Jestr

Journal of Engineering Science and Technology Review 13 (4) (2020) 240 - 248

Research Article

JOURNAL OF
Engineering Science and Technology Review

www.jestr.org

The Recent Trends in Malware Evolution, Detection and Analysis for Android Devices

Kakelli Anil Kumar, A. Raman, C. Gupta and R. R. Pillai

School of Computer Science and Engineering, Vellore Institute of Technology, Vellore, Tamil Nadu, India.

Received 21 May 2020; Accepted 31 July 2020

Abstract

In our work, we have analyzed various malware detection techniques and methods for mobile devices and study how these techniques have been changed over the years. The work has investigated viruses and other forms of malware that have affected the mobile market and how malware analysts have proposed various suitable solutions to detect them efficiently. Android-based mobile devices are our prime goal because of high usage, cost-effective, easy to handle and use, and most importantly easily allow third-party applications in their framework. This work also discusses modern threats, the magnitude of their influence, and how they were discovered alongside analysis of said threats.

Keywords: Malware, Android, Evolution of Mobile Malware, Modern Threats, Detection Techniques

1. Introduction

In the modern world, mobile phones have become so popular and it is difficult to imagine the life of mankind without them. The number of mobile users has risen from 4.15 billion in 2015 to a projected 4.78 billion users in 2020. Researchers in Checkpoint have examined different Android malware attacks in the first 6 months of 2019 to find out that hackers are targeting smartphones 50% more than the other years. These findings are highlighted in the Cyber Attack Trends: 2019 Mid-Year Report [1]. The main intent of hackers is malicious advertising of the Android devices, credential thefts, and surveillance of users. Research has found that Malware builders are also available underground to build on the malware. They follow distribution techniques similar to the spreading of normal desktop-based malware applications. This pattern is proved to be effective almost 65% of the time without the user knowing that their device has been affected. Due to the pervasive nature of Android, the threats against it are rising day by day and so there is a need to reliably detect and classify tools. Before understanding in-depth, the different malware attacks, it is necessary to understand the Android operating system [2, 3, 62].

2. Background

With the rise of mobile phones in the 2000s, it took a mere 3 years for malware to begin developing for the same. They started with variations of Cabir, a harmless worm at first, it eventually developed into a family of viruses that targeted the Symbian OS. By 2006, the malware had spread to every platform. Even Android, despite being completely new in 2008, could only evade them for two years. From SMS Fraud to location transmitting Trojans, when 2011 ended, Android had crossed Symbian and J2ME to become the lead platform for malware. This is also the year that saw the first Man in the Middle attacks hit the mobile platform. Despite many advances in both detection techniques and security features since then, Malwares have only grown more complex. At their peak, they could record full conversations and all private exchanges and upload them at will. Just last year in 2019 Kaspersky detected approximately 3,503,952 malicious installation packages, 69,777 new mobile trojans, and 68,362 new mobile ransomware trojans. Android continues to be the leading platform and considering that it owns 70% of the smartphone market, that title is unlikely to go away. Malware creation has become exceedingly profit-driven. According to 2017 statistics, in the Carbon Black report [4], malware authors earn $163,000 per year, which is more than double the average salary for a software developer working on legal projects [5, 6, 54].

3. Android Operating System

Android OS is an open-source, Linux kernel-based operating system that was initially released in 2008. In 2014, the original Dalvik virtual machine that used a Just in Time (JIT) compiler was replaced by Android Runtime, a new virtual machine that used an Ahead of Time compiler. In 2017, a JIT compiler was also added to Android Runtime. These constant changes have taken a toll on malware detection frameworks [7, 45,59]. A mobile antivirus app has a limited amount of permissions granted by the OS for what it can analyze; due to this, mobile virus or malware detection rarely takes place on a device and a signature-based approach is considered preferable. The variation in hardware, versions, and custom additions by individual companies may have paved the path to a wide range of Android products and massive market domination but it has prevented unity within the platform and has left devices fragmented and exposed to many threats [8]. The effect of the changes between the OS versions was studied and it was

*E-mail address: anilsekumar@gmail.com
ISSN: 1791-2377
doi:10.25103/jestr.134.25

concluded that having a universal malware classifier for all the versions reduced accurate detection rates. For higher security and accuracy, the changes must be kept in mind and antiviruses made accordingly [9, 44, 53].

4. Evolution of Mobile Malware

Recently Kaspersky released the statistics for mobile malware for the year 2017-18 [10] as shown in figure 1. From this graph, it is observed that much mobile malware is either Risk Tools or Trojan Droppers. Risk Tools are locally operating programs that are used to conceal files, hide and terminate running processes or remotely access resources. They're not inherently malicious but are used for by cybercriminals for crimes like cryptocurrency mining. Trojan Droppers are used to install or update malicious programs in the victim's device. They usually consist of a bundle of unrelated malware, often including a hoax to distract from the real effects of the bundle. Undetectable trojans are much harder to create than a dropper and hence new droppers are often made to carry and deploy pre-existing trojans. Mobile malware has evolved to vary from traditional malware a lot [11]. Mobile security principles although derived from traditional desktop-based malware have to be adapted to the differences in the two systems. For example, mobile devices have comparatively scarce resources like battery or processing power and varying applications that can limit static analysis. Through a study from Symantec, in the year 2014, an average of 272 new malware was discovered attacking the Android Operating System every month, these numbers have been significantly higher in recent years. Android devices are also increasing in popularity due to which they are becoming a desirable target [12]. We can see as per Fig 1, 80% of the Android malware are forms of Trojans. In March 2020, AV-Test, an Independent IT-Security Institute, conducted tests on various Antiviruses and monitored the performance of each antivirus to detect malware and also its false positive detection rate. All the products were in their updated versions and had their standard protection layers. The objective of this study was to identify the best antiviruses that can detect malicious software on the mobile. It also revealed that the built-in malware protection for android, Google Play Protect was not enough protection and fell short on both protection and usability [57].

5. Android Malware Traits

A mobile worm is a type of mobile malware that makes use of the physical movements of the host to propagate among various mobile networks. This principle is similar to real-life worms but the difference is that applications in Android can easily be installed by either a touch or a swipe, unlike PCs that use a keyboard or mouse. Hence, static analysis of mobiles becomes more difficult because these swipe patterns must also be considered (degree of touch). Also, mobile devices are prone to multiple connections and have other features that traditional desktop systems lack such as SMS, Bluetooth, General Packet Radio Service (GPRS), 3G, and even Multimedia Messaging Services along with access to the web browsers. Tools such as GPS, Camera, and Compass can also be tracked by this malware. A common trait that Android malware carry is that they send background messages to remote numbers most likely like a toll number so that the attackers can easily access information anonymously [13].

