FR. CONCEICAO RODRIGUES COLLEGE OF ENGINEERING

Department of Computer Engineering

1. Course, Subject & Experiment Details

Practical No:	
Title:	**Setting up personal Firewall using iptables**
Name of the Student:	**Warren Fernandes**
Roll No:	**8940**
Date of Performance:	**21-03-2022**
Date of Submission:	**21-03-2022**

Evaluation:

Sr. No.	Rubric	Grade
1	On time submission/completion (2)	
2	Preparedness (2)	
3	Skill (4)	
4	Output (2)	

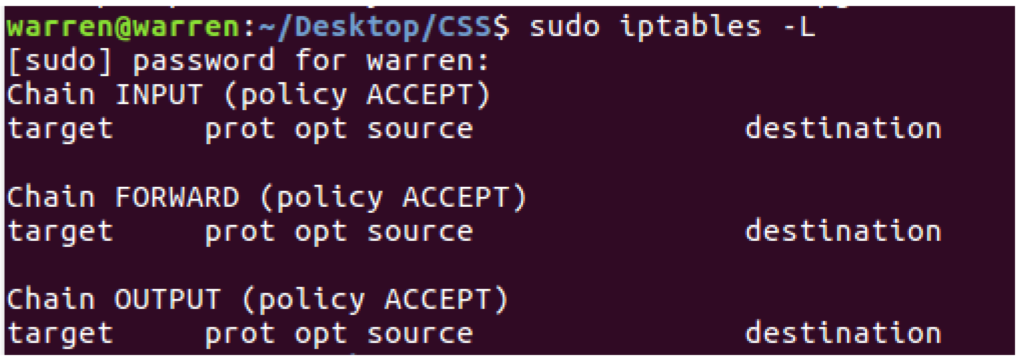
Signature of the Teacher

Setting up a good firewall is an essential step to take in securing any modern operating system. Most Linux distributions ship with a few different firewall tools that we can use to configure our firewalls.

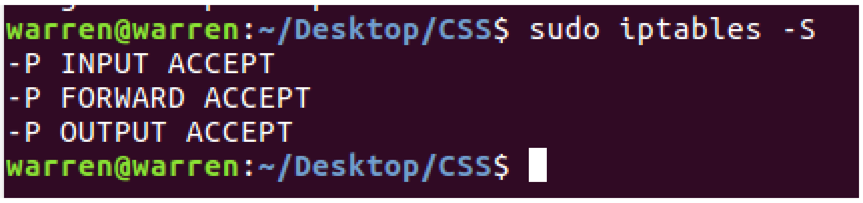
Iptables is a standard firewall included in most Linux distributions by default (a modern variant called nftables will begin to replace it). It is actually a front end to the kernel-level netfilter hooks that can manipulate the Linux network stack. It works by matching each packet that crosses the networking interface against a set of rules to decide what to do.

Basic Iptables Commands

First, you should be aware that iptables commands must be run with root privileges. A good starting point is to list the current rules that are configured for iptables.



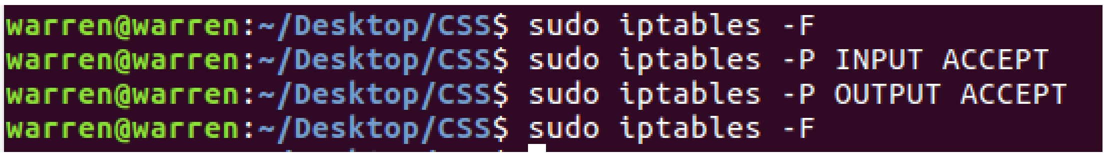
```
warren@warren:~/Desktop/CSS$ sudo iptables -L
[sudo] password for warren:
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

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



