Todd Wenker

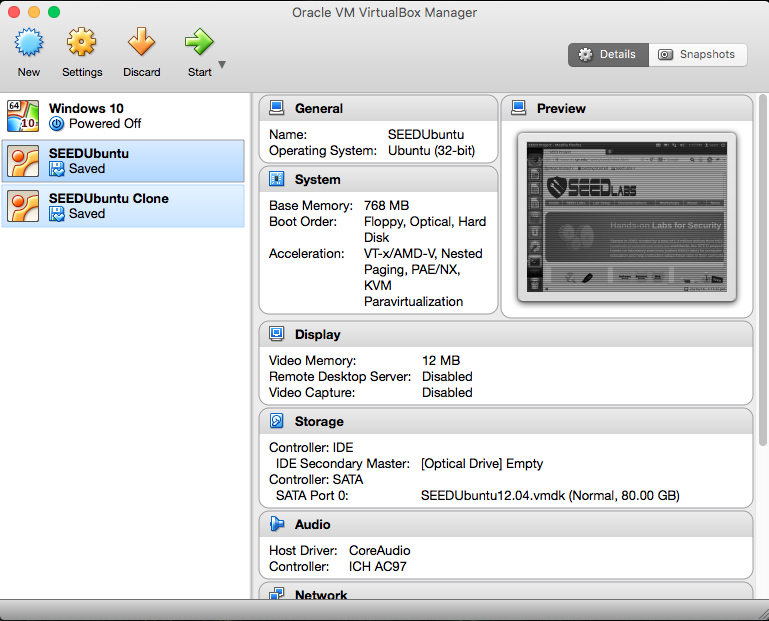
ASU ID: 120623882

Lab Assignment 2

TCP Attacks

**Summary:** The goal of this lab is to conduct different TCP based attacks on a foreign host. These attacks are SYN flooding, TCP RST attack, and TCP session hijacking.

**Network Setup:**



This lab will be conducted with two Ubuntu virtual machines, SEEDUbuntu and SEEDUbuntu Clone, with the former being the host machine, and the latter being the victim. The host will be referred to as VM1 while the victim will be VM2.

**Software Packages Used:**

* Wireshark
* Netwox

**Section 1**

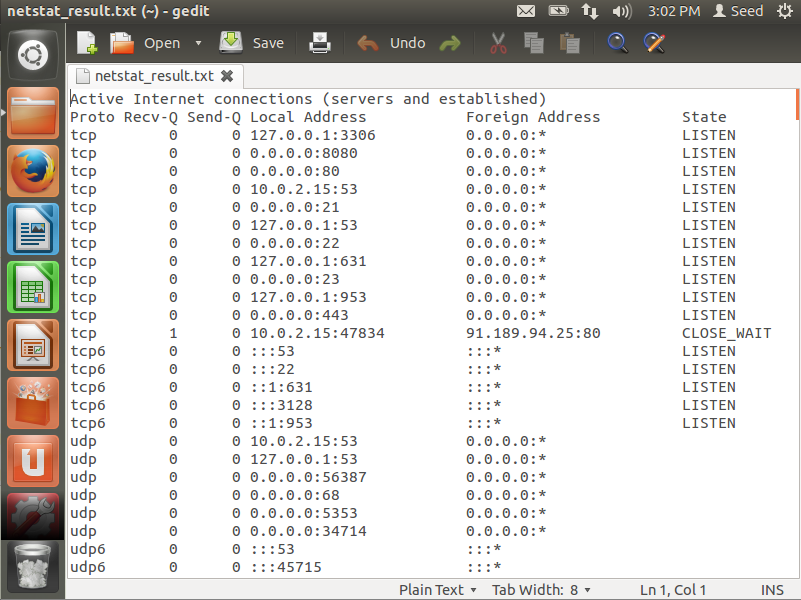
After the virtual machines are set up, they are both set up on a NAT network so that the two machines are capable of contacting each other. The IP address of VM1 is 10.0.2.4 and the IP address of VM2 is 10.0.2.5.

