**Exploring TCP: Vulnerabilities and Practical Exploits**

First Name: Azzeddine

Last Name: Zebiri

**Purpose:**

The primary goal of this project is to gain a comprehensive understanding of TCP packets by exploring their vulnerabilities and exploiting them. Through hands-on experimentation, we aim to delve into the intricacies of TCP's workings, focusing on its header structure, to develop a deeper insight into network protocols and potential security risks. And the different type of TCP packet that can be used for different purpose.

**Credential:**

A:

Ipv4: 10.9.0.6

B:

Ipv4:10.9.0.7

V:

Ipv4:10.9.0.5

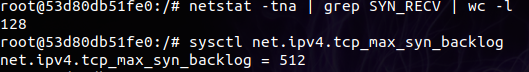
M:

**Task:**

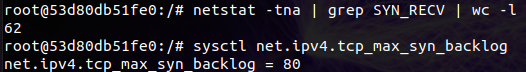
**Task1:**

**Task 1.1:**

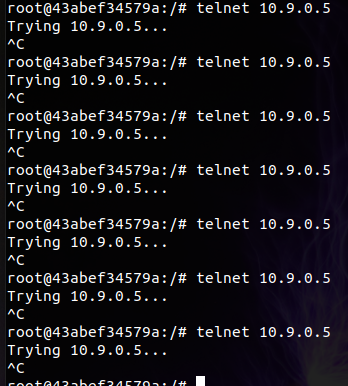
I could always connect using telnet for two reasons. Even though I could open four terminals or so, it was because the maximum value of 'sysctl net.ipv4.tcp\_max\_syn\_backlog' for my case is 512, and I had only 128 open with two terminals. So, I had to adjust it only to 80 so that I wouldn't need to open new terminals. Additionally, since it deletes every 5 seconds, my maximum was 128.



So I used this command sysctl -w net.ipv4.tcp\_max\_syn\_backlog=80



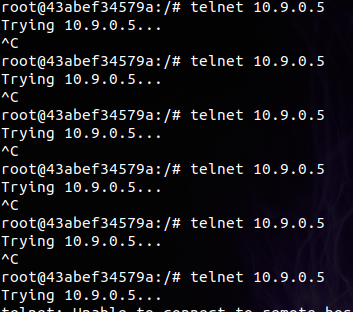
Even though it was a full SYN, I could still connect using Telnet most of the time. There were only two instances or so when it didn't and was stuck in the "trying" state. Now I know why; it was written in the PDF that it's because of TCP cache. So I used “ip tcp\_metrics flush”.



I had a 100% success rate when removing the cache of TCP.

**Task 1.2:**

Using the synflood.c, this was also successful because TCP was flushed

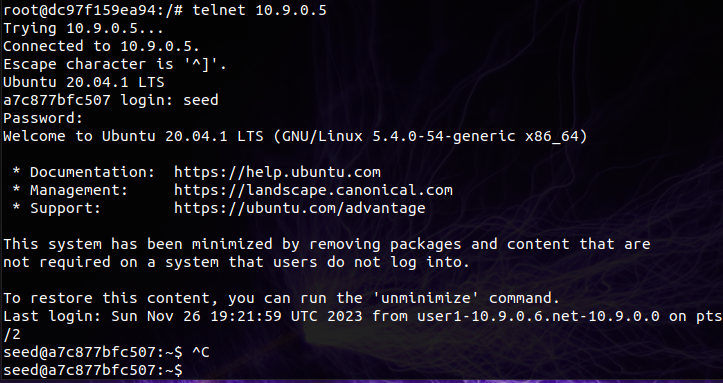


Note: I attempted the procedure without flushing the server, and I could still connect.

**Task 1.3:**

“sysctl -w net.ipv4.tcp\_syncookies=1”

We can see the attack is not successful when activating SYN cookies because it could detect our SYN flooding. It puts the information in cookies, preventing the allocation of resources and leaving them available for legitimate hosts who will finish with an ACK.

