# Laboratory Assignment 2:
# The Linux Firewall

## 1 Usage of the Virtual Machines

Students are recommended to use their own equipment to do the lab assignments as that will make it easier for them to prepare for the lab. It is also possible to run the virtual machines in the computers at the lab. A more detailed description about the necessary tools, instructions and links to download the virtual machines is available on Canvas.

Follow the steps below before starting with the lab assignment:

1. Download the lab machines and extract them using 7zip
2. Start *Oracle VM VirtualBox Manager*
3. Click on the menu Machine → Add
4. Go to the location where you have downloaded and unzipped the virtual machines, and select the *.vbox* file.
5. Repeat for all VMs needed in the lab.
6. If a snapshot called "lab-start" or similar does not exist yet, create a Snapshot by following the instructions below:
   (a) Select the VM in *Oracle VM VirtualBox Manager*
   (b) Click on *Machine Tools* → Snapshots
   (c) Right-click on *Current State* → Take
   (d) Name the Snapshot *START_EDA491* and click OK.
7. Follow the lab instructions.

If using the lab machines, remember to delete, when you are done, your local copies of the virtual machines (from virtual box first and then from the extracted folder) to ensure the system has free space for the next student.

## 2 How to ask questions and demonstrate the lab

Labs will be performed on room ED4225. Attending the room physically to do the demonstration is required to pass the labs. All students in the group need to: be physically present; and

have understood, and be able to answer all the questions. Demos can take from 15 minutes to half an hour. Since there may be other groups before you, you should never consider coming to do your demonstration less than one hour before the lab pass ends.

You are also welcome to come to the labs to do the assignment, ask us questions, and even perform your demo once you are done. That said, other than for the demo, it is up to you to decide where and how to perform the assignment. During the lab, TAs will prioritize attention to students doing their demonstration during the time they booked.

If you need to do demos or plan on joining the lab session after 18:00 or before 10:00, make sure to perform your booking at least 24 hours in advance to ensure that a TA will wait for you. Please remove your bookings (or ask us to do it for you if it is already locked) if you cannot attend your booking. We have to plan our workload based on the number of students signing up and we would rather leave early than wait an hour for an student which will not come.

# 3 Purpose and scope

In this assignment you will learn about the concepts used in network firewalls by configuring a firewall for Linux. We only cover IPv4 in this lab, but do not forget to set the same rules for IPv6 when securing your own machines.

This assignment consists of the following exercises:

1. Get to know the firewall admin toolkit – `iptables`
2. Write scripts to configure the firewall
3. Test your firewall
4. Discuss your firewall configuration with the TAs

# 4 Preparation prior the lab

It is necessary to study these documents **before** doing the lab. Make sure you are well prepared, it will save you much time!

You should read:

- On-line doc, Linux 2.4 Packet Filtering HOWTO, (`https://www.netfilter.org/documentation/HOWTO/packet-filtering-HOWTO.html`)
- William Stallings: Cryptography and Network Security (6th edition), Chapter 23 (online chapter). (*Also available in* William Stallings: Computer Security (2nd edition), Chapter 9.)

# 5 Reporting

There is one requirement to pass this lab:

1. You need to **demonstrate your firewall configuration** to a teaching assistant (TA) in the lab, explain your solution, and discuss your answers to the questions during a lab session. During the discussion, it is important that you are able to explain and reflect on your work, simply answering the questions is not enough. **No hand-in is required**.

When you argue the correctness of your firewall configuration, consider the following questions:

- What are the requirements, and have they all been covered and tested?
- Are your firewall rules in the correct order?
- Is your configuration stateful?

# 6 Lab setup

This assignment will be performed using two VirtualBox virtual machines (VMs), you can download the images from the link on canvas. It is important that you create a snapshot

of the VMs called *lab-start* if it doesn't exist yet. If it already exists, restore the snapshot *lab-start*. A short step-by-step guide about importing the VMs is available on canvas (see *Virtual Machines*).

