

Observational Report LAB 2: Packet sniffing and spoofing

ACS 545| Cryptography and Network Security

January 24, 2024

Ashutosh Mishra

Professor: ZesHeng Chen

* Observation Criteria: -

Note: -

1. Used Ubuntu Seed (v20.04 Focal) on Google Cloud Platform
2. Used Real VNC Viewer to perform the actions on GUI
3. All actions are being performed under username - **seed**

* Environment Setup: -

A folder named Crypto was created and the Labsetup file was downloaded from the Seed Documentation site. Post unzipping the folder we find a docker-compose.yml file which contains the configurations for three machines that are connected to the same LAN. The three machines are: -

1. Attacker (10.9.0.1)
2. Host A (10.9.0.5)
3. Host B (10.9.0.6)

To create the three containers using the .yml file, firstly one can, use either **docker-compose build** or **dcbuild** (alias).

A screenshot of a computer

Description automatically generated

As per the attached image, we find that the steps are being skipped. The reason for this is because the containers have already been created. Now we will use either the **docker-compose up** or **dcup** (alias) to start the containers.

A screenshot of a computer

Description automatically generated

Now, as per the above screenshot, we can see that the containers are up. However, just to cross check we can use either **docker ps –format “{{.ID}} {{.Names}}”** or **dockps** (alias) to show the status of the containers.

A screenshot of a computer

Description automatically generated

To run a particular container we can use first the **dockps** command to find the container name and use the **docksh <id>** command to start the shell on the target container.

A screenshot of a computer

Description automatically generated

To exit from the shell, we can type exit to get back to seed shell.

A screenshot of a computer

Description automatically generated

For this lab we will be requiring a name for our interface in vm. For this we can get it either using the **ifconfig** command or using the **docker network ls** command.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

* Task 1 (Sniffing Packets): -
  + Task 1.1 A (Perform Packet Sniffing using Scapy with and without root privilege): -

For the first task, we are using the same code that is described in the document. This program simply provides the sniffed packet information.

A screenshot of a computer

Description automatically generated

To run this program, we can simply use **./Task1.1A.py**. While running this program we get an error message “**Permission Denied”**. The reason for this is that for running scapy one shall be using root privilege, and the mode of file shall be **a+x**.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Post using **sudo./Task1.1A.py** the code runs fine.

A screenshot of a computer

Description automatically generated

From host A ping to host B.

A screenshot of a computer

Description automatically generated

The sniffing program sniffs the packet and shows output as below: -

A screenshot of a computer

Description automatically generated

* + Task 1.1 B (Perform Packet Sniffing using Scapy using BPF): -
    - Capture only the ICMP pack: - For this task I’ve referred and used the example code provided in BrightSpace.

A screenshot of a computer

Description automatically generated

Post running the new code i.e. **sudo ./Task1.1B\_1.py** the program runs and the filtered ICMP packets are captured as below:-

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

* + - Capture any TCP package that comes from a particular IP and with a destination port 23: - For this task we need to modify our source code slightly as mentioned below.

A screenshot of a computer

Description automatically generated

Now we run the task using **./Task1.1B\_2.py**

A screenshot of a computer

Description automatically generated

Post this we will telnet from 10.9.0.6 to 10.9.0.5 and capture the packets

Screenshot from 10.9.0.6

A screenshot of a computer

Description automatically generated

Screenshot from sniffer program

A screenshot of a computer

Description automatically generated

* + - Capture packets from a particular subnet: - For this task we need to modify our source code slightly as mentioned below.

A screenshot of a computer

Description automatically generated

Post Executing the program using **sudo** **./Task1.1B\_3.py**

A screenshot of a computer

Description automatically generated

Now we enter one of the hosts and ping to some ip in subnet and to 8.8.4.4 We will find that the packets are only captured for the defined subnet.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

* Task 2 (Spoofing ICMP Packets): -

For the first task, we use a modified program of the example provided on Brightspace.

A screenshot of a computer

Description automatically generated

Here I have kept the destination IP for a Target host in our container and the Source Ip as an arbitrary one.

Now we will perform a tcpdump to capture the packets and save in tmp. A screenshot of a computer

Description automatically generated

Post that we will run the program

A screenshot of a computer

Description automatically generated

And we see that the number gets incremented.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

* Task 3 (Traceroute): -

For traceroute implementation I have used the following code.

A screenshot of a computer

Description automatically generated

Post running this code the output is as below:

A white background with black text

Description automatically generated

* Task 4 (Sniffing and then Spoofing): -

For this task the code provided in Brightspace was tweaked for ICMP as mentioned in the screenshot below.

A screenshot of a computer

Description automatically generated

Post running this output for

1. ping 1.2.3.4 the output was as below.

From the target system output was

A screenshot of a computer

Description automatically generated

From the attacker system output was as

A screenshot of a computer

Description automatically generated

After every ping it took more time than usual

1. ping 10.9.0.99 the output was as below.

From the target system output was

A screenshot of a computer

Description automatically generated

We don’t get any output on the attacker machine as the host itself is unreachable on physical network.

1. ping 8.8.4.4 the output was as below.

From the target system output was

A screenshot of a computer

Description automatically generated

From the attacker system output was as

A screenshot of a computer

Description automatically generated

Here we get two replies, one which was originally intended to send and another that was spoofed.

* Conclusion: -

The overall experience was exciting. I will be going deeper into the subject and try to discover new ways to handle how we had practiced in lab. Having a base knowledge of how sniffing and spoofing is conducted, I will like to do a deep dive in this subject to find out more about how to protect the target machines from attacks that occur in a simple manner.