A logo of a university of windsor

Description automatically generated with low confidence

|  |
| --- |
| **Course Name** |
| Networking and Data Security (COMP-8677) |

|  |
| --- |
| **Document Type** |
| Lab Assignment 7 |

|  |
| --- |
| **Professor** |
| Dr. Shaoquan Jiang |

|  |
| --- |
| **Team - Members                               Student ID** |
| Manjinder Singh                                 110097177 |

|  |  |
| --- | --- |
| Docker containers are up and running fine with the commands executed (shown in RHS). | A screenshot of a computer  Description automatically generated |

|  |
| --- |
| **1.** Use the following commands on **router** to set the default policies for a table**.**  **sudo iptables –P INPUT ACCEPT**  **sudo iptables –P OUTPUT ACCEPT**  **sudo iptables –P FORWARD DROP**  Recall, INPUT is to check incoming packet; OUTPUT is to check outgoing packet; FORWARDING is to check the passing packet (at router). Further, the commands assume the default table **filter (-t filter)**.  • On 192.168.60.6, run **$ ping 10.9.0.5** and then ping 192.168.60.11. Does it succeed? Explain your observation.  • Change **DROP** to **ACCEPT,** for FORWARD case. Try the pings in the above step again. Now does it succeed? |

**My Implementation of above question 1:-**

|  |  |
| --- | --- |
| **Commands executed in** | |
| **Terminal 1** | **Terminal 2** |
| docksh 30  ls  iptables -P INPUT ACCEPT  iptables -P OUTPUT ACCEPT  iptables -P FORWARD DROP | dockps  docksh b5  ping 10.9.0.5  ping 192.168.60.11 |
|  |  |
| **Observations:**  After executing the iptables command on seed-router in Terminal, we observed the below results in Terminal 2:   * **Ping to 10.9.0.5 (hostA):** In the "host2-192.168.60.6" container, the ping to "10.9.0.5" results in 100% packet loss. This is due to the FORWARD chain in the "seed-router" container being set to **DROP**, preventing the forwarding of packets between containers. * **Ping to 192.168.60.11:** The ping to "192.168.60.11" is successful with 0% packet loss. This signifies that the "192.168.60.11" IP address is reachable from the "host2-192.168.60.6" container. The successful ping indicates that the network allows communication between the containers, or the target IP is likely on the same container host. * To summarize the above, pinging "10.9.0.5" fails due to the DROP policy in the FORWARD chain of the "seed-router" container whereas pinging "192.168.60.11" is successful, indicating that the network allows communication between containers or the target IP is on the same host and are not subject to the FORWARD chain policies. | |
| Now, Changing **DROP** to **ACCEPT,** for FORWARD case and Trying the pings in the above step again. | |
| iptables -P FORWARD ACCEPT | ping 10.9.0.5  ping 192.168.60.11 |
|  |  |
| **Observations:**   * **Ping to 10.9.0.5 (hostA):** The change in the FORWARD chain policy to ACCEPT allows the "host2-192.168.60.6" container to successfully communicate with the "hostA-10.9.0.5" container. All packets are received without any loss. * **Ping to 192.168.60.11:** Similar to the previous observation mentioned in “**Ping to 10.9.0.5 (hostA)**”, the change in the FORWARD chain policy to ACCEPT enables successful communication with the specified IP address "192.168.60.11." All packets are received without any loss.   To summarize, Changing the FORWARD chain policy to ACCEPT has resolved the packet loss issue, and now both pings are successful. The containers can communicate with each other without restrictions imposed by the FORWARD chain policies. | |

|  |
| --- |
| **2.** [**blocking an IP**]  • On 192.168.60.11, if we want to block packets **from** an ip address IP1, use command  **sudo iptables -A INPUT -s IP1 -j DROP**  **/\***this uses INPUT chain because it is incoming packet**\*/**  **On IP1, ping 192.168.60.11 and what can be observed? Explain.**  • On 192.168.60.11, if we want to block packets **to** an ip address IP1, use command  **sudo iptables -A OUTPUT -d IP1 -j DROP**  **/\***this uses OUTPUT chain because it is outgoing packet**\*/**  **On 192.168.60.11, ping IP1 and what can be observed? Explain.** |

**My Implementation of above question 2:-**

|  |  |
| --- | --- |
| iptables -A INPUT -s 192.168.60.6 -j DROP | ping 192.168.60.11 |
|  |  |
| **Observation from above execution of commands:** As per the above commands, we blocked the 192.168.60.6 on router, and if we try to ping from 192.168.60.6 to router then all the packets will be lost, and there will be no received packets on router end. | |
| iptables -A OUTPUT -s 192.168.60.6 -j DROP  ping 192.168.60.6 | ping 192.168.60.11 |
|  | |
| **Observation from above execution of commands:** As per the above commands, we blocked the OUTPUT to the IP Address - 192.168.60.6, so when we are trying to send packets from router to this IP, it will drop all the packets. | |

|  |
| --- |
| **3**. **[List all rules] do it on Router.**  • You can see all the firewall rules by the following command  **$ sudo iptables -L**  /\* again, this assume filter table (i.e., **-t filter**) by default\*/  • You can see all the fire rules in each chain with index number. The index will be used for other operation such as deletion later.  **$ sudo** **iptables -L --line-number** |

