Isfahan University of Technology
Network Security
Dr. Manshaei
Homework 1: Module 2 (Firewall)
Deadline: Sunday 7th Aban, at 23 : 59

We only accept the homework **delivered via Yekta (yekta.iut.ac.ir), before the deadline**. If you have any question or concerns about this homework, feel free to contact Mr. Alireza Katani via Telegram: @*Arewk* (Preferred) or email: alreka1379@gmail.com.

Theoretical Questions

Q1: What type of actions can we define in filtering table for firewalls? Briefly explain the function of each action?

Q2: What is the output of nmap for different states of the filtering table?

Q3: Explain the masquerading action in the filtering table? What sort of problems masquerading can solve for us?

Q4: How does port forwarding work? And what problems does it solve for us?

Practical Questions

Q5: As we discussed during lectures, "iptables" is a powerful command-line utility in Linux used for configuring and managing packet filtering rules within the kernel's netfilter framework. This will act as a firewall for the system. It allows administrators to define rules for packet filtering, given source or destination IP addresses, ports, and protocols. We have prepared two short video lectures, one for routing/iptables tables (the video is for a general introduction to firewall tables and order of applying the tables), and another for iptables command lines (In this video, we discuss how to write NAT tables and filtering rules). After watching these videos, write a rule for each of the following situations. Also specify in which chain and table it should be placed.

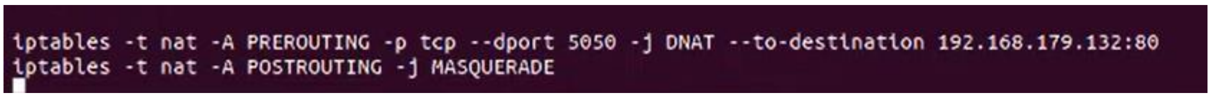
- Destination IP is the firewall IP, Source IP is in the range of 192.168.0.0/16, The protocol used is telent, Action = Drop, Establish a connection by Telnet (with cmd or hping3) and show our connection will not be established with a screen shot.
- Destination IP is the firewall IP, If the reception of SYN packets exceeds 20 packets per minute, it will refuse to accept more packets, For test use hping3 and make the situation and take screen shot.
- We want to use DNAT for port 8080 of our server, Translate Destination IP to 192.168.179.150:80, The connection should be established without any problems (for extra point test this rule with nginx server in your VM like Video).
- Use SNAT to translate source IP from 192.168.179.178 to 10.1.1.1, Interface eth0 (Test this rule with your local IPs and take a screenshot of the IP translation in Wireshark).
- The default mode for packets originating from the firewall is to drop packets.
- All packets destined for 192.168.179.120 and port 21 must be dropped. The source and destination of these packets will not be the firewall and will only pass through the firewall.

Isfahan University of Technology
Network Security
Dr. Manshaei
Homework 1: Module 2 (Firewall)
Deadline: Sunday 7^{th} Aban, at $23:59$

Bonus Exercise

As we addressed in the video, we assume that there exist 3 hosts with 3 IP addresses. Answer the question according to the given information and the video.

- Consider 192.168.179.1 as the main host (windows host).
- Consider 192.168.179.148 as a virtual host, run web server on port 443 (the server on which iptable rules are written), if a request for port 80 reaches this server the desired system redirects it to port 443.
- Consider 192.168.179.132 as another virtual host, we have web server on this IP with port 443 (192.168.179.132:443), if a request for port 80 reaches this server the desired system redirects it to port 443 (you can see the redirection request, *302 Error*, in the Tcpdump image). We write a configuration for port forwarding like Figure 1.

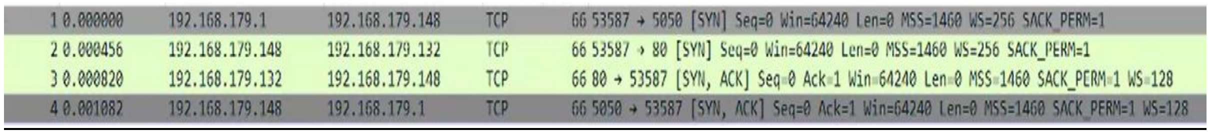


```
iptables -t nat -A PREROUTING -p tcp --dport 5050 -j DNAT --to-destination 192.168.179.132:80
iptables -t nat -A POSTROUTING -j MASQUERADE
```

Figure 1: IPTable rules

- **Question:** When we request the IP 192.168.179.148:5050 from the main Windows host, which web server do we see? Explain the reason and present a solution for this problem.
- **Help**: This question should be solved with the help of Wireshark and Tcpdump images. It is also suggested to implement for this question. In case of implementation, take a screen shot of what happened.

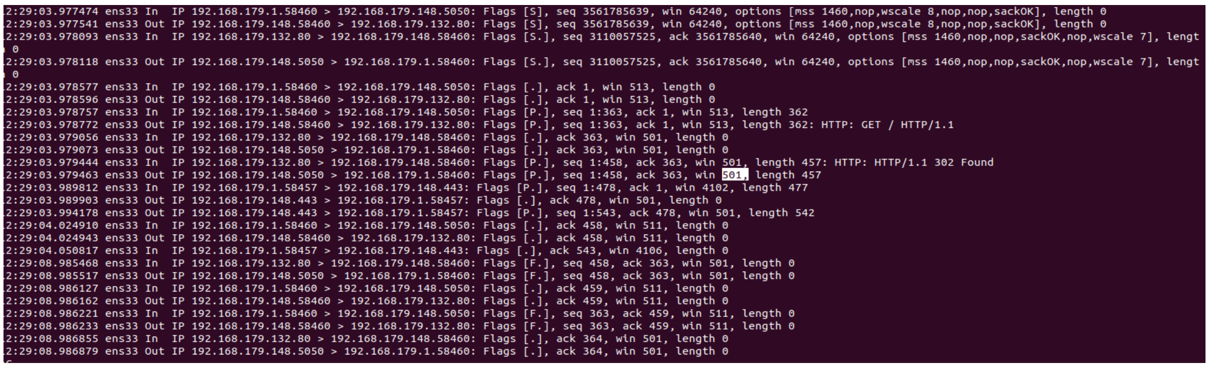
Figure 2 illustrates the Wireshark capture.



```
1 0.000000   192.168.179.1     192.168.179.148   TCP   66 53587 → 5050 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
2 0.000456   192.168.179.148   192.168.179.132   TCP   66 53587 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
3 0.000820   192.168.179.132   192.168.179.148   TCP   66 80 → 53587 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM=1 WS=128
4 0.001082   192.168.179.148   192.168.179.1     TCP   66 5050 → 53587 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM=1 WS=128
```