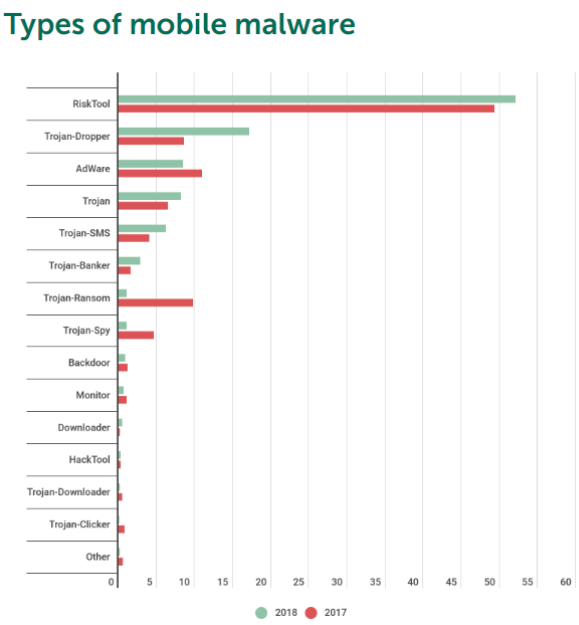

Fig. 1. Mobile Malware Statistics Released by Kaspersky for 2017-18 [10]

This trait was observed when gaming applications like Angry Birds, Luxury Car Parking, Firefighting Simulator (etc.) were revealed to be scams. As a result, Google had to remove 13 such applications from the Play Store. These applications automatically hide and generate a fake store called Game Center which cannot be deleted by the user. They also start sending SMS messages once the game begins. The attackers charge roughly 15 GBP per text which the user needs to pay. Attackers choose applications that look attractive on the outside but have no purpose when a user tries to play the game. This is how malware authors gain profit and have popularized their malware across the whole Android platform. In general, Android malware is often spread via phishing, social engineering, or downloading from the internet [14]. They then launch a Privilege Escalation attack on the mobile device. A privilege escalation attack is a type of attack where a bug or design flaw is exploited in the operating system or software that enables them to gain access to the resources that are normally protected [15]. In the Android OS, the vulnerability is the kernel through which the malware gains root access to the system.

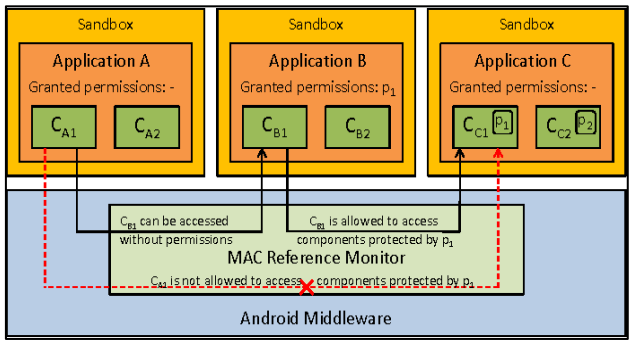

Fig. 2. Applications interact in an Android device

Figure 2 depicts a privilege escalation attack on the Android device. The malware has access to the lower and higher-privileged sections of the OS [16]. This is how the data is sent to the attackers and how spyware/bots can easily be monitored remotely. These bots can also be responsible for

DOS (Denial of Service) attacks by rerouting the traffic of packets to different addresses. They can also overuse the resources and tamper with other files. Given below is the Android OS architecture. The 4 layers- Applications, Application Framework, Libraries, and the Linux Kernel are indicated and the system calls between them are also shown in Fig. 2. Android architecture is modified using a Linux based Kernel and has mainly 4 layers as shown above. The Java written applications are run in isolation. After 2019, since Android's latest version is 10.0, and Ahead of Time (ART), the compiler-based compilation is used for compilation in each of their virtual machines. The hardware consists of an ARM processor (Advanced RISC Machine) that is integrated with a memory chip and other different processors along with features such as GPS and Bluetooth. To access the device each of the applications must be granted permissions by the Android Permission System when they are installed. This is the general agreement that users see when they install any new application. Once the application is installed, the permissions are granted by the kernel and they can interact with each other through API calls [17]. The main components of Android are the content providers and the receiving broadcasts as shown in figure 3. Content providers are responsible for providing access control mechanisms for the process by grouping them into structured sets that are enforced by security mechanisms [18, 61]. Broadcasting of services occurs in the background and the user's activities are the most visible component that handles the user's actions (e.g. Touch and activation of applications).

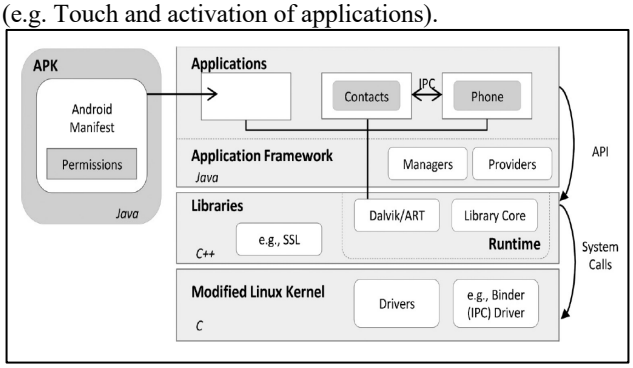

Fig. 3. Android Operating System Architecture

6. Android Malware Statistics

The most recent mobile malware that was popular in 2019 is given below. According to Kaspersky reports [19], the major malware that affected 2019 was potentially unwanted software like Adware and Risk Tool. For example, xHelper is a popular malware that cannot be removed even if the Android phone is the factory reset. It affected around 45,000 Android phones in October 2019. This malware used encryption in its full implementation and because of this it could not be traced by antiviruses that use behavioral analysis to locate malware from figure 4. xHelper evolved from displaying ads for monetary benefits to installing other malware into the device and is considered by experts to be a work in progress. xHelper infecting an Android phone is shown below. Other malware that was dangerous in 2019 was Trojan Droppers and Bankers [20]. For example, Trojan-Dropper.AndroidOS.Necro.n, Trojan.AndroidOS.Boogr.gsh and Trojan.AndroidOS.Hiddapp.ch.