**My Implementation of above question 3 Part:-**

iptables -L (Helps to view firewall rules)

iptables -L --line-number (Helps to view firewall rules in each chain with index number)

A screenshot of a computer screen

Description automatically generated

A computer screen shot of a computer code

Description automatically generated

|  |
| --- |
| **4. [Delete a rule]** on **Router,** delete a rule in a chain (such as INPUT) in two steps:  first, list with index:  **$ sudo iptables –L INPUT --line-number**  Then, delete the rule using the index:  **$sudo iptables -D INPUT 1**  Now use the method to delete the first rule in your current INPUT table and then  $sudo iptables -L INPUT to verify whether rule 1 is deleted or not. |

**My Implementation of above question 4 Part:-**

* 1. iptables -L INPUT --line-number
  2. iptables -D INPUT 1
  3. iptables -L INPUT

The above commands are used to display, manage, and update the firewall rules for incoming network traffic on a Linux system. The first command shows the rules with line numbers,

the second command deletes a specific rule, and

the third command shows the updated list of rules after the deletion.

A screenshot of a computer program

Description automatically generated

Again we deleted the remaining rule by using the below commands:-

iptables -D INPUT 1

iptables -L INPUT

A close up of a white background

Description automatically generated

|  |
| --- |
| **5.[Delete all rules in a TABLE]** On **router,** flush the rules in a table (e.g., **filter**):  **$sudo iptables -t filter -F**  /\*again,**-t filter** can be omitted\*/  Then, run **$sudo iptables -L** and you will not see any rule. |

**My Implementation of above question 5 Part:-**

1. iptables -L (First we will preview)
2. iptables -t filter -F
3. iptables -L

The above written commands are used to (i) preview the existing firewall rules, (ii) then clear all the rules from the filter table, and (iii) finally, show the updated list of rules after the flush operation.

It's a way to start with a clean slate for configuring new firewall rules.

A screen shot of a computer program

Description automatically generated

|  |
| --- |
| 6. [**Drop all incoming connections, except telnet**] On **router**, block incoming connections to any service except for telnet. To do this, we can set default policy for INPUT chain of filter Table to be DROP and then specify a rule to accept incoming telnet connection.  **$ sudo iptables -P INPUT DROP**  **$ sudo iptables -A INPUT -p tcp - -dport 23 -j ACCEPT**  /\* A default policy is applied only if all the rules in the chain have been executed without making a decision (either ACCEPT or DROP or REJECT). For example, if we ssh to router, then the rule does not ACCEPT but also not REJECT. So the default policy applies. Note: here **-p** stands for protocol. \*/  Then, ping and telnet to 192.168.60.11 (from other VM). Which succeeds (telnet or ping)?  /\*after this problem, run **$ sudo iptables -F** to flush all rules in **filter** table and recover the default policy: **$ sudo iptables -P INPUT ACCEPT** \*/ |

**My Implementation of above question 6 Part:-**

|  |  |
| --- | --- |
|  |  |
| iptables -P INPUT DROP  iptables -A INPUT -p tcp –dport 23 -j ACCEPT | ping 192.168.60.11  telnet 192.168.60.11 |
|  |  |

**Observation -**Telnet to "192.168.60.11" from the "host2-192.168.60.6" container succeeds. The connection is established, and the prompt is received.

|  |  |
| --- | --- |
|  |  |
| iptables -F  iptables -P INPUT ACCEPT | ping 192.168.60.11  telnet 192.168.60.11 |
|  |  |

**Observation -** After executing **iptables -F** and **iptables -P INPUT ACCEPT commands**, all the incoming traffic is allowed. Therefore, both the ping and telnet commands to 192.168.60.11 are succeeded while assuming there are no network issues or restrictions on the target host.

The firewall has been effectively disabled by allowing all incoming traffic, so both ping and telnet worked as per above screenshot without any restrictions.

|  |
| --- |
| **7.** [**drop outgoing DNS request to 8.8.8.8**] In this case, since it is outgoing packet, we add rule to OUTPUT chain. Since it is DNS request, the destination should be the DNS server, which has a port number 53. Finally, since DNS is implemented using UDP, we use protocol UDP. Hence, we add the following rule:  **$ sudo iptables -A OUTPUT -p udp - - dport 53 -d 8.8.8.8 -j DROP**  Then, try **$ dig www.uwindsor.ca** and **dig @8.8.8.8 www.uwindsor.ca**. Which succeeds?  /\* delete the rule in order not to affect the following experiment \*/ |

**My Implementation of above question 7 Part:-**

iptables -A OUTPUT -p udp - - dport 53 -d 8.8.8.8 -j DROP

dig www.uwindsor.ca

dig @8.8.8.8 www.uwindsor.ca

**A screen shot of a computer

Description automatically generated**

**A close up of a number

Description automatically generated**

dig [www.uwindsor.ca](http://www.uwindsor.ca) - succeeds where dig @8.8.8.8 [www.uwindsor.ca](http://www.uwindsor.ca) - connection timed out.