The netfilter framework (known as the Linux firewall) is a kernel module which is loaded on the VMs (and in fact, on almost all Linux installations). The program to configure the firewall in Linux is called `iptables`, which requires privileged access. In the VMs you have granted access to iptables via `sudo`. This means that all commands to `iptables` have to be run with `sudo`:

```
$ sudo /sbin/iptables <iptables arguments>.
```

> *The IPv6 version works the same way, only the command to use is `ip6tables`.*

The names, ip addresses, username and password are shown in Table 1 together with a short description. Keep in mind that the VMs are not connected to the internet and make sure you are only scanning one of the two VMs. Figure 1 provides an overview of the two VMs used in this lab.

| VM name | IP-Address | Description | username/password |
|---|---|---|---|
| firewall | 10.0.0.3/24 | Configure the firewall on this host. | eda491/EDA491! |
| kali-linux | 10.0.0.1/24 | Test the firewall with this host. | eda491/EDA491! |

Table 1: Name of the VMs, including their ip addresses and username/password combination.

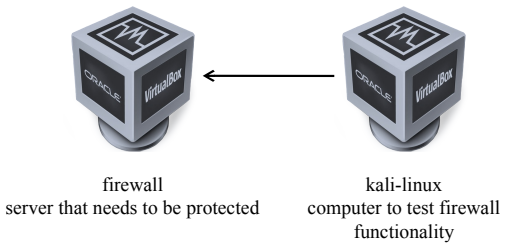


Figure 1: Overview of the VMs used throughout this lab assignment.

# 7 Lab assignments

The following exercises have to be done with the above mentioned VMs. We recommend that you write down the answers to the questions because you need them for the final discussion and demonstration with one of the TAs.

## 7.1 Preparing host firewall

Before you begin, you need to prepare *firewall* for this lab:

*Run the script* `/home/eda491/netsec-lab2/prepare.sh`.

Another script with a basic firewall configuration has been prepared for you. You need to update this script with your own configuration. The script is located at `/home/eda491/netsec-lab2/firewall.sh`. Try to understand what it is doing.

**Services running on your host.** The services running on your host are listed in Table 2. Later, you will need this information to configure your firewall in order to permit access to these services.

| Name | Service | Init-script | Registered port name (port number) |
|---|---|---|---|
| OpenSSH | SSH Server | sshd | SSH (22) |
| Apache | Web Server | apache | http (8080) |
| Rpcbind | portmapper | rpcbind | sunrpc (111) |

Table 2: Services running on your firewall VM.

## 7.2 Introduction to `iptables`

If you have not done so, have a look at the manual page for `iptables` (`man iptables`). Then list the current configuration with the command

```
$ sudo /sbin/iptables -nvL
```

(`-nvL` is used to avoid name resolution.)

There are three predefined chains of rules (in the filter table), `INPUT`, `FORWARD`, and `OUTPUT`.

**Q1: How and when is each chain used?**

### 7.2.1 First steps: blocking ping

To showcase how `iptables` can be used, you will block ping echo replies. First try to ping your own host (`localhost`) and confirm it is alive using `ping localhost` (use `Control-C` to quit). Then execute

```
$ sudo /sbin/iptables -I OUTPUT 1 -p icmp --icmp-type echo-reply -j DROP
```

Try to ping localhost again.

**Q2: What is the effect of dropping echo-reply packets in the OUTPUT chain? Use Figure 2a to illustrate the path of the packets. Mark the path with arrows and use an X to mark the point where the packets are dropped.**

Remove the rule from the `OUTPUT` chain and try icmp-type echo-request in the `INPUT` chain instead. Use `"sudo /sbin/iptables --line-numbers -L OUTPUT"` to find the line that contains the rule that you want to remove.

```
$ sudo /sbin/iptables -D OUTPUT 1
$ sudo /sbin/iptables -I INPUT 1 -p icmp --icmp-type echo-request -j DROP
```

**Q3: What is the effect of dropping echo-request packets in the INPUT chain? Use Figure 2b to illustrate the path of the packets. Mark the path with arrows and use an X to mark the point where the packets are dropped. From a security point of view, is this better or worse than dropping echo-replies in the OUTPUT chain? What about performance?**