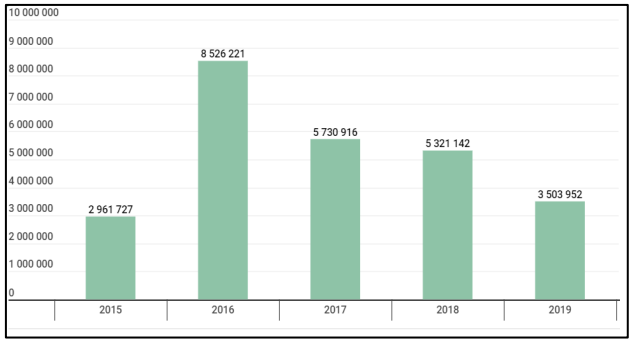

Fig 4. Variations of Malicious Downloads by Android users

7. Malware Detection Techniques

The research on security in mobile environments builds on decades of traditional malware research, however, the need to optimize has led to a focus on developing more targeted, robust, and effective ways to classify malware in smart devices [21]. One of the better-known methods is the use of antivirus products. However, since they are limited by smartphone permissions like any other application, they are considered to be not as effective. Malware detection in antiviruses is usually signature-based [22]. A malware signature is a sequence of bytes to identifies the malware's presence, i.e. it exists in an infected file and doesn't in uninfected ones [23]. Obfuscation is used often by malware developers to escape this analysis and so although still called signatures, antivirus entries in their databases to classify malware have become far more complex and advanced. P. Vinod et Al [24] proposed an Android framework for better user trust of mobile users. The proposed system has used machine learning to investigate the system calls used mobile malware of Andriod. The experimentation has been conducted on the synthesized system calls and generated area-under-curve of accuracy 99.9% and also received better classifier accuracy [60].

Static analysis is another well-known method of malware detection. It examines the malware's code without executing it and despite the limitations may reveal several paths of execution. Obfuscation can be used here again to limit or remove access to malware codes in mobiles [46, 63]. Android malware also tends to involve plenty of runtime activity that can't be detected via this method. The source code for Android apps is usually inaccessible and so most mobile detection frameworks analyze the bytecode in the application package (APK). McLaughlin et al [25] proposed a novel Android malware detection system that uses a deep convolutional neural network (CNN). The classification [52] of malware is based on static analysis of the raw opcode sequence from the disassembly of the program. Features that indicate which malware is being looked at can be learned by feeding the CNN with raw opcode sequences of known malware. Once the network is fully trained, it's capable of scanning 3000 files per second when executed on the Nvidia GTX 980 GPU. However, the accuracy rate of detections was greatly less than desired on large (5000 samples, 80% accuracy) and very large (20000 samples, 87% accuracy) datasets. Li et al [26] wanted to combat the high rate of malware production with a scalable malware detection approach that can effectively and efficiently identify malware apps [56]. They suggested the use of their self-developed Significant Permission IDentification (SigPID) system. SigPID works by extracting and analyzing android

permissions based on three levels of pruning, leading to the mining of the most significant permissions that can be effective in distinguishing malware from non-malware. It uses machine-learning-based algorithms to classify different families of malware and non-malware. When SigPID (using 22 permissions) was compared to a baseline approach that analyzes all permissions, the results indicated that over 90% accuracy could be acquired in detecting malware, which is very close to the accuracy of the baseline classification. SigPID however was 4 to 32 times faster.

Dynamic Analysis runs the programs in a controlled environment to determine malicious intents. Unlike static, it only shows one path per execution- a better result is achievable by further stimulation. Android apps [55] are highly interactive, dynamic analysis helps trigger intents that would have otherwise remained concealed. Wang et al [27] has proposed a light-weight framework for the identification of the Android-malware. A mobile app is based on a multi-level network to analyze the network traffic at the server end. Machine learning is used for Android malware detection for better accuracy of 97.89%. The results have focused on an in-depth analysis of malware behavior. M. A. El-Zawawy [28] has focused on Andriod apps and their vulnerabilities. NIV is the prominent vulnerability that may try to gain the app permissions for getting the crash reports to information steal. The proposed work has focused on the implementation of the detector for NIV to the Androids APIs. NIVD is efficient and faster and detected many prominent vulnerabilities.

Hybrid Analysis methods combine static and dynamic analysis for more efficient detection to increase code coverage [51]. Static analysis is utilized to discover possible activity paths and assesses danger. Dynamic analysis is then guided through these paths to log and analyze the maliciousness of the result. Yuan et al [29] created an android app that implements an online deep-learning-based android malware detection engine, DroidDetector. It combines elements present in static analysis and dynamic analysis and is capable of applying both techniques to characterize malware. The results produced by the app show that deep learning is a suitable technique for characterizing Android malware especially when training data is available. It is capable of achieving over 96% detection accuracy, which is higher than that of more traditional machine learning techniques.

8. Advances in Malware Detection

Dynamic analysis-based methods were much more experimented on than static ones [47]. Isohora et al [30] has conducted a dynamic analysis on the Android kernel layer using a log collector and the logs produced from the collector. There was also a little bit of machine learning used in the form of Shabtai et al [24] using Andromaly, a system that continuously monitors networks and classifies malware as malicious using machine learning algorithms. Using this dynamic system to monitor the whole mobile while minimizing CPU and memory usage and watching out for spikes in processor usage is very similar to how the current "Windows Defender" functions, showing that this dynamic method was also very slightly experimented on during the early 2010s. Yang et al [28] used dynamic analysis, in the form of behavioral analysis to detect malware using two common methods malware employs to steal money, notification suppression, and hardcoded exfiltration methods. A few years later, Yuan et al [29] created an app called DroidDetector to use both dynamic and static methods and a deep learning algorithm to characterize malware extremely accurately. This app showed that deep learning was a viable machine learning algorithm to be used in future projects. As the years passed, the detection techniques and tools used changed. Research papers became more focused on static analysis as the prime method for identification as it's easier and cheaper to do in mass. McLaughlin et al [25] used a Convolutional Neural Network that was trained on the opcode created by the disassembly of previously identified malware samples. Resulting in a CNN capable of analyzing over 3000 files per second, unfortunately, the lower accuracy result proved that Yuan's deep learning implementation might be a more optimal algorithm for malware detection. This eventually led to Li et al [26] creating their SigPID system that statically analyzes permissions using several pruning levels to optimally and efficiently classify malware. The accuracy this system provided was similar to that of Yuan's DroidDetector but also had the speed of McLaughlin's CNN. At this point, static analysis and machine learning-based algorithms became highly favored for all future papers. Malware authors turned more towards countering static methods using obfuscation techniques. Millar et al [30] created DANdroid, a malware detection model that uses a Discriminative Adversarial Network (DAN), that did this by focusing on being resilient to four popular obfuscation techniques. Qian Han et al [31] created a Feature transformation-based Android Malware detector (FARM), that is aimed at efficiently detecting rooting malware, spyware, and banking trojans. The three feature transformations introduced in this paper are extremely hard to reverse engineer and hence make evasion by malware developers harder. There has also been researching into evolving the earlier methods of dynamic analysis that were considered inefficient. Alzaylaee et al [32] presented a state-based input generation approach called DL-Droid. It enhances code coverage and is accurate up to 97.8% while using dynamic analysis only and if static is added this number increases to 99.6%, outperforming several traditional techniques.

