Chapter 1: Introduction

People have started using smart phones, since various companies provide various utility features in devices. According to Garter studies, market of cellular devices grew 10% in last few months of the year 2011 whereas the market of cell phones/smart devices has risen up by 39%. With much interest, Android Operating System itself accounts for more than 55 percent of smart phones sales since its origin. Current day mobile devices have four capabilities –, computing, communication, sensing and high utility. Besides being at a high sale rate and such capabilities, these devices have also made the malicious attackers ready to attack and steal data. This idea is complemented by Lookout Threat report which has done great effort with respect to malware. As the sale has increased exponentially the malicious coders have also increased. Mobile malware performs malicious activities like stealing private information, sending sms, reading contacts and can even harm by exploiting data. Malware authors can cause much damage to device users as so many users use capabilities of devices such as money transfer, online bank payment etc. Recent news and survey states that android platform is the mostly attacked platform for malwares. Since the malware can enter from the network, so its users responsibility to install malware free application. Even a malicious author can even repackage a famous application. So the first right is with user to check the permissions which an application asks during its installation. Once the user allows the application, he grants the application to use the permissions mentioned completely. Otherwise the user can deny installing the application. Our aim is to reduce the risk of malicious applications which creates problem to the one who uses android phones, by enhancing the security which performs white listing of network permission by analyzing packet using tools such as snort, wire-shark etc. As we know Android platform has a permission model, which ask the user before installing the applications. Android Apps contains some permissions for performing executions that may create consequence for cost or harm confidentiality and even integrity of personal details stored in the device. One of the very common permission is access to the Internet. Generally, 58% of the applications that are in the Android market request this permission, can communicate and have allowance for any port on the Internet. Permission plays an important role in android device security. Before restricting access to an application component, you need to define the set of permission in the manifest. You have to use permission tag and within it you can specify the level of access the permission will allow (dangerous, signature, normal, signature Or System). But if the

malicious author repackages the applications with malicious code, then the victim may lose his precious data. Adding access level grants security up to some level. We were solving our problem by grouping the applications, according to permissions stated in it. Applications having risky permissions are further analyzed with static and dynamic process and the destination address being blacklisted. We are using approach for looking for malware in application such as reverse engineering in android, and comparing by installing third-party application.

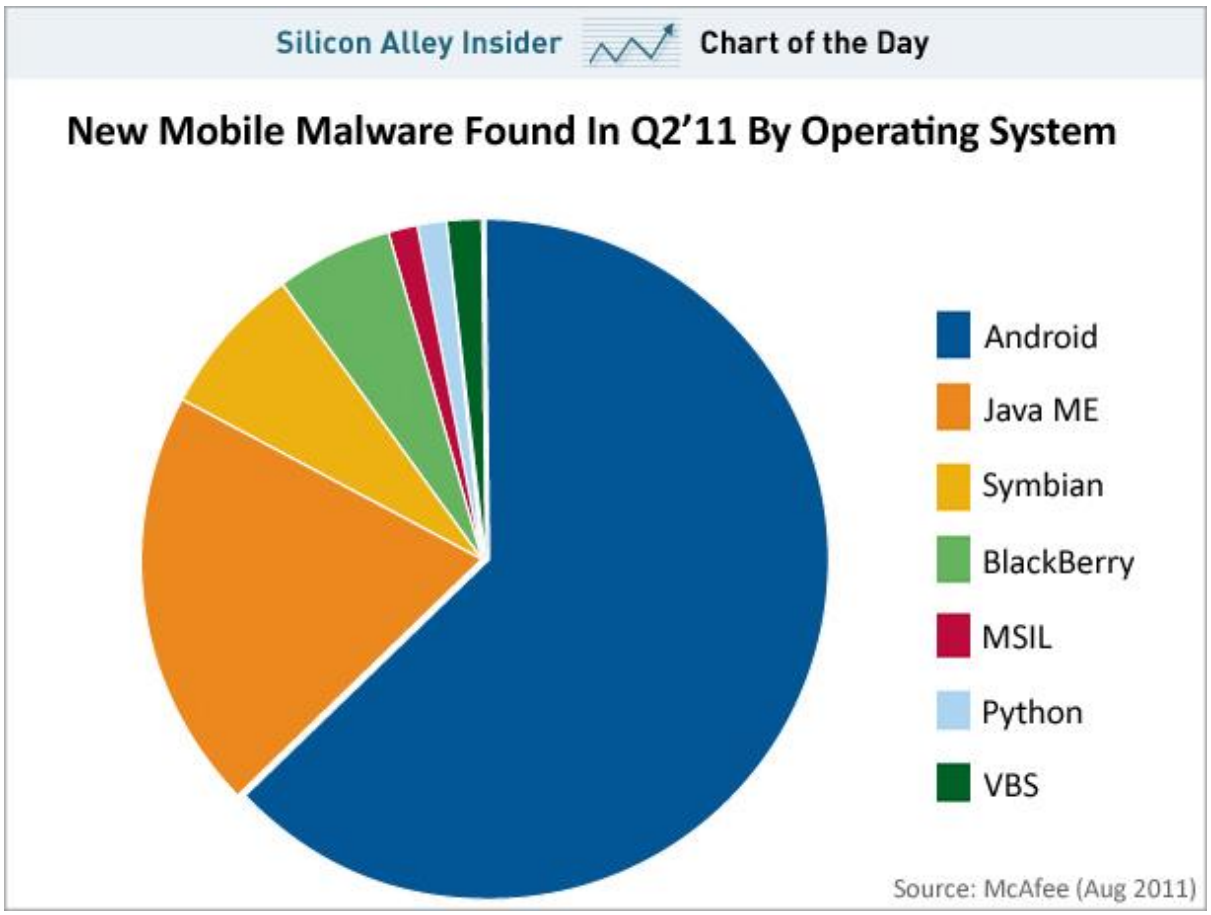


Fig 1: Malware study according to platform

This could be seen in existing literature have used tools to find the over privilege permissions, user's behavior, attention while installing applications as this the basic step

before user gets influenced to harmful application on their device. Our work will define the list of sites used in the internet by the application developer

1.1 Mobile Trends

A cellular phone is a device which makes and receives phone calls over a radio link as it moves around a broad geographical area. It is done by connecting it to cellular network which is provided by a cellular phone operator, and allows access to the public telephone network. In case of, a cordless telephone, it is used within the limited range of a single, secure base station.

Including basic feature of telephony, today's era devices contains variety of wide additional services such as gaming, photography, messaging, MMS, mailing, accessing Internet, using short-range wireless communications (Bluetooth, infrared), and business applications. Cellular phone which offers such capabilities of general computing are called as smart phones.

The demonstration of newly hand-held cell device was done with the help of JF. Mitchell and Dr M. Cooper from the company Motorola in the year 1972, which weighs about 2.2 lbs 1 kg [4]. In year 1983, the Dyna TAC 8000x was the first phone to be launched. From 1990 to 2011, worldwide the users grew from 12.39 million to 5.99 billion, increased about 86% of the total population, which reached bottom of the economical pyramid.

1.2 Different Platforms

1.2.1 Mobile Platforms

The very available cellular operating systems (O.S) which exists are:

Android platform which is open source and free is now owned by Google.

Fig 2:Android device version 4.2 known as (Jelly Bean) is available on Google handset Nexus 4

Google has bought android platform from a very beginner company. In the year 2005, Android is a based on Linux-driven Operating System which has full support of company Google, along with other software/hardware developers which are known as Samsung, Motorola-Power, MTS, HTC, ARM and also eBay, and all these come under Open Handset Alliance (OHA).It was released on 5th November 2007, this Operating System was very well used up by developer who begin their career with it. Android's startup releases such as 1.5, 1.6 and 2.0 were deployed exclusively on cell devices. Different type of these OS devices, and few tablets of this current Operating system, started using a release of 2.000 xs and more. But Android version 3.0xxx was a tablet release and was not currently for device with less resolution than a tablet. Current version is 4.2.2 and these versions are named after dessert

items such as Cupcake 1.5, Froyo2.., Honeycomb 3.0, Ice Cream Sandwich-4.0 and Jelly Bean (4.1). Many mobile service providers carry an Android device. When HTC dream was in the market, the sales of android platform grew larger as it contains Android OS with the number of companies using android platform rising more, the sale grew with much 72% in second half November 2011. The market sales of android finally touched 52.5% of the smart phone market share globally.

Fig 3: Apple's company iPhone and iPad platform use the iOS..

BlackBerry 10 Operating System is the upcoming platform for this decade which can be used for smart devices and tablets made by BlackBerry. This states that there will be only one Operating System for both Blackberry tablets and smart phones going forward. It was previously known as BBX

iOS discovered by Apple

Apple’s iOS operating system which was a part of MAC OSX was used by Apple (iPhone, iPad, iPod touch). It is close source and even native third party applications have not support official manner till the support of operating system iOS 2.0 releases which was on 11 July, 2008. Prior to this, "jailbreaking" conceded third party apps for installing, which still exists? Recently every iOS device is developed by Apple and being manufactured by Foxconn .

Nokia platform for Nokia Asha series, NokiaSailfish Operating System. Nokia Sailfish is an open source platform and GPL. After Nokia's failing in 2011 along with MeeGo company most of MeeGo people left the job at Nokia, and organized a different company which was named as Jolla for using MeeGo and MER business privileges. In year 2012 OS named Linux Sailfish based on MeeGo Company that use MER core distribution finally launched for public usage. Jolla device (mobile phone) has been revealed on May 20th2013.

Windows Phone from Microsoft (proprietary and closed source)

On 15th February, 2010, Microsoft disclosed its upcoming-generation mobile Operating System, Windows Phone. Microsoft's "Metro Design Language" inspired the upcoming launched mobile OS which includes a fully new dynamic user interface. It accommodates complete combination of Microsoft services like Microsoft Office and Microsoft SkyDrive, Xbox Video, Music; Live gaming etc., including many other services provided by non-Microsoft such as like Google and Facebook accounts. This new platform has achieved some optimistic results from many technology presses.

Windows RT from Microsoft (proprietary and closed source)

Windows RT is an Operating System specifically designed for tablets running on ARM processors was formally announced by Microsoft. It appears like Windows 8.It is unable to run x86 programs. Windows Store is the only store for downloading Apps. Microsoft Office 2013 is already installed in it.

1.2.2 Other Mobile applications software platforms

Samsung Electronics introduced BADA (it is closed source)

Samsung Electronics developed the mobile Operating system named as Bada. Samsung declared that badaos will fast change its rights feature device platform, changing feature device to smart devices. The term ‘bada' is originated from Korean word (바다), which is means sea. 'Wave' was the latest device using BADA operating system, which includes a complete touch User Interface, was announced to the public at the event known as Mobile World Congress 2010. Samsung launched a app store known as Samsung apps and it contains more than 3000 applications. Samsung announced that they don’t like to continue with BADA operating system and it is like an Operating System which includes a configurable kernel architecture, allowing the use of either a the Linux kernel or a proprietary real-time operating system (OS). On February 25 2013, Samsung declared that developing Bada, will be stopped and start developing Tizen platform.

BlackBerry Operating System discovered by BlackBerry (it is closed source)

BlackBerry Operating System has been designed for business class and it provides easy operations. The third-party apps of blackberry have risen up in the market with the full support of multimedia and provide various other offerings. If we check, Blackberry's Application World has created space more than 60,000 applications for users to download it. Blackberry is concentrating on QNX in future, which have already been launched for the BlackBerry Playbook tablet which is running on QNX and the phones launching is expecting in 2012

BlackBerry Tablet Operating System by QNX

BlackBerry Tablet operating system developed by QNX which is a commercial use and is for Unix real time, aiming mainly at the market for embedded systems. It was developed by Canadian company named as QNX SS and this company was later owned by Blackberry. Blackberry was named as RIM in past. But this Os is for Blackberry Playbook and tablets

Fusion Garage new platform known as Grid OS

Android kernel’s source code has assisted for this platform. After taking help from Android kernel this OS provides Grid 4 which is device phone and Grid 10 which is the tablet.

Linux based OS (open source, GPL)

Docomo in Japan and Motorola in china use the Linux Operating system, the most. Rather than an OS for itself, Linux has prepared itself for various different companies, which

includes Openmoko, MeeGo, Android, GridOS, LiMo, Maemo, Boot to Gecko, Qt Extended, which are mostly not compatible Linux and it also runs Palm Source software platform based on Linux is developed by NTT Do Como, Panasonic, NEC, Vodafone and Samsung.

A community-driven operating system known as "Mer project"

It is an open, optimized for cell phones, core distribution aimed at manufacturers of device, Qt/QM, HTML5 and is powered by it – It is open source developed, and meritocratic ally governed.

S40 (Series40) from Nokia (it is closed source)

Nokia uses S40 operating system for low end phones. Using these platform more than 160 models has been launched till now. Earlier it just provides monochrome low resolution and now it provides full 256k color touch User Interface.

Qualcomm new launch "Brew"

This platform is used by few cellular phone manufacturers and mobile networks, but most often the end-user don't knew about this since cell phones running it most often prohibits its branding. Brew runs in background with custom "skins" of the cell phone operator or manufacturer on-top. Brew is used by metro PCS, Sprint Nextel, United States Cellular and Verizon in the United States and with the other networks in most of the UK, and Europe, and Australia on many cell phones developed especially for their own network. Some manufacturers such as INQ Mobile, Amoi, LG CDMA, Huawei, and Samsung Mobile amongst others use this platform in some of their devices and it is also characterized in Three UK phones like as the 3 Skype phone, INQ1 and Huawei u7510. Two of HTC's cell phones uses Brew's successor platform Brew MP.

SHR (community-driven)

SHR is a Linux-based operating system for smart phones and similar devices. It includes various projects of free and open source software in a robust platform flexible enough to run on a wide selection of mobile phone hardware as the Nokia N900, Palm Pre, Openmoko Neo Free Runner and variants, HTC HD2, iPhone 3Gs T -Mobile G1 and more. The build system of SHR is based on Open Embedded based - often from the Yocto Project known. For telephony and networking, etc. freesmartphone.org framework mostly used. At the beginning it is easy to use graphical user interface centering on the education libraries, which are used to provide cell phone calls and messaging use. An increasing number of available apps

provide its users with most of the things expected of a smart phone. It also uses many classical programs often can be made available from another known Linux distributions.

SHR is 100% community driven and based on open source and free software. This provides power to all to realize their ideas or support for new hardware - without the need to ask you something first strategy manager or CEO.

Symbian OS from Nokia and Accenture (open public license)

Symbian has the highest smart phone share in most of the markets in the world, but still remains behind many other companies in the relatively small but visible North American market. This concept matches the success of Nokia in all the markets excluding Japan. But in Japanese market Symbian is effective due to a relationship with companies like NTT DoCoMo, with only single among 44 Symbian handsets released in Japan which comes from Nokia. It has been used by number of handset manufacturers, including, Motorola, Nokia, Samsung, Sharp Sony Ericsson, BenQ, Fujitsu, LG and Mitsubishi. Fujitsu is one of others making recent Symbian-based devices. Others being made by Nokia, Samsung, Sony Ericsson and Sharp .Before 2009 Symbian supported multiple UI, that UIQ from UIQ Technologies, S60 from MOAP and Nokia from NTT DOCOMO. For the formation of the Symbian OS in year 2009 these three user interface were combined into a single operating system which is now totally open source. Currently, the shipping of devices with Symbian have increased, the OS's worldwide market share has gone from over 50% to just about 40% from year 2009 to 2010.Accenture has taken up the development of Symbian from Nokia, which will stand up to support the operating system until 2016.

Web OS from LG (certain parts open sourced)

It is a mobile OS running on the Linux kernel, developed by Palm, which developed with the Palm Pre. It was taken over by HP, the two phones are Veer and the Pre 3 and the Touch Pad tablet running web OS in 2011 were introduced. In 2011 18thAugust explained that HP's Web OS hardware is stopped, but HP will also continue to support and upgrade Web OS software and can develop the web OS ecosystem, if necessary. HP has this platform released as open source and called as open web OS and has planned to upgrade it with many additional features. In 25 February 2013 it was stated that HP has sold out Web OS to LG Electronics which has further planned to use their "smart" or Web-connected TVs. But HP is still in patents based Web OS and cloud-based services like the App Catalog.

