CySA+ Lab Series

Lab 08: Understanding ACLs and Firewalls

Document Version: 2022-10-10

Material in this Lab Aligns to the Following	
CompTIA CySA+ (CS0-002) Exam Objectives	1.7 - Given a scenario, implement controls to mitigate attacks and software vulnerabilities 2.1 - Explain software assurance best practices 3.2 - Given a scenario, implement configuration changes to existing controls to improve security
All-In-One CompTIA CySA+ Second Edition ISBN-13: 978-1260464306 Chapters	7: Mitigating Controls for Attacks and Software Vulnerabilities 8: Security Solutions for Infrastructure Management 12: Implement Configuration Changes to Existing Controls to Improve Security

Copyright © 2022 Network Development Group, Inc.
www.netdevgroup.com

NETLAB+ is a registered trademark of Network Development Group, Inc.
KALI LINUX ™ is a trademark of Offensive Security.
ALIEN VAULT OSSIM V is a trademark of AlienVault, Inc.
Microsoft®, Windows®, and Windows Server® are trademarks of the Microsoft group of companies.
Greenbone is a trademark of Greenbone Networks GmbH.
SECURITY ONION is a trademark of Security Onion Solutions LLC.
Android is a trademark of Google LLC.
pfSense® is a registered mark owned by Electric Sheep Fencing LLC ("ESF").
All trademarks, logos, and brand names are the property of their respective owners.

Contents

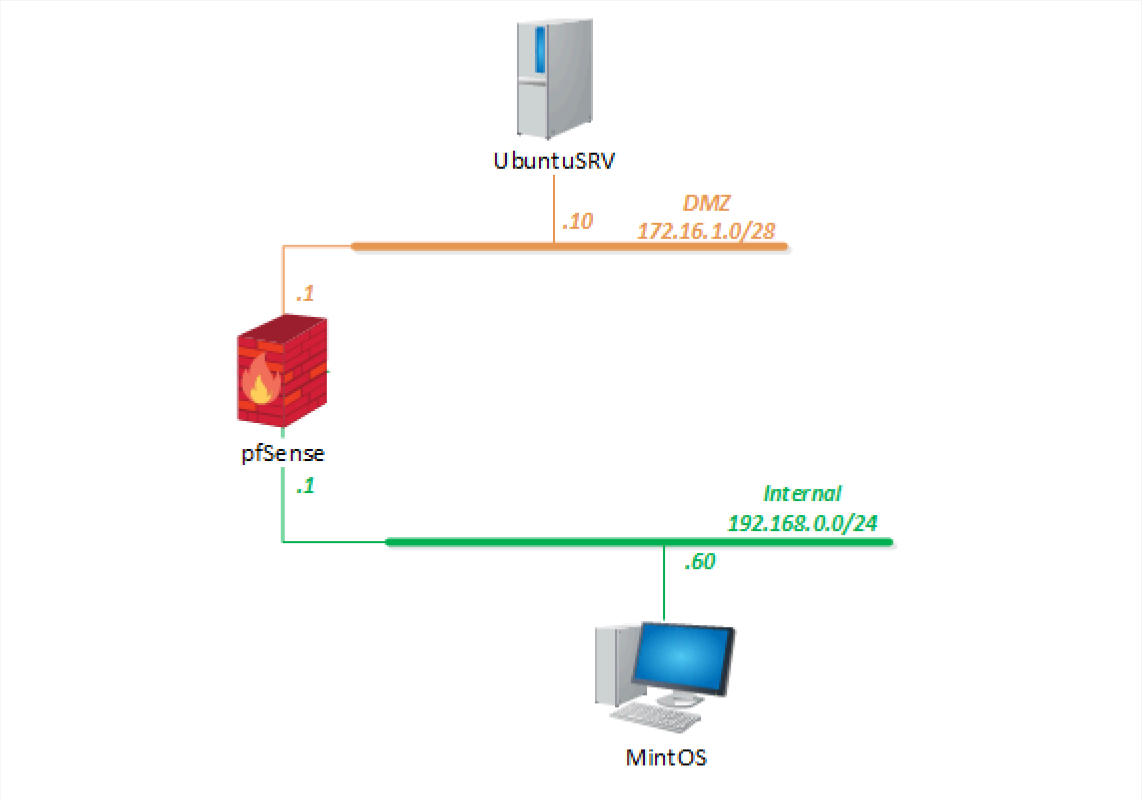
Introduction

A host-based firewall is used to filter incoming and outgoing network traffic and operates as the first line of defense to a server OS. A set of rules (*Access Control List/ACL*), can be applied to allow only the incoming and outgoing connections that are required and to block all others. Most modern OS-based firewalls implement stateful packet inspection.

Objective

- Understand *Stateful* vs. *Stateless* firewalls
- Use *IPTables* to create and implement firewall rules

Lab Topology



Copyright © 2022 Network Development Group, Inc. www.netdevgroup.com

Lab Settings

The information in the table below will be needed in order to complete the lab. The task sections below provide details on the use of this information.

Virtual Machine	IP Address	Account	Password
WinOS (Server 2019)	192.168.0.50	Administrator	NDGlabpass123!
MintOS (Linux Mint)	192.168.0.60	sysadmin	NDGlabpass123!
OSSIM (Alien Vault)	172.16.1.2	root	NDGlabpass123!
UbuntuSRV (Ubuntu Server)	172.16.1.10	sysadmin	NDGlabpass123!
Kali	203.0.113.2	sysadmin	NDGlabpass123!
pfSense	203.0.113.1 172.16.1.1 192.168.0.1	admin	NDGlabpass123!

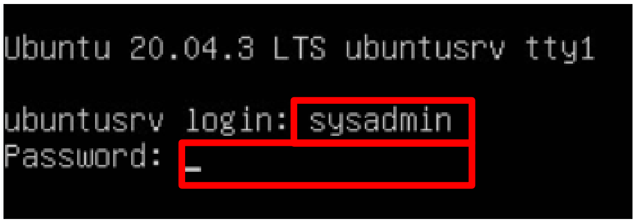
Copyright © 2022 Network Development Group, Inc. www.netdevgroup.com

1 Setting up Stateful Firewalls Using IPTables

Stateful and **Stateless** firewalls work differently. A **Stateless** firewall will make all **DROP/ACCEPT** decisions on a packet-by-packet basis. There is no understanding of how one packet relates to any other packet coming into the firewall. A **Stateful** firewall is aware of the connections that pass through it. It adds and maintains information about a user's connections in a state table, referred to as a **Connection Table**. It then uses this table to implement the security policies for that user’s connections. Examples of software that functions as a **Stateful** firewall include *IPTables*, *Windows Firewall*, *Palo Alto Firewall*, and *Checkpoint Firewall*. In this task, you will examine how to use *IPTables*.

IPTables functions as a **Stateful Firewall**. In this task, you will flush all standing *IPTables* chains and rules, set up basic rules to block unwanted traffic, and allow traffic on certain ports.

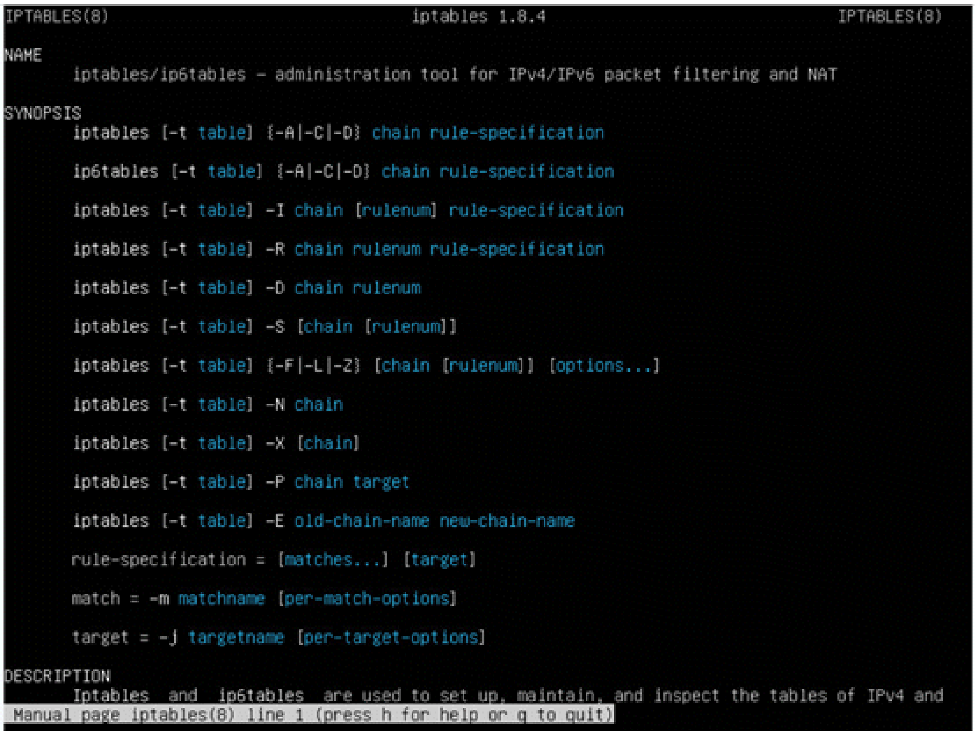
1. Set the focus on the **UbuntuSRV** and log in as `sysadmin` with the password: `NDGlabpass123!`



```
Ubuntu 20.04.3 LTS ubuntusrv tty1

ubuntusrv login: sysadmin
Password: _
```

2. Type the following command to view the *IPTables* manual pages. Spend a few minutes looking over the information before continuing with this lab. When you are finished reviewing the content of the *IPTables man pages*, press **q** to quit.

