Department of Information Engineering (DEI)
Master degree on ICT for Internet and Multimedia Engineering (MIME)

INTERNET – LAB Experience 5: Firewall

First Example

In this experience, we are going to use Kathará to learn about firewalls, using the Linux tool `iptables` (see User Guide). The first Kathará lab is set up as in Figure 1, and the firewall resides on the same machine we want to protect. What we want to do is to allow the computer to access any Internet service, while blocking access (except for `ping` and `sh`) from any other machine.

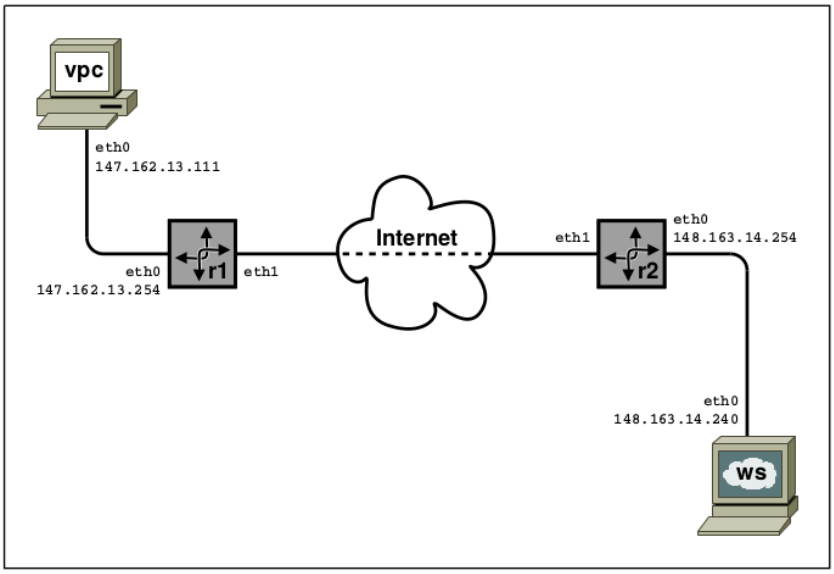

Figure 1: Network topology configuration details for the first example.

First of all, we need to start the machines and set up the connections between them. You can see the "Internet" as a network between r1 and r2, and you can assign any valid IP configuration to such network. Then, you need to start two Apache web servers on the WS and VPC machines using the command `/etc/init.d/apache2 start`. Also, you need to start an SSH daemon on VPC using the command `/etc/init.d/ssh start`. After the SSH daemon has started, it is important to enable root operations and set up a password. In order to do so, from VPC you need to:

- Modify the `/etc/ssh/sshd_config` file and change the line `PermitRootLogin` to `yes`. Any screen-oriented text editor (e.g., `vim, vi, nano`) can be used for such operation.
- Restart SSH using the command `/etc/init.d/ssh restart`.
- Set up a password using the command `passwd`. You will be asked to choose a password. In this example, let's choose `rootpwd`.

Once the configuration is completed, you can test the connectivity by opening the web pages on WS and VPC.

Exercise 1: *Write* `links 127.0.0.1` *to access the webpage on VPC from VPC itself. What do you see? What happens if you connect to VPC from WS? You can quit lynx using the* `q` *and* `y` *keys.*

You can also SSH into VPC from WS by typing `ssh root@147.162.13.111`. The root password for VPC is `rootpwd` and you can exit from SSH by pressing `Ctrl+D`.

