ROYAL INSTITUTE
OF TECHNOLOGY

School of Computer Science and Communication
Department of Theoretical Computer Science

LAB F

Firewall: Packet filtering with `iptables`

NAME	KTH USERNAME

DATE :: _____-_____-_____

TEACHING ASSISTANT'S NAME :: _____

LAB F PASSED (TA'S SIGNATURE) :: _____

Computer Security

DD2395 / HT2017

r(None)

(None)

Contents

1 Introduction

During this lab you will be introduced to the concepts of firewalls, i. e., packet filtering in GNU/Linux using iptables to set up and configure filtering rules. Moreover, you will learn how to use the network mapper tool nmap to explore systems on the Internet. The lab is also a refresher of your basic GNU/Linux and computer networking skills.
iptables (also known as netfilter) is the user space command line program used to configure the Linux 2.4.x and later packet filtering ruleset, a successor of ipchains. Both netfilter and ipchains were started in 1999 by Paul Russell, a free software developer.

Deadlines: Preparation questions before coming to the lab, tasks by the end of the session

The questions in Section 1.3 are to be answered **before** coming to the lab session you signed up for. The lab is to be finished by the end of the lab session you signed up for. Therefore, you are encouraged to come prepared and use the time wisely.

1.1 References

Before and during the lab, you might want to consult some of the references that you will find below.

For iptables, a concise overview of the most important commands can be found at:
`https://www.centos.org/docs/4/html/rhel-rg-en-4/s1-iptables-options.html`
Two more extensive documentations, including a refresher of general networking knowledge, can be found at:
`https://www.frozentux.net/documents/iptables-tutorial/`
`https://www.netfilter.org/documentation/HOWTO//packet-filtering-HOWTO.html`

For nmap there is a reference at: `https://nmap.org/book/man.html`

Especially when looking for the syntax and the meaning of arguments, the manual pages are a good reference (on the command line type `man nmap` or `man iptables`, to search type `/searchterm` and press enter, to repeat the search type `n`, to quit press `q`).

If you are not used to working on the command line, have a look at Appendix D.

1.2 Downloading and running the Oracle VirtualBox Virtual Machine

The environment to carry out this lab is contained in a virtual machine, which you can run either in the lab rooms or on your own computer. However, running this software requires at least 1 GB of RAM and 10 GB of disk space. On the lab computers, the required software Oracle VirtualBox is already installed. If you want to work on your own computer, you can simply download and install it from `https://www.virtualbox.org`. The prepared virtual machine can be downloaded from `http://www.csc.kth.se/utbildning/kth/kurser/DD2395/dasak17/DASAK_F_VM.ova` or a mirror[1] and we call it simply VM throughout this lab. Since the main focus is on the lab computers, we will only go into detail on how to set up the VM on the these. The steps are very similar for importing the virtual machine on your own computers using the VirtualBox graphical user interface (GUI).

[1] KTH Box mirror - `https://kth.box.com/v/DASAKFVMv5`

Downloading the VM takes time

Downloading may take some time, so you may want to start the download first and then proceed with the following reading and tasks.

On the lab computer, we will work in the temporary directory available for and accessible to all users. Once you are logged in using your KTH credentials, do the following to download and import the VM while using the temporary file system location instead of the home directory. The same steps can be done using the GUI.

1. Open a terminal.
2. Execute `cd /tmp`
3. Execute `mkdir /tmp/vboxvmdasak`
4. On the computers in the lab, try to execute `cp /NOBACKUP/DD2395/DASAK_F_VM.ova /tmp/`. If it succeeds, proceed to the next step. Otherwise,
   - `wget http://www.csc.kth.se/utbildning/kth/kurser/DD2395/dasak17/DASAK_F_VM.ova`, or
   - try to download from a mirror in the footnote, but download directly to `/tmp`.
5. Execute `vboxmanage setproperty machinefolder /tmp/vboxvmdasak`
6. Execute `vboxmanage import DASAK_F_VM.ova`
7. Wait for the import to complete.
8. Execute `vboxmanage setproperty machinefolder default`
9. Execute `vboxmanage modifyvm DASAK_F_VM --usbehci off`
10. Execute `vboxmanage startvm DASAK_F_VM`
11. Don't forget to delete the VM, and all the files from the disk, after the lab to clean up your user profile.