Remove the rule from the `INPUT` chain:

```
$ sudo /sbin/iptables -D INPUT 1
```

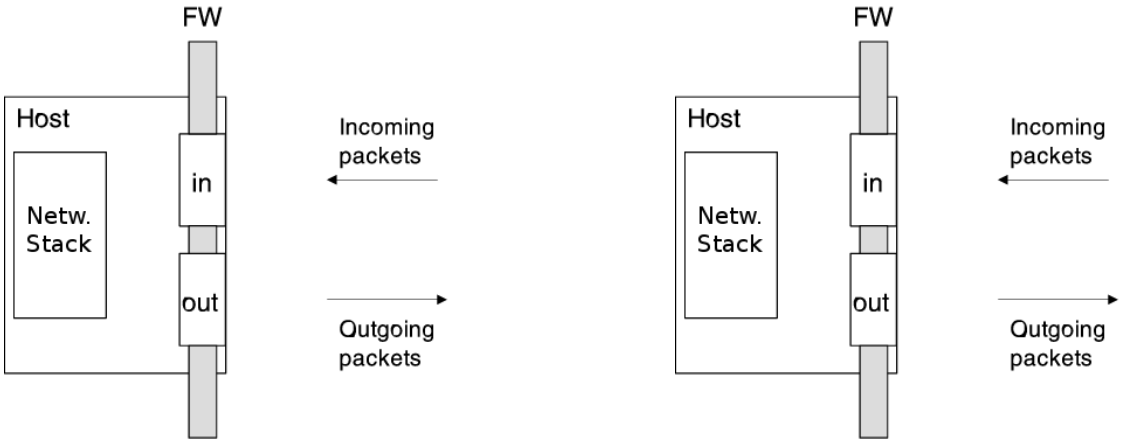


Figure 2: Mark where pings are dropped in the INPUT and OUTPUT chain.

### 7.2.2 Logging

There are several different types of action that can be taken when a packet matches a rule, for example to log the packet in the system log.

Try to log the ping packet in the same way as you dropped it earlier, by using `LOG` instead of `DROP`. Then open a new terminal window and execute `watch -n 0,2 'dmesg | tail'` The command `dmesg | tail` shows the last entries of the kernel's log file and `watch -n 0,2` reruns the command every 0.2 seconds. It keeps running until you interrupt it by pressing `Control-C`. Now try `ping localhost` again. As new messages are added to the log, they will also pop up in the terminal you are running `dmesg` in.

Every log entry contains several items, which can be useful for creating new rules.

**Q4: What do the log items `IN`, `OUT`, `SRC`, `DST` and `PROTO` mean and why might they be useful?**

## 7.3 Writing scripts to configure the firewall

The firewall configuration is usually kept in a script, so that it can be reset quickly and edited easily. Some necessary parts for setting up the firewall have already been configured in the script `firewall.sh`.

**Q5: When configuring a firewall, the order of the rules is of greatest importance! Why?**

### 7.3.1 Common firewall rules

There are some rules that almost every firewall uses to block unauthorized traffic. The following steps will guide you through some of them.

To reset your firewall to a clean state without predefined rules, you will have to flush the individual chains using `iptables -F [chain]`. This has already been setup in the initial script you received (`firewall.sh`). Look at the script and make sure you understand how it works.

**Set the default policies to default deny (Requirement #1).** Currently the default policy of all chains is set to `ACCEPT`. This means that all traffic to your host is permitted. Add additional rules to set the *default policy* to `DROP`, so that all traffic not matching any rules will be discarded.

**NOTE:** When flushing a chain, its default policy stays the same. This is important to keep in mind when you are configuring a firewall remotely, and the policy for a chain has been set to `DROP`. If you have no easy way to physically access the machine, it may be better to keep the default policy on `ACCEPT`, and to add an explicit `DROP` rule at the end of each chain. It is easy to accidentally lock yourself out by flushing the chains otherwise.

By setting the default policy to `DROP` we are following one of the two roads of the *fundamental security philosophy* in firewalling, "everything not specifically permitted is denied" (`default deny`). The other one is to set the default policy to `ACCEPT`, "everything not specifically denied is permitted" (`default permit`).

