

## Tutorial 4

### Practical: Firewall configuration: getting started with iptables

#### Objectives

See separate document on Moodle for the specification for an iptables exercise that you are required to complete. This document provides some background on understanding iptables and how to use it, with examples.

iptables is a program that allows the system administrator to configure packet filtering rules on network interfaces of a Linux system.

#### Getting setup on Linux

To do this practical, you need to have root privileges on a Linux system. This may for example be a local Linux machine or VM or an Amazon EC2 instance. Access to a VMware licence or Amazon EC2 is available on request.

#### Quick reference: main iptables commands and options

| <i>Command</i> | <i>Meaning</i>                |
|----------------|-------------------------------|
| -P             | Policy (the default rule)     |
| -A             | Append                        |
| -I             | Insert (specify line number)  |
| -D             | Delete (specify line number)  |
| -R             | Replace (specify line number) |
| -F             | Flush (delete all)            |
| -L             | list (show tables)            |

| <i>Option</i> | <i>Meaning</i>                   |
|---------------|----------------------------------|
| -j            | "jump" (indicate ACCEPT or DROP) |
| -p            | protocol                         |
| -s            | source address                   |
| -d            | destination address              |
| -i            | inbound interface                |
| -o            | outbound interface               |
| -v            | verbose                          |
| -n            | numeric                          |

| <i>Option</i> | <i>Meaning</i>    | <i>Condition</i>              |
|---------------|-------------------|-------------------------------|
| --sport       | source port       | If protocol is tcp or udp     |
| --dport       | destination port  | If protocol is tcp, udp, sctp |
| --icmp-type   | ICMP message type | If protocol is icmp           |

### Part 1: Start with a clean (empty) iptables configuration.

If you are starting on a fresh VM where you have not previously set iptables rules then you do not need to do this step. However when you have started experimenting with iptables you will want to revert back to where you started to clean things up and to carry out the exercise. At any time, you can enter the following sequence of commands to get back to where you started by clearing all rules you have set and allow everything in and out:

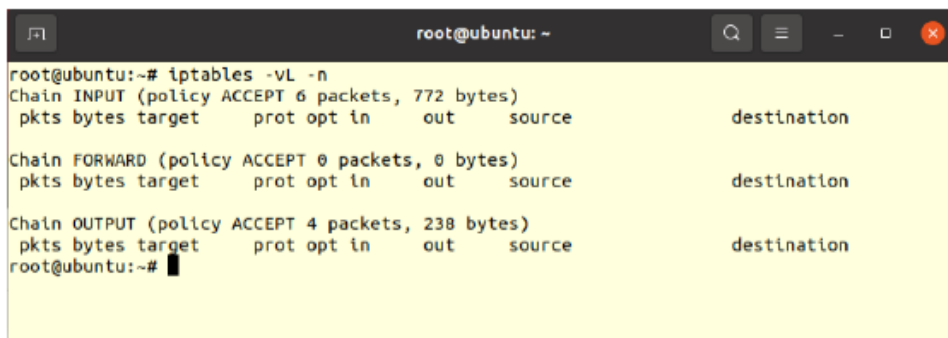
```
sudo su      # enter this to become root (if you are not already root)
iptables -F
iptables -F
iptables -F
```

### Part 2: Practice with iptables – follow tutorial linked here.

Start by going through Parts 1 and 2 of this short tutorial on the basics of iptables (you can skip Part 3 on persisting changes as we will not do this: <https://www.hostinger.com/tutorials/iptables-tutorial>)

To list the current configuration of the default filter table, enter the command:

```
iptables -L -v -n
```



```
root@ubuntu:~# iptables -L -v -n
Chain INPUT (policy ACCEPT 6 packets, 772 bytes)
 pkts bytes target    prot opt in     out     source   destination
Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target    prot opt in     out     source   destination
Chain OUTPUT (policy ACCEPT 4 packets, 238 bytes)
 pkts bytes target    prot opt in     out     source   destination
root@ubuntu:~#
```

The filter table in iptables has three chains (sets of rules), all initially empty:

- The INPUT chain is used for packets coming into the system.
- The OUTPUT chain is for packets leaving the system.
- The FORWARD chain is for packets that are routed through the system.

We will use the INPUT and OUTPUT rule chains to filter packets entering and leaving our system.

### Part 3: Working with iptables – some principles.

#### Decide on your default policy on each chain – “ACCEPT” or “DROP”

There are two main approaches:

- (i) DROP everything by default and use rules to specify what exceptions to ACCEPT
- (ii) ACCEPT everything by default and use rules to specify what exceptions to DROP

There is no point in having ACCEPT as default and then specifying ACCEPT rules. Likewise there is no point in having BLOCK as default and then specifying BLOCK rules.

For a new configuration, the policy is initially set to ACCEPT for each of the chains, so you need to set a DROP policy on either INPUT or OUPUT to take the first approach (DROP by default).

### Test each rule you implement.

For each rule you implement, you should do the following:

- (i) View the iptables configuration before you add the new rule.
- (ii) Test the functionality you wish to allow or block.
- (iii) Enter the new iptables rule.
- (iv) View the updated iptables configuration.
- (v) Test the functionality you allowed or blocked to ensure that it has changed as expected.