```
warren@warren:~/Desktop/CSS$ sudo iptables -S
-P INPUT ACCEPT
-P FORWARD ACCEPT
-P OUTPUT ACCEPT
warren@warren:~/Desktop/CSS$
```

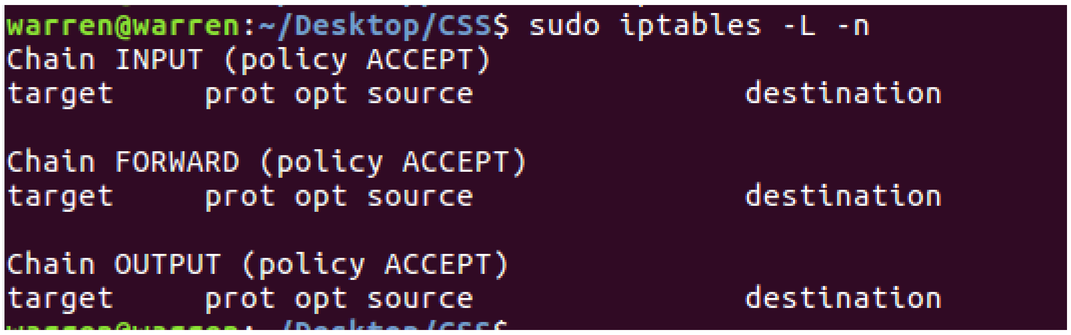
Once again, the default policy is important here, because, while all of the rules are deleted from your chains, the default policy will not change with this command. That means that you are connected remotely, you should ensure that the default policy on your INPUT and OUTPUT chains are set to ACCEPT prior to flushing your rules.

- **-i** (**interface**) — the network interface whose traffic you want to filter, such as eth0, lo, ppp0, etc.
- **-p** (**protocol**) — the network protocol where your filtering process takes place. It can be either **tcp**, **udp**, **udplite**, **icmp**, **sctp**, **icmpv6**, and so on. Alternatively, you can type **all** to choose every protocol.
- **-s** (**source**) — the address from which traffic comes from. You can add a hostname or IP address.



```
warren@warren:~/Desktop/CSS$ sudo iptables -F
warren@warren:~/Desktop/CSS$ sudo iptables -P INPUT ACCEPT
warren@warren:~/Desktop/CSS$ sudo iptables -P OUTPUT ACCEPT
warren@warren:~/Desktop/CSS$ sudo iptables -F
```

Additionally, if you want speed up your work with iptables, you can include -n in the command. This option disables DNS lookups and prevents the command from trying to find the reverse of each IP in the ruleset. You could use this to list rules, as an example:



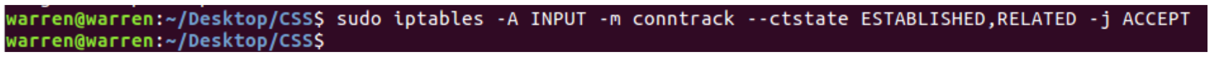
```
warren@warren:~/Desktop/CSS$ sudo iptables -L -n
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

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

Setting up a Basic Firewall

We're going to start to build our firewall policies. As we said above, we're going to be working with the INPUT chain since that is the funnel that incoming traffic will be sent through. We are going to start with the rule that we've talked about a bit above: the rule that explicitly accepts your current SSH connection.

The full rule we need is this:

sudo iptables -A INPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT



```
warren@warren:~/Desktop/CSS$ sudo iptables -A INPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT
warren@warren:~/Desktop/CSS$
```

- **-A INPUT:** The -A flag appends a rule to the end of a chain. This is the portion of the command that tells iptables that we wish to add a new rule, that we want that rule added to the end of the chain, and that the chain we want to operate on is the INPUT chain.

- **-m conntrack:** iptables has a set of core functionality, but also has a set of extensions or modules that provide extra capabilities.

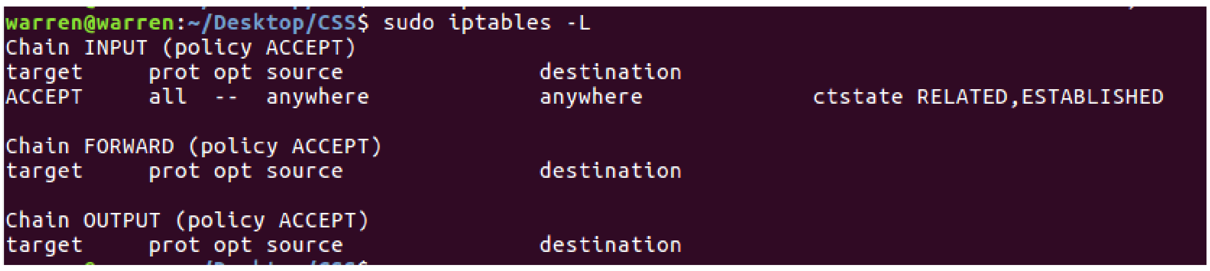
In this portion of the command, we’re stating that we wish to have access to the functionality provided by the conntrack module. This module gives access to commands that can be used to make decisions based on the packet’s relationship to previous connections.