1.3 Mobile Security

Security of device has been an important part in Mobile Computing. Actually nowadays user stores their precious data on mobile device and hence security is a major concern. Cell phone are so much used these days that besides using them for communication tools these devices are used for planning purpose in daily life. Inside these companies, these technologies are causing deep changes in the organization of information systems and so it became the source for new risks. Truly, cell phones gather and collect and compile a rising amount of sensitive data information for which accessing should be controlled for protecting the privacy of the user and the intellectual property of the company. According to ABI Research the Services for Mobile Security market will reach up to around $1.88 billion till the year end 2013.

All smart devices are preferred targets of attacks, just like computer system. Such attacks exploit weaknesses related to devices which come from various means of communication such as Wi-Fi networks, SMS, MMS and GSM. Some attacks destroy the vulnerabilities of web browser and operating systems (O.S). In the end, there is some malicious software which relies on average knowledge users.

The different levels of security counter measures developed and applied for smart phones, ensuring the security in any software layers for the dissemination of information for end users. There are many good practices at every level are observed to operate from the design, through to the development of operating system and application software layers that are downloaded.

1.3.1 Overview of Android Security

Introduction

Android is an open source, and new mobile platform. It guarantees by providing data security and protection also for the device users, applications and the network.

If you want to secure an open platform, it requires rigorous and robust security architecture. Android platform was designed as complex type did certainly provide a flexible environment for an open platform is required, and at the same time it provides protection for any user to use the platform.

Android platform has been designed for both users and developers. For developers security some controls were designed for reducing the burden on developers, so that they can simply work with and can rely on controls that have flexible security. For users, they are provided visibility of working and control of an application. User Behavior and attention[1].The creepy people that is attackers would attempt to perform few attacks, which are attacks of social-engineering to convince users of device to install malware, but the permissions allows users to completely allow or deny installation.

But this new Android platform was introduced to reduce the likelihood of such attacks and limit the impact factor of always attack in the event forever successful. This described the goal of the program android security, described the basic principles of security architecture android, and answers the most important questions for system architects and securities analysts. This document focuses on the safety features of the platform, the core of android and does not discuss the security issues that are unique to the specific applications, such as in relation to the browser or application SMS. The practices recommended to establish the devices of Android, deploy the Androids device, or to develop requests for android platform are not the aim of this work and are provided elsewhere

Android platform provides an open source environment and application framework for cellular devices. There are basically four building blocks in android:

1. Activity 2. Service 3. Content Provider& 4. Broadcast Receiver

Activity

It is a component/part in which user can interact with screen or User interface. An Activity represents the presentation layer of any Android app. For example it is basically a screen on device which the user sees. There can be several activities in android application which can be switched among them during the running of the application.

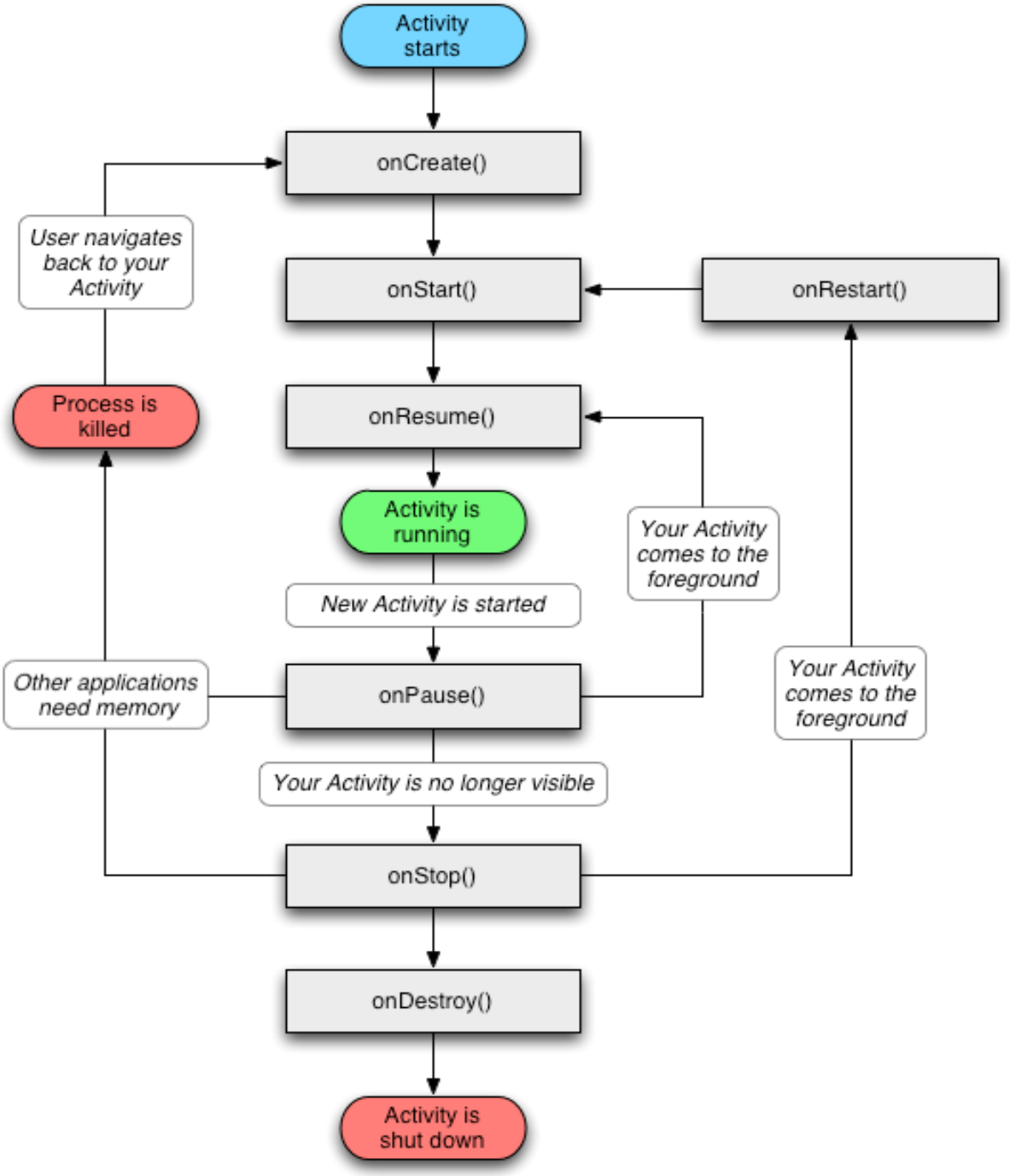


Fig 4: Android Activity Lifecycle

Service

It performs the background tasks without providing an User Interface. Services can notify the user through the notification framework. Every service class should contain a <service> declaration in its package's AndroidManifest.xml.

Content Provider

Content providers stores and retrieve the data and make it easy accessible for all the applications. They are share data across applications and have no other method and there's no storage area that is common for any Android packages to access. But android contains a SQLite Database which serves as data provider.

Broadcast /Intent Receiver

A class namely broadcast receiver extends BroadcastReceiver and is actually registered with the help of a receiver in an Android app through the AndroidManifest.xml file. This class receives intents. Intents are generated through the method of Context.sendBroadcast().The class Broadcast Receiver class defines the onReceive() method. With this method your broadcast receiver object is valid, but after it the Android system will consider you're inactive. Then you can't perform any type of asynchronous operation

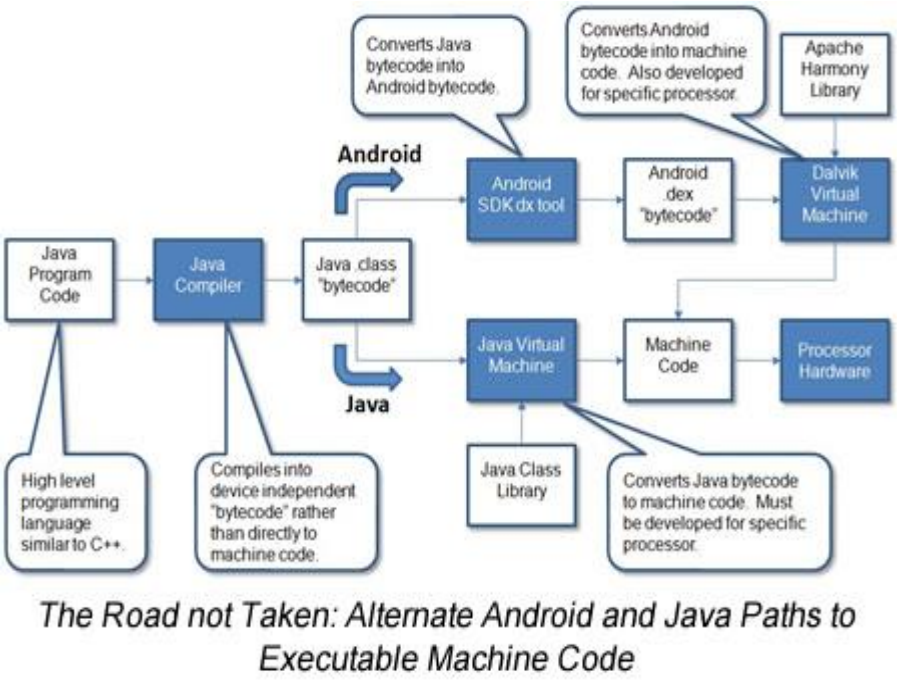


Fig 5: Android compilation

1.4 Android Platform Security Architecture

Android claims to be secure and usability platform for mobile OS reclassification of traditional OS security controls to:

• Protect user data

• Protect system resources (including the network)

• Provision of application isolation

To achieve these objectives, Android has provided these important functions:

• At the OS level, robust security from the Linux kernel

• Mandatory application sandbox for all applications

• Safe and secure communication between

• Application signing

• Easy application and permissions granted and defined

The following sections explain the security features of the Android platform. In Figure 1, we find the summary of the safety components and different levels considerations of the Android software stack. Each component is believed that they are properly secured. Leaving the exception of a small amount of Android operating system code running as root is restricted code above everything else the Linux kernel to the application sandbox.

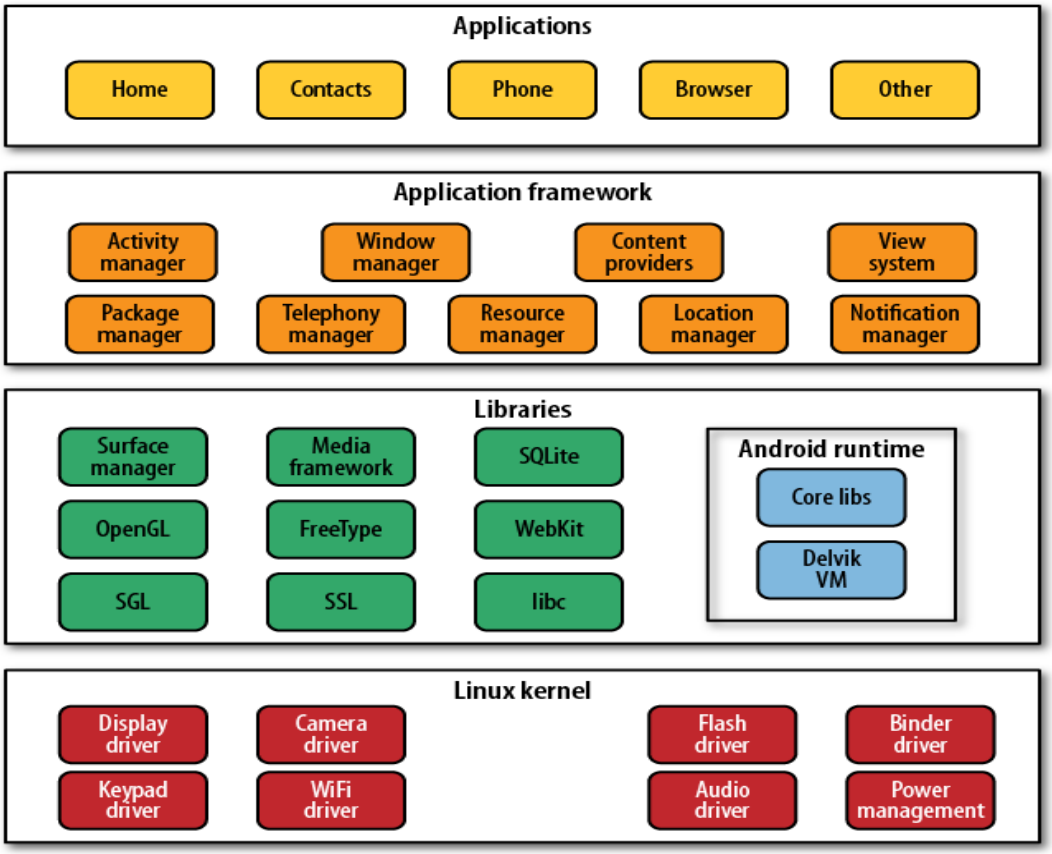


Fig 6 :Android Architecture

The android architecture consists of the following:-

Application: - Android phones generally comes with some default applications installed in it such as browser, calendar, SMS, maps, email client etc. These applications are programmed in Java.

Application Framework: - Android developers are offered various utilization of information of accessing the location, setting alarms, configuring device hardware, and running background services, etc. Developers of Android have access to the framework APIs that is used by the core applications. Reusing components is exercised by the design of application architecture. In the applications we can have a set of services and systems underlying them:-

Views: - These are used for building applications, such as, text-boxes, grids, buttons, and even a web browser that can be embedded.

Content Providers: - They allow applications to share their data and to access other application's information.

Resource Manager: - R.M provides access to resources such as graphics, localized strings and layout files.

Notification Manager: - Using it, custom alerts are displayed on the status bar.

Activity Manager: - AM has a navigation back stack and deals with application's lifecycle.

Libraries: - Android provides a set of C/C++ libraries that is used by several components of the system. Such libraries are provided for the development purpose with using the Android App Framework.

Few core libraries and their characteristics are shown below:-

System C Library: - Used for embedded Linux-based devices. This library is a BSD derived implementation of the C system Library libc.

Media Libraries: - These libraries basically support recording of video and audio formats, playback and static image files, e.g., MP3, PNG, AAC, JPG, AMR, MPEG4 and H.262

SGL: - This comprises of 2D graphics engine in it.

SQLite : -It is a relational database engine, accessible to all applications.

Surface Manager: - It controls the access for 2D, 3D graphics layers from various applications and to the display subsystem.

Android Runtime: - Android has few core libraries that offer much functionality and are also part of the core libraries of the programming language in Java. All applications of android run in their own processes in their own instances of the Dalvik Virtual Machine. Dalvik allows a device for running several virtual machines efficiently and DVM executes all the files in the

(.dex) format. The DVM is dependent on the Linux kernel for underlying the functionalities for example we use thread management and low-level memory management.

Linux Kernel:-Services of android such as process management, security and network security, driver model, memory management, of the core system mainly depends on Linux version 2.6. The Linux kernel also behaves as a layer of abstraction between the hardware and software stack.

1.5 Android Permission Model (Access the Protected APIs)

Each application for Android device made runs in a sandboxed application has already been described earlier in the document. Android app can only be on a short reach of the resources of the system by default. Android application is managed by the system and got access to resources in case if used incorrectly or maliciously, it could badly affect the user experience of users or data on the device, or possibly the network.

