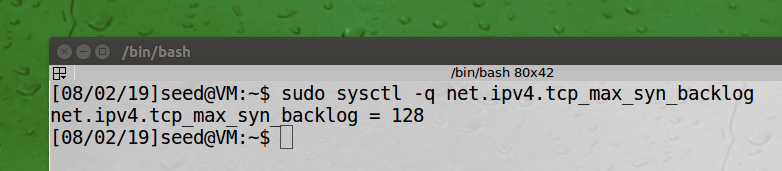
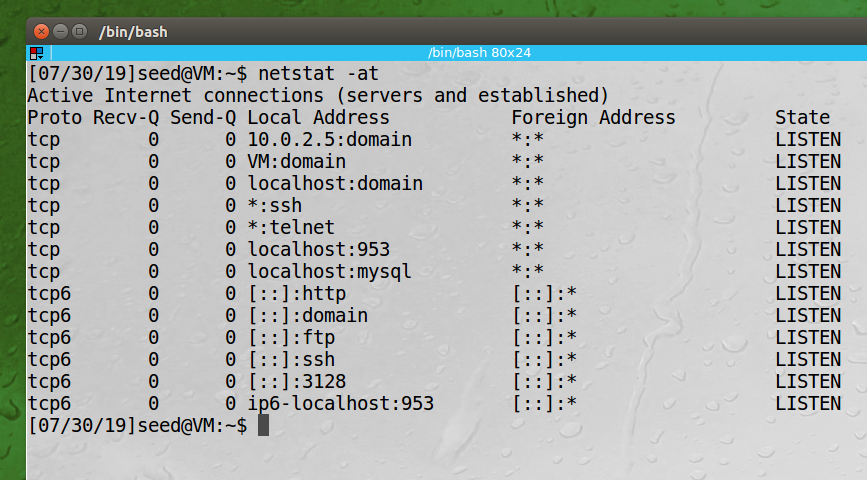
Internet Security - Lab 3 Report

# SYN Flooding Attack

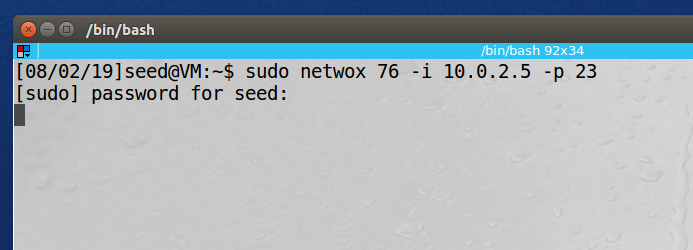
Showing Machine B (victim) the maximum size for the queue for half open connections, this machine allows up to 128 connections.



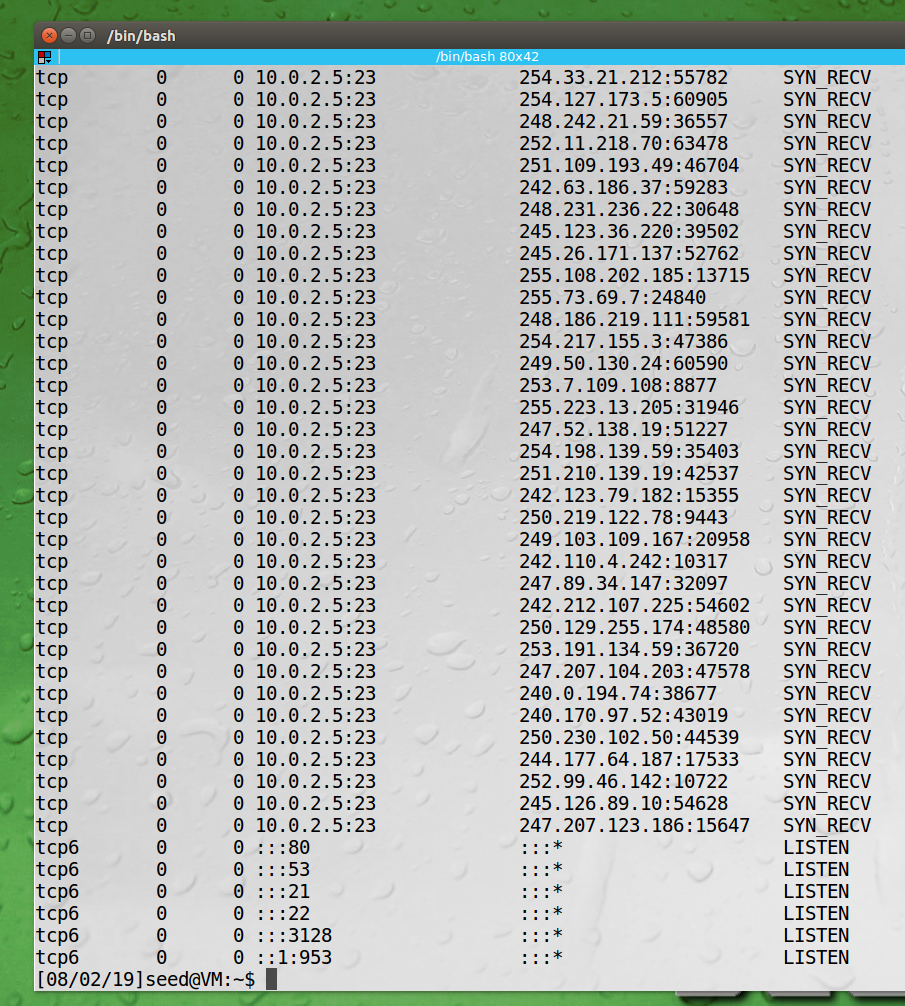
Showing the Open TCP connection on Machine B, there are no half open connections prior to the attack.



