CHAPTER I

INTRODUCTION

Malware is malicious software which cause harm to data devices. In our day to day life we are hearing about the malwares like worms, viruses, Trojans etc…. Now a day the sale of android devices is increased highly, this was a problem, when once malicious maker get interest in android to produce attack. The android is a open source software for smart phones lead by the market and third party sites. The mobile key features are the device that will perform a kind of application code on application in running. When a malicious program is infected to the device the user will face the issues regarding information misuses, permission etc.

The sophisticated Euro grabber attack showed that mobile malware may be a very lucrative business by stealing an estimated 36,000 from over 30,000 customers of over 30 banks in Italy, Germany, Spain and Holland. This attack worked by infecting victims PCs and mobile with a modified version of Zeus Trojan.

Android open device allows the users to install applications that don't necessarily originate from Google play store. With over one million apps obtained for download by Google authorized station, possibly million apps were spread by the third parties. Approximately 20,000 apps were releasing in every month, these apps need to be analyzed to detect and prevent from malicious activities in them.

Malware analysis is a process which uses to determine the functionality of specific malware samples. By doing the malware analysis we can know how the malware functions, how to identify it and finally how to eliminate it. Malware analysis is of two types one is Code (Static) analysis and other one is Behaviour (Dynamic) analysis. Both malware analysis techniques provide good understanding about the functioning and working of the malware. Both of the technique use different tools for analysis.

Static or code analysis **:** In this method real code of the application is examined by performing reverse engineering to find out the malicious functionality of the them. This method involves via disassemble, debugging and decompiler. Disassembler means which converts the machine language into human readable assembly kind.Decompiler means a converts the executable program (machine language) into a format understandable to software programmers. Debugger or debugging tool is a computer program that is used to test and debug the other programs. OllyDbg is a debugger that may handle by windows programs.

Behavioural analysis**:** Where the behaviour of the malware is monitored by executing during sandbox surrounding or in real time or in virtual time. The objective is to find the error while it's running. The behaviour is monitored, comparable to creation or deletion of method, adding or deleting the entries within the registers.

To perform these work investigators infects the isolate system with malicious applications and monitoring tools were used to observe the specific execution. Some of the get from online are free tools those are Process monitor, Process explorer, Regshot and Wireshark. These tools are used to monitor the behaviour of the malware.The behaviour based detection is done by comparing the threat system cell with already access malware behaviour on file system cell series. This type of detection is help to avoid by attack method. The behaviour based malware analysis will create a malicious files with similar behaviour however it has completely different feature.

I proposed behaviour based analysis system malware detection for android or smart phone devices. The system contains log collector on smart device and the log analyzer on the system. Log collector monitor all application activities on kernel layer. Kernel level logcat generates a large size file quantity of information. It contains all activates of application that runs on device; there are so much noise for actual malware detection. Thus log collector monitors the events that are valuable system calls to detect a malicious activity. The malicious activity is detected by the signature matching approach.

There are different types of malwares present which leads to perform malicious activities in the system. They are defined as follows:

APT (Advance persistent threat): APT doesn't perform any malicious activities on the system initially. They use to listen and watch on the system and gather the information as much as they can. The attacker who used to exploit the APTs takes the use of gathered information and become a threat for organization or individual.

Backdoor: Backdoor is use to perform the malicious activities i.e. to access the system without authentication or authorization. It allows the attacker to get the full access of everything on the system.

Bot: Bot can used to perform the malicious activities i.e gather the password, obtain financial information, to capture and analyze packets.

Droppers: Dropper viruses are hidden, droppers are new type of virus which cannot detect by the anti-virus easily because of they are new in nature and have specific functionality. Adropper doesn't harm the system itself. It installs other threats through which other malware can be pushed.

Malware: Malware defines a term which applies to any piece of code or program which can be harmful for user.

RAT (Remote access Trojan): RAT useful which creates backdoor in the background and provides the full unauthorized access of the effected system. It behaves same as Trojan.

Rootkit: Rootkit is malicious software which provides administrative access to an attacker. There are difference type of rootkit i.e. Kernel rootkit, firmware rootkit, application rootkit, memory rootkit and library rootkit [12]. All rootkit behaves differently but perform the same operation such as provide the administrator level access to an attacker.

Scareware: It is designed to play tricks with the users and make them to purchase and download malicious software. It generates a pop-up message which contain your system contains an information regarding malicious files and ask to remove them by downloading an application which is supposed to be malicious.

Back ground information

Android is an open source Linux Operating System for mobile devices. Android devices were established by the Open Handset Alliance, run by Google, and others companies. The goal of the Android Open Source Project is to create a successful real-world product that improves the mobile experience for the end users.