9. Modern Android Threats and Analysis

In today's world, there are many types of Android malware that have created a lot of trouble for Android users. Each of this Android Malware uses different strategies to infect the devices [48, 49, 50].

xHelper:

xHelper is the latest form of Android Malware that has infected more than 45,000 Android devices over the past 6 months [33]. This malware first started showing up in May 2019. Symantec reports that xHelper apps started rising from March 2019. They felt that back then the malware's code was relatively easy to analyze and it was used to visit advertisements. This helped users to trust the app and download it. It was only later that people noticed that the xHelper's code gradually changed. Initially, the application could connect to a C & C server (Command and Control Server) and this was written with the application code itself. But later it was observed that the application is connected to an encrypted payload to evade signature detection. In the code, empty classes were also added which was not present

before. It is believed that each class in the malware represents a mobile service provider. For example, there was a class named "Jio" indicating that this class attacks Jio users alone. So far, this malware has infected users in India, the US, and Russia [34]. xHelper hides once installed and secretly downloads other threats and also displays advertisements. This application is very persistent and this application can reinstall itself even when the user uninstalls it from his device. It hides by not showing up on system launcher. An important thing to notice is that xHelper does not have a regular user interface it is an application component and so it can cause more harm being undercover. A sample of the source code for xHelper is shown in figure 5. Also, xHelper cannot be launched manually, it is launched by external events like when another application gets installed or uninstalled or when the device is charging. In case the application is detected, it restarts its service. It also registers itself as a foreground service and lowers its chances of being destroyed when it's in memory. Once xHelper has gained control of the device, it connects to the attacker's C & C server and waits for the attacker to give his commands. The attacker can either steal the data or prepare for a ransomware attack.

```
<receiver android:name="com.mufc.umbtts.WakeUpReceiver" android:exported="true">
    <intent-filter>
        <action android:name="android.intent.action.USER_PRESENT"/>
        <action android:name="android.intent.action.ACTION_POWER_CONNECTED"/>
        <action android:name="android.intent.action.ACTION_POWER_DISCONNECTED"/>
        <action android:name="android.intent.action.BOOT_COMPLETED"/>
        <action android:name="android.net.conn.CONNECTIVITY_CHANGE"/>
        <action android:name="android.intent.action.PACKAGE_ADDED"/>
        <action android:name="android.intent.action.PACKAGE_REMOVED"/>
        <action android:name="com.mufc.umbtts.BRACTION"/>
    </intent-filter>
</receiver>
```

Fig. 5. Possible events to trigger xHelper

The xHelper malware works in the following manner:

1. Once xHelper is easily installed and connects to an encrypted payload and it adds modules/classes that were not present before. It also hides and starts bombarding the user with various advertisements.
2. Once xHelper does this, it removes itself from the application launcher and connects to the C&C server to connect the attacker and follow their commands. The attacker can then steal data, host another malicious attack, or gain administrator-level access also.
3. Even if the user tries to uninstall xHelper it will not be removed because it does not have a regular user interface and is launched by external events. Examples of such events are the device charging, the device not charging, the device rebooting, another application being installed, an application being removed, or even when the device's connection is changed.

Overall, xHelper is a very dangerous malware that destroyed the phones of almost 33,000 phones in India. xHelper can be stopped by using **SSL certificate** pinning for any communication between the device and the attacker users can use Jio Security which is backed up by Norton Mobile Security to prevent this attack. When this malware was examined using VirusTotal [35] as shown in Fig 6. Anubis Banking Trojan is a malware that was popular in 2018 but its true effect was only shown in 2019 from figures 6, 7, and 8. Anubis was first used for cyberespionage and later was re-coded to become a banking Trojan that combines information theft and ransomware routines. In mid-January 2019, a unique feature of this malware was that this malware will only begin its execution once the motion sensors that detect the Android device are moved.

This strategy effectively avoids this malware to get detected by many sandboxes that use static and dynamic analysis to understand the features of the malware. The Anubis malware follows a series of steps which causes it to be very effective [36, 37].

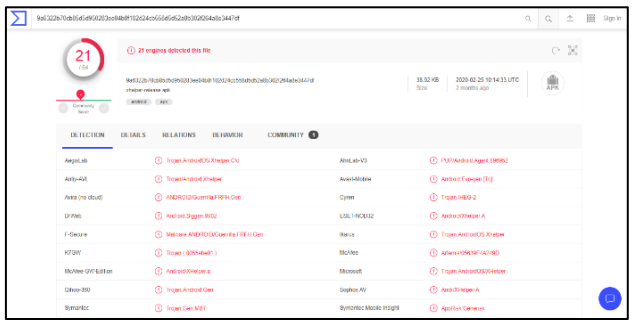

Fig. 6. Detection of xHelper in VirusTotal

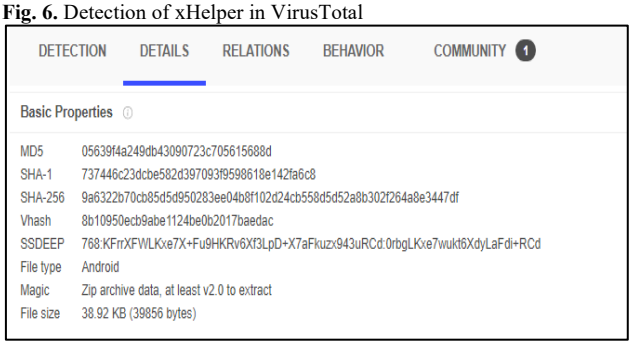

Fig. 7. Details of the xHelper malware

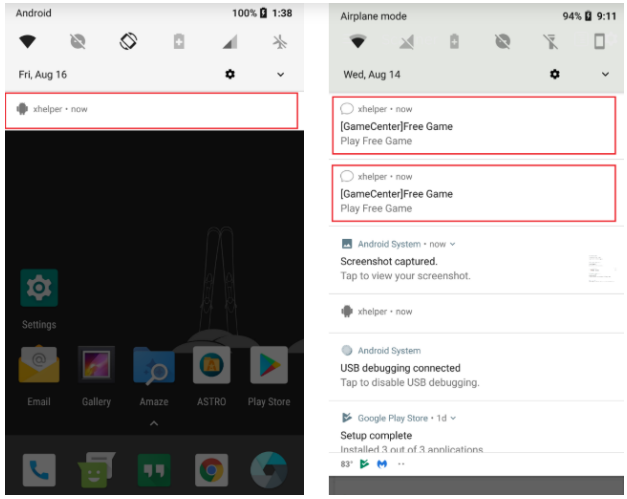

Fig. 8. Attack of xHelper Malware on an Android device

1. The attack starts with a phishing email, that appears to be from a trusted contact where users must download an invoice.
2. When the E-mail link is opened from an Android device a typical APK file (Fattura002873.apk) is downloaded in the background.
3. Once this is done, the normal Google Play Protect screen appears to the user, but this is not the genuine one. This Google Play Protect is accepted, gives access to the application to disable the actual Google Play Protect.
4. Once this is done, Anubis scans the device for any financial applications or any online shopping applications like Amazon or Flipkart.

