

# Sprint 2 ASIST

## Table of contents

Context .....	1
Installation of the needed packages .....	1
Configuration of the firewall .....	1
Modification of the firewall rules.....	1
Checking iptables rules.....	2

## Context

We chose the user case number 2: « As the system administrator I want only clients on the DEI's internal network (cabled or via VPN) to be able to access the solution.»

The operations will be operated on a machine running debian.

## Installation of the needed packages

To control who can access or not our services, we will need the firewall iptables. To get it, we use the command `sudo apt-get install iptables`.

```
user@debian:~$ sudo apt-get install iptables
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
iptables is already the newest version (1.8.7-1).
iptables set to manually installed.
The following package was automatically installed and is no longer required:
  libeatmydata1
Use 'sudo apt autoremove' to remove it.
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
user@debian:~$
```

In our case, iptables is already installed so we got this message (see screenshot).

## Configuration of the firewall

### Modification of the firewall rules

Our service listens on two ports: 3000 and 5000. The page [rede.dei.isep.ipp.pt/myip](http://rede.dei.isep.ipp.pt/myip) informs us that the addresses belonging to the DEI internal network belongs to the network 10.8.0.0/16:

The 10.8.41.253 node address belongs to the IPv4 10.8.0.0/16 private network used in DEI laboratories wired networks, this address has been assigned by the DHCP server operating on that network

First, we must allow all the traffic from the DEI internal network. This command must be executed before the ones that block the traffic otherwise, we may not have access to the ssh service anymore. We will allow the traffic for the ssh service (Port 22), and the 2 ports of our program (3000 and 5000).

```
sudo iptables -I INPUT -p tcp --dport 22 -s 10.8.0.0/16 -j ACCEPT
```

```
sudo iptables -I INPUT -p tcp --dport 5000 -s 10.8.0.0/16 -j ACCEPT
```

```
sudo iptables -I INPUT -p tcp --dport 3000 -s 10.8.0.0/16 -j ACCEPT
```

Then, we must allow all already established TCP connections by using this command:

```
sudo iptables -A INPUT -m conntrack --ctstate ESTABLISHED,RELATED -s 10.8.0.0/16 -j ACCEPT
```

Then, we allow the same type of connection, but for the outgoing traffic:

```
sudo iptables -A OUTPUT -m conntrack --ctstate ESTABLISHED -s 10.8.0.0/16 -j ACCEPT
```

Without these commands, the traffic would be interrupted.

Finally, we block all the ingoing traffic that don't respond to the previous criteria:

```
sudo iptables -A INPUT -j DROP
```

## Checking iptables rules

We can check our rules with the command `sudo iptables -L`:

```
user@debian:~$ sudo iptables -L --line-numbers
Chain INPUT (policy ACCEPT)
num target      prot opt source                destination              tcp dpt:5000
1  ACCEPT        tcp  --  10.8.0.0/16            anywhere
2  ACCEPT        tcp  --  10.8.0.0/16            anywhere                  tcp dpt:3000
3  ACCEPT        tcp  --  10.8.0.0/16            anywhere                  tcp dpt:ssh
4  ACCEPT        all  --  10.8.0.0/16            anywhere                  ctstate RELATED,ESTABLISHED
5  DROP          all  --  anywhere               anywhere

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

Chain OUTPUT (policy ACCEPT)
num target      prot opt source                destination              ctstate RELATED,ESTABLISHED
1  ACCEPT        all  --  10.8.0.0/16            anywhere
```