Two Layer Android Multistage Malware Detection

Major Project - 2

BY

ANKUR GUPTA

M.Tech (6th Sem)

2014/SWT/505

COMPUTER SCIENCE & ENGINEERING DEPARTMENT

DELHI TECHNOLOGICAL UNIVERSITY

Delhi – 110042, INDIA

Dec 2017

Student Undertaking

Delhi Technological University

(Government of Delhi NCR)

Bawana Road, New Delhi-42

This is to certify that the project entitled **"Two-Layer Android Multistage Malware Detection"** done by me for the Major project 2 for the award of degree of **Master of Technology** Degree in **Software Technology** in the **Department of Computer Engineering**, Delhi Technological University, New Delhi is an authentic work carried out by me under the guidance of **Dr. Kapil Sharma**.

ANKUR GUPTA
2K14/SWT/505

Above Statement given by Student is Correct.

Project Guide: Dr. Kapil Sharma
Department of Information Technology
Delhi Technological University, Delhi

Certificate

This is to certify that the Dissertation entitled **Two Layer Android Multistage Malware Detection** is a bonafide record of independent research work done by **Ankur Gupta** (Reg. No.: **2K14/SWT/505**) under my supervision and submitted to **Delhi Technological University** in partial fulfillment for the award of the Degree of **MASTER OF TECHNOLOGY**.

Signature of the supervisor

Acknowledgment

Writing this report was a time-consuming process and the result has benefited greatly from the input of a lot of different people whom I would like to thank on this page. I would like to thank my guide, Dr. Kapil Sharma who has been directly involved in developing this report and set me up on this interesting topic. And then there are my colleagues Farhan Ahmad and Prajeed Chathuar whom I want to thank especially for their valuable suggestions and remarks, for both the presentation as well as the actual report.

ANKUR GUPTA
Master of Technology
Software Technology
2K14/SWT/505

Abstract

“Better be despised for too anxious apprehensions, than ruined by too confident security.” — Edmund Burke

Malware instances are increasing their popularity among public. They are every now and again specified in the media and talked about by experts. This proofs that these Android malware have an undeniably major effect on our everyday lives. It raises questions like how we are ensuring our smart phones, and if this level of protection is sufficient. Without any doubt, the impact of the malware, e.g. WannaCry, which influenced the British health system by disabling certain clinics and crisis services, can be seen as a huge advance in the disastrous effect of malwares. This incident prompted either death or postponed treatment on a remarkable scale. Undoubtedly exceptional progress has been made in securing android systems in the recent years. The detection of an expanding number of vulnerabilities, added with logically shorter periods between updates, is reinforcing the reliability of android smart phones.

The proposed thesis work devises malware detection system solely based on machine learning for Android smart phones and provides an extra layer of security and protection to Android based smart phones. This system inspects different features and events collected through the android applications. This system analyses these collective features and perform categorization to label the application as benign or malware with the help of different machine learning based classifiers. In this thesis work, following question is addressed: do malicious applications on Android ask for typically unexpected permissions in comparison to real applications? In view of examination of 1250 malware examples of malicious and 895 benign Android applications, we propose a two level Android malware identification strategy. In this work, Granted permissions are seen as behavioral markers and hence a machine learning classifier is manufactured on those markers. This classifier is used to consequently recognize for never seen applications which may perform unsafe behavior based on permission combinations.

Table of Content

List of Figures

CHAPTER 1: Introduction

Fishy Android applications continue bypassing Google play-store Protect mechanism as they do not seem suspicious at the time of installation Rather; they are using a new technique called multistage attacks. This is the most recent in a string of malware in the Google Play store that is by all accounts proceeding with unabated despite the presence of Google Play Protect, which provided facility to stop malware from being scattered in the market. Google play store protect feature is used to restrict malware and unauthorized applications being published in the store but despite this method play store is not able to face challenges incorporated by new multistage attack enabled malicious applications. [1]

The most recent series of similar incidents signal that they all avoided Google Play Protect in a similar way, by utilizing a multistage attack. Multistage malware can pretend to be benign applications, although doesn't contain suspicious nature at the time of installation, but can work accordingly to its vicious developer choice. What they do contain are a few layers of encoded payloads that in the long run download malware from a site hard-coded into the payloads.

As per recent study, the possible objective is to introduce virus on the compromised device. At the point when the application is at first downloaded from Google Play store, it doesn't ask for any suspicious looking permission. All its intended work is done silently as it decodes and runs its first payload, which decodes and runs the second one. The second-stage payload connects with the malware publishing site and downloads the third-stage payload. It's now that the malware prompts the client to authorize an establishment of what is by all accounts a trusted application. If the user does not allow installation at this point, all the purpose of malware can be halted, and the device can be saved from harmful damage.

Today malware have become a prime cyber threat due to exponential development of Internet. Malware can be defined as any program performing vicious actions which maybe unauthorized access of data, spying, etc. Kaspersky Labs (2017) define malware

as “a kind of PC software programmed to influence a valid customer's PC and infect or spread damages on it in various methods.” [2]

Mobiles are a part of a huge and developing extent of computing devices. Android specifically is the quickest developing mobile platform and has 88% share in markets. [3] The true strength of the Android platform manages applications giving different services, including confidential services like managing a bank account.

Panda Security states that “Computer hackers are progressively working for innovative methods to trap users into downloading and installing malicious software. One among the known worked solutions is to disguise malware such that it seems to user as a benign application; in most of the cases the application will work similar to actual applications, taking confidential users’ information in the back ground.” [4] .

Most devices are now equipped with in-device protections mechanism to prevent

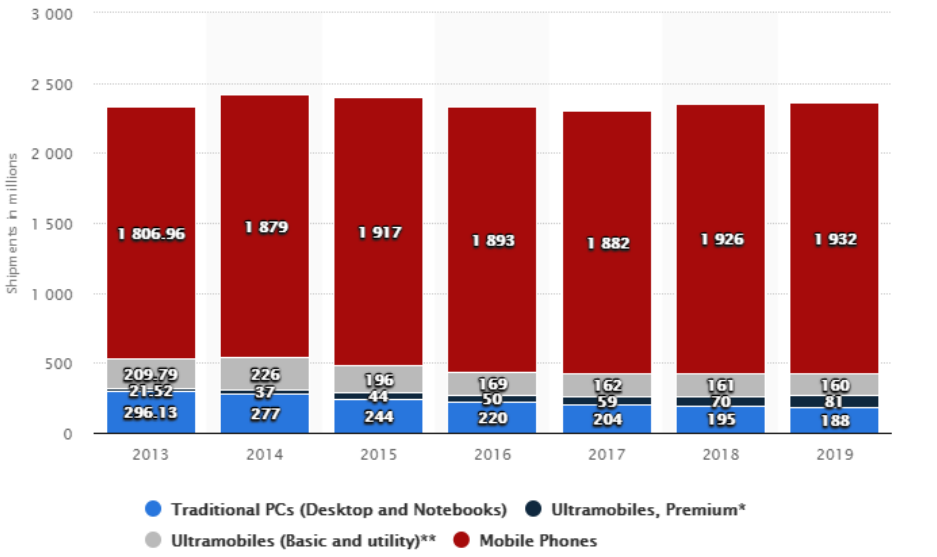


Figure 1 : Count of mobile devices produced on yearly basis

unauthorized access and such malicious events. It is usually performed by restricting computers so that software download and installation can be done through checked sources like official play store. It will restrict user to install only authorized applications from android market place. Irrespective of huge efforts, malware is able to bypass security – usually through a phished email attachment or suspected web pages. To stop

these kinds of attacks, our solution will identify, stop malicious applications and secure users' personal information.

Wireless Smartphone Strategies/WSS is a strategy analytics service which forecasts that global android phone shipment will grow +5% in 2018. [5] Even in 2017, Android will occupy first position among the dominant mobile platforms. This report predicts global mobile phones installed base, by more than ten operating systems, for more than eighty countries worldwide, for a span of 10 years (2007-2017). [6]

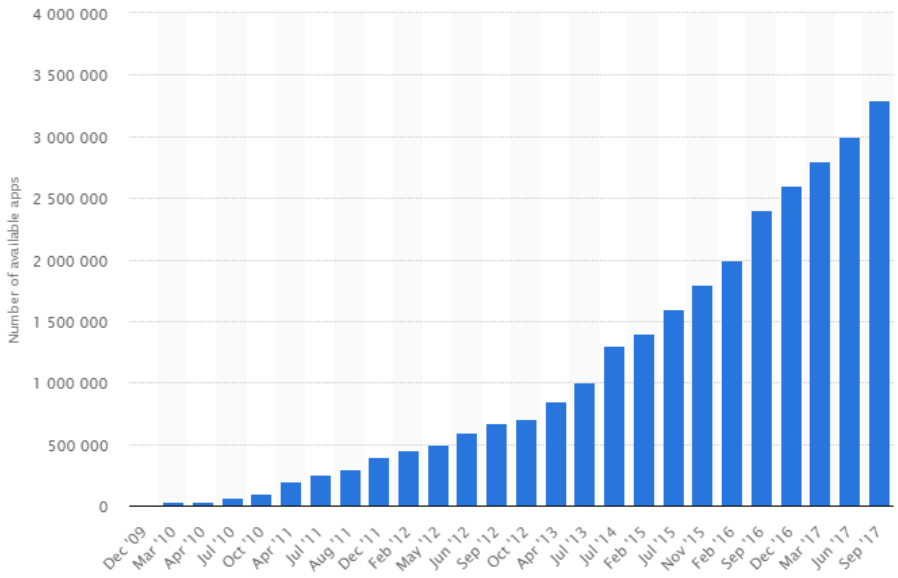


Figure 2 : Number of available play store applications.

Content sharing web sites and syncing service like torrents are another easy source of malware attack. Malware and ransom ware are usually inserted inside pirated movies and applications, and user is not aware becomes prone to data theft.

This power can likewise be utilized by malware. Gigantic development in Android showcase from 2,300 applications in March 2009 to 400,000 applications by January 2012 has additionally pulled in a noteworthy development in malware for Android. Trend Micro, a worldwide pioneer in antivirus, has anticipated the development of Android Malware by December 2012 to be 129,000 malware.

