# COMPUTER NETWORKS

LAB LIST 9

## 1 General remarks

Before you begin run `netmode lab` command and then configure `eth0` using *DHCP* (`dhclient eth0`). Check that the resulting IP address belongs to `172.16.1.0/24` subnet. During today's workshops computers are connected with `eth0` interface to the switch, adjacent pairs of computers are connected with `eth1` card.

To configure firewall use `iptables` command. The firewall itself is simply a list of rules, which are applied from beginning to end (the order is important!). The configuration can be performed interactively by adding the appropriate entries to the rule tables or removing items from those tables. However, it is more convenient to use the following approach:

- Create `firewall.sh` file and put commands at the beginning to clear the list of rules.
- Following commands from `firewall.sh` call `iptables` program in order to append rules to the tables.

Then, if you execute `firewall.sh` script, firewall rule tables will look exactly as written in the file.

## 2 Exercises

**Exercise 1.** In this exercise, you will create basic set of firewall rules. Create `firewall.sh` file with following command:

```
$> touch firewall.sh
```

... and change its permission to make it executable:

```
$> chmod u+x firewall.sh
```

Then type in the following lines skipping the numbers and the following colons (i.e. the first line should be `#!/bin/bash`).

```
1: #!/bin/bash
2: set -x
3:
4: sudo iptables -F
5: sudo iptables -t nat -F
6:
7: sudo iptables -P INPUT DROP
8: sudo iptables -P FORWARD DROP
```

```
 9: sudo iptables -P OUTPUT ACCEPT
10:
11-15: # firewall rules here
16:
17-21: # NAT rules here
22:
23: sudo iptables -A INPUT -j LOG --log-prefix "blocking INPUT "
24: sudo iptables -A FORWARD -j LOG --log-prefix "blocking FORWARD "
```

After typing in all commands into `firewall.sh` file, run the script:

```
$> ./firewall.sh
```

And then view the current configuration of the firewall:

```
#> iptables -L -n
#> iptables -t nat -L -n
```

The commands should be issued after *each* edit of `firewall.sh` file. (Following exercises will not mention that.)

In another console issue command (and leave it running):

```
$> tail -f /var/log/messages
```

It will display finishing lines of the log file, in particular dropped or rejected packets.

Now try to connect to SSH server with following command:

```
$> ssh eagle-server.example.com
```

Have you managed to establish a connection? Watching the log file can you identify the cause of the problem – i.e. why responses from SSH server are filtered out?

**Exercise 2.** Allow in the packets that belong to already established connections by adding following rule to `firewall.sh` file:

```
11: sudo iptables -A INPUT -m state --state ESTABLISHED -j ACCEPT
```

Check the connection to SSH server again:

```
$> ssh eagle-server.example.com
```

This time it should work. See if you can connect to your neighbor's SSH server:

```
$> ssh your_neighbor_address
```

What appears in your log file and what is in his / her log file? The connection should not succeed, cause any attempt to connect to port 22 is rejected. For the same reason, you should not be able to connect to yours SSH server with:

```
$> ssh 127.0.0.1
```

To fix that, allow in any incoming connection on *local* interface and any SSH connection from the outside. Add following lines to `firewall.sh` file:

```
12: sudo iptables -A INPUT -i lo -j ACCEPT
13: sudo iptables -A INPUT -p tcp --dport 22 -j ACCEPT
```

Try again to connect via SSH with your neighbor and yourself.

**Exercise 3.** Try pinging yours and your neighbor's computer. Ping packets should be discarded by your neighbor's computer. Check out messages in the log file. To let `ICMP` packets in (these packets are used by the ping – namely *ICMP echo request*), add the following lines to `firewall.sh` file:

```
14: sudo iptables -A INPUT -p icmp --icmp-type echo-request -j LOG
    --log-prefix "Somebody pings! "
15: sudo iptables -A INPUT -p icmp --icmp-type echo-request -j ACCEPT
```

(Line 14 should be one line, it's broken into two to fit on the sheet.)

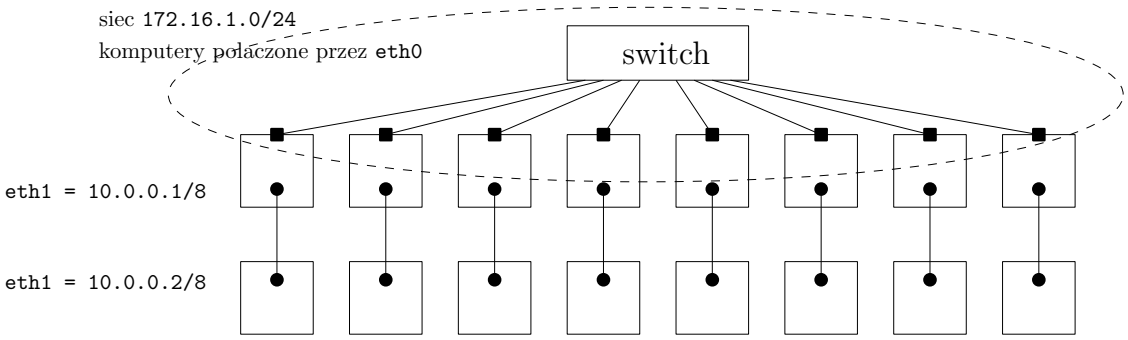
Line 14 will record pings in the log file. `-j LOG` rule is peculiar: even if it is applied to a packet, the following rules are still considered. Check the messages appearing in the log files (your and your neighbor's) as you ping each other, and when you ping your own computers. Display firewall statistics with following command:

```
#> iptables -L -nv
```

