CENG 489
2023 - 2
PA2 - 2

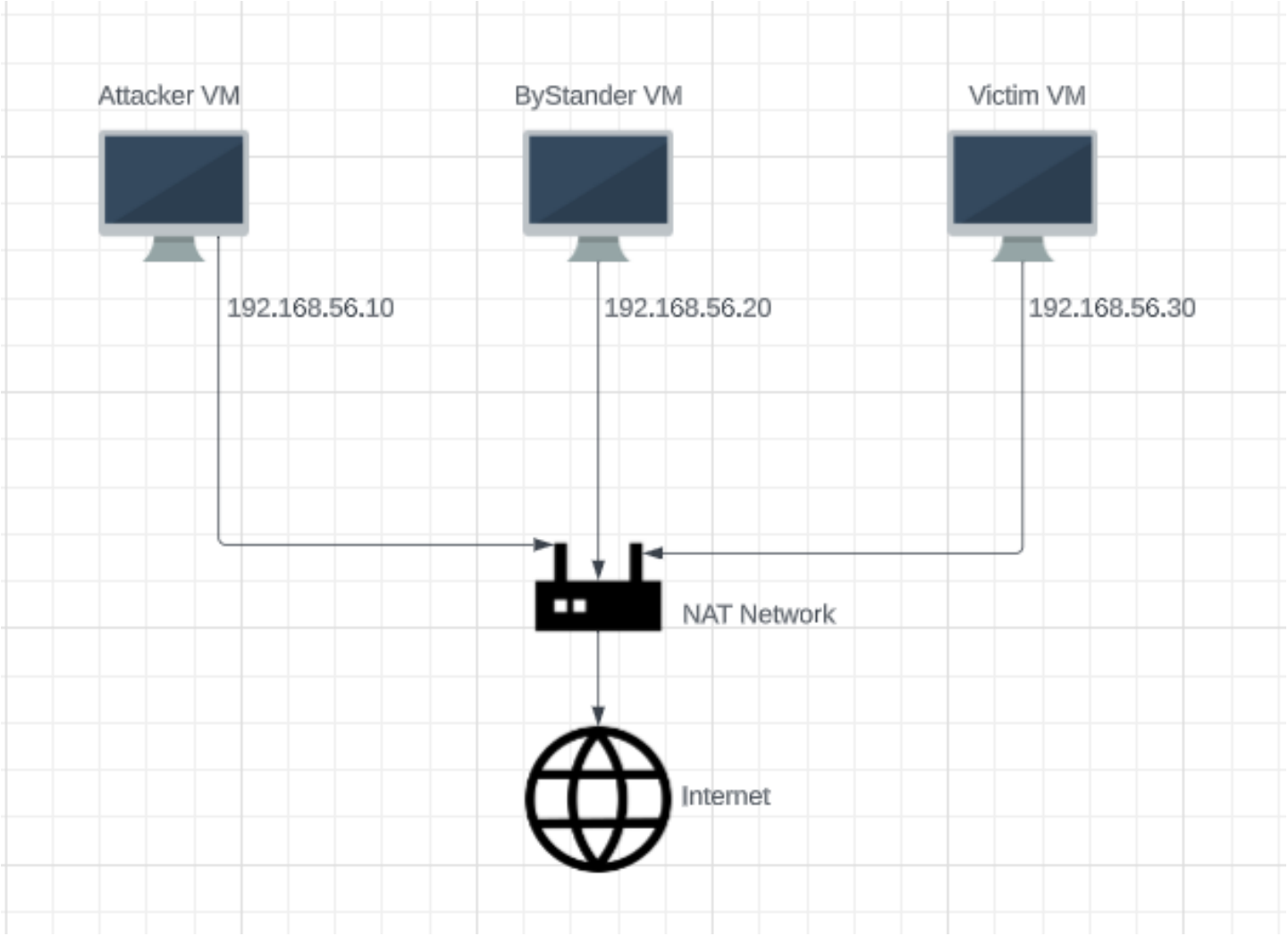
Ünlü, Berke Can
e2381028@ceng.metu.edu.tr

June 21, 2023

---

1 Network Topology

Figure 1: Network Topology



I've used vagrant for the first two attacks since they did not require NAT network between VMs. The attacker uses Kali linux while other two VMs use ubuntu/focal64. For the TCP RST Attack, I've configured virtualbox network settings, therefore the ip addresses of these VMs changed. I will talk about it later.

2 Slowloris

I've used a github repo to perform Slowloris attack[1]
This Denial of Service attack creates multiple sockets to send GET HTTP request to a server to exhaust the threading pool of the server. We can change socket number and sleeping duration of this attack while using this repository. I've set 1024 sockets and 15 seconds sleep duration.
I've created a http server in Victim VM. Then, I've performed attack on Attacker VM, while attacking, when I try to access the server via web browser, it stuck at loading state. I've send over 10.000 packets in this attack as you can see in the figures.

