.
e) RECEIVE_SMS
Permits an application to receive the messages through SMS.
f) SYSTEM_ALERT_WINDOW
Permits an application to draw over/overlay other applications.

6.6 Requested Intents and their Comparison

Fig. 10 Shows the intent filters of benign and malicious games. After permissions, this is our second item which we have extracted from Android manifest. Fig. 10. Shows that the main intent is assigned to all 50 games (benign/malicious). Boot completed is assigned to 3 benign and 5 malicious games, and remaining all other intents are only found in malicious games.

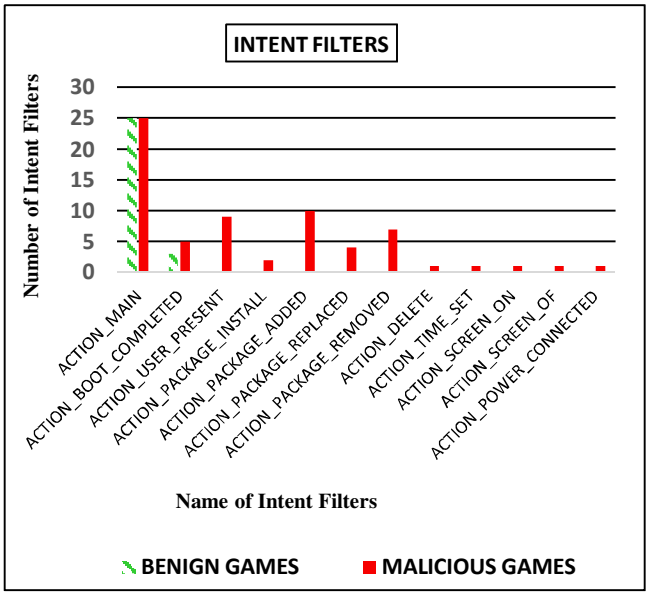

Fig. 10: Comparison of Requested Intent Filters

7. MALWARE PREVENTION GUIDELINES

Malware prevention guidelines are specially designed for end users' safety. The end users' can easily understand and follow the guidelines to avoid of being affected by malware, and to enjoy the safe and malware-free usage of the Android operating system. These guidelines are outlined below.

1. Always download applications which are from certified developers and trustworthy sources like Google play store.
2. Always check permissions before installing any

application on Smartphone. If an application is asking more than what it is intended for, simply don't install it.

3. Keep a decent antivirus application on your device that can identify such malware before it can affect your device. Continuously stay-up with the latest.
4. Never tap on links sent to your Smartphone received through SMS or e-mail. Regardless of whether the e-mail looks genuine, go specifically to the site of beginning and confirm any conceivable updates.
5. Be watchful which applications you give administrator rights to. Administrator rights are intense and can give an application full control of your device.
6. Avoid obscure and unsecured Wi-Fi hotspots and keep your Wi-Fi switched off when not in use.
7. Check for regular updates, if any update is available for your Smartphone, download and install it as soon as possible. Carriers, device manufacturers, and Google are always updating their software with bug fixes, new features, and major improvements to make Smartphones and underlying operating systems more secure.

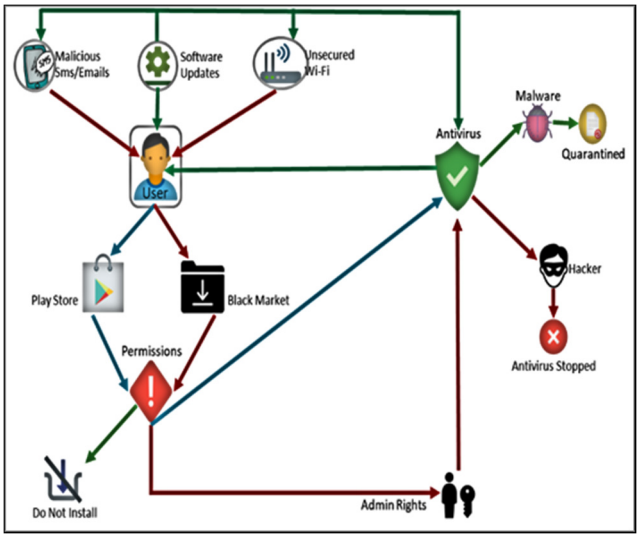

Fig. 11: Graphical illustration of defined prevention techniques

Fig. 11 illustrates defined guidelines and their association with each other. The concepts are further explained below:

a) When a user tries to download any game, he has two options: (i) the user can either go directly to play store and download the game of his choice, or (ii) he can go to the black market by means of any other website or store.