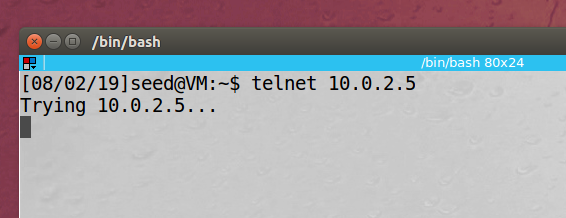
Launching the attack from Machine A:



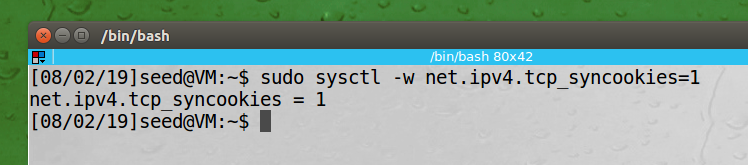
Showing the number of TCP connections on Machine B while the attack is in progress:



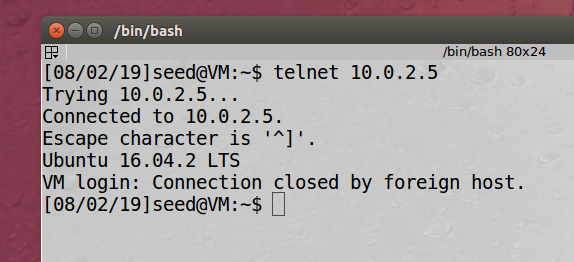
Trying to TELNET from Machine C to Machine B during the attack, unable to connect:



Turing on the SYN Cookie mechanism on Machine B (victim):



After Launching the attack again with the SYN Cookie mechanism on, Machine C can now TELNET to Machine B while being attacked:

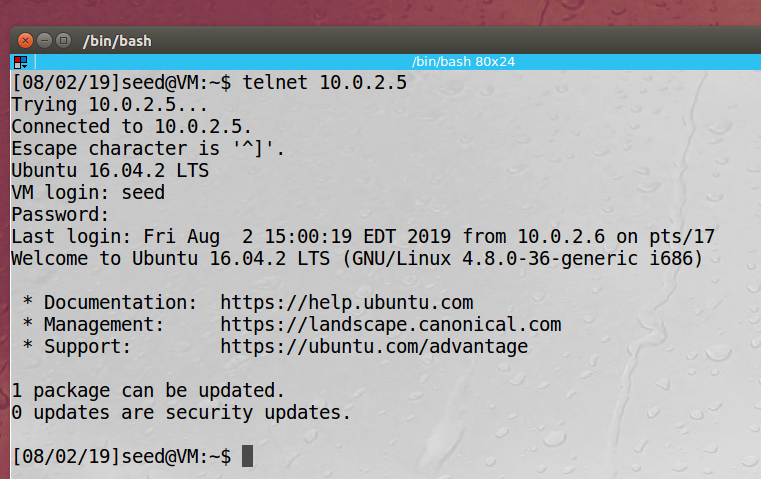


**Observation:** The above screenshots show an example of a TCP SYN flooding attack. We launch the attack from Machine A using the NETWOX tool which generated dozens of TCP connection SYN packets originating from various IP addresses which quickly filled up Machine B’s queue. Then when we try to TELNET in to Machine B from Machine C, we don’t get a response. After the first attack we enable a TCP SYN cookie flag on Machine B and reattempt the attack. This time it is unsuccessful, and we can log via TELENT from Machine C.

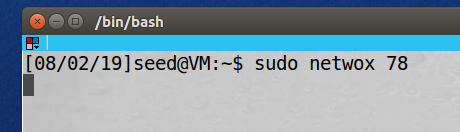
**Explanation:** TCP SYN flooding is an attack on the three-way handshake that TCP uses to establish connections. By sending many initial connection requests to the server, and not following up with fulling opening the connection we can flood and fill the connect queue and the server will not be able to accept any new TCP connection requests.

# TCP RST Attacks on TELNET and SSH Connections

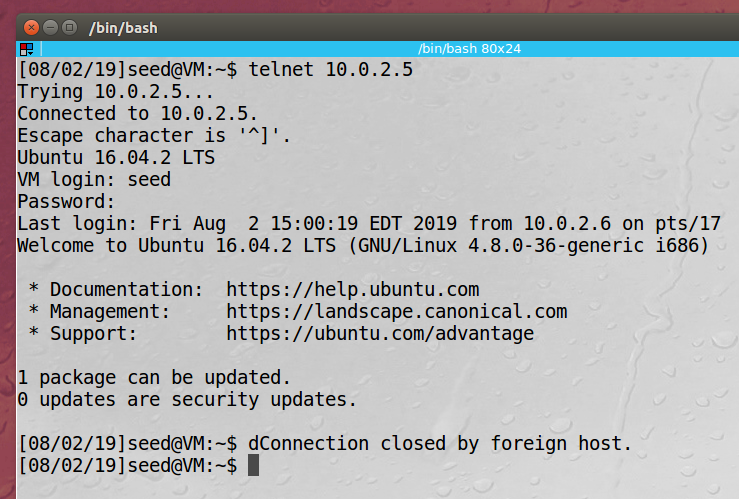
Machine C connects via TELNET into Machine B:



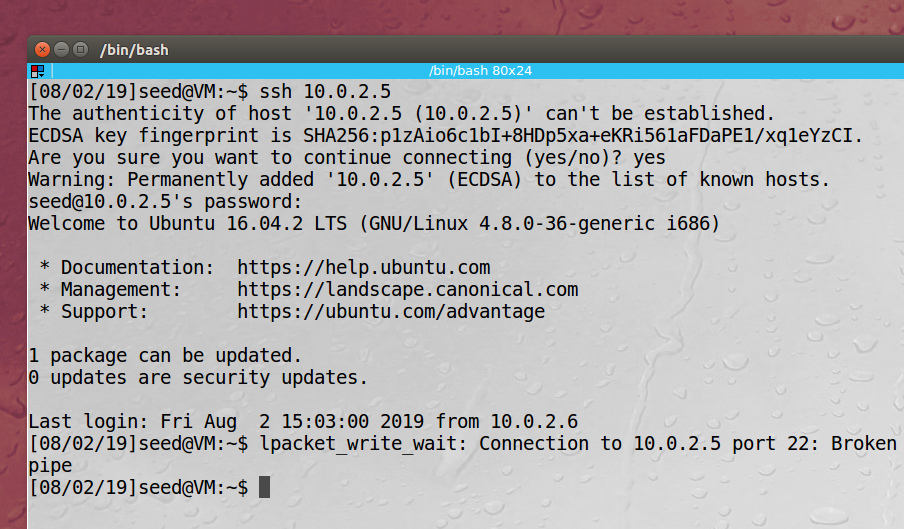
On Machine A we launch the attack using NETWOX 78:



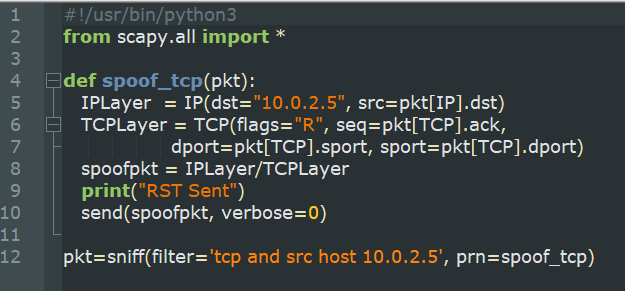
On Machine C, we type any character (in this case “d”) and the connection is dropped:



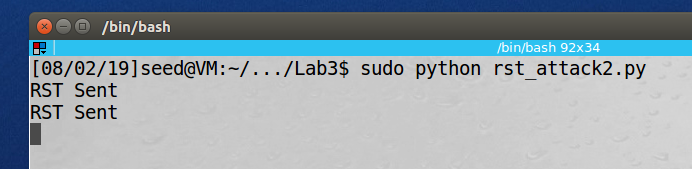
Attempting the attack again using SSH, the connection is dropped again with the “broken pipe” message:



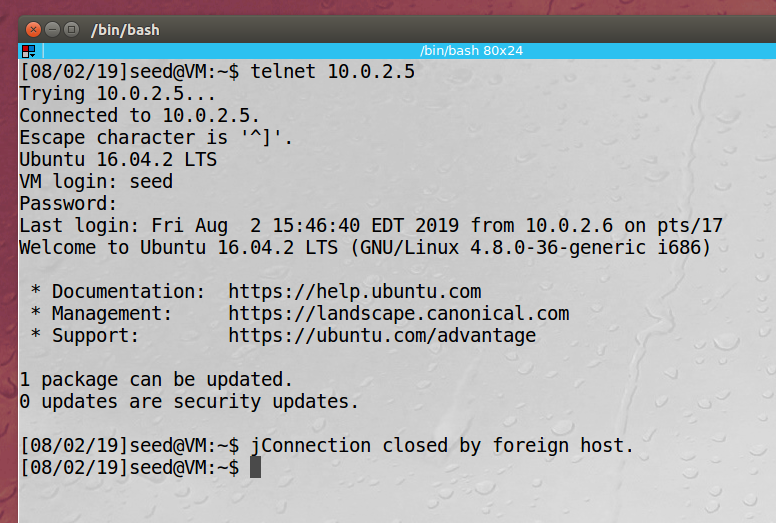
Python Code for TCP RST attack:



Running the Attack from Machine A:



Machine C is disconnected from Machine B TELNET connect when a character is input:

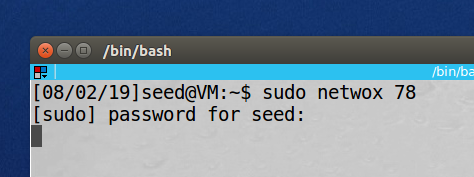


**Observation:** In this task we launched a TCP RST attack from Machine A. This attack drops all TCP connection on the LAN. We were able to observe this when we first TELNET from Machine C to Machine B and then launch the attack from Machine A. The connection was dropped. We then reattacked, but this time while there was an SSH connection between the machines. Again, the connection was dropped during the attack. We then repeated this same attack using a Python script instead of the NETWOX and observed the same behavior.

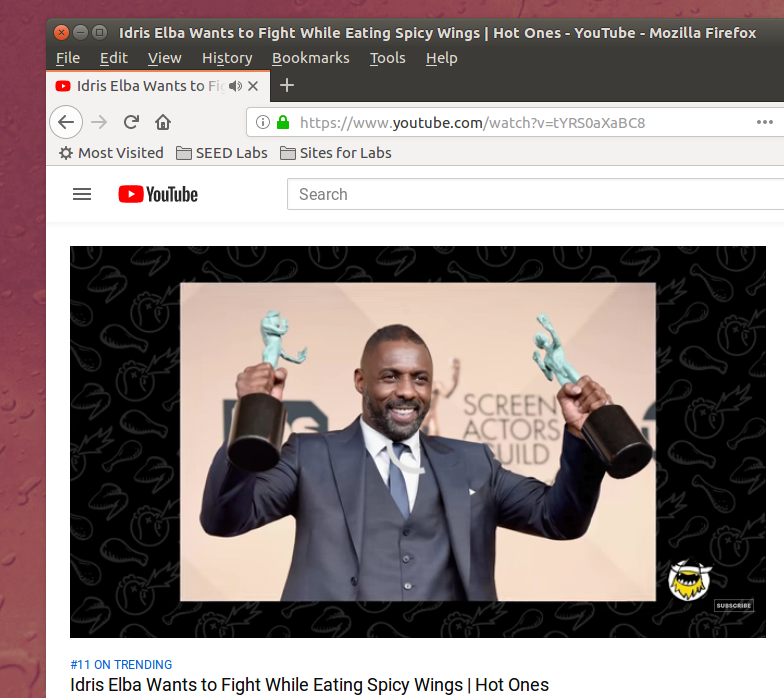
**Explanation:** TCP RST or reset attack spoofs either the server or the client to drop the TCP connection by sending a TCP packet with a special flag bit, called the RST bit. Once this packet is received the connection is dropped. We launched our attack using the NETWOX program and then a Python program and were able to drop both TELNET and SSH connections between machines on our LAN.

# TCP RST Attacks on Video Streaming Applications

Running the NETWOX 78 program on Machine A



Machine C that was streaming video from YouTube:



**Observation:** Similar to the previous task we launched a TCP RST attack from Machine A using NETWOX. This attack drops all TCP connection on the LAN. We were able to observe this when we were streaming from YouTube on Machine C and the stream ended/the connection was dropped.

**Explanation:** TCP RST or reset attack spoofs either the server or the client to drop the TCP connection by sending a TCP packet with a special flag bit, called the RST bit. Once this packet is received the connection is dropped. We launched our attack using the NETWOX program and then a Python program and were able to drop our YouTube stream.

