

Firewall

Lab Report 5

Information Systems Security course (01TYM, 02KRQ)

prepared by:

Anuar Elio Magliari (s317033@studenti.polito.it)

George Florin Eftime (s303483@studenti.polito.it)

Alekos Interrante Bonadia (s319849@studenti.polito.it)

Packet filter

- Which authorisation policy is configured by default on your machine (on each of the three chains)?
 - By default ACCEPT policy is configured as it is possible to see in the image below.

```
# iptables -L -v -n
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target    prot opt in     out     source                   destination

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target    prot opt in     out     source                   destination

Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target    prot opt in     out     source                   destination
```

- Which chain (out from INPUT, FORWARD, and OUTPUT) do you have to modify to protect your machine from connections originating from the external users?
 - INPUT and FORWARD chains need to be modified in order to have protection from external users.
- Write down the iptables command to modify the authorisation policy of Alice's host, so that to reject any traffic (hint: you need to modify the default policy for the INPUT chain from ACCEPT to DROP):

```

(root@geo)-[/home/george]
# iptables -P INPUT DROP

(root@geo)-[/home/george]
# iptables -L -v -n
Chain INPUT (policy DROP 0 packets, 0 bytes)
 pkts bytes target    prot opt in     out     source                   destination

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target    prot opt in     out     source                   destination

Chain OUTPUT (policy ACCEPT 300 packets, 55360 bytes)
 pkts bytes target    prot opt in     out     source                   destination

```

- Does Bob receive any responses (to the ping) from Alice's host?
 - No, he doesn't

```

(elion-man@Elion-Man-on-Kali)-[~]
$ ping 172.22.17.139
PING 172.22.17.139 (172.22.17.139) 56(84) bytes of data.
^C
— 172.22.17.139 ping statistics —
31 packets transmitted, 0 received, 100% packet loss, time 30701ms

```

- Can Bob connect to Alice's host via SSH and HTTP (with the browser)?
 - No, he doesn't. In both cases, Bob doesn't receive any response from Alice because Alice's firewall has the INPUT chain set to DROP, so it discards all requests from the external source.
- Check with nmap (running the above indicated nmap command on Bob's host) the status of the ports 22 and 80 on Alice's host. What is their status now?
 - Using nmap the status of the ports 22 and 80 are set as "filtered" (details in the next image).

```

# nmap -sT -Pn -n -p 80,22 -v 172.22.17.139
Starting Nmap 7.94 ( https://nmap.org ) at 2023-12-05 12:24 CET
Initiating Connect Scan at 12:24
Scanning 172.22.17.139 [2 ports]
Completed Connect Scan at 12:24, 3.00s elapsed (2 total ports)
Nmap scan report for 172.22.17.139
Host is up.

PORT      STATE      SERVICE
22/tcp    filtered  ssh
80/tcp    filtered  http

Read data files from: /usr/bin/./share/nmap
Nmap done: 1 IP address (1 host up) scanned in 3.04 seconds

```

- Write down the iptables command to add a rule to the authorization policy on Alice's host (for the input traffic), so that to enable all ICMP traffic (for simplicity, we provide you some of the parameters of the command)
 - iptables -A INPUT -p icmp -j ACCEPT

```

(root@geo)-[/var/www/html]
# iptables -L -v -n
Chain INPUT (policy DROP 0 packets, 0 bytes)
 pkts bytes target    prot opt in     out     source           destination
    6   504 ACCEPT     1    --  *      *       172.22.16.108    0.0.0.0/0

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target    prot opt in     out     source           destination

Chain OUTPUT (policy ACCEPT 1292 packets, 547K bytes)
 pkts bytes target    prot opt in     out     source           destination

