Lab: breaking telnet using a synflood.

Setup

An ubuntu 20.04 LTS VM is needed for this lab, on the VM download the DOSLab.zip file from https://richardlu.ca/DOSLab.zip

Step 1 : In order to unzip the file, we first need to install the unzip program using the following command. After that, unzip the file. You may need to install the docker-compose library as well(**apt install docker-compose**).

```
sudo apt update
sudo apt -y install unzip
unzip DOSLab.zip
cd DOSLab
```

step 2: After unzipping the file, you will see a src-cloud folder. Enter this folder, and run the following command to install software and configure the system(around 8-10 mins).

```
chmod 777 install.sh
./install.sh
```

Note: This shell script will download and install all the software needed for the SEED labs. The whole process will take a few minutes. Please don't leave, because you will be asked twice to make choices:

- During the installation of Wireshark, you will be asked whether non-superuser should be able to capture packets. Select No.
- During the installation of xfce4, you will be asked to choose a default display manager. Choose LightDM.

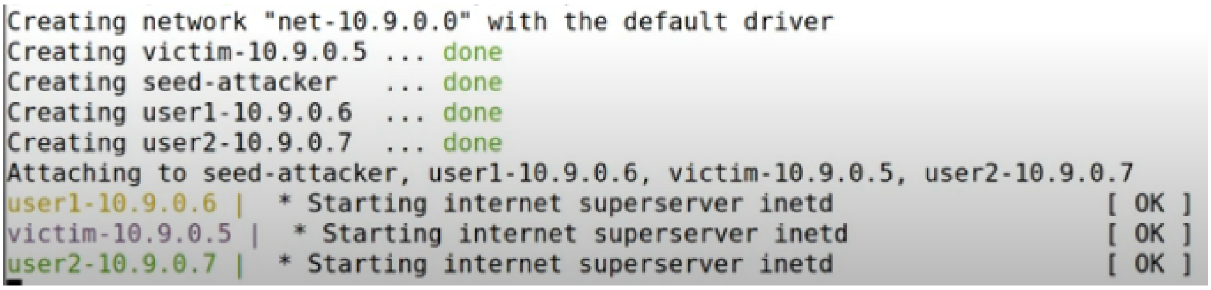
After running this command all files should be installed for the lab, in addition to that another account was created called seed which will be used to login to the containers using telnet(password: dees)

Now you can cd .. and begin setting up the docker containers as administrator (**sudo -s**):

To build the container run the command :

docker-compose -f docker-compose.yml build

To run the containers: Run the command : **docker-compose up** which will give you something like this to signify the containers are running:



```
Creating network "net-10.9.0.0" with the default driver
Creating victim-10.9.0.5 ... done
Creating seed-attacker   ... done
Creating user1-10.9.0.6  ... done
Creating user2-10.9.0.7  ... done
Attaching to seed-attacker, user1-10.9.0.6, victim-10.9.0.5, user2-10.9.0.7
user1-10.9.0.6 |  * Starting internet superserver inetd                    [ OK ]
victim-10.9.0.5 |  * Starting internet superserver inetd                   [ OK ]
user2-10.9.0.7 |  * Starting internet superserver inetd                    [ OK ]
```

Note: for our lab we are only making 3 conatiners as user2 would be useless for our lab.

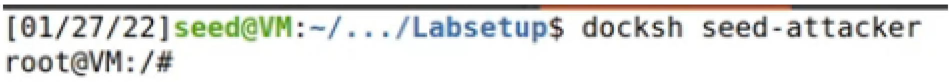
Create the aliases **dockps** and **docksh**, keep this saved somewhere as you will need it any time you open a new terminal to connect to a container:

alias dockps='docker ps --format "{{.ID}} {{.Names}}"'

docksh() { docker exec -it $1 /bin/bash; }

dockps can we used to display the container that are running and can help you with **docksh**.

We can access the other containers created by using the **docksh** command for example:



```
[01/27/22]seed@VM:~/.../Labsetup$ docksh seed-attacker
root@VM:/#
```

Now everything should be set up and ready to be used for the lab.

<u>Note A</u>: attack could fail due to:

- TCP cache issue
  - A kernel mitigation mechanism. On Ubuntu 20.04, if machine X has never made a TCP connection to the victim machine, when the SYN flooding attack is launched, machine X will not be able to telnet into the victim machine. However, if before

the attack, machine X has already made a telnet (or TCP connection) to the victim machine, then X seems to be "immune" to the SYN flooding attack, and can successfully telnet to the victim machine during the attack. It seems that the victim machine remembers past successful connections, and uses this memory when establishing future connections with the "returning" client.

    - Use the command: **ip tcp_metrics flush** to remove any known connections
- vm issues
    - Try restarting VM
- TCP transmission issue
    - Evey time an item is removed a slot opens up which might lead to issues with competing with telnet for that slot, try to Run multiple instance of the program by adding a **"&**" at the end of the call.
    - You can look at the instance by tying the command: **jobs**
    - You can terminate your instances by doing **kill %<process number>**
- Size of the queue
    - How many half-open connections can be stored in the queue can affect the success rate of the attack
    - The size of the queue be adjusted using the following command:
        - **sysctl -w net.ipv4.tcp_max_syn_backlog=80**

Task 1: DOS Technical Questions

1. In your own words briefly describe what a DOS attack is and how it affect a user.
2. What is a DDoS?
3. What are some types of DOS attacks? List 2 of them.
4. What are 2 ways you can mitigate a DoS attack

Task 2: SYN floods and TCP in Linux

1. LIst the 3 ways to mitigate against a SYN flood attack.
2. Briefly descibe each stage of the TCP 3 way handshake.
3. What does the value returned by **sysctl net.ipv4.tcp_synack_retries** mean?
4. Let's learn how to establish a TCP connection between two containers(username: seed, pass: dees): Launch two terminals where one is connected to the victim and the other to user1. In the user1 terminal run **netstat -nat** which will show the active internet connection the user has. Next telnet to the victim using the command: **telnet <IP>** and log into victim. Finally go back to user1(might have to open a new terminal) and run **netstat -nat** to see the established connection. Take a screenshot of the result and save it for submission. run ip tcp_metrics flush to reset the known connections.

Task 3: launching a SYNFLOOD attack

First fill out the synflood.py file in the lab folder so that it corresponds to the victim. As well as setting the port to telnet. For the iface value use the command ifconfig, think about where the packets are going to be sent from.

Next have a victim and attacker terminal open using **docksh**.
Let's adjust the size of the victim's backlog queue in the victim side by using the following command:

```
sysctl -w net.ipv4.tcp_max_syn_backlog=40
```

Note that ¼ of the space is saved for "proven destinations" so when setting it to 40 we actually have a capacity of around 30.

Next let's run the python file using the command from our root terminal(our file speicifes that the attacker container sends packets to the victim with the iface value we changed so we dont need to be in those containers):

```
python3 synflood.py
```

While the attack is running run one of the following commands in the victim's container a few times to see how many items are on the queue:

```
netstat -tna | grep SYN_RECV | wc -l
 ss -n state syn-recv sport = :23 | wc -l
```

Save a screenshot of this and submit it along with the python file..

Let the attack run for at least one minute, then try to telnet into the victim machine, and see whether you can succeed. Very likely that your attack will fail. Multiple issues can contribute to the failure of the attack. Check **note A**.

Lastly submit a screenshot of your attacking working, i.e telnet connection timed out.