Figure 2: Server in Victim VM

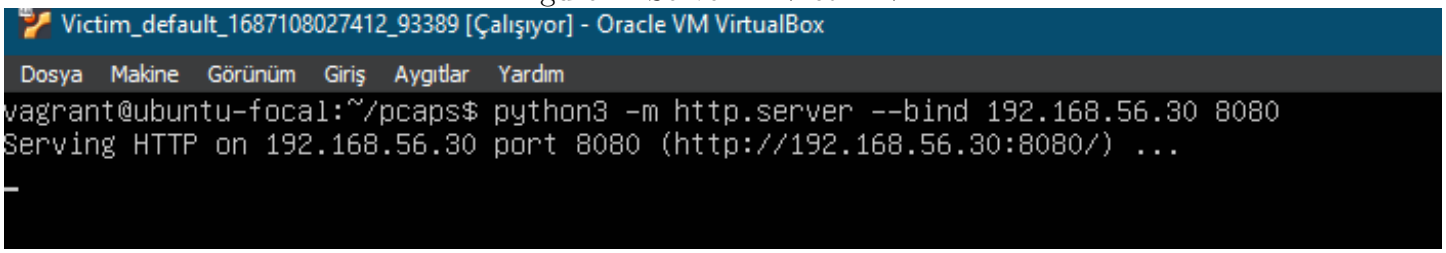


Figure 3: Slowloris Config

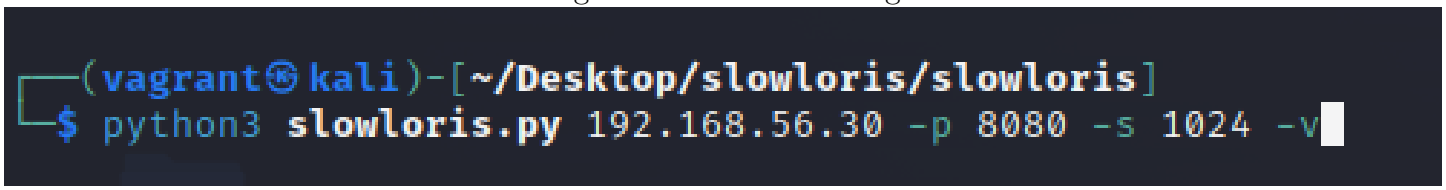
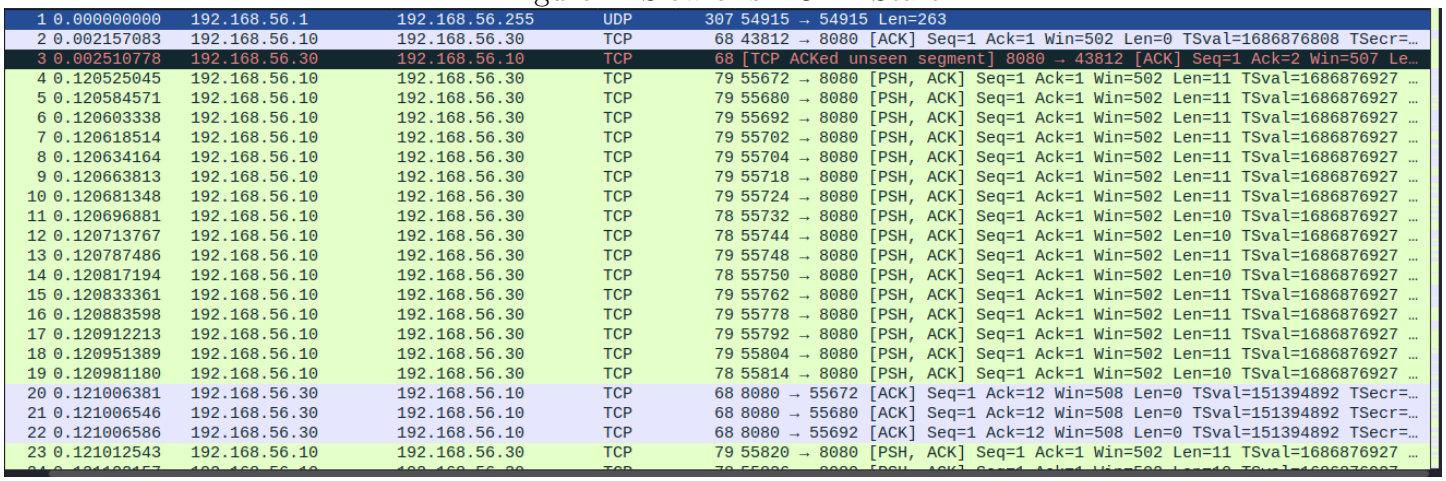