- **-ctstate:** This is one of the commands made available by calling the conntrack module. This command allows us to match packets based on how they are related to packets we’ve seen before.

  We pass it the value of ESTABLISHED to allow packets that are part of an existing connection. We pass it the value of RELATED to allow packets that are associated with an established connection. This is the portion of the rule that matches our current SSH session.

- **-j ACCEPT:** This specifies the target of matching packets. Here, we tell iptables that packets that match the preceding criteria should be accepted and allowed through.

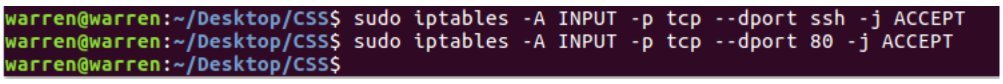
We put this rule at the beginning because we want to make sure the connections we are already using are matched, accepted, and pulled out of the chain before reaching any DROP rules.



```
warren@warren:~/Desktop/CSS$ sudo iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     all  --  anywhere             anywhere             ctstate RELATED,ESTABLISHED

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

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

Accept Other Necessary Conditions

Let’s assume that we want to block all incoming traffic, except for those coming in on 2 common ports: 22 for SSH and 80 for web traffic. We proceed by allowing all traffic on the designated ports with the following commands:



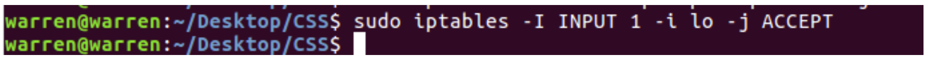
```
warren@warren:~/Desktop/CSS$ sudo iptables -A INPUT -p tcp --dport ssh -j ACCEPT
warren@warren:~/Desktop/CSS$ sudo iptables -A INPUT -p tcp --dport 80 -j ACCEPT
warren@warren:~/Desktop/CSS$
```

In both commands, the **-p** option stands for the protocol with which the connection is being made, in this case **tcp**, while the **--dport** specifies the port through which the packet is being transmitted (the destination port number of a protocol, such as **22** (**SSH**), **443** (**https**), etc.)

There is one more accept rule that we need to ensure that our server can function correctly. Often, services on the computer communicate with each other by sending network packets to each other. They do this by utilizing a pseudo network interface called the loopback device, which directs traffic back to itself rather than to other computers.

So if one service wants to communicate with another service that is listening for connections on port 4555, it can send a packet to port 4555 of the loopback device. We want this type of behaviour to be allowed, because it is essential for the correct operation of many programs.

sudo iptables -I INPUT 1-I lo -j ACCEPT