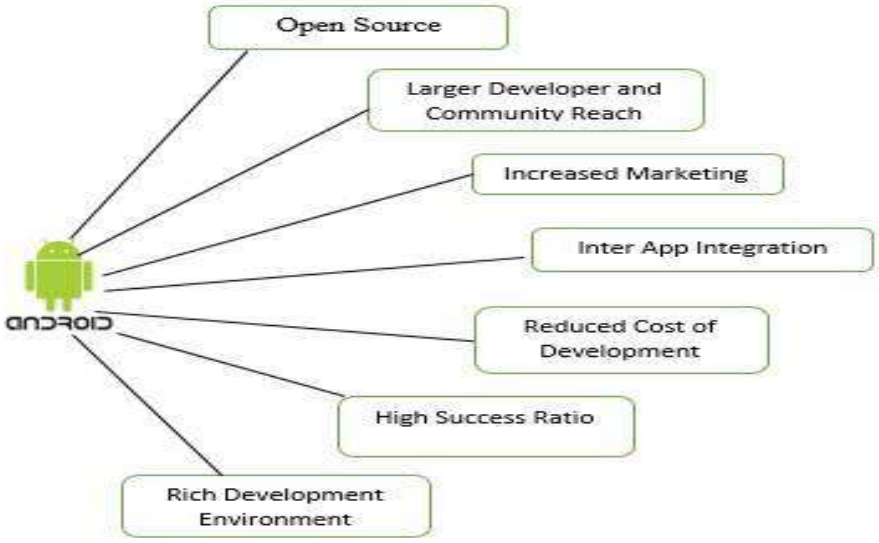

Figure 1: Android Overview

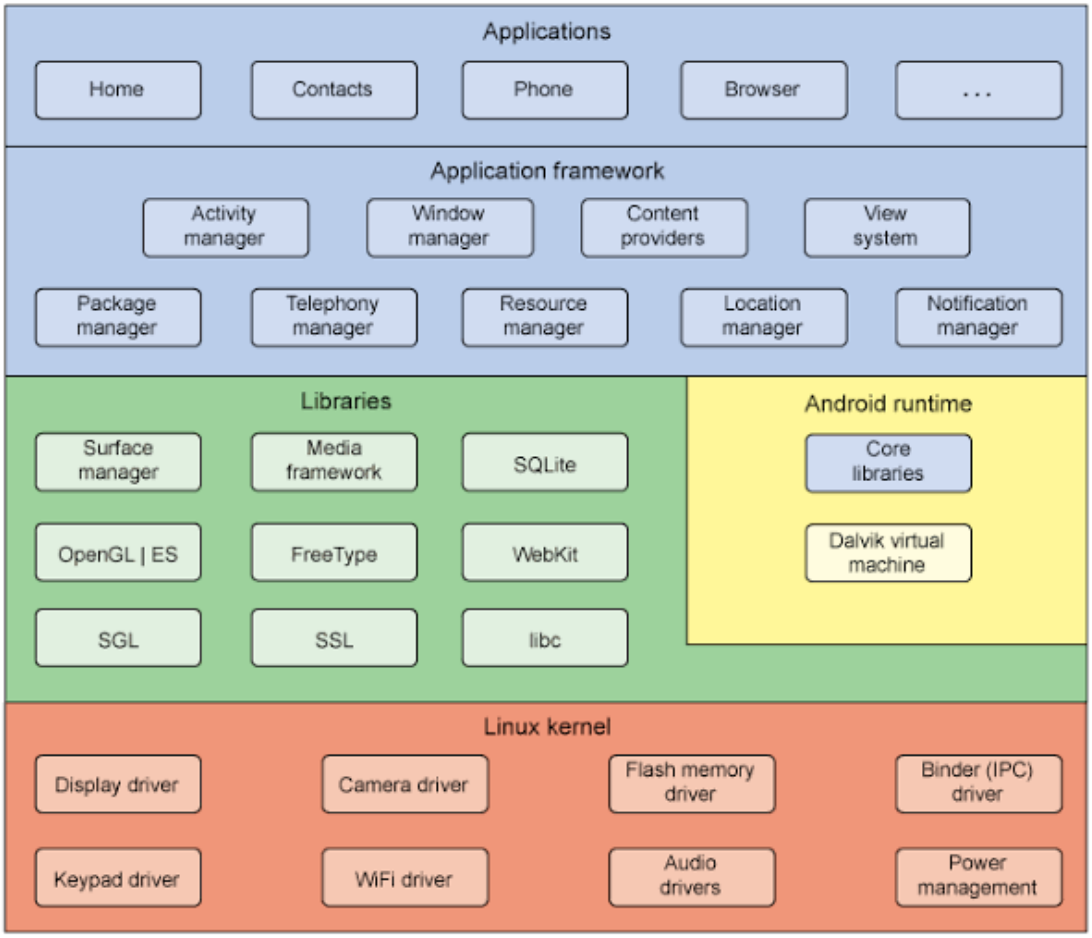

Figure 2: Android architecture [22]

Linux Kernel

- Generic System Services
    1. Permissions
    2. Memory and process management
    3. Device Drives
- Preemptive Multitasking
- Lean, effective and secure
- Open source

Android Runtime [8]

Android Runtime is an application runtime environment used by the android operating system.

- Core Libraries: provides the functionality of the JAVA Programming Language.
- Dalvik VM
    1. Executes Android Applications
    2. Each application runs within its own VM
    3. Each app is “Sandboxed”
    4. Optimization for low memory requirements
    5. Executes .dex (Dalvik-Executable) files instead of .class
    6. DX tool converts classes to .dex format

Android Libraries

Libraries carry a set of instructions to guide the device in handling different types of data. For case, the playback and recording of many audio and video formats are showed by the Media Background Library.

Application Framework

- Higher Level Services to Application
- Environment in which application are run and manage

1. android.app: Carries contact to the appliance good and is that the foundation of all android applications.

2. android.content: Simplifies content admittance, publication and electronic messaging between applications and application working

Application

- This is where our applications are placed.
- Pre-installed applications are there like web browser, contact messages, dialer and SMS client apps.

Android kernel and System calls

To reiterate, the android device can be changed Linux kernel at the core. The change in Linux kernel is in serious worry adjusting this software for the devices. The bodymaker, specific kernel improvement involves management of power, memory shared drivers, binders, alarm driver, kernel, feller and low memory killers. System calls are used by android applications to allow the services of the kernel. While the actives the manager call directions, there is a switch from user mode to kernel modeto perform the penetrating operations. The system calls are theedge between user and kernel. This means all requests from the applications can call edgebefore its Show through the hardware. So, capturing and analyzing the system call cangive information about the application

Behaviour-Based Malware Detection techniques Tools

The behavioural analysis examines the malware specimen's interactions with environment: the file system, the registry, the network, as well as other processes and OS components. We need to note the interesting behavioural characteristics. When performing a behavioural analysis look at how the malware behaves and what changes the malware makes on a baseline system. We need remember that when performing behavioural analysis the malware lab should not connected to any other network.

1. Crowdroid: Behaviour-based malware detection [1]

The process malware detection from device is a new idea. Then they definite to make the all analysis technique on a possessed remote server and that server was used fully to collect information and spot malware and suspicious apps within android. The framework was collected of several mechanisms which provide sufficient resource and instruments to notice malware on the Android platform. First, they had established a consumer referred to as Crowdroid, which may be taken and fitted from Google's Market. This application was in the responsibility of 24-hour care Linuxkernel level system calls and transfers them pre-processed to a central server. The crowdsourcing, users were help with transfer no personal, besides the data behaviour of every application they are use. These application could had been downloaded two side from one is android market and other is unauthorized sources.

Now the remote server will analyze the application and detect the presence of malware. After that it will separate the system call and malicious applications. Crowdroid framework building designedtrustson three components: data acquisition, data manipulation, and the malware analysis and detection system.

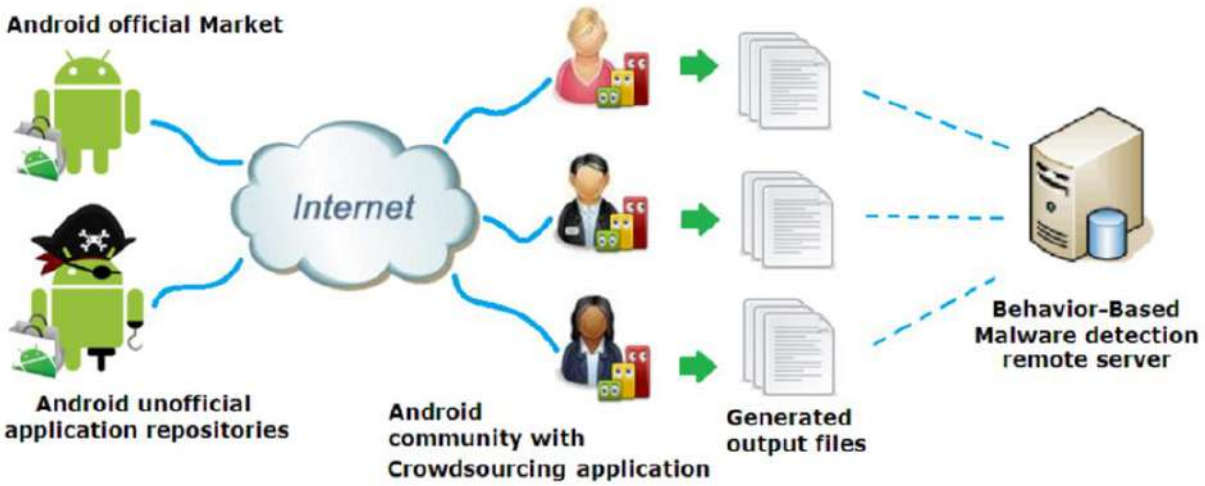

Figure 3: Behavior-Based Malware Detection Framework

2. MADAM (Multi-level Anomaly Detector for Android Malware) Malware Detection [14]

MADAM concurrently monitors Android at the kernel-level and user level to detect real malware infections, using machine learning techniques to distinguish between standard behaviour and malicious behaviour. To detect the different misbehaviour i.e. known misbehaviour pattern and anomalies, authors have defined two distinct sets of features i.e. Per-App Monitor and Global Monitor, which are used. They were also able to identify more complex misbehaviour whose detection is based on the features analyzed by both the components.

Global Monitor generates an alert. Pre-App monitor removes the malicious Apps. Per-AppMonitor is launched in parallel with Global Monitor to detect and stop known behavioural patterns along with kernel and API features. It detect the behavioural pattern continuously, Signature-based Detector runs in the background. Misbehaviour is blocked by Per-App Monitor by locating the responsible app in the Suspicious List

3. Kernel - based behaviour malware detection system [2]

Kernel based behaviour analysis system detect malware activities of the application. There are two components of the system one is smart device and another one is machine server. Smart device is used to collect s logcat data of running applications. After collecting the logcat data it is send to the machine server for detection.

