Network Security

Assignment: Transport Layer

SYN Flood [20 points]

In the topology in the image below, the webserver is vulnerable to the SYN-flooding attack.

Exploit this vulnerability to make the web server running on port 80 unavailable. Before starting with the assignment, unzip the provided file and navigate to the unzipped directory. Generate the environment using the following Docker commands.

```
docker compose build
docker compose up -d
```

To connect to the attacker container, execute the command below.

```
docker exec -ti assignment_4_syn_flood_attacker bash
```

There's a challenge: the webserver has a firewall that allows only 1 SYN packet per second from the same source. You can check whether the webserver is accessible using the following command:

```
docker exec assignment_4_syn_flood_client curl http://192.168.124.20/
```

After finishing the assignment, you can destroy the environment using the command below. Make sure your script is inside the mounted `/solution` folder, so it is saved when you destroy the environment.

```
docker compose down
```

Goal

Produce a Python3 script that accepts two arguments: the destination address and the destination port. The script should be invoked as follows:

```
python3 syn_flood.py 192.168.124.20 80
```

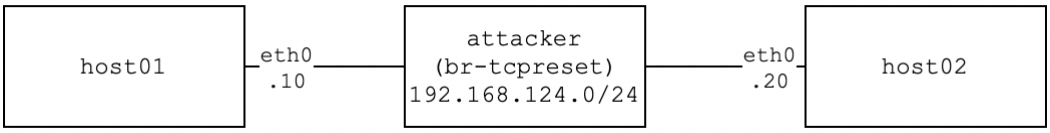
The script should continuously run until it is terminated. It doesn't have to generate any output. The assignment is successful if the client can no longer connect to the web server.

Submission Instructions

Your script should run on the *attacker* container. You may not use external libraries or tools different from those installed on the provided *attacker* container.

TCP Reset [40 points]

Consider the topology in the image below. The attacker has already established a *Man-in-the-middle* (MITM) presence between host 1 and host 2. This allows the attacker to see and manipulate the traffic between the two hosts. You should exploit this position to conduct a TCP-reset attack.



First, unzip the provided file and navigate to the unzipped directory. Afterward, execute the commands below to generate the topology.

```
docker compose build
docker compose up -d
```

To connect to the attacker container, execute the command below.

```
docker exec -ti assignment_4_tcp_reset_attacker bash
```

The two hosts exchange data with each other via a TCP connection. Host 2 connects to host 1 on port 1337. You can view the messages the two hosts exchange in real time via the following command:

```
docker logs assignment_4_tcp_reset_host2 --follow
```

After finishing the assignment, you can destroy the environment using the command below. Make sure your script is inside the mounted *solution* folder so it is saved when you destroy the environment.

```
docker compose down
```

Goal

Create a Python3 script which accepts three arguments. These arguments should be the source IP address, destination IP address, and destination port of the TCP connection. The way the script should be invoked is shown below.

```
python3 tcp_reset.py 192.168.124.20 192.168.124.10 1337
```

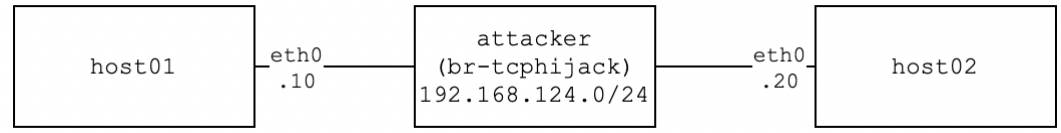
The assignment is successful if the TCP connection breaks when executing the TCP-reset attack.

Submission Instructions

Your script should run on the attacker container. You may not use external libraries or tools different from those installed on the provided *attacker* container.

TCP Hijacking [40 points]

Consider the topology in the image below. Once again, the attacker has a man-in-the-middle position and can observe the exchanged packets between the two hosts.



First, unzip the provided archive. Afterward, navigate to the unzipped directory. Next, execute the commands below to set up the environment.

```
docker compose build
docker compose up -d
```

You can connect to the attacker with the command below.

```
docker exec -ti assignment_4_tcp_hijack_attacker bash
```

Host 1 has a listener running on port 1337, which allows (only) host 2 to connect to it and execute commands. Host 2 connects and automatically chooses commands to execute. You can view the commands in real time as follows:

```
docker logs assignment_4_tcp_hijack_host2 --follow
```

After finishing the assignment, you can destroy the environment using the command below. Make sure your script is inside the mounted *solution* folder so it is saved when you destroy the environment.

```
docker compose down
```

Goal

Write a Python3 script that accepts three arguments: the source IP address, and destination IP address, and the destination port of the TCP connection you're hijacking. The way the script should be called is shown below.

```
$ python3 tcp_hijack.py 192.168.124.20 192.168.124.10 1337

waiting for packet...
Received reverse shell connection from ('192.168.124.10', 35998)
sh: 0: can't access tty; job control turned off
$ ls -l /home/user
$ total 4
drwxr-xr-x 2 user user 4096 May 10 11:59 pwned
```

As proof of a successful hijack your script should **create a directory** called `pwned` inside the `/home/user` directory of host 1.

Furthermore, you should also **set up a reverse shell** back to the attacker container. Your script should implement a reverse shell listener and allow the user to enter commands that are interactively executed on host 1 and see their output. Consider that the host's firewall doesn't allow inbound TCP connections from the attacker to be established, i.e., a bind shell won't work.

Submission Instructions

Your script should run on the attacker container. You may not use external libraries or tools different from those installed on the provided *attacker* container.