For example, say we have a current default policy that accepts everything inbound but drops everything outbound and we want to add a rule to accept outbound DNS queries; i.e. we are starting with:

```
iptables -P INPUT ACCEPT
iptables -P OUTPUT DROP
```

- (i) Enter `iptables -L -vn` to view the iptables configuration
- (ii) Try a DNS request and verify that it doesn't work (e.g. `nslookup www.setu.ie`)
- (iii) Enter the new iptables rule  
`iptables -A OUTPUT -d 127.0.0.1 -j ACCEPT` # if loopback not already allowed  
`iptables -A OUTPUT -p udp --dport 53 -j ACCEPT`
- (iv) Enter `iptables -L -vn` again to inspect the iptables configuration and verify that the new rules appears as expected
- (v) Try a DNS request again to verify that it works now (e.g. `nslookup www.setu.ie`)

### You need to consider both directions in two-way communications.

Most traffic is client-server – i.e. a client like a web browser sends a request to a server (e.g. a HTTP request) and subsequently gets a response (a HTTP response). As iptables is not stateful when writing a rule you need to think about both requests and responses.

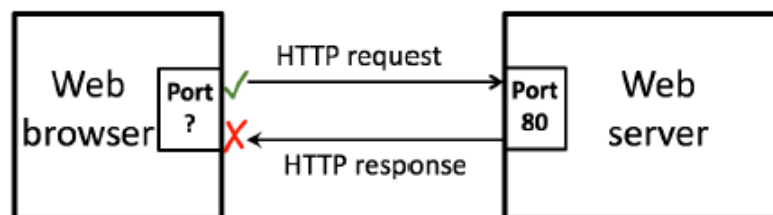
For example, let's say we decide to have a policy that DROPS all INPUT by default but ACCEPTs all OUTPUT by default. Now, we decide that we want to enable outbound web browsing.

In this scenario we do not need to specify an OUTPUT rule for outbound web browsing as our default OUTPUT policy is ACCEPT. However we need to add an INPUT rule so that we accept HTTP responses!

This rule will do it (for HTTP and HTTPS)

```
# allow inbound http/https responses
iptables -A INPUT -p tcp --sport 80 -j ACCEPT
iptables -A INPUT -p tcp --sport 443 -j ACCEPT
```

Without the above rule, outbound HTTP requests would get through but the inbound HTTP responses would be dropped, as illustrated here:



### Filter on server port number.

In the example above, we identify HTTP responses using the source port of packets coming back from a web server. In iptables syntax we use --sport to specify this source port.

We can usually filter based on server port number but not client port number because the port number of a server is usually a well known<sup>1</sup> fixed value (e.g. 22 for SSH, 53 for DNS, 80 for HTTP, 443 for HTTPS, ...). Client port numbers are assigned dynamically and therefore vary so we cannot filter based on these.

Note if we were filtering HTTP requests on the OUTPUT chain for some reason, we would have to use --dport as in this case the destination port (i.e. the server) is well known.

### Some iptables examples for practice.

#### Example 1

##### (a) Block ping and re-enable it

*N.B. always test access before putting rule in place, and then verify it is working after you've done it.*

Start with your default INPUT/OUTPUT policies set to ACCEPT and flush any existing rules.

```
iptables -F INPUT ACCEPT
iptables -F OUTPUT ACCEPT
iptables -F
```

Check if you can get your system to ping itself:

```
ping localhost
```

What happens? Now execute:

```
iptables -A OUTPUT -p icmp -j DROP
```

and try to ping again. What has changed?

##### (b) Delete an iptables rule

View current rules by line number

```
iptables -L --line-numbers
```

Find the line number of the line, and then remove it with:

```
iptables -D OUTPUT <line number>
```

Delete the first rule that you created above – i.e.

```
iptables -D OUTPUT 1
```

You should be able to ping again.

##### (c) Block ping on INPUT chain instead

Now, try to block ICMP on the INPUT chain instead. Verify that it works as you intended. Do you notice any difference?

### Example 2: Block a specific website

Imagine that you want to block access to [www.facebook.com](http://www.facebook.com). First verify that you can reach Facebook in your browser and also that you can ping [www.facebook.com](http://www.facebook.com). (As before, start with your default INPUT/OUTPUT policies set to ACCEPT and flush any existing rules).

The following command will block access to facebook.com

```
iptables -A OUTPUT -d www.facebook.com -j DROP
```

The -d option specifies destination (name or IP address). By omitting the -p option we are blocking all ports.

Now try to access via ping and your browser. Delete the rule again when finished testing.

### Example 3: Block by default and add exceptions to allow things through

As before, start with your default INPUT/OUTPUT policies set to ACCEPT and flush any existing rules

Now block everything inbound – i.e. make the INPUT policy DROP<sup>2</sup>

```
iptables -P INPUT DROP
```

Now the system feels completely cut off from the network. Try pinging another system or browsing to a website or doing a DNS lookup. None should work.

Make an exception for input from 127.0.0.1 (localhost). This is required for DNS to work.

```
# allow input from 127.0.0.1
iptables -A INPUT -s 127.0.0.1 -j ACCEPT
```

Always check the table to see the effect of the rule change

```
iptables -L -vn
```

And test it:

```
ping localhost
```

Now you can start allowing more services such as DNS and HTTP/HTTPS – see exercise in separate document. For example, to allow HTTP/HTTPS, the commands are:

```
# allow DNS (actually inbound DNS responses)
iptables -A INPUT -p tcp --sport 53 -j ACCEPT
iptables -A INPUT -p tcp --sport 443 -j ACCEPT
```

To understand why we are filtering based on source ports (web server), see sections 4.3 and 4.4 of this document.

To test the last example, visit a URL that does not rely on DNS (as DNS hasn't yet been enabled), for example <https://1.1.1.1/>

### Running services locally (for testing filtering of inbound requests)

To be able to demonstrate the use of iptables with internal servers, you will need to run services on the Linux machine where you are configuring iptables. You can do this by either:

- **Installing and running local servers**
  - For example you can install and run a web server such as *apache2* on Ubuntu with the commands:

```
sudo apt install apache2
sudo service apache2 start
```
  - and you can install an SSH server with the commands:

```
sudo apt install openssh-server
sudo service sshd start
```