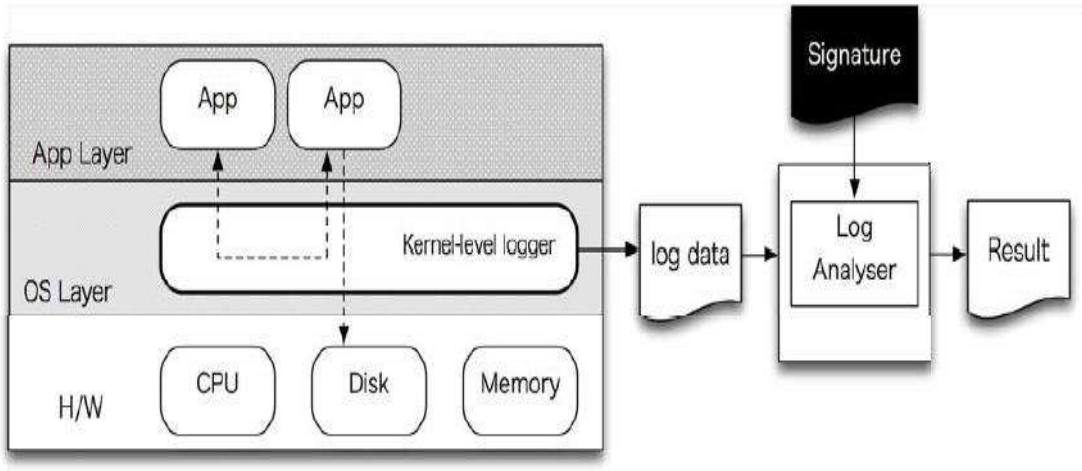

Figure 4: Kernel - based behaviour malware detection system

4. MONET: Behaviour based detection for android [14]

Monet is a behavior based detection for android, and monitors the installation events in the background. It extracts the static information including component information from the apps manifest file. Before the launching the application a behavior based graph is generated by MONET which is based on static structure of the app. Once the app is launched MONET starts to monitor and collect runtime information along with system calls. If an intrusive action is detected by the system, a pop-up is generated mentioning the warning about the suspicious action. For further analysis, MONET user both the suspicious system call and runtime behavior graph as the runtime behavior signature to the

Backhand detection server.The MONET backenddetection server matches the uploaded signature with existing malware signatures in thedatabase, and provides the results to the mobile device and notifies users about the detection result.

5 DroidLogger Reveal suspicious behaviour of Android applications [16]

DroidLogger consists of several components: a decompiler, a translator, an instrumentation tool and a compiler, which are shown in Figure7. Intermediate Language (IL) instrumentation is used because dex syntax is hard to understand. Decompiler is the response to extract the apk file, translates the dex file into IL. The translator translates the java code into the IL. Data is read by the Instrumentation tool from suspicious API profile. After instrumentation; IL is complied into a dex file and signed it again. Then, emulator is used to test the instrumented apk and get its log file, extracts the suspiciousAPIs and their arguments. There are different components which are following Decompiler, Translator, Instrumentation tool, Compiler.

1. Decompiler

When performing the instrumentation, each file should be traversed to use the code as IL. For decompilation, authors used APKtool, using which apk file decompiled into different smali code.

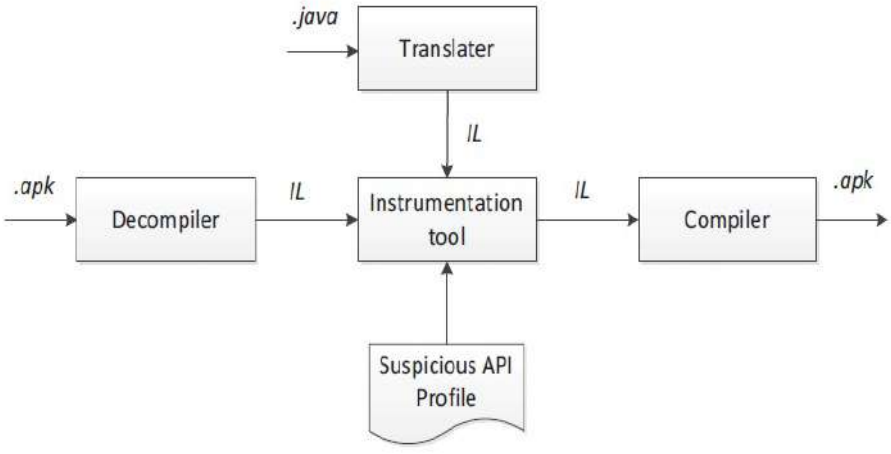

Figure 5: System Architecture of DroidLogger

2. Translator

Translator was designed by the authors to translate the Java code into IL code automatically as it is very hard to write the IL code directly.

3. Instrumentation tool

Instrumentation tool is used to fetch the translated code, and then traverse all the IL files and perform the searching for the suspicious APIs and for each of them insert the corresponding code.

4. Compiler

The compiler compiles the instrumented IL code into a .class file and then repackage to the .apk file. It should be noted that we cannot run the repackaged apkdirectly because we already broke the signature mechanism of the original apk.

Malware Distribution

A problem with third-party marketplace is the lack of accountability. There are developers who used to create poor and unreliable applications then the attacker can be pushed malicious applications to the third party marketplace. Many of the users who download it in their systems and execute them and become a victim. According to Juniper Networks, malicious applications often create from these marketplaces. Any technique invented to prevent the malicious behaviour will never reach to the android user until they buy new android mobile. New Android publications also come with bug fixers. These activities are also used to find a higher privilege level without a user's permission by the malware. During installation this approach allows the malware to request some permission to the device but provides the access to the whole device once started in the device.

CHAPTER II

LITERATURE REVIEW

TakamasaIsohara et.al (2013) studied Kernel-based Behavior Analysis for Android Malware Detection.They have researched on most major threat of Android users is malware infection via Android application markets. In case of the Android Market, as security inspections are not applied for many users have uploaded applications. Therefore, malwares, e.g., Geimini and Droid Dream will attempt to leak personal information, getting root privilege, and abuse functions of the smart phone. An audit framework called log cat is implemented on the Dalvik virtual machine to monitor the application behavior. However, only the limited events are dumped, because an application developers use the log cat for debugging. The behavior monitoring framework that can audit all activities of applications is important for security inspections on the market places. In this paper, we propose a kernel-base behavior analysis for android malware inspection. The system consists of a log collector in the Linux layer and a log analysis application. The log collector records all system calls and filters events with the target application. The log analyzer matches activities with signatures described by regular expressions to detect a malicious activity. Here, signatures of information leakage are automatically generated using the smart phone IDs, e.g., phone number, SIM serial number, and G mail accounts. We implement a prototype system and evaluate 230 applications in total. The result shows that our system can effectively detect malicious behaviors of the unknown applications. In this research they have conclude that, they have proposed a kernel-based behavior analysis for Android malware inspection. The system collects system call events generated at application runtime and analyzes it. The system achieves collecting log data that only contains data of target activities. Log data is analyzed by signature-based pattern matching. We define three categories of threats and generate signatures correspond to each category of threat. We have implemented prototype system and analyze total 230 application collected from a marketplace or voluntary web sites. The system detects 37

applications that leak some kind of personal sensitive data, 14 applications that execute exploit code and 13 destructive applications. In other words, kernel-based behavior analysis can be applied for security inspections for Android application markets.