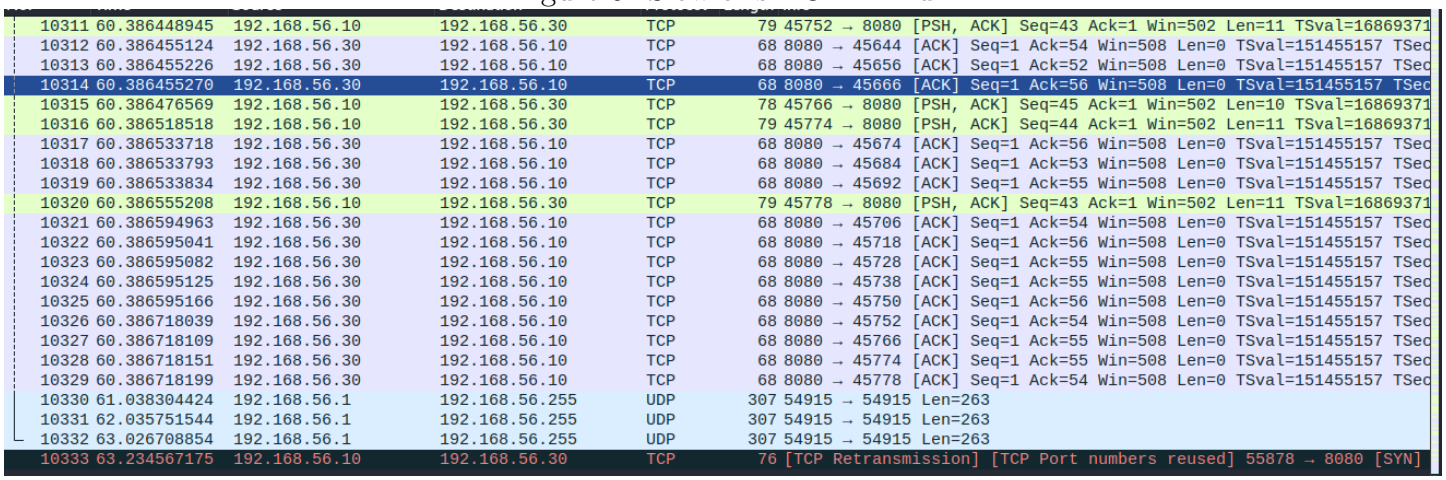
Figure 4: Slowloris PCAP Start



1	0.000000000	192.168.56.1	192.168.56.255	UDP	307	54915 → 54915 Len=263
2	0.002157083	192.168.56.10	192.168.56.30	TCP	68	43812 → 8080 [ACK] Seq=1 Ack=1 Win=502 Len=0 TSval=1686876808 TSecr=…
3	0.002510778	192.168.56.30	192.168.56.10	TCP	68	[TCP ACKed unseen segment] 8080 → 43812 [ACK] Seq=1 Ack=2 Win=507 Le…
4	0.120525045	192.168.56.10	192.168.56.30	TCP	79	55672 → 8080 [PSH, ACK] Seq=1 Ack=1 Win=502 Len=11 TSval=1686876927 …
5	0.120584571	192.168.56.10	192.168.56.30	TCP	79	55680 → 8080 [PSH, ACK] Seq=1 Ack=1 Win=502 Len=11 TSval=1686876927 …
6	0.120603338	192.168.56.10	192.168.56.30	TCP	79	55692 → 8080 [PSH, ACK] Seq=1 Ack=1 Win=502 Len=11 TSval=1686876927 …
7	0.120618514	192.168.56.10	192.168.56.30	TCP	79	55702 → 8080 [PSH, ACK] Seq=1 Ack=1 Win=502 Len=11 TSval=1686876927 …
8	0.120634164	192.168.56.10	192.168.56.30	TCP	79	55704 → 8080 [PSH, ACK] Seq=1 Ack=1 Win=502 Len=11 TSval=1686876927 …
9	0.120663813	192.168.56.10	192.168.56.30	TCP	79	55718 → 8080 [PSH, ACK] Seq=1 Ack=1 Win=502 Len=11 TSval=1686876927 …
10	0.120681348	192.168.56.10	192.168.56.30	TCP	79	55724 → 8080 [PSH, ACK] Seq=1 Ack=1 Win=502 Len=11 TSval=1686876927 …
11	0.120696881	192.168.56.10	192.168.56.30	TCP	78	55732 → 8080 [PSH, ACK] Seq=1 Ack=1 Win=502 Len=10 TSval=1686876927 …
12	0.120713767	192.168.56.10	192.168.56.30	TCP	78	55744 → 8080 [PSH, ACK] Seq=1 Ack=1 Win=502 Len=10 TSval=1686876927 …
13	0.120787486	192.168.56.10	192.168.56.30	TCP	79	55748 → 8080 [PSH, ACK] Seq=1 Ack=1 Win=502 Len=11 TSval=1686876927 …
14	0.120817194	192.168.56.10	192.168.56.30	TCP	78	55750 → 8080 [PSH, ACK] Seq=1 Ack=1 Win=502 Len=10 TSval=1686876927 …
15	0.120833361	192.168.56.10	192.168.56.30	TCP	79	55762 → 8080 [PSH, ACK] Seq=1 Ack=1 Win=502 Len=11 TSval=1686876927 …
16	0.120883598	192.168.56.10	192.168.56.30	TCP	79	55778 → 8080 [PSH, ACK] Seq=1 Ack=1 Win=502 Len=11 TSval=1686876927 …
17	0.120912213	192.168.56.10	192.168.56.30	TCP	79	55792 → 8080 [PSH, ACK] Seq=1 Ack=1 Win=502 Len=11 TSval=1686876927 …
18	0.120951389	192.168.56.10	192.168.56.30	TCP	79	55804 → 8080 [PSH, ACK] Seq=1 Ack=1 Win=502 Len=11 TSval=1686876927 …
19	0.120981180	192.168.56.10	192.168.56.30	TCP	78	55814 → 8080 [PSH, ACK] Seq=1 Ack=1 Win=502 Len=10 TSval=1686876927 …
20	0.121006381	192.168.56.30	192.168.56.10	TCP	68	8080 → 55672 [ACK] Seq=1 Ack=12 Win=508 Len=0 TSval=151394892 TSecr=…
21	0.121006546	192.168.56.30	192.168.56.10	TCP	68	8080 → 55680 [ACK] Seq=1 Ack=12 Win=508 Len=0 TSval=151394892 TSecr=…
22	0.121006586	192.168.56.30	192.168.56.10	TCP	68	8080 → 55692 [ACK] Seq=1 Ack=12 Win=508 Len=0 TSval=151394892 TSecr=…
23	0.121012543	192.168.56.10	192.168.56.30	TCP	79	55820 → 8080 [PSH, ACK] Seq=1 Ack=1 Win=502 Len=11 TSval=1686876927 …

