

SECURITY OF IS  
FIREWALL AND HTTPS

---

Lab 1

---

*Author(s):*  
Gabriel PADIS  
Anastasia DUCHESNE

*Teacher:*  
MR. HAMON

# Contents

<b>1</b>	<b>Flow matric of the server</b>	<b>2</b>
<b>2</b>	<b>iptables</b>	<b>2</b>
<b>3</b>	<b>Configuring iptables</b>	<b>2</b>
<b>4</b>	<b>Stateful</b>	<b>3</b>
<b>5</b>	<b>Testing configuration</b>	<b>4</b>
5.1	HTTP . . . . .	4
5.2	Pinging server from our computer . . . . .	4
<b>6</b>	<b>Allow server ping from administrator</b>	<b>4</b>
<b>7</b>	<b>Log the connections</b>	<b>5</b>
<b>8</b>	<b>Make it permanent</b>	<b>5</b>

# 1 Flow matrix of the server

We have been given the following architecture for the server :

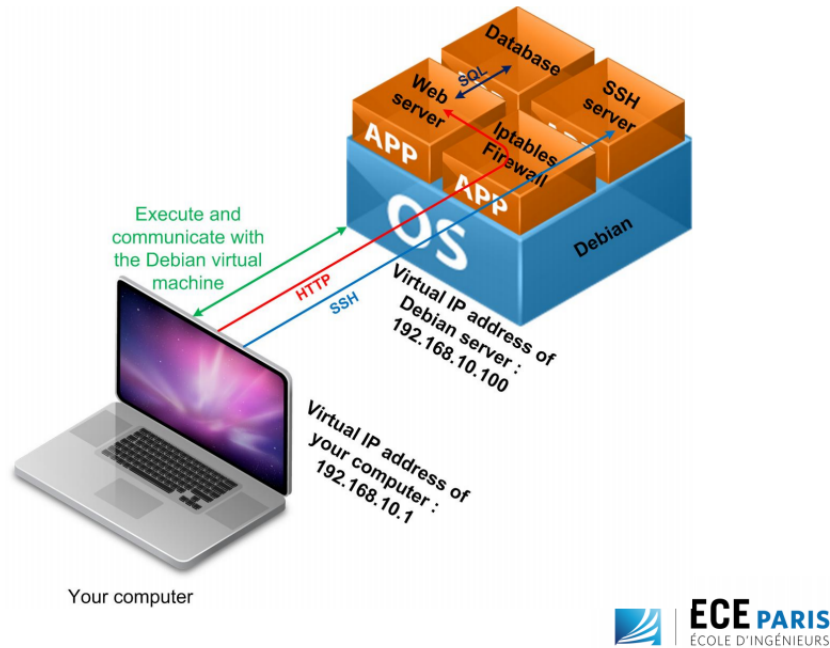


Figure 1: Architecture for the server

In order to configure a firewall that contains all the application needed, we wrote the following flow matrix :

Source IP	Source Port	Dest IP	Dest Port	Action	Description
ANY	ANY	192.168.10.100	80	ACCEPT	HTTP
127.0.0.1	3306	192.168.10.100	3306	ACCEPT	MySQL
192.168.10.100	3306	127.0.0.1	3306	ACCEPT	MySQL
192.168.10.1	ANY	192.168.10.100	22	ACCEPT	SSH from administrator
ANY	ANY	ANY	ANY	DROP	Clean-up rule

## 2 iptables

**iptables** set ups, maintains and inspects the filtering rules of the IP packets in the Linux Kernel. You can define several tables. A table is made of built-in chains (eg: OUTPUT or FORWARD) and user-defined chains. A chain is a list of rules matching a set of packets, and defining what should be done with them (reject, accept...).

## 3 Configuring iptables

In order to configure iptables, we have to use the following command line for each new rule we're adding :

```
iptables -A <chain> -s <src IP> --sport <src port> -d <dest IP>  
-p <protocol> --dport <dest port> -j <rule target>
```

**-A** or **--append** appends one or more rules at the end of INPUT, the chain we selected.

**-s** or **--source** source specification, in our case an IP address.

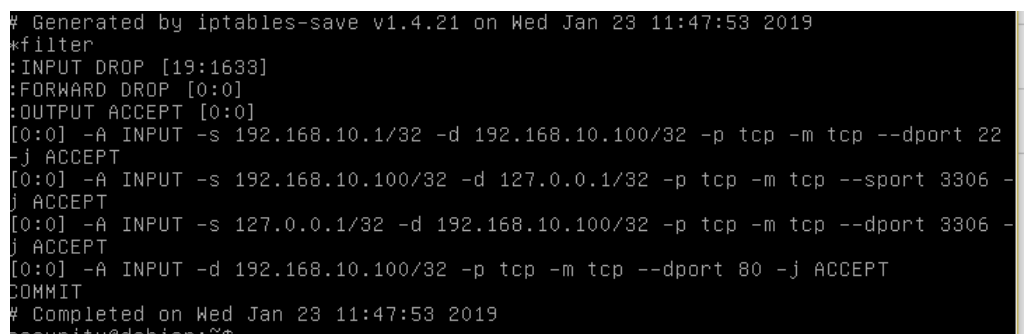
**--sport** source port specification.

**-p** or **--protocol** The protocol of the rule or the packet to check. In our case, it is TCP. Later, we'll also need icmp to allow ping.