**Section 2**

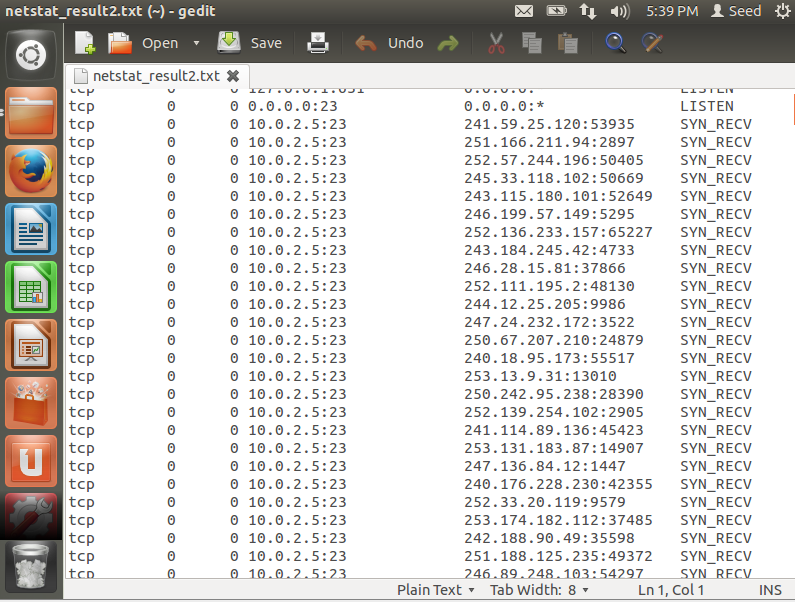
**Task 1 --- SYN Flooding Attack**

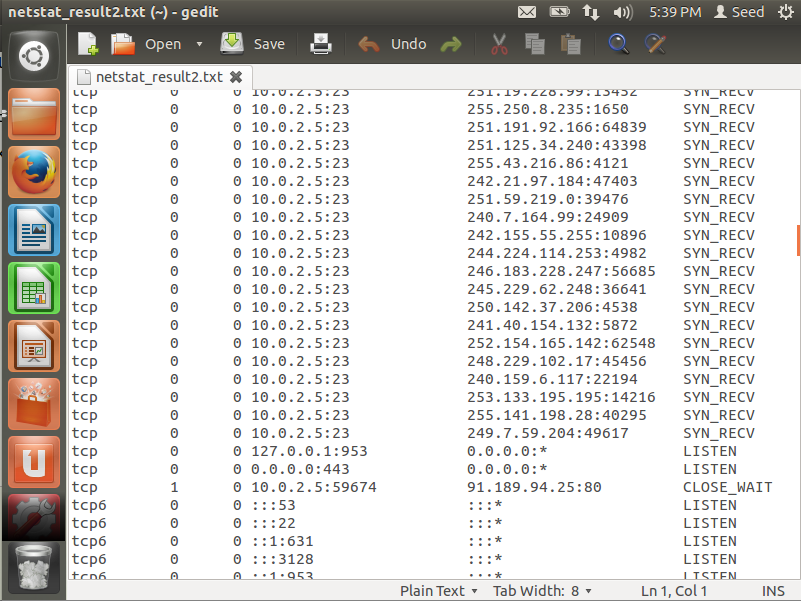
A SYN flood attack is a form of DoS attack, inhibiting the TCP requests of other machines towards the host. The attack is conducted by filling the victim’s TCP queue with SYN requests using spoofed IP addresses. These SYN requests cause the victim machine to return a SYN-ACK message to the spoofed IP address. These requests will fill up the queue and they will not be removed until they time out. With no space left in the queue, valid TCP SYN requests will be unable to be responded to.

To check the size of the TCP queue, the command *sysctl –q net.ipv4.tcp\_max\_syn\_backlog* is used. The returned value is 512. This means that for the attack to be successful, at most, the victim will have 512 unresolved SYN requests. To disable the SYN cookie feature, the command *sudo sysctl –w net.ipv4.tcp\_syncookies=0* will disable the security protecting against SYN flood attacks. To actually perform the attack, the netwox suite of programs will be used. The command used was *sudo netwox 76 –i “10.0.2.5” –p “23.* The 76 denotes which function in the netwox suite will be used, with 76 correlating to a SYN flood attack. The –i “10.0.2.5” denotes the IP address of the target, VM2, and –p “23” denotes the port number of the target, with 23 correlating to the Telnet port. Wireshark is used on the victim machine to observe the received SYN messages.