These restrictions are implemented in various forms diversity. Few skills are by a lack of web service restricted to the sensitive functional behavior. In some cases, separating roles as a safety measure are provided by app-storage insulation. In some other cases, the delicate web Service are for use by trusted applications determined and protected with a safety mechanism known as permissions. The protected APIs include:

- Camera functions
- Location data (GPS)
- Bluetooth functions
- Telephony functions
- SMS/MMS functions
- Network/data connections

Resources are defined only on Android OS for the use of the protected application package interfaces to the cellular device, an application within the capabilities it needs must be defined in its manifesto. When you prepare to install an application, the system dialogue is to indicate to the user, appear confirming if continue the installation or not the requested permissions. If the user enables and continues with the installation, the system accepts and

understands the user has allowed all the permissions requested. The user can not grant or deny individual rights - the user must grant or deny all permissions requested completely.

Once user acknowledges the permissions for the application will remain as long as it does not get uninstalled. To avoid confusion for users, please inform the system the user only once on the permissions granted to the application, and some applications that are included in the core OS or OEM bundled it so not to request permissions from the user. Each permission is removed when an application is uninstalled from device, a frequent re-installation can lead again to show permissions.

Within the settings of device, users are fashionable to view the permissions for applications they have installed in previously. But users can turn off little functionality globally, whenever they like, as disabling radio, GPS, or Wi-Fi.

In the event if an application attempts to use the protected feature which is not declared in the application's manifest, the failure through permission will basically result in a security exception generally thrown back to the application. The protected API permission checks are applied at the lowest possible level preventing from circumvention. In case if the user message application is installed while requesting access for protected Application Package Interfaces shown in *Fig 2*.

The default permissions of the system are described in www.developer.android.com / reference / android / Manifest.permission.html. Android apps can specify unique permissions for other applications to use purposes. Fewer permits are not listed in the above situation.

When defining entitlement to protection level attribute, it is the system as the user applications with the permission required information, or who is not limited in order to keep the license. Much information on creating and using application specific permissions stated under the following link in www.developer.android.com / guide /issues / security / security.html.

There are only few phones features such as the ability to SMS broadcast intents that are not available to send the third-party applications, but the applications can be used pre-installed by the OEM. These use the signature or system permission.

1.6 Android Security Model

The security model of Android is designed so that no application has permission to carry out an execution, which can badly affect other applications, the user or the operating system (OS). Each application runs in its own process and therefore android is considered as a multi-process system. The platforms of Linux devices for the user, developer and group identity are assigned to the process level and access rights to specific information for applications to impose the security of the system and the applications.

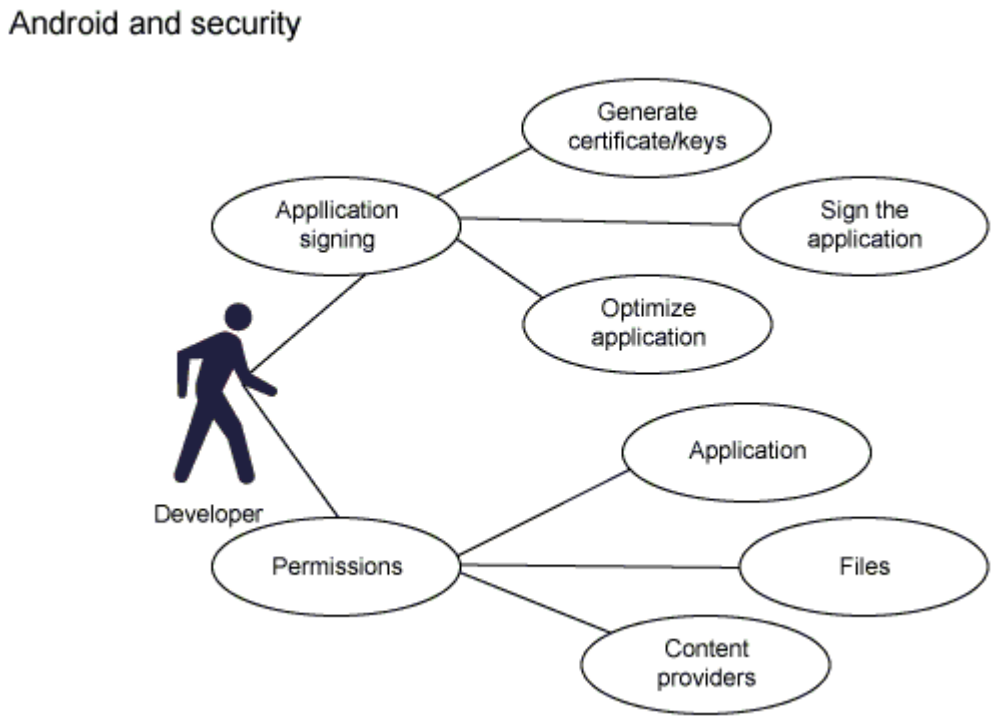


Fig 7: Android Security

Sandbox and Permission mechanism is said to be the base of the Android Security Model. It is fact that each specific user ID assigned to it. Every application runs in isolation from the other applications.

Author Jesse Burns states that,

Android supports building applications that use device features while defending users by avoiding the need for complex policy configuration files/reports for the sandboxes. This gives granting the application additional rights´.

As per Luke Jeter, Mena Mani and Tia Schmidt net, each Android application with a certificate whose private key will be signed only known to the owner of the application. This enables the author of the application may be identified. If an application on the phone is assigned a unique Linux user ID, which is installed by impact on other applications by creating a sandbox. This user ID is always on this device and applications that can be executed in a single operation the same user ID. This is one way to ensure that a malicious application cannot access / compromise the data of the real application [9].

It is obligatory for an application to list all the resources it will access during installation. The permissions that is requested by an app, during install time, are first approved by user interaction or/and based on checks alongside the signatures of the application (9).All three authors say that Android permissions are needed at various stages in the life span of an app.

At the time of a system call in order to prevent the application from performing specific functions that is undesirable. The beginning of an activity in order to avoid applications from initiating activities of other applications. Permissions to manage who can send and receive a broadcast message from you. Binding or beginning of a service. In order to access and operate on a content provider.

Kamran Habib Khan & Mir Nauman Tahir indicate that there are four levels of security provided by Android, naming them

1 Normal:-In this case no permission is required by the user, hence normal permissions are granted to the application.

2. Dangerous: -These permissions are prompted by an application for approval by the user during installation. The user can either accept or reject all the permissions. The denial of permissions completed installation by stopping it.

3. Signature:-These permissions are acknowledged by the system provided the granting and the requesting application have the same certificate.

4. Signature System:-This is similar to Signature but applicable to system applications only.

The numerous built-in security methods of Android make it secure only as long as it is handled by responsible users who understands the repercussions of the several permissions requested by various applications. But, the challenge has been to enforce more security without limiting the freedom to the user.

Chapter 2: Literature Review

Our goal is to reduce the risk of malicious apps harming the user of a device to reduce, by strengthening the permission model that white listing [7] of network authorization. As we know Android platform has a license model to ask the user before the application. Applications are required privileges if they confidentiality and integrity of personal data stored on the device operations, monetary costs incurred or injuries and may have to be performed. A resource of an authorization to use Internet allows protected. Usually requires 58% of the applications that are available on the Android Market this internet to communicate with any host access to the Internet. Our thesis presents the permission model of Android and the way implemented was used. Our thesis work provides another level of security the user from running a malware that is not in the existing research literature is given, would protect.

The attacks on mobile devices are keeping increasing, more and more malware are affecting the user. While performing testing using Kirin [5] with around 314 famous applications states that the rules flagged for 10 android apps for which the attitude of 5-6 was discovered to be questionable. Kirin solely checks for permission of application author's requests and are not able to judge how the application takes the use of these permissions.

David Barrera [2] and partners have per-formed permission checks on about 1,100 Android applications and have used self-organizing maps (SOMs) to foresee the relationship among the applications and the permissions requested. But, self-organizing maps SOMs also focus completely on the permission of the application author's requests and could not examine how the application uses them. While studying android apps permission requests which included 858 free applications and 100 paid, around 94 % free apps and 83 % paid apps had at least one request of dangerous permission. The permission of Internet is the most common one and very dangerous request. But, wholly scrutinizing the permission request are not fulfilling for detecting mobile malware; and could have been performed in parallel with static or dynamic analysis. In year 2011, Felt [8] et al. analyzed 46 pieces of iOS, Android, and Symbian malware that have spread in the wild from 2009 to 2011.Andromaly [3] which monitors both the smart phone and user's behaviors by observing several parameters, spanning from sensors activities to CPU usage, 88 features are used to describe these behaviors; the features are then pre-processed by feature selection algorithms. The malware authors developed four malicious

applications to check the ability to detect anomalies. This paper contains further research and illustrates latest malwares, detection and defense techniques by referring several papers, blog posts, vendor technology and specifications.

David [2] et al. discusses permission re-delegation attacks on Android. Make the problem and make an attack on a vulnerable deputy. We carry a greater analysis of applications and discuss how to change platforms in order to prevent these attacks.

Chin et al. Com Droid present a static analysis tool to prevent developers from accidentally components is public. They are also recommendations for changes to the Android platform in order to reduce the rate of unintended deputies. Although her tools and recommendations for their platform would prevent some cases of permission re-delegation would remain intentional attacks on deputies.

Taint Droid [4] performs dynamic taint analysis. It tracks the real-time flow of sensitive data through applications to detect inappropriate sharing. The taint source is API data, and the network is the sink. They track only flow of data, but not control flow. Taint Droid [4] is complementary to IPC Inspection because they track API return values but do not prevent API calls from being made. Another tool, Scan Droid [1], uses static analysis to determine data flow through Android applications; it is intended for use similar to Taint Droid [4]. Scan-Droid, however, requires access to application source code. Kirin [5] checks application permission requirements and recommends against the installation of applications with certain permission combinations. Their rules are intended to help detect malware.

Apex [6] offer extensions to the Android framework to provide fine grained control over an app's access to potentially sensitive resources. Most of these efforts are aimed at addressing this problem on the user's phone; Risk Ranker [7], on the other hand, attempts to identify such risky behaviors at the app market, offer extensions to the Android IPC model that allow the ultimate implementer of a privileged feature to check the IPC call chain to ensure unprivileged apps cannot launch confused-deputy attacks unnoticed. Similarly, besides the above defenses, some work has been proposed to apply common security techniques from the desktop to Mobile devices. Some work has focused on malware and the overall market health. Felt et al. [8] surveyed 46 malware samples on three different smart phone platforms, discussing the incentives that motivated their creation and possible defenses against them. But, this work did not discuss how to discover this malware. Our work has a much stronger emphasis on malware detection than privacy leak detection. Mal Genome [9] aims to systematically characterize existing Android malware from various ways, including installing

methods, activation mechanisms as well as the nature of carried malicious risks posed by existing in-app ad libraries. These are the various works.

Permission plays an important role in security. Before you can restrict access to an any application component, it is required to define permission in the manifest file. Use the permission tag for creating these permission definitions. Application components require them by adding the android: permission attribute. Other applications will then need to include a uses-permission tag in their manifests (and have it granted) unless it can be used in protected components. Inside the permission tag, you can define the access level which the permission will allow to be as normal, dangerous, signature, signature Or System.

This could grant security upto quite much level. Existing literature have used tools to find the over privilege permissions, user's behavior, attention while installing applications as this the basic step before user gets influenced to harmful application on their device. Our work will define the list of sites used in the internet by the application developer. There are another approach for looking for malware in application such as reverse engineering [6] in android, and also by installing third-party application.

Chapter 3: Android malware and security issues

Mobile phones have become a common target for cyber-criminals. Mobile contains a huge amount of data from personal contacts to emails and it is also possible to carry out all types of online transactions. A malware is a malicious code or piece of software that is designed to perform functions without the permission of the user. Miss Joanna Rutkowska also agrees with the above definition and has defined malware as follow, Malware is a piece of some code which as a result change the behavior of acting O S kernel or few security applications which behaves as a sensitive, without the consent of user and in such a way that it is then not possible to identify those changes using a documented characteristics of the operating system(O.S)or the application. S-mobile Systems has stated that malware is categorized based on what the malware actually does once it has infected a network system. There are nine types of malware categorized by Troy Vennon, naming them:

1. Virus: - A virus is defined as a destructive or malicious program that lacks the capacity to self-reproduce.

2. Worm: - This is a malicious code that can control a system vulnerability or a network in order to automatically duplicate to another system.

3. Trojan: - A Trojan allows an attacker to obtain unauthorized access or remote access to a system while it appears to be executing a required operation.

4.Spyware:- This destructive application conceals itself from the user while it collects Authors Kamran Habib Khan & Mir Nauman Tahir have categorized the attacks on mobile phones as below :-

5. Ad-ware: - This category includes the advertisements on cell phones that are in reality malwares

6. Direct Payoff: - This category consists of malwares that send SMS without the consent of the user.

7. Destructive: - An example of these kinds of attacks is erasure of phonebook entries

8. Information Scavengers: - This category includes the checking cookies and address

9. Premeditated Spyware: - This represents remote listening and location tracking.

3.1 Malware: Definition

Malware, it is a small term for malicious or malicious software is programmed by attackers so that computer operation to disrupt collects sensitive information or gain access to private computer systems. It appears in the form of scripts, active content and other software or code type.

'Malware' is a term used to refer to different form of intrusive or hostile software. Malware can be computer viruses, root kits, key loggers, ransom ware, worms, Trojan horses, dialers, malicious BHOs, rogue security, spyware, adware, and other malicious code; the majority of active malware threats are usually worms or trojans rather than viruses. According to law, malware is referred as a computer contaminant, and also in the legal codes of several U.S. states. Malware is not like defective software, which we can call legitimate software rather it contains harmful bugs that were defected before it is released. But, there can such malware available acting as true and original software, and arrive through an official organization in the form of an attractive program which has the harmful malware attached in it along with other tracking software and gathers marketing statistics.

Software such as firewalls, anti-virus, anti-malware, are trust upon by users at home, small and big organizations around the world to protect against any malware attacks which helps in detecting and preventing the further spreading of malware in the network.

Many known early programs which were infectious were written as pranks, which are included as the first Worm on Internet. Nowadays, malware is used generally to steal private information of business, personal, financial, or importance with harmful intentions by black hat hackers.

Malware is often used widely against websites of government or corporate to collect the hidden information, or to affect their operation in general. But, malware is generally used against single to steal personal info which can be bank, credit card numbers, social security

numbers, and so on. Remained unguarded, these threats are a risk for personal and networked computers users. These are most frequently controlled by various types of network hardware, firewalls, and anti-virus software.

After the rise of widely spread broadband Internet access, malicious software has more commonly been made for gain. Since year 2003, the majority of viruses which have widespread and worms that have been built to gain control over the users' computers for exploitation of black-market. Infected, zombie computers are used for sending email spam, to host contraband data such as child pornography, or to involve in distributed denial-of-service (DOS) attacks as a form of extortion.

Malware which are created by malware authors for profit only, can be referred as spyware. These software programs are specially designed to check users' browsing, redirect affiliate or marketing revenues display unsolicited advertisements, or to the spyware creator. These programs do not spread like viruses but instead they are usually installed by exploiting the security holes. Spyware programs can also be packaged together along with user-installed software, like peer-to-peer applications.

3.2 Mitigation of Threats/Attacks

Malware in smart phones have been on the rise especially in the last some years. The graph defined shows the evolution of malwares along with their behavior. There exist many measures that can be incorporated in order to mitigate malware attacks on Android.

1. **Security Responsibilities of Developers**: - Each developer needs to keep in mind security of users data. As mentioned earlier, each application is assigned its own UID; but it is not impossible for any program to request the Activity Manager to start an application that runs with the UID of the application. Mr. Jesse Burns states that it is for this reason that each application needs to be signed by the developer.

In Android, signing of applications is usually done by self-signed certificates. The major reason for insisting on code signing is to allow developers to perform updates to their existing applications. Various other apps that are signed using the same key can request to run with the same identity ID as they are developed by the same developer. However it is important to state that this application signing does not validate the developer's user identity .Therefore

developers cannot be trusted easily hence Android should warnings and trusted signer rules in order to protect security.