**-d** or **--destination** destination specification, in our case an IP Address.

**--dport** destination port specification.

**-j** target specification. It is ACCEPT to allow, and DROP to drop.

A screenshot of a terminal window showing the output of the 'iptables-save' command. The output lists the current iptables rules for the 'filter' table. It includes default policies for INPUT (DROP), FORWARD (DROP), and OUTPUT (ACCEPT). There are four custom rules: 1) Allow TCP connections from 192.168.10.1/32 to 192.168.10.100/32 on port 22. 2) Allow TCP connections from 192.168.10.100/32 to 127.0.0.1/32 on port 3306. 3) Allow TCP connections from 127.0.0.1/32 to 192.168.10.100/32 on port 3306. 4) Allow TCP connections to 192.168.10.100/32 on port 80. The output ends with a COMMIT command and a timestamp.

```
# Generated by iptables-save v1.4.21 on Wed Jan 23 11:47:53 2019
*filter
:INPUT DROP [19:1633]
:FORWARD DROP [0:0]
:OUTPUT ACCEPT [0:0]
[0:0] -A INPUT -s 192.168.10.1/32 -d 192.168.10.100/32 -p tcp -m tcp --dport 22 -j ACCEPT
[0:0] -A INPUT -s 192.168.10.100/32 -d 127.0.0.1/32 -p tcp -m tcp --sport 3306 -j ACCEPT
[0:0] -A INPUT -s 127.0.0.1/32 -d 192.168.10.100/32 -p tcp -m tcp --dport 3306 -j ACCEPT
[0:0] -A INPUT -d 192.168.10.100/32 -p tcp -m tcp --dport 80 -j ACCEPT
COMMIT
# Completed on Wed Jan 23 11:47:53 2019
```

Figure 2: iptables configuration

## 4 Stateful

A **stateless firewall** treats each packet individually. It considers only the static elements of a TCP connection, and inspect the information contained in the packet header (source and destination IP address, and source and destination ports).

A **statefull firewall** also considers the state of the ongoing TCP connection. It owns a connection table. To switch to a statefull firewall, we have to add some new rules precising a connection state. To do so, we'll use the following command line :

```
iptables -A INPUT -m conntrack --ctstate <states to match> -j <rule target>
```

**-m** specifies a match to use.

**conntrack** ongoing connection tracking state access. Returns ongoing tcp connection state.

**--cstate** connection state(s) that should match the ongoing connection state (conntrack). Acts like a filter, if it matches with conntrack, it applies the rule to the ongoing connection.

**INVALID**

The packet is associated with no known connection.

**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.

Figure 3: States of `-cstate` used

Source IP	Source Port	Dest IP	Dest Port	Action	State
ANY	ANY	192.168.10.100	80	ACCEPT	NEW,ESTABLISHED, RELATED
127.0.0.1	3306	192.168.10.100	3306	ACCEPT	NEW,ESTABLISHED, RELATED
192.168.10.100	3306	127.0.0.1	3306	ACCEPT	NEW,ESTABLISHED, RELATED
192.168.10.1	ANY	192.168.10.100	22	ACCEPT	NEW,ESTABLISHED,RELATED
ANY	ANY	ANY	ANY	DROP	INVALID

## 5 Testing configuration

### 5.1 HTTP

When we try to access to access to `http://192.168.10.100/index.html` from our host machine (our computer), it does work. We have configured our firewall so it contains the application that allows our web server to receive any HTTP request from a client from the network.

### 5.2 Pinging server from our computer

Pinging the server from our computer doesn't work. We don't have any rule defined permitting ping to our server.

## 6 Allow server ping from administrator

Previously we couldn't ping the server because it didn't allow any ping (so an icmp protocol). In order to do so we wrote the following command :

```
iptables -A -s 192.168.10.1 -d 192.168.10.100 -p icmp -j ACCEPT
```

Now we can ping the server but only from the administrator's computer

```
C:\Users\Gabriel>ping 192.168.10.100

Pinging 192.168.10.100 with 32 bytes of data:
Reply from 192.168.10.100: bytes=32 time<1ms TTL=64
Reply from 192.168.10.100: bytes=32 time<1ms TTL=64
Reply from 192.168.10.100: bytes=32 time<1ms TTL=64
Reply from 192.168.10.100: bytes=32 time<1ms TTL=64

Ping statistics for 192.168.10.100:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 0ms, Average = 0ms
```

Figure 4: Pinging server from admin

## 7 Log the connections

In order to log the connections we had to :

- Add a new chain named LOGGING
- At the end of the INPUT chain jump to LOGGING
- Limit to 2 per min and log the denied connection
- Drop the log connection

So we did the following commands:

```
iptables -N LOGGING
iptables -A INPUT -j LOGGING
iptables -A LOGGING -m limit --limit 2/min -j LOG\
    --log-prefix "IPTables-Dropped: " --log-level 4
iptables -A LOGGING -j DROP
```

## 8 Make it permanent

To make it permanent we add to tell him that every time it was downed it needed to save our configuration in the file **/etc/iptables.rules**. Every time it was up it need to restore from that same file.

So we added the following lines to the **/etc/network/interfaces** file :

```
pre-up iptables-restore < /etc/iptables.rules
post-down iptables-save > /etc/iptables.rules
```