Daniel Arp1 et.al (2014) studied DREBIN (Effective and Explainable Detection of Android Malware in Your Pocket)**.**They have completed researched on malicious applications pose a threat to the security of the Android platform. The growing amount and diversity of these applications render conventional defences largely ineffective and thus Android Smartphone often remains unprotected from novel malware. In this paper, we propose DREBIN, a lightweight method for detection of Android malware that enables identifying malicious applications directly on the Smartphone. As the limited resources impede monitoring applications at run-time, DREBIN performs a broad static analysis, gathering as many features of an application as possible. These features are embedded in a joint vector space, such that typical patterns indicative for malware can be automatically identified and used for explaining the decisions of our method. In an evaluation with123, 453 applications and 5,560 malware samples DREBIN outperforms several related approaches and detects 94% of the malware with few false alarms, where the explanations provided for each detection reveal relevant properties of the detected malware. On five popular Smartphone, the method requires 10 seconds for an analysis on average, rendering it suitable for checking downloaded applications directly on the device. In this research they had concluded that, android malware is a new yet fast growing threat. Classic defences, such as anti-virus scanners, increasingly fail to cope with the amount and diversity of malware in application markets. While recent approaches, such as Droid Ranger and AppPlayground, support filtering such applications off these markets, they induce a 12 run-time overhead that is prohibitive for directly protecting Smartphone.

Dong-Jie Wu et.al (2012) studied DroidMat: Android Malware Detection through Manifest and API Calls Tracing. They have said that recently, the threat of Android malware is spreading rapidly, especially those repackaged Android malware. Although understanding Android malware using dynamic analysis can provide a comprehensive view, it is still subjected to high cost in environment deployment and manual efforts in investigation. In this study, we propose a static feature-based mechanism to provide a static analyst paradigm for detecting the Android malware. The mechanism considers the static information including permissions, deployment of components, Intent messages passing and API calls for characterizing the Android applications behavior. In order to recognize different intentions of Android malware, different kinds of clustering algorithms can be applied to enhance the malware modeling capability. Besides, we leverage the proposed mechanism and develop a system, called Droid Mat. First, the Droid Mat extracts the information (e.g., requested permissions, Intent messages passing, etc) from each application's manifest file, and regards components (Activity, Service, Receiver) as entry points drilling down for tracing API Calls related to permissions. Next, it applies K-means algorithm that enhances the malware modeling capability. The numbers of clusters are decided by Singular Value Decomposition (SVD) method on the low rank approximation. Finally, it uses KNN algorithm to classify the application as benign or malicious. The experiment result shows that the recall rate of our approach is better than one of well-known tool, Androguard, published in Black hat 2011, which focuses on Android malware analysis. In addition, Droid Mat is efficient since it takes only half of time than Androguard to predict 1738 apps as benign apps or Android malware. They have concluded that we proposed *DroidMat*, a novel approach to distinguish and detect Android malware with different intentions. *DroidMat* has the following properties. *Effectiveness:* it is effective, that is, it is able to distinguish variant of Android malware between distinct purposes of them. it achieves up to 97.87 percentage points in accuracy. Scalability: It is scalable, that is, it is linear in the size of the problem (i.e., the number of non-zeros in the input matrix). *Efficiency:* it does not need to dynamically investigate the Android application behavior from the sandbox or by emulation, which saves the cost in environment deployment and manual efforts in investigation.

KabakusAbdullahTalhaaet.al (2014) studied APK Auditor: Permission-based Android malware detection system. They have said that aandroid operating system has the highest market share in 2014; making it the most widely used mobile operating system in the world. This fact makes Android users the biggest target group for malware developers. Trend analyses show large increase in mobile malware targeting the Android platform. Android's security mechanism is based on an instrument that informs users about which permissions the application needs to be granted before installing them. This permission system provides an overview of the application and may help gain awareness about the risks. However, we do not have enough information to conclude that standard users read or digital investigators understand these permissions and their implications. Digital investigators need to be on the alert for the presence of malware when examining Android devices, and can benefit from supporting tools that help them understand the capabilities of such malicious code. This paper presents a permission-based Android malware detection system, *APK Auditor* that uses static analysis to characterize and classify Android applications as benign or malicious. *APK Auditor* consists of three components: (1) A signature database to store extracted information about applications and analysis results, (2) an Android client which is used by end-users to grant application analysis requests, and (3) a central server responsible for communicating with both signature database and Smartphone client and managing whole analysis process. To test system performance, 8762 applications in total, 1853 benign applications from Google's Play Store and 6909 malicious applications from different sources were collected and analyzed by the system developed. The results show that *APK Auditor* is able to detect most well-known malwares and highlights the ones with a potential in approximately 88% accuracy with 0.925 specificity.

Andrea Saracinoet.al (2016) studied MADAM: Effective and Efficient Behaviours-based Android Malware Detection and Prevention. They have said that aandroid users are constantly threatened by an increasing number of malicious applications (apps), generically called malware. Malware constitutes a serious threat to user privacy, money, and device and file integrity. In this paper we note that, by studying their actions, we can

Classifymalware into a small number of behavioral classes, each of which performs a limited set of misbehaviors that characterize them. These misbehaviours can be defined by monitoring features belonging to different Android levels. In this paper we present MADAM, a novel host-based malware detection system for Android devices which simultaneously analyzes and correlates features at four levels: kernel, application, user and package, to detect and stop malicious behaviors. MADAM has been specifically designed to take into account those behaviors that are characteristics of almost every real malware which can be found in the wild. They have concluded that, MADAM detects and effectively blocks more than 96 percent of malicious apps, which come from three large datasets with about 2,800 apps, by exploiting the cooperation of two parallel classifiers and a behavioral signature-based detector. Extensive experiments, which also includes the analysis of a tested of 9,804 genuine apps, have been conducted to show the low false alarm rate, the negligible performance overhead and limited battery consumption.·

Latika Singhet.al (2017) studied Dynamic behavior analysis of android applications for malware detection. Android id the most popular operating system for the Smartphone and small devices with 86.6% market share (Chau 2016). Its open source nature makes it more prone to attacks creating a need for malware analysis. Main approaches for detecting malware intents of mobile applications are based on either static analysis or dynamic analysis. In static analysis, apps are inspected for suspicious patterns of code to identify malicious segments. However, several obfuscation techniques are available to provide a guard against such analysis. The dynamic analysis on the other hand is a behaviour-based detection method that involves investigating the run-time behaviour of the suspicious app to uncover malware. The present study extracts the system call behaviour of 216 malicious apps and 278 normal apps to construct a feature vector for training a classifier. Seven data classification techniques including decision tree, random forest, gradient boosting trees, k-NN, Artificial Neural Network, Support Vector Machine and deep learning were applied on this dataset. Three feature ranking techniques were used to select appropriate features from the set of 337 attributes (system calls). These techniques of feature ranking included

informationgain, Chi-square statistic and correlation analysis by determining weights of the features. They have concluded that, after discarding select features with low ranks the performances of the classifiers were measured using accuracy and recall. Experiments show that Support Vector Machines (SVM) after selecting features through correlation analysis outperformed other techniques where an accuracy of 97.16% is achieved with recall 99.54% (for malicious apps). The study also contributes by identifying the set of systems calls that are crucial in identifying malicious intended of malicious apps.

CHAPTER III

AIM & OBJECTIVES

AIM

To detect the malicious android applications and to detect which application is causing what type of threat.

OBJECTIVES

1. To determine types of threats
2. To identify the activities/behaviour of threat
3. To identify the malicious software on android
4. To determine how malicious applications are installed in the android
5. To identify which type of information they are carrying
6. To determine how third party applications are working

CHAPTER IV

MATERIALS & METHODOLOGY

Materials

1. Tools: Crowdroid: Behaviour-based malware

MADAM (Multi-level Anomaly Detector for Android Malware) Malware

Detection

2. Logcat Data

Methodology

Collecting Logcat data, executing the tools and conducting the analysis Logcat data with the help of tools.

Android Logs

Log in a simple language can be defined as record of events. Android logging system provides a mechanism for collecting and viewing system debug output. Logcat dumps a log of system message.

- Log.e display error message
- Log.w display warnings
- Log.d is used to log debug message
- Log.i display debug message
- Log.v display all log message

Android Malware

Here we are going to see how malware is installed on user device, how malware is activated and which type of risk threat related with them. Let here we see talk about malware installation method, their activation and malicious type activation perform.

Installation methods

Android malware families are categorized by their installation method on user's device. The following methods are used by the android malicious application for its successful installation.

