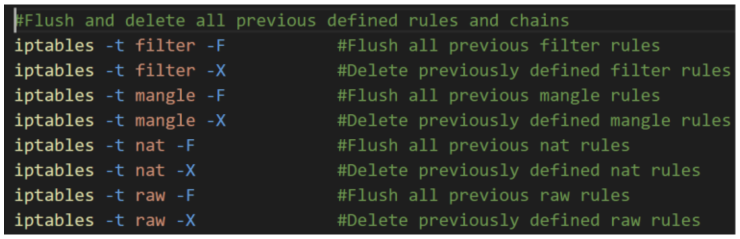
Homework Number: 9

Name: Andrew Wu

ECN Login: wu1795

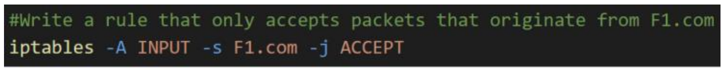
Due Date: 3/28/24

Explanation:



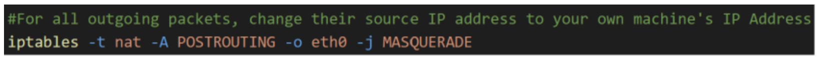
```
#Flush and delete all previous defined rules and chains
iptables -t filter -F           #Flush all previous filter rules
iptables -t filter -X           #Delete previously defined filter rules
iptables -t mangle -F           #Flush all previous mangle rules
iptables -t mangle -X           #Delete previously defined mangle rules
iptables -t nat -F              #Flush all previous nat rules
iptables -t nat -X              #Delete previously defined nat rules
iptables -t raw -F              #Flush all previous raw rules
iptables -t raw -X              #Delete previously defined raw rules
```

For step one, I run through each of the four linux kernel tables, flushing and deleting the previously defined rules for each of them. The -t option represents which table I want to select, with capital f designating flush and capital x designating delete. So for the first line, I call iptables, specify the filter table, and flush the rules currently in the filter table. I do the same command next, except replace f with x, to delete the rules currently in the filter table. I then repeat both commands for the other three tables mangle, nat, and raw.



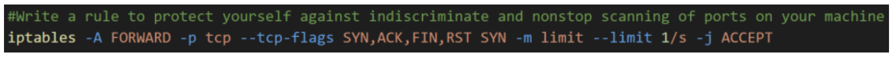
```
#Write a rule that only accepts packets that originate from F1.com
iptables -A INPUT -s F1.com -j ACCEPT
```

For step two, I run a command that only accepts packets from a certain domain, in this case F1.com. The capital a option denotes which chain in the specified table I want to append a new rule to. The s option denotes the specific address where packets are coming from. The j option specifies another target or what to do with the packets, that being accept, reject, drop, or return the packets. So, in the command, I call iptables, denote the input chain in the filter table since we want to only accept packets from a certain domain, denote F1.com since it is the domain we want to only accept packets from, and accept since we want to accept these packets coming from F1.com. Since the filter table is the default table, I did not specifically specify the filter table.



