**Crypto and Network Security**

**Lab 4**

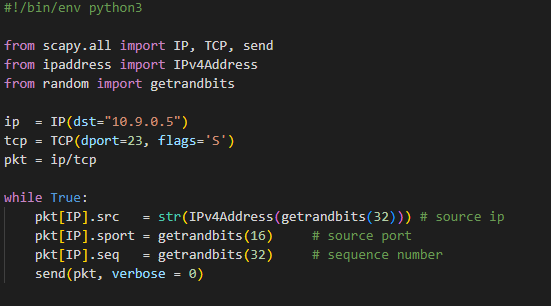
**Name: Gaurav Setty**

**Email:** [**settgm01@pfw.edu**](mailto:settgm01@pfw.edu)

**Task 1: SYN Flooding attack**

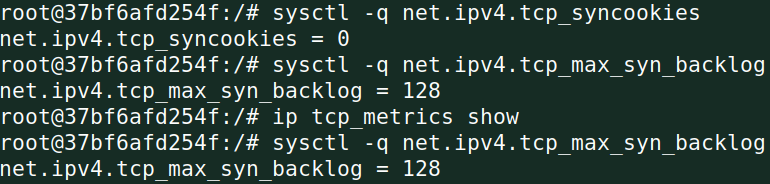
**Task 1.1: Launching the attack using python**

**Synflood.py:**

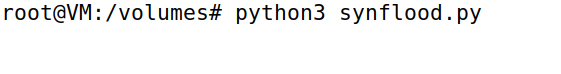


To overflow the victim's cache with false half-open connections, the code above bombards the victim server IP=10.9.0.5 with SYN flag enabled TCP packets.

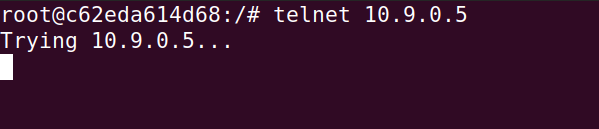
We first check if victim’s syncookies is active and make sure it is not active.

****

Then we run the code:



The Synflood.py attack is now launched from the attacker machine. Try to telnet into the victim container from user1 after a little delay.

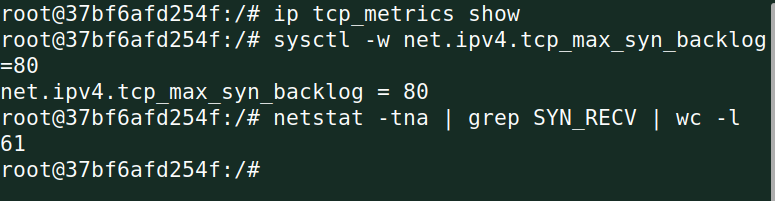
****

The connection is clearly failed.

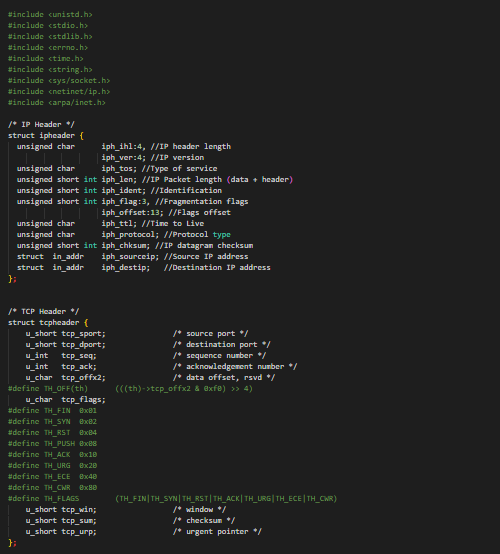
Now, we’ll check the count of network connection count. The screenshot shows the connection is flooded. About one fourth of the connections are reserved and the rest have been flooded by bogus connection requests.

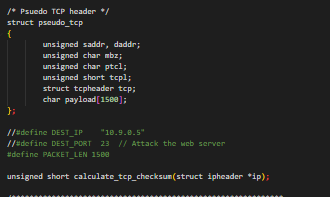
****

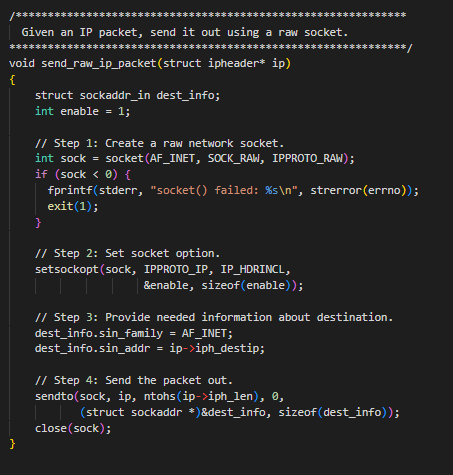
The same process was tested by changing the connection backlog to 80.