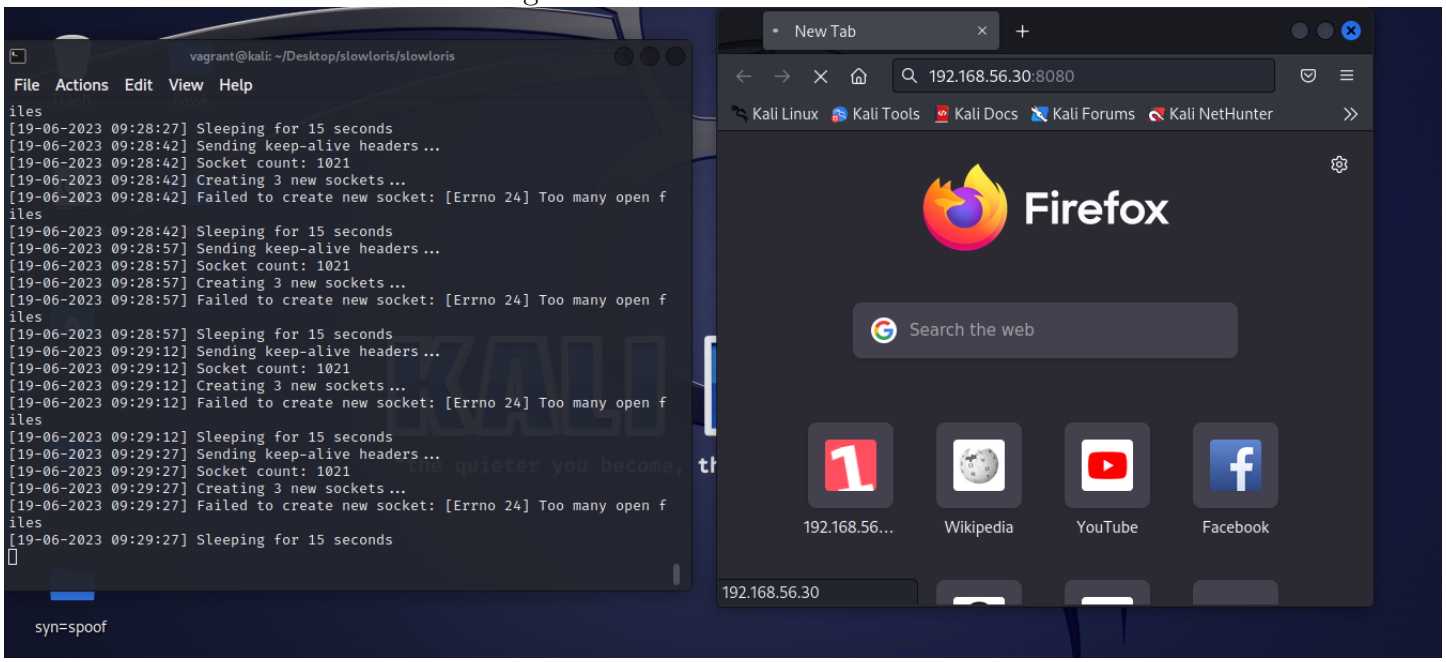
[1]https://github.com/gkbrk/slowloris

Figure 5: Slowloris PCAP End



10311	60.386448945	192.168.56.10	192.168.56.30	TCP	79	45752 → 8080 [PSH, ACK] Seq=43 Ack=1 Win=502 Len=11 TSval=16869371
10312	60.386455124	192.168.56.30	192.168.56.10	TCP	68	8080 → 45644 [ACK] Seq=1 Ack=54 Win=508 Len=0 TSval=151455157 TSec
10313	60.386455226	192.168.56.30	192.168.56.10	TCP	68	8080 → 45656 [ACK] Seq=1 Ack=52 Win=508 Len=0 TSval=151455157 TSec
10314	60.386455270	192.168.56.30	192.168.56.10	TCP	68	8080 → 45666 [ACK] Seq=1 Ack=56 Win=508 Len=0 TSval=151455157 TSec
10315	60.386476569	192.168.56.10	192.168.56.30	TCP	78	45766 → 8080 [PSH, ACK] Seq=45 Ack=1 Win=502 Len=10 TSval=16869371
10316	60.386518518	192.168.56.10	192.168.56.30	TCP	79	45774 → 8080 [PSH, ACK] Seq=44 Ack=1 Win=502 Len=11 TSval=16869371
10317	60.386533718	192.168.56.30	192.168.56.10	TCP	68	8080 → 45674 [ACK] Seq=1 Ack=56 Win=508 Len=0 TSval=151455157 TSec
10318	60.386533793	192.168.56.30	192.168.56.10	TCP	68	8080 → 45684 [ACK] Seq=1 Ack=53 Win=508 Len=0 TSval=151455157 TSec
10319	60.386533834	192.168.56.30	192.168.56.10	TCP	68	8080 → 45692 [ACK] Seq=1 Ack=55 Win=508 Len=0 TSval=151455157 TSec
10320	60.386555208	192.168.56.10	192.168.56.30	TCP	79	45778 → 8080 [PSH, ACK] Seq=43 Ack=1 Win=502 Len=11 TSval=16869371
10321	60.386594963	192.168.56.30	192.168.56.10	TCP	68	8080 → 45706 [ACK] Seq=1 Ack=54 Win=508 Len=0 TSval=151455157 TSec
10322	60.386595041	192.168.56.30	192.168.56.10	TCP	68	8080 → 45718 [ACK] Seq=1 Ack=56 Win=508 Len=0 TSval=151455157 TSec
10323	60.386595082	192.168.56.30	192.168.56.10	TCP	68	8080 → 45728 [ACK] Seq=1 Ack=55 Win=508 Len=0 TSval=151455157 TSec
10324	60.386595125	192.168.56.30	192.168.56.10	TCP	68	8080 → 45738 [ACK] Seq=1 Ack=55 Win=508 Len=0 TSval=151455157 TSec
10325	60.386595166	192.168.56.30	192.168.56.10	TCP	68	8080 → 45750 [ACK] Seq=1 Ack=56 Win=508 Len=0 TSval=151455157 TSec
10326	60.386718039	192.168.56.30	192.168.56.10	TCP	68	8080 → 45752 [ACK] Seq=1 Ack=54 Win=508 Len=0 TSval=151455157 TSec
10327	60.386718109	192.168.56.30	192.168.56.10	TCP	68	8080 → 45766 [ACK] Seq=1 Ack=55 Win=508 Len=0 TSval=151455157 TSec
10328	60.386718151	192.168.56.30	192.168.56.10	TCP	68	8080 → 45774 [ACK] Seq=1 Ack=55 Win=508 Len=0 TSval=151455157 TSec
10329	60.386718199	192.168.56.30	192.168.56.10	TCP	68	8080 → 45778 [ACK] Seq=1 Ack=54 Win=508 Len=0 TSval=151455157 TSec
10330	61.038304424	192.168.56.1	192.168.56.255	UDP	307	54915 → 54915 Len=263
10331	62.035751544	192.168.56.1	192.168.56.255	UDP	307	54915 → 54915 Len=263
10332	63.026708854	192.168.56.1	192.168.56.255	UDP	307	54915 → 54915 Len=263
10333	63.234567175	192.168.56.10	192.168.56.30	TCP	76	[TCP Retransmission] [TCP Port numbers reused] 55878 → 8080 [SYN]