5. Once the application is chosen, Anubis creates a fake login screen for that application, this causes Anubis to perform various tasks such as capturing screenshots, recording audio, initiating calls, changing the administrator settings, and even opening specific URLs.

Anubis is also a **keylogger** and can be initiated once the attacker has gained full control of the device as shown in figure 9. Anubis can also attack a specific application by monitoring the activity of the targeted application and then the attacker can damage the WebView feature that displays applications on a web page. By doing this, Anubis can **perform phishing attacks** using attack vectors and also send the notification strings in the device to the C & C server as per the figure 10.

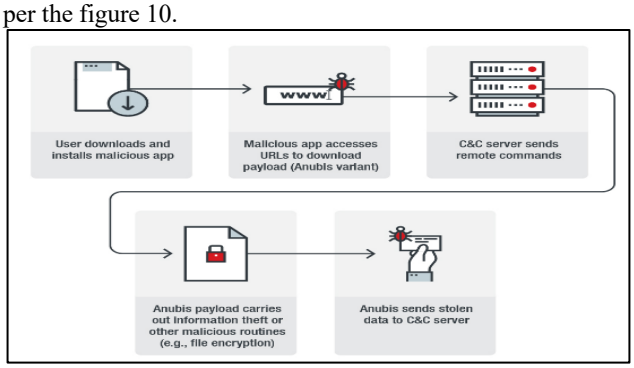

Fig. 9. Block diagram of the Anubis Trojan Backdoor

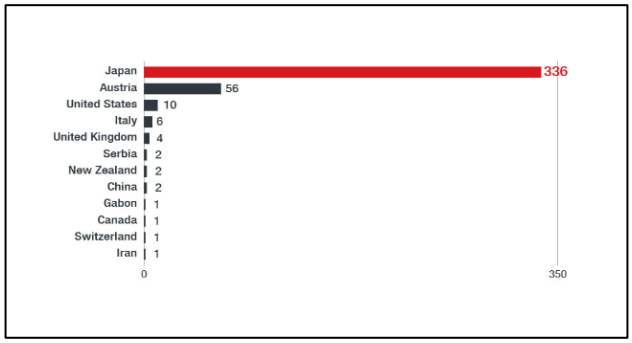

Fig. 10. Number of Victims affected by Anubis

So far Anubis has been present in applications like **CurrencyConvertor, BatterySaverMobi** which when installed can trigger Anubis by the motion of the device [38]. The only solution to Anubis given is that users must limit the installation of applications on their devices and only use applications from trusted developers as shown in figure 11.

Agent Smith Malware is a newly discovered variant of Android malware that replaces real apps with tainted fake versions that flood the devices with ads. So far Agent Smith has only been used to push ads, but respected security professionals agree that it can be used for far more nefarious purposes [39, 40]. There is a multitude of different apps that infect an Android system with the Agent Smith malware, these are usually acquired from unofficial or cracked Android app stores. Checkpoint Software Technologies says that the malware-filled apps were downloaded on more than 25 million devices operating across the US, UK, and India. 15 million of those users are based in India. Unlike most other forms of Android malware, no variant of it currently steals any form of user data or credentials, but it is known to replace genuine versions of popular apps like "Whatsapp", "Facebook", "Flipkart" and "Amazon" [41]. These corrupted versions of the famous apps were created from hackers stealing the source code of those apps and replacing code from the original program to allow for adware and forcing the app to be disconnected from the real app's servers. This prevented security updates from going through and automatically repairing and securing the app [42]. Research has narrowed down the point of origin to a third-party app store known as "9apps.com", owned by China's Alibaba. The malware possesses the ability to hide its icon from the launcher and is known to be capable of impersonating any popular existing apps on a device, making the malware's potential to harm nearly endless. Below are details about how the malware comes to infect a system also shown in figure 12.

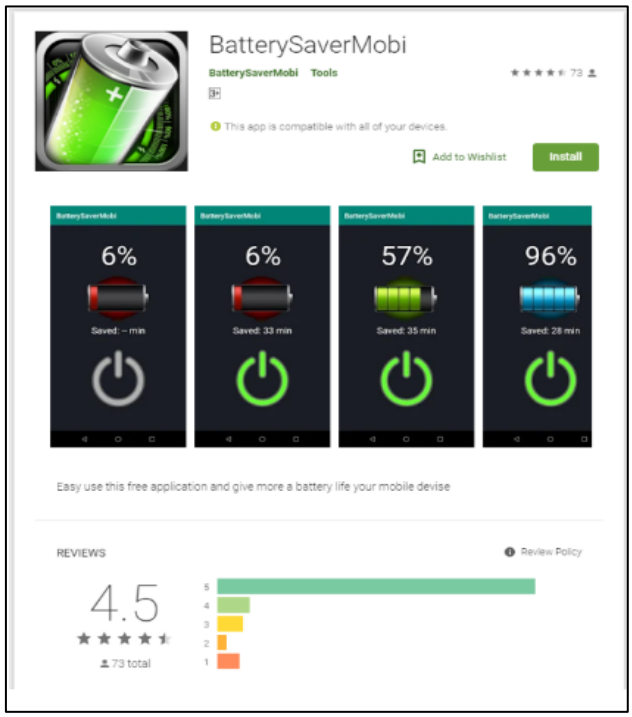

Fig. 11. Fake BatterySaverMobi application

1. *Users download an app from the store, these apps are typically games or adult-themed apps*
2. *The app then silently installs the malware, pretending to be a Google updating tool. No icon appears during this installation, not on the lock screen or the notification bar.*
3. *During the installation, the malicious app scans the device for popular apps such as Opera Browser, Whatsapp, Facebook, Twitter, Amazon, etc.*
4. *The app updates the code of those famous apps to allow for the aggressive monetized ads to be served to the user*
5. *The ads themselves were not found to be harmful, but it is assumed that the hackers gain money per-click, as typical for a pay-per-click system*

There is a possibility that the hackers intend to move to the official google play store, this was evidenced by dormant code found in 11 popular apps on the play store found by Check Point researchers.