2. Users Security responsibility: -A user when installs applications needs to grant access to all the permissions requested by them. It is thereby very important for a user to understand the permissions needed by an application. There exist several applications like Skype that needs many access to various data on the phone; but there are a few applications like Wallpapers, Calendar, etc. that require very less permissions. Mr. RobiSen, an Analysis Director in company Entity X, has made a summary of some permission that each user must be familiar so that it can recognize if it is safe for an app to access them and if they are actually necessary for the proper functioning of the device application.

3. Install Antivirus: - According to article written by J. Knudson, installing antivirus on the mobile phone, just as on a desktop, will help catching malwares and prevent a user from installing them. There are several antivirus soft-wares available for Android devices that help clean up malwares and protect the phone from the same.

4. Controlling Network Access: - Miss. Julie Knudson quotes A. Hoog who states that it is vital for organization to control the connection to network as attackers can simply plug in an Android device to a system, thereby using it as an alternate network connection. This has provided an easy route for an attacker to evade the John Engels suggests that IT organizations should provide separate networks and access controls to mobile devices that accesses their infrastructure and they also need to ensure that all traffic passes through both DLP and Web Security.

3.3 Privacy Leak in android device

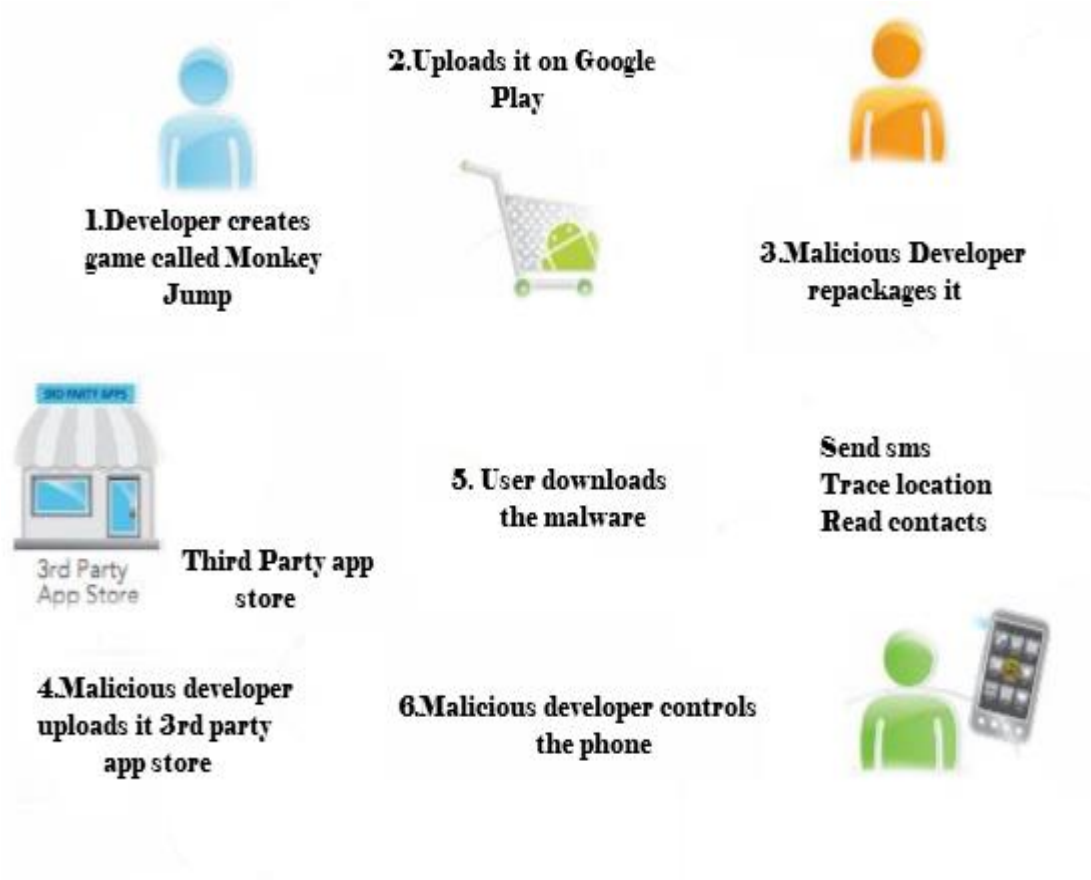


Fig 8 Privacy leak flow representation

This is a simple demonstration of how privacy information can be leaked from a user of android device. As per the example the developer creates an application say Monkey Jump and then uploads it's on Google Play. Now malicious author download it and repackage it by performing reverse engineering into the code and sell it through some third party app store. Once the user downloads such application, then he becomes victim of malware theft. He can be duped by the malware author with the malicious code written in malicious application. Now malicious developer can control the phone.

By tracking the location or send sms or may even read contacts depending upon his malicious code written in an application. So it's on user to be secure at his end. As percentage of malware apps are increasing even on Google Play the renowned market.

Chapter 4: Methodology

MALWARE DETECTION TECHNIQUES:

Such techniques are available for identifying device malware and other security vulnerabilities have changing strengths and weaknesses.

4.1 Static analysis

This approach is generally used when we looking for malicious code inside the suspected application. The malware application can cause a serious harm to the user by exploiting his precious data or stealing it. In static analysis, basically de-compilation of installer file is done. An .apk file of android is analyzed with reverse engineering process. We are analyzing upon the code based approach, so it is termed as static analysis approach. We use various tools for analyzing it. Tools used are Winzip, Java Decompiler, dex2jar etc. Reverse engineering in android is done to open and view the java and xml files for any malicious code. Following the few steps, we get view of all the activities of android. We can even view the xml files of android application installer, this process exactly pin points the false code. With the following steps we get complete knowledge of a developer's intention.

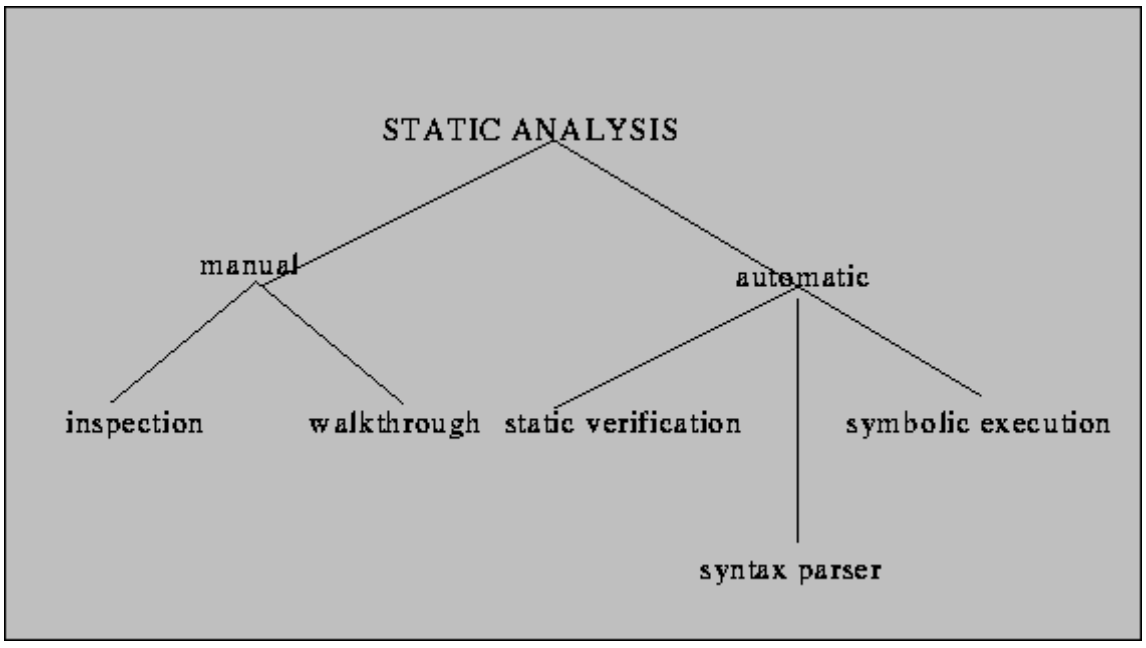


Fig 9: Static Analysis flow

a) Reverse Engineering

It is a process of analyzing on existed code or piece of software in order to check the software for any vulnerability or any existing errors. R.E is the ability for obtaining the source code from an executable file. This method is used to examine the proper functioning of a program or to check security mechanisms, etc. It can therefore be stated as an approach or process of changing a program in order to make it look like in a manner that the reverse engineer wants.
J. Boutet has quoted L.Shwartz, saying,
Whether it is the concept of rebuilding the engine of a car or drawing a sentence, people may learn about so many other things by separating them and placing them back again. In a nutshell, the real concept behind reverse-engineering is to break something down in order to understand it and then can rebuild accordingly.
Since starting of year 2009 research scientists began with many different ways to reverse the Dalvik for generating a JAR file from an Android APK installer file, which could then be again be converted to JAVA using tools like JAD and applications; but it created many problems after dealing with complex code of Dalvik Bytecode.
The Dex2Jar tool was originated then. It does similar job to undx; but this tool also has came up with issues while dealing with complex code of Dalvik Bytecode Authors B. Wissingh and T. Kriiger gave opinion that, An ideal approach to learn more about the possible functionality of an app would be checking into the source code thoroughly.
The application is generally in its pre-compiled binary format and is distributed and therefore it is not possible to debug the source code directly. But, there are dissemblers available which convert the Dalvik Bytecode into readable format in reverse manner. The binaries for Dalvik Virtual Machines (DVM)are in the .dex format. Backsmali, a disassembler is used for .dex files in Dalvik VM. APK tool is used for altering the source code, which can be done by programmers. They can repackage it. Many programmers have already reversed various applications of Android for studying any vulnerabilities existing in it and have also scrutinized the code.

Let us take an example below: Fig:10 Implementation of Static Analysis

We can perform reverse engineering of mobile app-MalwareApp.apk

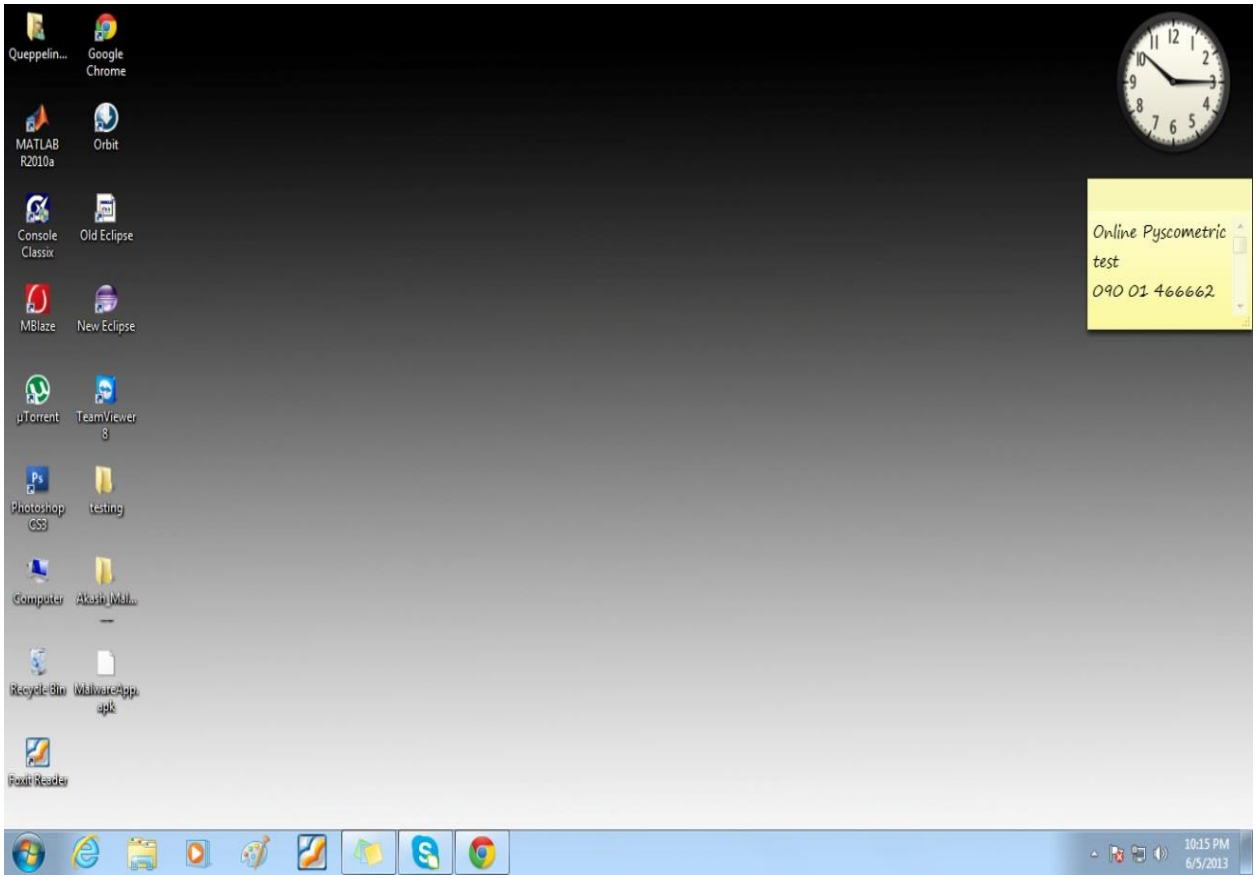
It is generally also referred to as DE compilation.

Using the following tools:

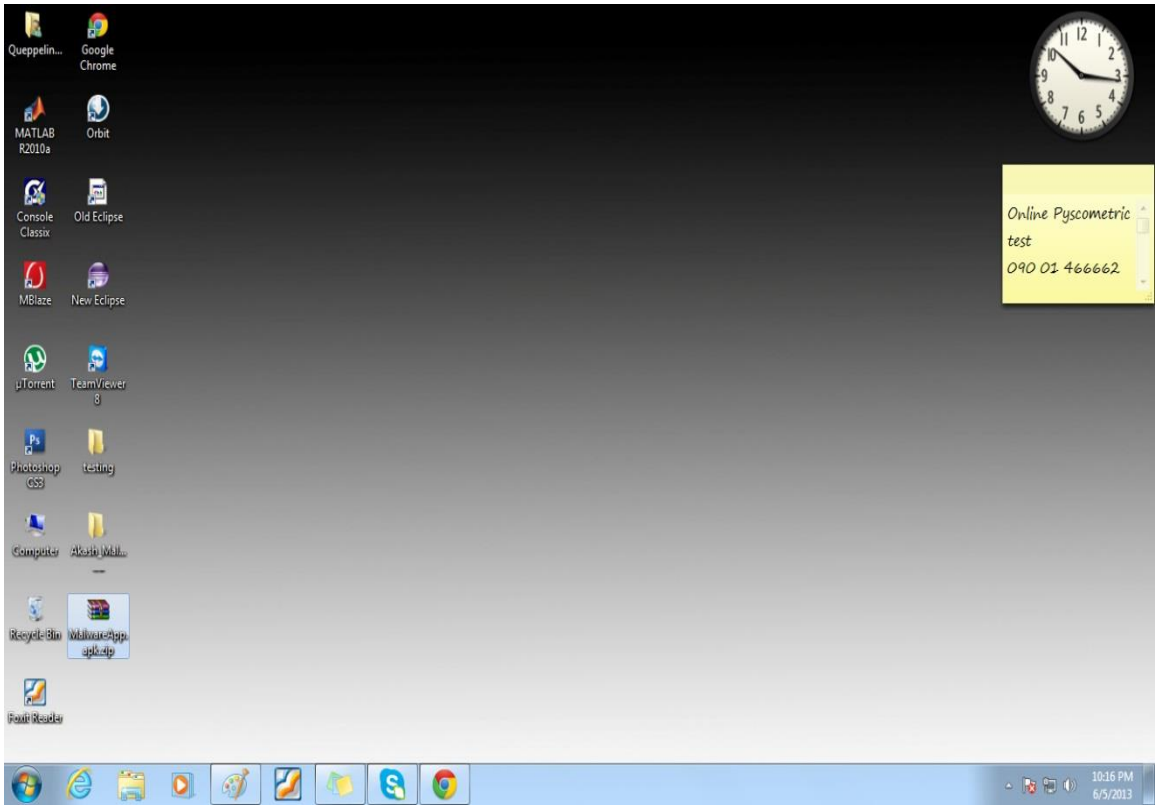
1. WinZip
2. Dex2jar
3. Jd-Gui

We have to follow few steps:

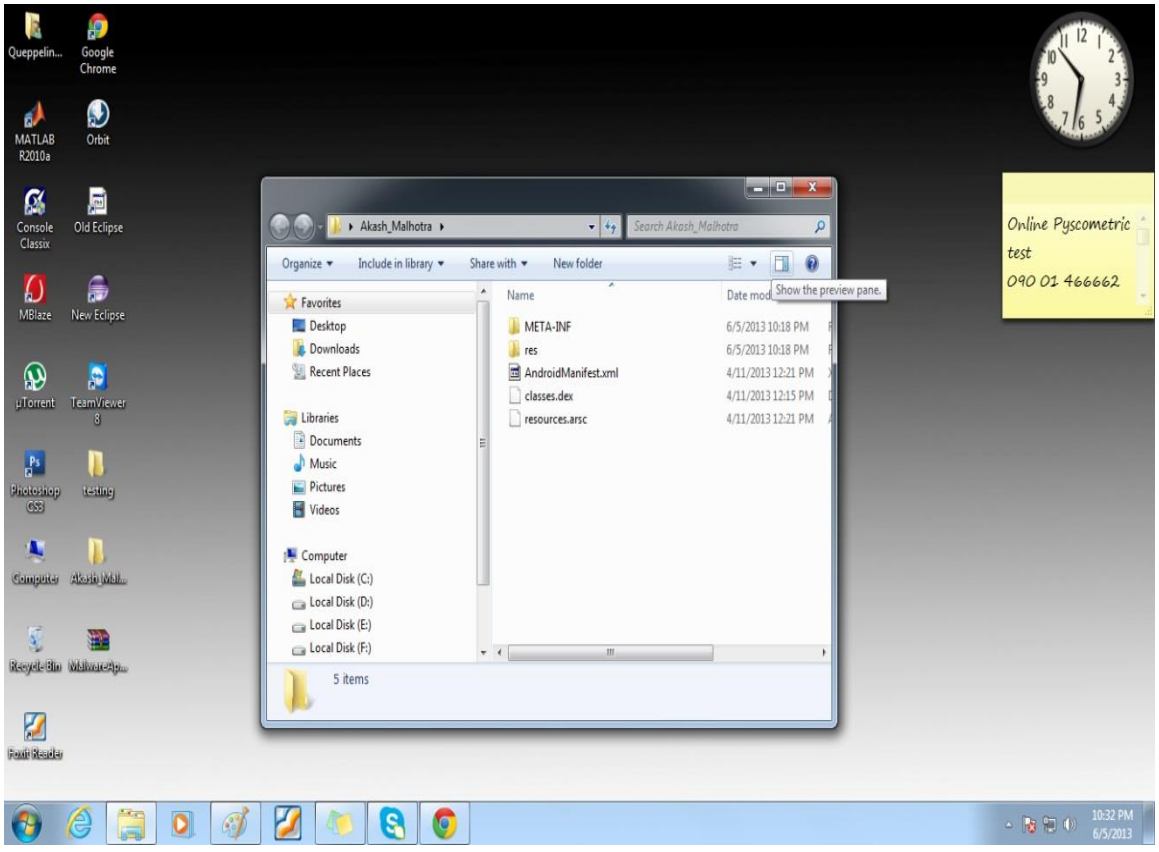
Step 1: Rename the doubtful Malware Android installer .apk to .apk.zip



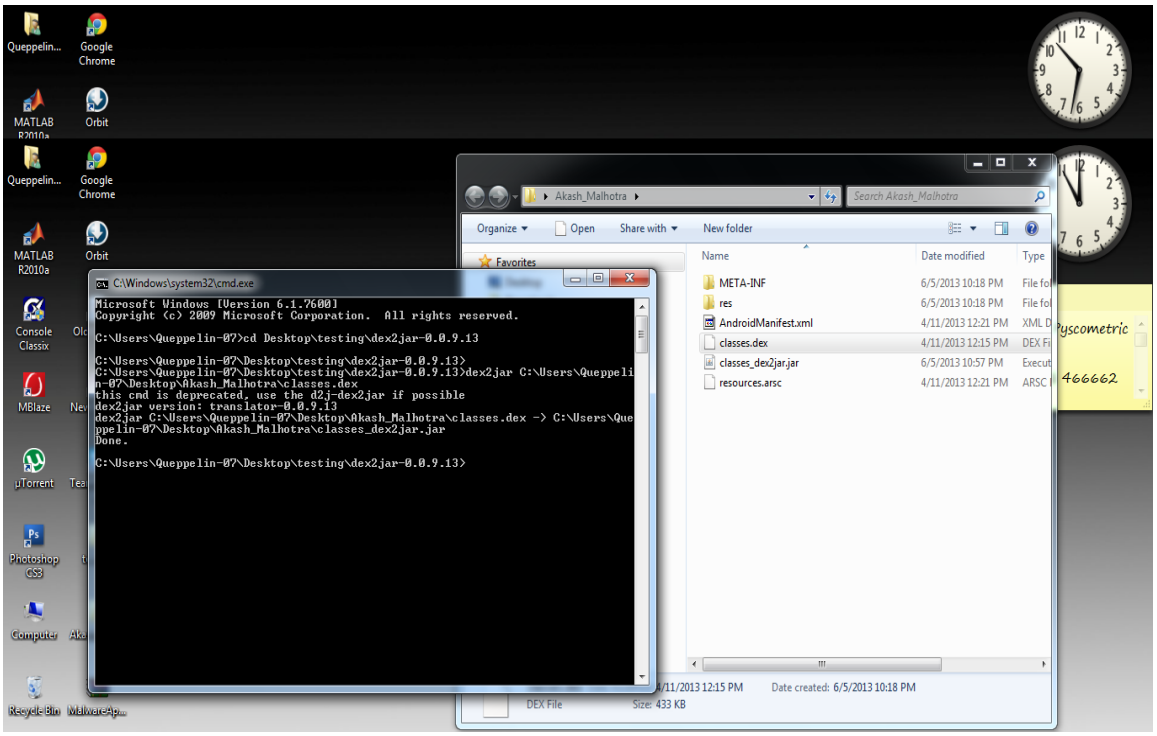
Step 2: Extract to new folder for example say Akash _Malhotra



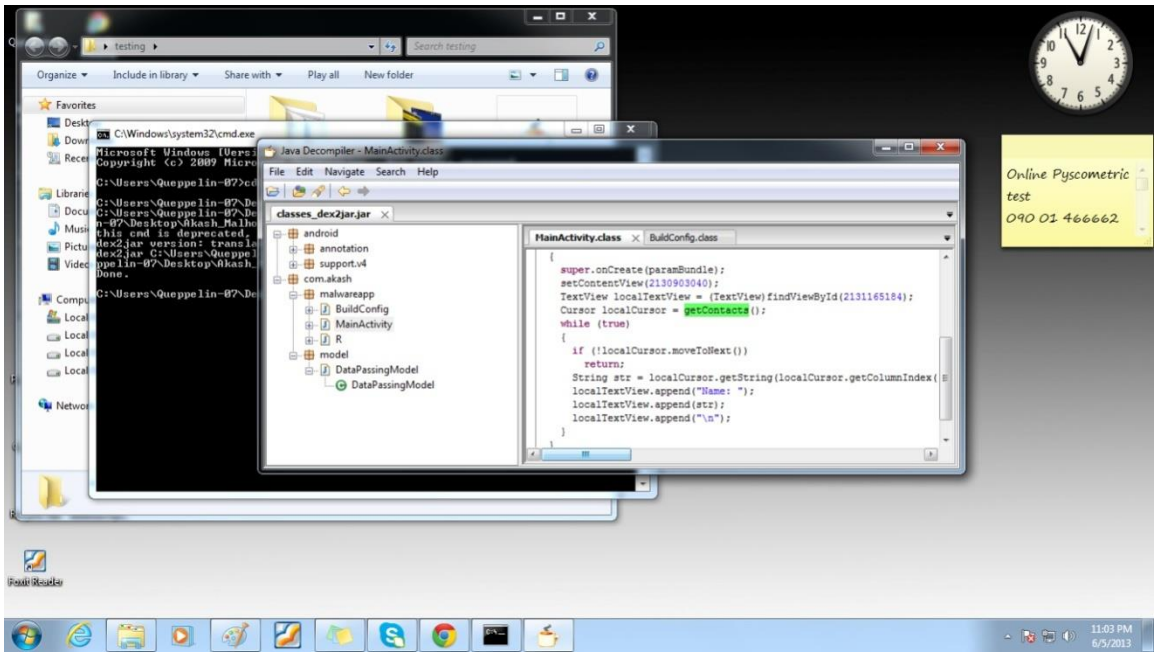
Step 3: Using the tool Dex2jar, convert classes.dex to classes_dex2jar.jar



Step 4: Now use the tool Java Decompiler to view the classes.



Step 5: You can look for the malicious code



Advantages

- It can find weaknesses in the code at the exact location.
- It can be conducted by trained software assurance developers who fully understand the code.
- Automated tools can scan the entire code base.

Disadvantages

- It is time consuming if conducted manually.
- There are not enough trained personnel to thoroughly conduct static code analysis.
- Automated tools can provide a false sense of security that everything is being addressed.
- It does not find vulnerabilities introduced in the runtime environment**.**

4.2 Dynamic analysis

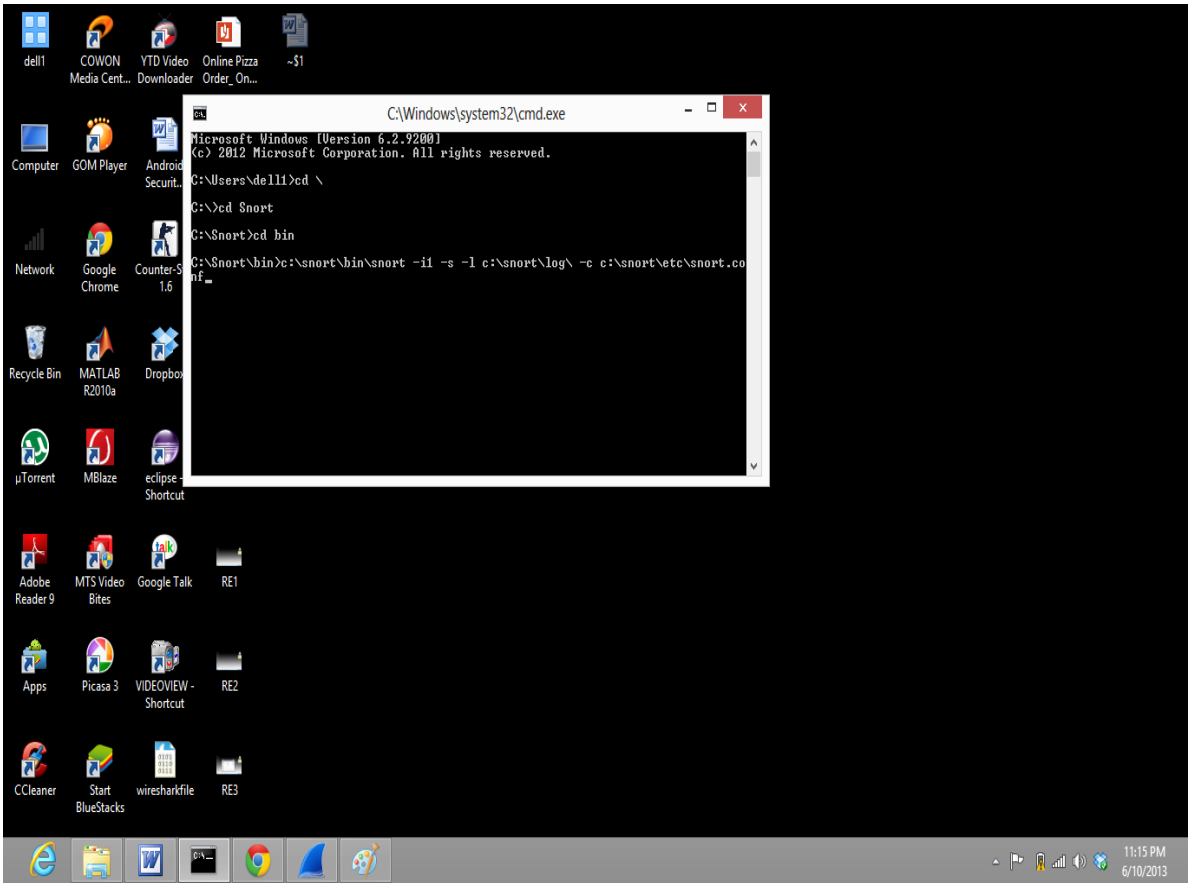
Dynamic analysis: The mobile application environment is different in this case in this we run it on a emulator or vm(virtual machine), so that researchers can check the dynamic behavior of android application. We are first performing static analysis of malware, and the further checking the malware for by checking its behavior. Our research study is done by using the both approach that is static and dynamic. In dynamic we are running the malware on android emulator and checking through snort tool for outgoing packets. Basically we are analyzing packets, for destination address. If the dynamic address is not in white list, the packet will be dropped, so any other device running in that environment is also safe as we are breaking the connection. With the help of snort tool we are generating logs for incoming and outgoing packets. These logs are further analyzed with tools such as wire shark. We can trace the packets. Testing using Taint Droid with 30 popular third-party Android applications states that 16 of them share the location of user with advertisers and eight distribute phone credentials with remote servers without knowledge of user. Many researchers have used Android Monkey (ADB Monkey) for generating inputs, which is not as efficient as testing with real users. Further, this ideology hasn't been tested against a particular malware that shows code fragment encryption or polymorphic behavior. Snort tool catches the incoming

and outgoing packets. We are going to catch the packets and analyze them. We are going to track the malicious code behavior at runtime by installing the suspicious malware, (specifically the one which steals information's) on android emulator. Once the malware makes HTTP Post or get request we will track it running snort on machine and trace the destination address where it is going to send our precious data, so that any other Device coming in that network, will get a warning that this URL does not fall under the white listing URL's.