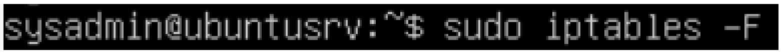
```
man iptables
```



```
IPTABLES(8)                          iptables 1.8.4                          IPTABLES(8)

NAME
       iptables/ip6tables — administration tool for IPv4/IPv6 packet filtering and NAT

SYNOPSIS
       iptables [-t table] {-A|-C|-D} chain rule-specification

       ip6tables [-t table] {-A|-C|-D} chain rule-specification

       iptables [-t table] -I chain [rulenum] rule-specification

       iptables [-t table] -R chain rulenum rule-specification

       iptables [-t table] -D chain rulenum

       iptables [-t table] -S [chain [rulenum]]

       iptables [-t table] {-F|-L|-Z} [chain [rulenum]] [options...]

       iptables [-t table] -N chain

       iptables [-t table] -X [chain]

       iptables [-t table] -P chain target

       iptables [-t table] -E old-chain-name new-chain-name

       rule-specification = [matches...] [target]

       match = -m matchname [per-match-options]

       target = -j targetname [per-target-options]

DESCRIPTION
       Iptables  and  ip6tables  are used to set up, maintain, and inspect the tables of IPv4 and
 Manual page iptables(8) line 1 (press h for help or q to quit)
```

3. It’s important to start with a clean *IPTables,* so that pre-existing rules do not cause unforeseen behavior. Chains are named ordered lists of rules, displayed 1 per line. To flush the current chains from *iptables*, type the following command. Use the password `NDGlabpass123!` If prompted:

```
sudo iptables –F
```

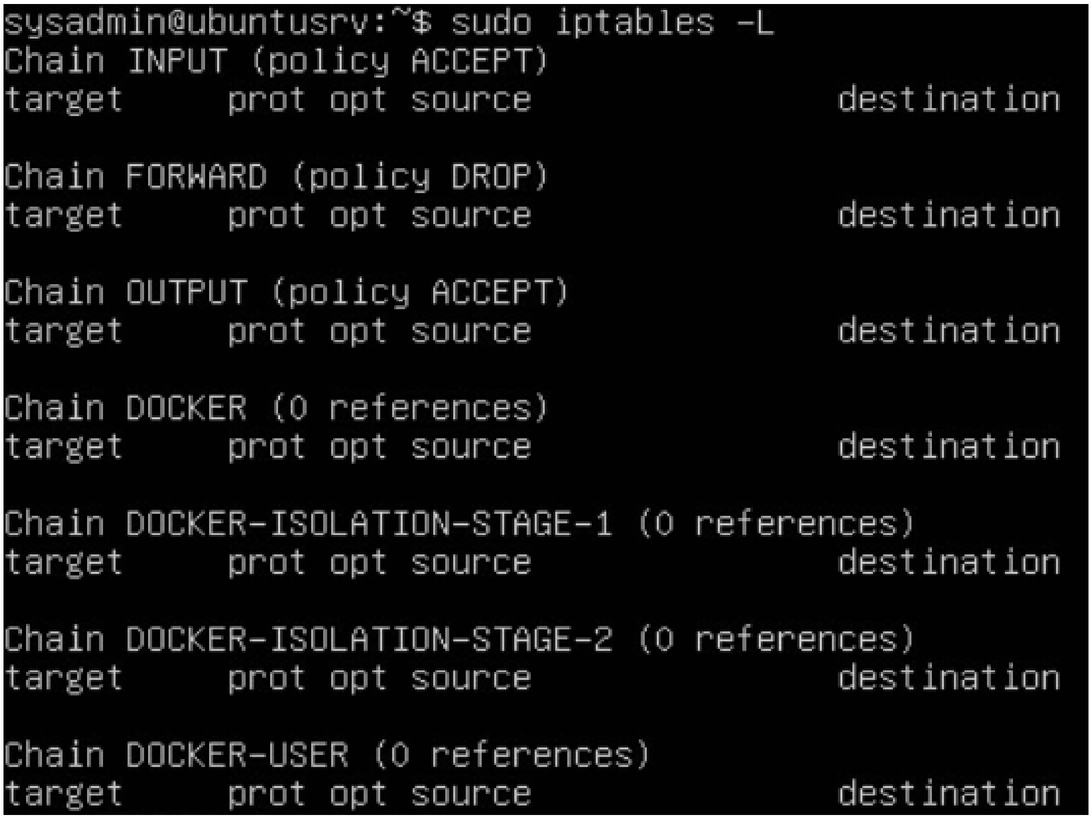


```
sysadmin@ubuntusrv:~$ sudo iptables -F
```

The **-F (**or **--flush)** switch flushes the selected chain. If no chains are specified, it will flush every chain in the table. This is equivalent to deleting all of the rules one by one.

4. Confirm the *iptables* have been flushed by viewing the rules that currently exist in the *iptables* chains by typing the following *iptables* command, using the –L option (list the contents of the table). Use the password `NDGlabpass123!` if prompted.

```
sudo iptables –L
```



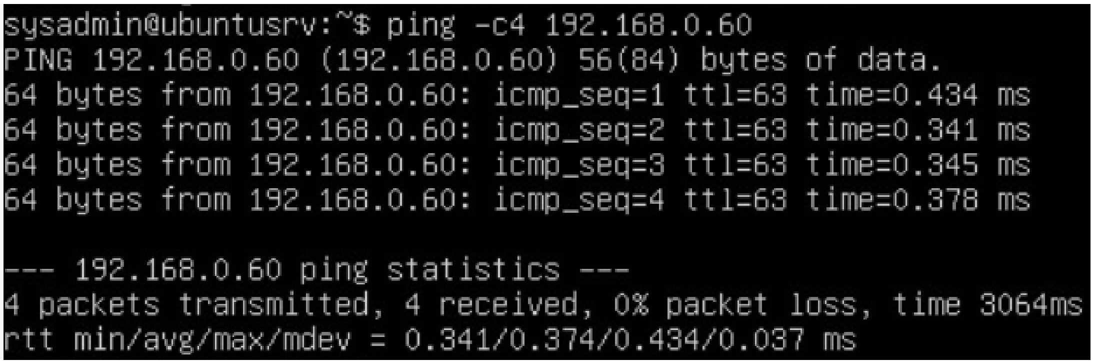
```
sysadmin@ubuntusrv:~$ sudo iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy DROP)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination

Chain DOCKER (0 references)
target     prot opt source               destination

Chain DOCKER-ISOLATION-STAGE-1 (0 references)
target     prot opt source               destination

Chain DOCKER-ISOLATION-STAGE-2 (0 references)
target     prot opt source               destination

Chain DOCKER-USER (0 references)
target     prot opt source               destination
```

The common chains are:

-P INPUT - This is for packets destined to local sockets. (Inbound Packets)
-P FORWARD - This is for packets being routed through the device.
(Interface to Interface)
-P OUTPUT - This is for locally generated packets. (Outbound)

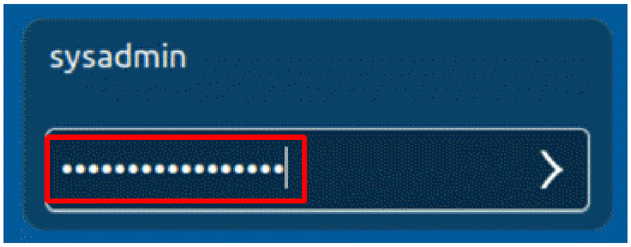
5. Test *icmp* functionality by pinging the **MintOS** computer with the following command:

```
ping -c4 192.168.0.60
```



```
sysadmin@ubuntusrv:~$ ping -c4 192.168.0.60
PING 192.168.0.60 (192.168.0.60) 56(84) bytes of data.
64 bytes from 192.168.0.60: icmp_seq=1 ttl=63 time=0.434 ms
64 bytes from 192.168.0.60: icmp_seq=2 ttl=63 time=0.341 ms
64 bytes from 192.168.0.60: icmp_seq=3 ttl=63 time=0.345 ms
64 bytes from 192.168.0.60: icmp_seq=4 ttl=63 time=0.378 ms

--- 192.168.0.60 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3064ms
rtt min/avg/max/mdev = 0.341/0.374/0.434/0.037 ms
```

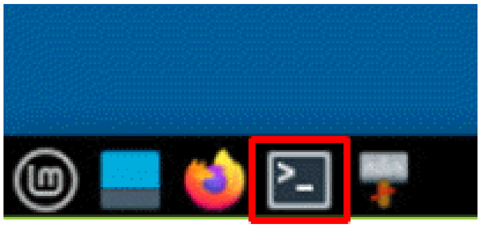
6. Set the focus to the **MintOS** computer tab.
7. Log in to the *sysadmin* account using the password: `NDGlabpass123!`



sysadmin

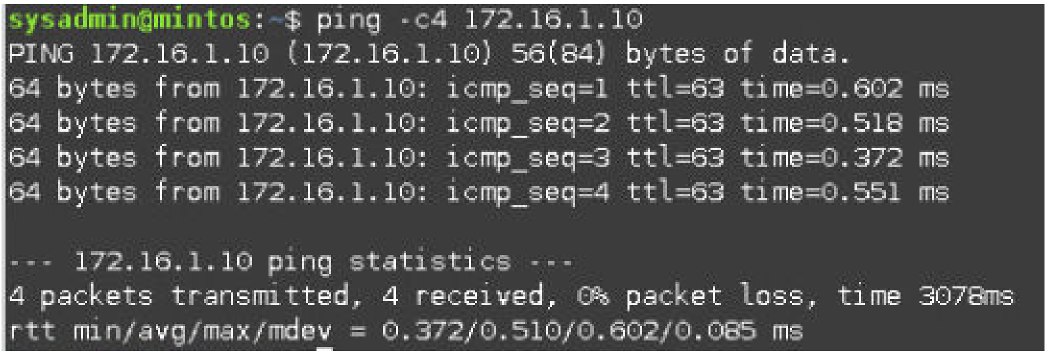
•••••••••••••••••

8. Click on the **Terminal** icon in the taskbar at the bottom of the screen.



9. Since rules can allow or block both inbound and outbound traffic, it is important to test pings both ways for *icmp* connectivity. To do so, type the following command:

```
ping -c4 172.16.1.10
```