How many packets were dropped and how many passed through some other rules? At this point rows 11-15 in `firewall.sh` file should be already filled in.

**Exercise 4.** Use `nmapfe` program to find out what ports are open on the computer next to you (enter its IP address in *Target* field). Select *Intense scan* in *Profile* field. Read the information being displayed by `nmapfe`. Sniffing packets in `wireshark` and messages in the log file about blocked packets, check which ports `nmapfe` tried to connect to.

**Exercise 5.** In this and the next exercise we will use `eth1` interface. The exercise should be carried out in pairs. One computer will play the role of a *NAT router* and the other of a *workstation*. Our aim is to obtain a working configuration shown in the figure below (the first tier is a NAT router and others – workstations).




We will not add any more commands to firewall on the workstation. For them you should disable `eth0` interface and configure `eth1` interface as follows:

```
#> ifconfig eth0 down
#> ifconfig eth1 10.0.0.2 up
```

Configure `eth1` interfaceof the NAT router by assigning `10.0.0.1` IP address:

```
#> ifconfig eth1 10.0.0.1 up
```

Check if the computers can see each other through `eth1` interface by using `ping` command.

Note that addresses from `172.16.1.0/24` network uniquely identify `eth0` interfaces of NAT routers, but same addresses from `10.0.0.0/8` network are assigned to multiple computers. Firstly, we will show possible problems caused by such setup.

On the workstation set nearby NAT router as a default gateway:

```
#> route add default gw 10.0.0.1
```

Lets denote the IP address of nearby NAT router `eth0` interface by *A* and an IP address of some other NAT router by *B*. Try pinging addresses *A* and *B* from your workstation. Observe what appears in the log file of neighbor NAT router. It turns out, that blocking traffic through a NAT router is to blame. Fix this by adding following lines to `firewall.sh` file on the NAT router:

```
17: sudo iptables -A FORWARD -m state --state ESTABLISHED -j ACCEPT
18: sudo iptables -A FORWARD -i eth1 -o eth0 -j ACCEPT
```

The second rule forwards all passing traffic originating from the workstation. The first one forwards all packets belonging to already established connections.

```
19: sudo iptables -t nat -A POSTROUTING -o eth0 -j SNAT --to 172.16.1.X
```

where `172.16.1.`*X* address is the IP address assigned to `eth0` card. Try again pinging addresses *A* and *B* from the workstation. Launch `wireshark` on the NAT router *B* and observe source IP addresses of incoming ping packets (it looks like router *A* sends ping packets, but the TTL field is one less than normal). This time both pings should succeed. Now, check on NAT router *A* that all packets are recorded twice in `wireshark`: before and after IP address mangling.

**Exercise 6.** In previous exercises we configured computers in such a way that workstations (representing local network) were able to connect to `172.16.1.0/24` network (representing the Internet). But what to do if you want to connect to the workstation (e.g. through SSH)?

One method is to redirect a port (e.g. 2222) of the NAT router to port 22 of workstation adjacent to the router by adding following lines to `firewall.sh` file:

Try again pinging addresses of *A* and *B* from the workstation. Check in `wireshark` that for *B*, the reply to ping packet sent by your workstation is received by workstation adjacent to router *B*! Why is this happening? To fix that we will enable source NAT in the NAT routers by appending following lines to `firewall.sh` file:

```
20: sudo iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 2222
    -j DNAT --to 10.0.0.2:22
21: sudo iptables -A FORWARD -i eth0 -o eth1 -p tcp -d 10.0.0.2
    --dport 22 -j ACCEPT
```

The first rule changes the destination IP address to `10.0.0.2`, and the second one forwards traffic through the NAT router, provided that it is addressed to port 22 of the workstation. (Rules of `nat` table `PREROUTING` chain are examined before FORWARD chain, so the addresses have already been mangled).

Check that every NAT router and workstation are able to connect via SSH to your workstation. In both cases, the command should look like this:

```
$> ssh -p 2222 IP_of_NAT_router
```

where *IP_of_NAT_router* is the address of `eth0` interface of NAT router adjacent to the workstation to which you want to connect via `SSH` Observe transmitted packets in `wireshark`. Examine what has been written out to a log file.

*Marcin Bieńkowski*
Translation: *Krystian Bacławski*