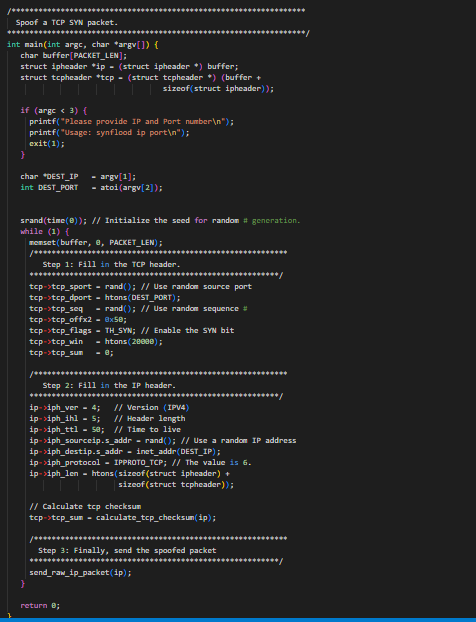


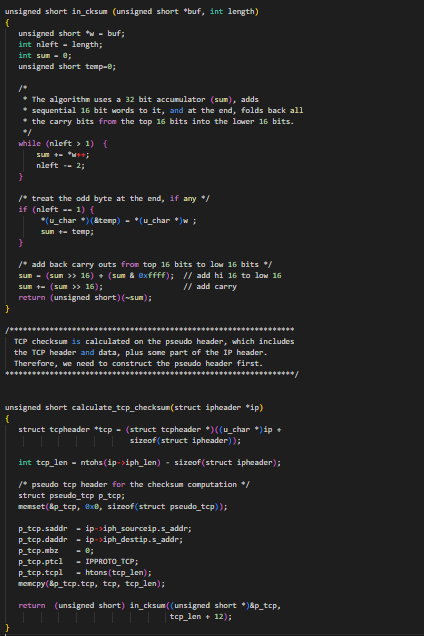
**Task 1.2: Launch the attack using C**





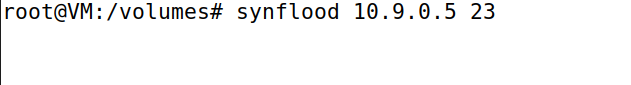




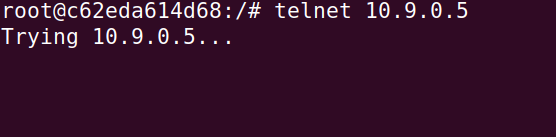


Although written in C, the above code performs the SYN flooding attack as well. It operates a lot more quickly than Python.

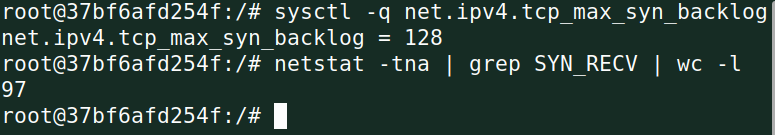
Run the attack code through attacker:



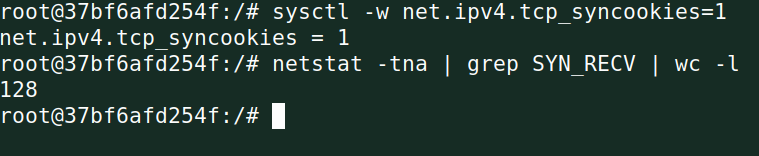
The Synflood.c attack is now launched from the attacker machine. Try to telnet into the victim container from user1 after a little delay.



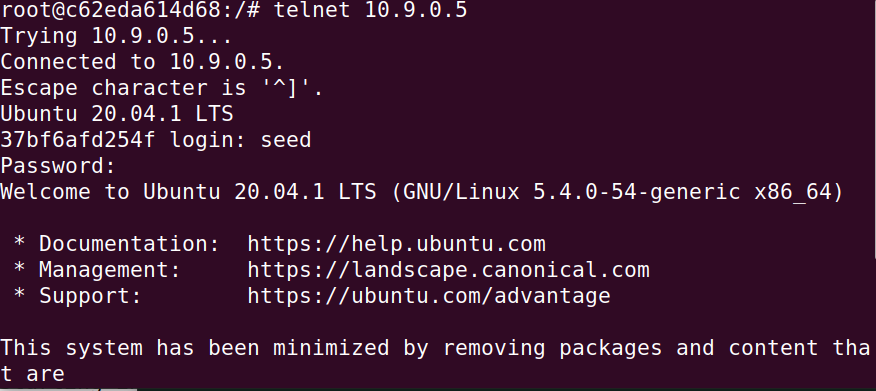
Now, we’ll check the count of network connection count. The screenshot shows the connection is flooded.