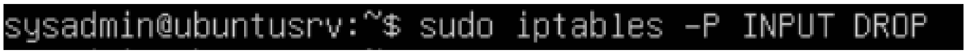


```
sysadmin@mintos:~$ ping -c4 172.16.1.10
PING 172.16.1.10 (172.16.1.10) 56(84) bytes of data.
64 bytes from 172.16.1.10: icmp_seq=1 ttl=63 time=0.602 ms
64 bytes from 172.16.1.10: icmp_seq=2 ttl=63 time=0.518 ms
64 bytes from 172.16.1.10: icmp_seq=3 ttl=63 time=0.372 ms
64 bytes from 172.16.1.10: icmp_seq=4 ttl=63 time=0.551 ms

--- 172.16.1.10 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3078ms
rtt min/avg/max/mdev = 0.372/0.510/0.602/0.085 ms
```

Copyright © 2022 Network Development Group, Inc. www.netdevgroup.com

10. Click the **UnbuntuSRV** tab to return to the Ubuntu server.
11. Start with the following command to drop all inbound traffic. Use the password `NDGlabpass123!` if prompted.

```
sudo iptables -P INPUT DROP
```

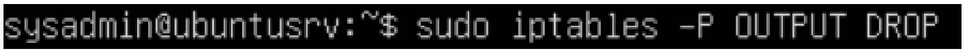


```
sysadmin@ubuntusrv:~$ sudo iptables -P INPUT DROP
```

The **-P** (or **--policy)** chain target sets the policy for the built-in (non-user-defined) chain to the given target. The policy target must be either **ACCEPT** or **DROP**.

12. Use the following command to drop all outbound traffic:

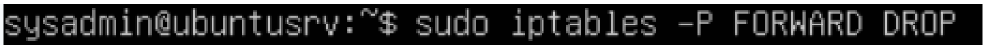
```
sudo iptables -P OUTPUT DROP
```



```
sysadmin@ubuntusrv:~$ sudo iptables -P OUTPUT DROP
```

13. To drop all forwarded traffic, type the following command:

```
sudo iptables -P FORWARD DROP
```

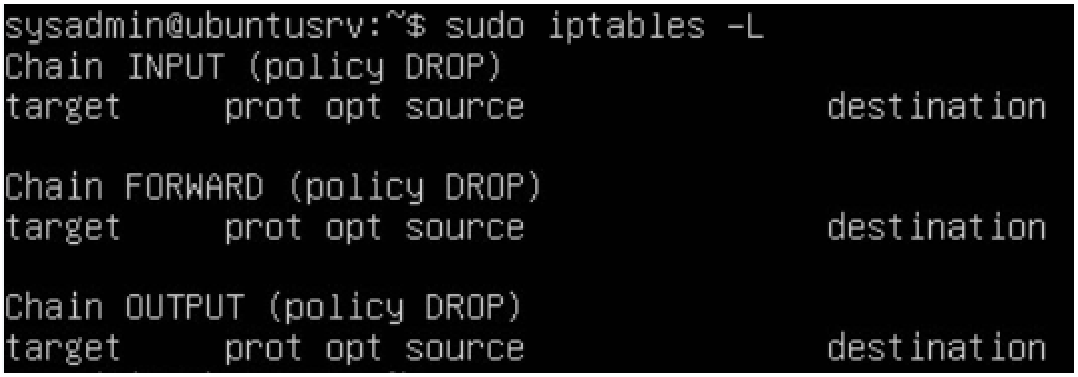


```
sysadmin@ubuntusrv:~$ sudo iptables -P FORWARD DROP
```

These commands take effect immediately. If you type these commands in remotely using SSH, you will cut off network connectivity and lose remote access capability.

14. To view the chains you just created, type the following command:

```
sudo iptables -L
```



```
sysadmin@ubuntusrv:~$ sudo iptables -L
Chain INPUT (policy DROP)
target     prot opt source               destination

Chain FORWARD (policy DROP)
target     prot opt source               destination

Chain OUTPUT (policy DROP)
target     prot opt source               destination
```

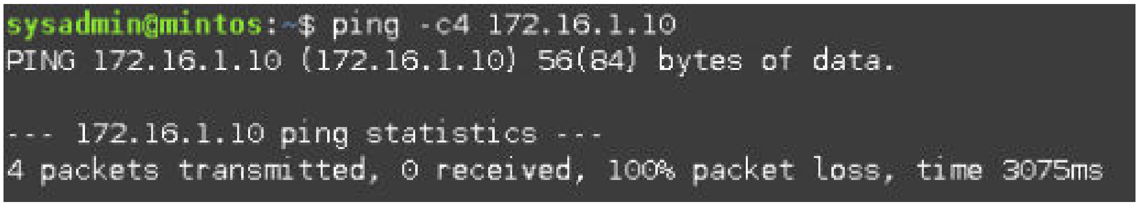
15. Return to the **MintOS** computer.

Copyright © 2022 Network Development Group, Inc. www.netdevgroup.com

16. Test *icmp* connectivity to the *MintOS* machine again.

```
ping -c4 172.16.1.10
```

Since you are now dropping all traffic, you will receive no packets in response.

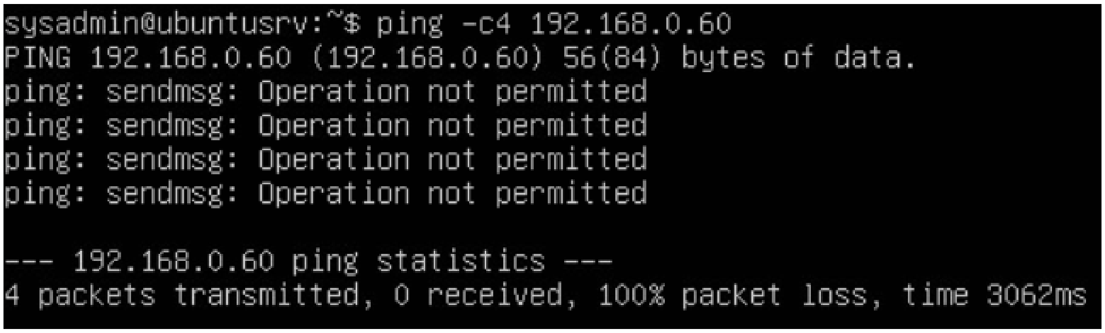


```
sysadmin@mintos:~$ ping -c4 172.16.1.10
PING 172.16.1.10 (172.16.1.10) 56(84) bytes of data.

--- 172.16.1.10 ping statistics ---
4 packets transmitted, 0 received, 100% packet loss, time 3075ms
```

17. Return to the **UbuntuSRV** computer.
18. Test *icmp* functionality by pinging the *MintOS* computer with the following command:

```
ping -c4 192.168.0.60
```

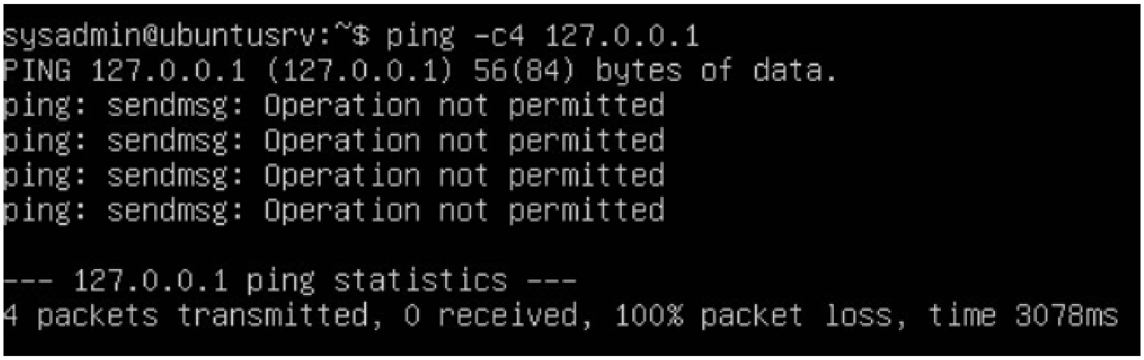
Since you are now dropping all outbound traffic, you receive an operation not permitted message.



```
sysadmin@ubuntusrv:~$ ping -c4 192.168.0.60
PING 192.168.0.60 (192.168.0.60) 56(84) bytes of data.
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted

--- 192.168.0.60 ping statistics ---
4 packets transmitted, 0 received, 100% packet loss, time 3062ms
```

19. Ping the **loopback** interface with the following command. Notice that the operation is still not permitted.

```
ping -c4 127.0.0.1
```