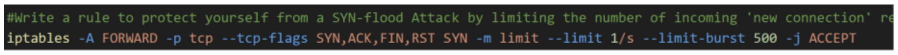
```
#For all outgoing packets, change their source IP address to your own machine's IP Address
iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

For step three, we want to change the source IP address to the machine’s IP address when sending packets out. To do so, I utilize the nat table to alter the packets outgoing IP address through its postrouting chain. To denote that I want to use my own machine’s IP address instead of the source IP address, I use the o option to specify the specific network interface packets are to be facilitated through. As such, I call iptables, specify the nat table, specify the postrouting chain as the chain I want to utilize, use eth0 to denote the machine’s network interface is what packets are to be received by, and masquerade the network IP to provide additional security when sending the packets.



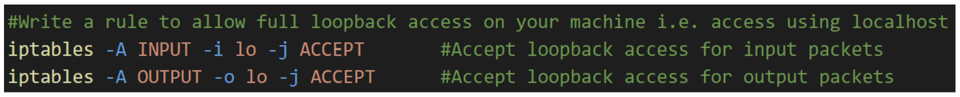
```
#Write a rule to protect yourself against indiscriminate and nonstop scanning of ports on your machine
iptables -A FORWARD -p tcp --tcp-flags SYN,ACK,FIN,RST SYN -m limit --limit 1/s -j ACCEPT
```

For step four, I want to limit the nonstop scanning of ports on the machine. To catch these packets coming into the machine, I can use the forward rule in the filter table to catch them using the A option. To differentiate the packets, I specify that the filtering must only take place in the SYN flag of the tcp through the p option. To limit the speed at which the packets are processed, I use the m option, an extension module, to limit the speed of packet processing. Finally, to accept the packets, I use the special value accept in the j option. All in all, I call iptables, set the a option to the forward chain in the filter table, set the p option to the SYN flag in the tcp, and use the m option to limit the speed of packet processing to one packet per second. I then use the special value accept in the j option to accept all of those packets.



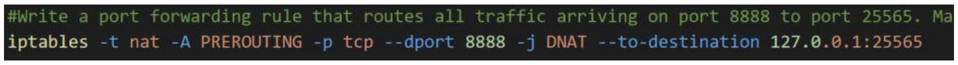
```
#Write a rule to protect yourself from a SYN-flood Attack by limiting the number of incoming 'new connection' re
iptables -A FORWARD -p tcp --tcp-flags SYN,ACK,FIN,RST SYN -m limit --limit 1/s --limit-burst 500 -j ACCEPT
```

For step five, I want to protect my machine from a SYN-flood Attack by limiting the 'new connection' packets to 1 per second after 500 packets have been received. Similarly to step four, I use the forward rule in the filter table via the A option, use the p option to make sure that only the SYN flag of the tcp flags is set, accept incoming packets using the j option, and limit the speed of the packets being processed to 1 per second using the m option. Adding onto the m option, I use limit-burst to limit the number of packets that are received before the rule is active and set it to 500. In the command, I call iptables, set the A option to forward, se the p option to only want the SYN flag to be set out of the tcp flags, use the m option to limit the speed of packet processing to 1 per second after 500 packets have been received, and accept these packets utilizing the j option



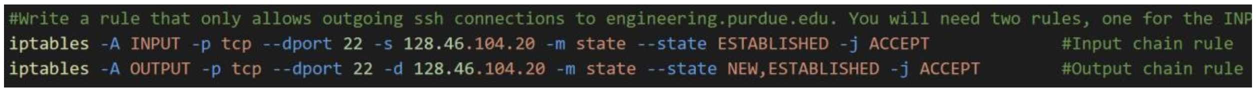
```
#Write a rule to allow full loopback access on your machine i.e. access using localhost
iptables -A INPUT -i lo -j ACCEPT       #Accept loopback access for input packets
iptables -A OUTPUT -o lo -j ACCEPT      #Accept loopback access for output packets
```

For step six, I want to allow full loopback access using localhost on the machine. To do so, I must enable the loopback interface through the i and o options, allow the full loopback access to sent and received packets via the A option, and accept these packets through the j option. In the commands, I call iptables, specify one command for input packets and one for output packets in the A option. For the input command, I use the i option and set the receiving interface to the loopback interface, and allow the packets to be received using the j option, using the special value accept. For the output command, I use the o option to set the sending interface for packets to the loopback interface, and allow the packets to be sent using the j option through the special value accept.



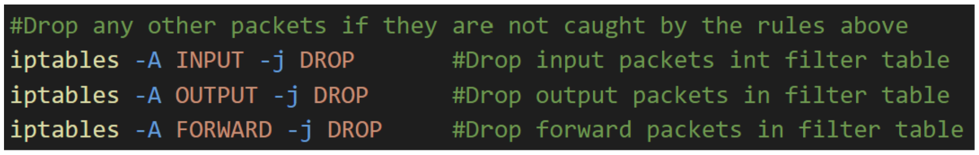
```
#Write a port forwarding rule that routes all traffic arriving on port 8888 to port 25565. Ma
iptables -t nat -A PREROUTING -p tcp --dport 8888 -j DNAT --to-destination 127.0.0.1:25565
```

For step seven, I want to write a forwarding rule for a port that routs all arriving packets on port 8888 to port 25565. To reroute packets to a different port, I use the nat table and its chaing prerouting to catch and alter the packets as soon as they are received. I want to not only target the destination port of the tcp to change any packet with a destination port 8888 to port 25565, but to also change the value of the destination port in the tcp to port 25565. In the command, I call iptables, specify the nat table using the t option, and specify the prerouting chain using the A option. I use the p option to specifically target any packet with the destination port 8888 in the tcp, and use DNAT to specify the new destination port, first specifying the loopback address to ensure the packet stays on the machine, and then routing it to port 25565.