Anybody and everybody can create Android applications and host it on the Android marketplace. Online markets don't have a procedure to check android applications for malware. Google included another security highlight Feb 2 this year to

its Android market for battling malware which will filter each new accommodation and current applications for vicious conduct. This new framework does not make a difference to other 3rd party markets.

According to the IDC Quarterly Mobile Phone Tracker, “Smart phone MNCs produced a major portion of 344.3 million devices all over the world in 2017 first quarter (1Q17). Due to the fact that it might visualize like a slowly pacing market, users continue to display interest for smart phones and OEM flagship publicity looks strongest as like never before. All over the world smart phone production raised 3.4% in 1Q17 year over year, which was a little lower than IDC's previous prediction of 3.6% raise.” [7] Malware regularly pretends them as typical applications, but Malware can cause money related incidents, steal private data. Clients require powerful malware discovery programming interface.

To productively identify malware from benign applications, present on standard and 3rd party marketplaces, numerous attempts have been made. Google tests applications for malicious nature using a utility named Bouncer. Bouncer analyzes Android market applications consequently by executing them in a custom Android stimulated environment based on Google’s cloud architecture. Although malware downloaded count reduced since the establishment of Bouncer, this framework does not give security against present-day malicious attack approaches.

In the past decade, mobile malware was considered an advanced and rare hazard. Majority of smart phone users used to think that they were safe from such hazards. As per McAfee labs “More than 1.5 million new stories of device malware have been identified by McAfee Labs in this year first quarter alone – for a total of over 16 million device malware incidents.” [8]

Study in same field clearly states that “Now, smart phones are facing continuously growing threats – and anyone can’t be escaped from such attacks. Around 20 percent of smart phone MNC studied by Dimensional Research for Check Point Software stated their phones have been compromised. One fourth of participants were not aware whether they have been under threat. Almost all (94 percent) presumed the occurrence count of device threats to grow, and 79 percent committed that it’s being very problematic to protect smart phones.” [9]

“Companies are preparing now to spread awareness of the future influence,” says Daniel Padon, mobile threat researcher at Check Point. “Actual, state level malware and the power of those malware, all together with huge promotions influencing lots and lots of phones, such as Gooligan and Hummingbad, are only the tip of the iceberg.” [10]

The Android OS utilizes the permission framework to limit applications benefits to secure the confidential assets of the clients. An application needs to get clients’ affirmation to provide protection to significant secure information. In this manner, the authorization framework was intended to save clients from applications with obtrusive means, yet its viability exceptionally relies upon the clients’ perception of affirmation.

The developer oversees deciding which authorizations an application requires, but a lot of clients don't have any clue that what any permission implies and indiscriminately agree with them, enabling the application to get to secure data of the client. [11] [12] Another defect is that the client can't choose to allow single permissions while denying others. Numerous clients, even though an application may ask for a suspicious permission among much apparently real permission, will, in any case, affirm the installation.

Google mostly classified permissions as coarse-grained. Particularly, the INTERNET authorization, the READ PHONE STATE consent, and the WRITE SETTINGS permission are coarse-grained as they provide for an application discretionary access to specific resources. The INTERNET consent enables the said application to forward HTTP(S) requests to all domains, and interface with random targets and sockets. Accordingly, the INTERNET consent gives inadequate expressiveness to uphold control over the Internet gets of the application. Considering the past issues, analysts have been included to decide systems that utilize singular permissions and the mix of permissions to identify and classified malware.

Study published in Statista claims that “The figures predict the count of present applications in the Google Play Store, earlier referred as Android Market (Dec,2009-Sep,2017). In Sep,2017 Google Play store was offering more than 3 million applications as compared to 1 million applications in July,2013. Google Play was actually inaugurated in October 2008 with the name Android Market. As Google’s official application store, it presents its users from applications and digital multimedia (books, music, film, magazines, and TV). With the large portion of applications present

from the Google Play Store with absolutely free of cost, the company requires to use working business models to save wealthy return. As of Feb,2017, the top earning Android applications all over the world including demanding gaming applications like, Candy Crush, Temple Run, and Clash of Clans etc. Even the strong gaming applications returns, most gaming applications are free to install; the source of income for these applications is in-game advertisements and game boosters." [13]

More than one billion users use internet browsers at smart phones, this mechanism importance and ever growing user base are a base factor. As many smart phones are available in market, malware are targeting these phones to harm in full force. [14]

Increasing use of Smartphones, the incidents for data theft, identity duplications, and monetary transactions are also spreading widely. Smart phones are easy to target and its user are generally do not know the impact of malicious application and dangers produced through them.

As smart phones differ in working environments (same as the typical Windows-based computer and Apple devices), cyber attackers usually modify their attacks. Although, they have become best at by passing the OS security for example Android, iOS and Windows-based devices security. [15] While downloading apps different platforms follow different techniques, Android stimulates a simple way for the same.

1. The first way is to set up apps from other party market set.

2. The specific application market has dedicated security service.

3. Botnet client are easily installable to android platform.

4. Android application programmers are uploading their applications without inspection.

5. As study suggests, very often general threats in smart phones that harm device owners generates from malware executed on smart phones supposed to provide:

A. Freedom appreciation to root.
B. Releasing personal data.
C. Face finest figures.

D. Botnet malicious behavior.
E. SMS based backdoor activity.
F. Private critical information data collection resided on a device.
G. Device Owner usage activity surveillance.
H. Forwarding SMS / MMS to other devices without user consent and they are charged for the same.
I. store malware documents onto smart phones
J. Support hackers to control devices through unauthorized means.
K. Locate users' current movement.
L. Change user's settings
M. Access personal pictures, videos etc.
N. Check bank transaction details – These kind of viruses are supposed to log each and every bank account activities and forward all confidential data including secret data, pass codes etc to intended malicious guys at other end.

Any cell phone that gets to the Internet by means of programs or applications is liable to assault. Some OS, anyhow, have turned out to be particularly in demand by cybercriminals due of their fame and the simplicity with which malware can be downloaded onto the mobile. Cell phones that utilize the Android OS are very in demand among the users, which mean they're also most loved among the terrible attackers. Malware attacks are among the worst dangers Android clients confront nowadays. The danger of risk has turned out to be so high; indeed, the U.S. Government investigation agency lately cautioned smart phone users to the possible catches related with Google marketplace and open to all source design.

Android OS, just similar to other OS that browses internet, are usually infected by many different kinds of malware that usually differentiate users and their sensitive data. When this information is received, the hackers use this data to pretend to be someone else, debited their bank accounts and so on. [15] In order to clearly understand the ways and mindset after malware, classification becomes mandatory.

Malware can be classified among several categories. The classes are as follows:

Virus: There are various ways in which a virus can be installed on the device and it can harm the user which ranges from normally annoying to extremely damaging. Malicious attackers might potentially use different viruses to root the device, fetch documents and personal data filled memory. A Virus can be defined as a malware that duplicates itself and spreads to different android devices. Viruses multiply and spread by attaching themselves to various executables and executing code when a device user sends any maligned executables. Viruses can be used to steal data, damage host smart phones, make chat bots, take financial advancements, issue instructions, and that is just the beginning.

Worm: Smart phone worms are identified as most popular malware. They amplify themselves across different computers due to security loopholes in operating systems. Worms normally make damage their host networks by expending data transfer capacity and over-burdening web servers. Similarly, System worms includes payloads that damages hosting network. Payloads are bits of code written to perform exercises on impacted Smart Phones, which is past the farthest point from simply spreading the worm. Payloads are ordinarily planned to take data, eradicate records, or make chat bots. Smart Phone worms can be deputized a sort of Smart Phone infection, yet there are a few features that differentiate Smart Phone worms from normal viruses. A remarkable contrast is that Smart Phone worms can repeat it and infect autonomously whereas dissemination of viruses is highly dependent on human interaction. Worms generally contaminate through bung messaging with malicious attachments to user's network.

Trojan: In view of this the normal influencing vector used in this class is social computation based that is influencing individuals to believe that they are downloading the valid software. [16] Mobile Trojan infects user devices by attaching itself to seemingly non dangerous or authentic applications, are presented along with the program and afterward perform harmful events. These programs are used to seize the application, force the device naturally forward unauthorized highly confidential messages, or store user login credentials from various programs, e.g., mobile banking. A Trojan horse/Trojan is a malware that pretends to be a normal document or

application to involve users into installing and consequently getting infected from malware. This Malware provides an unethical hacker host to get to an influenced Smart Phone. Once a hacker approaches a influenced Smart Phone, it makes easy for the hacker to steal data, produce multiple malwares, adjust saved information, user interaction (screen watching, key logging, and so on), utilize the Smart Phone to chatbots, and hampers internet surfing.

Adware: The main motivation behind this malware composes is showing promotions on the Smart Phone. Frequently adware is a subclass of spyware and it will be impossible to prompt sensational outcomes. Adware aka promotions enabled malware that generally displays ads. Basic cases of this malware incorporate pop-up advertisements at sites and ads which are shown through the application. Regularly programs and executables provide "costless" questionnaire containing adware. Mostly these malwares are reinforced or written by promoters and fills in as an money raising method. Though some adware is only expected to pass on advancements, it isn't remarkable for adware to be bundled with spyware that is best suited for given customer action and making information. Due to the extra limits of spyware, adware bunches are basically more destructive than adware in solitude.

Spyware: The malware which accomplishes surveillance are termed as spyware. Regular activities of spyware incorporate following web surfing history to send customized promotions, following exercises to sell them to the outsiders in this manner. [17] Spyware subtly assembles confidential information of the mobile user and forwards this data to an unauthorized person afterwards. It is regularly introduced without user agreement by disguising as a true verified program (e.g., a straightforward game) or by compromising its load to a benign program. Spyware uses the user's mobile connection with hand-off personal data, e.g., contacts, area, messaging nature, installation history, and user choices or downloads. Spyware that collects customer data, e.g., operating system choice, item ID, IMEI number and IMSI number can be utilized for future assaults. Spyware is a sort of malware that operates by keeping an eye on client action without their consent. These spying capacities can consolidate activity watching, collecting key pressed by user, data gathering (account information, logins, money related data), and that is just a glimpse of a larger problem. Spyware routinely

has additional power as well, going from adjusting security settings of programming or software to ending with network associations. Spyware amplifies by abusing programming vulnerabilities, bundling itself with substantial projects, or in Trojans.