Figure 2: Wireshark capture

And Tcpdump capture is illustrated in Figure 3.



```
2:29:03.977474 ens33 In  IP 192.168.179.1.58460 > 192.168.179.148.5050: Flags [S], seq 3561785639, win 64240, options [mss 1460,nop,wscale 8,nop,nop,sackOK], length 0
2:29:03.977541 ens33 Out IP 192.168.179.148.58460 > 192.168.179.132.80: Flags [S], seq 3561785639, win 64240, options [mss 1460,nop,wscale 8,nop,nop,sackOK], length 0
2:29:03.978093 ens33 In  IP 192.168.179.132.80 > 192.168.179.148.58460: Flags [S.], seq 3110057525, ack 3561785640, win 64240, options [mss 1460,nop,nop,sackOK,nop,wscale 7], lengt
 0
2:29:03.978118 ens33 Out IP 192.168.179.148.5050 > 192.168.179.1.58460: Flags [S.], seq 3110057525, ack 3561785640, win 64240, options [mss 1460,nop,nop,sackOK,nop,wscale 7], lengt
 0
2:29:03.978577 ens33 In  IP 192.168.179.1.58460 > 192.168.179.148.5050: Flags [.], ack 1, win 513, length 0
2:29:03.978596 ens33 Out IP 192.168.179.148.58460 > 192.168.179.132.80: Flags [.], ack 1, win 513, length 0
2:29:03.978757 ens33 In  IP 192.168.179.1.58460 > 192.168.179.148.5050: Flags [P.], seq 1:363, ack 1, win 513, length 362
2:29:03.978772 ens33 Out IP 192.168.179.148.58460 > 192.168.179.132.80: Flags [P.], seq 1:363, ack 1, win 513, length 362: HTTP: GET / HTTP/1.1
2:29:03.979056 ens33 In  IP 192.168.179.132.80 > 192.168.179.148.58460: Flags [.], ack 363, win 501, length 0
2:29:03.979073 ens33 Out IP 192.168.179.148.5050 > 192.168.179.1.58460: Flags [.], ack 363, win 501, length 0
2:29:03.979444 ens33 In  IP 192.168.179.132.80 > 192.168.179.148.58460: Flags [P.], seq 1:458, ack 363, win 501, length 457: HTTP: HTTP/1.1 302 Found
2:29:03.979463 ens33 Out IP 192.168.179.148.5050 > 192.168.179.1.58460: Flags [P.], seq 1:458, ack 363, win 501, length 457
2:29:03.989812 ens33 In  IP 192.168.179.1.58457 > 192.168.179.148.443: Flags [P.], seq 1:478, ack 1, win 4102, length 477
2:29:03.989903 ens33 Out IP 192.168.179.148.443 > 192.168.179.1.58457: Flags [.], ack 478, win 501, length 0
2:29:03.994178 ens33 Out IP 192.168.179.148.443 > 192.168.179.1.58457: Flags [P.], seq 1:543, ack 478, win 501, length 542
2:29:04.024910 ens33 In  IP 192.168.179.1.58460 > 192.168.179.148.5050: Flags [.], ack 458, win 511, length 0
2:29:04.024943 ens33 Out IP 192.168.179.148.58460 > 192.168.179.132.80: Flags [.], ack 458, win 511, length 0
2:29:04.050817 ens33 In  IP 192.168.179.1.58457 > 192.168.179.148.443: Flags [.], ack 543, win 4106, length 0
2:29:08.985468 ens33 In  IP 192.168.179.132.80 > 192.168.179.148.58460: Flags [F.], seq 458, ack 363, win 501, length 0
2:29:08.985517 ens33 Out IP 192.168.179.148.5050 > 192.168.179.1.58460: Flags [F.], seq 458, ack 363, win 501, length 0
2:29:08.986127 ens33 In  IP 192.168.179.1.58460 > 192.168.179.148.5050: Flags [.], ack 459, win 511, length 0
2:29:08.986162 ens33 Out IP 192.168.179.148.58460 > 192.168.179.132.80: Flags [.], ack 459, win 511, length 0
2:29:08.986221 ens33 In  IP 192.168.179.1.58460 > 192.168.179.148.5050: Flags [F.], seq 363, ack 459, win 511, length 0
2:29:08.986233 ens33 Out IP 192.168.179.148.58460 > 192.168.179.132.80: Flags [F.], seq 363, ack 459, win 511, length 0
2:29:08.986855 ens33 In  IP 192.168.179.132.80 > 192.168.179.148.58460: Flags [.], ack 364, win 501, length 0
2:29:08.986879 ens33 Out IP 192.168.179.148.5050 > 192.168.179.1.58460: Flags [.], ack 364, win 501, length 0
```

Figure 3: Tcpdump capture