

Observational Report LAB 4: TCP/IP attack lab

ACS 545| Cryptography and Network Security

February 7, 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: -

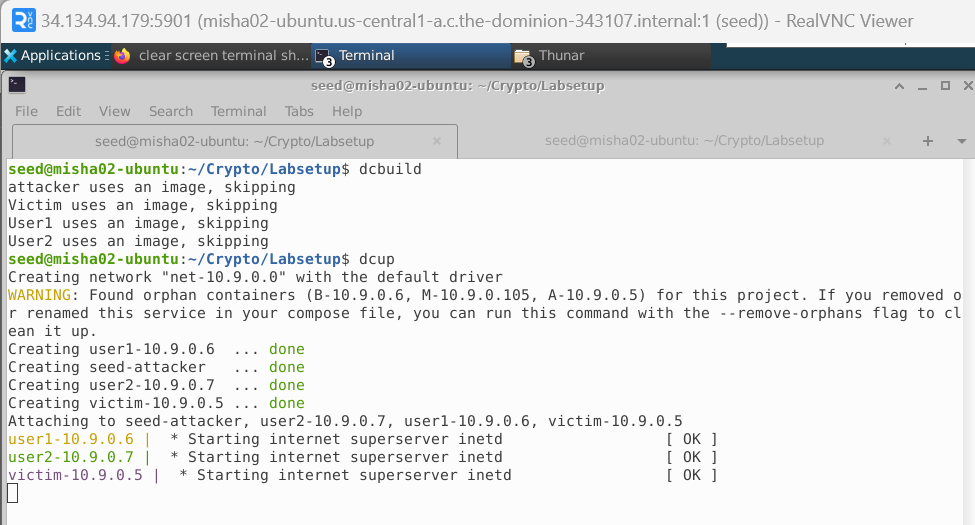
1. seed - attacker
2. user1 (10.9.0.6)
3. user2 (10.9.0.7)
4. victim (10.9.0.5)

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.



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

* Task 1 (SYN Flooding Attack): -
  + Task 1 A (Launching the Attack using Python): -

For the first task, I have referenced the code shared in class. In this script there is a very small change as per the asked question. The requirement is to send out TCP SYN packets, with randomly generated source IP address, source port and sequence number. So, I have added the TODO fields.

A screenshot of a computer

Description automatically generated

Logging into victim and attacker machines

(Victim)

A screenshot of a computer

Description automatically generated

(Attacker)

A screenshot of a computer

Description automatically generated

Launching Attack

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

As we can see from below screenshot that initially when tcp\_max\_syn\_backlog was kept at default, the attack was not successful.

A screenshot of a computer

Description automatically generated

However, when the value was set to 64, the attack was successful.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

* + Task 1 B (Launch the Attack Using C): -

For part B task, I have reused the same C code provided in class. Below is a screenshot of the code file. The task is the same as that mentioned in Task 1 A.

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

Running the code: -

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

As we can see that the connection got disconnected, the attack was successful.

* + Task 1 C (Enable the SYN Cookie Counter Measure): -

For part C task, we needed to enable sync cookies and run both the C and Python code to see if the attacks were successful. As the sync cookies were turned on, the attack never killed the user’s Telnet session hence resulting in a failure.

Tweaking the Sync Flag

A screenshot of a computer

Description automatically generated

Running the attack with C code and monitoring outputs

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Using the Python Script

A screenshot of a computer

Description automatically generated

* Task 2 (TCP RST Attacks on telnet Connections): -

For the second task, I have referenced the code shared in class and modified as shown below.

Code Snippet

A screenshot of a computer

Description automatically generated

Wireshark screenshot for output

A screenshot of a computer

Description automatically generated

Output of Session being killed due to code execution

A screenshot of a computer

Description automatically generated

* Task 3 (TCP Session Hijacking): -

For this task the code snippet is as below:-

A screenshot of a computer

Description automatically generated

Starting nc and launching attack

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

As we can see the message in nc, the attack was successful.

* Task 4 (Creating Reverse Shell using TCP Session Hijacking): -

This task is like Task 3. The only change is in the data field.

Code Snippet: -

A screenshot of a computer

Description automatically generated

Wireshark details to fill the values.

A screenshot of a computer

Description automatically generated

Starting nc

A screenshot of a computer

Description automatically generated

Running the script in Attacker Machine

A screenshot of a computer

Description automatically generated

Attack was successful and Telnet Session had crashed.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

* 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 TCP/IP Attacks work, 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.