Rootkit: Its usefulness empowers the hackers to get to the information with higher authorizations than is permitted. For instance, it can be utilized to give an unapproved client unauthorized access. "Rootkits dependably conceal its reality and regularly are unnoticeable on the framework, making the identification and accordingly evacuation unbelievably hard." [18] A rootkit is a kind of unsafe projects proposed to remotely access or control a Smart Phone without being recognized by customers or security programs. Once a rootkit has been presented it is workable for the programmers behind the rootkit to remotely execute reports, get the opportunity to/take information, modify system outlines, change programs (especially any security programming that could distinguish the rootkit), present masked malware, or control the Smart Phone as a segment of a botnet. Rootkit foresight, recognizable proof, and removal can be troublesome due to their stealthy errand. Since a rootkit determinedly hides its quality, normal security things are not effective in perceiving and clearing rootkits. In this manner, rootkit acknowledgment relies upon manual procedures, for instance, checking Smart Phone lead for bizarre exercises, signature inspecting, and capacity dump examination. Affiliations and customers can shield themselves from rootkits by reliably settling vulnerabilities in projects, applications, and working systems, reviving infection definitions, avoiding suspicious downloads, and performing static examination filters.

Backdoor: Without anyone else's input, it doesn't cause any damage however furnishes hackers with more extensive assault platform. Along these lines, secondary passages are never utilized freely. For the most part, they are going before malware assaults of different kinds

Keylogger: Without anyone else's input, it doesn't cause any damage however furnishes hackers with more extensive assault platform. Along these lines, secondary

passages are never utilized freely. For the most part, they are going before malware assaults of different kinds. [19]

Ransomware: This kind of malware intends to encode every one of the information on the machine and request that a casualty exchange some cash to get the unscrambling keys. For the most part, a machine contaminated by recover product is "solidified" as the client can't open any document, and the work area picture is utilized to give data on hackers' requests. [20] Ransomware is a sort of malware that fundamentally holds a Smart Phone system victim while asking for an installment. As far as possible customer access to the Smart Phone either by making archives encoded on the hard drive or securing the structure and indicating messages that are wanted to compel the customer to pay the malware producer to empty the constraints and recover access to their Smart Phone. Ransomware generally spreads like an average Smart Phone worm ending up on a Smart Phone by methods for a downloaded record or through some other weakness in network associations.

Phishing Apps: Mobile surfing of the web is developing with Smartphone and tablet devices. Similarly as with desktop usage, hackers are making mobile phishing websites that may resemble a verified service however may take client secret information or more awful. The smaller screen of cell phones is making harmful phishing systems simpler to hide from users, less complex on cell phones than Smart Phones. Some phishing plans utilize harmful mobile applications, programs which can be considered "trojanized", masking their actual goal as a OS update, advertising offer or game. Others contaminate verified applications with malicious code that is just found by the client subsequent to installation of applications.

Botnet: Mobile malware is getting more complex with executables can work silently in background on the client device, covering themselves and lying in wait for specific actions like a web based banking session to occur. Hidden procedures can execute totally undetectable to the client, run executables or contact Bot masters for new guidelines. The following wave is relied upon to be significantly further developed, with botnet behavior to capture and control contaminated devices. Bots are software made to normally perform specific exercises. While a couple of bots are made for reasonably safe purposes (video gaming, web tenders, online tests, etcetera.), it is winding up logically fundamental to see bots being used maliciously. Bots can be used

as a piece of botnets (collections of Smart Phones to be controlled by outcasts) for DDoS assaults, as spam bots that render promotions on destinations, as web crawlers that rub server data, and for flowing malware fake files, also known pursuit things on download sites. Locales can make arrangements for bots with CAPTCHA tests that check customers as valid user.

Spam: Spam can be considered as the virtual forwarding of huge personal messages. The most generally perceived medium for spam is email, yet it isn't wonderful for spammers to use writings, informing, web diaries, web dialogs, web look devices, and online person to person communication. While spam isn't generally a sort of malware, it is to a great degree consistent for malware to spread through spamming. This happens when Smart Phones that are affected with contaminations, worms, or other malware are used to scatter spam messages containing more malware. Customers can balance getting spammed by avoiding new messages and keeping their email addresses as private as could be normal considering the present situation.

The transient development of the Android Smartphones has made it a fundamental focus of digital world criminals. Mobile malware particularly focusing on Android has surged and developed couple with the rising prevalence of the platform. Accordingly, the burden is on safeguards to raise the difficulty of malware development to check its widespread growth and to devise successful detection methods particularly focusing on Android malware to better secure the end-clients.

The main aim to develop this project is to construct an Android permission based application using advanced machine learning basics. Based on the permission asked by individual application, that application is categorized as benign or malicious. This design decompresses sample applications to take out permission set which acts as input for machine learning based tool. The objective is to decide how the permissions ought to be removed and the logic that can recognize the malwares with the most reduced error rate. The present approach utilizes a machine learning classifier to consequently distinguish harmful nature of downloaded software from the commercial center in light of the arrangement of authorizations they need their clients to concur with. Along these lines, applications those are not known previously and zero-day malware can be

identified. Some motives behind malware detection on the Android platform based mobile devices include:

i. The identification technique must utilize memory and computational assets effectively and not deplete the phone battery.

ii. The identification strategy must have low false alert rate i.e. considering a non-malware record as malware.

iii. The recognition technique must be simple/taken a toll effective to refresh over the remote system.

iv. To show a straightforward way to deal with ordering benign/malicios applications on Android based devices.

v. To consolidate correspondence interfacing access conduct as the second level of assurance.

Remembering the pin points for malware identification on a cell phone, the permission based recognition strategy is appropriate. It ought to likewise be noticed that permission recognition will remain a piece of versatile antivirus frameworks regardless of whether different methods like heuristic filtering are created since it encourages speedy location of known infection and viruses. Here, the point is to build up an permission based technique to such an extent that it has high filtering speed and low memory use which makes it appropriate for cell phones.

In particular, an application utilizes a few consents (and appropriately to plays out some framework activities) without expressly pronouncing them in its Android manifest. An Application must ask permission as per its conduct. The proposed work executes differencing logic on Server side and henceforth web accessibility is must and at times reaction may take tad additional time in light of activity. In addition, Stats for network use of the application is additionally logged which needs some additional storage.

Android is easy to access and popular OS for various devices like smart phones, televisions, auto mobiles, Gear equipments. Android is Linux based, provided by Google and launched on 23 September, 2008. Many organizations and producers use Android as base for their devices. Android also allows manufacturers to root devices as per requirement. Android comes with UI tools which user can easily use in their product development. For example, Android Software development kit, Android Native library development kit, ADT tools for eclipse. Whenever Android new version is launched, SDK is also updated. SDK provides developer a bundle of Java native classes, run time jars and debugging tools. It also presents an Emulator to help programmers. This emulator is mainly used to verify new applications on different Android operating system version released. The other tools NDK supports C, CPP and other additional languages libraries which can be embedded in Java code through API System.loadLibrary(). ADB tool is redistributed along with Android framework which is composed of server, client and daemon. Android devices run Clients and daemon executes in background. Server is used as bridge between client and daemon. ADB also presents user to verify their applications for possible error without executing on actual devices.

Android applications mainly run on Java platform. It also supports C, CPP for native libraries development. Google Marketplace is standard place for Google or other developers' android applications. User can surf, execute on device and can use new versions provided by developer. As per statistics presented in 2017, more than half a trillion applications were downloaded through Android marketplace. It makes very obvious that Android open source architecture makes contributors and hackers to post their applications on market place. Google play store has a standard mechanism to identify non benign applications and uninstalled them at the same time. It has a tool called 'Bouncer' that scrutinize every applications posted on play store prior making them available to public. Bouncer uses a dynamic approach to identify applications by executing them in simulated behavior to inspect the application's nature.

After this tight security provided by Google, hackers knows the method to bypass their application from scrutinize system. They use encryption to seal their applications. As per report submitted by Columbia University, R & D centers have found security concerns in aforementioned Bouncer system. Because of these loopholes hackers are able to submit their malicious apps as benign apps in Google play store. The well

verified applications can be redistributed and reformed as malicious ones. That's why Android is a burning researched topic among developers. At the same time it is crucial for User's personal data. Android operating system is basically explained in sections which comprise methodology, secure components, apps features, permission android architecture.

For example, if an app wants to communicate over the network, it would need to have an entry like what it is shown in Figure 5.

```
<uses-permission android:name="string" />
```

Figure 3 : Uses permission structure in Android Manifest file.

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
package="com.example.testapps.test1">
...
<uses-permission android:name="android.permission.INTERNET" />
...
</manifest>
```

Figure 4 : Example of test application Manifest file.

Table 1: Example of permissions

CALL PHONE	Allows an application to initiate a phone call without going through the Dialer user interface for the user to confirm the call being placed.
MODIFY PHONE STATE	Allows modification of the telephony state such as power on
WRITE SMS	Allows an application to write SMS messages.
READ CONTACTS	Allows an application to read the user's contacts data.

Android provides 130+ built-in permissions; and at the same time developers can also declare new permissions called dynamic permissions. The android by default permission can be labeled as four security categories: normal, dangerous, signature, and signatureOrsystem.

Normal	a low-risk permission that enables the app to access the least critical system resources; it is automatically granted by the system without notifying the user.
Dangerous	a high-risk permission that enables the app to access the user's private data and the phone's hardware, system, and software. The dangerous permission is shown on the screen at installation time to the user who is for granting this kind of permission. The user must grant the requested permissions under his understanding and acceptance of the consequences, otherwise, the installation process is terminated by the system.
Signature	a permission granted by the system only if the app requests a permission that is declared by another app and both apps are signed with the same certificate. This permission is granted automatically by the system without notifying the user if both apps have the same certificate.
Signtureorsystem	a permission granted by the system only if the apps are in the Android system image or signed with the same certificate of the app that declared the permission. This permission is granted automatically by the system without notifying the user.