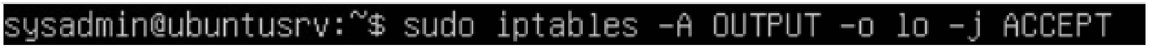


```
sysadmin@ubuntusrv:~$ ping -c4 127.0.0.1
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted

--- 127.0.0.1 ping statistics ---
4 packets transmitted, 0 received, 100% packet loss, time 3078ms
```

Copyright © 2022 Network Development Group, Inc. www.netdevgroup.com

20. Allow outbound traffic to the **loopback** interface by changing the *iptables* rule by typing the following command. Type the password `NDGlabpass123!` if you are prompted again.

```
sudo iptables -A OUTPUT -o lo -j ACCEPT
```



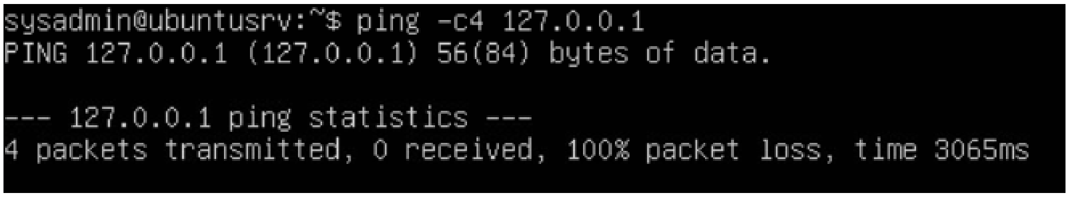
```
sysadmin@ubuntusrv:~$ sudo iptables -A OUTPUT -o lo -j ACCEPT
```

The **-A** (or **--append)** chain rule-specification appends one or more rules to the end of the selected chain. When the source and/or destination names resolve to more than one address, a rule will be added for each possible address combination.

21. Now that you have allowed outbound traffic to the loopback interface, once again, type the following command:

```
ping -c4 127.0.0.1
```

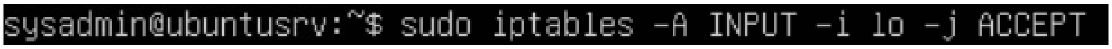
Notice that the ping still fails because returning traffic is not allowed, but the operation is now permitted.



```
sysadmin@ubuntusrv:~$ ping -c4 127.0.0.1
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.

--- 127.0.0.1 ping statistics ---
4 packets transmitted, 0 received, 100% packet loss, time 3065ms
```

22. Allow incoming traffic from the loopback interface with the following command:

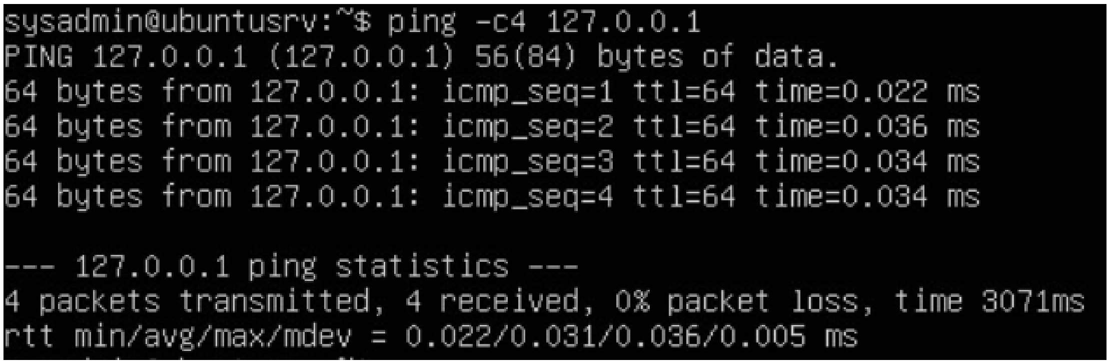
```
sudo iptables -A INPUT -i lo -j ACCEPT
```



```
sysadmin@ubuntusrv:~$ sudo iptables -A INPUT -i lo -j ACCEPT
```

23. Type the following command to confirm all inbound and outbound traffic is now allowed to/from the loopback interface.

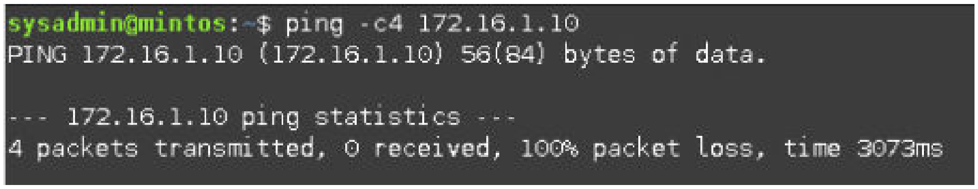
```
ping -c4 127.0.0.1
```



```
sysadmin@ubuntusrv:~$ ping -c4 127.0.0.1
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.022 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.036 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.034 ms
64 bytes from 127.0.0.1: icmp_seq=4 ttl=64 time=0.034 ms

--- 127.0.0.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3071ms
rtt min/avg/max/mdev = 0.022/0.031/0.036/0.005 ms
```

24. Return to the **MintOS** computer.
25. Verify that outside traffic still isn't allowed to the *UbuntuSRV* computer with the following command:

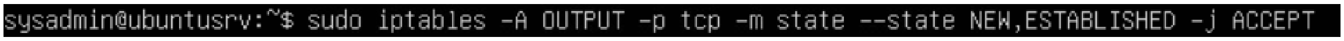
```
ping -c4 172.16.1.10
```



```
sysadmin@mintos:~$ ping -c4 172.16.1.10
PING 172.16.1.10 (172.16.1.10) 56(84) bytes of data.

--- 172.16.1.10 ping statistics ---
4 packets transmitted, 0 received, 100% packet loss, time 3073ms
```

26. Return to the **UbuntuSRV** computer.
27. Type the following command to allow outbound **tcp** traffic from the **UbuntuSRV** machine. Type the `NDGlabpass123!` credentials, if prompted.

```
sudo iptables -A OUTPUT -p tcp -m state --state NEW,ESTABLISHED -j ACCEPT
```



```
sysadmin@ubuntusrv:~$ sudo iptables -A OUTPUT -p tcp -m state --state NEW,ESTABLISHED -j ACCEPT
```

There are several options you can use when allowing outbound traffic.

The **-m state** switch enables statefull session handling.

The **-p** (or **--protocol)** switch defines the protocol of the rule or of the packet to check. The specified protocol can be one of the following:

- **“all”** - This keyword is equivalent to the number zero, and will match all protocols. This is the default when protocol is omitted.
- **tcp, udp, udplite, icmp, icmpv6, esp, ah, sctp, mh**
- A numeric value which represents a protocol
- A protocol name from /etc/protocols

Packets can be in one of several states:

NEW - The packet has started a new connection, or otherwise associated with a connection which has not seen packets in both directions.

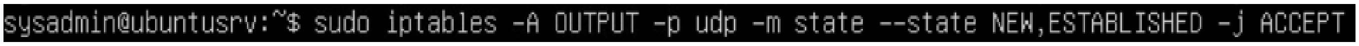
ESTABLISHED - The packet is associated with a connection which has seen packets in both directions.

RELATED - The packet is starting a new connection, but is associated with an existing connection, such as an FTP data transfer or an ICMP error. This is handled by a series of kernel modules called **ip_conntrack_*** each of which is written for a particular protocol that uses unrelated connections. (Such as FTP)

An **INVALID** packet state indicates that the traffic could not be identified.

28. Type the following command to allow outbound **udp** traffic.

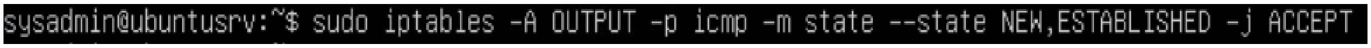
```
sudo iptables –A OUTPUT –p udp –m state ––state NEW,ESTABLISHED –j ACCEPT
```



```
sysadmin@ubuntusrv:~$ sudo iptables -A OUTPUT -p udp -m state --state NEW,ESTABLISHED -j ACCEPT
```

29. Type the following command to allow outbound **icmp** traffic.

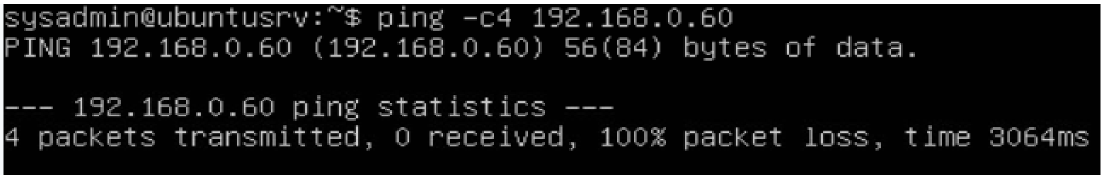
```
sudo iptables -A OUTPUT -p icmp -m state --state NEW,ESTABLISHED -j ACCEPT
```



```
sysadmin@ubuntusrv:~$ sudo iptables -A OUTPUT -p icmp -m state --state NEW,ESTABLISHED -j ACCEPT
```

30. Test to ensure outbound *icmp* traffic is now permitted with the following command. Note that while the action is now permitted, you still receive no responses. This is because inbound and returning traffic is still under the implicit **deny** rule you set up.