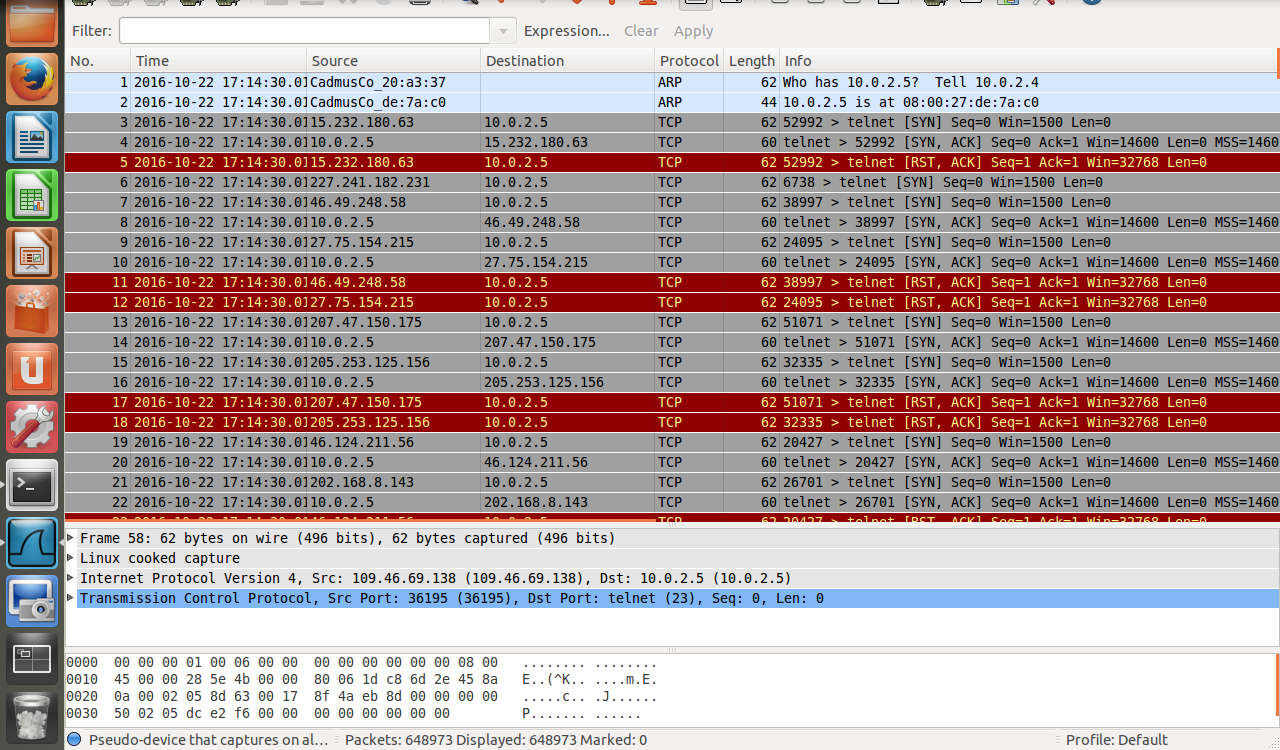


By using the command *netstat –na,* we can view the active internet connections which will display the TCP queue. Above is the queue before the attack is executed. There are only a few TCP connections that are set to listen.





The above two images show the beginning and the end of the TCP queue respectively. Noticeably, there are significantly more TCP connections with a randomized collection of source IP addresses. The connections with the state SYS\_RECV are waiting for an ACK from the source IP.

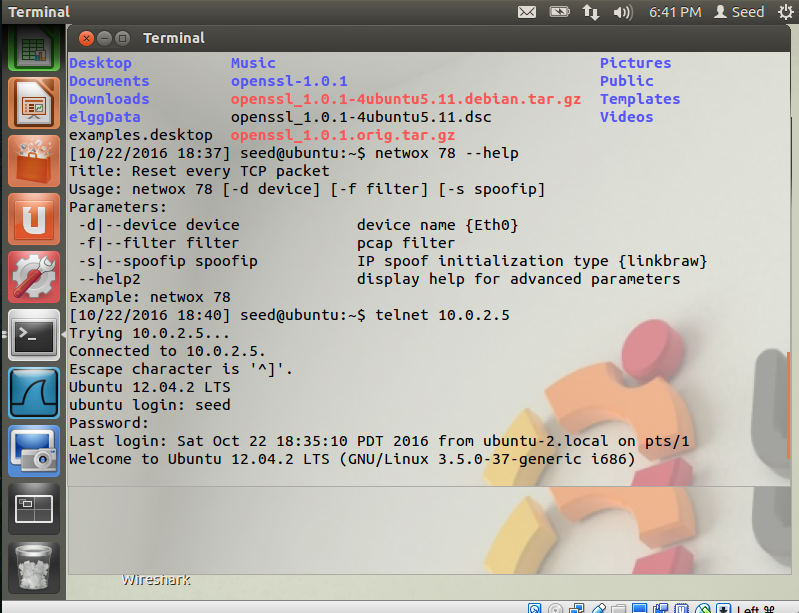


Above are the Wireshark statistics gathered during the SYN flood attack. Each of the spoofed IP address only send a single TCP SYN message to the victim, prompting the victim to return a SYN-ACK. Without responding to the SYN-ACK message, the attempted connection takes up a space in the TCP queue.