```
<manifestxmlns:android="http://schemas.android.com/apk/res/android"
                package="com.example.testapps.test1">
...
<permission android: name="com.example.testapps.test1.perm.READ_INCOMING_EMAIL"
android: label= "Read incoming email"
android: description="description of the permission "
android: protectionLevel="dangerous"
android: permission ="android.permission-group.PERSONAL_INFO"
 />
 ...
</manifest>
```

Figure 5 : Permission attributes in test application

Malware identifiers that depend on signatures can perform well on beforehand known malware that was at that point found by some antivirus companies. In any case, it can't recognize polymorphic malware that has the power to change its signatures, and in addition, the new malware also, for which signatures have not been made yet. Thusly, the exactness of heuristics-based indicators isn't generally adequate for satisfactory detection, bringing about a considerable measure of false-positives & false-negatives. [21]

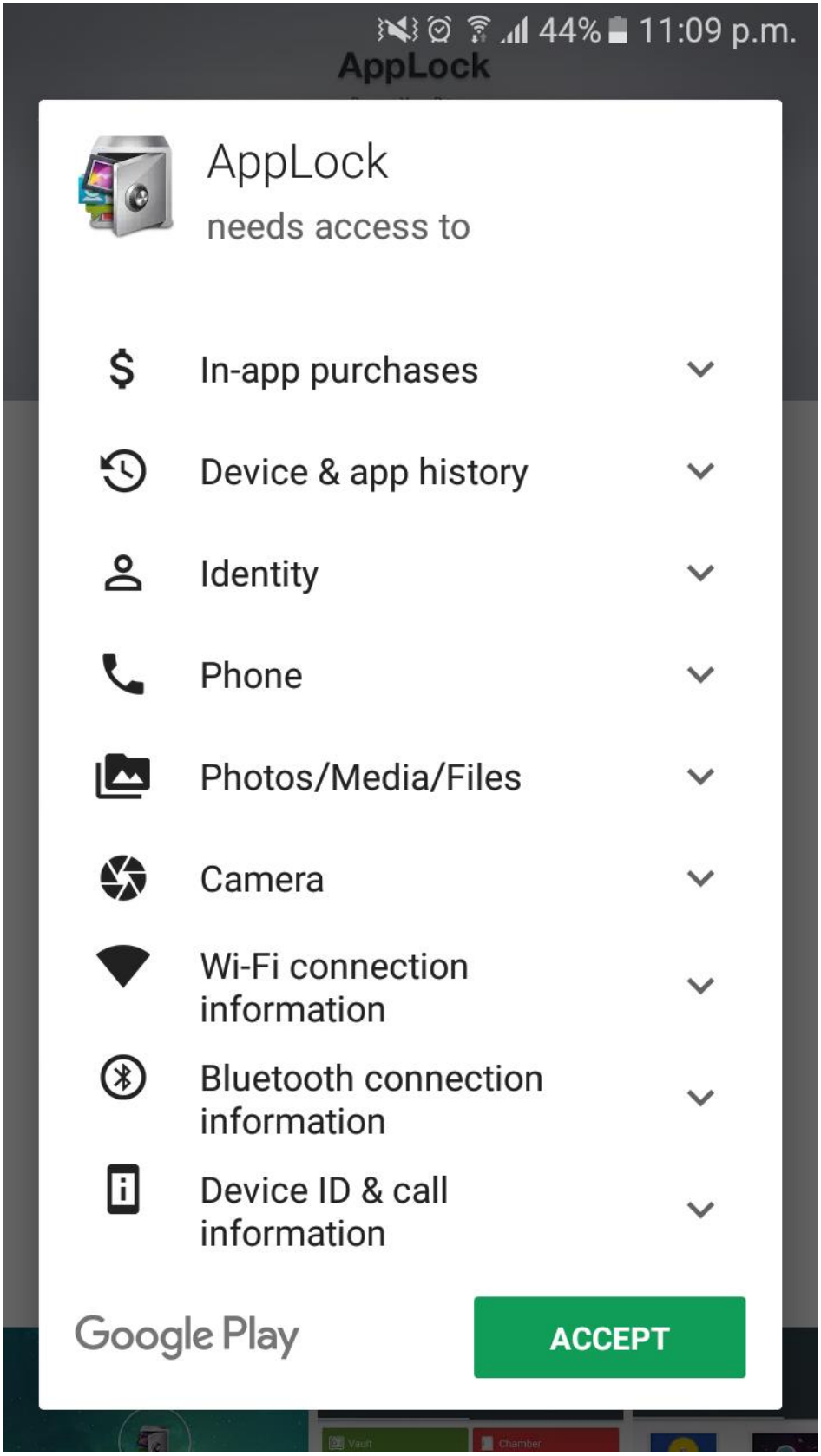


Figure 6 : A Sample application asking user to affirm given permissions.

New clasiification mechanism is to devised as current malware system is infecting android devices at exponential rate. Out of several techniques, higly effective way is to accommodate current heuristic-based technique with wisely selected machine learning approaches.

The problem with heuristic-based technique, a malware detection threshold value should be selected which acts as heuristic measure to label an application as malware application. To consider these connections and give a more efficient solution, machine learning techniques can be utilized.

How does malware sneak past the obstruction made by conventional, Smart Phone-based anti-virus frameworks? Indeed, it has numerous ways — the greater part of which are client made:

- Anti-Virus isn't operational: The effect of constant anti-virus security on a phone's battery, memory and processor can be more than the normal buyer is ready to acknowledge. Therefore, numerous contaminations are shrunk by clients who kill anti-virus software since they see it to contrarily influence the execution of the games and applications they need to utilize.

- Anti-Virus software isn't updated: Many web clients overestimate the adequacy of their anti-virus software, ignorant that it is just in the same class as the most recent updates gave by the security seller. Contingent upon the source, just 50– 80 percent of individuals have updated anti-virus software introduced on their Smart Phones. Past this present, it's not simply security programming that should be stayed up with the latest — older forms of apps, modules, and working frameworks would all be able to be confiscated by malware.

- The product isn't arranged effectively: As malware turns out to be more complex, so does the security programming intended to ensure against it — making it progressively troublesome for the normal client to know whether the product is doing its activity. Much of the time, the contrasts between on-request checks, planned sweeps, email filters, download examines and on-utilize filters are not clarified. Thus, clients are not fit for designing these alternatives effectively or are uncertain of the effect certain features will have on the security of their framework.

- The product isn't totally viable: Malware creators utilize robotized mechanism to always repackage and jumble their malware to confiscate location by anti-virus software. Also, with security sellers contending with a huge number of new malware tests every day, scholars have demonstrated that customer based anti-virus software can identify just 50– 75 percent of malware, leaving a possibly extensive extent totally unidentified. Some anti-virus software additionally can't distinguish spyware or adware infections; as a rule, completely extraordinary projects are required for this reason.

- More devices, more associations: Both settled and versatile specialists in providing services are presently looked with a multi-directional flow of malware. For instance, Internet specialist providers are liable to cell phone/tablet malware when these devices are associated with home WIFI switches; likewise, mobile vendors are liable to malware originating from Smart Phones fastened to Smartphones or associated with portable Internet sticks or 3G/Wi-Fi hotspots. Phones running Google's Android working OS have turned out to be especially defenseless as of late, with Kind sight Security Labs revealing a 400 percent expansion in malware from early September to late November 2011. Indeed, even with the most updated software introduced and security appropriately empowered on their Smart Phones, clients are as yet not totally safe from contamination. In mid 2012, the Zeus Tracker site revealed that the normal anti-virus location rate was 36% for firmware of the Zeus Trojan — malware particularly intended for saving money based data fraud. In like manner, in 2011, inquire about from Surf Right found that 32 percent of clients with state-of-the-art software were contaminated with malware.

WEKA is an information mining framework created by the University of Waikato in New Zealand that explores data mining techniques. WEKA is a cutting edge tool for creating machine learning (ML) strategies and their application to true information mining issues. It is a gathering of machine learning strategies for information mining work. The logic are connected straightforwardly to a dataset. WEKA executes design implementations for information preprocessing, arrangement, relapse, bunching, association rules; it likewise incorporates representation tools. The new machine learning plans can likewise be created with this bundle. It is open source programming issued based on the GNU Public License. [22] [23]

CHAPTER 2: Literature Reviews

All malware identification procedures can be partitioned into signature-based and behavior-based techniques. Before going into these techniques, it is basic to comprehend the fundamentals of two malware investigation approaches: static and dynamic malware examination. As it suggests from the name, static analysis is performed "statically" which means without execution of the record. Conversely, dynamic analysis is led on the record while it is being executed for instance in the virtual machine.

Static examination frequently depends on specific tools. Past the basic investigation, they can give data on security procedures utilized by malware. The primary favorable position of static investigation is the capacity to find all conceivable behavioral situations. Looking into the code itself enables the specialist to see all methods for malware execution, that isn't constrained to the present circumstance. In addition, this sort of examination is more secure than dynamic, since the document isn't executed, and it can't bring about awful outcomes for the framework. Then again, static investigation is significantly more tedious. As a result of these reasons it isn't typically utilized as a part of genuine dynamic conditions, for example, against anti-virus frameworks, however is regularly utilized for look into purposes, e.g. when creating signatures for zero-day malware. [24] Another investigation compose is dynamic one. Not at all like static investigation, here is the behavior of the document checked while it is executing, and the properties and aims of the record are gathered from that data. Typically, the document is keep running in the virtual condition, for instance in the sandbox. Amid this sort of investigation, it is conceivable to locate every behavioral property, for example, opened documents, made mutexes, and so on. Besides, it is substantially speedier than static examination. Then again, the static investigation just demonstrates the behavioral situation important to the present framework properties. For instance, if our virtual machine has Android OS 7.0 introduced, the outcomes may be unique in relation to the malware running under Android 8.1. [25]

In any case, ill intended people began to create malware in a way that it can change its signature. This malware highlight is named to as polymorphism. Clearly, such malware can't be distinguished utilizing simply signature-based identification methods.

In addition, new malware can't be recognized utilizing signatures, until the point when the signatures are made. In this manner, AV merchants needed to find another method for recognition – behavior based likewise named to as heuristics-based investigation. In this strategy, the real behavior of malware is seen amid its execution, searching for the indications of malicious behavior: altering host documents, registry keys, setting up suspicious associations. Without anyone else's input, every one of these activities can't be a sensible indication of malware, yet their mix can raise the level of suspiciousness of the document. There is some limit level of suspiciousness characterized, and any malware surpassing this level raises a caution. [26]

