Information and Organisational Security

Guides for Practical Classes

João Paulo Barraca and Vitor Cunha

Department of Electronics, Telecommunications and Informatics
University of Aveiro

2019–2020

9

Firewalls with `iptables` in the Linux OS

9.1 Introduction

A firewall is a service allowing to filter the traffic flowing through a host. When using the Linux Operating System, the firewall can be implemented using the `iptables` mechanism. It interacts with the `netfilter` kernel framework, allowing detailed control over the packets exchanged.

In this guide, the objective is to explore the creation and exploitation of a Linux based firewall. For this purpose it is required to create two virtual machines.

- The **FW VM** will have two network interfaces, one bridged with the Host wireless interface with a dynamic configuration (DHCP), and another connected to an internal network with a static configuration (e.g., 192.168.1.1)
- The **Server VM** will have a single network interface, connected to the same internal network (e.g., 192.168.1.2), and using the FW VM as the default route.

After all machines are running, connect the Wireless card to the Access Point provided for this class. The FW VM should be able to `ping` the address `192.168.0.1`, and the Server VM should be able to `ping` the internal interface of the FW VM.

It is not expected for the Server VM to reach the outside network.

9.2 Default policies

For each `iptables` chain we should define a default policy, which represents the rule that is applied when no other rule is present. Like in any other security aspect we should choose a defensive approach, denying traffic from potentially malicious sources. In our case we will allow outgoing traffic, generated from the local machine.

For this purpose execute the following commands in the FW machine:

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

You can check the correct application of the rules using the following command:

```
iptables -L -n
```

You can also check that everything is working if you try to `ping` the FW from the Host.

9.3 Establish basic connectivity

With the policies that are applied, no host can communicate with the external network. Not even the Firewall.

If we wish to allow the `ping` command to be used, we can add a rule for this purpose. If a packet matches this rule, the default policy will not be applied to it and the packet will be handled according to the rule. The following commands insert rules that allow for `ICMP` requests to be accepted:

```
iptables -A INPUT -p icmp --icmp echo-request -j ACCEPT
```

Check if the `ping` command operates as expected.

What happens if you issue a `ping` from the Host to the FW or to your colleagues FW VM? Check with Wireshark which packets are exchanged and describe what happens.

Can you create a rule that allows for the correct operation of the `ping` command. Take in consideration that while the availability of this protocol is not mandatory, blocking all `ICMP` packets can have negative consequences.

9.4 Filter internal traffic

Up to this moment the Server VM has no connectivity to the outside world, because all packets will be blocked by the rules active in the FW VM.

In order to allow this connectivity, the FW VM must enable the following functionality: (i) forward IP traffic, (ii) do not block the forwarded traffic, (iii) apply a Network (Port) Address Translation (NAT) mechanism to the traffic coming from the internal network.

In order to enable routing, one of the following commands can be executed:

```
echo 1 > /proc/sys/net/ipv4/conf/all/forwarding
```

or

```
sysctl -w net.ipv4.ip_forward=1
```

This change can be made permanent by editing the file `/etc/sysctl.conf`, and activating the line `net.ipv4.ip_forward=1`.

Allowing the traffic to be forwarded requires the creation of a rule matching the traffic from the Server VM. Considering that the internal interface of the FW VM is named `ethint0`, configured with the address `192.168.1.1`, and the external interface is named `ethext0`, the following rule can be used:

```
iptables -A FORWARD -s 192.168.1.0/24 -i ethint0 -j ACCEPT
iptables -A FORWARD -o ethint0 -d 192.168.1.0/24 -j ACCEPT
```

Check that the Server VM can access a service in the external network. Use Wireshark in the FW VM in order to analyze what is happening.

Finally, enabling NAT mechanism will allow the FW VM to hide the Server VM from the outside world, while providing connectivity. Without these mechanisms, packets will reach the destination but the destination will be unable to send replies due to the lack of routes.

The Linux OS can do NAT by issuing a rule stating that all packets from the Server VM network should be masqueraded using the FW VM external IP address.

```
iptables -t nat -A POSTROUTING -s 192.168.1.0/24 -o ethext0 -j MASQUERADE
```

Check that the Server VM can access a service in the external network.

9.5 Filter traffic for specific hosts

Frequently, it is required to block the access to a specific host in order to comply with the security policy of the domain. As an example, consider that users should be be allowed to access social networks, such as Facebook. A rule blocking access to a host has the following shape:

```
iptables -I FORWARD -d ip_address -j DROP
```

Using the `host` command, find all address of `www.facebook.com` and insert the appropriate rules. Then, validate the effectiveness of the setup.

Note: The command uses `-I` (insert) and not `-A` (append). Rules are processed in a sequential manner, and the rules already present allow all traffic from the Server VM. Therefore, new rules must be inserted before the ones already existing. If you did some mistake, you can use `-D` to delete a rule.

9.6 Using DROP vs. REJECT

In the previous situation the decision was to silently drop all packets. However, other decisions can be used, such as REJECT. The different decisions will influence how the packet is handled.