**Task 1.3: Enable the SYN Cookie**



The countermeasure kicks in, and the half open attacks are not put to the SYN Queue until SYN-ACK answer is received, therefore even after waiting for a long period after running the Python code, the attack still fails.

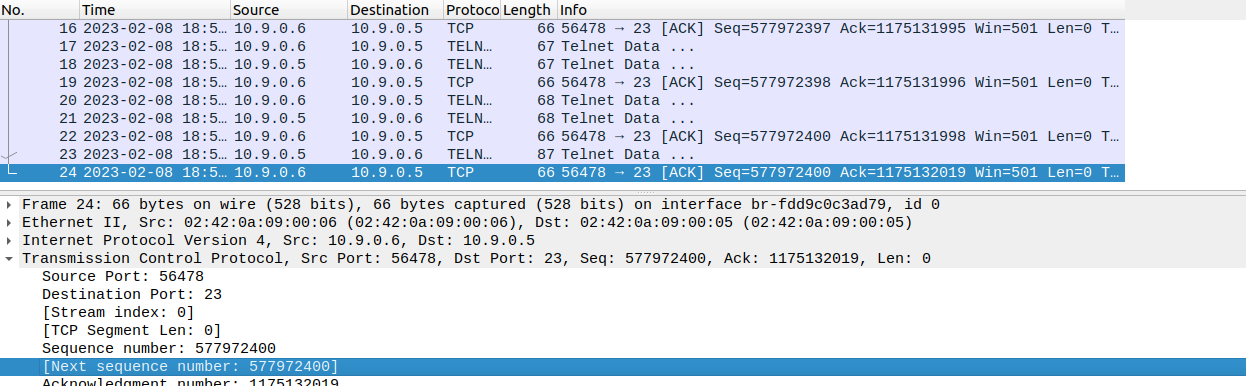


**Task 2: TCP RST Attacks on telnet connections**

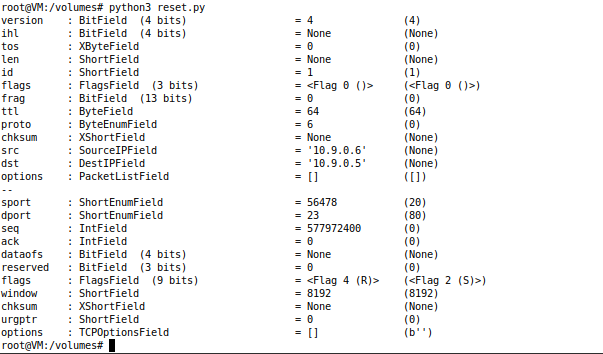
**Reset.py:**



The wireshark is used to sniff the packets coming from user1 going to victim. By which we get sequence number and source port number to use it for the code to generate a valid TCP RST packet.

****

Now, we run the code through attacker machine to perform reset attack.

****

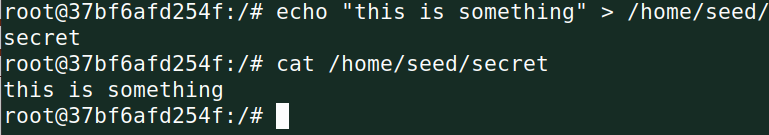
As soon as I pressed a key on the telnet shell to conduct the reset attack, I noticed that the connection had been cut off by the foreign host. The TCP RST attack was thus effective since we were able to persuade the victim container to cut off communication with the attacker machine.

****

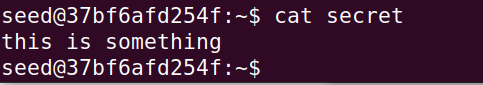
**Task 3: TCP Session Hijacking**

We must first create a scenario in order to simulate a session hijacking assault.