The precision level of heuristics-based discovery very depends with respect to the execution. The best ones use the virtual condition, e.g. the sandbox to run the record and screen its behavior. In spite of the fact that this strategy is additional tedious, it is considerably more secure, since the record is checked before really executing. The principle favorable position of behavior based discovery technique is that in principle, it can distinguish referred to malware families as well as zero-day attacks and polymorphic infections. Be that as it may, by and by, thinking about the high spreading rate of malware, such investigation can't be viewed as powerful against new or polymorphic malware.

A. Barrera et al. have proposed examination on the basis of permission-based safe structure and proposed some that are self-sorting logic gives two-dimensional thought of high dimensional information and furthermore proposed crowoid which utilizes Linux framework calls for find malware framework calls are open ( ) to open it, read ( ) for perusing record, access ( ) to getting to it, chmod ( ) for evolving mode, chown ( ) for evolving owner. Next one is exhibited SAAF offer program. It considers 136 000 delicate applications and 6100 vindictive applications. SOM logic which preserves instantaneousness and outfits a rearranged and social perspective of a significantly complex informational collection. [27]

B. Portokalidis et al. have proposed an approach paranoid android that is finished malware examination. It used to perform security investigation android that depends on portable reproductions and cloud storage. It was dynamic behavior examination of framework so it hard to distinguish at runtime. [28]

C. Zhou et al. likewise proposed droidMoss which takes fuzzy hashing method. It used to perform security investigation with method NFS storage and ZFS document framework. Fuzzy hashing procedure is harder to perform investigation of recognizing malware. [29]

D. Enck et al. have suggested that gives recommendations on to clients concerning application and that interest boycotted sets of authorization. Their result shows the open utilize wrongly of protection insightful information, the check of phone abuses, extensive including of advertisement libraries the Android application, and the debilitating to safely utilize Android APIs of numerous designers. [30]

E. Felt et al. have proposed stowaways to test over benefit in android application and used to figure 940 applications from the market of Android, they recognized and evaluated engineers' example prompting over benefit. They decide androids get to control approach through automotive testing method. Their result close-by a fifteen-fold improvement more than the android documentation and uncover that most condemn are endeavoring to pursue the govern of minimum advantage yet not prevail because of the absence of reliable authorization data. [31]

F. Elish et al. utilized an investigation implies make information reliance diagrams statically with between procedural call network data that catches the information use dealings in the program through distinguishing the coordinated ways between client sources of info and section focuses to process given that genuine framework services. Some malware may attempt to dodge their information reliance assessment by abusing the client's sources of info while playing out the malignant conduct, so their work should be better in these circumstances.

G. Kantiya Junhom have proposed Cloudbroid is the application rely upon android applications and cloud stack, it is an intersection point amongst portable and cloud clients. In the cloudbroid the application that oversees cloud stack administration and which relies upon the android application utilizing REST

Principles. The fundamental thought behind this is getting to the cloud stack framework wherever and whenever with numerous applications to build a possess business event.

CHAPTER 3: Feature Identification of Malware

The key objective of our analysis is to comprehend whether the collection of permissions required by an application connects with its malicious or benign nature. And how it can be used automatically detect it. In this segment decisions and usage of a detection system that implements a classifier based on those permissions are depicted.

An Android malware recognition system can be constructed in many ways. A proactive approach is to deploy solution for all users on android commercial play store, for example, Google Play Store or 3rd party stores and verifies against every upload. But these changes are not worthy and should just be issued after the techniques have been completely approved. Here, a proof-of-concept design concentrated on ensuring every customer is portrayed.

Mobile devices generally have lower memory resources than Smart Phones. This implies that applications running on mobile devices must be designed to utilize memory efficiently. To perform permission matching, an array of bytes corresponding to all the permissions in applications is sent to the server. One way is to implement detection logic on the mobile device itself, but this way logic controller needs to be updated frequently which in turn is a battery consuming process. Meanwhile, it can let malicious applications be installed in our method during the update.

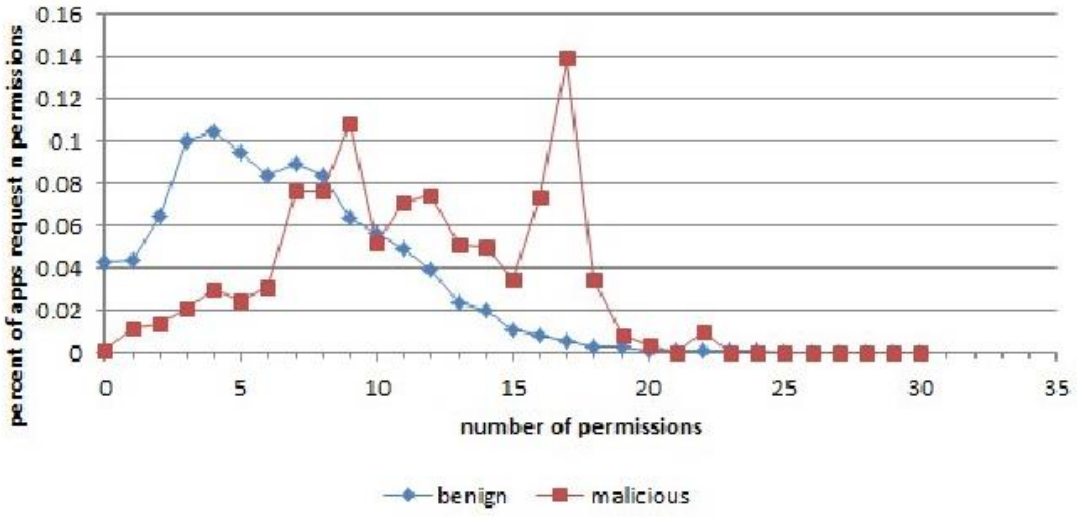


Figure 7: Apps ratio as per number of permissions

This solution is implemented as cloud architecture. Such architecture has a few extra benefits. An imperative one is that it can be generalized to non-android applications. And the second advantage is that it uses fewer device resources, which thus will enhance the battery lifetime contrasted with an on-device solution.

This solution scans every application installed through any means (Either play store or added directly) which generates a limitation that solution can be executed only after new application has been added which in turn creates a race condition between the newly added application executes, performs its intended malicious work and results received from server where actual scan is performed declaring the application malicious.

The server-side application is the essence of our solution as it is the segment in charge of declaring an application's behavior as either normal or maliciousness.

Permission used by applications are collected by using python script and a bit value for each permission are logged in a file that is used as training data to train the classifier. Classifier thus created is used for predicting the result for each application downloaded through the market play store.

If a permission is absent in manifest of Android application 0 is written while 1 is for opposite situation. And thus, set of bit values for each application is stored along with its state (either benign: 0 or malicious: 1). the static array length is thus created is always fixed and its index represents a standard android manifest permission.

0|0|0|0|1|1|0|0|0|1|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|1|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|
0|0|0|0|0|1|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|
0|0|0|0|0|0|1|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|1|0|0|0|0|0|0|0|0|
0|0|0|0|0|0|0|0|0|0|0

Figure 8 : Benign Sample Feature Vector extracted from its manifest

0|0|0|0|0|0|0|1|0|0|0|0|0|0|0|0|0|0|1|0|0|0|1|0|0|0|0|0|0|0|1|0|0|0|0|0|1|0|0|0|1|0|0|0|1|1|0|0|0|
1|0|0|0|0|0|1|0|0|1|1|0|0|0|0|0|0|1|0|1|0|0|0|0|0|1|1|0|0|0|0|0|0|0|0|0|0|0|1|0|0|0|1|0|0|0|0|0|
1|0|1|0|0|0|0|1|0|0|1|0|0|0|0|0|0|0|0|1|0|1|0|0|0|0|0|0|1|1

Figure 9 : Malware Sample Feature Vector extracted from its manifest

The general actions we have taken after for every application are:

1. We downloaded and gathered malware and benign applications from application marketplace.

2. We decompress applications to separate the content.

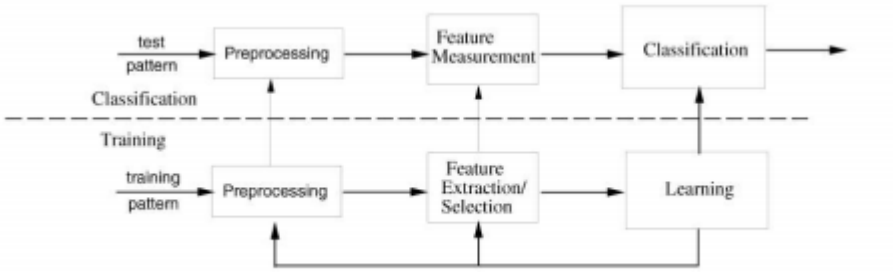
3. We separate the permission request features from every application.

4. We construct a dataset in an ARFF [4] record format with the extracted information.

In the first place, we decompress the android application bundle file to separate the content. Amid the initial three stages, we recover the data from this source. We process the AndroidManifest.xml record to retrieve this information.