# TCP Session Hijacking

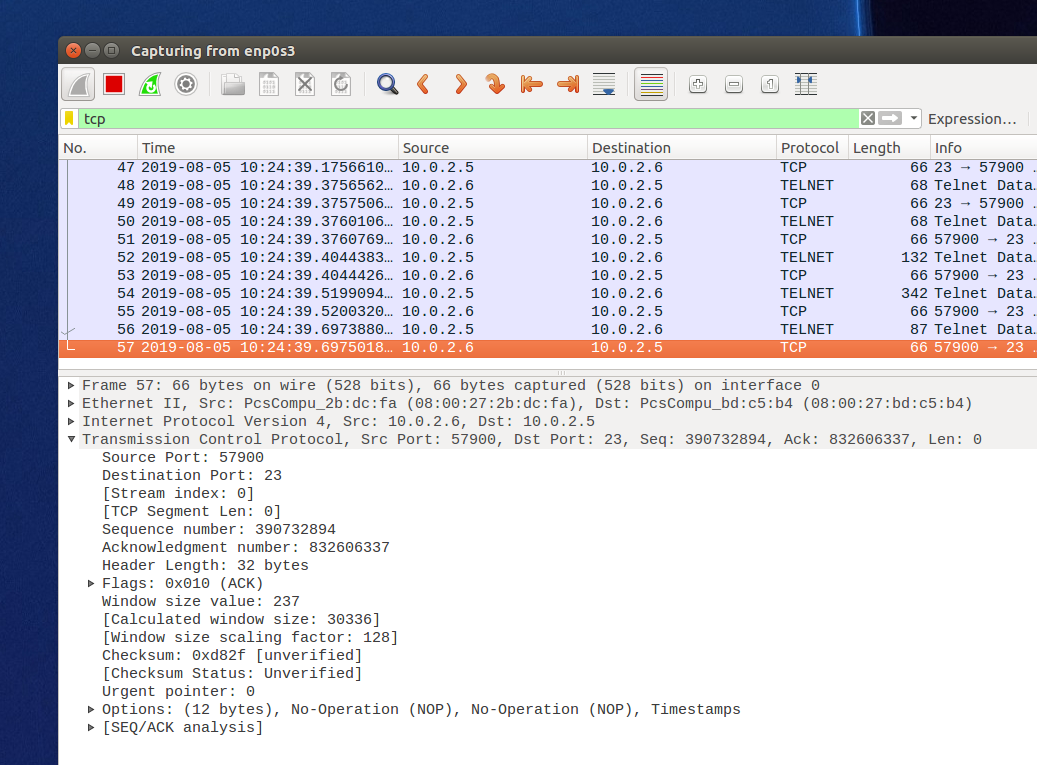
Attacker (10.0.2.4)

Client (10.0.2.6)

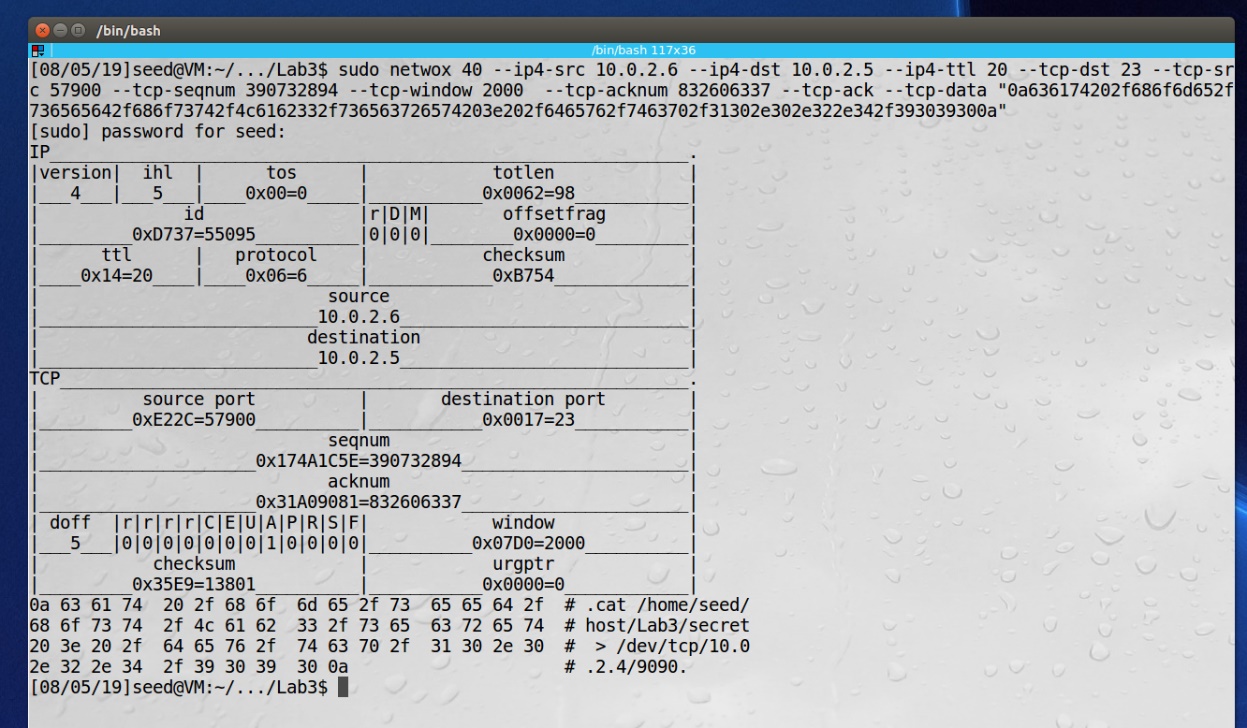
Server (10.0.2.5)

## 4.a Using Netwox

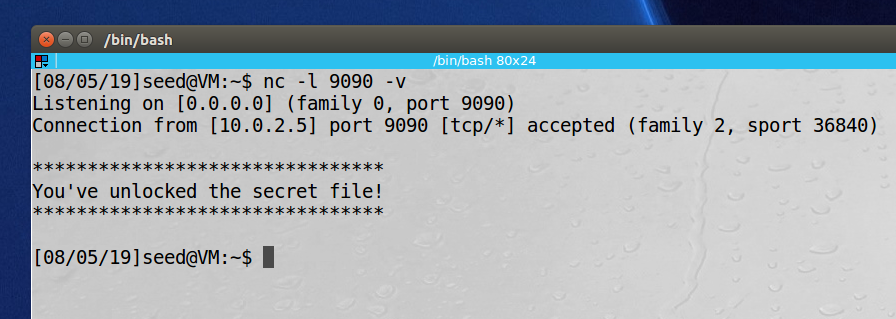
Attacker inspecting the last packet from the client to the server prior to launching attack:



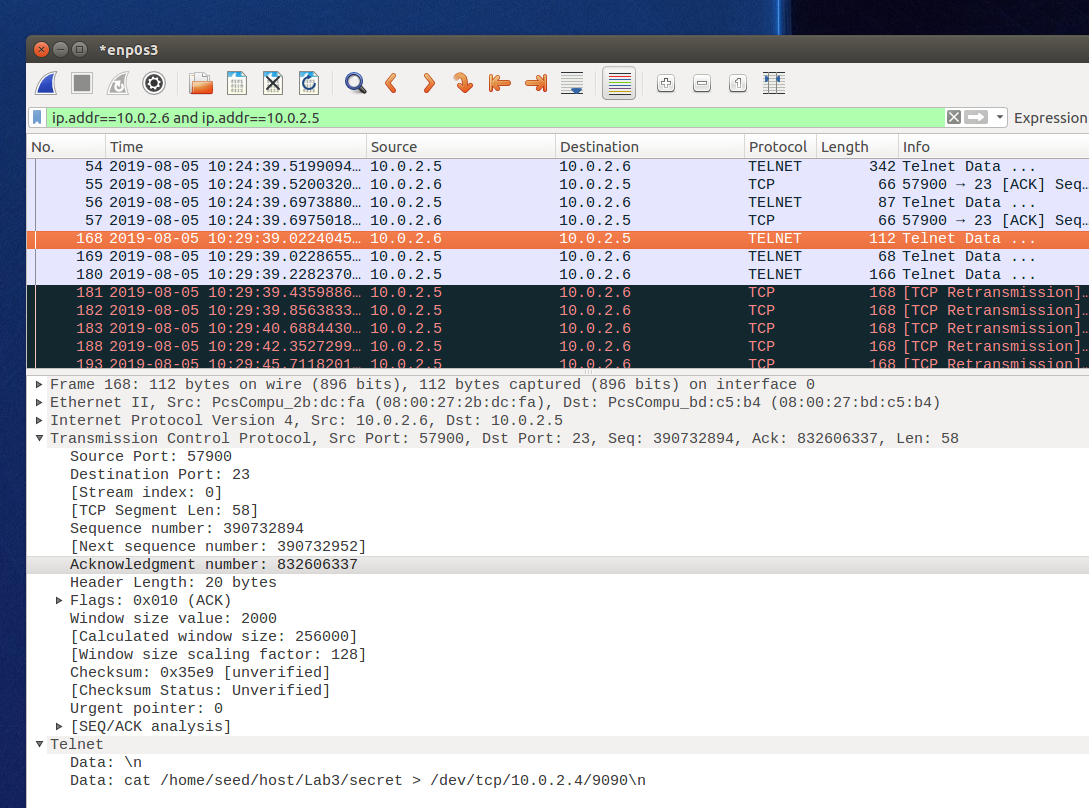
Attacker using Netwox to launch the attack, using the same values from the packet capture above:



Attacker receives the response from the spoofed packet:



Wireshark of the spoofed packet hijacking the TELENT connection between the client and server:

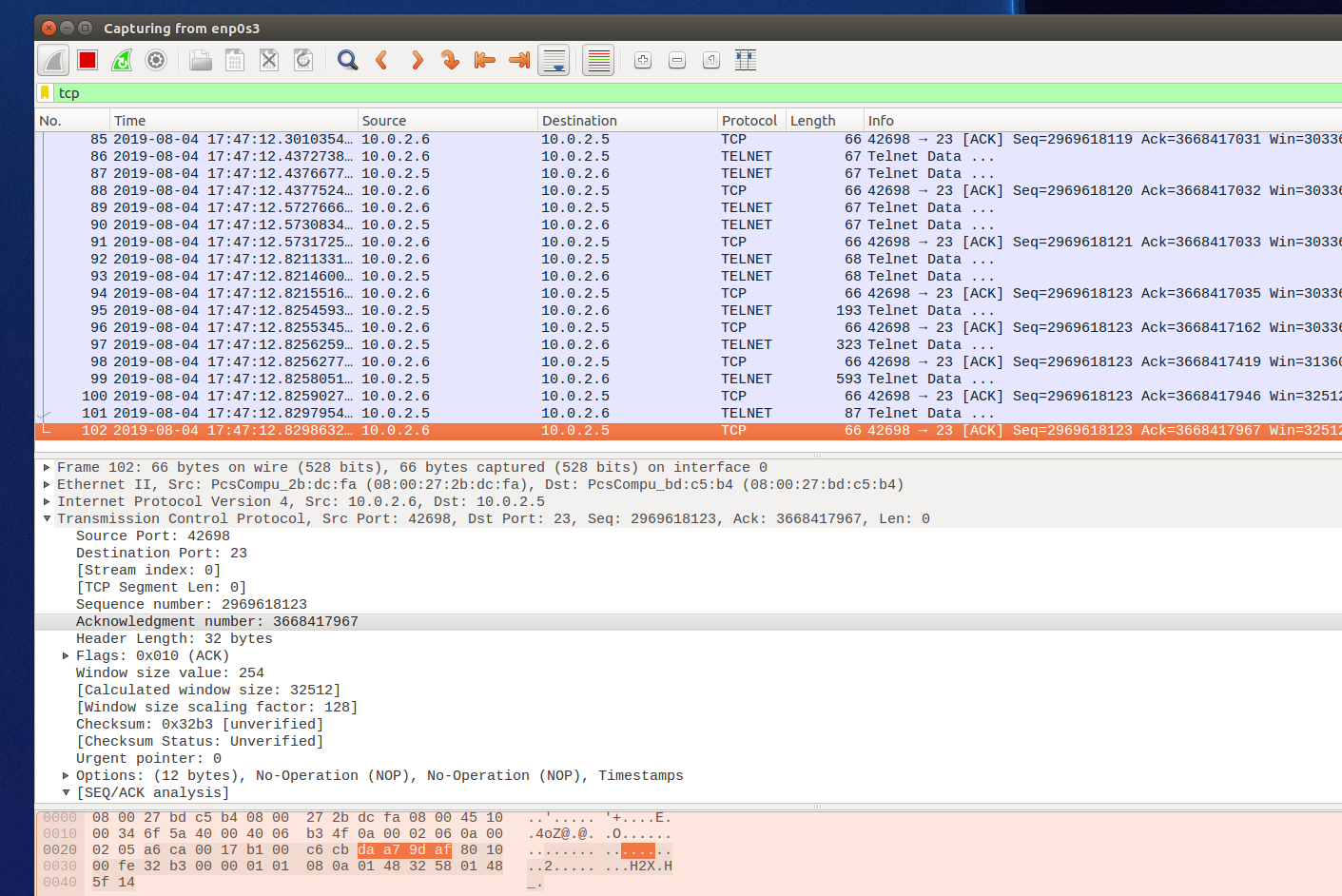


**Observation:** This attack used the Netwox 40 tool to hijack a TCP connection between the client and the server. The first thing need was for the attacker to sniff the connection to get some needed values to do the attack, including: source and destination IP addresses and port numbers, sequence number, and acknowledgement numbers. Then the attacker set up a TCP listening server so that the file that the attacker was trying to access could be read. Then the attack command was sent using the sequence and acknowledgement numbers from the previous capture. The command sent was to read a secret file and send the output to attacker’s machine. After the attack the terminal on the client was frozen due to the client and server being in a packet deadlock since the sequencing was broken.