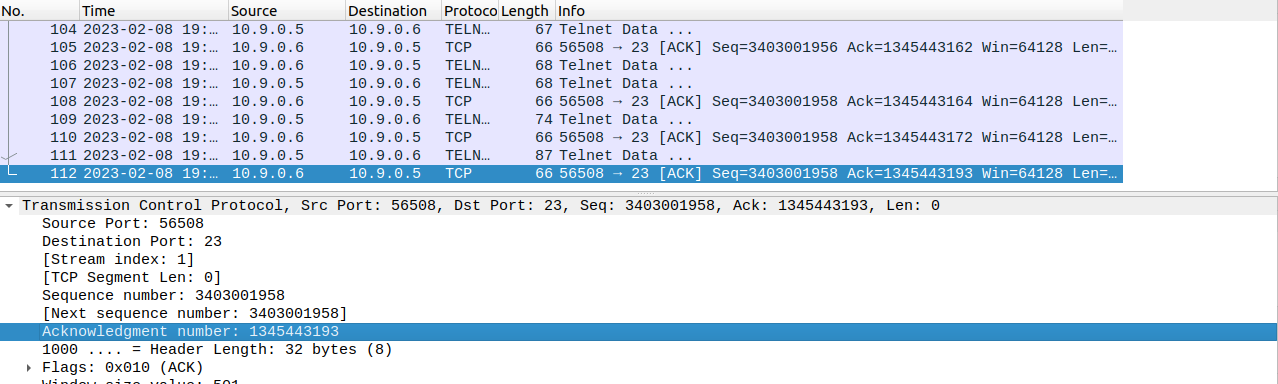
Let's assume that the victim container's root directory contains secret text, which the attacker is aware of. Let's create the confidential document that will be taken:



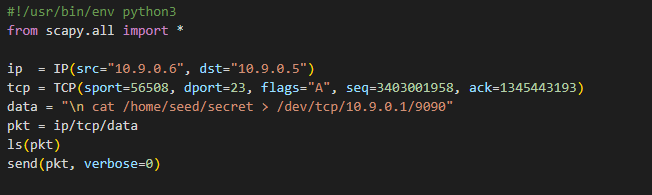
This screenshot below is to verify from destination if the secret text is accessible.



The wireshark is used to sniff the packets coming from user1 going to victim. By which we get sequence number, acknowledgment number and source port number to use it for the code to generate a valid TCP hijacking attack.



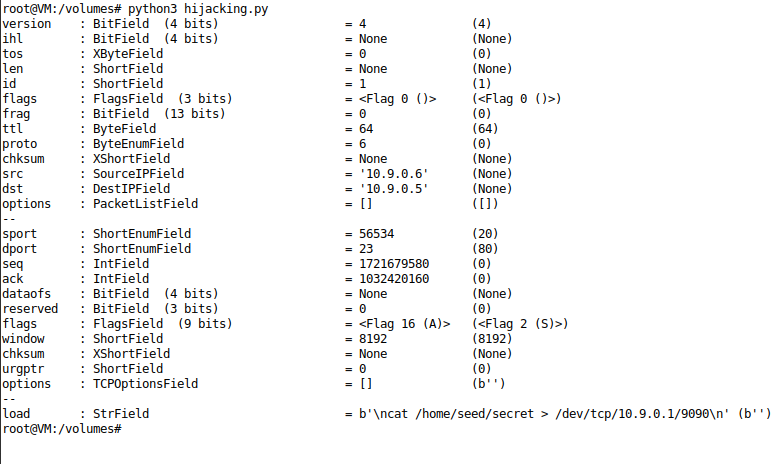
**Hijacking.py:**



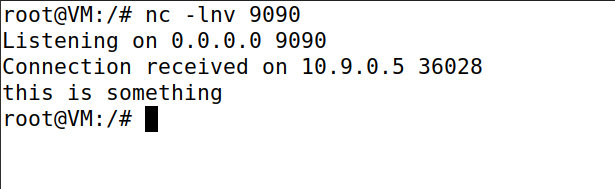
Now we start a listener on the attacker machine using netcat:



Now we launch the session hijacking python code:



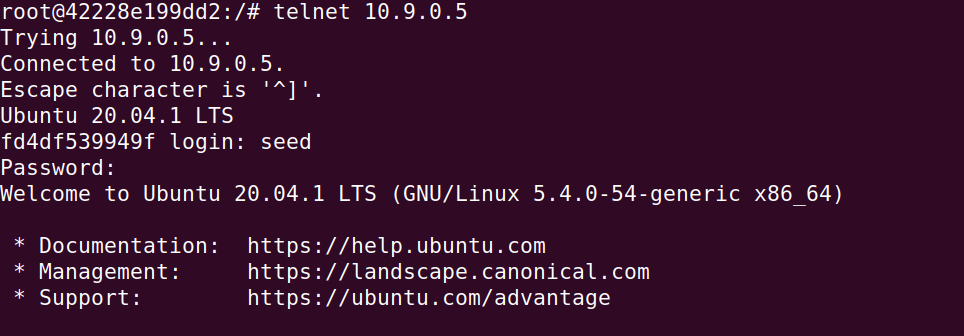
Let us look at the output of the Netcat listener:



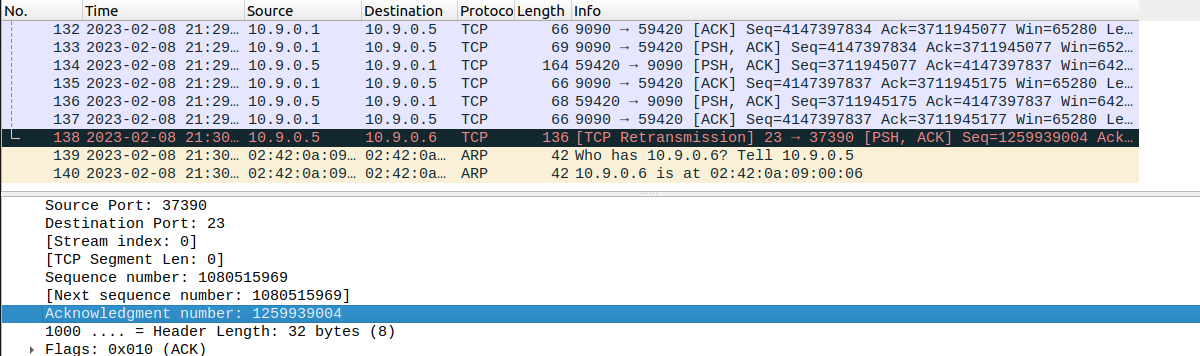
Session hijacking attack is successfully executed and the contents of secret text were stolen by the attacker.

**Task 4: Creating Reverse Shell using TCP Session Hijacking**

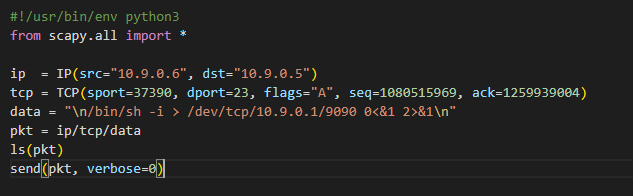
We launch telnet connection between user1 and victim so that we can sniff the connection to gather information to run the code.



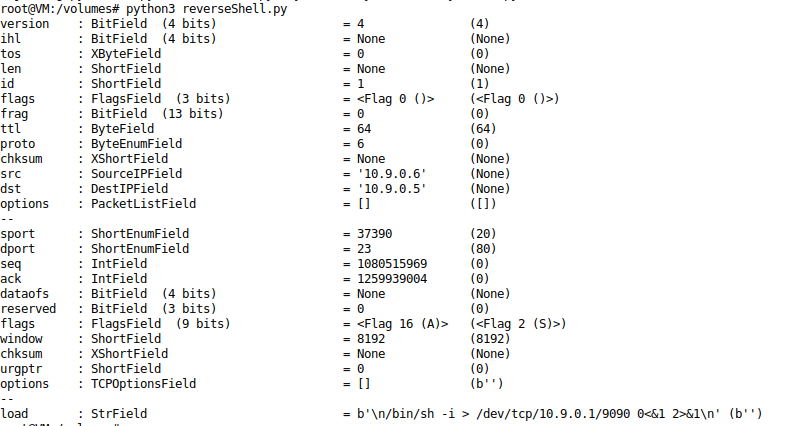
The wireshark is used to sniff the packets coming from user1 going to victim. By which we get sequence number, acknowledgment number, destination port number and source port number to use it for the code to generate reverse shell using TCP hijacking attack.



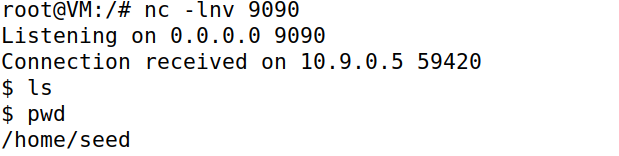
reverseShell.py:



Let’s run the above code to hijack the telnet session:



Now let us look at the shell of the victim to see if these commands were executed.



Reverse shell attack has been successfully executed.