```
#Write a rule that only allows outgoing ssh connections to engineering.purdue.edu. You will need two rules, one for the INP
iptables -A INPUT -p tcp --dport 22 -s 128.46.104.20 -m state --state ESTABLISHED -j ACCEPT          #Input chain rule
iptables -A OUTPUT -p tcp --dport 22 -d 128.46.104.20 -m state --state NEW,ESTABLISHED -j ACCEPT      #Output chain rule
```

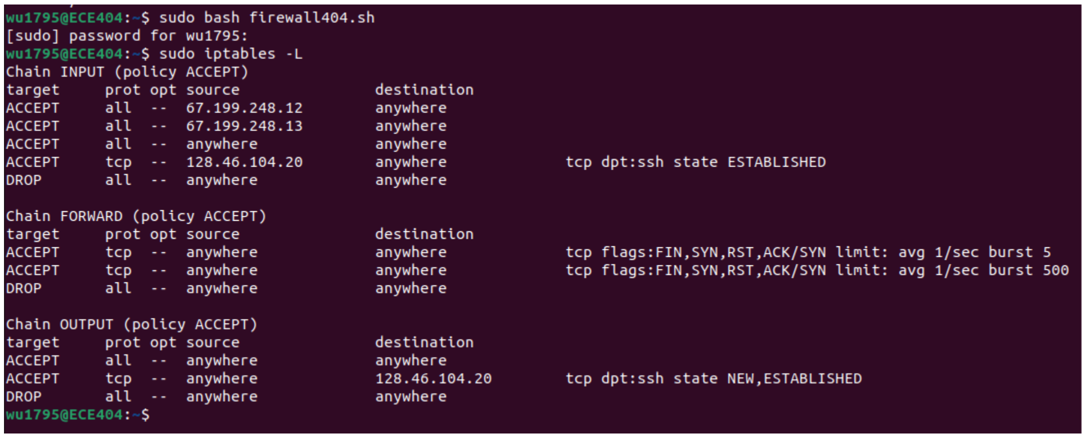
For step eight, I want to make sure that outgoing ssh connections only go to engineering.purdue.edu. I want to ensure that incoming packets have engineering.purdue.edu as their source IP address, and that the outgoing packets have engineering.purdue.edu as their destination IP address. To do so, I call iptables, and use two commands for the input and output chain. For the input chain, I use the p option to ensure that the destination port is port 22, which is the default port for SSH connections, and the s option to check that the source IP address is engineering.purdue.edu. I use the m option to check the state of the packet, ensure that the packet came from an already existing connection by using ESTABLISHED. I then use the j option to accept packets meeting these requirements. I do the same for the output command, except I use the d option to make sure the destination IP address is engineering.purdue.edu instead of the s option, and that the state of the packet in the m option is NEW and ESTABLISHED, since the outgoing packets may not have an existing connection.



```
#Drop any other packets if they are not caught by the rules above
iptables -A INPUT -j DROP       #Drop input packets int filter table
iptables -A OUTPUT -j DROP      #Drop output packets in filter table
iptables -A FORWARD -j DROP     #Drop forward packets in filter table
```

For step nine, I want to drop all other packets that are not encompassed by the rules created above. To drop these packets, I utilize the special drop value in the j option. As such, I call iptables, specify the input, output, and forward chains in the A option, and use the special drop value in the j option to drop all remaining packets not caught by previous rules.

Output:



```
wu1795@ECE404:~$ sudo bash firewall404.sh
[sudo] password for wu1795:
wu1795@ECE404:~$ sudo iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     all  --  67.199.248.12        anywhere
ACCEPT     all  --  67.199.248.13        anywhere
ACCEPT     all  --  anywhere             anywhere
ACCEPT     tcp  --  128.46.104.20        anywhere             tcp dpt:ssh state ESTABLISHED
DROP       all  --  anywhere             anywhere

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination
ACCEPT     tcp  --  anywhere             anywhere             tcp flags:FIN,SYN,RST,ACK/SYN limit: avg 1/sec burst 5
ACCEPT     tcp  --  anywhere             anywhere             tcp flags:FIN,SYN,RST,ACK/SYN limit: avg 1/sec burst 500
DROP       all  --  anywhere             anywhere

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     all  --  anywhere             anywhere
ACCEPT     tcp  --  anywhere             128.46.104.20        tcp dpt:ssh state NEW,ESTABLISHED
DROP       all  --  anywhere             anywhere
wu1795@ECE404:~$
```