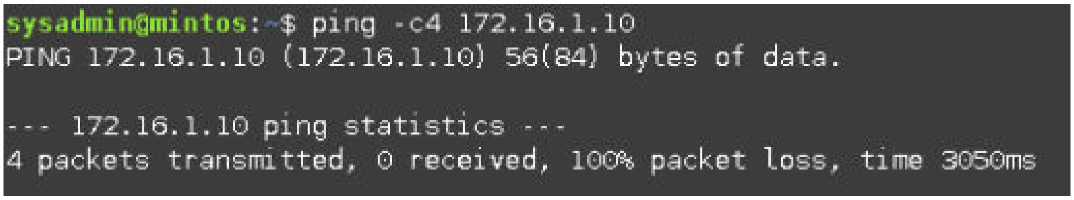
```
ping -c4 192.168.0.60
```



```
sysadmin@ubuntusrv:~$ ping -c4 192.168.0.60
PING 192.168.0.60 (192.168.0.60) 56(84) bytes of data.

--- 192.168.0.60 ping statistics ---
4 packets transmitted, 0 received, 100% packet loss, time 3064ms
```

31. Return to the **MintOS** computer.
32. You did not allow inbound **icmp** traffic, so the *MintOS* computer should not be able to reach the *UbuntuSRV* machine. Confirm this with the following command:

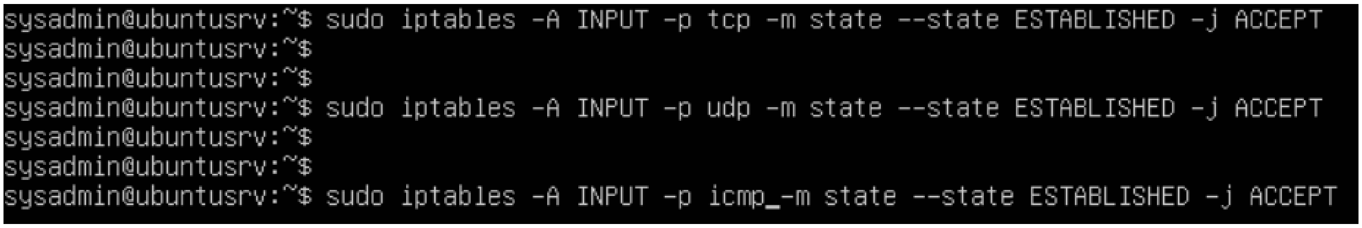
```
ping -c4 172.16.1.10
```



```
sysadmin@mintos:~$ ping -c4 172.16.1.10
PING 172.16.1.10 (172.16.1.10) 56(84) bytes of data.

--- 172.16.1.10 ping statistics ---
4 packets transmitted, 0 received, 100% packet loss, time 3050ms
```

33. Return to the **UbuntuSRV** computer.
34. Set up rules to allow returning **tcp, udp,** and **icmp** traffic. The use of **ESTABLISHED**, without the **NEW** option, will only allow packets that were generated on the *UbuntuSRV* computer to be returned. Type the following commands:

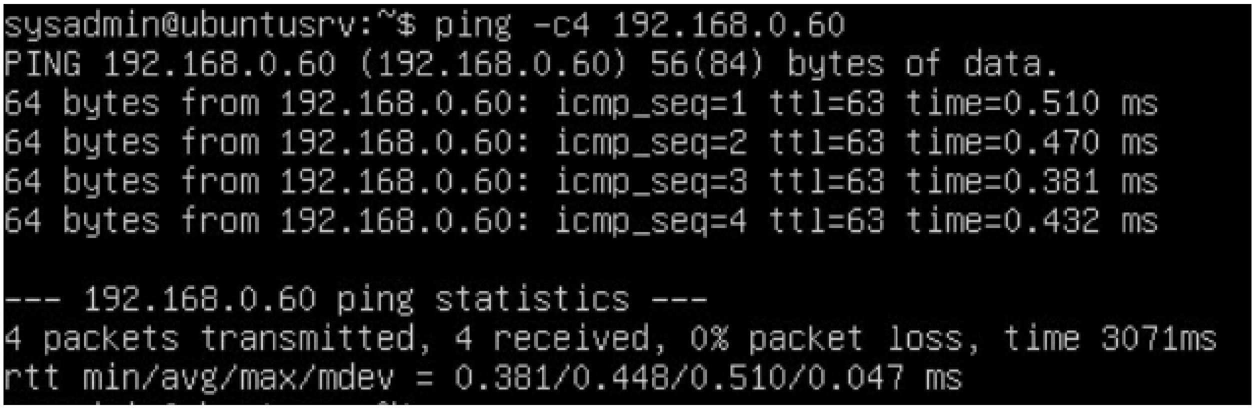
```
sudo iptables -A INPUT -p tcp -m state --state ESTABLISHED -j ACCEPT
sudo iptables -A INPUT -p udp -m state --state ESTABLISHED -j ACCEPT
sudo iptables -A INPUT -p icmp -m state --state ESTABLISHED -j ACCEPT
```



```
sysadmin@ubuntusrv:~$ sudo iptables -A INPUT -p tcp -m state --state ESTABLISHED -j ACCEPT
sysadmin@ubuntusrv:~$
sysadmin@ubuntusrv:~$
sysadmin@ubuntusrv:~$ sudo iptables -A INPUT -p udp -m state --state ESTABLISHED -j ACCEPT
sysadmin@ubuntusrv:~$
sysadmin@ubuntusrv:~$
sysadmin@ubuntusrv:~$ sudo iptables -A INPUT -p icmp_-m state --state ESTABLISHED -j ACCEPT
```

35. Confirm that packets generated on the *UbuntuSRV* machine can be sent out and return with the following command:

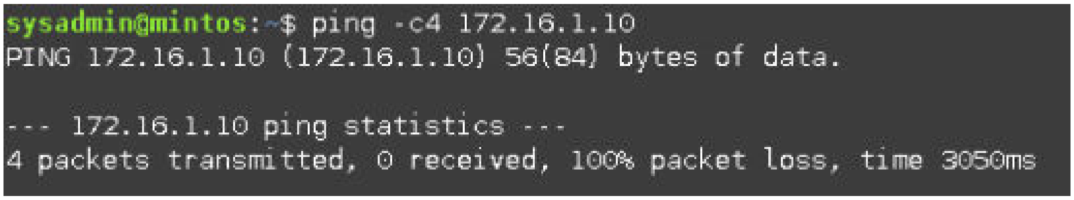
```
ping -c4 192.168.0.60
```



```
sysadmin@ubuntusrv:~$ ping -c4 192.168.0.60
PING 192.168.0.60 (192.168.0.60) 56(84) bytes of data.
64 bytes from 192.168.0.60: icmp_seq=1 ttl=63 time=0.510 ms
64 bytes from 192.168.0.60: icmp_seq=2 ttl=63 time=0.470 ms
64 bytes from 192.168.0.60: icmp_seq=3 ttl=63 time=0.381 ms
64 bytes from 192.168.0.60: icmp_seq=4 ttl=63 time=0.432 ms

--- 192.168.0.60 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3071ms
rtt min/avg/max/mdev = 0.381/0.448/0.510/0.047 ms
```

36. Return to the **MintOS** computer.
37. Ping from the *MintOS* computer to the *UbuntuSRV* computer. Type the following command:

```
ping -c4 172.16.1.10
```

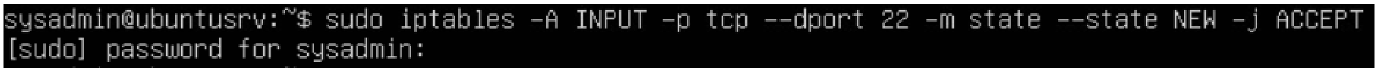


```
sysadmin@mintos:~$ ping -c4 172.16.1.10
PING 172.16.1.10 (172.16.1.10) 56(84) bytes of data.

--- 172.16.1.10 ping statistics ---
4 packets transmitted, 0 received, 100% packet loss, time 3050ms
```

Notice that when you ping from the **MintOS** computer to the **UbuntuSRV** machine, the packet is generated on the **MintOS** computer. Therefore, it is a **NEW** packet, and you only allowed **ESTABLISHED** inbound packets. Therefore the ping will still fail.

38. Return to the **UbuntuSRV** computer.
39. Now that you have configured basic options for your *iptables* access list, you can view more advanced options that are configured to allow traffic in. Let's set up more advanced firewall rules. Allow incoming traffic on **port 22**. This is the port used for **SSH**. Type the following command. Type the `NDGlabpass123!` if asked.