Figure 6: Slowloris Attack Result

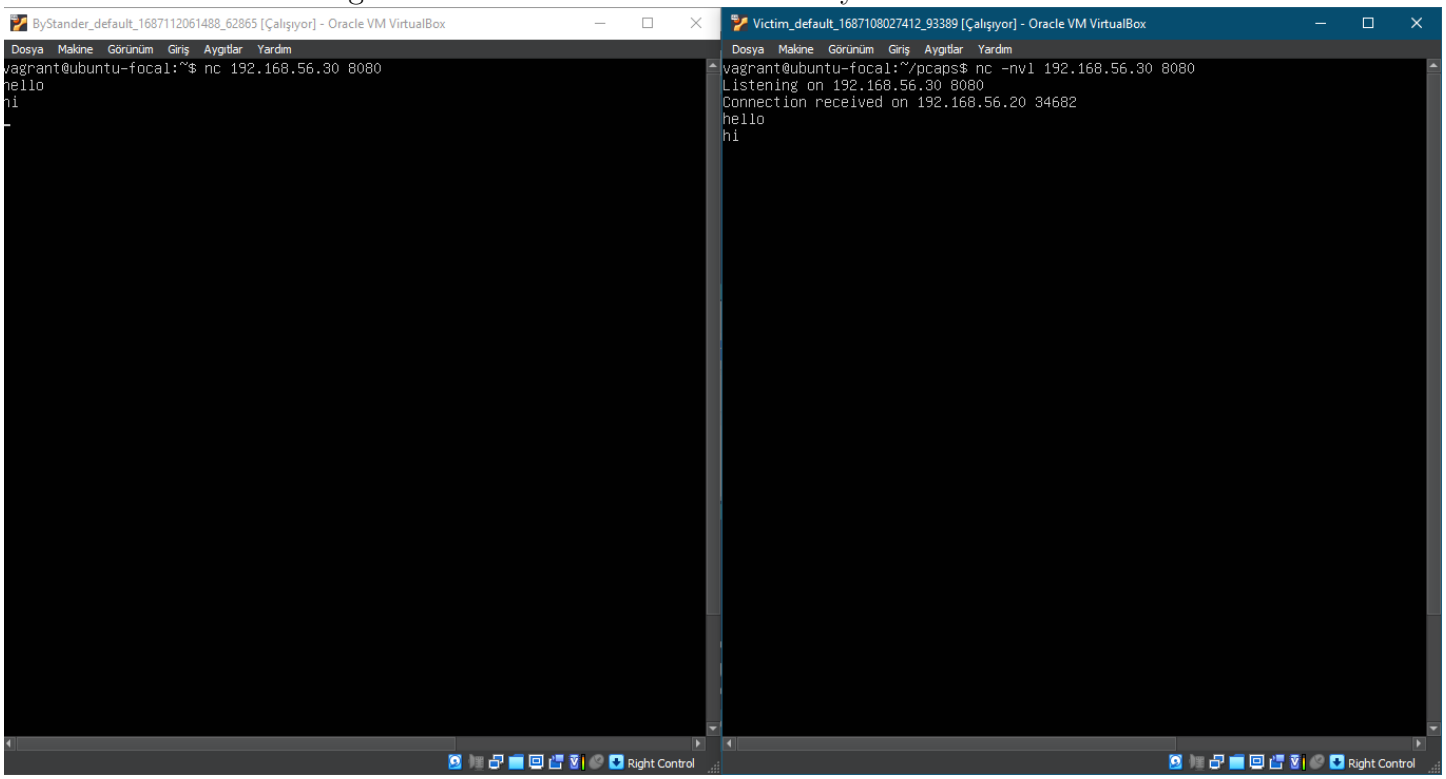


3 SYN Flooding

SYN Flood is a Denial of Service attack that aims to send lots of SYN requests to the victim and consume its threading pool. It exploits TCP 3-way handshake property by just sending SYN requests. The server waits for ACK while accepting all these SYN requests; however, we do not send ACK requests and fill the connection queue by continuing to send SYN requests. I've used netwox[2] tool to perform this attack.