```
warren@warren:~/Desktop/CSS$ sudo iptables -I INPUT 1 -i lo -j ACCEPT
warren@warren:~/Desktop/CSS$
```

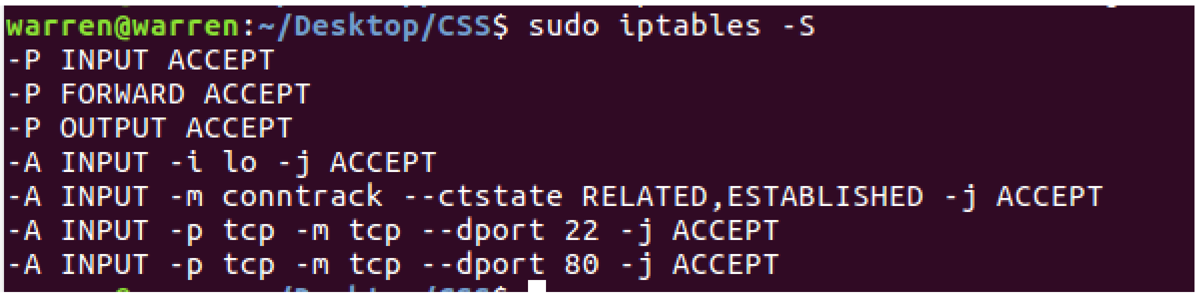
- **-I INPUT 1:** The -I flag tells iptables to insert a rule. This is different than the -A flag which appends a rule to the end. The -I flag takes a chain and the rule position where you want to insert the new rule.

  In this case, we're adding this rule as the very first rule of the INPUT chain. This will bump the rest of the rules down. We want this at the top because it is fundamental and should not be affected by subsequent rules.

- **-i lo:** This component of the rule matches if the interface that the packet is using is the "lo" interface. The "lo" interface is another name for the loopback device. This means that any packet using that interface to communicate (packets generated on our server, for our server) should be accepted.

  To see our current rules, we should use the -S flag. This is because the -L flag doesn't include some information, like the interface that a rule is tied to, which is an important part of the rule we just added:

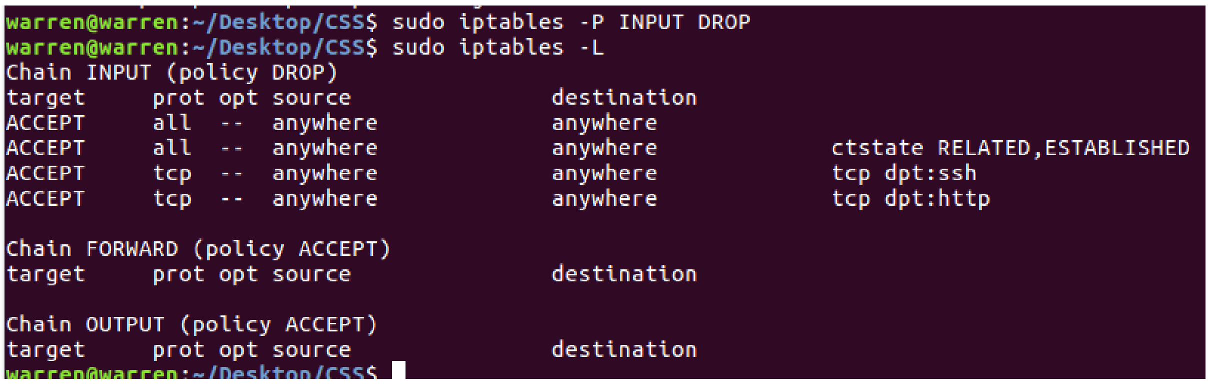
sudo iptables-S



```
warren@warren:~/Desktop/CSS$ sudo iptables -S
-P INPUT ACCEPT
-P FORWARD ACCEPT
-P OUTPUT ACCEPT
-A INPUT -i lo -j ACCEPT
-A INPUT -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT
-A INPUT -p tcp -m tcp --dport 22 -j ACCEPT
-A INPUT -p tcp -m tcp --dport 80 -j ACCEPT
```

Implementing a Drop Rule

We now have four separate rules that explicitly accept packets based on certain criteria. However, our firewall currently is not blocking anything.

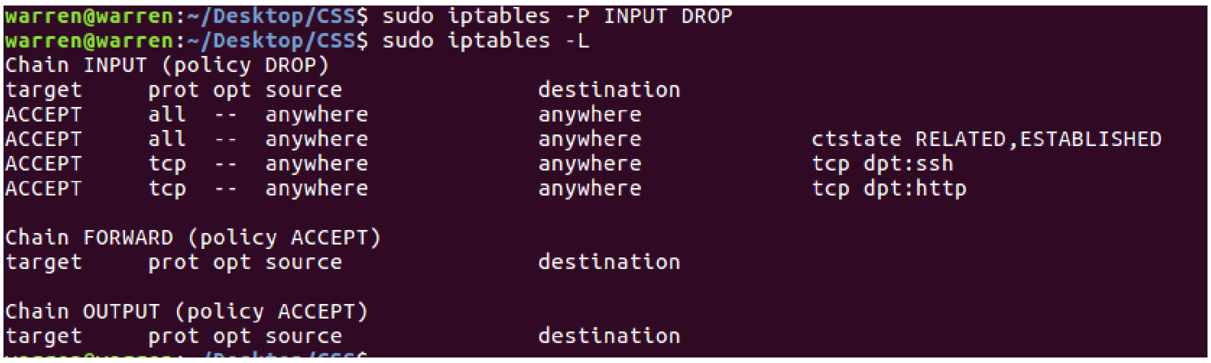