At this point, you should see the login screen of Ubuntu in the VirtualBox window. The login procedure follows in Section 2.

1.3 Preparation Questions

Please answer the following questions before coming to the lab session. All questions can be answered with the references mentioned in Section 1.1.

1.3.1 iptables

Chains The rules in iptables are always parts of a chain. Chains can either be user created or one of the built-in chains. In this lab we will use three standard chains: INPUT, FORWARD and OUTPUT. Explain which packets will pass through each of these chains:

INPUT:

FORWARD:

OUTPUT:

Operators To change a chain, you need to use an operator. Look up the short form command for the following operators and explain briefly what they do:

Append:

Insert:

Delete:

List:

Flush:

Filters Filters are used to choose which packets match a rule. Each filter is used to create matches. Some matches requires modules to be loaded with `-m MODULENAME` to be available. Explain the following matches and how they can be used:

-p:

-s:

-d:

-i:

-o:

--sport:

--dport:

Jump Targets Jump targets are used to decide what to do with a packet once it has matched a rule. Explain the following targets:

ACCEPT:

DROP:

REJECT:

LOG:

1.3.2 nmap

The nmap tool provides different scan methods, to discover and analyse remote hosts. Each of these scan methods uses a certain network protocol. Each network protocol operates on a certain OSI-model layer (e. g., ARP on OSI layer 2, IP on OSI layer 3, etc.). So we can group the nmap scan methods according to OSI layers. Please name some examples for each group mentioned below and describe them (name of the scan method, what command/arguments starts it, which network protocol does the scan method use, what is the purpose of the scan method, what

are the limitations of the scan method).

OSI layer 3:

OSI layer 4:

OSI layer 7:

Milestone

Report your progress to a lab assistant.

2 Getting Started

Saving time: Work in parallel and ask questions (using the queue system "Stay A While")

In order to be able to complete the lab during the allocated time, you can do more than one task at a time as not all of them have to be done sequentially. If you have to wait for a program to finish or a milestone to be signed, consider continuing with another part and/or start reading the details. If you get stuck somewhere, don't hesitate to ask for help (using the queuing system "Stay A While", choosing the `Dasak` queue: `http://queue.csc.kth.se/#/queue/Dasak`).

The laboratory system consists of a virtualized system with three virtual hosts running in a VirtualBox VM and simulating two interconnected networks (see Appendix A). These hosts are running GNU/Linux (Ubuntu Server 14.04.4 LTS). On the course web page, your groups are assigned with numbers. These group numbers determine the IP addresses for your networks according to the network topology map in Appendix A. Please, note that the virtual hosts will not have access to the general Internet to contain disturbances due to our lab experiments.

Group Number, Letter and Login Credentials

Please determine and fill in the following.

group number #: _____ (used in the IP addresses)
inside subnet: _____ (simulated private network)
outside subnet: _____ (simulated public network)

login (VirtualBox VM, `inside-host`, `firewall` and `outside-host`)
username: `student`
passphrase: `time2work`

2.1 Connecting to the Virtual Hosts

After starting the VM as described in Section 1.2, login to the VM using the username and passphrase provided in the box above. You will see a Ubuntu desktop, where you will only need three terminal windows - one for each virtual host. Open them and type each of the following commands in one of the windows.

```
sudo lxc-start -F -n inside-host
sudo lxc-start -F -n firewall
sudo lxc-start -F -n outside-host
```

This boots the hosts and gives you a terminal for them. You can now log in to them and the shell prompts show the hostnames to help you to distinguish them. Some boot messages might be interleaved with the login prompt on the virtual hosts after starting them. In this case you can either just type the username or just press return to see the prompt properly.