b) No matter which source the downloaded game belongs to, it will request for some permissions at the time of installation. Here, users have a great chance to verify the downloaded game's permissions before installing it. If any game is asking for more than what it is meant for, the user simply should not install it.

c) If the user somehow ignores or does not verify the permissions before installing the games, an active antivirus installed on user's phone can detect and eliminate the malware, or warns the user about the risks of that installed malicious game depending on the privileges given to the particular antivirus.

d) Now back to the permissions, if the user grants administrative rights to an application, that application becomes so powerful that it obtains full control of user's device. In this case, the application can easily shut down any active antivirus and steals user's private data or use the infected device for the benefit of its creator.

e) In cases of malicious SMS or e-mails, if the user responds to those fake SMS or e-mails and visits that particular URL, he will end up inadvertently downloading or installing malware without his knowledge.

f) It is strongly recommended that user should not connect to an unknown or unidentified open wireless network, the information which is sent on an unsecured or without WPA / WPA2 (Wi-Fi Protected Access) can be dangerous. By connecting to an unsecured or open wireless network, you are actually opening your device on that network for someone else.

g) Continues software updates are really necessary and highly recommended for device' system as well as antivirus for fixing bugs, making enhancements, and adding new features for more security.

8. Conclusion

This study presented an approach to detect and avoid Android malware in game applications. With the help of APK tools, experiments were performed on 50 Android game applications: 25 benign and 25 malware games. The main advantage of this approach is its utilization of Android manifest files (already available on every Android application) for the detection of malware. Results show dangerous permissions and intent filters which originated after the comparison of benign and malicious games. To identify the permissions and intent filters related to malicious programs, it was essential to use reverse engineering method to obtain the source code of several malicious and benign game applications. Results confirmed

that malware programs profoundly target device settings, network, and Wi-Fi states, SMS and applications overlapping. Moreover, malware prevention guidelines are also deliberated. It is proved that examining the manifest files only is very economical in terms of resource consumption. Furthermore, proposed method and guidelines can be easily applied by end-users to detect and prevent malware for the safe usage of Android applications.

ACKNOWLEDGMENTS

This paper is based on our Master's thesis defended in the month of November 2018 at Quaid-e-Awam University of Engineering, Science & Technology, Nawabshah.

REFERENCES

1. Tam K., Feizollah A., Anuar N.B., Salleh R., and Cavallaro, L., "The Evolution of Android Malware and Android Analysis Techniques", *ACM Computing Surveys*, Vol. 49, No. 4, pp. 1–41, 2017.
2. Mariam N., Sayed Z., Malik M. S., Fahiem M. A., "An Evaluation Model for Measuring the Usability of Mobile Office Applications through User Interface Design Metrics", *Mehran University Research Journal of Engineering and Technology*, Vol. 38, No. 3, pp. 641–654, 2019.
3. Bhutto A., Hussain D.M.A., "An Android Based Cooperative Knowledge Acquiring Application", *Mehran University Research Journal of Engineering and Technology*, Vol. 37, No. 3, pp. 453–460, 2018
4. Medina L.V.M., Rueda S.J., "Identifying Android malware instructions", *Proceedings of the 6th IEEE Latin-America Conference on Communications (LATINCOM), Cartagena de Indias, Colombia*, pp. 1-7, 5-7 November 2014.
5. Mylonas A., Dritsas S., Tsoumas B., Gritzalis D., "On the Feasibility of Malware Attacks in Smartphone Platforms", *Proceedings of the International Conference on E-Business and Telecommunications*, Vol. 314, pp. 217–232 Springer, Berlin, Heidelberg, 2012.
6. Google Play. [Online]. Available: https://play.google.com/store. [Accessed: 21-Jan-2018].
7. "Android", Virusign. [Online]. Available at: http://www.virusign.com/search.php?q=android. [Accessed: 17-Jul-2018].
8. Wallace J., "Android Apps for Absolute Beginners", An Introduction to Android 7.0 Nougat, pp. 1–15, *Apress*, Berkeley, CA, 2017.
9. Faruki P., Bharmal A., Laxmi V., Ganmoor V., Gaur M.S., Conti M., Rajarajan M., "Android Security: A Survey of Issues, Malware Penetration, and Defenses", *IEEE Communications Surveys and Tutorials*, Vol. 17, No. 2, pp. 998–1022, 2015.
10. Malik V., Malik N., Goyal S. K., "Analysis of Android malwares and their detection techniques", *Proceedings of the 4th IEEE International Conference on Parallel, Distributed and Grid Computing (PDGC)*, pp. 597-602, Solan, Himachal Pradesh, 22-24 December 2016.
11. Willems C., Holz T., Freiling F., "Toward Automated Dynamic Malware Analysis Using CW Sandbox", *IEEE Security and Privacy Magazine*, Vol. 5, No. 2, pp. 32–39, 2007.
12. Yin H., Song D., Egele M., Kruegel C., Kirda E., "Panorama: capturing system-wide information flow for malware detection and analysis", *Proceedings of the 14th ACM Conference on Computer and Communications Security*, pp. 116–127, Alexandria, Virginia, USA, 2007.
13. Park, J.K., Choi S.Y., "Studying Security Weaknesses of Android System", *International Journal of Security and Its Applications*, Vol. 9, No. 3, pp. 7–12, 2015.
14. "The Judy Malware: Possibly the largest malware campaign found on Google Play", Check Point Software Blog, 30-May-2017. [Online]. Available: https://blog.checkpoint.com/2017/05/25/judy-malware-possibly-largest-malware-campaign-found-google-play/. [Accessed: 05-Nov-2017].
15. Sato R., Chiba D., Goto S., "Detecting Android Malware by Analyzing Manifest Files", *Proceedings of the Asia-Pacific Advanced Network*, Vol. 36, p. 23–31, 2013.
16. "Permissions overview: Android Developers," Android Developers. [Online], Available: https://developer.Android.com/guide/topics/permissions/index.html. [Accessed: 10-Mar-2018].
17. "Android Developers," Android Developers. [Online], Available:https://developer.Android.com/guide/topics/