```
warren@warren:~/Desktop/CSS$ sudo iptables -P INPUT DROP
warren@warren:~/Desktop/CSS$ sudo iptables -L
Chain INPUT (policy DROP)
target     prot opt source               destination
ACCEPT     all  --  anywhere             anywhere
ACCEPT     all  --  anywhere             anywhere             ctstate RELATED,ESTABLISHED
ACCEPT     tcp  --  anywhere             anywhere             tcp dpt:ssh
ACCEPT     tcp  --  anywhere             anywhere             tcp dpt:http

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
warren@warren:~/Desktop/CSS$
```

If a packet enters the INPUT chain and doesn't match one of the four rules that we made, it is being passed to our default policy, which is to accept the packet anyways. We need to change this.

There are two different ways that we can do this, with some pretty important differences. The first way we could do this is to modify the default policy of our INPUT chain. We can do this by typing:

sudo iptables -P INPUT DROP



```
warren@warren:~/Desktop/CSS$ sudo iptables -P INPUT DROP
warren@warren:~/Desktop/CSS$ sudo iptables -L
Chain INPUT (policy DROP)
target     prot opt source               destination
ACCEPT     all  --  anywhere             anywhere
ACCEPT     all  --  anywhere             anywhere             ctstate RELATED,ESTABLISHED
ACCEPT     tcp  --  anywhere             anywhere             tcp dpt:ssh
ACCEPT     tcp  --  anywhere             anywhere             tcp dpt:http

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

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

This will catch any packets that fall through our INPUT chain, and drop them. This is what we call a default drop policy. One of the implications of this type of a design is that it falls back on dropping packets if the rules are flushed.

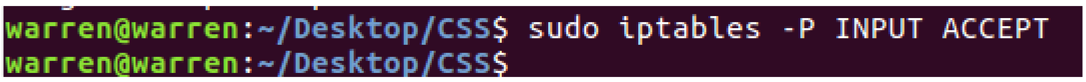
This may be more secure, but also can have serious consequences if you don't have another way of accessing: server.

You may like your server to automatically drop all connections in the event that the rules are dumped. This would prevent your server from being left wide open. This also means that you can easily append rules to the bottom of the chain easily while still dropping packets as you'd like.

The alternative approach is to keep the default policy for the chain as accept and add a rule that drops every remaining packet to the bottom of the chain itself.

If you changed the default policy for the INPUT chain above, you can set it back to follow along by typing:

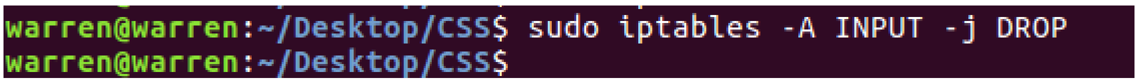
sudo iptables -P INPUT ACCEPT



```
warren@warren:~/Desktop/CSS$ sudo iptables -P INPUT ACCEPT
warren@warren:~/Desktop/CSS$
```

Now, you can add a rule to the bottom of the chain that will drop any remaining packets:

sudo iptables -A INPUT -j DROP



```
warren@warren:~/Desktop/CSS$ sudo iptables -A INPUT -j DROP
warren@warren:~/Desktop/CSS$
```

The result under normal operating conditions is the same as a default drop policy. This rule works by matching every remaining packet that reaches it. This prevents a packet from ever dropping all the way through the chain to reach the default policy.

Basically, this is used to keep the default policy to accept traffic. That way, if there are any problems and the rules are flushed, you will still be able to access the machine over the network. This is a way of implementing a default action without altering the policy that will be applied to an empty chain.