Machine learning is a popular sub branch of AI which achieved a great popularity in recent years. The base for this approach is to continuously learn from data set so that it can successfully categorize unseen values. Machine learning is defined as the AI field used to find purposeful pattern among given data set. The main objective is to build a model that can work standalone or at least classify new data from trained history. There are two methods for the same:

1. Clustering: To group similar data.
2. Two class classification: Identification whether data belongs to which of given classes.



In Machine Learning applications, numerous separated features, some of which repetitive or not of much relevance, show a few issues, for example, deluding the learning logic, over-fitting, diminishing generalness, and expanding model comprehensiveness and run-time. These worst impacts are considerably more critical while applying Machine Learning techniques on cell phones since they are frequently limited by preparing and storage abilities, and also battery control. Applying fine feature selection in a preliminary stage empowered to utilize our malware finder all the more productively, with a speedier recognition cycle. By and by, diminishing the quantity of features ought to be performed while protecting an abnormal state of precision. In this area k best features are chosen from the extricated features of android APK zip by utilizing determination strategy.

Signature DB is created from these chosen features and at the same time these features are separated in two sets called Testing and training data sets. These data sets are passed to any standard classification techniques of machine learning category. The current thesis work uses decision tree classifiers for its work.

Decision tree mechanism is a general, east to implement and fast result yielding method [15]. Its development procedure is top-down, divide-and-rule. Basically it is a greedy algorithm. Beginning from the root node, for each non-leaf node, firstly choose an attribute to examine the example set; Secondly, partition training test set into a few sub-test sets as indicated by testing outcomes, each sub-test set constitutes another leaf node; Thirdly repeat the above division process, until having achieved particular end conditions. During the time spent developing a decision tree, choosing the testing attribute and how to partition test set are exceptionally critical. Diverse decision tree

logic utilizes distinctive techniques. By and by, on the grounds that the span of preparing test set is normally substantial, the branches and layers of the produced tree are likewise more. What's more, abnormity and irrelevant data existed in preparing test set will likewise cause some abnormal branches, so we have to prune decision tree. One of the best favorable circumstances of decision tree grouping logic is that: It doesn't expect clients to know a great deal of foundation information in the learning procedure.

The machine learning based J4.8 decision tree algorithm is used for classifying malicious behavior of the Android application based on previous history. The algorithm is best described as follows:

J4.8 is regularly classified to as a statistical classifier. Creators of the Weka machine learning programming portrayed the J4.8 calculation as "a historic point decision tree program that is likely the machine learning workhorse most broadly utilized as a part of training to date".

J4.8 fabricates decision trees from a training of preparing information utilizing the idea of data entropy. The training data is a set S= {s {1}, s {2} ...} of already classified samples. Each sample s{i} consists of a p (=331 + 1)-dimensional vector (x {1, i}, x {2, i} ..., x {p, i}), where the x {j, i} represent jth permission value for that sample and the last value represents class (0/1) in which s{i} falls.

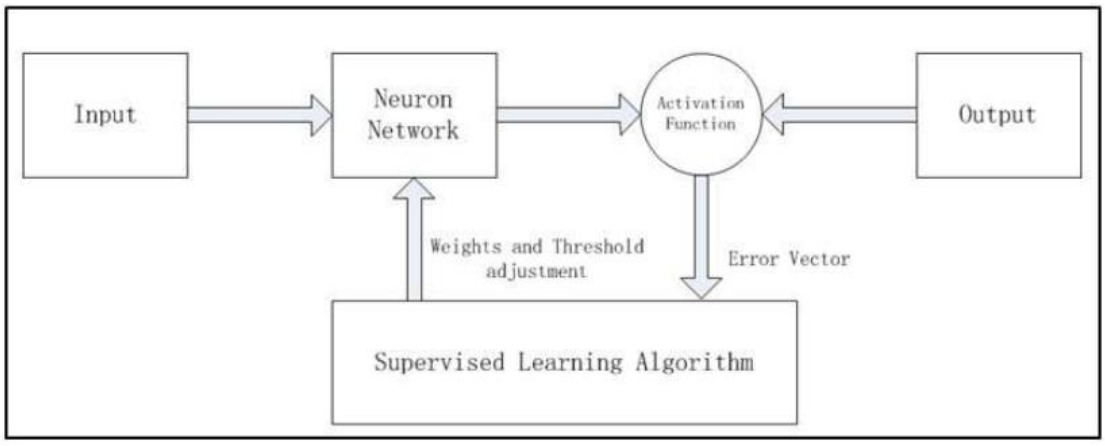


Figure 10 ; Workflow for Supervised Learning Algorithm

There are three unique kinds of malware identification systems: attack or intrusion identification, misuse recognition (signature-based) and anomaly identification (behavior based) [6]. Attack or intrusion identification tries to identify unapproved access by unauthorized people. But, misuse recognition (signature-based) tries to distinguish misuse by insiders and depicts great discovery outcome about for indicated, surely understood attacks. Clearly the main profits in identifying misuse are:

There is no such output where is no benign application is identified as malware and can recognize unauthorized entry quickly. The inconvenience isn't fit for differentiating new unauthorized entry, while these new outcomes are just slightest modification of already seen malware. Anomaly identification (behavior based) points to indentifying patterns in each dataset that do not go with regular behavior. It also makes hard to do evaluation the not so generalized behavior of framework for which security is to be provided and raise anomaly alert at the point when the variation between a provided perception at a use case and general behavior crosses a predefined threshold. The only advantageous situation is possibility to differentiate already unknown, not clearly visible intrusion incidents and loss is very cases where benign applications are marked as malware and expects an extensive setup of preparing data to build general behavior profile. In order to avoid these shortcomings of misuse recognition and inconsistency identification profiles must to be refreshed at consistent interim of time interval with the large datasets a [16]. However, a lot of the datasets additionally expands the issue of irregularity, excess, and uncertainty. A few information mining procedures have been connected for intrusion identification.

K-Mean Clustering is an unsupervised information retrieving method for intrusion detection and it is anything but difficult to execute. Three noteworthy downsides of K-mean clustering are

1. Class strength issue,

2. Force task issue, and

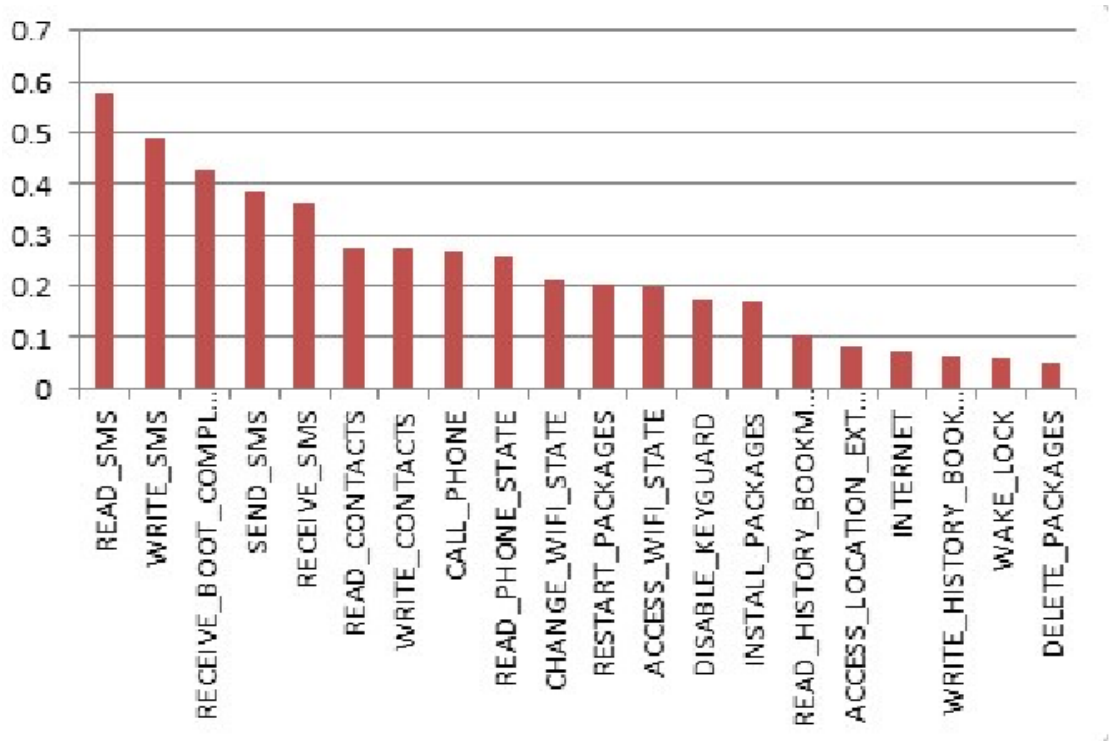
3. No class Problem.

It has been viewed that single model can't give better outcome as far as review and exactness.

CHAPTER 4: Identification of patterns in malware behavior

Client-Side application on android device lists out all the permissions declared in the manifest of each other application. Like Server side, 1 bit is assembled for permission which denotes presence (1) or absence (0) of permissions. These permissions are declared in an array in the same order as was used while creating arff file for WEKA tool classifier running on the server. Thus, for application under scan, a vector of bit values is formed containing 0 and 1. This vector is of the same length that of vector used for training classifier, but the last value is passed as a placeholder (?). This represents classification result to be returned based on input permission vector for that application.

The action of adding a new application or replaced/updated is captured by interfering the installation process of the application downloaded through play store or 3rd party marketplace. As soon as it is identified that a new application installation is about to complete, all permission declared by that applications are logged in input vector file discussed above. This fixed length vector is passed to the server for classification and server returned a single bit declaring the application as benign or malicious.



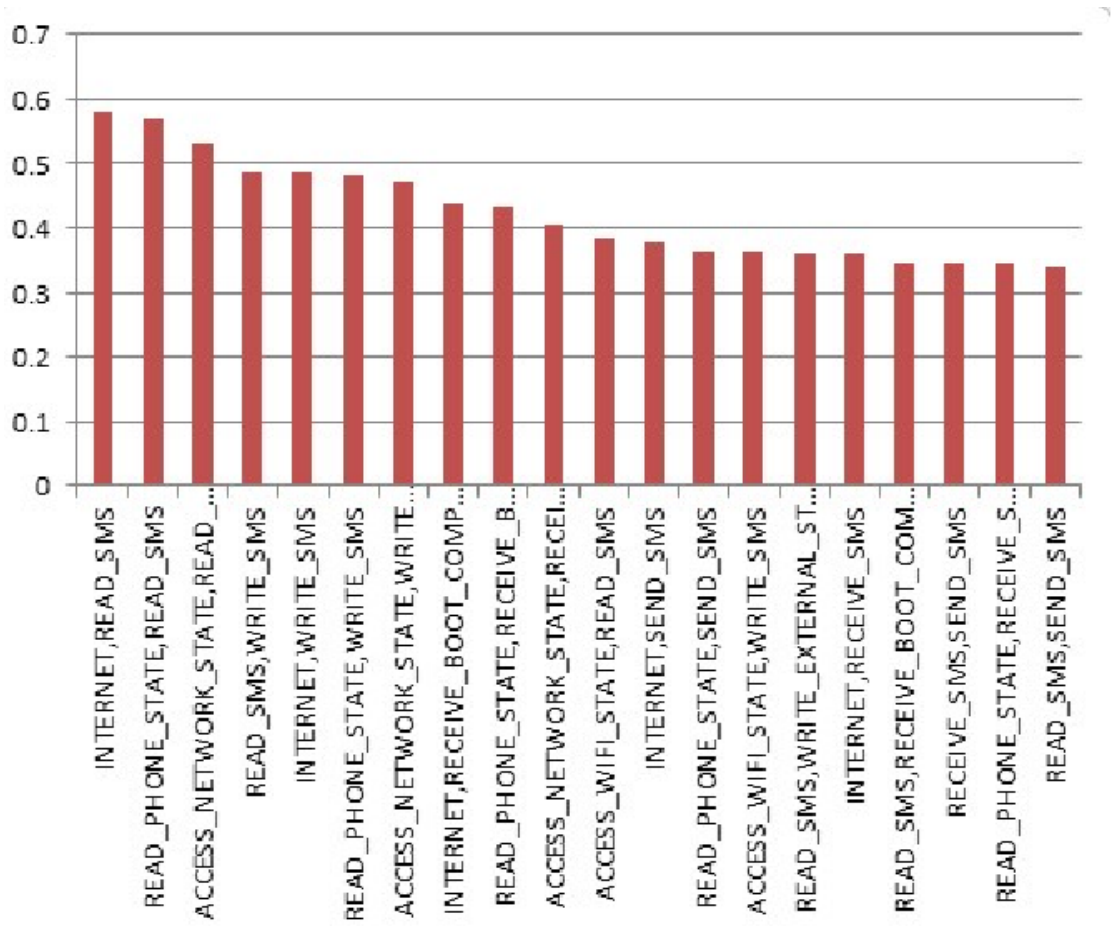


Figure 11: Top permissions accessed in dataset

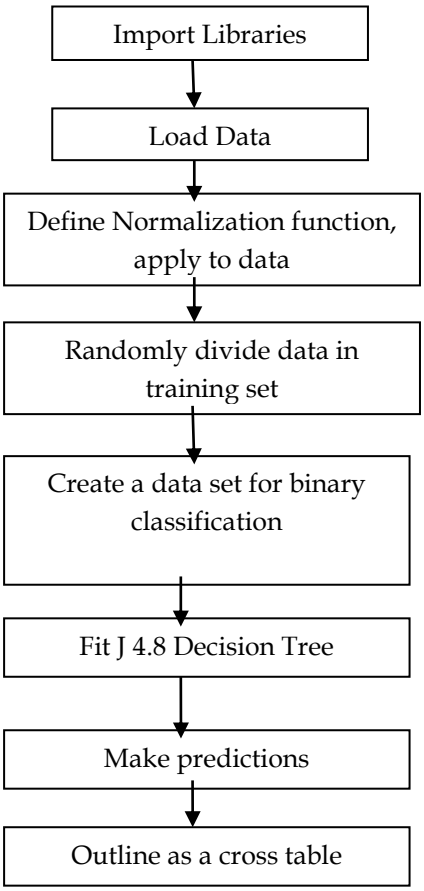


Figure 12 : Flow Diagram for proposed solution

CHAPTER 5: Results

In this work, application files are considered which is already collected and saved in the input database. This application file is considered for the experiment.

A data-set of 2000 applications are used which consists an equal amount of benevolent and harmful applications. Applications are downloaded from third party marketplace and some dangerous applications are written by declaring sensitive permissions in the manifest of sample applications and remaining from different open malware collections sites (For academic purpose).

To validate the stability of J4.8 decision tree based algorithm used for training classifier Cross fold validation technique is used. In k (=10) fold cross validation method, iterations are carried out k times. It can be illustrated as follows:

1. Firstly, data is distributed in k sets.
2. Validation is carried out k times and every time a set thus formed is treated as validation set and rest sets are used as training data,
3. Total accuracy is estimated as an average value of k times error estimates.
4. This way biasing and variance is controlled effectively in Cross fold validation technique. Empirically k is selected as 5 or 10. This thesis selects a value of 10 for the same.