```

- Does Bob receive this time any response from Alice's host in response to the ping command?
 - Yes, now he does because Alice allows all the icmp packets in input.

```

(Elion-man@Elion-Man-on-Kali)-[~]
$ ping 172.22.17.139
PING 172.22.17.139 (172.22.17.139) 56(84) bytes of data.
64 bytes from 172.22.17.139: icmp_seq=1 ttl=64 time=7.40 ms
64 bytes from 172.22.17.139: icmp_seq=2 ttl=64 time=7.32 ms
64 bytes from 172.22.17.139: icmp_seq=3 ttl=64 time=5.06 ms
64 bytes from 172.22.17.139: icmp_seq=4 ttl=64 time=4.27 ms
64 bytes from 172.22.17.139: icmp_seq=5 ttl=64 time=17.3 ms
64 bytes from 172.22.17.139: icmp_seq=6 ttl=64 time=19.5 ms
64 bytes from 172.22.17.139: icmp_seq=7 ttl=64 time=30.6 ms
64 bytes from 172.22.17.139: icmp_seq=8 ttl=64 time=7.02 ms
^C
— 172.22.17.139 ping statistics —
8 packets transmitted, 8 received, 0% packet loss, time 7013ms
rtt min/avg/max/mdev = 4.271/12.311/30.586/8.699 ms

```

- Next, on Alice's host, write down the iptables command to allow the TCP input traffic towards the port 80 (on Alice):
 - `iptables -A INPUT -p tcp --dport 80 -j ACCEPT`
- Check out the configuration of IPtables on Alice's host. Which chain has been modified?
 - The chain modified is the INPUT chain as specified in the command

```

(root@geo)-[/var/www/html]
# ./script_firewall.sh
Chain INPUT (policy DROP 0 packets, 0 bytes)
  pkts bytes target    prot opt in     out     source               destination
    0    0 ACCEPT    1    --  *      *        0.0.0.0/0            0.0.0.0/0
    0    0 ACCEPT    6    --  *      *        0.0.0.0/0            0.0.0.0/0            tcp dpt:80

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
  pkts bytes target    prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 1544 packets, 609K bytes)
  pkts bytes target    prot opt in     out     source               destination

```

- Does Chuck receive any response (to the ping) from the Alice's host? Why?
 - Yes, Chuck receives ping responses from Alice because she doesn't drop the icmp requests (source is set to anyone).
- Check again with nmap (running on Bob's host) the status of the ports 22 and 80 on Alice's host. What is their status?

```

(kali㉿kali)-[~]
└─$ nmap -sT -Pn -n -p 80,22 -v 172.22.17.186
Starting Nmap 7.94 ( https://nmap.org ) at 2023-12-05 15:35 UTC
Initiating Connect Scan at 15:35
Scanning 172.22.17.186 [2 ports]
Discovered open port 80/tcp on 172.22.17.186
Completed Connect Scan at 15:35, 1.61s elapsed (2 total ports)
Nmap scan report for 172.22.17.186
Host is up (0.061s latency).

PORT      STATE      SERVICE
22/tcp    filtered  ssh
80/tcp    open       http

Read data files from: /usr/bin/../../share/nmap
Nmap done: 1 IP address (1 host up) scanned in 1.66 seconds

```

- How can nmap distinguish between filtered ports and closed ports? Verify by analysing the traffic exchanged between the two machines (e.g. with wireshark).
 - Nmap distinguishes between these two states based on whether it receives a response to its probe. In the case of a closed port, it receives a TCP RST packet in response, whereas for a filtered port, it doesn't receive any packet.
- What is the status of the port 80 (as listed by the above nmap command)?
 - The status of the port 80 is set to filtered.

```

└─$ nmap -sT -Pn -n -p 80 -v 172.22.17.170
Starting Nmap 7.94SVN ( https://nmap.org ) at 2023-12-12 09:41 CET
Initiating Connect Scan at 09:41
Scanning 172.22.17.170 [1 port]
Completed Connect Scan at 09:41, 2.00s elapsed (1 total ports)
Nmap scan report for 172.22.17.170
Host is up.

PORT      STATE      SERVICE
80/tcp    filtered  http

Read data files from: /usr/bin/../../share/nmap
Nmap done: 1 IP address (1 host up) scanned in 2.04 seconds