- Repack
- Drive -by-Download attack
- Standalone
- Misleading Disclosure
- Phishing Scams

Behaviours-Based Android Malware

System building

To disclose unknown threat in an application first performed effective dynamic detect of activities of application. The idea of a behaviour based android malware detection system to represent the malicious applications. Android system components are smart device and commands. Smart device are used to collect the logcat data from running application. After the collection of logcat data is stored on the pc for analysis.

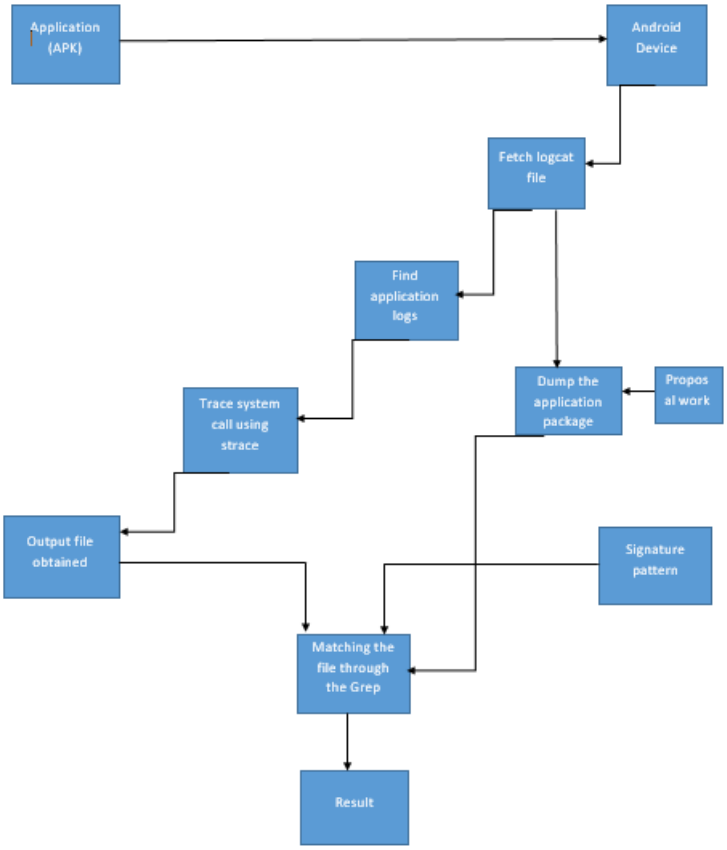

Figure6: System Architecture

Collection of logcat data

I collected the logcat data for analysis of malware activities. The collect logcat consists of big amount of useful content along with some noise content.

The collected logcat consists of kernel data. The kernel data was extracted by using adb commands.

Command 1: adb device

Command 2: adb install (we will install sdk or apk file regarding adb shell)