When you're verified that everything works properly, we can start to set up the firewall. By default, `iptables` starts out unconfigured, with all policies set to `ACCEPT`; however, if you need to clear a table later, you can type the following commands:

```
iptables --table filter --flush
iptables --table filter --delete-chain
```

These commands work for the `filter` table, but you can swap in the name of any other table. Let's start by setting up the default policies for the `input` and `output` chains of the `filter` table:

```
iptables --table filter --policy INPUT DROP
iptables --table filter --policy OUTPUT ACCEPT
```

To avoid breaking local processes, we need to add an exception for the loopback interface:

```
iptables --table filter --append INPUT --in-interface lo --jump ACCEPT
```

We can now verify that VPC will not answer any pings, and its web service is unavailable: all incoming traffic from outside is blocked. In order to allow `ping` and `ssh` requests, we need to add two exceptions:

```
iptables --table filter --append INPUT --protocol icmp --icmp-type echo-request --jump ACCEPT
iptables --table filter --append INPUT --protocol tcp --destination-port 22 --jump ACCEPT
```

Port 22, which is used by SSH, is now open; we can verify that it is possible to SSH into VPC, while its web service is still unavailable. However, VPC still can't access the web service provided by WS, as its incoming packets are blocked. We need to tell the firewall to accept packets for "conversations" initiated by VPC itself:

```
iptables --table filter --append INPUT --match state --state ESTABLISHED,RELATED --jump ACCEPT
```

Second Example

In this exercise, we are going to connect more PCs: LANPC1 and LANPC2 are machines connected to a LAN, on which LANWS provides a public web service. We want to configure R1 to allow incoming traffic to LANWS, but not to the other two machines, as well as letting them connect to the internet. We also want to block all incoming SSH traffic, except for requests from LANPC1 and LANPC2 to R1 (so that we can manage the firewall from inside the local network). The connections are shown in Figure 2.

First of all, we can start the lab, which also sets up the web servers and SSH daemon on R1. R1's root password is still `rootpwd`.