Save your script and run it. Use `sudo /sbin/iptables -nvL` to list the rules. (If you can not run the script, check the file permission. You might need to set the executable flag on the script, `chmod +x firewall.sh`.)

**Q6: Consider a firewall which uses a default permit policy, and has no other rules set. Is the system secure? Is it useful?**

**Q7: Consider the reverse: a firewall which uses a default deny policy, and has no other rules set. Is the system secure? Is it useful?**

**Allow all traffic from the loopback device (Requirement #2).** The loopback device is often used by programmers developing networked applications. By using

```
iptables -A <chain> -i <interface> -o <interface> -j <target rule>
```

you can add rules to specific interfaces. The `-i` flag specifies the interface the packet arrived at and the `-o` flag specifies the interface the packet will be leaving from. (The `-i` and `-o` options can be used separately.)

Add the rules to allow all traffic to and from your loopback device. The loopback device (interface) is named `lo` in Linux. Save your script and run it to update the firewall. Now you can test your configuration for the loopback device, e.g., by running `ping localhost`.

**Allow traffic from your host (Requirement #3).** At the moment no traffic is allowed to leave your computer (since the default policy for the `OUTPUT`-chain is `DROP`).

Add the rule to allow all traffic leaving your host on your network interface. Once you have updated your script, save and run it.

**Drop spoofed packets (Requirement #4).** There are four networks in the Internet Protocol that are reserved for private networks [RFC1918, RFC3927], 10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16, and 169.254.0.0/16. Traffic to/from these networks should never occur on your network (except when you are connected to one of them[1]) and should therefore be dropped immediately! This protects you from spoofed packets (specially crafted IP packets with false address information).

Add rules to drop all packets you do not expect to see, i.e., packets from the private networks. Use the `-i` and `-o` options to specify the incoming and outgoing interfaces and use the `-s` and `-d` flags to specify the source and destination networks. If you are interested in getting information about potentially malicious spoofed packets, you can first log the packet and then drop it. Save your changes and rerun your script.

The Linux kernel feature *Source Address Verification* automatically drops spoofed packets. In order to sufficiently verify the firewall rules you have to comment the following lines in `/etc/sysctl.d/10-network-security.conf` and restart the VM:

```
net.ipv4.conf.default.rp_filter=2
net.ipv4.conf.all.rp_filter=2
```

**Allow established connections (stateful inspection) (Requirement #5).** One of the features in Linux Kernel 2.4+ is the ability to track connection status, which allows a more fine-grained control of your network traffic. This feature is named *State Match* and can be used to match a connection to one of four states, `NEW`, `ESTABLISHED`, `RELATED` and `INVALID`. Such rules are added using the `"-m state --state <states>"` option.

The two states `ESTABLISHED` and `RELATED` are very useful for programs which need to open new related connections (like the `ftp-data` connection) or send related information about a connection (like error messages).

Add rules to let all packets belonging to an established connection, or in some way related to one, be accepted. Save your changes and rerun your script.

**Q8: How can you check if your rule(s) for established connections is (are) working?**

**Allow ping and add protection from ping-flooding (Requirement #6).** Another feature in Linux Kernel 2.4+ is the ability to set limits on the number of packets received by a specific protocol and port during a given time period. This can be used to alleviate the problems caused by ping-flooding.

[1] Notice that 10.0.0.0/24 is a subnet of 10.0.0.0/8, there are three ways to solve this: creating a second chain and using a RETURN statement (recommended), using an ipset (most performant), or using the iprange module.

Add rules to limit the received ping packets (`echo-request`) to a maximum of one per second[2]. Save your changes and rerun your script. Log into *kali-linux* and try to ping-flood your host *firewall*. Flooding is done with the command "`ping -f <hostname>`", but this requires root privileges. Instead you can use "`ping -i 0.2 <hostname>`", which generates 5 packets per second, the maximum number of packets a regular user can generate.