Command 3: adb pull

Command 4: adb push

Command 5: **get-serialno**

Command 6: adblogcat

Command 7: adb get-state

Command 8: adbbugreport

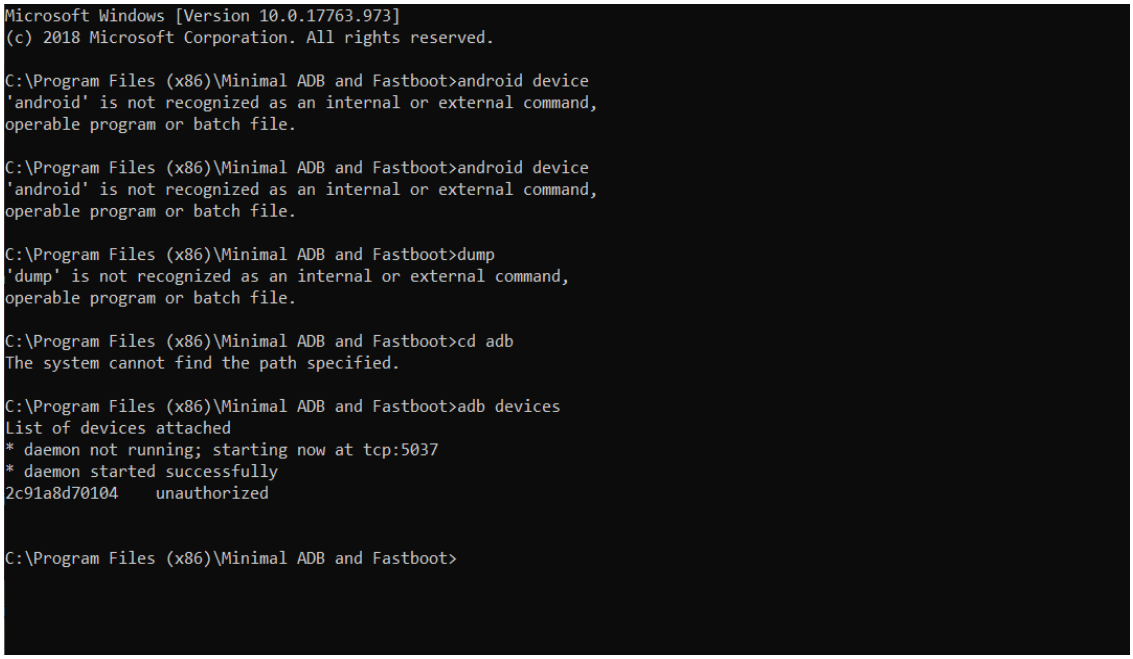

Figure 7: Device attached

Figure 8: adb pull

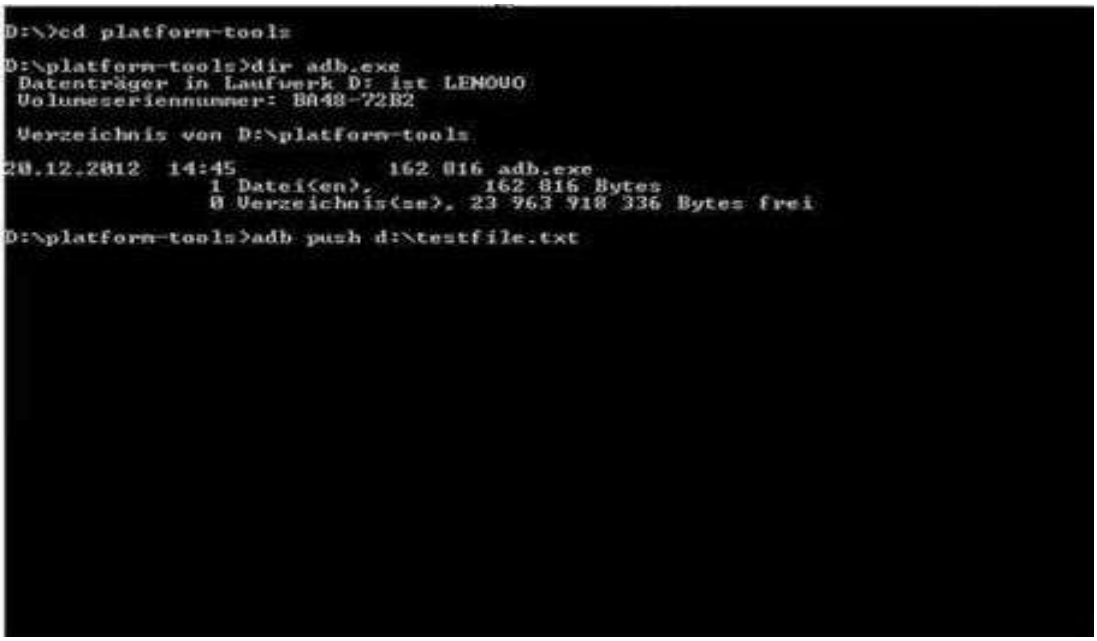

Figure 9: adb push

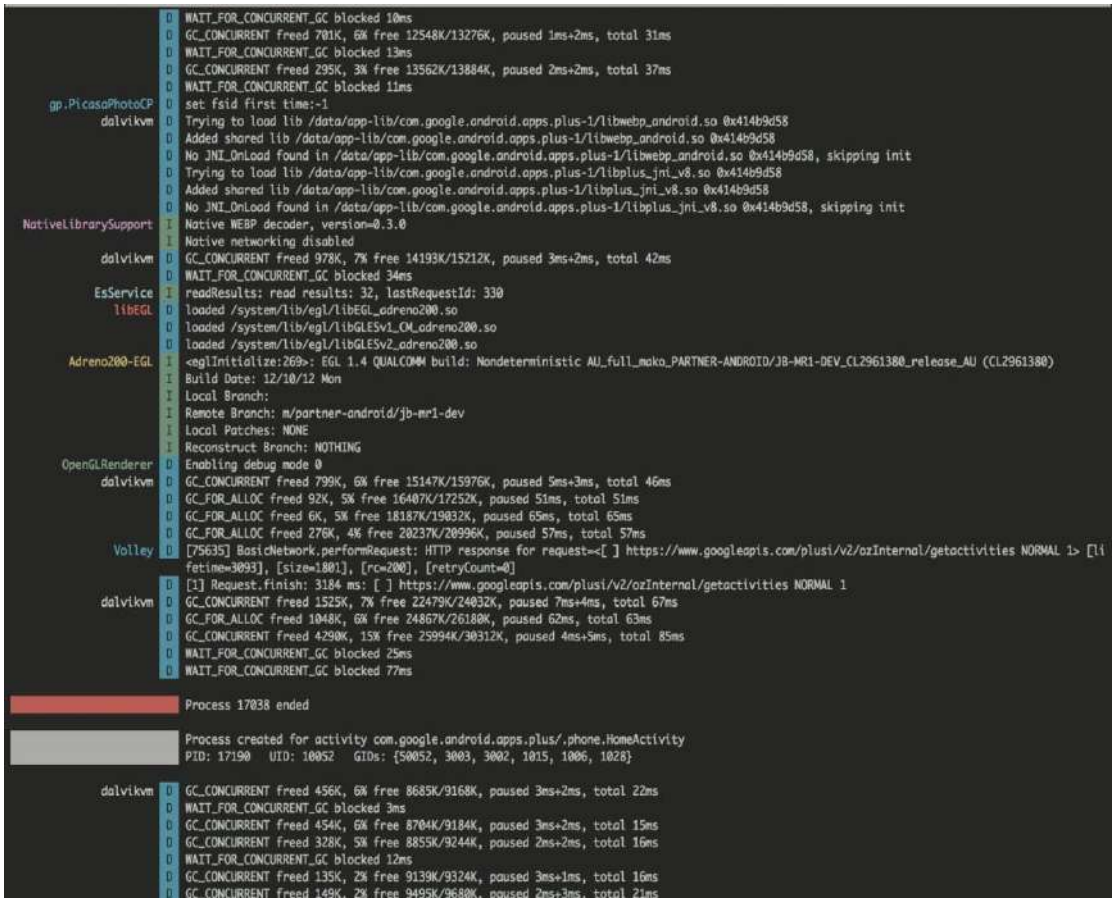

Figure 10: adblogcat

Here I successfully collected the logcat data. The collected logcat data stored for further analysis. Before the performing the tools I removed the false and negative errors for detection of malware activities.Whenever the instruction set of the systems are unsatisfactory these will be a reason for negative false high. I have reduced these false and negative errors by writing the code correctly. As I removed the errors, then I have performed the execution of tools.

Execution of tools

Crowdroid: Behaviour-based malware

Step1: Install the tool

Step 2: Crowdroid users will get connected to remote server

Step 3: Data is captured from users through remote server

Step 4: Upload the data

Step 5: click ok for analysis

Step 6: The malicious application along with sensitive behaviour and features are listed out

Category	Sensitive behavior	Feature
Android API	Send text message	com.android.internal.telephony.ISms.sendText()
	Make phone call	com.android.internal.telephony.ITelephony.call()
	Set phone state	com.android.internal.telephony.ITelephony.silenceRinger()
	Get IMEI	com.android.internal.telephony.IPhoneSubInfo.getDeviceId()
	Get cell information	com.android.internal.telephony.ITelephony.getNeighboringCellInfo()
	Get GPRS state	com.android.internal.telephony.ITelephony.isDataConnectivityPossible()
	Set GPRS	com.android.internal.telephony.ITelephony.enableDataConnectivity()
	Get Wi-Fi state	android.net.wifi.IWifiManager.getWifiEnabledState()
	Set Wi-Fi	android.net.wifi.IWifiManager.setWifiEnabled()
	Get network state	android.net.IConnectivityManager.getActiveNetworkInfo()
	Get Bluetooth state	android.bluetooth.IBluetooth.getState()
	Set Bluetooth state	android.bluetooth.IBluetoothManager.enable()
	Get location	android.location.ILocationManager.getLastLocation()
Shell command	Switch to super user	su
	Escalate privilege	chmod, chown

Figure 11: Crowdroid: Behaviour-based malware

MADAM

Step 1: Install the tool

Step 2: Check for pre app monitor (blocks the suspicious behaviour).

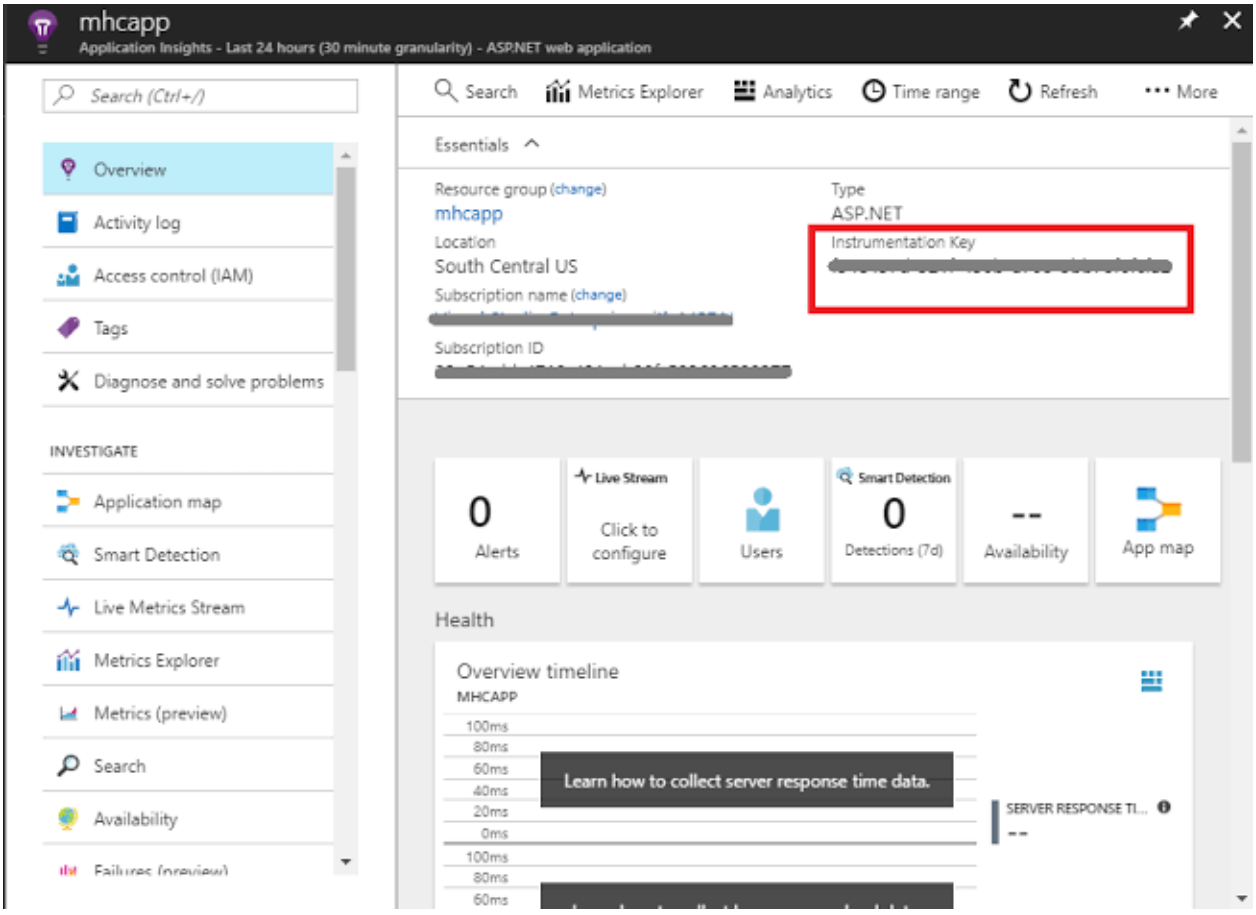

Figure 12: Pre app monitor

Step 4: Then check forGlobalmonitor (detects the suspicious applications)