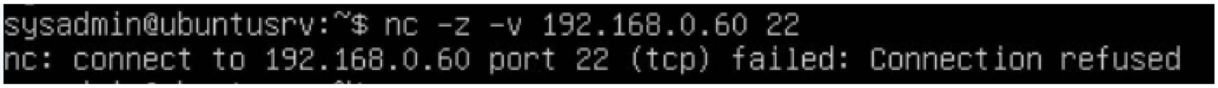
```
sudo iptables -A INPUT -p tcp --dport 22 -m state --state NEW -j ACCEPT
```



```
sysadmin@ubuntusrv:~$ sudo iptables -A INPUT -p tcp --dport 22 -m state --state NEW -j ACCEPT
[sudo] password for sysadmin:
```

40. Type the following command to test outbound connectivity on **Port 22**. Notice that because you only allowed **inbound** traffic, this test fails.

```
nc -z -v 192.168.0.60 22
```

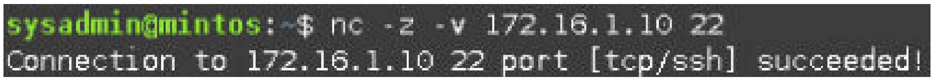


```
sysadmin@ubuntusrv:~$ nc -z -v 192.168.0.60 22
nc: connect to 192.168.0.60 port 22 (tcp) failed: Connection refused
```

41. Click on the **MintOS** machine.
42. Test the inbound connectivity for the *UbuntuSRV* on **Port 22** with the following command.

```
nc -z -v 172.16.1.10 22
```

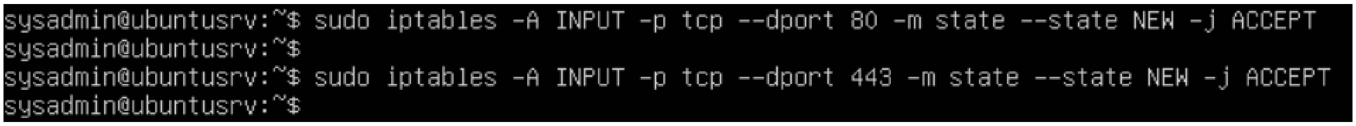
From the **MintOS** machine, note that the command succeeds.



```
sysadmin@mintos:~$ nc -z -v 172.16.1.10 22
Connection to 172.16.1.10 22 port [tcp/ssh] succeeded!
```

43. Return to the **UbuntuSrv** computer.
44. Add two rules to allow inbound traffic on **Port 80 (HTTP)** and **Port 443 (HTTPS)** with the following commands:

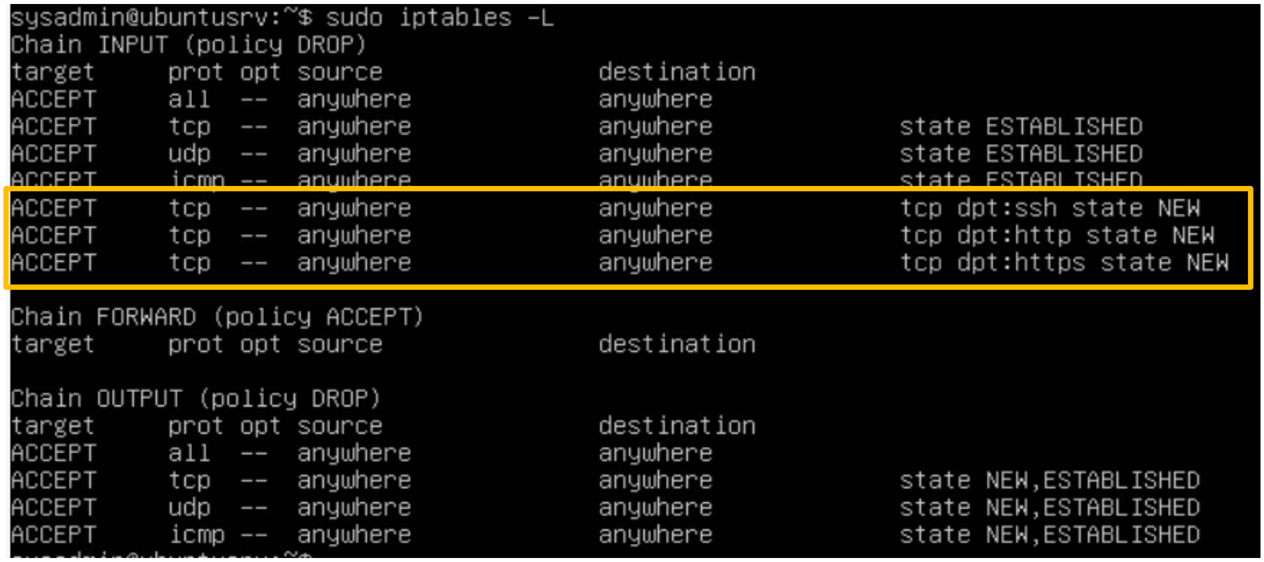
```
sudo iptables -A INPUT -p tcp --dport 80 -m state --state NEW -j ACCEPT
sudo iptables -A INPUT -p tcp --dport 443 -m state --state NEW -j ACCEPT
```



```
sysadmin@ubuntusrv:~$ sudo iptables -A INPUT -p tcp --dport 80 -m state --state NEW -j ACCEPT
sysadmin@ubuntusrv:~$
sysadmin@ubuntusrv:~$ sudo iptables -A INPUT -p tcp --dport 443 -m state --state NEW -j ACCEPT
sysadmin@ubuntusrv:~$
```

45. To see the new rules that have been added, display the *iptables* by using the following command:

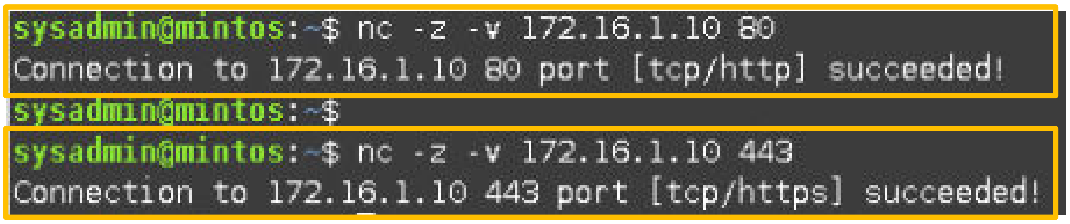
```
sudo iptables -L
```



```
sysadmin@ubuntusrv:~$ sudo iptables -L
Chain INPUT (policy DROP)
target     prot opt source               destination
ACCEPT     all  --  anywhere             anywhere
ACCEPT     tcp  --  anywhere             anywhere             state ESTABLISHED
ACCEPT     udp  --  anywhere             anywhere             state ESTABLISHED
ACCEPT     icmp --  anywhere             anywhere             state ESTABLISHED
ACCEPT     tcp  --  anywhere             anywhere             tcp dpt:ssh state NEW
ACCEPT     tcp  --  anywhere             anywhere             tcp dpt:http state NEW
ACCEPT     tcp  --  anywhere             anywhere             tcp dpt:https state NEW

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy DROP)
target     prot opt source               destination
ACCEPT     all  --  anywhere             anywhere
ACCEPT     tcp  --  anywhere             anywhere             state NEW,ESTABLISHED
ACCEPT     udp  --  anywhere             anywhere             state NEW,ESTABLISHED
ACCEPT     icmp --  anywhere             anywhere             state NEW,ESTABLISHED
```

46. Return to the **MintOS** computer.
47. Test to ensure that inbound **HTTP** and **HTTPS** traffic is allowed by typing the following commands:

```
nc -z -v 172.16.1.10 80
nc -z -v 172.16.1.10 443
```



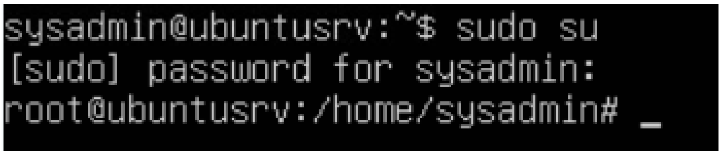
```
sysadmin@mintos:~$ nc -z -v 172.16.1.10 80
Connection to 172.16.1.10 80 port [tcp/http] succeeded!
sysadmin@mintos:~$
sysadmin@mintos:~$ nc -z -v 172.16.1.10 443
Connection to 172.16.1.10 443 port [tcp/https] succeeded!
```

2 Configuring and Managing IPTables

iptables can be a little confusing as it does not run as a service. It is just a command that allows you to interact with a list of rules that are applied to decide if a network session is interesting or not and what to do with it as a result. In this task, you will create *iptables* rules and determine if the rules you have created are working as intended.

1. Set the focus to the **UbuntuSRV** computer.
2. To save the *iptables* settings from the previous section to a file, obtain **root** privileges with the following command, typing the password `NDGlabpass123!` if prompted.

```
sudo su
```



```
sysadmin@ubuntusrv:~$ sudo su
[sudo] password for sysadmin:
root@ubuntusrv:/home/sysadmin# _
```

3. Under root access, type the following command to create a directory for the *iptables*.

```
mkdir /etc/iptables
```

```
root@ubuntusrv:/# mkdir /etc/iptables
```

4. Use the following command to save the rules you just created as **rules.v4** in the /etc/iptables folder you created in the previous step.

```
iptables-save > /etc/iptables/rules.v4
```

```
root@ubuntusrv:/# iptables-save > /etc/iptables/rules.v4
```

5. Exit *root* mode with the following command:

```
exit
```

```
root@ubuntusrv:/# exit
```

When entering *IPtables* rules at the command line you are only modifying the current firewall configuration. You need to save your rules after you make changes. If you reboot without saving, the last saved firewall rules will be used.

In *Ubuntu* you can manually restore an *IPtables* saved configuration with the following command:
`iptables-restore < /etc/sysconfig/rules.v4`

6. Using switches with the **iptables** command can allow you to view more detailed information. Type the following command. If prompted, use `NDGlabpass123!` as the password.