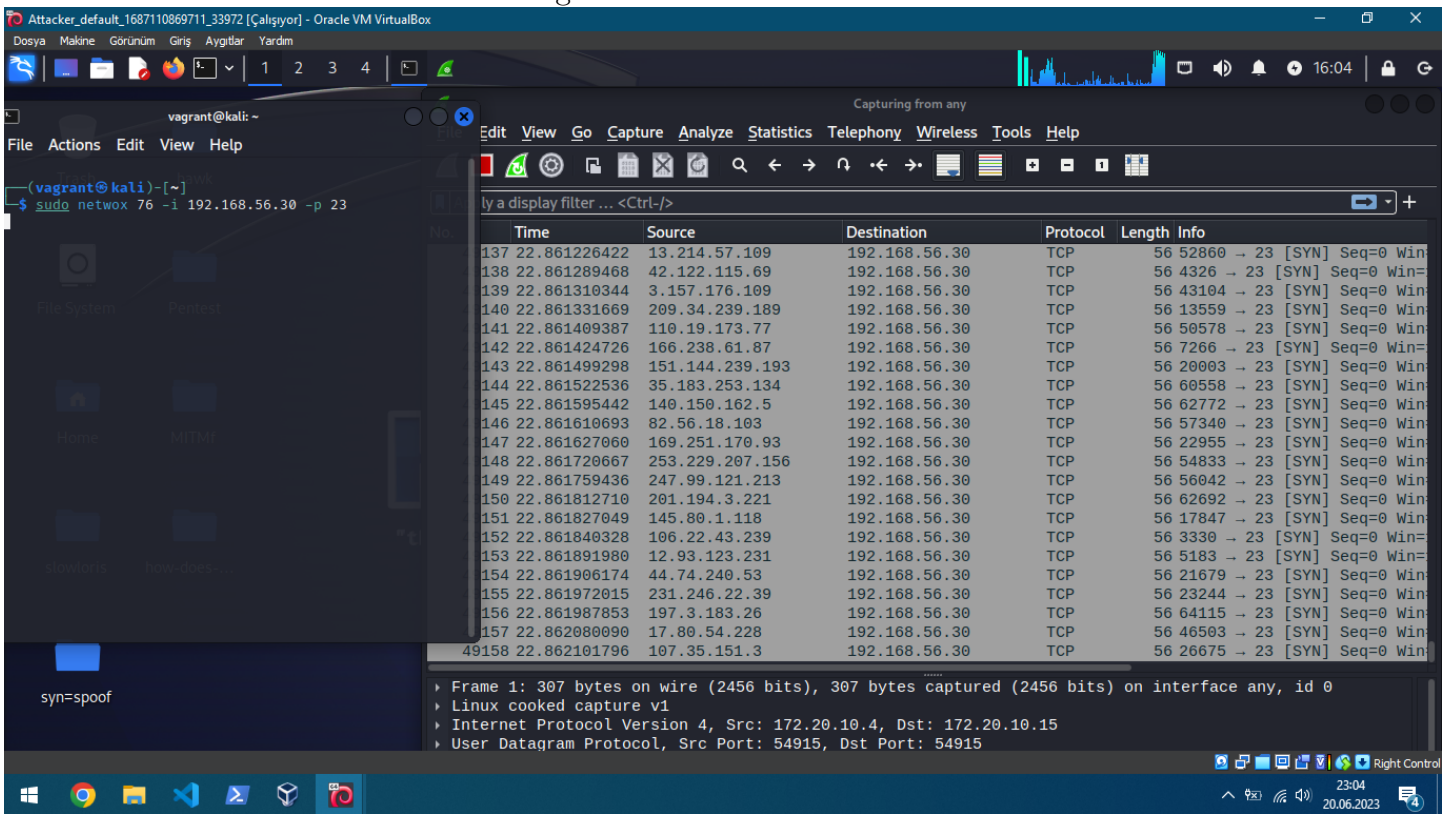
I've created TCP connection between ByStander and Victim before the attack to check whether I can create a TCP connection between two VMs.