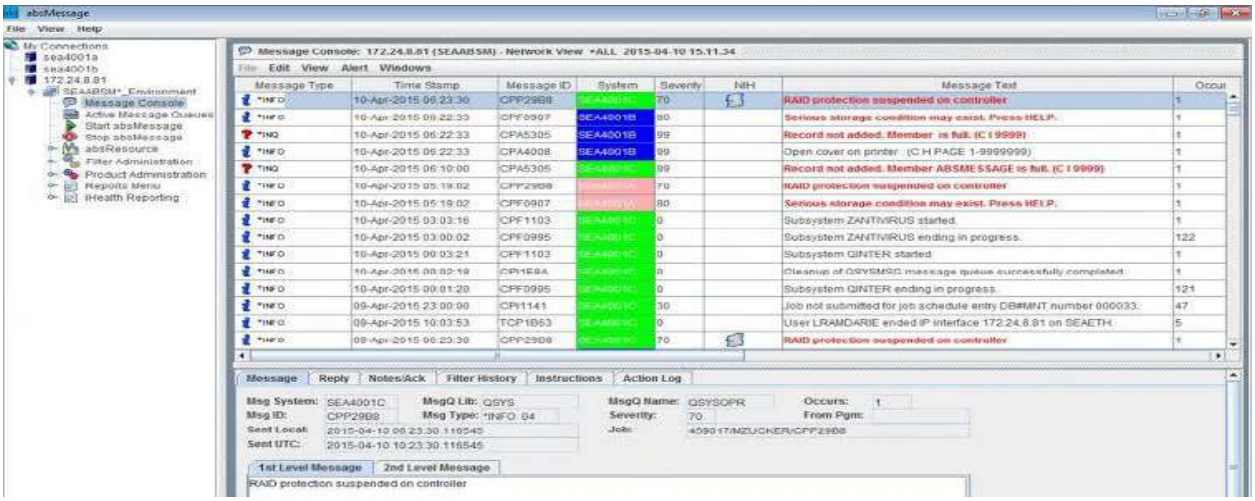

Figure 13: Goble monitor

Step 5: It detect and list out the malicious application and the application carrying information like whether it belongs to application level or kernel level of data, group, features, description and targeted misbehaviour.

Level	Group	Feature	Description	Targeted Misbehavior
Kernel	Sys Calls	`open, read, ...`	System calls concerning file and inter-component communication	Sudden and unmotivated activity increase
Application	SMS	Number of SMS (SMS Num)	Amount and recipient of outgoing SMS	Unsolicited outgoing message
Application	SMS	Suspicious SMS (SMS Susp)	Amount of SMS sent to recipients not in contacts	Spyware or registration to premium services
Application	Critical API	Administrator App	Verify if an app attempts to get admin privileges	Apps which attempt to take control of the device
Application	Critical API	New App Installation	Verify if an app attempts to install a new one	Unauthorized app installation
Application	Critical API	Process List	Verify if an app generates high number of processes	Buffer overflow (Rootkit) attac
Application	Critical API	Critical SysCalls	Amount of critical system calls generated by an app	Apps that access files and resources in backround (Spyware, Botnet and Trojan)
Application	Critical API	SMS Default App	Check default SMS manager	Unsolicited outgoing SMS
Application	Critical API	Foreground App	Check which app is interacting with user	Unsolicited SMS and preventir user from interacting with the device (Ransomware)
User	User Activity	User Presence	If the user is interacting with the device	Unsolicited activities of Spywa Botnet, Installer and Rootkit
User	User Activity	On Call	Verify if a phone call is ongoing	Unsolicited activities of Spywa Botnet, Installer and Rootkit
User	User Activity	Screen On	Verify if the device screen is on	Unsolicited activities of Spywa Botnet, Installer and Rootkit
Package	App Metadata	Permissions requested (`manifest.xml`)	Riskiness of app	Suspicious requests of dangerous permissions
Package	App Metadata	Market Info (User scores, ...)	Popularity of app	Trojan

Figure 14: MADAM level analysis and feature

After completion of executing the tools there I detected sensitive information of user, and then I select a sensitive id and smart device id.These contains advises such as number of telephone, mail details, device id, IMEI,SIM number, imsi. Besides, examined that some devices advertising features collect privacy personal data.

CHAPTER V

OBSERVATION

Signature

To clarify the behaviour of every application assessment has done on the device or system and it captured 30signature patterns. The signatures can be divided into four categories based on malicious activities i.e. Information leakage, Jailbreak, Abuse Root, Critical Permission. The table list signature describes for each malicious activity.

Table 1: Signature Pattern for Malware Detection

s.no	Signature	Threat	Description
1	ˆclone(.*jCLONE FILESj.* $	Files Misuse	Cloning the files
2	LeadBolt	Information Leakage	Mobile advertising network that provides efficient methods for both advertiser and developers to promote their app and generate revenue for it.
3	QuattrowirelessSDK	Information Leakage	A global mobile advertising company, enables advertiser and publishers to reach and engage their target audiences
4	(AdMobjAdMobjadmob)	Information Leakage	It is a mobile advertising network that helps the app developers to monetize and promote their mobiles and tabletsAppwith advertisement.

5	AdWhirl	Information Leakage	It is an advertising platform for iPhone applications that allow developers to switch between ad networks on-the-fly.
6	Flurry Agent	Information leakage	The Flurry Android Analytic Agent allows you to track the usage and behaviour of your android application on users 'devices for viewing the Flurry Analytics system.
7	ˆrecv(.*(%IMEI%j%ANDROID%).*$	InformationLeakage	Receive IMEI and device id which is a unique number
8	ˆrecv(.*(%TEL%j%MAIL%).*$	Information Leakage	Receive emails details
9	ˆrecv(.*(%IMSI%j%SIM%).*$	Information leakage	Receive IMSI and SIM number
10	ˆrecvfrom(.*MSG DONTWAIT.*	Information Leakage	Receive message by unknown socket
11	ˆ(execvejread)(.*/asroot	JailBreak	It allows to run a command as super user
12	ˆ(execvejread)(.*/rageagainstthecage	JailBreak	It used to root the device
13	ˆ(execvejread)(.*/exploit	JailBreak	Make full use of device
14	ˆ(execvejread)(.*/gingerbreak	JailBreak	Application use to root the device
15	ˆexecve("/(system/)?(binjsbinjxbin)su",	Abuse Root	Root the device by path
16	ˆopen(".*/iptables",	Abuse Root	It is an administration tool forIPv4 packet filtering and NAT

17	ˆsuperuser(n....)?	Abuse Root	It is a special user account used for system administration
18	ˆexecve(”.*/busybox”,	Abuse Root	Command line tool in a single binary
19	ˆexecve(”.*/(chmodjchownjlsnbj mkdirjrmdir)”,	Abuse Root	Use for change directories
20	android.permission. SEND SMS NO CONFIRMATIONS	Critical Permission	Allows an application to send SMS messages
21	android.permission. WRITE SMS	Critical Permission	Allows an application to write the SMS
22	android.permission. READ LOGS	Critical Permission	Allows an application to read the low-level system log files
23	android.permission. MOUNT UNMOUNT FILESYSTEM	Critical Permission	Allows mounting and unmounting file systems for removable storage.
24	android.permission.INSTALL PACKAGE	Critical Permission	Allows an application to install packages
25	android.permission. READ HISTORY BOOKMARKS	Critical Permission	Allow an application to Read browser history
26	android.permission. WRITE HISTORY BOOKMARKS	Critical Permission	Allow an application to Write in browser history
27	android.permission. READ PHONE STATE	Critical Permission	Allows read only access to phone state, including the phone number of the device, current cellular network information, the status of any ongoing calls, and a list of any Phone Accounts registered on the device
28	android.permission. READ SMS	Critical Permission	Allows an application to send SMS messages.