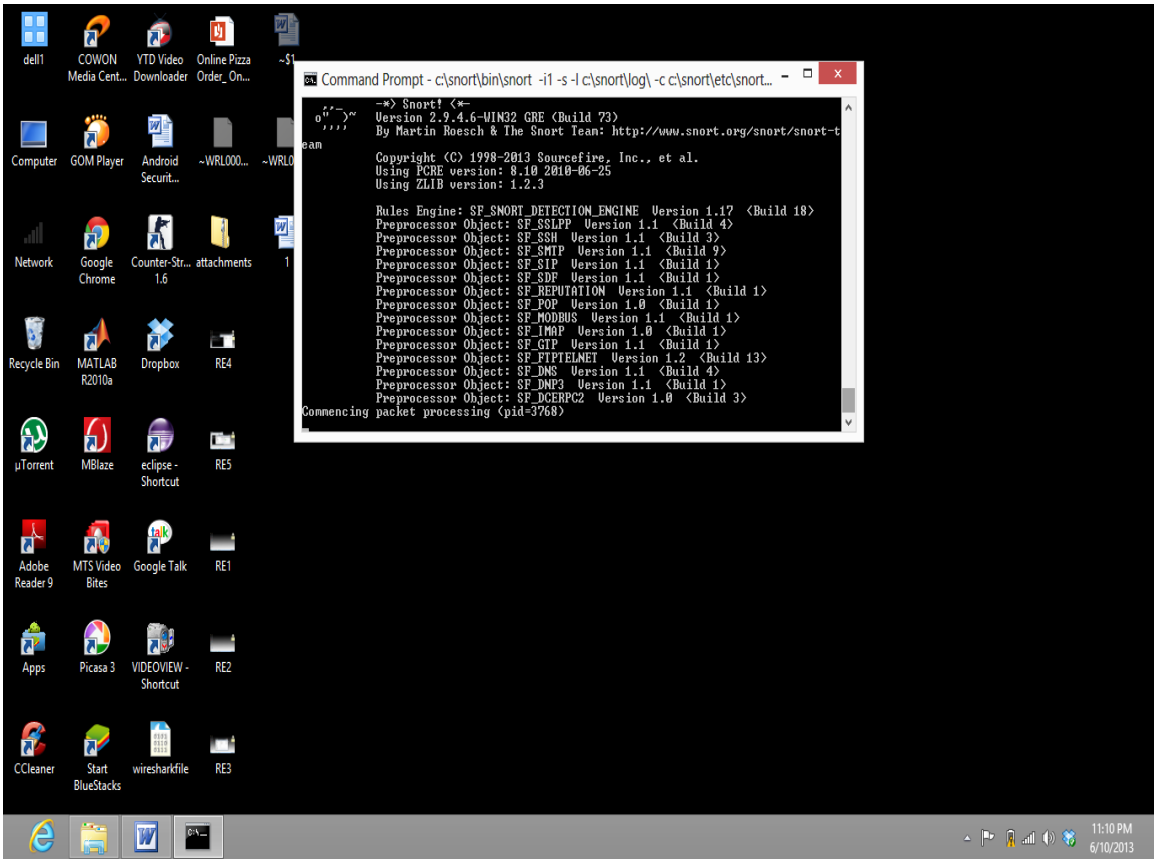
4.2.1 Using Snort tool:

Fig 11: Implemetation of Dynamic analysis

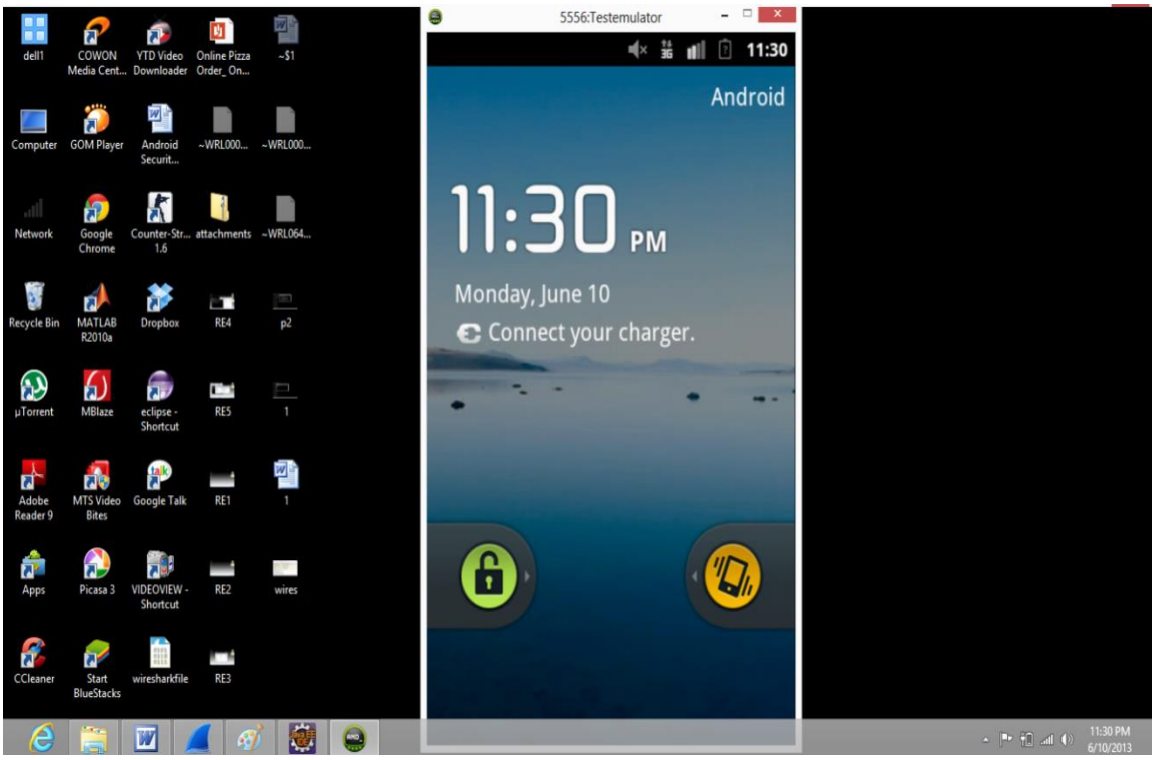
Step 1: Start snort tool



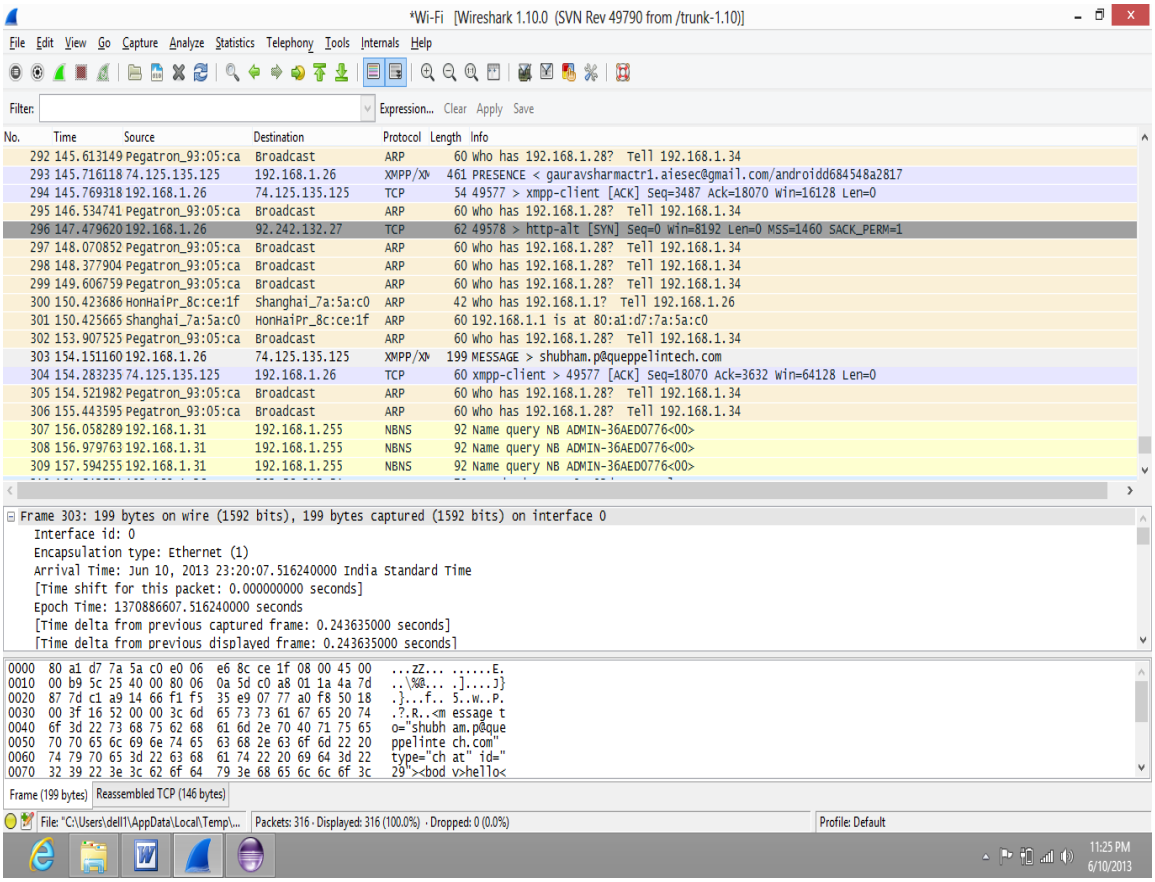
Step 2: Install android malware on Android emulator



Step 3: Run the command and log file will be generated.



Step 4: Trace the packet from where the malware is making connection



Step 5: If it is unsafe, blacklist it or drop the packet.

Step 6: Protect other devices with alert message that this cannot make connection.

4.2.2 Analysis of Packet and Frame

A packet analyzer is known as a network analyzer or protocol analyzer or packet sniffer or other types of networks, such as Ethernet or wireless network sniffer packet analyzer, which is computer hardware or computer programs, capture and protocol in the digital network traffic, or other part of the network.
In the case where the data flow of the stream of network sniffer for each data packet to obtain capture and, if necessary, to decode the original data packet and the packet shows various parameters value, and can further check its contents, or other depending on the exact specifications of RFC. Packet capture is that in which we intercept and record traffic.

Capabilities

On Wired broadcast LANs, depending on the network structure of hub or switch, we can capture the traffic on the respective parts of the network from a single machine in the network, there are some techniques to avoid traffic narrowing by switches to allow access for the traffic benefit other systems in the network (such as ARP spoofing).
For network monitoring purposes, it is also desirable that all data packets on a LAN by allowing them to check the use of a network switch with a monitoring port that made to forward packets that pass respectively through each port of the switch's mirror if (computer) is connected to a switch port called. To provide a network tap water is not suitable solution to use as a monitoring port since taps are not likely to drop packets at high traffic load.

On wireless LANs, we can easily capture traffic on a particularly one channel, or on more channels with using other multiple adapters.

Cable radio and wireless LAN, capturing traffic, rather than unicast traffic is sent to the machine running the sniffer software, multicast traffic is sent to the machine is listening to broadcast a bundle type and broadcast traffic, transport networks are used to capture adapter must be put into promiscuous mode; some sniffers support this, but others do not. In the wireless local area network, when the adapter in promiscuous mode state, the packet does not set the adapter is configured services in general will be ignored. That bag, the adapter must be in monitor mode.

Traffic is captured, whether it is a complete message may be logged, or may be recorded in the header, but there is no record of the entire contents of the packet. This can reduce storage requirements, and to avoid legal problems, but there are still a lot of data needed to diagnose the problem important disclosures.

From the original type of information stored in digital form, and then decoded into readable format, stop protocol analyzer users can easily analyze the information, which is exchanged. Protocol analyzer differ in their abilities, the data show that in the opinion of many, the automatic error detection, to determine the root cause of the error, the resulting timing diagram, reconstruction UDP and TCP data streams and so on.

Traffic can also produce some protocol analyzer and as a reference device, so these can be used as protocol tester. Protocol traffic is generated correctly testers for functional testing, etc., knowing that describes the ability to test the DUT fault ability to cope with such an error condition can have.

Protocol analyzer is hardware-based, either the probe format, such as increased, more commonly, combined with an array disks. These devices record data packet or packet one, a disk array. This allows the user does not have a long history of forensic re-create any error packets without consideration.

Its uses:

The versatility of packet sniffers which means that they can be used to:

- Analyze network problems
- Detect network intrusion attempts

Several tools and techniques were proposed in the past to improve the security of the Android mobile operating system (Alonso Parrizas , 2011) (link). For example, removing the possibility to install any non-authorized software or install an antivirus or antimalware. This could be easily used as part of the preparation phase of the incident handling process, but in this case we will concentrate on our efforts on the network flows.

1. **Preparation:**

During preparation, a security analyst with intrusion analysis skills should be designated. The analyst will most probably pursue Android malware news, security forums, etc, in order to be aware of new malware, which could pose a significant threat for the business. One good resource for Android malware related information is: http://contagiominidump.blogspot.com/ where samples can be downloaded. Also, it is absolutely necessary to prepare and build a lab where the analyst can perform his tests. This lab must consist of:
An Android device: where the malware can be executed. A computer running Linux with USB ports to connect to the Android device and push the malware. Also, this machine needs to have network interfaces to analyze the traffic going through it (Snort, tcpdump, Wire shark). This system will receive the traffic from a mirrored port in a Switch and a Wi-Fi Access point/router with an isolated network (disconnected from the production/development environments) but with Internet access. The traffic from the lab network must be mirrored to the network card of the computer (e.g.: there are several Linksys models which support this functionality).In addition to that, a backup and restore policy must be created in order to recover the Android device after each malware installation. In other words, a clean backup of the base image and the base setup (baseline image) has to be made. This image will then be installed again onto the device after every test with any provided malware is completed.

2. Identification:

During this phase, the security analyst will identify any possible malware that could potentially become an issue for the business. As the malware in question has been already been reported, most likely, its analysis will be publicly available. If such type of report exists, then it is important to understand what network communications is the malware involved with – and simply, what the malware ultimately does (e.g: stealing sensitive information directly from the device and sending it through a TCP connection). Once a sample of such malware exists, the analyst needs to test it himself by pushing it to the Android device and executing. The traffic needs to be captured and stored on the lab computer with snort running. As there probably will not be any signatures available yet, no alert will be fired – unless of course, other signatures matched (e.g: the one regarding the non-standard HTTP ports). With the information gathered from the network flows and utilizing the analyst's skills, it should be possible to create a custom signature for use with snort (as we have done and outlined in the sections 3.2.1 - 3.2.4). This is where the experience of the particular analyst and his knowledge of snort are crucial. Note that should the malware use SSL, the analyst has to use

additional tools and techniques (Fahl, Harbach, Muders, Smith, Baumgärtner and Freisleben, 2012). Once the signatures have been created, the malware can be executed again in order to test if Snort now triggers an alert.

3. Containment:

As the malware is executed in a controlled environment there is no need of applying any additional countermeasures.

4. Eradication:

It is similar as for containment.

5. Recovery:

During this phase and after the detection phase is over, the Android device needs to be restored to its initial state. This means that it needs to be flashed with the baseline image. Even when done, we still need to monitor the network flows in order to detect any anomaly in the outgoing traffic.

6. Lessons learned:

Although these tests are carried out in a contained environment and under full control, we still need to notify Android user's bout the malware its potential impact for the end user (e.g.: information leakage or identity theft).

4.2.3 White listing and blacklisting

White list is a list or register of the entity is used for particular reasons, and have specific advantages, service, liquidity or recognized. Only storage entities on the list will be accepted, approved and / or recognized. White list Blacklist Instead, find the practice of the entity is denied, not recognized. In our approach, we have to keep the URL blacklist and white list,

thus protecting other equipment to install malicious applications list. In this website we have added security

Blacklist (or black list) is a list or register of entities, in particular, are deprived of the advantages of a particular service, mobility, access or rights. Just blacklist can mean to deny someone work in a particular area, or to stop people from your circles. We maintain snort rules under it. In the first case, we will look for malicious software have been reported, but not yet have any snort signature.

Dynamic code analysis advantages:

- It identifies vulnerabilities in a runtime environment.
- Automated tools provide flexibility on what to scan for.
- It allows for analysis of applications in which you do not have access to the actual code.
- It identifies vulnerabilities that might have been false negatives in the static code analysis.
- It permits you to validate static code analysis findings.
- It can be conducted against any application.

Dynamic code analysis limitations:

- Automated tools provide a false sense of security that everything is being addressed.
- Automated tools produce false positives and false negatives.
- Automated tools are only as good as the rules they are using to scan with.
- There are not enough trained personnel to thoroughly conduct dynamic code analysis [as with static analysis].
- It is more difficult to trace the vulnerability back to the exact location in the code, taking longer to fix the problem.

Detection of 0-day malware

This scene is quite different as there is no signature, no definition, and no information of our malware. As a result, the analysis in terms of definition is much difficult and in many other cases; the malware will not be caught in the early stages. The incident handles this process will therefore be a bit different from the previous one.

1. Preparation: In this phase, we have to run a production environment all Android handsets are deployed in accordance with the architecture proposed in sections 2.1 and 2.2 – including the steps described in the paper 'Securely Deploying Android Devices. This means that in real time all the traffic is captured and is verified by an instance of Snort. Under this phase, an Incident handler has to be designated in order to perform the analysis once a mobile device becomes compromised. For recovering such a device later, it is important to have a backup policy, in which either daily or weekly backups of each Android device are done and kept in secure repository (there are several tools available on the market). The incident handler will be having SSH access through keys to each Android device in order to perform investigations.