Figure 7: TCP Connection between ByStander and Victim



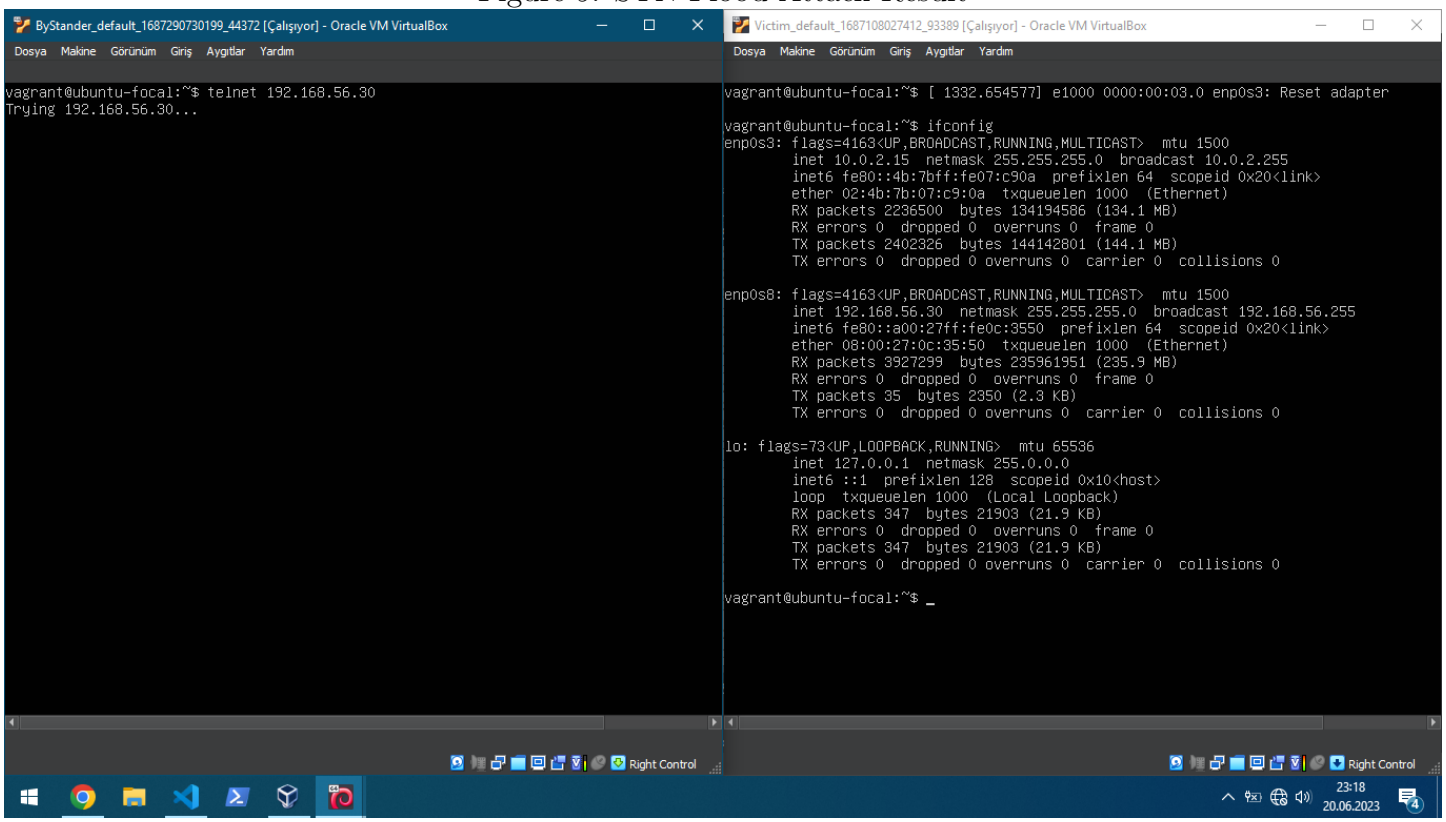
[2]https://linux.die.net/man/1/netwox

Figure 8: SYN Flood Attack



While attacking, the connection cannot be established as you can see in the following figure.

Figure 9: SYN Flood Attack Result



4 TCP Reset Attack

While trying to perform this attack, I've figured out that a VM cannot see packets between the other VMs. Therefore, I've done some research and I've configured network settings of the VMs in VirtualBox application. I've changed First Adapter of every VM to NAT network that I created. After that, the IP addresses of these VM's changed but changed to "192.168.56.*". They are still in private network and I allowed everything in the setting of Promiscuous Mode. After configuring, Attacker VM can see the packets between ByStander and Victim.

TCP Reset attack aims to shut down the connection between two victims. In this case ByStander is a victim too. Attacker spoofs a TCP packet in the network and sends TCP RST packet to one of the victims.

I've used netwox tool to perform this attack. 192.168.56.6 is Victim and 192.168.56.5 is ByStander. 192.168.56.4 is Attacker.

