Luke Evans

February 14, 2021

Lab 3 – TCP/IP Attacks

**Attacker IP = 10.0.2.15**

**Victim IP = 10.0.2.4**

**Server = 10.0.2.5**

* 1. *SYN Flooding Attack*

We check the size of the half open connection queue on our victim machine in the screenshot below:

Text

Description automatically generated

We can check the half open connections using the command nestat -at, shown in the screenshot below:

A picture containing table

Description automatically generated

Next we will launch the attack from our attacking VM:

Text

Description automatically generated with low confidence

We can again run the netstat -at command to view the states of the connections while the attack is being launched:

Text

Description automatically generated

TCP syn cookies was enabled on our victim machine by default so for example purposes we first turn it off.

Graphical user interface, text

Description automatically generated with medium confidence

We then try to telnet to the machine from our server computer and are unable to connect because the half open connection queue is full and no other connection to port 23 can be made:

Graphical user interface, text

Description automatically generated

Next, we turn tcp syn cookies back on on our victim machine:

Text

Description automatically generated with medium confidence

Then we try to telnet to the victim machine again from the server and we are able to make a connection:

Text

Description automatically generated

Observations: The screenshots above show an example of a SYN flood attack. We used the netwox tool to send many TCP SYN packets from spoofed source IP addresses which causes the victim machines half open connection queue to fill up. We target port 23 specifically so when we try to telnet to the victim machine from our server machine it does not work because the victim machine is no longer able to accept any other connections and we disabled the tcp syn cookie counter measure. When we reenable the tcp syn cookie counter measure, we can then make a telnet connection from our server VM to our victim VM even though our attacker is flooding the victim with TCP SYN packets.

Explanation: The TCP SYN flooding attack targets the bottle neck in the three way handshake. By filling the half open connection queue for a specific port, the victim machine will no longer be able to accept any other connections to that port number even though the machine has plenty of computing resources available. When the TCP SYN cookie counter measure is enabled our attack is unsuccessful because once the half open connection queue fills up, the victim machines OS will automatically start attaching tcp session cookies to the syn ack request and will no longer buffer the half open connections. Then when the victim machine receives an ACK packet it can check the SYN cookie to verify if it is a valid connection request before establishing a TCP session.

*2.1 TCP Reset Attacks on Telnet and SSH*

The server VM telnets to our victim VM:

Text

Description automatically generated

On our attacking machine we launch our attack using the netwox tool:

A picture containing text

Description automatically generated

We then try to type something from our server that had previously opened a telnet connection to our victim machine and we can see that the connection as been closed:

Text, letter

Description automatically generated

Next we will repeat the TCP RST attack on an open SSH connection. First we establish an SSH session between the server VM and the victim VM:

Text, letter

Description automatically generated

Instead of using netwox for our attack, this time we will use pythons scapy library. Our code is shown below:

Text

Description automatically generated

We then launch the attack from our attacking VM:

Graphical user interface, text

Description automatically generated

Next, we go back to our ssh terminal on the server machine and try to type a command:

Text

Description automatically generated

Observation: In this task, we launched a TCP RST attack from our attacking VM targeting a telnet session and ssh session between our server VM and victim VM. The first attack using the netwox utility resets all TCP connections on the LAN. We can see this by the fact that we did not specify any parameters. The netwox tool simply sniffs the TCP packets and forms and send out the RST packets. This is why our telnet connection was interrupted. In our second attack using scapy, we actually specify the IP that we want to target. In our case, it is IP address 10.0.2.4. The python sniffs packets from source 10.0.2.4 and then sends our RST packets when one is found which interrupts the tcp session. This is seen when the pipe is broken in our SSH connection between our server machine and the victim machine.

Explanation: The TCP reset attack’s objective is to break an existing connection between two victim hosts. The way it is done is by spoofing a TCP reset packet to cause one of the victim hosts to think the other victim host is closing the connection. This is done by sending a packet with the RST bit set to 1. When a properly formed RST packet is received, it causes the connection to be immediately broken. Under normal circumstances this is used in emergency situations or when errors are detected; however in the TCP reset attack the protocol rules are targeted as an attack surface.

The important areas of the reset packets are the TCP packet signature which includes source IP address, source port number, destination IP address, and destination port number. These four fields in the spoofed packet need to be the same as the fields used in the open connection otherwise the packet will be discarded. In addition, the sequence number used in the spoofed packet needs to be correct or the receiver will discard the packet. Theoretically, for the sequence number to be correct it just has to be in the receivers window in Linux it seems that it has to be the next sequence number or it will be discarded by the receiver.

*3.1 TCP RST Attacks on Video Streaming Applications*