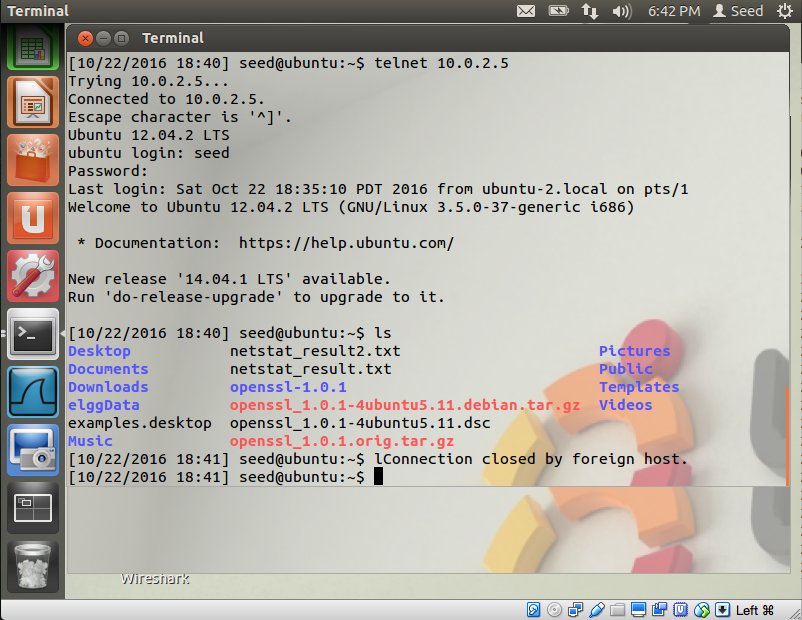
While the attack was commencing, I used VM1 to attempt to connect to the victim machine by using the command *telnet 10.0.2.5* but the connection was not established.

**Task2 --- TCP RST Attack on *telnet* Connections**

A TCP RST attack sends a RST packet to a port on a victim computer. When the RST flag is set on a TCP packet, it terminates the TCP connection between the two machines. The command used to conduct a TCP RST attack is *sudo netwox 78 –i 10.0.2.5*, with 10.0.2.5 being the IP address of the victim.



Above is the telnet connection established with VM1 on VM2, created by using the command *telnet 10.0.2.5*.

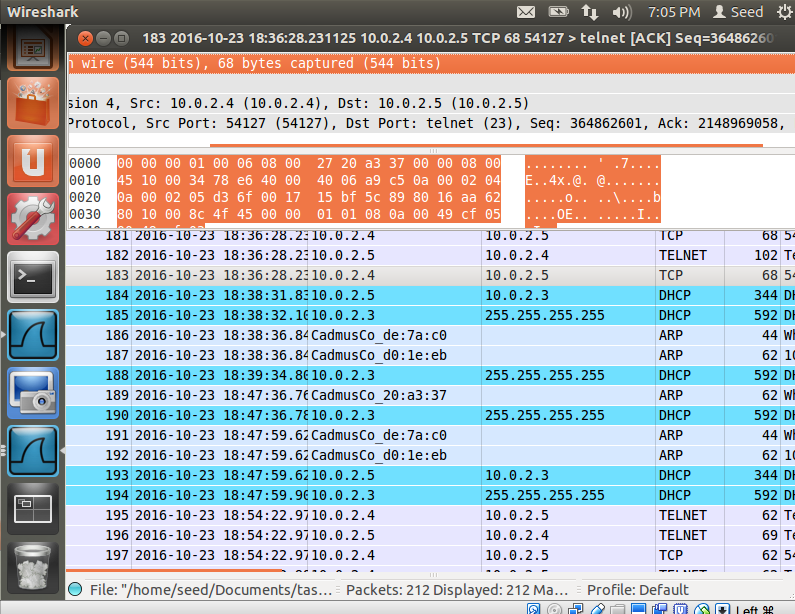


Above is the result of the RST attack, with the status ‘Connection closed by foreign host.’ An important aspect of this attack is that the connection was not terminated until I entered the letter ‘l’, meaning that the netwox program waited until VM2 received a TCP packet before the RST message was sent. If the attack is started before the connection is established, the attack shuts it down immediately before it can be completed. The results were the same when the connection was made using SSH instead of Telnet. Also, the victim was unable to load any webpages as google.com and asu.edu both stated that a connection couldn’t be established. This shows that the RST attack does not just target the Telnet port.