```

- What is now the status of the port 80 on Bob (as listed by the above nmap command on Alice's host)?
 - The status of the port 80 is still set to filtered.

```

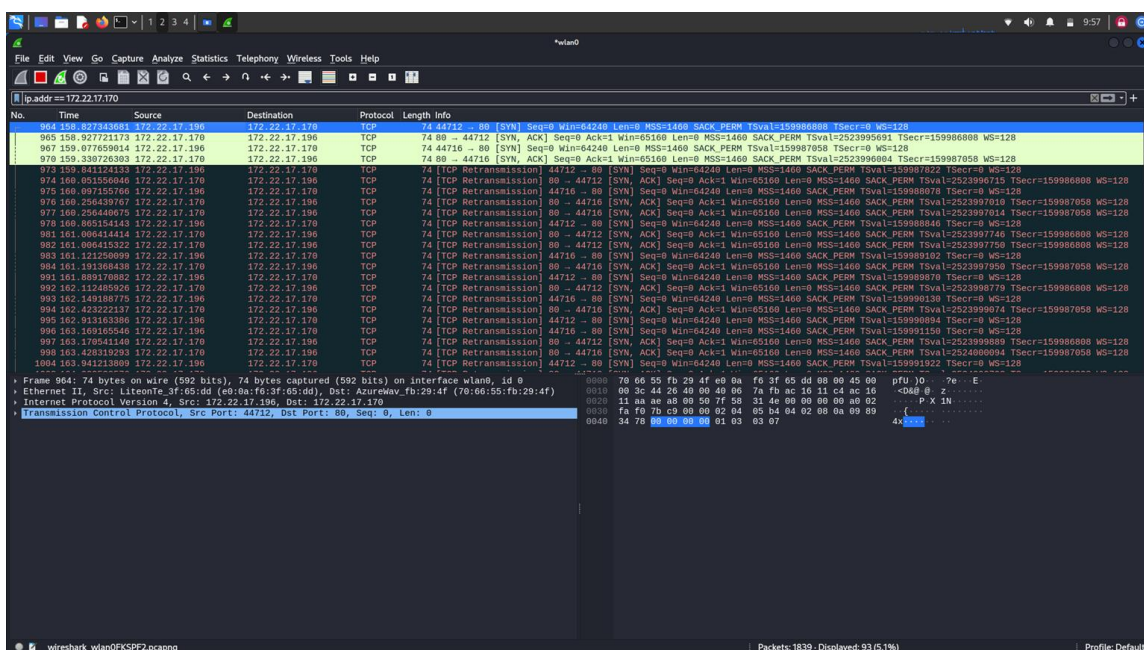
$ nmap -sT -Pn -n -p 80 -v 172.22.17.170
Starting Nmap 7.94SVN ( https://nmap.org ) at 2023-12-12 09:42 CET
Initiating Connect Scan at 09:42
Scanning 172.22.17.170 [1 port]
Completed Connect Scan at 09:42, 2.00s elapsed (1 total ports)
Nmap scan report for 172.22.17.170
Host is up.

PORT      STATE      SERVICE
80/tcp    filtered  http

Read data files from: /usr/bin/./share/nmap
Nmap done: 1 IP address (1 host up) scanned in 2.04 seconds

```

- Can Alice browse web content on Bob's host? Verify by analysing the traffic among the two machines (with Wireshark). After analyzing the traffic with Wireshark, describe what is happening with the SYN, SYN ACK, and ACK messages exchanged between Alice and Bob when Alice tries to browse web content on Bob's host.
 - Alice is unable to browse the web content on Bob's host because she sends a SYN packet to Bob, and Bob responds with a SYN-ACK packet. However, Alice drops the response because the policies specify allowing packets from the destination port equal to 80, whereas it is not allowed to accept packets with the source port equal to 80. Consequently, Alice retransmits the SYN packet because, from her perspective, she doesn't receive the response (SYN-ACK). When the timeout expires, she retries to send the SYN packet.