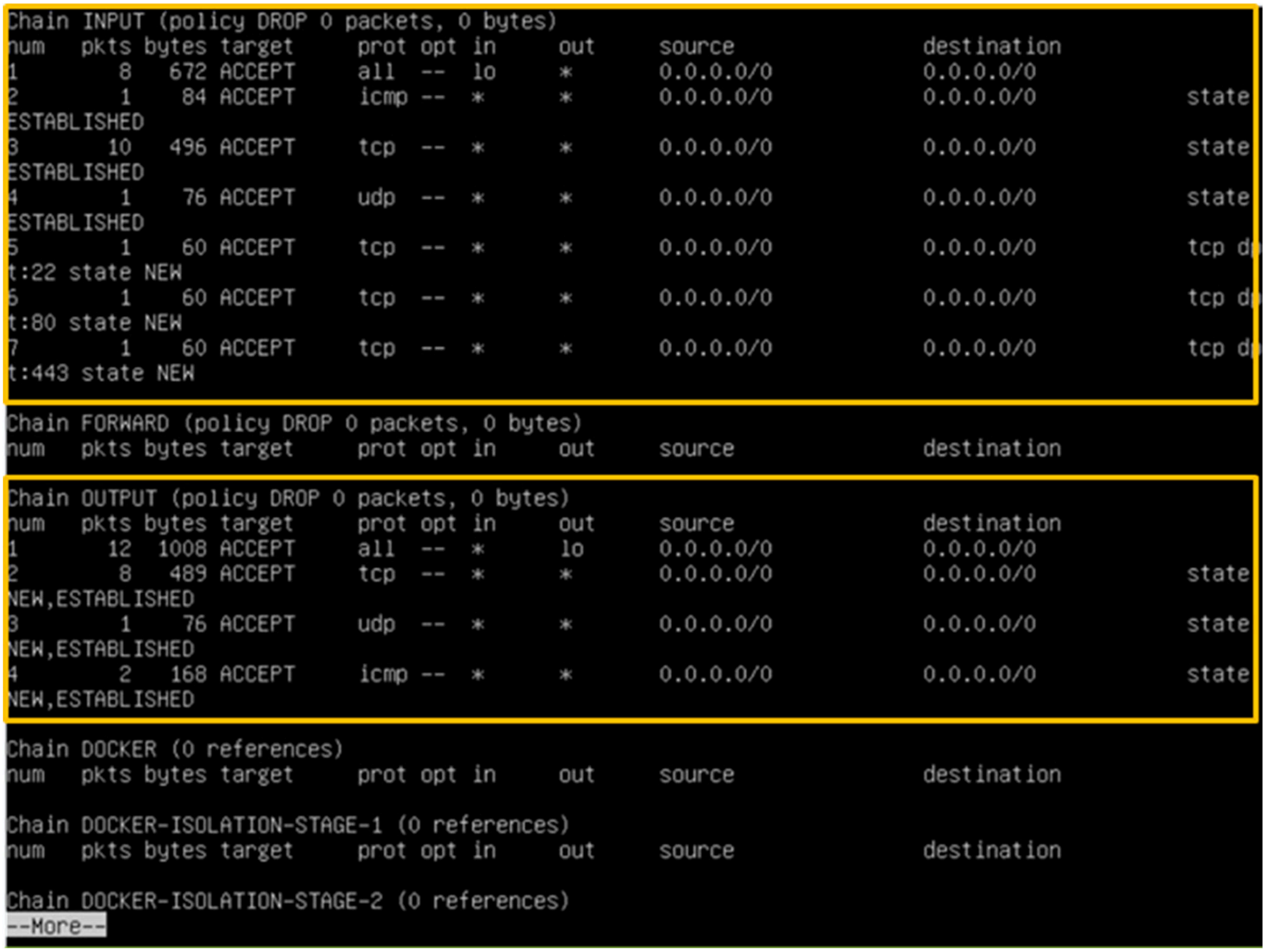
```
sudo iptables -n -v -L --line-numbers | more
```

- The **-n** (or **--numeric)** switch gives you numeric output. IP addresses and port numbers will be printed in numeric format. Without this switch, the program will try to display them as hostnames, network names, or services whenever applicable.

- The **-v** (or **--verbose)** switch gives you more detailed output. (This includes rule hit counters which is great for debugging.)

- The **--line-numbers** option adds line numbers to the beginning of each rule, corresponding to that rule's position in the chain.

The **| more** will stop the output when the screen fills. After viewing the first screen, press the **Spacebar** to finish the list.

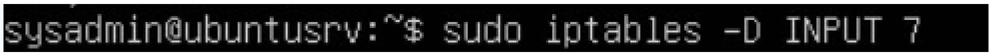


```
Chain INPUT (policy DROP 0 packets, 0 bytes)
num   pkts bytes target     prot opt in     out     source               destination
1        8   672 ACCEPT     all  --  lo     *       0.0.0.0/0            0.0.0.0/0
2        1    84 ACCEPT     icmp --  *      *       0.0.0.0/0            0.0.0.0/0            state
ESTABLISHED
3       10   496 ACCEPT     tcp  --  *      *       0.0.0.0/0            0.0.0.0/0            state
ESTABLISHED
4        1    76 ACCEPT     udp  --  *      *       0.0.0.0/0            0.0.0.0/0            state
ESTABLISHED
5        1    60 ACCEPT     tcp  --  *      *       0.0.0.0/0            0.0.0.0/0            tcp d
t:22 state NEW
6        1    60 ACCEPT     tcp  --  *      *       0.0.0.0/0            0.0.0.0/0            tcp d
t:80 state NEW
7        1    60 ACCEPT     tcp  --  *      *       0.0.0.0/0            0.0.0.0/0            tcp d
t:443 state NEW

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

Chain OUTPUT (policy DROP 0 packets, 0 bytes)
num   pkts bytes target     prot opt in     out     source               destination
1       12  1008 ACCEPT     all  --  *      lo      0.0.0.0/0            0.0.0.0/0
2        8   489 ACCEPT     tcp  --  *      *       0.0.0.0/0            0.0.0.0/0            state
NEW,ESTABLISHED
3        1    76 ACCEPT     udp  --  *      *       0.0.0.0/0            0.0.0.0/0            state
NEW,ESTABLISHED
4        2   168 ACCEPT     icmp --  *      *       0.0.0.0/0            0.0.0.0/0            state
NEW,ESTABLISHED

Chain DOCKER (0 references)
num   pkts bytes target     prot opt in     out     source               destination

Chain DOCKER-ISOLATION-STAGE-1 (0 references)
num   pkts bytes target     prot opt in     out     source               destination

Chain DOCKER-ISOLATION-STAGE-2 (0 references)
--More--
```

Make note that there are currently **7 INPUT** chains and **4 OUTPUT** chains.

7. You can delete rules one line at a time using the **-D** switch. Delete INPUT rule number 7 by typing this command:

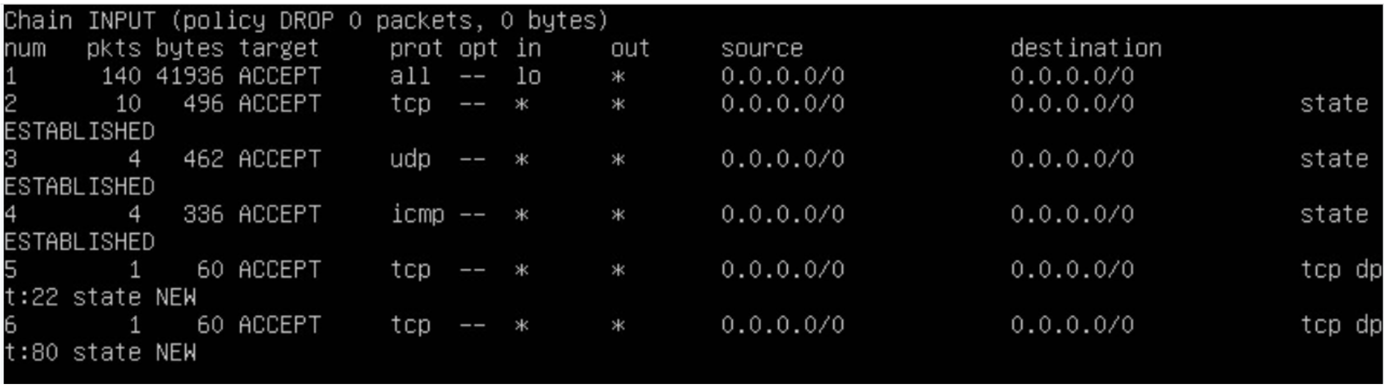
```
sudo iptables -D INPUT 7
```



```
sysadmin@ubuntusrv:~$ sudo iptables -D INPUT 7
```

8. View the *iptables* detail again by typing the command one more time. Notice that you now only have 6 rules in the chain. The rule for **Port 443** has been deleted.

```
sudo iptables -n -v -L --line-numbers | more
```

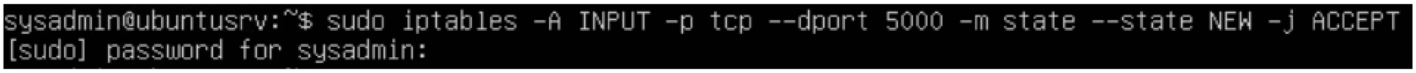


```
Chain INPUT (policy DROP 0 packets, 0 bytes)
num   pkts bytes target     prot opt in     out     source               destination
1      140 41936 ACCEPT     all  --  lo     *       0.0.0.0/0            0.0.0.0/0
2       10   496 ACCEPT     tcp  --  *      *       0.0.0.0/0            0.0.0.0/0            state
ESTABLISHED
3        4   462 ACCEPT     udp  --  *      *       0.0.0.0/0            0.0.0.0/0            state
ESTABLISHED
4        4   336 ACCEPT     icmp --  *      *       0.0.0.0/0            0.0.0.0/0            state
ESTABLISHED
5        1    60 ACCEPT     tcp  --  *      *       0.0.0.0/0            0.0.0.0/0            tcp dp
t:22 state NEW
6        1    60 ACCEPT     tcp  --  *      *       0.0.0.0/0            0.0.0.0/0            tcp dp
t:80 state NEW
```

Press the **Spacebar** to finish the list.