2. Identification: This is the most difficult part, as there are no signatures, which could detect the malware in a possible manner and triggers an alert.

Packet Structure

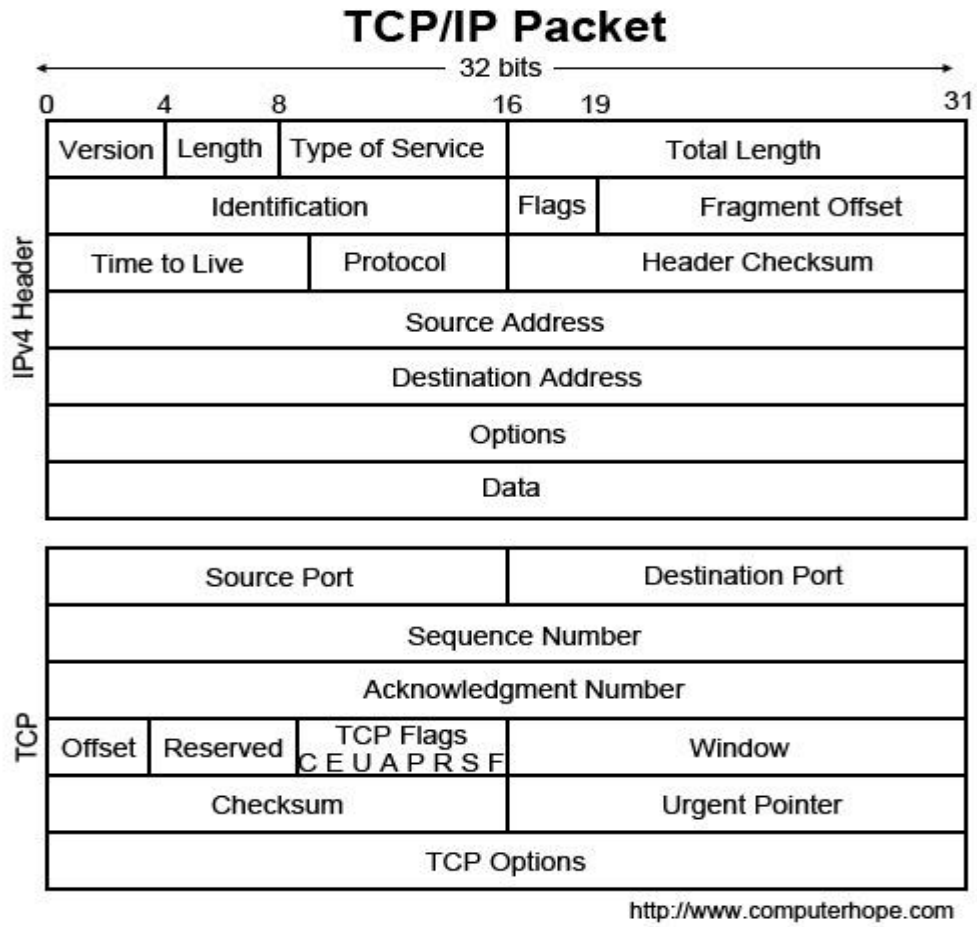


Fig 12: TCP/IP Frame

• Source Port 16 - send port identification

• Destination Port 16 - Identification of the receiving port

• Serial number 32 - has a dual role:

If the SYN flag is set, then (1), which is the initial sequence number. Number. The original first data byte, and confirm that no. In the next ACK, then add the serial number 1.

• If the SY N flag is clear, then (0), it is the cumulative sequence. Segment of the current session of the first data byte.

• Make sure the serial number 32 - if the ACK flag is set, then the value of this field is the corresponding serial number. The receiver is expected. Therefore, it acknowledges receipt of all bytes before (if it exists). Sending a first end of each of the other end of the ACK starting serial number. Itself, but no data.

• Data offset 4 - specification of the TCP header size, then 32. The minimum size of the first five characters of the maximum word size of the head is 15, thereby providing a minimum

size of 20 bytes and a maximum size of 60 bytes, allowing up to 40 bytes in the header option. Such fields find its name from the fact that it is a TCP segment from the original raw data of the offset.

• Reserved 3 - for future use, must be set to zero

• Flag 9 (also known as control bits) - including nine one flag

NS 1 位 - ECN-random numbers concealed protection has been added to by the RFC 3540 headers.

• Seamless Line 1 - means reducing congestion window sign which is determined by the sending host to indicate that it has been a TCP segment with the ECE flag set to respond congestion control mechanism; it is added to by the RFC 3168 headers.

• ECE 1 位 - ECN echo indicates

• If the SY N flag is set (1), then the ability to TCP ECN.

• If the SY N flag is clear (0), then a package, with the experience congestion flag in the Internet Protocol header set obtained by adding to the normal transmission by the RFC 3168 headers.

URG bit - is vital to show the urgent pointer field

• ACK 1 Bits - Confirm field is very important. Start the client sends a SYN packet, which is all of the data packets; you must have this flag set.

PSH 1 位 - Push functionality, It is asked to push buffer receiving application data.

• RST 1 Bits - Reset the connection

• SYN 1 Bits - synchronization sequence. Only from each end of the first data packet is sent, you must have this flag set. Other signs also changed the meaning of this logo is based on, and some only applies when the flag value is set, and others will be very obvious.

• FIN 1 bit - No data from the sender

• Window sizes 16 - receive window size, which indicates no. Average size of the window unit does not exceed the sequence of bytes. In the Confirm field sender of this segment is currently willing to receive

• Parity 16 - 16 checksum field data and header error checking deployment.

• Urgent Pointer 16 - if the URG flag is set, then the 16-bit field is not offset from the sequence. Refers to the last byte of urgent data and

• Options Variable 0-320 bits, which means that about 32 - Adjust the length of this field is the offset data in the field. Options include three areas: - Option real one byte, the option data

variables, option length 1 byte. Option-Kind field indicates the type of option, and is the only field, which is not optional. It depends on what kind of options, we actually dealing with the two remaining areas, therefore, the option length field indicates the length of the option, the option data, including the value of the option field, if it can be set to be suitable. For example, the option-kind byte instructions given to 0x01, which is a no-operation option is only deployed to fill, is not there an option length or data bytes, which is following it. OK option byte 0 end, only a single byte. OK byte is 0x02, indicating that it is the largest segment of the market size, it should be 0x04's. It should be noted that this length is provided by the option field, including the "OK" and "OL total length in bytes. Accordingly, in the usual MSS value of 2 bytes, which means the length of the parameter must be 4 bytes, or 2-byte type and length of words, will the MSS option field contains a value 0x05B4 displayed in the TCP option 0x04 0x02 in the 0x05option and will pursue B4 Options section in the TCP MSS field specifies the length of the bytes.

• Fill - TCP header TCP header padding deployed to ensure both ends and data begins a 32-bit boundary. From zero padding.

Some options may be sending a SYN is set as follows. Types and options available in standard lengths (OK, OL).

• 0 8 Bits - the end of the option list

• 1 x 8 - no operation NOP, which is used to fill a better performance on 32-bit boundary alignment option parameter.

2,4 SS 32 spaces - maximum segment size (MSS)

3,3 S 24 spaces - window size

• 4 x, two 16 - to allow selective acknowledgment.

• 5, N, BBBB, EEEE, variable bit rate, N is 10,18,26,32 or 34 - selective Acknowledgement SACK of. 1-4 blocks being acknowledged selectively list, specified as a 32-bit pointer to the beginning or end of the pursuit of the first two bytes.

8, 10 TTTT EEEE 80 bits) - Timestamp and echo the previous timestamp

14,3 S 24 Bits - Transmission Control Protocol Alternate Checksum Request

• 15, N, variable bit - alternate checksum data transmission control protocol.

(Other options are experimental, outdated, not standardized, or unassigned)

Analysis

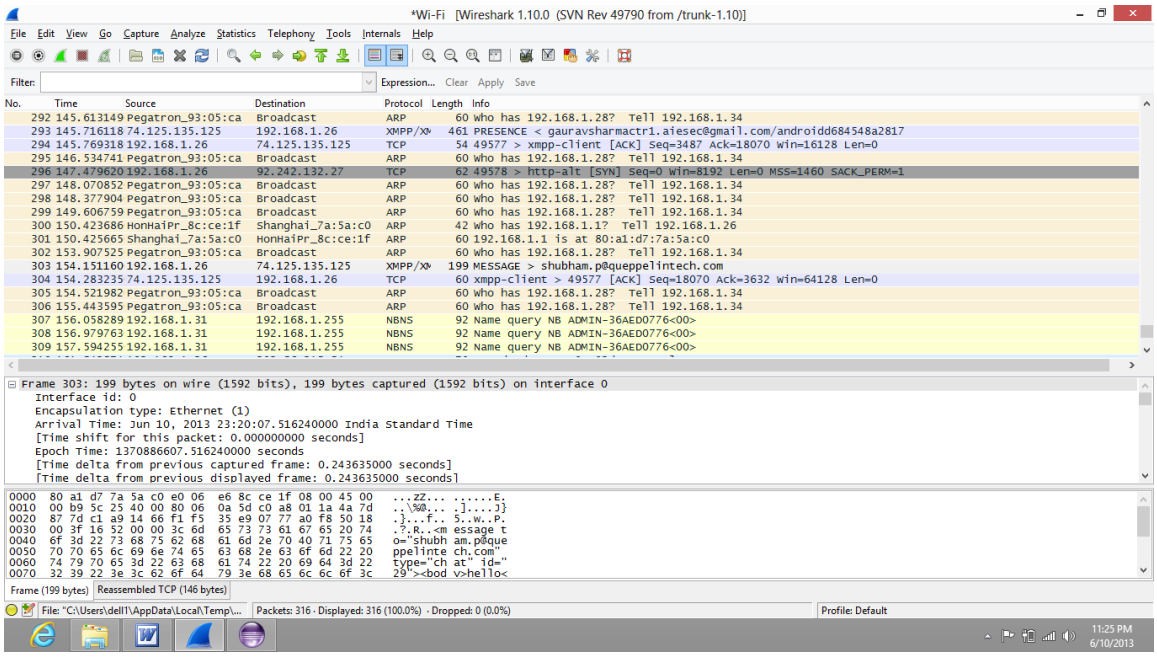


Fig 13: Detecting of Packet Destination Address

We are going to analyze this report with the help of tool name as Wire shark. In this we have traced out the destination address where the malware is forwarding data, once the destination address is traced out by running malware on android Emulator we can perform this task in an ad-hoc network. We can connect android device to the ad-hoc network and stop this malware from getting installed. Since we have traced the destination address, we can blacklist the URL or destination address either in router devices or we can perform this with the help of Snort tool. We can perform this by writing white listing rules.

This approach can stop the malware getting installed on a various devices, connected to that ad-hoc network.

We can create ad hoc network with the android devices and provide security to it, by not allowing the malware to get installed on it

How to Create Ad hoc network?

1. You can open the "Start" menu.

2. Now click Control Panel.

3. Now click "Network and Internet".

4. Now click on the Options Network and Sharing Centre.

5. In the "Change your network settings, click the" Start a new connection or network. "

6. Select "Set up a wireless temporary (computer to computer) or a network (computer-to-phone).

7. Click "Next" twice.

8. Enter the network name such as "MalwareTestNetwork".

9. You can choose to secure WAP (or WEP).

10. Enter the security key or password.

11. Check the Save this network "checkbox.

12. Click "Open Internet Connection Sharing.

Continue to Part 2 below.

Part 2: Connect your mobile device

1. On your Android phone's main menu, click on "Settings" icon.

2. Click Wi-Fi.

3. Your ad hoc networks "MalwareTestNetwork" should appear in the list.

4. Click on your network.

5. Enter the password.

Now you can connect to the network

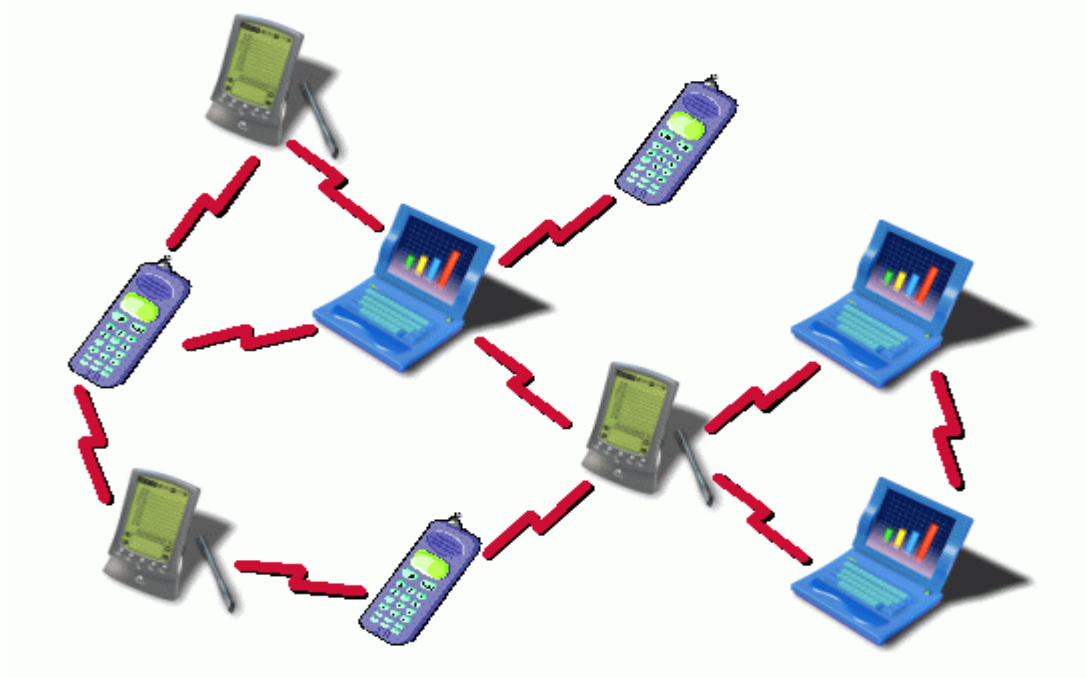


Fig 14: Creating an adhc network

By performing this we can manage various devices with our tool snort and not allow them the false application to get installed on android device.

4.3 Third Party Application

Lookout Mobile Security-antivirus has suggested a few mitigation steps whichare adopted many of users in order to keep their devices free of malwares

The user downloads such applications from trusted and reputable sources or application stores, such as Android Market or Google Play for Android devices. The user should bear in mind to application few points

1. It is very important to cross check that the web link address should match the website it is supposed to be.
2. A password should always be set on the device in cases of theft so that it is tough to access data on the device.
3. Firmware updates can be seen for the device must be downloaded whenever they are available.
4. The user must always be alert and pay attention with respect to any strange behavior on the device such as sudden decrease in battery life, odd text messages, and additional charges to the phone bill, etc. These could indicate that the device is compromised.

Data set-Lookout

When Tested with Lookout Third Party Application, we found 250 Malwares out of 2000 application.

200—47,400—52+47,600—53+52+47,800—62+53+52+47, 1000-36+62+53+52+47

Fig 15: Scanning for Malware with Lookout application

4.4 Tools:

1. Eclipse

Eclipse is a community designed for organizations and individuals who wish to work together on business-friendly open-source software. Its projects are concentrated on building an open development platform which is made up of extensible tools, frameworks, and runtimes for deploying, building and managing software across the lifecycle. The Foundation of Eclipse is a not designed for profit, **Eclipse projects** are hosted by member supported cooperation and helps cultivate both ecosystem of complementary products and services and open source community.

In November 2001, IBM created the Eclipse Project and it was sustained by consortium of software vendors. Eclipse Foundation is a self-supporting not-for-profit corporation which was created in January, 2004 to act as representative of Eclipse community. This self-supporting not-for-profit corporation was built to allow a vendor open and neutral, transparent community which was to be made around eclipse. Presently, Eclipse community contains organizations and individuals from a cross section of software industries.