manifest/action-element.html. [Accessed: 15-Mar-2018].

18. "Show Java - A Java Decompiler - Apps on Google Play," Google. [Online]. Available: https://play.google.com/store/apps/details?id=com.njlabs.showjava. [Accessed: 29-Dec-2017].
19. ".JAR and .Class to Java decompiler," Java decompiler online. [Online]. Available: http://www.javadecompilers.com/APK. [Accessed: 24-Dec-2017].
20. Sawle P.D., Gadicha A.B., "Analysis of Malware Detection Techniques in Android", *International Journal of Computer Science and Mobile Computing*, Vol. 3, No. 3, pp. 176–182, 2014.
21. Souri A., Hosseini R., "A state-of-the-art survey of malware detection approaches using data mining techniques", *Human-centric Computing and Information Sciences*, Vol. 8, No. pp. 1–22, 2018.
22. Choudhary M., Kishore B., "HAAMD: Hybrid Analysis for Android Malware Detection", *Proceedings of the IEEE International Conference on Computer Communication and Informatics (ICCCI)*, pp. 1–4, Coimbatore, India, 2018.
23. Wu B., Lu T., Zheng K., Zhang D., Lin X., "Smartphone malware detection model based on artificial immune system". *China Communications*, Vol. 11, No. 13, pp. 86–92, 2014.
24. Mujumdar A., Masiwal G., Meshram D. B., "Analysis of signature-based and behavior-based anti-malware approaches, *International Journal of Advanced Research in Computer Engineering and Technology (IJARCET)*, Vol. 2, No. 6, pp. 2037–2039, 2013.
25. Dini G., Martinelli F., Saracino A., Sgandurra D. "Probabilistic contract compliance for mobile applications", *IEEE International Conference on Availability, Reliability and Security*, pp. 599–606, Regensburg, Germany, 2-6 September 2013.
26. Sheen S., Anitha R., Natarajan V., "Android based malware detection using a multifeature collaborative decision fusion approach", Neurocomputing, Vol. 151, pp. 905–912, 2015.
27. Zhao C., Zheng W., Gong L., Zhang M., Wang C., "Quick and Accurate Android Malware Detection Based on Sensitive APIs", *Proceedings of the IEEE International Conference on Smart Internet of Things (SmartIoT)*, pp. 143–148, Xi'an, China, 17-19 August 2018.
28. Virus Total. [Online]. Available: https://www.virustotal.com/gui/home/upload. [Accessed: 24-Jul-2018].