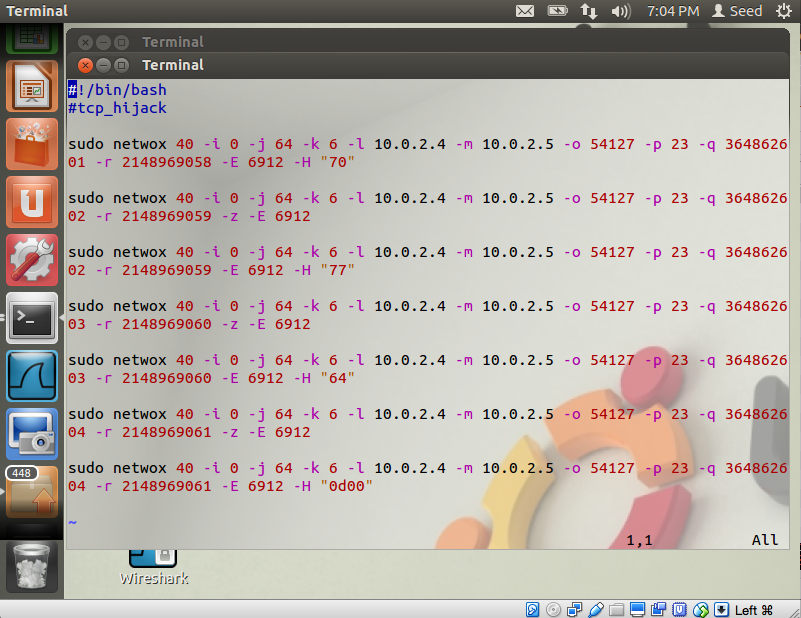
**Task 3 --- TCP Session Hijacking**

TCP session hijacking is when an attacker injects malicious code into a valid TCP connection between two machines. In the case of a Telnet session, the attacker can inject shell commands into the session. In order to successfully conduct a session hijack, the attacker needs to spoof both the machines IP addresses and port numbers and, importantly, the sequence number and acknowledgement number of the connection.

The TCP session is established by using the command *telnet 10.0.2.*5 on VM1. By using Wireshark on VM2, we can monitor the received and sent segments during the Telnet connection. By finding the last TCP ACK message sent by VM1, the required sequence number and acknowledgement number can be found so that we can constructed spoofed TCP messages that VM2 will think are valid and sent by VM1.



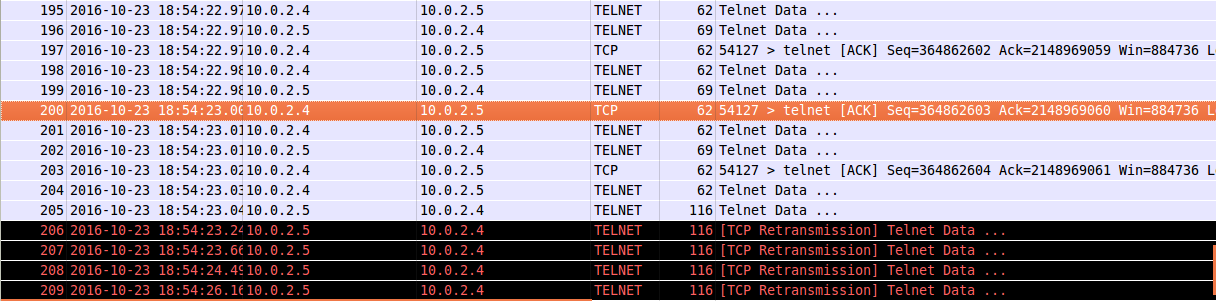
Above shows the expanded packet number 183, sent by VM1. To create a spoofed TELNET packet, the sequence number 364862601 and the acknowledgement number 2148969058 need to be used. Since the connection to VM2 is a Telnet connection, we know the destination port is number 23. For the source port though, it can vary. In this instance, it is port number 54127.