To check this, insert rules to block `www.google.pt` using the REJECT decision.

Access both services and compare the results. Wireshark can also help diagnosing the behavior of the firewall.

9.7 Traffic logs

In this section we shall test a different decision target which creates records of the network activity. They may be useful to identify the communication endpoints, detecting unusual patterns or communication activity. It should be noticed that this will only allow detection after the communication actually took place, and will not allow blocking the offending exchange.

One possible approach is to configure `iptables` so that some packets, both from successful connections and rejected connections are logged to a registry.

Considering the next command, if the destination address (`ip_address`) is the same as the one used in the previews rules, this will log all dropped

packets. Using `-A` would not work in this case, as the rule would never be reached.

```
iptables -I FORWARD -d ip_address -j LOG --log-prefix "DROP "
```

In order to log all successful TCP connections, we will need to add rules that apply to every new packets from TCP connections that were not dropped. In this case, the rules must be added to the middle of the table, after the rules that drop packets, and before the rules that accept traffic. The following command achieves this when using the appropriate value of `N`.

```
iptables -I FORWARD N -d ip_address -p tcp -m state
    --state NEW -j LOG --log-prefix "TCP NEW "
```

To list all rules and find the appropriate place to inject the previous rule, you can use the following command:

```
iptables -L
```

As a curiosity, if we wanted to apply a rule to the remaining TCP packets we could use the following match:

```
iptables -I FORWARD -m state -p tcp --state RELATED,ESTABLISHED
    -j LOG --log-prefix "TCP CONTENT "
```

9.8 Filter traffic from specific services

Frequently it is required to allow only a set of services over well known ports, blocking all remaining traffic. One practical example would be to allow only DNS and HTTPS traffic, which can be implemented using the following rules. As in the previous case, the value of `N` should be used so that the rules are injected after the rules that block specific hosts.

```
iptables -I FORWARD N -s 192.168.1.0/24 -p dns ! --dport 53 -j DROP
iptables -I FORWARD N -s 192.168.1.0/24 -p tcp ! --dport 443 -j DROP
```

Check the command is applied correctly by accessing a page that uses HTTP (port 80), and a page that uses HTTPS (port 443).

9.9 Forward traffic from the outside to internal services

Because we are using NAT, it is not possible for external hosts to access services provided by internal servers (in the Server VM). Additional rules can be added in order to implement the adequate *port forwarding* mechanisms.

Assuming that the internal server is running a SSH server [1] we can create a rule that forwards connections to this server. Specifically we will forward TCP packets, to port 22 of the external interface of the FW VM, to port 22 of the Server VM.

This can be implemented by the following command:

```
iptables -t nat -A PREROUTING -p tcp --dport 22 -d X.X.X.X
  -j DNAT --to 192.168.1.2
```

where `X.X.X.X` represents the external IP address of the FW VM.

You can use another laptop connected to the same external network to check if the service is available.

9.10 Save and restore `iptables` rules

The rules added to `iptables` are always temporary, and will be cleared if the host reboots, or the `-F` argument is specified to the command.

Therefore, it is important to save rules to a file, restore these rules at a later time.

The following commands will save the rules to a file named `/etc/iptables.save`, clear the tables, list the content (they should be empty), and then restore the rules:

```
iptables-save > /etc/iptables.save
iptables -F
iptables -t nat -F
iptables -L
iptables-restore < /etc/iptables.save
```

You can edit the file `/etc/iptables.save` and see how rules are saved. If required, you can also edit the rules.

[1] In Server VM you can install `openssh`

9.11 Dynamic Host Firewall

While the rules in the FW VM are able to apply several restrictions to traffic, they are unable to react to all attacks targeting the services exposed. One important situation that must be addressed is the exposition of the SSH service. Once a server exposes this port to the Internet, it will receive tens or hundreds of login attempts per hour. The attempts will come from human attackers, but also from programs doing large scale service discovery, login attempts with dictionary attacks. It is very important to block offending addresses, after they reach a determined number of failed connection attempts. Because this type of behavior depends on the software running on the Server VM, it is more practical to deploy such filtering at the Server VM, in the form of a dynamic set of rules.

For this purpose, install the `fail2ban` package. Then, edit `/etc/fail2ban/jails.conf` and enable several jails there related to `pam` and `ssh`.

Restart the `fail2ban` service, list the rules that are automatically created, and the jails:

```
service fail2ban restart
iptables -L
fail2ban-client status
fail2ban-client status ssh
```

Now try to access the SSH service multiple times, failing the username or password. After some tries you should be blocked. List the existing rules and the status of the jails and you should see your IP address listed.

9.12 Bibliography

- *Network Address Translation* (NAT), `http://en.wikipedia.org/wiki/Network_address_translation`
- `iptables`, `http://en.wikipedia.org/wiki/Iptables`
- Fail2ban, `https://www.fail2ban.org/wiki/index.php/Main_Page`