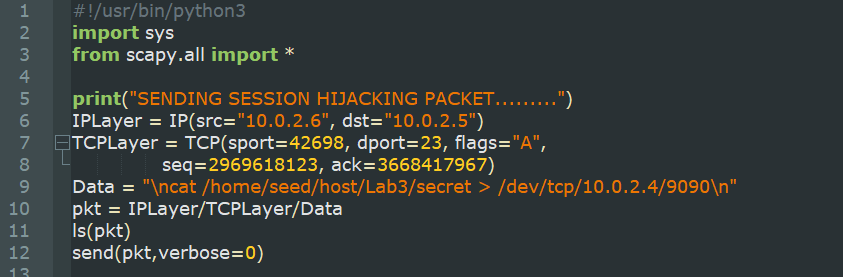
**Explanation:** Using the Netwox tool we can spoof a packet between two machines that are communicating via TELNET. We were able to get information from a server that only the client had access to, and send the output to the attacker’s machine.

## 4.b Using Scapy

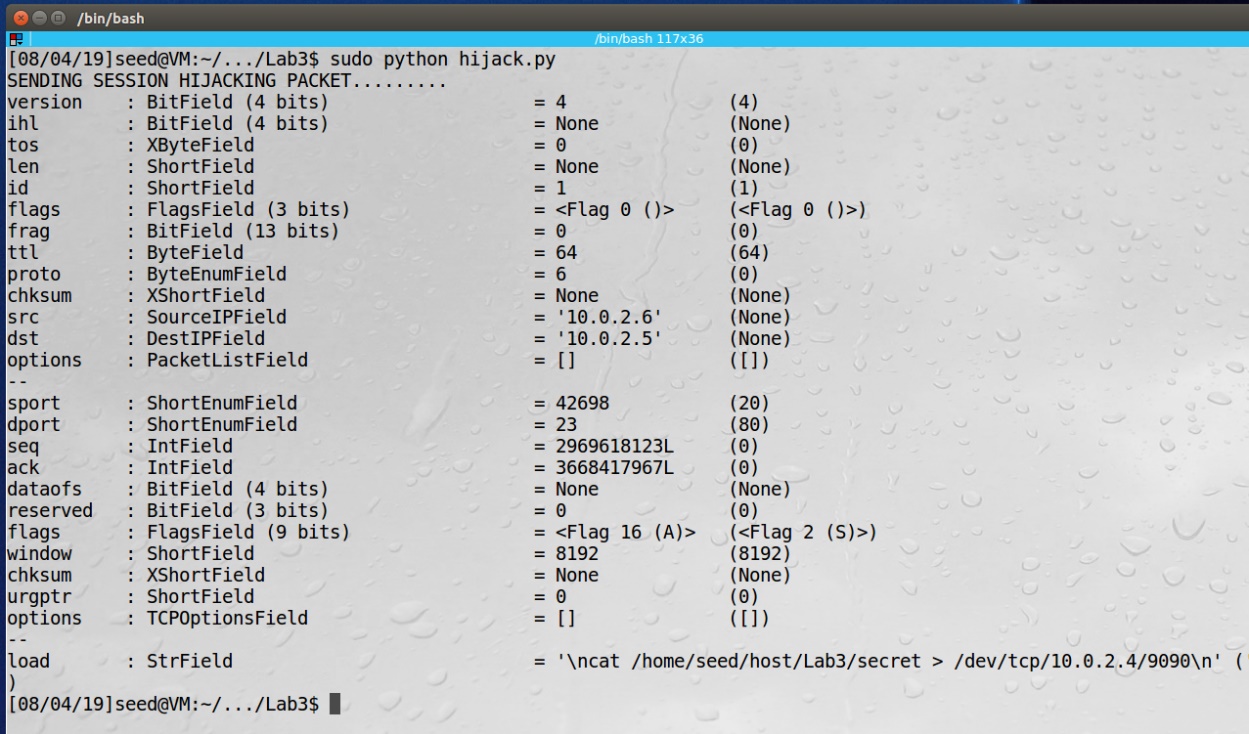
The attacker first sniffs the previous communications between the client and the server:



The attacker’s code for using Scapy to launch the TCP session hijack:



The attacker launching the attack using Scapy:



The attacker receiving the packet from the hijack session:

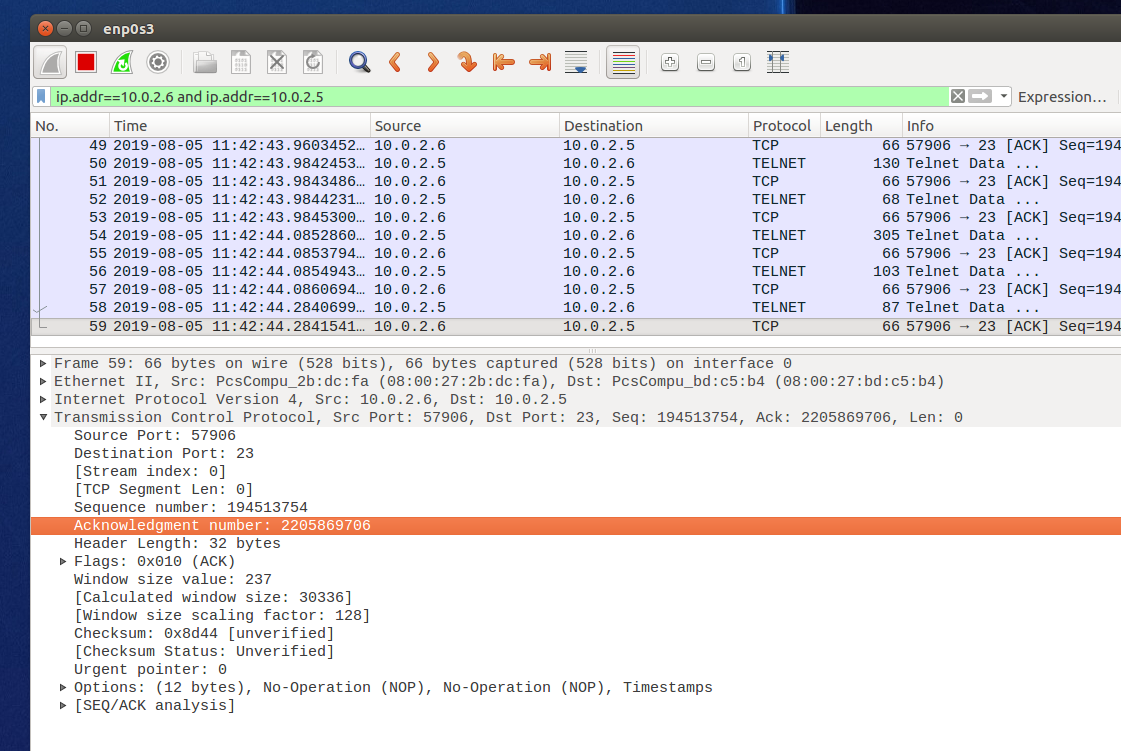


**Observation:** Similar to the previous task, we hijack a TCP connection between the client and the server but this time using Scapy. Again, the first thing need was for the attacker to sniff the connection to get some needed values to do the attack, including: source and destination IP addresses and port numbers, sequence number, and acknowledgement numbers. Then the attacker set up a TCP listening server so that the file that the attacker was trying to access could be read. Then the attack Python program was run, using the sequence and acknowledgement numbers from the previous capture. The command sent was to read a secret file and send the output to attacker’s machine. After the attack the terminal on the client was frozen due to the client and server being in a packet deadlock since the sequencing was broken.

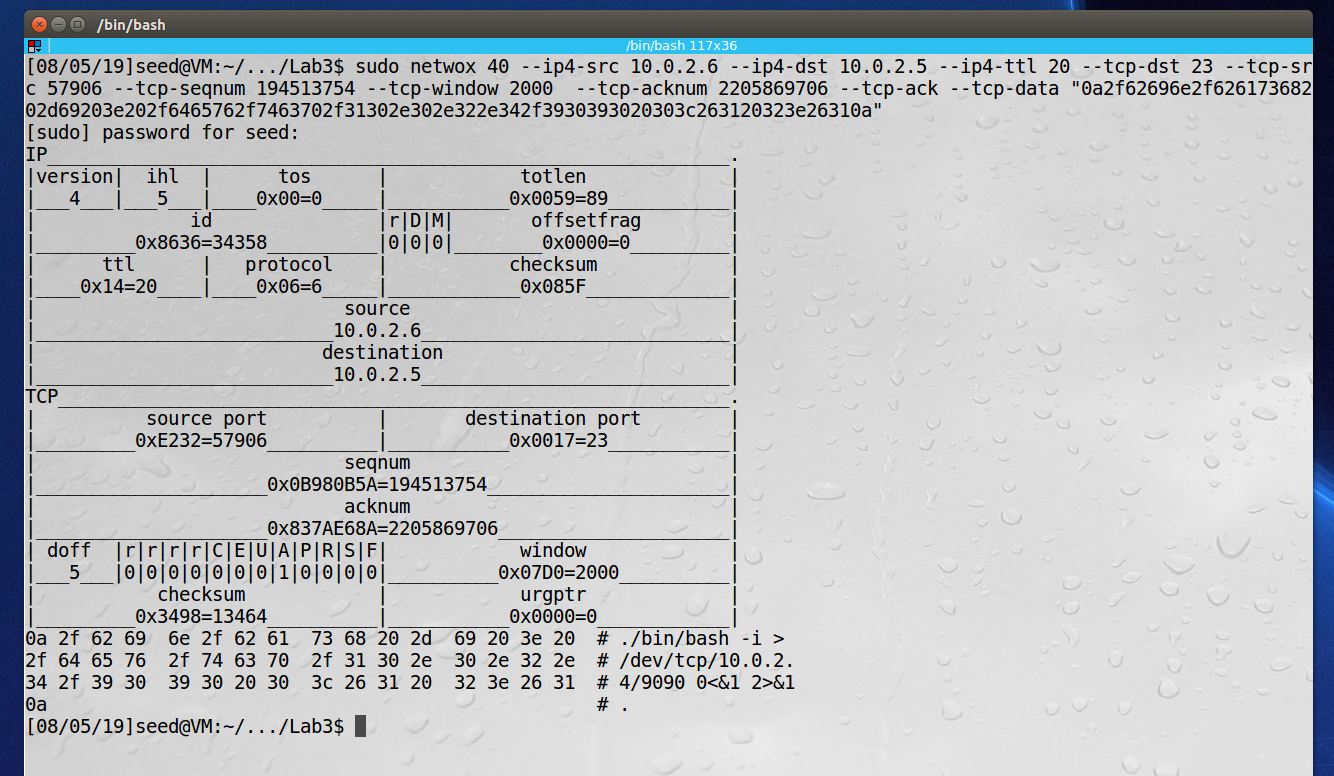
**Explanation:** Using the Scapy we can send a spoofed packet, hijacking the TELNET communication between two machines. We were able to get information from a server that only the client had access and send the output to the attacker’s machine.