Agent Smith has a modular structure and consists of 6 different modules each with its function. The first module is the **loader**, it extracts and runs the **core** module of the malware. The core module communicates with the command and control server, which it then uses to retrieve the predetermined list of popular apps that the device is scanned for. Since the core module looks to the C&C server for a list, the list of apps can be very easily updated by the hacker, which allows for easy upgrading of the malware. The **boot** module is the next component that acts. If any application from the list is found on the device, the malware utilizes the

Janus vulnerability to inject the boot module into the detected application. The next time the infected application is run, the boot module will run the **patch** module which hooks the methods from known ad software development kits to its implementation as shown in figure 13. The initial attack vector used by Agent Smith is the 9Apps market. It uses over 360 dropper variants to maximize profits and is commonly known to infect the same devices repeatedly. There are estimated to be around 2.8 billion infections in total. The majority of the droppers on 9Apps are games, but adult entertainment, media player apps, photo utility apps, and even system utility apps are used. The most popular instance of a dropper is a game called "Kiss Game: Touch Her Heart". The algorithm of malware update installation and detailed statistics on the infection distribution shown in Table 1 and 2 shows that India was the target region of the malware.

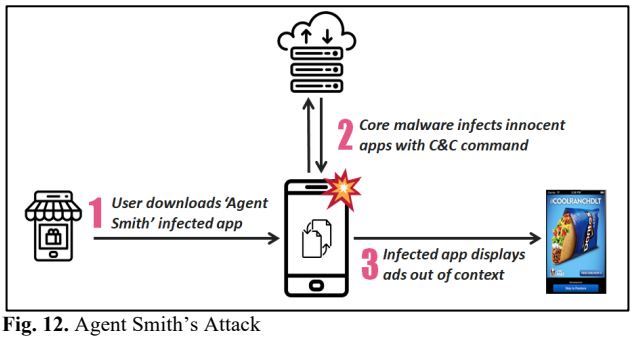

Fig. 12. Agent Smith's Attack

Triada is a modular mobile Trojan that uses the user's root admin access to substitute different system files. It exists on the system's RAM and so cannot be found easily. What is unique about this malware is that smaller Trojan's like Ztorg, Leech, and Gopro if infected in the device can download bigger more dangerous malware, Triada.

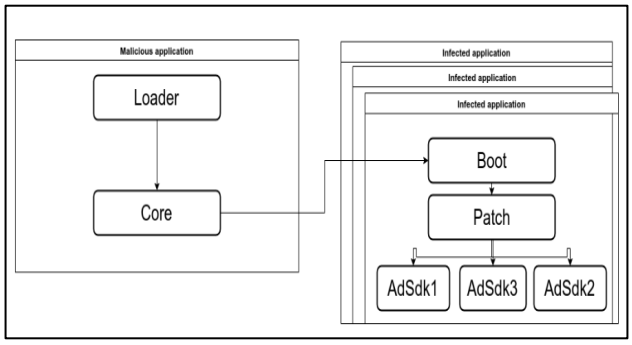

Fig. 13. Modular Structure of Agent Smith

Table 1. Algorithm for the malicious updates

1. *creates a new bundle object*
2. *apk_path, inflect_pkg and fake_name values the bundle object holds are replaced by values given by the malware author*
3. *A new activity is started and its class name is set to "Android"*
4. *a new component is created which is an android package installer.*
5. *a component is a malicious update that is later installed*

Table 2. Devices infected with Agent Smith country-wise

Country	Total Devices	Total Infection Event Count	Avg App Swap	Avg Droppers Per Device	Avg Months Device Remained Infected
India	15,230,123	2,017,873,249	2.6	1.7	2.1
Bangladesh	2,539,913	208,026,886	2.4	1.5	2.2
Pakistan	1,686,216	94,296,907	2.4	1.6	2
Indonesia	572,025	67,685,983	2	1.5	2.2
Nepal	469,274	44,961,341	2.4	1.6	2.4
US	302,852	19,327,093	1.7	1.4	1.8
Nigeria	287,167	21,278,498	2.4	1.3	2.3
Hungary	282,826	7,856,064	1.7	1.3	1.7
Saudi Arabia	245,698	18,616,259	2.3	1.6	1.9
Myanmar	234,338	9,729,572	1.5	1.4	1.9

Triada once installed tries to get information about the device such as the OS version, a model of the device, amount of space in the SD card, and all the applications installed on the device. Similar to the other malware are discussed, it sends all this to a C & C server which is built by the attackers. Kaspersky has detected around 17 C & C servers and 4 different domains used by this malware. Once the data gets analyzed, the C & C server sends a configuration file that consists of the passcode/PIN of the Android device, the list of modules to be installed, the time of contact with the server. After getting these details, Triada installs certain modules and makes sure that they are kept in the short memory, hence the detection of Triada is very difficult. The unique feature of Triada is that other than the above actions, Triada has found a way to modify the [10] **Zygote process** in an Android device from figure 14. The Zygote process of an Android device is the core process in the Android OS that forms the blueprint/template for every Android application. So, if the Zygote gets infected by Triada, the malware literally can control the launch and usage of every application in the Android device.

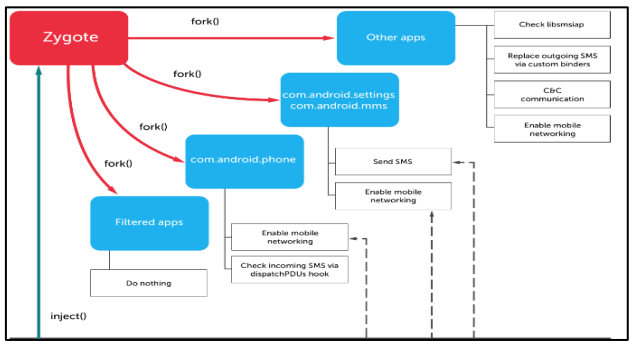

Fig. 14. Normal Zygote Flow Execution for Android apps

Triada also substitutes the different system functions and hides its modules from the currently running applications/processes. Thus, the system does not find any strange processes running and so it doesn't raise the alarm. Triada also **reads the outgoing SMS and filters any incoming ones**. This can be a huge problem because many transactions rely on SMS over an Internet connection because to send an SMS, the Internet is not required. Triada can modify these SMS so that the money is sent to the malware authors and not to the owner of the product. This way Triada steals money either from the users, in case they have not completed their purchase or from the owner of the product in case they have finished the purchase of the product. The way Triada works is that it can affect a lot of users. As it is spread

through smaller Trojans, Triada affected almost every user among 10 Android users during the second half of 2015 as per figure 15. So, the only way to protect from Triada is to constantly update the device in case any patch occurs, install mobile anti-viruses and most of all have a good security solution.