- Before passing to the execution of the next exercise, restore on Alice's host the authorisation policy of type "ACCEPT ALL". Hint: You need to delete all the current rules and specify afterwards the rules for the input chain.
 - `sudo iptables -F INPUT`
 - `sudo iptables -P INPUT ACCEPT`

```
(elion-man@Elion-Man-on-Kali)-[~/Desktop/LAB_IIS/LAB_5]
$ sudo iptables -L -v -n

Chain INPUT (policy ACCEPT 996 packets, 619K bytes)
 pkts bytes target    prot opt in     out     source    destination
Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target    prot opt in     out     source    destination
Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target    prot opt in     out     source    destination
```

Packet filter stateless

- Can Alice browse the web server of Bob?
 - No, with this settings in the Frank's iptables, she cannot browse the web server of Bob.
- On which chain operate the above commands (on Frank)?
 - Frank is operating on the FORWARD chain acting similar as a router between Alice and Bob.
- At this point, can Alice connect to the web server running on Bob's host?
 - Now yes she can because for tcp protocol there is a rule that enables the forwarding through Frank.
- Which is the purpose of the last rule?
 - The purpose of the last rule is to allow the forwarding of the tcp segments when the source is Alice and the destination port is equal to 80, and the same thing when the destination is Alice with source port equal to 80.

- Can Alice connect to the web server running on another machine (let's say Deborah), after you have configured Deborah to make her traffic pass through Frank's host? Why?
 - Yes, she can. Because, with this rule enables all the tcp traffic for Alice through Frank (it's not specified Bob, Deborah or someone else) allowing her to browse http contents because it operates over tcp with source/destination port equal to 80.
- Start the Apache web server on Alice (if not already started). Can Bob connect with his browser to the web server running on Alice's host?
 - No, he cannot. Because, as said in the previous answer, Frank enables the traffic for Alice but not for Bob.
- Can Bob download the web page from Alice's host?
 - No, he cannot (the same reason explained in the previous answer).
- Can Alice and Bob connect to the corresponding SSH servers of the counterparts, that is can Alice connect to SSH server of Bob and viceversa? (hint: think what happens if Bob uses the port 80 as source port)
 - No, she cannot, because ssh operates with source/destination port equal to 22, the traffic through port 22 it will be dropped by Frank as specified in the rules chain FORWARD (policy = default-deny).
- Write down (and execute) the iptables command which modifies the policy on Frank's host, so that to enable only the web connections from Alice towards any external user (any IP address):
 - `iptables -I FORWARD 2 -p tcp -d IP Alice --sport 80 --syn -j DROP`

```
# iptables -L -v -n
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
pkts bytes target      prot opt in     out     source            destination

Chain FORWARD (policy DROP 0 packets, 0 bytes)
pkts bytes target      prot opt in     out     source            destination
 6   264 DROP        6    -- *     *       0.0.0.0/0         172.20.10.13      tcp spt:80 flags:0x17/0x02
 9   540 ACCEPT     6    -- *     *       172.20.10.13      0.0.0.0/0         tcp dpt:80
 7   280 ACCEPT     6    -- *     *       0.0.0.0/0         172.20.10.13      tcp spt:80

Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
pkts bytes target      prot opt in     out     source            destination
```

- Have you been successful in running the commands above?

- No, I haven't.

Public service

In this section, we are expected to apply the previous policy incrementally, step by step, in iptables. Although it was not specified to clean the iptables, the rules will not conflict with each other anyway.