Our members fund the Eclipse Foundation and it is governed by Board of Directors. Strategic Consumers and Strategic Developers are holding seats on this Board as representatives elected by Open Source committers and Add-in Providers. A full time professional staff is employed by the Foundation to grant services to the community but open source developers known as committers are not employed, which are actual workers on Eclipse projects. Organizations typically employ these Eclipse committers or they are self-supporting developers which volunteer time for working on open source project.

Generally, four services are provided by the Eclipse Foundation to the Eclipse community these are as follows:

1) **IP Management**
2) **Ecosystem Development**

3) **I T Infrastructure**

4) **Development Process**

Many staff who works full-time is related with each of these areas and work with the greater Eclipse community to help in meeting the requirement of the stakeholders.

2. Android SDK

The Android development kit (SDK) includes a complete set of development tools, which includes a handset emulator based on QEMU, a debugger, libraries, documentation, sample code, and also learning tutorials. Lately supported development platforms include systems which runs on Linux any modern desktop Linux distribution, Mac OS X 10.5 or later version, Windows XP or later version; for the moment one cannot create Android software on Android platform environment by itself. The officially supported integrated development environment -IDE is Eclipse using the Android Development Tools (ADT) Plug in, though IntelliJ. IDEA IDE uses all editions that also support Android development out of the box, and Net Beans IDE also supports Android development through a plugin. In addition, developers can use any text editor for editing Java and XML files, then use command line tools such as Java Development Kit and Apache Ant which are required for creating, building and debugging Android apps as well as control attached Android phones for example trigger a reboot or installing a software package(s) in a remote manner).

Android's SDK takes use of enhancements which required very much with overall Android platform development. The Android SDK also supports older versions of the Android platform if developers wish to target their apps at older devices. Tools for development are downloadable components, so after someone has downloaded the latest version and platform, old platforms and tools may also get downloaded for looking a compatible testing.

Android applications are packaged in .apk format and stored under /data/app folder on the Android OS (the folder can be accessed only to the root user for security purpose). APK package contains .dex files compiled byte code files called Dalvik executable or resources files etc.

3. Java Decompiler

JD-GUI is an independent graphical utility which shows Java source codes of ".class" files. We can easily browse reconstructed source code along with the JD-GUI for fast access to fields and methods.
For non-commercial use JD-GUI is free. It means that JD-GUI will not be embedded or included into commercial software products. But this project can be freely used personally in non-commercial or commercial environments.

4. Dex2jar

There are four components contained in Dex2ar
1. Basically built dexamethasone readers to read the Dalvik executable files that .dex / .odex format. It includes a lightweight API like ASM.
2. DEX-translator to perform the conversion work. Agile format translation of DEX-IR, and further optimized, to convert read DEX ASM format instruction.
3. DEX-IR deployment translation dexamethasone, specifically built to represent an indexed instruction
4. Agile tool that works with the class file. Some examples are:
a) DE obfuscate a jar file
b) Modify or change an apk file

Snort is an open source tool for network prevention, intrusion detection system (IDS / IPS) is a source of fire brands. If we merger agreement, checked exceptions and signature bonus, it is usually deployed IDS or IPS technology around the world. With about millions of downloads and approximately 400,000 registered users, Snort is to become the de facto standard for IPS.

6. Wire shark

It is the world's foremost network protocol analyzer. Help, we can easily see in our network is how, at the micro level. It is in fact a general legal principle in a variety of educational institutions and industry standards.

Its booming worldwide network of all these experts contributed gratitude. It is initialized in 1998 and is still continuing.

Features and characteristics

Wire shark contains a rich feature set, some of the following:

• Deep inspection of hundreds of protocols, many of them are added all the time.

• Real-time capture and offline analysis.

It has a standard three-pane packet browser

• It is a multi-platform: runs on Windows, Linux, Net BSD's, Mac OS X is, Solaris,
Free BSD's, and many others.

• Capture network data can browse through a GUI, or via the TTY-mode tshark tool in the industry is the most powerful display filters.

• Rich VoIP analysis is completed.

• Read / write many different capture file formats: tcp dump's libpcap's Catapult DCT2000 PCAP Wu Cisco Secure Intrusion Detection System, Network Monitor Microsoft (MSN), Network General Sniffer (compressed and uncompressed), Sniffer ® PRO, net radiation, network equipment observed, Screen / Finisar Corporation Shomiti Novell company LANalyzer, surveyors, RADCOM WAN / LAN Analyzer, Tektronix K12xx, Visual Networking visual uptime, Ether Peek wild pack / Airo Peek's / Token Peek of of, and many others snoop

• Capture files can be decompressed on the fly with gzip compression tool

• Real-time data is read from Ethernet, PPP / HDLC, Frame Relay, IEEE 802.11, Token Ring, FDDI ATM, Bluetooth, USB, and others depend on your platform

• Decryption support for multiple protocols, including the IPSec, Kerberos identity, WEP, ISAKMP devices, SNMPv3, SSL / TLS and WPA/WPA2

• Coloring rules apply to quick and intuitive analysis of the packet list

• Output Export to XML, CSV's PostScript ®, or plain text

7. TCP Dump

It is a common packet analyzer, can be completely controlled via the command line. It allows the user to easily intercept and the TCP or IP, and is attached to other computer systems through a network to receive or send packets. This was originally a distributed BSD license, and completely frees software.

tcpdump acting on many Unix likeos's like Solaris, Mac OS X, Linux's, BSD, AIX and HP-UX and so on. In their system, tcpdump use library thelibpcap capture packets. tcpdump port for Windows is called as WinDump of, it uses WinPcap and port Windows flibpcap's.

4.5 Procedure

Our approach is to increase the detection rate of malwares, specifically the one which steals information from android devices by performing our static and dynamic approach

In static approach we are going to check the android application for malicious code by performing reverse engineering of android installer (.apk file). This installer file is being checked for malicious code by decompiling the file with the help of following tools mentioned in previous section. Tools used are Java Decompiler,Dex2Jar,WizZip etc.

This will look for HTTP request and HTTP response being done without users notice or any background service which takes away the precious data. This is called static as it is code based. Our next approach is dynamic which is done on network layer to perform further analysis of malicious activity. This is dynamic as it is checked on behavior.

For dynamic approach we are using android Emulator to install the malware then we are using, the Snort tool which is an IDS (Intrusion Detection system), to generate log for incoming and outgoing traffic. For viewing log files we are using Wire shark tool.

Once the android malware is running we can catch the activity of it through delivering packets. The frames can be further processed for checking the Destination Address.

Once we found the destination address we can perform blacklisting of the URL. So that if any other device running the malicious app, could be alerted with the popup message. Do you want to allow your device to connect to particular URL? If user allows then he may be the victim of loosing precious data otherwise he will be alerted.

Our approach has taken 1000 samples out of which they are clustered according to permission based. We will be forming four clusters ABCD

In cluster A-Permission using Network Communication

In cluster B-Risky permission with Network Communication

In cluster C-without Network Communication
In cluster D-Other Harmful nature of an App can be traced

We will be performing our analysis on cluster B and compare them with third Party application antivirus efficiency - Lookout, AVG etc.

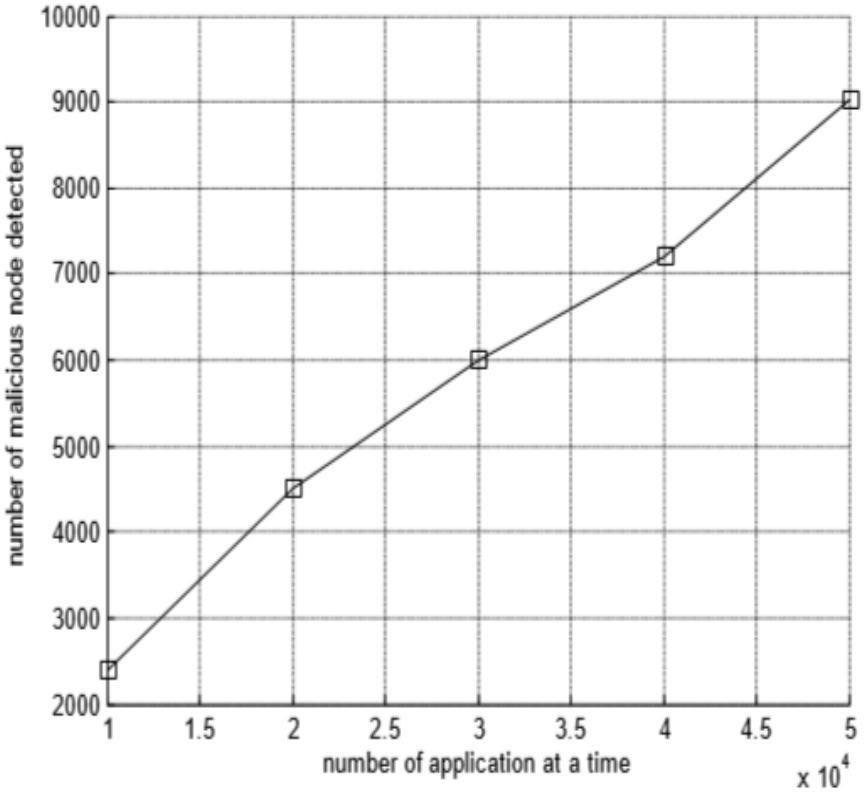
Chapter 5: Results and Experiments.

For large volume of data

Fig 16: Standard graph-Lookout

Table1

s/n	Total number of dataset	Total number of malicious node(standard)
1	10000	2400
2	20000	4502
3	30000	6001
4	40000	7202
5	50000	9020

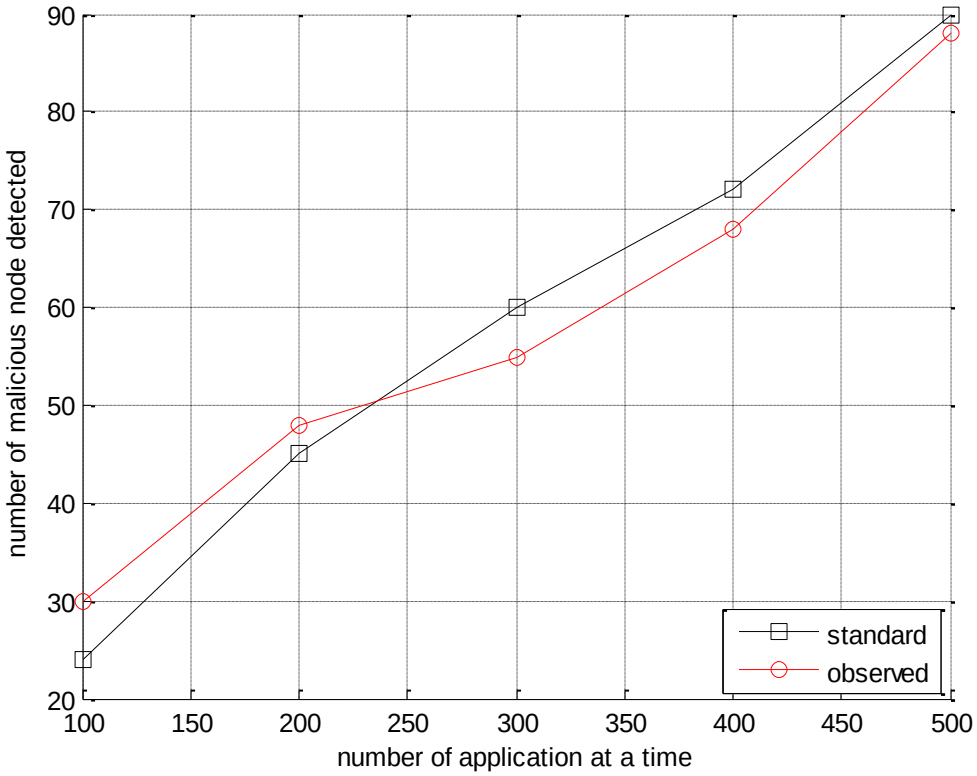


For small volume of data

Fig 17: Observed graph (my methodology) with respect to standard graph

Table2

Total number of dataset	Total number of malicious dataset (standard)	Total number of malicious dataset (observed)
100	24	30
200	45	48
300	60	55
400	72	68
500	90	88



Chapter 6: References

1. Adam, P. F , Chauduri, A., & Foster, J. S. (2009). Scan Droid: Automated security certification of android applications. In IEEE symposium of security and privacy.
2. D. Barrera, H. G. Kayacik, P. C. van Oorschot, and Somayaji. A methodology for empirical analysis of permission-based security models and its application to android. In the Proceedings of the 17th ACM conference on Computer and communications security, CCS '10, pages 73–84, NY, USA, 2010. ACM
3. A. Shabtai, U. Kanonov, Y. Elovici, C. Glezer, Y. Weiss: Andromaly: a behavioral malware detection framework for the android devices. Journal of IIS i.e. Intelligent Information Systems 38(1) (January 2011) 161.
4. Enck, W., Gilbert, P., Chun, B.-g., Cox, L. P.,Jung, J., McDaniel, P., and Sheth, A. N.TaintDroid: An Information-Flow Tracking System for Real-time Privacy Monitoring on Smartphone's. In the Proceedings of the 9th USENIX Symposium on Operating Systems Design and Implementation (2010),USENIX OSDI '10
5. Contagio mobile malware mini dump. https://contagiominidump.blogspot.com/
6. M. Nauman, S. Khan, and X. Zhang. Apex- Extending Android Permission Model and Enforcement with User Defined Runtime Constraints. In the Proceedings of the 5th ACM Symposium on Information, Computer and the Communications Security, ASIACCS '10, 2010
7. Risk Ranker https://www.csc.ncsu.edu/faculty/jiang/pubs/MOBISYS12.pdf
8. Felt, A. P., Chin, E., Hanna, S., Song, D., and Wagner. Android Permissions Demystified. In the Proceedings of the 18th ACM Conference on Computer and Communications Security (2011), CCS 11.
9. Mal Genome: Proceedings of the 33rd IEEE Symposium on Security and Privacy San Francisco, CA, May 2012
10. Felt, A. P., Wang, H. J., Moshchuk, A., Hanna, S., and Chin, E. Permission Re-Delegation: Attacks and Defenses. In the Proceedings of the 20th USENIX Security Symposium (2011), USENIX Security '11.
11. Lookout Mobile Security. https://www.mylookout.com/.
12. A. P. Felt, H. J. Wang, A. Moshchuk, S. Hanna, and E. Chin.

13. Permission Re-Delegation: Attacks and Defenses. In Proceedings of the 20th USENIX Security Symposium, USENIX Security ’11, 2011
14. M. Ongtang, S. McLaughlin, W. Enck, and P. McDaniel. Semantically Rich Application-Centric Security in Android. In Proceedings of the 2009 Annual Computer Security Applications Conference, ACSAC ’09, 2009
15. M. Nauman, S. Khan, and X. Zhang. Apex: Extending Android Permission Model and Enforcement with User Defined Runtime Constraints. In Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security, ASIACCS ’10, 2010.
16. L. Xie, X. Zhang, J.-P. Seifert, and S. Zhu. pBMDS: A Behavior-based Malware Detection System for Cellphone Devices. In Proceedings of the 3rd ACM conference on Wireless Network Security, WiSec ’10, 2010
17. W. Zhou, Y. Zhou, X. Jiang, and P. Ning. Droid MOSS: Detecting Repackaged Smartphone Applications in Third-Party Android Marketplaces. In Proceedings of the 2nd ACM Conference on Data and Application Security and Privacy, CODASPY’12, 2012.
18. ang, and V. W. Freeh. Taming Information-Stealing Smartphone Applications (on Android). In Proceeding of the 4th International Conferenceon Trust and Trustworthy Computing, TRUST ’11, 2011.
19. Dong-Jie Wu1, Ching-Hao Mao2 “DroidMat: Android Malware Detection through Manifest and API Calls Tracing”2012 Seventh Asia Joint Conference on Information Security. 978-0-7695-4776-3/12 /IEEE