**Reason -** The first dig command (dig www.uwindsor.ca) succeeds because it uses the local DNS resolver (127.0.0.11), and the iptables rule only blocks traffic to the external DNS server ("8.8.8.8").

The second dig command (dig @8.8.8.8 www.uwindsor.ca) fails due to the iptables rule, as it explicitly attempts to reach an external DNS server ("8.8.8.8"), which is blocked by the rule.

|  |
| --- |
| **8.** [**block incoming ping request**] You can not ping uwindsor webserver. Most likely, this is blocked by firewall of uwindsor. Here is the way to block an incoming icmp request.  **$ sudo iptables -A INPUT -p icmp --icmp-type echo-request -j DROP**  Run this on **router** and ping router from another VM. Do you get any reply? Explain. |

**My Implementation of above question 8 Part:-**

|  |  |
| --- | --- |
|  |  |
| iptables -A INPUT -p icmp --icmp-type echo-request -j DROP | ping 192.168.60.11 |
|  |  |

**Explanation -** The **iptables** rule specifically targets incoming ICMP echo requests (**--icmp-type echo-request**) and drops them (**-j DROP**). As a result, when attempting to ping the router from another VM, the router will not respond to the ICMP echo requests, i.e. , the ping requests will not receive any replies due to the **iptables** rule blocking incoming ICMP echo requests on the router.

|  |
| --- |
| **9.** Suppose that you want to block all incoming connections while you do not want your visit to external servers to be affected. However, if you send a request to an external server, the server will reply to you while this packet will be blocked by your firewall. To resolve this issue, you should regard the response packet (to your request) as related to your outgoing request packet and allowed to come in. This is achieved using the *conntrack* module.  **$ sudo iptables -P INPUT DROP**  **$ sudo iptables -A INPUT -p tcp -m conntrack --ctstate RELATED, ESTABLISHED -j ACCEPT**  Try this on **router** VM. Then, telnet to a VM (e.g. 192.168.60.7).  Next, telnet from the latter (192.168.60.7) to **router**. Which telnet session directly succeeds? |

**My Implementation of above question 9 Part:-**

|  |  |
| --- | --- |
|  |  |
| iptables -P INPUT DROP  iptables -A INPUT -p tcp -m conntrack --ctstate RELATED, ESTABLISHED -j ACCEPT  telnet 192.168.60.7 [# Attempt telnet from the router to another VM (e.g., 192.168.60.7)] |  |
|  |  |
|  | telnet 10.9.0.1 [# Attempt telnet from another VM (e.g., 192.168.60.7) to the router ] |
|  |  |

**Explanation –** Both telnet sessions directly succeed. The iptables rules ensure that responses related to established connections are allowed through the firewall, providing the necessary bidirectional communication for telnet sessions to work.

This configuration allows outgoing connections to external servers and permits the corresponding response packets to come back in, ensuring that your visits to external servers are not affected by the incoming connection restrictions.

|  |
| --- |
| **10.** (optional) [save your firewall rules and restore it] After you have done firewall, you want to save your rules to a file you can run  **$ sudo iptables-save >myiptables.rules**  Later, you can restore your rules by running  **$ sudo iptables-restore <myiptables.rules**  /\* to see the effect, you can flush your firewall after running iptables-save command and then run iptables-restore command to see if you have restored your firewall \*/ |

**My Implementation of above question 10 Part:-**

1. iptables-save > myiptables.rules
2. iptables -F
3. iptables-restore < myiptables.rules

The first command saves the current iptables rules to a file.

The second command clears (flushes) all existing firewall rules.

The third command restores the previously saved rules from the file, effectively applying the saved configuration.

The above sequence of commands is a way to temporarily disable the firewall (by clearing rules) and later restore it to a known state using the saved rules. It can be useful for testing or managing firewall configurations.

Displaying firewall rules after restoring :

A screenshot of a computer program

Description automatically generated

**References: -**

1. Lab Manual for Lab 7 from Brightspace
2. Lecture Notes for Lab 7 from Brightspace

**One Drive Link for Lab 7 Solution(Word File and PDF Document) for Lab 7 Work:-**

[Networking and Data Security - Lab 7 - Submitted to Doc](https://uwin365-my.sharepoint.com/:f:/g/personal/lnu8_uwindsor_ca/EgoKyw80utpCl9pf2esCxn0Bn_eW-Cs0HAGWAJQ8KLrvWw?e=cPSNmf)