- Can Bob connect to Alice's host via SSH?
 - No, he cannot because he tries to access to a service on port 22 but it is not allowed by the forward policy in Frank's iptables.
- Write down the (complete) iptables commands to modify the authorisation policy on Frank's host, so that to forward packets belonging to SSH connections towards Alice's host coming from any IP address:
 - `iptables -A FORWARD -p tcp -d 172.20.10.5 --dport 22 -j ACCEPT`
 - `iptables -A FORWARD -p tcp -s 172.20.10.5 --sport 22 --syn -j DROP`
 - `iptables -A FORWARD -p tcp -s 172.20.10.5 --sport 22 -j ACCEPT`

```
(root@kali)-[/home/kali]
# iptables -L -v -n
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
pkts bytes target      prot opt in      out     source            destination
Chain FORWARD (policy DROP 0 packets, 0 bytes)
pkts bytes target      prot opt in      out     source            destination      tcp dpt:80
0 0 ACCEPT    6  --  *      *      172.20.10.5      0.0.0.0/0
0 0 DROP      6  --  *      *      0.0.0.0/0        172.20.10.5      tcp spt:80 flags:0x17/0x02
0 0 ACCEPT    6  --  *      *      0.0.0.0/0        172.20.10.5      tcp spt:80
0 0 ACCEPT    6  --  *      *      0.0.0.0/0        172.20.10.5      tcp dpt:22
0 0 DROP      6  --  *      *      172.20.10.5      0.0.0.0/0        tcp spt:22 flags:0x17/0x02
0 0 ACCEPT    6  --  *      *      172.20.10.5      0.0.0.0/0        tcp spt:22
Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
pkts bytes target      prot opt in      out     source            destination
```

- At this step, can Bob connect to Alice's host via SSH?
 - Now, yes he can. It is possible because in Frank's enabled the forward policy for the services on port 22 (usually SSH port).
- At this step, can Alice connect to Bob's host via SSH?

- No, she cannot because her packets are dropped by Frank. The destination address specified is different from Alice's (in this case with Bob), causing them to be automatically rejected.

ICMP traffic

In this section, we are expected to apply the previous policy incrementally, step by step, in iptables. Although it was not specified to clean the iptables, the rules will not conflict with each other anyway.

- Try to ping Alice's host from Bob. Are you successful? Why?
 - Bob cannot ping Alice because ICMP is not included in the rule chains of the FORWARD in Frank. By default, all traffic that is not explicitly specified is rejected (default-deny, the policy of FORWARD is set to DROP).

The modification needed are written below as commands:

- `iptables -A FORWARD -p icmp -s 172.20.10.5 --icmp-type echo-request -j ACCEPT`
- `iptables -A FORWARD -p icmp -d 172.20.10.5 --icmp-type echo-reply -j ACCEPT`
- `iptables -A FORWARD -p icmp -d 172.20.10.5 --icmp-type echo-request -j ACCEPT`
- `iptables -A FORWARD -p icmp -s 172.20.10.5 --icmp-type echo-reply -j ACCEPT`

```
(root@kali)-[/home/kali]
# iptables -L -v -n
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
pkts bytes target      prot opt in     out     source               destination

Chain FORWARD (policy DROP 0 packets, 0 bytes)
pkts bytes target      prot opt in     out     source               destination
 0      0 ACCEPT      6    -- *     *       172.20.10.5          0.0.0.0/0          tcp dpt:80
 0      0 DROP        6    -- *     *       0.0.0.0/0            172.20.10.5         tcp spt:80 flags:0x17/0x02
 0      0 ACCEPT      6    -- *     *       0.0.0.0/0            172.20.10.5         tcp spt:80
 0      0 ACCEPT      6    -- *     *       0.0.0.0/0            172.20.10.5         tcp dpt:22
 0      0 DROP        6    -- *     *       172.20.10.5          0.0.0.0/0          tcp spt:22 flags:0x17/0x02
 0      0 ACCEPT      6    -- *     *       172.20.10.5          0.0.0.0/0          tcp spt:22
 7    588 ACCEPT      1    -- *     *       172.20.10.5          0.0.0.0/0          icmp type 8
 7    588 ACCEPT      1    -- *     *       0.0.0.0/0            172.20.10.5         icmp type 0
123 10332 ACCEPT      1    -- *     *       0.0.0.0/0            172.20.10.5         icmp type 8
120 10080 ACCEPT      1    -- *     *       172.20.10.5          0.0.0.0/0          icmp type 0

Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
pkts bytes target      prot opt in     out     source               destination
```