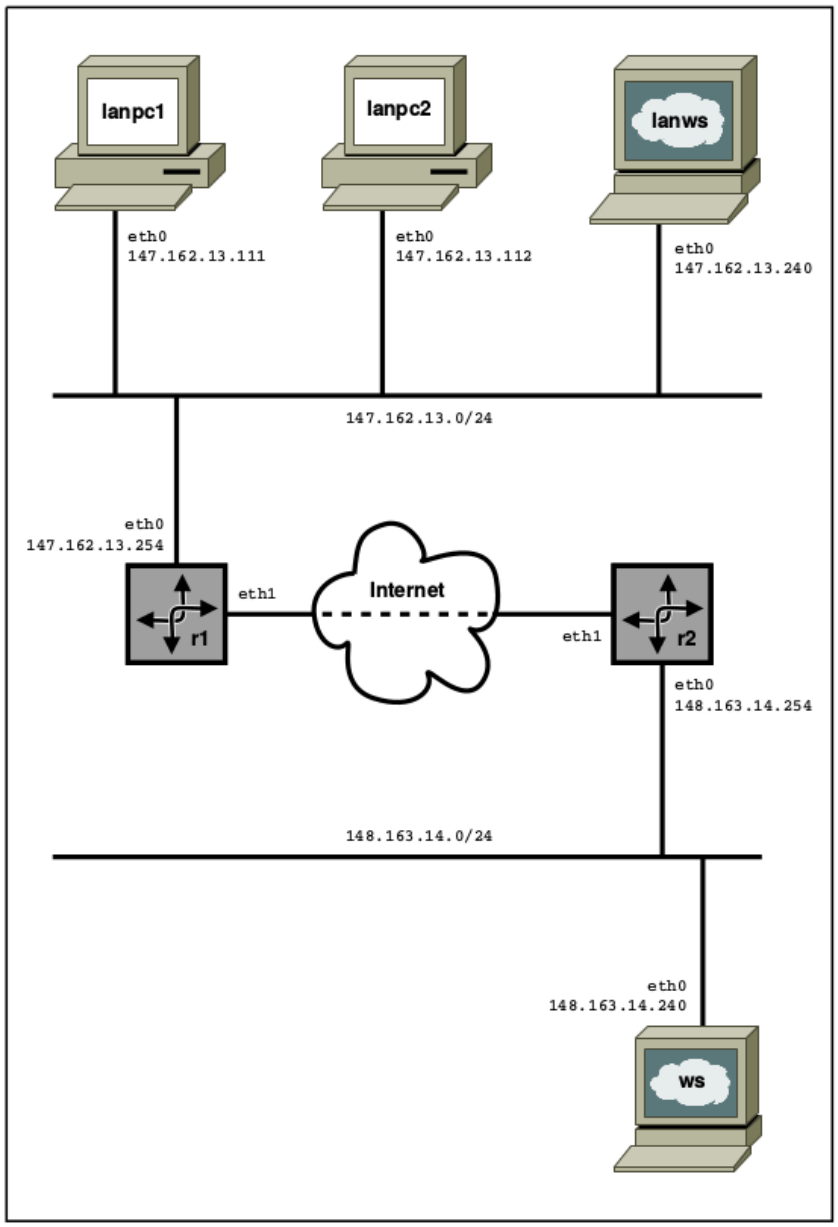

Figure 2: Network topology configuration details for the second example.

Exercise 2: *Try and SSH into R1 from WS: is it accessible from outside? Can WS ping LANPC1?*

We now set up the same default policies on R1:

```
iptables --table filter --policy INPUT DROP
iptables --table filter --policy OUTPUT ACCEPT
iptables --table filter --append INPUT --in-interface lo --jump ACCEPT
```

At this point, R1 does not accept any traffic from outside, so R1 itself is not reachable by any other machine in the network. However, the LAN is still connected to the Internet: to block traffic to and from it, we need to add another rule to the `FORWARD` chain.

```
iptables --table filter --policy FORWARD DROP
```

Now the LAN is effectively disconnected from the internet, on both sides. The next step is to allow traffic from the LAN to the outside:

```
iptables --table filter --append FORWARD --in-interface eth0 --out-interface eth1 --jump ACCEPT

iptables --table filter --append FORWARD --in-interface eth1 --out-interface eth0 --match state
--state ESTABLISHED,RELATED --jump ACCEPT
```

Exercise 3: *Try to connect from LANPC1 to WS: does it work now? Can WS access the LANWS web service?*

To make LANWS accessible, we need to allow incoming connections to its HTTP port:

```
iptables --table filter --append FORWARD --protocol tcp --destination-port 80 --in-interface eth1
--out-interface eth0 --destination 147.162.13.240 --jump ACCEPT
```

The last step is to allow incoming SSH traffic from LANPC1 and LANPC2, to open the management interface:

```
iptables --table filter --append INPUT --protocol tcp --destination-port 22 --in-interface eth0
--source 147.162.13.111 --jump ACCEPT

iptables --table filter --append INPUT --protocol tcp --destination-port 22 --in-interface eth0
--source 147.162.13.112 --jump ACCEPT
```

Note that LANWS cannot access the router.

Advanced Exercise

In this exercise, we are going to use the same network depicted in Figure 2, but the router will also need to translate private IP addresses and perform NAT services. First of all, we can start the lab, which also sets up the web servers. In this case, connectivity is limited, as the LANs are on different address spaces and they cannot reach each other. We now set up R1 to perform NAT on outgoing packets:

```
iptables --table nat --append POSTROUTING --source 192.168.10.0/24 --out-interface eth1 --jump
SNAT --to 161.175.30.253
```

Packets from inside the LAN can now reach the outside. Replies will be sent to R1's public address, but the rule will detect "conversations" and forward the appropriate packets.

Exercise 4: *Test connectivity by connecting to WS from LANPC1.*

Now we need to expose LANWS to the Internet, as it needs to provide a public service:

```
iptables --table nat --append PREROUTING --protocol tcp --destination 161.175.30.253
--destination-port 80 --in-interface eth1 --jump DNAT --to 192.168.10.240:80
```

This command forwards any web request sent to R1 to LANWS, on the same port: in this way, WS can now reach LANWS's web service, while it still cannot ping any machine inside the LAN.

Exercise 5: *Set up SSH access to R1 from inside the LAN as in the second example. Write down the* `iptables` *commands you used*