# Creating Reverse Shell using TCP Session Hijacking

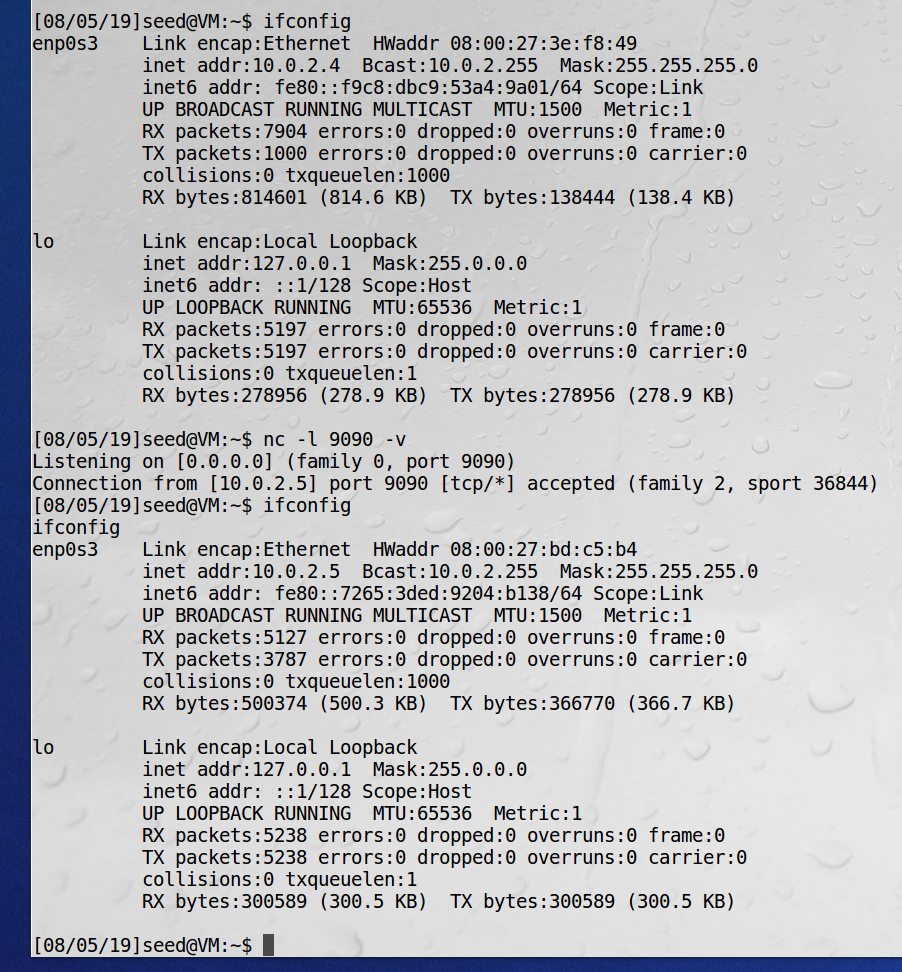
The attacker first sniffs the previous communications between the client and the server:



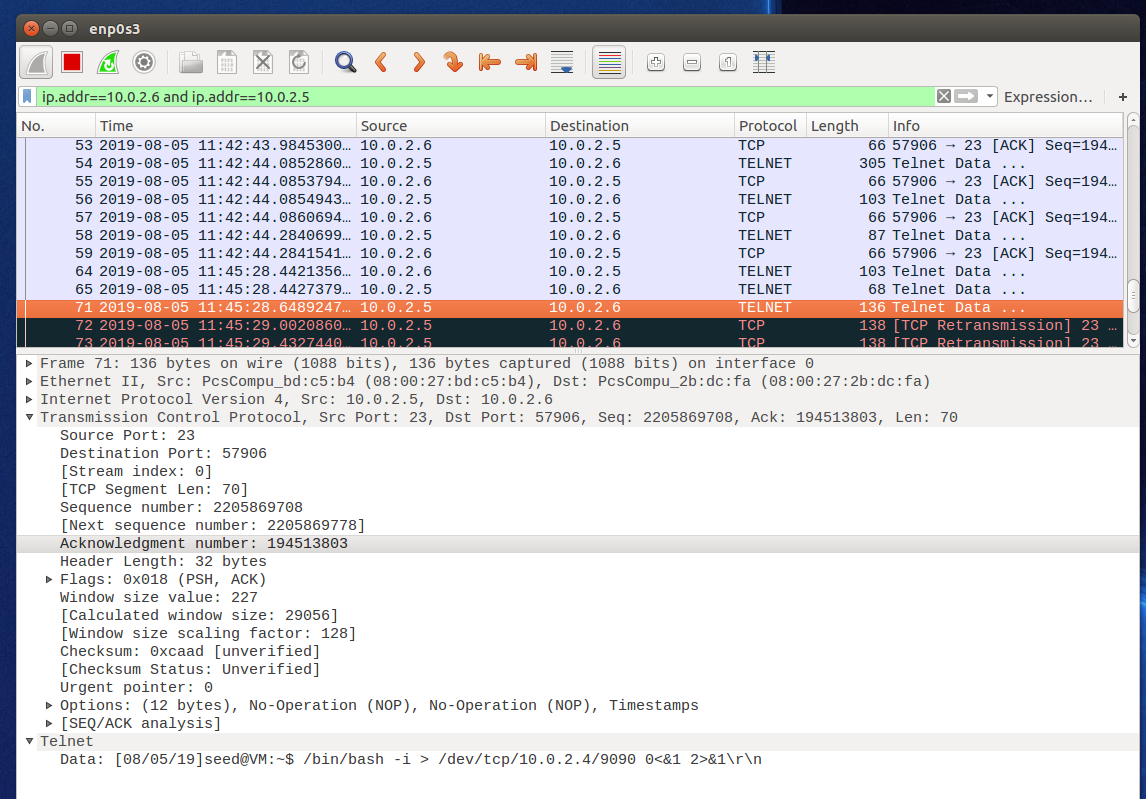
The attacker now has the needed values and launches the reverse shell attack:



The attack is successful and now the attacker has full shell control from the TELNET session between the client and the server:



The captured attack packet from Wireshark:



**Observation:** This attack was the same as the previous two with only the data in the spoofed packet being different. In the previous task we just wanted to read the contents of a file but this time we were able to send the command to redirect the shell input and output to the attacker’s computer. Again, the first thing need was for the attacker to sniff the connection to get some needed values to do the attack, including: source and destination IP addresses and port numbers, sequence number, and acknowledgement numbers. Then the attacker set up a TCP listening server so that the file that the attacker was trying to access could be read. Then the attack command was sent, using the sequence and acknowledgement numbers from the previous capture. The command sent was to redirect the shell’s input and output the attacker’s computer. The attack was successful, and you can see in the images that the attack gained full access to the shell.

**Explanation:** Using Netwox we can send a spoofed packet, hijacking the TELNET communication between two machines and creating a reverse shell. We were able to redirect the shell of the of the client-server connection to the attacker’s computer, essentially giving the attacker access to issue whatever commands they wanted.