- Now try again to ping Alice's host from Bob. Are you successful?
 - Yes, I can because now the policies for ICMP are enabled (icmp type= 0 for echo reply and icmp type=8 for echo request).
- Why is it dangerous to enable ICMP traffic without any restriction? Hint: execute `iptables -p icmp -h` for a list of types of ICMP messages.
 - Enabling ICMP traffic without any restrictions has made internal hosts vulnerable to various attacks, such as Denial of Service (DoS) and network scanning. Some attacks may leverage ICMP to overload another host (ping bombing), thereby disabling all its services. Alternatively, ICMP can allow attackers to scan the network, providing them with information such as network topology, open ports on a host, and so on.
- Suppose that Frank must filter the traffic toward the subnet 1.2.3.0/24 instead of filtering just the traffic directed to Alice's host. What type of attack could be performed if Frank receives an ICMP echo request packet with destination IP address 1.2.3.255 and Frank is configured to enable the ICMP traffic (as mentioned above)?
 - In this scenario, given that the address is a broadcast address, all hosts inside the network will receive the echo request, and each one will respond to the request, either to a specific host or to the network (broadcast address). Consequently, they will collectively amplify the initial packet.

Bandwidth limitation

In this section, we are expected to apply the previous policy incrementally, step by step, in iptables. Although it was not specified to clean the iptables, the rules will not conflict with each other anyway (this is for http and ssh).

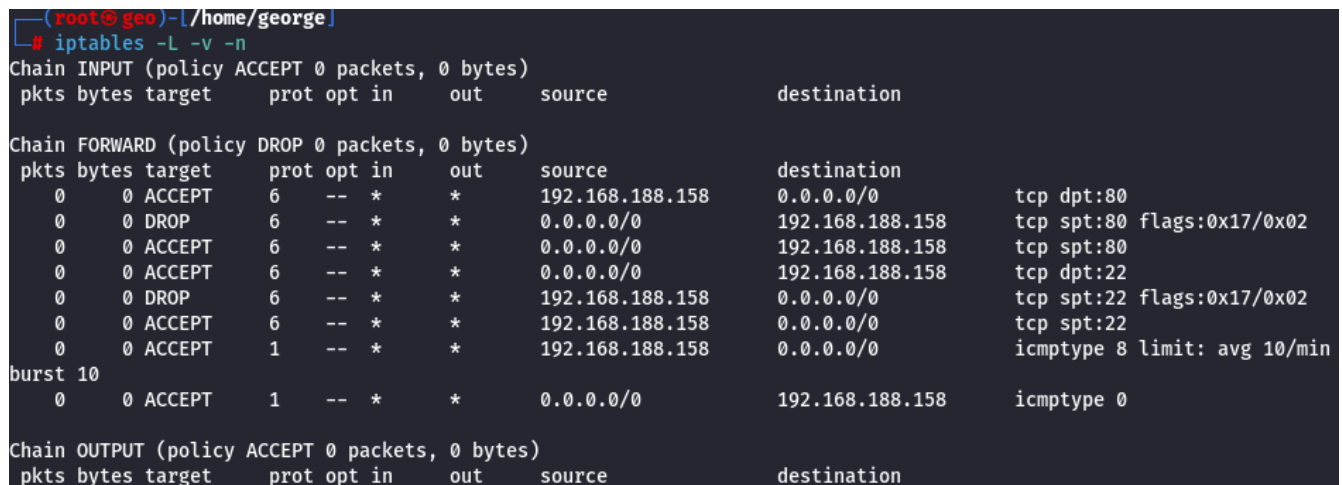
- What is the effect of replacing the rule:

```
iptables -A FORWARD -p icmp -d IP Alice --icmp-type echo-request -j ACCEPT
```

- with the following rule?

```
iptables -A FORWARD -p icmp -d IP Alice --icmp-type echo-request -m limit --limit 20/minute --limit-burst 1 -j ACCEPT
```