9. Let's open another port in *iptables* to see how the open port can be exploited. Type the following command to allow new input connections on **Port 5000**. Use the password `NDGlabpass123!` if prompted.

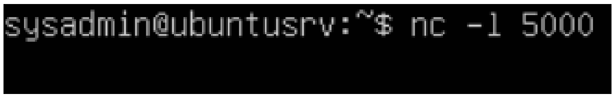
```
sudo iptables -A INPUT -p tcp --dport 5000 -m state --state NEW -j ACCEPT
```



```
sysadmin@ubuntusrv:~$ sudo iptables -A INPUT -p tcp --dport 5000 -m state --state NEW -j ACCEPT
[sudo] password for sysadmin:
```

10. Press **Alt+F2** to open another session and log in with the username `sysadmin` and `NDGlabpass123!` as the password. Set the *UbuntuSRV* computer to listen on **Port 5000** by typing the following command:

```
nc -l 5000
```



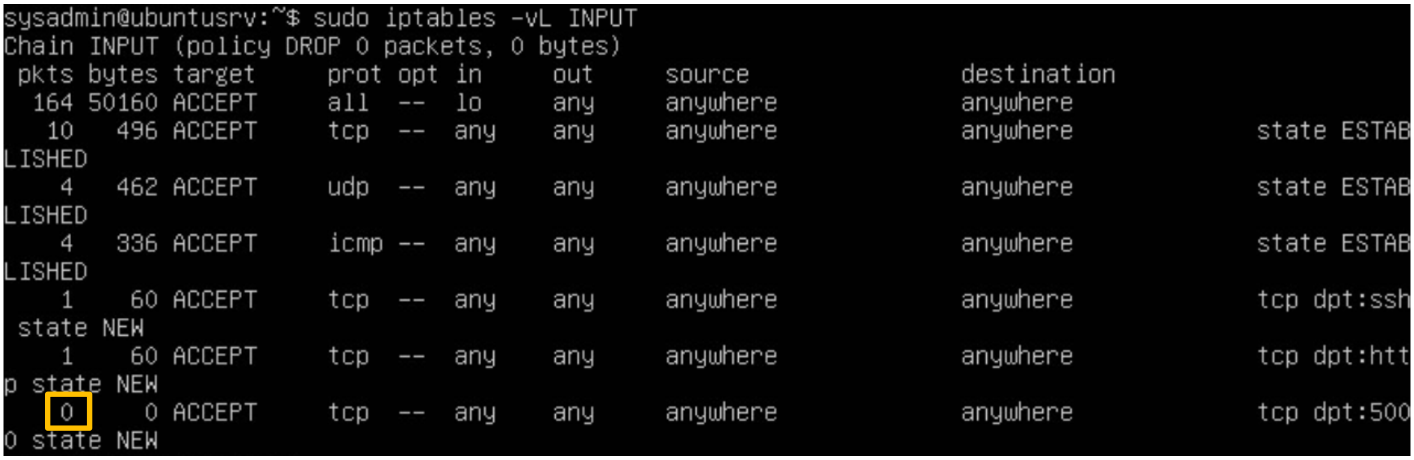
```
sysadmin@ubuntusrv:~$ nc -l 5000
```

Since the *nc* command opens and listens on a port, there will be no response after typing the command. But, the service is listening and waiting.

Copyright © 2022 Network Development Group, Inc. www.netdevgroup.com

11. Press **Alt+F1** to return to the first session and type the following command to list *iptables* and observe the **hit counter**. The counter should show **0** since there has not been a connection established on this port. This is a very useful tool for determining if the rules you have created left the port open for connections.

```
sudo iptables -vL INPUT
```

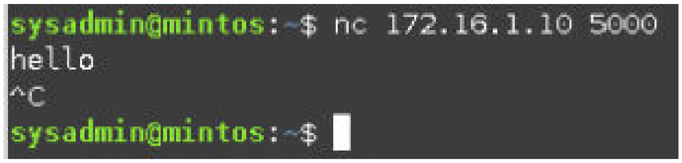


```
sysadmin@ubuntusrv:~$ sudo iptables -vL INPUT
Chain INPUT (policy DROP 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
  164 50160 ACCEPT     all  --  lo     any     anywhere             anywhere
   10   496 ACCEPT     tcp  --  any    any     anywhere             anywhere             state ESTAB
LISHED
    4   462 ACCEPT     udp  --  any    any     anywhere             anywhere             state ESTAB
LISHED
    4   336 ACCEPT     icmp --  any    any     anywhere             anywhere             state ESTAB
LISHED
    1    60 ACCEPT     tcp  --  any    any     anywhere             anywhere             tcp dpt:ssh
 state NEW
    1    60 ACCEPT     tcp  --  any    any     anywhere             anywhere             tcp dpt:htt
p state NEW
    0     0 ACCEPT     tcp  --  any    any     anywhere             anywhere             tcp dpt:500
0 state NEW
```

12. Set the focus to the **MintOS** computer.
13. Use the **nc** command to connect to the *UbuntuSRV* machine via **Port 5000** and send a **hello** by typing the following commands:

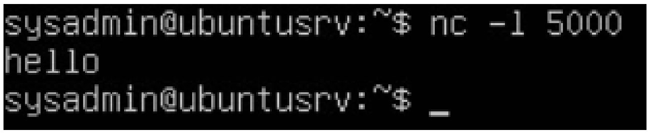
```
nc 172.16.1.10 5000
hello
```

Press **Ctrl+C** to end the **nc** process.



```
sysadmin@mintos:~$ nc 172.16.1.10 5000
hello
^C
sysadmin@mintos:~$
```

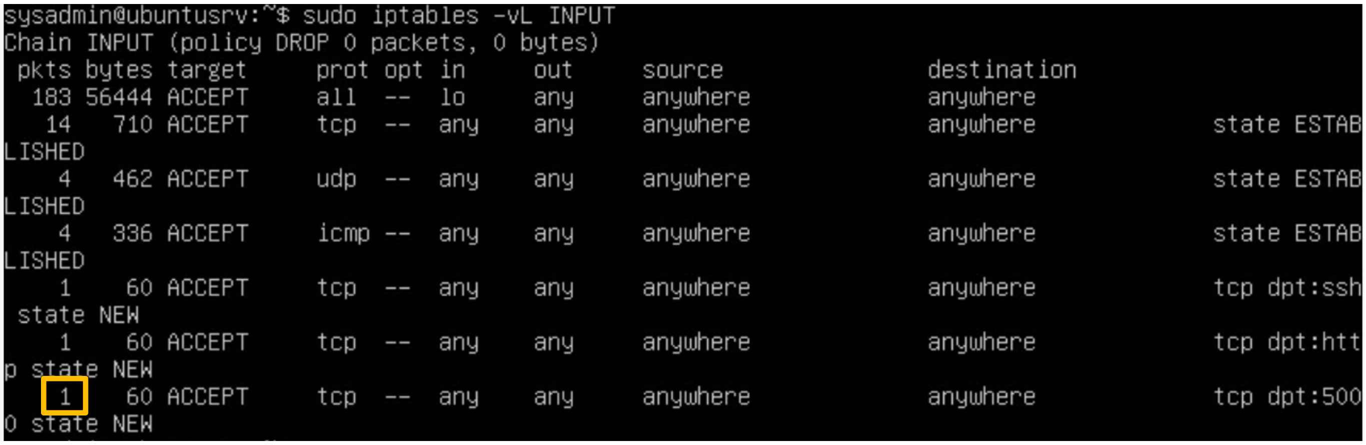
14. Return to the **UbuntuSRV** computer, and then click on **ALT+F2** to change to the second session and confirm the word **hello** has appeared in the terminal window.



```
sysadmin@ubuntusrv:~$ nc -l 5000
hello
sysadmin@ubuntusrv:~$ _
```

15. Press **Alt+F1** to return to the first session and type the following command to list *iptables* and observe the **hit counter**. Note that the **Hit Counter** has been incremented by 1 due to the traffic from the **MintOS** computer.

```
sudo iptables -vL INPUT
```



```
sysadmin@ubuntusrv:~$ sudo iptables -vL INPUT
Chain INPUT (policy DROP 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
  183 56444 ACCEPT     all  --  lo     any     anywhere             anywhere
   14   710 ACCEPT     tcp  --  any    any     anywhere             anywhere             state ESTAB
LISHED
    4   462 ACCEPT     udp  --  any    any     anywhere             anywhere             state ESTAB
LISHED
    4   336 ACCEPT     icmp --  any    any     anywhere             anywhere             state ESTAB
LISHED
    1    60 ACCEPT     tcp  --  any    any     anywhere             anywhere             tcp dpt:ssh
 state NEW
    1    60 ACCEPT     tcp  --  any    any     anywhere             anywhere             tcp dpt:htt
p state NEW
    1    60 ACCEPT     tcp  --  any    any     anywhere             anywhere             tcp dpt:500
0 state NEW
```

For the security analyst, the takeaway from this task is that ports that are open on a host, whether by design or by neglect, are vulnerable to exploitation. But, by monitoring the status of host security protections, such as *iptables* firewalls, they can find, examine, and correct problems early.

16. The lab is now complete; you may end the reservation.