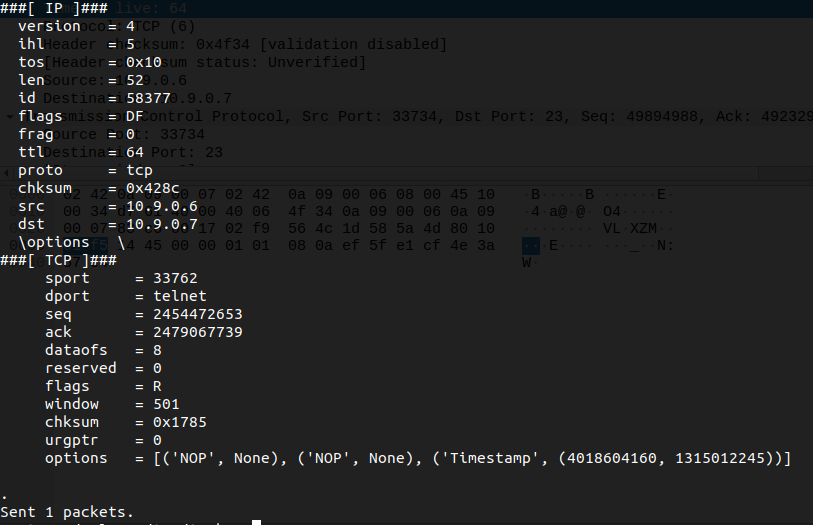


**Task 2:**

The connection was directly closed when trying to write something. By making it look like an error from the sender's side with a false sequence number, we can force the server to close the connection (reset it).



**Auto:**

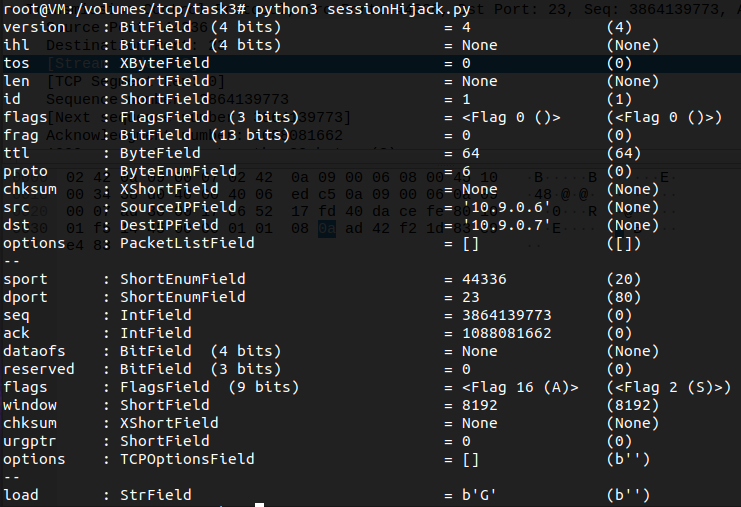
By changing the script I could sniff the packet and send a new one based on the last  
 



**Task 3:**



By using this





The moment I type something it be directly G and if I want to send another one I juste have to do +1 for seq and ack

**Auto:**

**Task 4:**

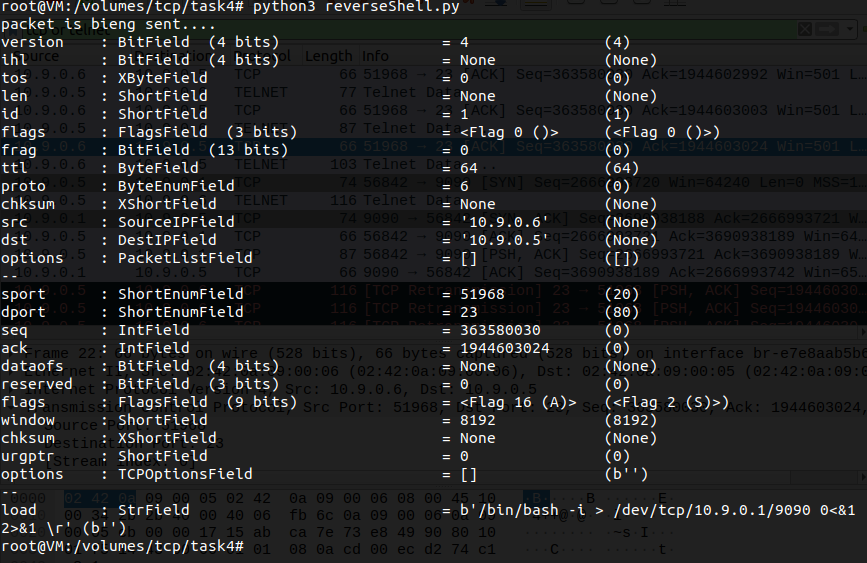
So by doing this “nv –lnv 9090” we are waiting for the attack to be successful and the victim will establish a connection with the attacker.

****

****

By adding the “\r” we say to directly execute this data.

****





This was a sign that the reverse Shell was successful.

**Auto:**