# CS 5153/5053 Network Security, Spring 2023

## Project 3: TCP Attacks

## Report

### Student: Austin Tyler Conn

Contents

[Link to Source Code 3](#_Toc130933103)

[Host Environment Used 3](#_Toc130933104)

[Docker Information 3](#_Toc130933105)

[Assumptions 3](#_Toc130933106)

[Task 1 4](#_Toc130933107)

[How did you perform the attack in your VM 4](#_Toc130933108)

[Screenshots 7](#_Toc130933109)

[Was the attack successful 8](#_Toc130933110)

[Task 2 - Manual 9](#_Toc130933111)

[How did you perform the attack in your VM 9](#_Toc130933112)

[Screenshots 13](#_Toc130933113)

[Was the attack successful 13](#_Toc130933114)

[Task 2 - Automated 14](#_Toc130933115)

[How did you perform the attack in your VM 14](#_Toc130933116)

[Screenshots 16](#_Toc130933117)

[Was the attack successful 16](#_Toc130933118)

[Task 4 17](#_Toc130933119)

[How did you perform the attack in your VM 17](#_Toc130933120)

[Screenshots 22](#_Toc130933121)

[Was the attack successful 22](#_Toc130933122)

[Task 5 23](#_Toc130933123)

[How did you perform the attack in your VM 23](#_Toc130933124)

[Screenshots 28](#_Toc130933125)

[Was the attack successful 28](#_Toc130933126)

Link to Source Code <https://github.com/austinc3030/tcp_m11809075>

## Host Environment Used

Operating System: Ubuntu 20.04 LTS



Hardware: Google Cloud E2 Instance

Links Used for Environment Setup:

* [seed-labs/seedvm-cloud.md at master · seed-labs/seed-labs (github.com)](https://github.com/seed-labs/seed-labs/blob/master/manuals/cloud/seedvm-cloud.md)
* [seed-labs/create\_vm\_gcp.md at master · seed-labs/seed-labs (github.com)](https://github.com/seed-labs/seed-labs/blob/master/manuals/cloud/create_vm_gcp.md)

## Docker Information

Text

Description automatically generated

Text

Description automatically generated

## Assumptions

1. Mapping between PDF document and docker containers provided:
   1. Client (10.0.2.5) = user1-10.9.0.6 (10.9.0.6)
   2. Server (10.0.2.6) = victim-10.9.0.5 (10.9.0.5)
   3. Attacker (10.0.2.7) = seed-attacker (10.9.0.1)

## Task 1

### How did you perform the attack in your VM

1. Write code for scapy.

Text

Description automatically generated

1. Check the size of net.ipv4.tcp\_max\_syn\_backlog on the victim/server.

Graphical user interface, text, application

Description automatically generated

1. Check the connections on the victim/server.

Text

Description automatically generated

1. Initiate a telnet session from user1/client to the victim/server.

Text

Description automatically generated

1. Check connections on the victim/server to see the new telnet connection.

Text

Description automatically generated

1. Disable SYN cookies on the victim/server per the assignment instructions (Note: the SEED Lab Docker Image for the victim/server already has SYN cookies disabled.)

Text

Description automatically generated

1. From the attacker, initiate a SYN attack using code from step 1.A screenshot of a computer

   Description automatically generated with medium confidence
2. Attempt to initiate a new telnet session from user1/client to the victim/server.

Text

Description automatically generated

1. Check netstat on the victim/server to see the active connections.

A picture containing background pattern

Description automatically generated

*Note: Full output of netstat -nat above, truncated output below for readability.*

A screenshot of a computer

Description automatically generated with medium confidence

### Screenshots

See screenshots in “How did you perform the attack in your VM”

### Was the attack successful

Yes, the attack was successful. I did find that running only 1 instance of *task1.py* seemed to have intermittent effects in that sometimes the telnet session would establish a connection and allow me to log in. I believe this may be due to the single instance of *task1.py* potentially not creating enough SYN packets fast enough to overwhelm the victim/server and as the victim/server frees a resource, the telnet session from the user1/client is allowed to establish. This makes sense considering other DOS attacks I am familiar with where it was not a single IP or machine causing the DOS but rather a botnet or network of many computers causing the DOS.

Further evidence to support that the attack was successful is the output of *netstat -nat* where it shows many ‘foreign’ IP addresses that are random and implausible in this lab network as our network is in the 10.9.0.0 address space.

*task1.py* could further be improved by allowing arguments to be passed to the script such as target IP, target port rather than having them hardcoded in the script. Also, making use of a parallel process such as python’s “threads” to spawn multiple loops, each sending out SYN packets. This would eliminate the need to run multiple instances from the command line.

## Task 2 - Manual

### How did you perform the attack in your VM

1. Start Wireshark monitoring traffic between attacker, user1/client, and the victim/server
   1. (Note: Filtering is required as Wireshark is running on the host VM and is monitoring traffic between the docker containers. Since we are only interested in the attacker, user1/client, and the victim/server, the following filter was used: “*ip.src==10.9.0.1 or ip.src==10.9.0.5 or ip.src==10.9.0.6”)*

Graphical user interface, text, application, email

Description automatically generated

1. Establish a telnet session between the user1/client and the victim/server and log in successfully.

Text

Description automatically generated

1. Run a command in the telnet session to generate packets.

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

1. In Wireshark, find the last packet sent **from** the victim/server **to** the user1/client. Note the destination port and the next sequence number in the packet.