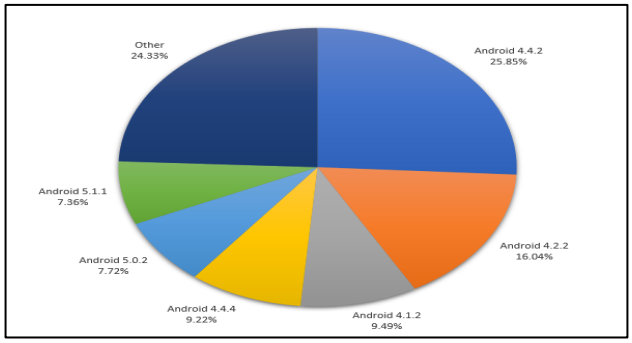

Fig. 15. Triada infected to Android versions

Lotoor is a threat that targets Android devices silently and looks for different vulnerabilities that are present in the device. If any vulnerability is found, Lotoor exploits that and will use that to get administrator privileges. In case Lotoor is successful in doing this, it can [43]:

1. *Collect sensitive data from the device*
2. *Monitor the installation of applications*
3. *Disable the security present in the device*
4. *Alter the settings on the device*

Further, Lotoor can be present in the mobile device as **rageagainstthecage** and **exploid**. To prevent being attacked by Lotoor, all the applications must be updated and install an antivirus for the device.

10. Conclusion and Future Plans

Over the years, malware authors have gotten increasingly craftier, but anti-malware authors have done the same. The cycle used to be set in stone; a new malware is born, it rises to fame, and it is then patched out. This life cycle is the life cycle of all popular malware we have seen so far. However, this article has demonstrated how anti-malware authors have attempted to break this cycle by moving more towards preventative measures, anti-malware systems that can predict and eliminate malware before it even executes, over the lacking cure-based system, eliminating the malware after the damage is done. Some modern threats were analyzed, showing how malware authors are also stepping up their game, creating "silent" malware and even malware capable of wearing disguises. Our review work is continuing to follow the recent advances and updates in the field of malware and their analysis for android and iOS versions of mobile environments.

References

[1] https://research.checkpoint.com/2019/cyber-attack-trends-2019-mid-year-report/
[2] P. Yan and Z. Yan, "A survey on dynamic mobile malware detection," *Software Quality Journal*, (2018).
[3] Q. Zhou, F. Feng, Z. Shen, R. Zhou, M. Y. Hsieh, and K. C. Li, "A novel approach for mobile malware classification and detection in Android systems," *Multimedia Tools and Applications*, (2019).
[4] https://www.carbonblack.com/2017/12/20/carbon-black-2017-threat-report-non-malware-attacks-ransomware-continue-spotlight/
[5] O. Or-Meir, N. Nissim, Y. Elovici, and L. Rokach, "Dynamic malware analysis in the modern era—A state of the art survey," *ACM Computing Surveys*, (2019).
[6] N. K. Gyamfi and E. Owusu, "Survey of Mobile Malware Analysis, Detection Techniques and Tool," in *2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference, IEMCON 2018*, (2019).
[7] P. Bhat and K. Dutta, "A survey on various threats and current state of security in android platform," *ACM Computing Surveys*. (2019).
[8] J. F. Lalande, V. V. T. Tong, M. Leslous, and P. Graux, "Challenges for Reliable and Large Scale Evaluation of Android Malware Analysis," in *Proceedings International Conference on High Performance Computing and Simulation, HPCS*, (2018).
[9] Nguyen-Vu L., Ahn J., Jung S., Android fragmentation in malware detection. Comput. Secur. Vol 87, pp. 9-10. (2019)
[10] https://www.kaspersky.com/blog/triada-trojan/11481/
[11] M. Wazid, S. Zeadally, and A. K. Das, "Mobile Banking: Evolution and Threats: Malware Threats and Security Solutions," *IEEE Consumer Electronics Magazine*, (2019).
[12] T. Chakraborty, F. Pierazzi, and V. S. Subrahmanian, "EC2: Ensemble Clustering and Classification for Predicting Android Malware Families," *IEEE Transactions on Dependable and Secure Computing*, (2020).
[13] G. Shrivastava and P. Kumar, "SensDroid: Analysis for Malicious Activity Risk of Android Application," *Multimedia Tools and Applications*, (2019).
[14] A. A. A. Samra, H. N. Qunoo, F. Al-Rubaie, and H. El-Talli, "A survey of static android malware detection techniques," in *IEEE 7th Palestinian International Conference on Electrical and Computer Engineering*, (2019).
[15] G. Meng, M. Patrick, Y. Xue, Y. Liu, and J. Zhang, "Securing Android App Markets via Modeling and Predicting Malware Spread between Markets," *IEEE Transactions on Information Forensics and Security*, (2019).
[16] K. Basu, P. Krishnamurthy, F. Khorrami, and R. Karri, "A Theoretical Study of Hardware Performance Counters-Based Malware Detection," *IEEE Transactions on Information Forensics and Security*, (2020).
[17] X. Chen *et al.*, "Android HIV: A Study of Repackaging Malware for Evading Machine-Learning Detection," *IEEE Transactions on Information Forensics and Security*, (2020).
[18] R. Taheri, M. Ghahramani, R. Javidan, M. Shojafar, Z. Pooranian, and M. Conti, "Similarity-based Android malware detection using Hamming distance of static binary features," *Future Generation Computer Systems*, (2020).
[19] https://securelist.com/mobile-malware-evolution-2019/96280/
[20] N. I. Aminuddin and Z. Abdullah, "Advances in Computing and Intelligent System Android Trojan Detection Based on Dynamic Analysis," *Advances in Computing and Intelligent System*, (2019).
[21] M. Shahpasand, L. Hamey, Di. Vatsalan, and M. Xue, "Adversarial Attacks on Mobile Malware Detection," in *AI4Mobile 2019 IEEE 1st International Workshop on Artificial Intelligence for Mobile*, pp. 17–20, (2019).
[22] F. Mercaldo and A. Santone, "Deep learning for image-based mobile malware detection," *Journal of Computer Virology and Hacking Techniques*, (2020).
[23] J. Hwang, J. Kim, S. Lee, and K. Kim, "Two-Stage Ransomware Detection Using Dynamic Analysis and Machine Learning Techniques," *Wireless Personal Communications*, (2020).
[24] P. Vinod, A. Zemmari, and M. Conti, "A machine learning based approach to detect malicious android apps using discriminant system calls," *Futue Generation Computing Systems*, (2019).
[25] N. McLaughlin, J. Martinez del Rincon, B. Kang, S. Yerima, P. Miller, S. Sezer, Y. Safaei, E. Trickel, Z. Zhao, A. Doupé, G. J. Ahn, "Deep Android Malware Detection", ACM, 301–308. (2017).