Figure 10: Connection between ByStander and Victim

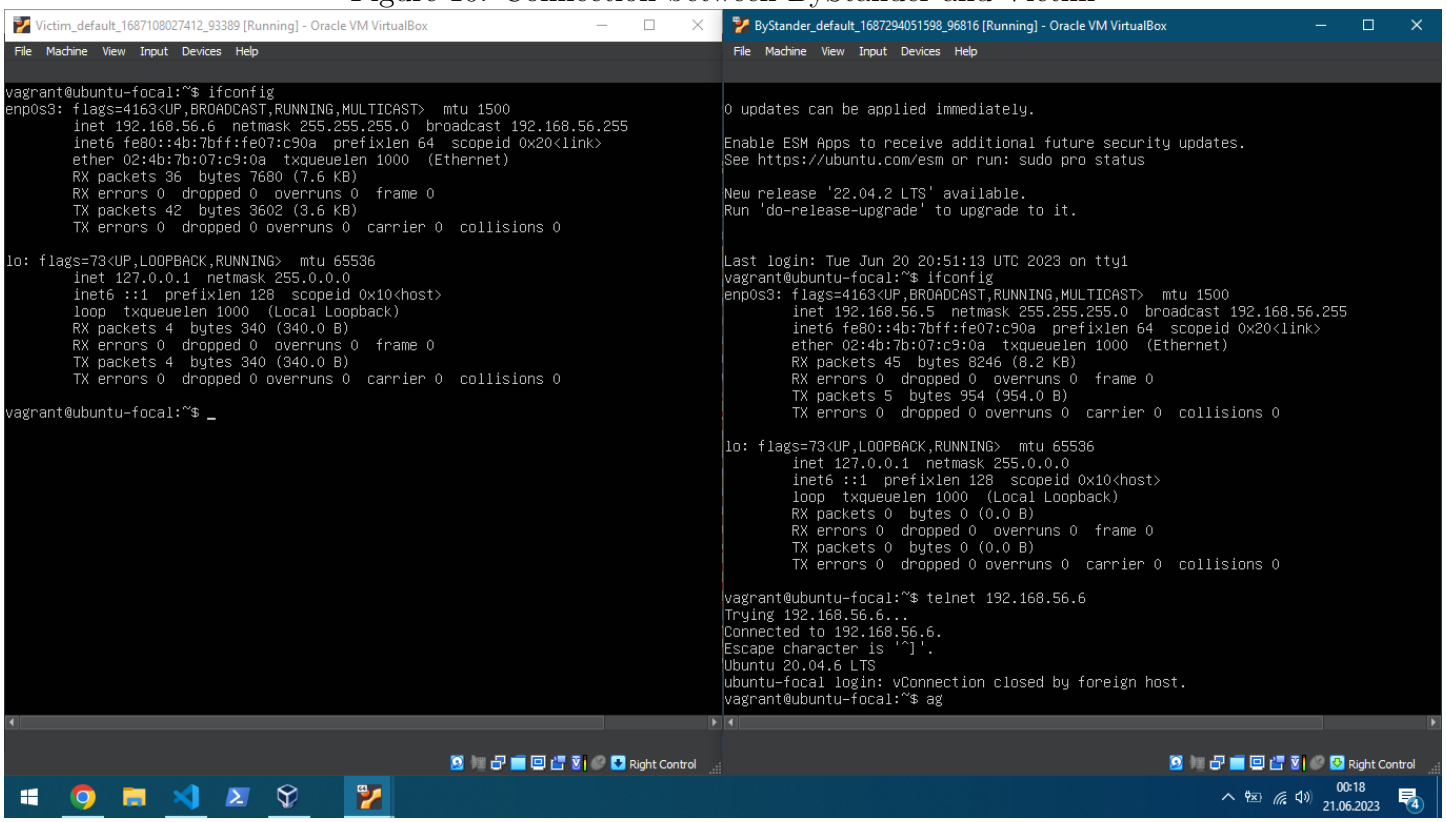
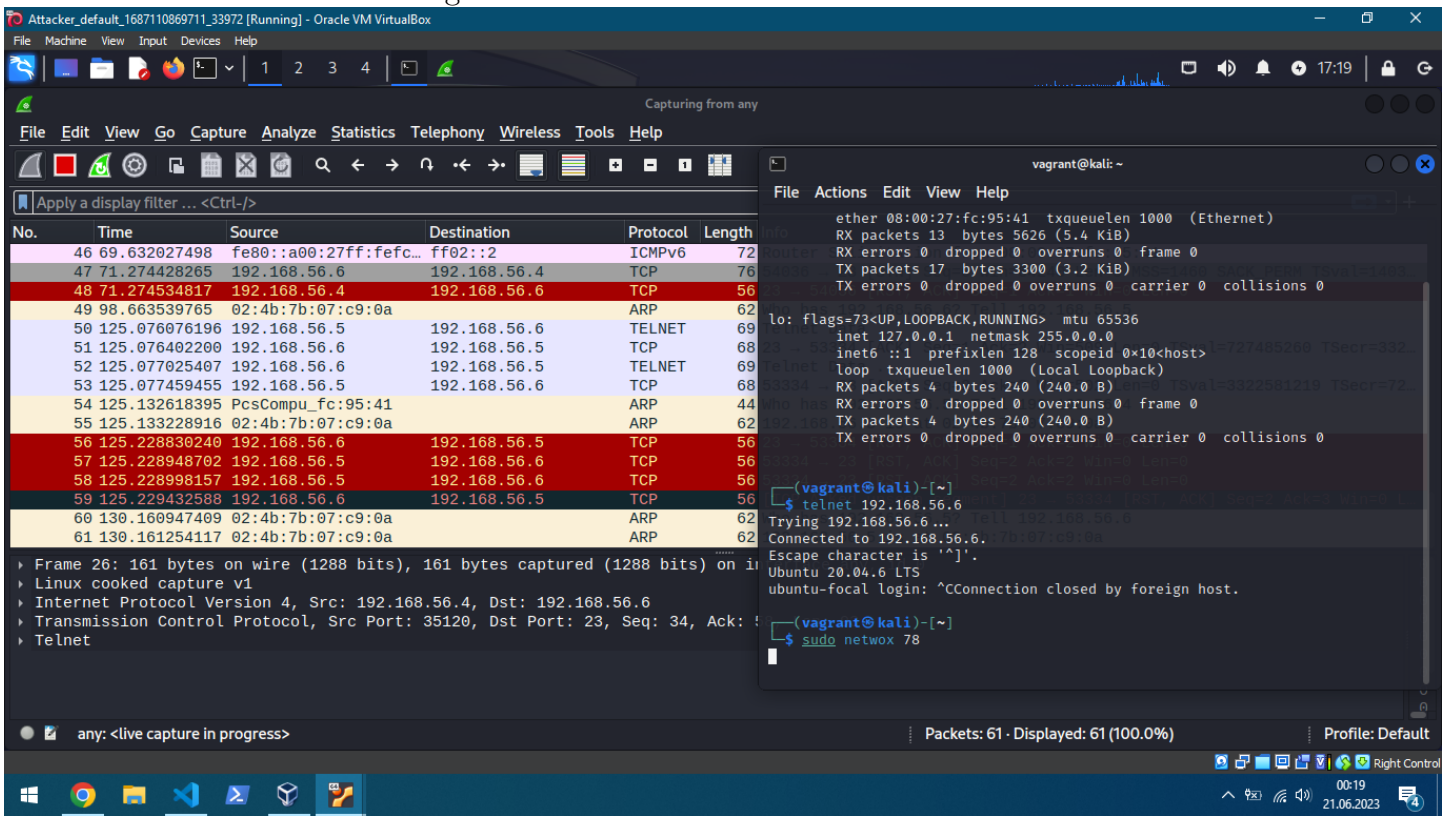


Figure 11: TCP Reset Command and PCAP



Attack started as 48th and 56th packets as Attacker (192.168.56.4) sends TCP RST packet to Victim(192.168.56.6) and TCP RST packet to ByStander(192.168.56.5) respectively.

Figure 12: PCAP of TCP Reset Attack