Graphical user interface, application

Description automatically generated

1. Replace the variables *intDestinationPort* and *intNextSequenceNumber* in the following code with the values found in step 4, respectively.

Text

Description automatically generated

1. Run *task2\_manual.py* to initiate a TCP Reset Attack

Text

Description automatically generated

1. Upon running *task2\_manual.py*, the telnet session between user1/client and the victim/server will be terminated immediately with the message “Connection closed by foreign host.”

Graphical user interface, text, application

Description automatically generated

### Screenshots

See screenshots in “How did you perform the attack in your VM”

### Was the attack successful

Yes, the attack was successful as is evident by the connection being closed immediately upon *task2\_manual.py* running, as well as the resulting packets inspected in Wireshark.

## Task 2 - Automated

### How did you perform the attack in your VM

1. Write code for scapy.

Text

Description automatically generated

1. Initiate a telnet connection from the user1/client to the victim/server and successfully log in.

Text

Description automatically generated

1. Run a command to verify the connection is fully connected.

Graphical user interface, text, application

Description automatically generated

1. Run *task2\_automated.py* on the attacker

Graphical user interface, text, application

Description automatically generated

1. Nothing will happen immediately, but when trying to type a new command in the telnet session, after the first character, the connection will be closed with the message “Connection closed by foreign host.”

Graphical user interface, text

Description automatically generated

### Screenshots

See screenshots in “How did you perform the attack in your VM”

### Was the attack successful

Yes, the attack was successful as is evident by the telnet session being terminated with the message “Connection closed by foreign host.” as soon as the first character is entered into the session after starting *task2\_automated.py*.

## Task 4

### How did you perform the attack in your VM

1. Start Wireshark monitoring traffic between attacker, user1/client, and the victim/server
   1. (Note: Filtering is required as Wireshark is running on the host VM and is monitoring traffic between the docker containers. Since we are only interested in the attacker, user1/client, and the victim/server, the following filter was used: “*ip.src==10.9.0.1 or ip.src==10.9.0.5 or ip.src==10.9.0.6”)*

Graphical user interface, application

Description automatically generated

1. Establish a telnet session between the user1/client and the victim/server and log in successfully.

Text

Description automatically generated

1. Run a command in the telnet session to generate packets.

Graphical user interface, text, application

Description automatically generated

1. In Wireshark, find the last packet sent **from** the user1/client **to** the victim/server. Note the destination port, the next sequence number, and the acknowledgement number in the packet.

Graphical user interface, application

Description automatically generated

1. Replace the variables *intSourcePort, intNextSequenceNumber,* and *intAcknowledgementValue* in the following code with the values found in step 4, respectively.

Text

Description automatically generated

1. In a second terminal on the attacker, run *nc -nlv 9090* to listen for connections on port 9090.

Graphical user interface, text, application, chat or text message

Description automatically generated

1. Run *task4.py* to initiate a TCP Hijacking Attack.

Text

Description automatically generated

1. Upon running *task4.py*, the telnet session on the user1/client will become unresponsive, and the contents of the *secret.txt* file will be output on the second terminal session run in step 6 on the attacker.

Graphical user interface, text, application

Description automatically generated

*Note: Telnet session (Above) becomes unresponsive. The terminal on the attacker listening for connections outputs the contents of secret.txt.*

Graphical user interface, text, application

Description automatically generated

### Screenshots

See screenshots in “How did you perform the attack in your VM”

### Was the attack successful

Yes, the attack was successful as evidenced by the output of the contents of the *secret.txt* file in the terminal on the attacker that was listening for connections.

## Task 5

### How did you perform the attack in your VM

1. Start Wireshark monitoring traffic between attacker, user1/client, and the victim/server
   1. (Note: Filtering is required as Wireshark is running on the host VM and is monitoring traffic between the docker containers. Since we are only interested in the attacker, user1/client, and the victim/server, the following filter was used: “*ip.src==10.9.0.1 or ip.src==10.9.0.5 or ip.src==10.9.0.6”)*

Graphical user interface, application

Description automatically generated

1. Establish a telnet session between the user1/client and the victim/server and log in successfully.

Text

Description automatically generated

1. Run a command in the telnet session to generate packets.

Graphical user interface, text, application

Description automatically generated

1. In Wireshark, find the last packet sent **from** the user1/client **to** the victim/server. Note the destination port, the next sequence number, and the acknowledgement number in the packet.

Graphical user interface, application

Description automatically generated

1. Replace the variables *intSourcePort, intNextSequenceNumber,* and *intAcknowledgementValue* in the following code with the values found in step 4, respectively.

Text

Description automatically generated

1. In a second terminal on the attacker, run *nc -nlv 9090* to listen for connections on port 9090.

Graphical user interface, text, application

Description automatically generated

1. Run *task5.py* to initiate a TCP Hijacking Attack.

A screenshot of a computer

Description automatically generated with medium confidence

1. Upon running *task5.py*, the telnet session on the user1/client will become unresponsive, and an interactive bash shell on the second terminal session run in step 6 on the attacker will be established.

Graphical user interface, text, application

Description automatically generated

*Note: Telnet session (Above) becomes unresponsive. The terminal on the attacker listening for connections gives the bash prompt from the victim/server.*

Graphical user interface, text, application

Description automatically generated

### Screenshots

See screenshots in “How did you perform the attack in your VM”

### Was the attack successful

Yes, the attack was successful as is evidenced by the interactive bash shell present in the second terminal session established on the attacker. This is confirmed by matching the hostname of the victim/server against the hostname shown in the newly connected interactive bash prompt.