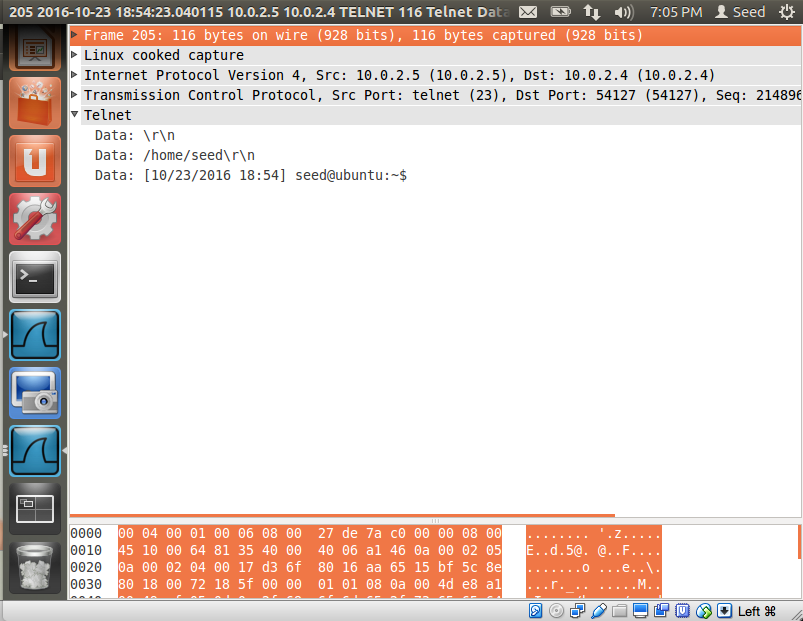
Above is a script created to send all the packets sequentially. The base command, *netwox 40*, will construct IPv4/TCP packets based on the parameters that follow. The parameters used are shown below.

|  |  |
| --- | --- |
| -i | Sets the fragment offset. All packets were set to 0. |
| -j | Sets the TTL (time to live). All packets were set to 64. |
| -k | Selects the IPv4 protocol. 6 denotes TCP. |
| -l | Sets the spoofed source IP address. 10.0.2.4 is the IP address of VM1 and was used in every packet. |
| -m | Sets the spoofed destination IP address. 10.0.2.5 is the IP address of VM2 and was used in every packet. |
| -o | Sets the spoofed source port number. Gathered from Wireshark, we found the port number to be 54127. |
| -p | Sets the spoofed destination port number. Because the connection to VM2 is made using Telnet, we can assume that the port is number 23, the Telnet default. |
| -q | Sets the spoofed sequence number. The first message uses 364862601, the second two messages use 364862602, the third two use 364862603, and the last two use 364862604. |
| -r | Sets the spoofed acknowledgement number. The first message uses 2148969058, the second and third use 2148969059, the fourth and fifth use 2148969060, and the last two use 2148969061. |
| -z | Sets to ACK flag in the TCP header. |
| -E | Sets the TCP window size. Each packet uses 6912. |
| -H | Sets the data that will be sent with the packet. |

The data sent by the packets, “70” “77” “64 “0d00”, is the hex representation of the characters ‘p’ ‘w’ ‘d’ ‘\r’. After each packet containing data is sent, an ACK message is sent with the sequence number and acknowledgement number incremented by 1.



Above are the seven sent spoofed packets and the responses received. Because the script did not send a final ACK message, VM2 keeps sending the last message, expecting a response.



Above is the last packet sent by VM2, after it received the Telnet data “70” “77” “64” and “0d00”. Because ‘pwd’ is the shell command that prints the working (aka current) directory, VM2 sends the data ‘/home/seed\r\n’, the current directory the Telnet connection is at.

**Conclusion**

By manipulating spoofed TCP segments, attackers can conduct DoS attacks, reset established TCP connections, and inject malicious code into established connections. SYN flooding is conducted by spamming the victim machine with SYN requests so that the queue storing TCP requests fills to capacity. Once capacity is reached, the victim will be unable to respond to legitimate TCP requests. TCP RST attacks send spoofed RST messages to the target machine, ending the TCP connection. TCP session hijacking requires the attacker to acquire the sequence and acknowledgment number of the target connection, on top of the IP addresses and port numbers. Once these are acquired, the attacker can build TCP messages that mimic the valid messages in the connection and send malicious data to one or both of the hosts.