29	android.permission. RECEIVE SMS	Critical Permission	Allows an application to receive SMS messages
30	android.permission. READ CONTACTS	Critical Permission	Allows an application to read the user's contacts data.

Chapter VI

RESULT & CONCLUSION

Result

This study concluded that totally 260 applications are found along with the files as an analysis sample. Few of them are shell file, .apk file, mobile tracker application etc….. In that total 43 applications that leak sensitive data such IMEI, Android-id etc. Additionally 2 applications leak e-mail address. From that 21 applications which match with our signature that are used for name of advertise services. Additionally, 12 applications detected running exploit code, 4 applications use as root,4 applications execute rageagainstthecage that are Trojan program which is infected application such as DroidDream.These applications are get a root privilege on target devices without user knowledge, applications are a serious threat. In that 16 applications which run su command in system call activities. The application that match to signature pattern were found superuser command, it must be detected like that malicious threat. It also found some of the applications take permissions READ SMS, RECEIVE CONTACTS, READ PHONE STATE and WRITE SMS etc. Additionally it has detected 42 applications, which were banned by Indian government in 2017, some of them, Weibo, Qq media, Cachecleaner Du app studio, parallel space, perfect cropetc……. According to Indian government these applications are malicious applications and harmful for user.

Table 2: List of Detection

Threat	**Number of detection**
Information Leakage	91
Jailbreak	10
Abuse Root	15
Critical Permission	227

Table 3: Indian Government Banned Application Detection

Threat	Number of detection
Information Leakage	50
Jailbreak	4
Abuse Root	9
Critical Permission	155

Conclusion

The study concluded that total 260 applications were found to be malicious to android. Depending upon signature malicious applications has been divided into four categories which include information like leakage, jailbreak, abuse root and permissions along with these applications it detected 40 malicious applications which are banned by Indian government. This project concludes that these applications are very harmful to android mobiles. Use of android applications can be carried out for behaviour analysis. Based on behaviors, android application will be developed. As it's only for android platform therefore only apk files will be accepted as inputs.Report will be provided whether the input apk is malicious in nature or not. Prevention measures will be applied to stop them.

CHAPTER VII

REFERENCES

[1] IkerBurguera and UrkoZurutuza, SiminNadjm-Tehrani"Crowdroid: Behaviour-Based Malware Detection System for Android"., Electronics and Computing Department Mondragon University 20500 Mondragon, Spain Dept. of Computer and Information Science Link ping University.

[2] TakamasaIsohara, Keisuke Takemori and Ayumu Kubota "Kernel based Behaviour Analysis for Android Malware Detection", KDDI R and D Laboratories Saitama, Japan, 2011 Seventh International Conference on Computational Intelligence and Security

[3] Raymond Canzanese and SpirosMancoridis, Moshe KamSystem Call-based Detection of Malicious Processes, Dept. of Electrical and Computer Engineering College of Computing and Informatics Drexel University, Philadelphia, PA, USA Newark College of Engineering New Jersey Institute of Technology Newark, NJ, USA 2015 IEEE International Conference on Software Quality, Reliability and Security.

[4] Andrew HoogAndroid Forensics, Android Forensics: Investigation, Analysis and Mobile Security.

[5] Weiqin Ma PuDuanSanmin Liu GuofeiGuJyh-CharnLiu"Shadow attacks: automatically evading system-call-behaviour based malware detection" J ComputVirol (2012) 8:113 DOI 10.1007/s11416-011-0157-5

[6] L.Wu, M.Grace, Y.Zhou, C. Wu, and X. Jiang. The impact of vendor customization on android security, In Proceedings of the 2013 ACM SIGSAC conference on Computer communications security, pages 623634. ACM, 2013.

[7] Y. Zhou and X. Jiang."Detecting passive content leaks and pollution in android applications", In Proceedings of the 20th Network and Distributed System Security Symposium, (NDSS), 2013.

[8] S. Arzt, S. Rasthofer, and E. BoddenInstrumenting android and java application as easy as abc. In Runtime Verification, pages 364381. Springer, 2013.

[9] Vanessa N. Cooper, HossainShahriar and Hisham M. Haddad A Survey of Android Malware Characteristics and Mitigation Techniques, Department of Computer Science Kennesaw State University Kennesaw, Georgia 30144 USA 2014 11th International Conference on Information Technology: New Generations

[10] A. Developer"Android sms manager api reference page", 2015.[Online]. Available:HTTP://developer.android.com/reference/android/telephony/SmsManager.html.

[11] "Android (operating system).http://en.Wikipedia.org/wiki/Android (operating system).Accessed: Feb. 2015",

[12]"Droidbenchbenchmarks.http://sseblog.ec-spride.de/tools/droidbench/. Accessed: Feb. 2015. ",

[13] Andrea Saracino, Daniele Sgandurra, Gianluca Dini and Fabio Martinelli."MADAM: Effective and Efficient Behaviour-based Android Malware Detection and Prevention, Citation information: DOI 10.1109/TDSC.2016.2536605, IEEE Transactions on Dependable and Secure Computing..

[14] Mingshen Sun, Xiaolei Li, John C.S. Lui, Fellow, IEEE, ACM, Richard T.B. Ma, and Zhenkai Liang Monet: A User-oriented Behaviour-based Malware Variants Detection System for Android Citation information: DOI10.1109/TIFS.2016.2646641,IEEE Transactions on Information Forensics and Security.

[15] Daniel Arp, Michael Spreitzenbarth, MalteHubner, Hugo Gascon, Konrad Rick DREBIN: Effective and Explainable Detection of Android Malware in Your Pocket,

[16] Shuaifu Dai, Tao Wei and Wei Zou "DroidLogger: Reveal suspicious behaviour of Android applications via instrumentation", Beijing Key Lab of Internet Security

[17] Fan Zhou, Yitao Yang, Zhaokun Dingy, Guozi Sun "Dump and Analysis of Android Volatile Memory on Wechat", Institute of Command Information System, PLA University of Science and Technology, Nanjing 210007, China.

[18] Jing Tao Yan Zhang, Pengfei Cao, ZhengWang, and Qiqi Zhao An Android Malware Detection System Based on Behaviour Comparison Analysis Ministry of Education Key Lab for Intelligent Networks and Network Security, Xian Jiaotong University, Xian 710049, China.

[19] TakamasaIsohara, Keisuke Takemori and Ayumu Kubota Kernel-based BehaviorAnalysis for Android Malware Detection KDDI RD Laboratories Saitama, Japan2011 Seventh International Conference on Computational Intelligence and Security.

[20]https://www.financialexpress.com/industry/technology/government-reportedlylists-42-chinese-apps-as-dangerous-including-truecaller-uc-browser-mi-store-checkif-your-phone-has-any-of-them/954335/.

[21]http://usa.kaspersky.com/about-us/press-center/press-releases/kaspersky-lab-and-interpol-survey-reports-60-percent-android-at

[22] http:// Android OS Architecture - Techplayon techplayon.com