```
 A decision tree generation form training records of data partition D
Algorithm : Generate_decision_tree

Input:
Data partition, D, which is a set of training records and their associated class labels.
attribute_list, the set of candidate attributes.
Attribute selection method, a procedure to determine the splitting criterion that best
partitions that the data records into individual classes. This criterion includes a
splitting_attribute and either a splitting point or splitting subset.

Output:
 A Decision Tree

Method
Generate a node N;

if records in D are all of the same class, C then
   return N as leaf node labeled with class C;

if attribute_list is empty then
   return N as leaf node with labeled
```

```
   with majority class in D;|| majority voting

apply attribute_selection_method(D, attribute_list)
to find the best splitting_criterion;
label node N with splitting_criterion;

if splitting_attribute is discrete-valued and
   multiway splits allowed then  // no restricted to binary trees

attribute_list = splitting attribute; // remove splitting attribute
for each outcome j of splitting criterion

   // partition the tuples and grow subtrees for each partition
   let Dj be the set of data tuples in D satisfying outcome j; // a partition

   if Dj is empty then
      attach a leaf labeled with the majority
      class in D to node N;
   else
      attach the node returned by Generate
      decision tree(Dj, attribute list) to node N;
   end for
return N;
```

The results are evaluated based on following standards parameters:

True positives- Benign applications that are rightly identified by the classifiers are called true positives. Let TP be the number of true positives.

True negatives- Malicious applications that are rightly identified by the classifiers are called true negatives. Let TN be the number of true negatives.

. False positives- Malicious applications that are falsely identified by the classifiers are called false positives (for example, application of benign class for which the classifier predicted malicious). Let FP be the number of false positives.

False negatives- Benign applications that are falsely identified by the classifiers are called false negatives. (for example, application of malicious class that are identified as benign by the classifier). Let FN be the number of false negatives.

Sensitivity/ true positive rate: TPR= TP/P= TP/ (TP+ FN)

Specificity or true negative rate: TNR= TN/N= TN/ (TN+ FP)

Accuracy: ACC = (TP +TN) / (TP +TN + FP +FN)

Error Rate: ER = (TP +TN) / (FP +FN+FP +FN)

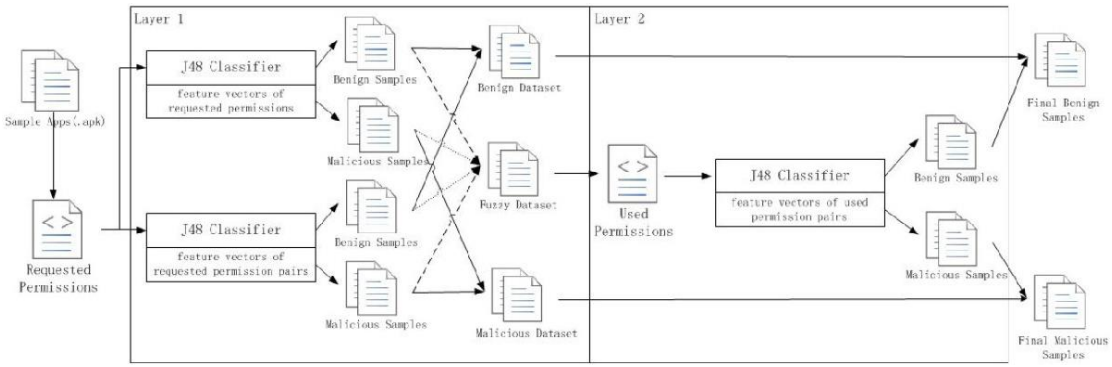


Figure 13: Overall Architecture of implementation

Our trial results demonstrate the application of utilizing classifiers to give malware detection on obscure and zero-day malware which are not identified by standard detection frameworks. This solution could recognize over 92-94% of new malware with an FPR of 1.52-3.93%. Our approach also demonstrates that a basic permission-based system might be utilized in combination with an existing solution for malware detection and in this manner, brings the security one level up for mobile devices.

```
=== Run information ===

Scheme:       weka.classifiers.rules.DecisionTable -X 1 -S "weka.attributeSelection.BestFirst -D 1 -N 5"
Relation:     permissions
Instances:    278
Attributes:   331
              [list of attributes omitted]
Test mode:    user supplied test set:  size unknown (reading incrementally)

=== Classifier model (full training set) ===

Decision Table:

Number of training instances: 278
Number of Rules : 26
Non matches covered by Majority class.
        Best first.
        Start set: no attributes
        Search direction: forward
        Stale search after 5 node expansions
        Total number of subsets evaluated: 5480
        Merit of best subset found:    0.18
Evaluation (for feature selection): CV (leave one out)
Feature set: 20,36,38,114,161,181,183,191,196,209,226,282,331

Time taken to build model: 1.54 seconds

=== Predictions on test set ===
```

Figure 14 : Results of running Classifier on full training set