[26] Li J., Sun L., Yan Q., Li Z., Srisa-an W., Ye H., "Significant permission identification for machine-learning-based android malware detection:, IEEE Transactions on Industrial Informatics, 14(7), 3216-3225. (2018).
[27] S. Wang, Z. Chen, Q. Yan, B. Yang, L. Peng, and Z. Jia, "A mobile malware detection method using behavior features in network traffic", Journal of Network and Computer Applications. (2019).
[28] M. A. El-Zawawy, E. Losiouk, and M. Conti, "Do not let Next-Intent Vulnerability be your next nightmare: type system-based approach to detect it in Android apps," International Journal of Information Security, (2020).
[29] Yuan Z., Lu Y., Xue Y., Droiddetector: android malware characterization and detection using deep learning. Tsinghua Science and Technology, 21(1), 114-123. (2016).
[30] S. Millar, N. McLaughlin, J. Martinez del Rincon, P. Miller, Z. Zhao, DANdroid: A Multi-View Discriminative Adversarial Network for Obfuscated Android Malware Detection. CODASPY '20. Association for Computing Machinery, pp353–364. (2020)
[31] Q. Han, V. S. Subrahmanian and Y. Xiong, Android Malware Detection via (Somewhat) Robust Irreversible Feature Transformations, in IEEE Transactions on Information Forensics and Security. (2020).
[32] M. K. Alzaylaee, S. Y. Yerima, S. Sezer, DL-Droid: Deep learning based android malware detection using real devices, Computers & Security, Vol 89, (2020).
[33] https://www.republicworld.com/technology-news/apps/beware-xhelper-malware-infects-45000-devices-over-the-past-6-months.html
[34] https://symantec-enterprise-blogs.security.com/blogs/threat-intelligence/xhelper-android-malware
[35] https://www.virustotal.com/gui/
[36] M. Mehra and D. Pandey, "Event triggered malware: A new challenge to sandboxing," in *12th IEEE International Conference Electronics, Energy, Environment, Communication, Computer, Control: (E3-C3)*, 2016.
[37] https://blog.trendmicro.com/trendlabs-security-intelligence/anubis-android-malware-returns-with-over-17000-samples/
[38] https://www.zdnet.com/article/these-malicious-android-apps-will-only-strike-when-you-move-your-smartphone/
[39] D. Hassan and M. Might, "A Similarity-Based Machine Learning Approach for Detecting Adversarial Android Malware, (2014).
[40] https://blog.avast.com/agent-smith-malware
[41] https://www.cybersecurity-insiders.com/meet-the-agent-smith-malware/
[42] https://www.enigmasoftware.com/lotoor-removal/
[43] https://www.microsoft.com/en-us/wdsi/threats/malware-encyclopedia description?Name=Exploit:Unix/Lotoor
[44] A. Qamar, A. Karim, and V. Chang, "Mobile malware attacks: Review, taxonomy & future directions," *Futur. Gener. Comput. Syst.*, (2019).
[45] E. J. Alqahtani, R. Zagrouba, and A. Almuhaideb, "A survey on android malware detection techniques using machine learning Algorithms," (2019).
[46] K. Tian, D. Yao, B. G. Ryder, G. Tan, and G. Peng, "Detection of Repackaged Android Malware with Code-Heterogeneity Features," (2020).
[47] G. D'Angelo, M. Ficco, and F. Palmieri, "Malware detection in mobile environments based on Autoencoders and API-images," *J. Parallel Distrib. Comput.*, (2020).
[48] A. Mehtab *et al.*, "AdDroid: Rule-Based Machine Learning Framework for Android Malware Analysis," *Mob. Networks Appl.*, (2020).
[49] C. Raghuraman, S. Suresh, S. Shivshankar, and R. Chapaneri, "Static and dynamic malware analysis using machine learning," Advances in Intelligent Systems and Computing, (2020).
[50] M. Odusami, O. Abayomi-Alli, S. Misra, O. Shobayo, R. Damasevicius, and R. Maskeliunas, "Android Malware Detection: A Survey," (2018).
[51] M. Choudhary and B. Kishore, "HAAMD: Hybrid Analysis for Android Malware Detection," In proceeding of International Conference on Computer Communication and Informatics (2018).
[52] Fan, Ming, Jun Liu, Xiapu Luo, Kai Chen, Zhenzhou Tian, Qinghua Zheng, and others, 'Android Malware Familial Classification and Representative Sample Selection via Frequent Subgraph Analysis', *IEEE Transactions on Information Forensics and Security*, (2018).
[53] Feng, Pengbin, Jianfeng Ma, Cong Sun, Xinpeng Xu, and Yuwan Ma, 'A Novel Dynamic Android Malware Detection System with Ensemble Learning', *IEEE Access*, (2018).
[54] Khanmohammadi, Kobra, Neda Ebrahimi, Abdelwahab Hamou-Lhadj, and Raphaël Khoury, 'Empirical Study of Android Repackaged Applications', *Empirical Software Engineering*, (2019).
[55] Gao, Jun, Li Li, Pingfan Kong, Tegawende F. Bissyande, and Jacques Klein, 'Understanding the Evolution of Android App Vulnerabilities', *IEEE Transactions on Reliability*, (2019).
[56] Blanc, William, Lina G. Hashem, Karim O. Elish, and M. J. Hussain Almohri, 'Identifying Android Malware Families Using Android-Oriented Metrics', in *Proceedings 2019 IEEE International Conference on Big Data*, (2019).
[57] Giovannitti, Eliana, Luca Mannella, Andrea Marcelli, and Giovanni Squillero, 'Evolutionary Antivirus Signature Optimization', in *Proceedings of 2019 IEEE Congress on Evolutionary Computation, CEC 2019*, (2019) .
[58] Niveditha, V. R., T. V. Ananthan, D. Usha, K. Amandeep Singh, A. Pooja, M. F. Zeenath Fathima Majeed, and others, 'Android Malware Inspection: Based on Memory Forensics', *Journal of Advanced Research in Dynamical and Control Systems*, (2019).
[59] Brayan Benett, A. S., L. Ranjitha, K. Vinushanth, R. Sam Abisherk, and Amila Nuwan Senarathne, 'Security Platform for Mobile OS', in *2019 International Conference on Advancements in Computing, ICAC 2019*, (2019).
[60] Liu, Kaijun, Shengwei Xu, Guoai Xu, Miao Zhang, Dawei Sun, and Haifeng Liu, 'A Review of Android Malware Detection Approaches Based on Machine Learning', *IEEE Access*, (2020).
[61] Sunali Jogsan, 'A Survey on Permission Based Malware Detection in Android Applications', *International Journal of Engineering Research*, (2020).
[62] Shalabi, Eman, 'On Malware Detection on Android Smartphones', *International Journal for Research in Applied Science and Engineering Technology*, (2020).
[63] Duong, Lai Van, Tisenko Victor Nikolaevich, Do Hoang Long, Nguyen Quang Dam, and Nguyen Quoc Hoang, 'Detecting Malicious Applications on Android is Based on Static Analysis Using Deep Learning Algorithm', *International Journal of Advanced Trends in Computer Science and Engineering*, (2020).