Filtering Packets Based on Source

Iptables allows you to filter packets based on an IP address or a range of IP addresses. You need to specify it after the **-s** option. For example, to accept packets from **192.168.1.3**, the command would be:

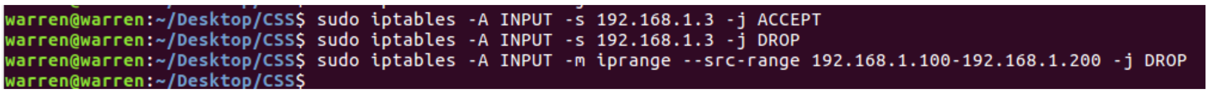
sudo iptables -A INPUT -s 192.168.1.3 -j ACCEPT

You can also reject packets from a specific IP address by replacing the **ACCEPT** target with **DROP**.

sudo iptables -A INPUT -s 192.168.1.3 -j DROP

If you want to drop packets from a range of IP addresses, you have to use the **-m** option and **iprange** module. Then, specify the IP address range with **–src-range**. Remember, a hyphen should separate the range of ip addresses without space, like this:

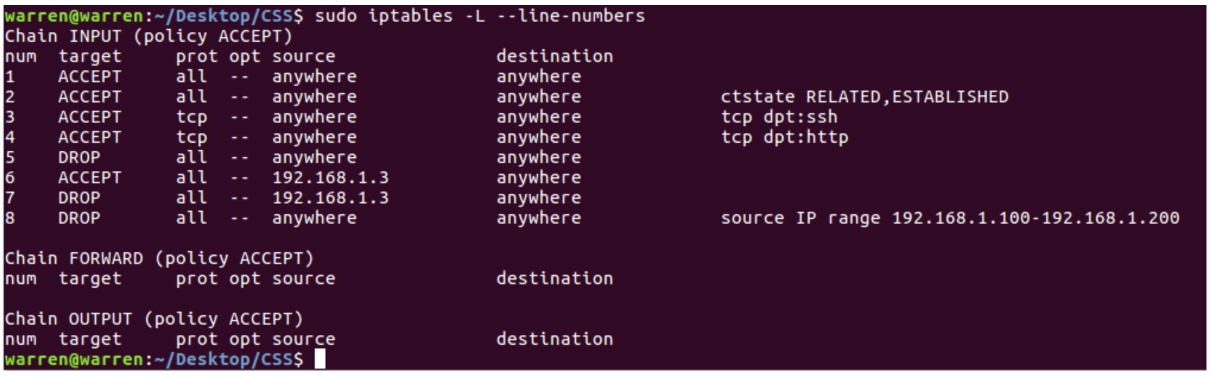
sudo iptables -A INPUT -m iprange --src-range 192.168.1.100-192.168.1.200 -j DROP



```
warren@warren:~/Desktop/CSS$ sudo iptables -A INPUT -s 192.168.1.3 -j ACCEPT
warren@warren:~/Desktop/CSS$ sudo iptables -A INPUT -s 192.168.1.3 -j DROP
warren@warren:~/Desktop/CSS$ sudo iptables -A INPUT -m iprange --src-range 192.168.1.100-192.168.1.200 -j DROP
warren@warren:~/Desktop/CSS$
```

To see all the available rules by enter the following command:

sudo iptables -L --line-numbers



```
warren@warren:~/Desktop/CSS$ sudo iptables -L --line-numbers
Chain INPUT (policy ACCEPT)
num  target     prot opt source               destination
1    ACCEPT     all  --  anywhere             anywhere
2    ACCEPT     all  --  anywhere             anywhere             ctstate RELATED,ESTABLISHED
3    ACCEPT     tcp  --  anywhere             anywhere             tcp dpt:ssh
4    ACCEPT     tcp  --  anywhere             anywhere             tcp dpt:http
5    DROP       all  --  anywhere             anywhere
6    ACCEPT     all  --  192.168.1.3          anywhere
7    DROP       all  --  192.168.1.3          anywhere
8    DROP       all  --  anywhere             anywhere             source IP range 192.168.1.100-192.168.1.200

Chain FORWARD (policy ACCEPT)
num  target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
num  target     prot opt source               destination
warren@warren:~/Desktop/CSS$
```

Saving Iptables Rules

Although the iptables rules are effective, they will automatically be deleted if the server reboots. To make sure that they remain in effect, we can use a package called IP-Tables persistent.

We can install it using apt-get:

sudo apt-get install iptables-persistent

During the installation, you will be asked if you want to save the iptables rules to both the IPv4 rules and the IPv6 rules. Say yes to both.

Your rules will then be saved in /etc/iptables/rules.v4 and /etc/iptables/rules.v6.

Once the installation is complete, start iptables-persistent running:

sudo service netfilter-persistent start

After any server reboot, you will see that the rules remain in place.