```
=== Evaluation on test set ===

Time taken to test model on supplied test set: 0.15 seconds

=== Summary ===

Correlation coefficient                  0.8432
Mean absolute error                      0.1086
Root mean squared error                  0.2726
Relative absolute error                 21.7198 %
Root relative squared error             54.5151 %
Total Number of Instances              120
```

Figure 15 : Evaluation of supplied test set on train model.

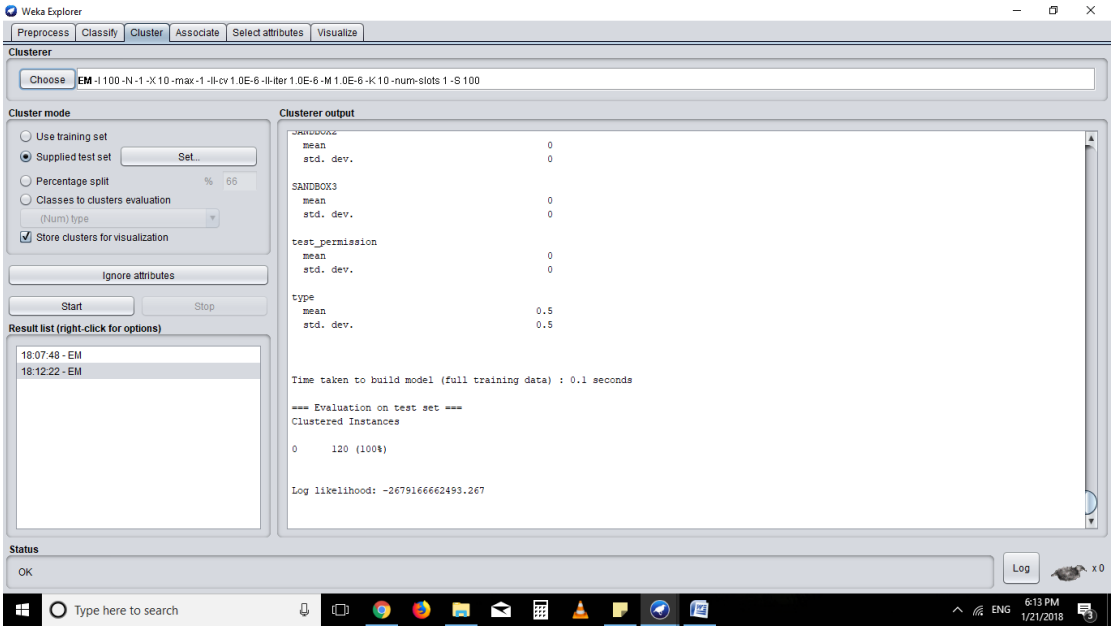


Figure 16 : Clustering results for evaluation of test data set.

CHAPTER 6: Conclusion

The effect of cell phones and mobile malware on our everyday lives can't be ignored. It is required to think regarding the computational constraints of cell phones with a specific end goal to implement a working solution. In this work, in the wake of the exponential development of the Android mobile smart phones, there is a fast increment of Android multistage malware. This work concentrates solely on how well permissions in Android are characteristic of the maliciously influenced pattern. This solution incorporates the essence of a server-side machine learning classifier that consequently distinguishes (possibly) unsafe practices of new malware with the help of permissions they require. This approach can be explored as the number of mobile malware increase and it becomes possible to obtain more permission features using many malware samples. Different classifiers will yield a more fruitful result and boosted machine learning classifiers will improve current results.

References

[1] "TechRepublic," 17 November 2017. [Online]. Available: https://www.techrepublic.com/article/new-google-play-store-malware-highlights-disturbing-trend-of-multi-stage-android-attacks/.

[2] K. L. 2017, "What is malware and how to defend against it?," [Online]. Available: https://www.kaspersky.co.in/resource-center/preemptive-safety/what-is-malware-and-how-to-protect-against-it.

[3] "Quartz Media LLC," Android just hit a record 88% market share of all smartphones, [Online]. Available: https://qz.com/826672/android-goog-just-hit-a-record-88-market-share-of-all-smartphones/.

[4] "Panda Security," 8 November 2017. [Online]. Available: https://www.pandasecurity.com/mediacenter/mobile-security/download-apps-safely/.

[5] "Strategy Analytics," 8 December 2017. [Online]. Available: https://www.strategyanalytics.com/strategy-analytics/blogs/devices/smartphones/smart-phones/2017/12/08/global-smartphone-shipment-will-grow-5-in-2018.

[6] "Strategy Analytics," Global Smartphone Installed Base by Operating System for 88 Countries, 2017. [Online]. Available: http://www.strategyanalytics.com/default.aspx?mod=reportabstractviewer&a0=7834.

[7] "IDC," 2017. [Online]. Available: https://www.idc.com/promo/smartphone-market-share/os.

[8] "McAfee Labs Threat Report," 2017. [Online]. Available: https://www.mcafee.com/us/resources/reports/rp-quarterly-threats-jun-2017.pdf.

[9] "The Growing Threat Of Mobile Device Security Breaches," April 2017. [Online]. Available: https://blog.checkpoint.com/wp-content/uploads/2017/04/Dimensional_Enterprise-Mobile-Security-Survey.pdf.

[10] S. Collett, "CSO From IDG," August 2017. [Online]. Available: https://www.csoonline.com/article/2157785/data-protection/five-new-threats-to-your-mobile-security.html.

[11] A. Felt, E. Ha, S. Egelman, A. Haney, E. Chin and D. Wagner, "Android permissions: User attention, comprehension, and behavior," in *Eighth Symposium on Usable Privacy*, 2012.

[12] P. G. Kelly, N. Sadeh, S. Consolvo, J. Jung, D. Wetherall and L. F. Cranor, "A

conundrum of permissions: installing applications on an android smartphone," *Financial Cryptography and Data Springer,* pp. 68-79, 2012.

[13] "Statista," December 2017. [Online]. Available: https://www.statista.com/statistics/266210/number-of-available-applications-in-the-google-play-store/.

[14] "Gartner Forecast," 2011-2018. [Online]. Available: https://www.gartner.com/doc/3645348/forecast-pcs-ultramobiles-mobile-phones.

[15] "Norton," 2017. [Online]. Available: https://www.nortonsecurityonline.com/security-center/mobile-threats-protection.html.

[16] Moffie, Micha, W. Cheng, D. Kaeli and Q. Zhao, "Hunting Trojan Horses. Proceedings of the 1st Workshop on Architectural and System Support for Improving Software Dependability," 2006.

[17] E. Chien, "Techniques of Adware and Spyware," 2005. [Online]. Available: https://www.symantec.com/avcenter/reference/techniques.of.adware.and.spyware.pdf.

[18] A. Chuvakin, "An Overview of Unix Rootkits," *iDEFENCE Labs,* 2003.

[19] Lopez, William, H. Guerra, E. Pena, E. Barrera and J. Sayol, "Keyloggers," *Florida International University,* 2013.

[20] Savage, Kevin, P. Coogan and H. Lau, "The Evolution of Ransomware. Symantec Corporation," 2015. [Online]. Available: http://www.symantec.com/content/en/us/enterprise/media/security_response/. [Accessed 08 December 2017].

[21] B. Baskaran and D. Ralescu, "Detection of Malicious Applications in android using Machine Learning," *university of Cincinnati,* 2016.

[22] "The University of Waikato," Attribute-Relation File Format (ARFF), [Online]. Available: http://www.cs.waikato.ac.nz/ml/weka/arff.html.

[23] "The University of Waikato," ARFF, [Online]. Available: http://weka.wikispaces.com/ARFF.

[24] Prasad, B. Jaya, H. Annagi and K. S. Pendyala, "Basic static malware analysis using open source tools," 2016.

[25] Egele, Manuel, T. Sholte, E. Kirda and C. Kruegel, "A Survey on Automated Dynamic Malware Analysis Techniques and Tools," *ACM Computing,* 2012.

[26] Harley, David and A. Lee, "Heuristic Analysis — Detecting Unknown Viruses," 2009.

[27] D. Barrera, H. G. Kayacik, P. C. v. Oorschot and A. Somayaji, "A methodology for empirical analysis of permission-based security models and its application to android," in *17th ACM conference on Computer and communications security*, 2010.

[28] Portokalidis, "Paranoid Android: Versatile Protection for Smartphones," in *ACM*, 2010.

[29] W. Zhou, Y. Zhou, X. Jiang and P. Ning, "Detecting repackaged smartphone applications in third-party Android marketplaces," in *Second ACM Conference on Data and Application Security and Privacy*, NY, USA, 2012.

[30] W. Enck, D. Octeau, P. McDaniel and S. Chaudhuri, "A study of android application Security," in *20th USENIX conference on Security*, Berkeley, USA, 2011.

[31] A. P. Felt, S. Hanna, E. Chin, H. J. Wang and E. Moshchuk, "Permission re-delegation: Attacks and defenses," in *20th Usenix Security Symposium*, 2011.

[32] "The Android Open Source Project," Application Fundamentals, [Online]. Available: http://developer.android.com/guide/components/fundamentals.html.

[33] "The Android Open Source Project," System Permissions, [Online]. Available: http://developer.android.com/guide/topics/security/permissions.html.

[34] "The Android Open Source Project," App Manifest, [Online]. Available: http://developer.android.com/guide/topics/manifest/manifest-intro.html.

[35] "The Android Open Source Project," Android Permissions, [Online]. Available: http://developer.android.com/guide/topics/security/permissions.html.

[36] "The Android Open Source Project," PackageManager, [Online]. Available: http://developer.android.com/reference/android/content/pm/PackageManager.html.

[37] "Google Play Store," Google, [Online]. Available: https://play.google.com/store.

[38] Horton, Jeffrey and J. Seberry, "Computer Viruses. An Introduction," *University of Wollongong,* 1997.

[39] Smith, Craig, A. Matrawy, S. Chow and B. Abdelaziz, "Computer Worms: Architectures, Evasion Strategies, and Detection," *Journal of Information Assurance and Security,* no. 4, 2009.

[40] Quinlan, J. Ross and Morgan Kaufmann, "C4.5: programs for machine learning," vol. 1, 1993.

[41] Leavitt, Neal, "Malicious code moves to mobile devices," *IEEE Computer,* vol. 33, pp. 16-19, 2000.

[42] "TrendLabs Security Intelligence blogs," TrendMicro, [Online]. Available: http://blog.trendmicro.com/trendlabs-security-intelligence/a-look-at-google-bouncer

/. [Accessed 8 December 2017].

[43] "Dynamic analysis tools for android fail to detect malware with heuristic evasion techniques," [Online]. Available: https://thehackernews.com/2014/05/dynamic-analysis-tools-for-android-fail.html.