**NOTE:** Once an ICMP-message has been accepted by a rule, a state is created for that sequence of messages. This means that as soon as the echo-request has matched this rule, it will subsequently also be matchable by the rule for an established connection.

**Q9: How can you check if your rule(s) for limiting echo-request packets to 1 per second is (are) working? Give two examples!**

**Drop Xmas and Null packets (Requirement #7).** Add rules to drop Xmas and Null packets. Reflect on the importance of why Null and Xmas packets should be dropped immediately by the firewall.

### 7.3.2 Application specific rules

After specifying all the general rules, it is time to add the rules needed by specific programs and services. Services running on your host are specified in Table 2.

**Allow specific applications (Requirement #8).** Add all necessary rules to let other hosts connect to the services in Table 2. (The protocol types are specified in `/etc/services`.) Also, make sure that your firewall is still stateful.

### 7.3.3 Logging all other packets

Logging is *very* important in firewalls. Sometimes it is the only way to see what the firewall has been exposed to.

**Log all other packets (Requirement #9).** Add rules to log all unmatched packets (i.e. packets that are not matched by any other rule). Save and rerun your script.

## 7.4 Testing your firewall

To verify the correctness of your firewall configuration, you should use `nmap` from the VM *kali-linux* to scan your own host. **You are not allowed to scan any other hosts!**

With `nmap` you only check the states of the ports: **Convince yourself that the firewall works correctly by trying out other appropriate commands!**

## 7.5 Reflection

**Q10: If you remember only one thing from this lab, what should it be?** (There is no "right" answer here.)

[2] Look for ways to set a limit, you might need two statements to achieve this requirement. `https://linux.die.net/man/8/iptables`

### 7.6 Demonstrate the finished script

1. Make sure your firewall rules are clearly written.
2. Make sure all students in your group have understood all answers to the questions.
3. Show your firewall script to a teaching assistant and discuss your answers via zoom.
4. Export your firewall rules .
5. Record the output of "iptables -nvL --line-numbers".

# 8 Optional task(s)

The following task(s) are completely and absolutely optional and are only provided as a reference to students who want to increase their knowledge further than the contents of the course. The TAs will never ask you any questions regarding these tasks. Similarly, the exam will be created assuming these tasks never existed. Consequently, most TAs will not be able to help you with the optional tasks in which case you should ask Francisco about them. Francisco will also be very thankful if you provide him feedback about the tasks and your experience performing them.

## 8.1 Rationale

Firewalls like iptables serve a critical defensive purpose. A successful attacker will be unable to affect any services it cannot reach from the systems it gains control of. Therefore, router level firewalls can be used to restrict communications between networks and endpoint level firewalls can be used to restrict access on a fine grained level and reduce the exposed attack surface.

IPtables has been around for more than twenty years and, although it is widely available, it also starts showing signs of its age through its flexibility and performance. Despite that it is a well known, widely available tool on modern Linux systems for which experts are also widely available.

Modern commercial firewalls still use rule based approaches like iptables but introduce further abstraction (like the concept of zones covering different computers), better grouping of services (like the possibility of creating dynamic groups of ports), better detection of services by analyzing the data being exchanged and better detection of threats using intelligence provided by their vendor. *Nftables*[3] was born with the idea of significantly increasing the flexibility of the rules to allow better competition against commercial solutions by increasing the system performance. Nftables provides better expressivity than iptables which also allows administrators to obtain better performance by reducing the number of matches the firewall has to do.

A different solution not covered in this task is BPFilter, a firewall where rules are applied using BPF. BPF is a programming language used for handling network packets which is converted to machine code by the kernel. This solution is still a bit inmature and needs better documentation which is why it is not considered.

## 8.2 Task details

The optional task for this lab involves using nftables instead of iptables to build your firewall. You should use optimizations like maps and sets (useful on requirements 4, 8, and maybe 7), meters (useful for requirement 6) and maybe even chains hooked on specific network interfaces to speed up processing.

The objective of this task is that you notice how, despite increasing complexity, the greater expressivity of nftables can result in simpler, faster and less error prone rules.

[3] `https://netfilter.org/projects/nftables/`