- The new rule introduces rate limiting using the 'limit' module, allowing a maximum of 20 ICMP echo requests per minute. The 'limit-burst 1' parameter permits a burst of one additional ICMP echo request beyond the specified rate limit. This implies that in the event of a brief burst of more than 20 requests within a minute, only one extra request will be accepted, and the others will be subject to the rate limit.
- Illustrate with a screenshot the current configuration of the packet filter on Frank



```
(root@geo)-[/home/george]
# iptables -L -v -n
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
pkts bytes target      prot opt in      out     source            destination
Chain FORWARD (policy DROP 0 packets, 0 bytes)
pkts bytes target      prot opt in      out     source            destination      tcp dpt:80
0      0 ACCEPT      6  --  *      *      192.168.188.158    0.0.0.0/0
0      0 DROP       6  --  *      *      0.0.0.0/0          192.168.188.158  tcp spt:80 flags:0x17/0x02
0      0 ACCEPT      6  --  *      *      0.0.0.0/0          192.168.188.158  tcp spt:80
0      0 ACCEPT      6  --  *      *      0.0.0.0/0          192.168.188.158  tcp dpt:22
0      0 DROP       6  --  *      *      192.168.188.158    0.0.0.0/0        tcp spt:22 flags:0x17/0x02
0      0 ACCEPT      6  --  *      *      192.168.188.158    0.0.0.0/0        tcp spt:22
0      0 ACCEPT      1  --  *      *      192.168.188.158    0.0.0.0/0        icmp type 8 limit: avg 10/min
burst 10
0      0 ACCEPT      1  --  *      *      0.0.0.0/0          192.168.188.158  icmp type 0
Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
pkts bytes target      prot opt in      out     source            destination
```

- What do you note? Analyze the captured ICMP packets on Frank and Bob with Wireshark.
 - Analyzing the traffic on Frank, we observe that Alice sends numerous packets to Bob. However, Frank rejects some packets based on the imposed limit, allowing only a portion of them to pass through. Consequently, Bob receives

only the packets that are not filtered by Frank, and Alice receives responses only for the packets that successfully reach Bob.

- Against what kind of attack is protected Bob (by configuring as above Frank)?
 - Bob will be protected from DOS attacks, as the number of packets is limited. Therefore, Bob will receive fewer packets, helping to prevent overloading.

Stateful packet filter

In this section, we are assumed to clear all the rules in iptables and apply only the rules specified in this section (only FORWARD DROP has been set).

- Start the Apache web server on Bob. Next, connect from Alice to Bob's web server (with a web browser). Are you successful?
 - No, she cannot. Because Frank has disabled all the HTTP traffic.
- Start the Apache web server on Bob. Next, connect from Alice to Bob's web server (with a web browser). Are you successful? Why?
 - Yes, she can. The second rule forwards HTTP traffic from Alice to another web server (Bob), as specified by the source address and destination port, enabling Alice to send the first packet to establish the connection. Simultaneously, the third rule permits the passage of all traffic that has been established and is related to the traffic initiated by Alice. Therefore, the rule allows the firewall to automatically accept packets associated with established connections without having to re-examine each individual packet against the basic filtering rules.
- Are you successful? Why?
 - No, he cannot. The rule allows only HTTP traffic for Alice, while Bob is attempting to send packets with a source port of 80 and a destination port of 22. However, this is not specified in the rule chains (default-deny).

```
(root@geo)-[/home/george]
# iptables -L -v -n
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target    prot opt in     out     source    destination
Chain FORWARD (policy DROP 0 packets, 0 bytes)
 pkts bytes target    prot opt in     out     source    destination    tcp dpt:80
 0      0 ACCEPT    6    --  *      *       192.168.188.158  0.0.0.0/0
 0      0 ACCEPT    0    --  *      *       0.0.0.0/0       0.0.0.0/0      state RELATED,ESTABLISHED
Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target    prot opt in     out     source    destination
```