Tips for working with the virtual host terminals

- Use 3 terminals for the 3 different virtual hosts.
- After starting a host using the commands above, it is possible to open additional shells using `sudo lxc-attach -n {hostname}`. This can be useful for just displaying the log for example. But better don't open too many because this can easily confuse more than it helps.
- You may maximize the VM window and resize the terminal windows in the VM in order to get a good overview.
- For shutting down a virtual host at the end of the lab, you can issue `shutdown -h now` in one of the consoles. But look at the prompt and make sure that you don't shut down the VM and possibly loose progress. Either way, remember that the network interface and iptables configuration we discuss in the following is not persistent and therefore lost after shutting down.

2.2 Getting Root Access

The next step is to permanently get root access on the virtual hosts.

```
student@firewall:~$ sudo bash
root@firewall:~#
```

Here the `#` indicates that you have root access on the host. While it is a bad habit (normally you should use `sudo` instead of a root shell), almost all commands in this lab requires root access, so this will save you from typing `sudo` constantly. Do this on all three virtual hosts.

2.3 Setting up Interfaces

Next, you need to configure your network interfaces. You do this by manually assigning them an IP address and a netmask. The network map in the appendix describes which addresses to use. Remember to replace # with your group number in all IP addresses (for example if your # is 4, then the host `inside-host` has the IP address 192.168.**4**.2). Use the `ifconfig` tool and make sure the configuration is applied on the correct interfaces (`eth0`, `eth1`) on each machine.

For example, to assign the IP address 10.18.0.33 with a 20 bit netmask to the interface `eth0`, issue the following command:

```
ifconfig eth0 10.18.0.33/20
```

Check the configuration by using `ifconfig` without any arguments.

2.4 Setting up Routing

Normal hosts don't have to route packets and therefore traffic forwarding is not enabled by default on a newly installed server. To ensure that data can be routed through the firewall host (`firewall`), one would have to enable it. Otherwise, it would be only possible to communicate or ping between the directly connected hosts (`inside-host` ↔ `firewall` and `firewall` ↔ `outside-host`) but *not* between the two end hosts (`inside-host` ↔ `outside-host`). This could be verified using the `ping` command (but continue reading). Since our lab is intended for routing experiments, which requires traffic forwarding, we enabled this feature already on all the virtual hosts on the VM. In the topology used for this lab however, only the firewall host needs to be able to forward traffic.

First, verify that packet forwarding is enabled on the firewall host.
This is done by checking, that the file `/proc/sys/net/ipv4/ip_forward` contains "1":
`cat /proc/sys/net/ipv4/ip_forward`
If it is not, set it to 1 to enable forwarding (only on the firewall host `firewall`):
`echo 1 > /proc/sys/net/ipv4/ip_forward`
For completeness, you could disable this on the other two hosts:
`echo 0 > /proc/sys/net/ipv4/ip_forward`
Either way, the end hosts `inside-host` (internal) and `outside-host` (external) have to learn how to reach each other. To configure them, use the following command only on the end hosts
`route add default gw ADDRESS`
to set their default rout to the central firewall host `firewall`. ADDRESS has to be the corresponding address of the directly connected interface on the firewall host.

Do not forget to check the routing table with the `route` command without arguments (be patient, the output may be delayed a bit).

Milestone
Verify that you can ping between the two end hosts of your network (`inside-host` ↔ `outside-host`). ◯

3 iptables: Building a Firewall

Note that all the `iptables` commands have to be run **only on the firewall host** (`firewall`)!

To view the iptables help, run the command:
`man iptables`
More references are listed in Section 1.1.
To list the current state (list of rules and policies for all chains), run the command:
`iptables -vL`

Saving your work

In case a virtual host fails, you might loose the rules you have created. Therefore keep a record of what you are doing (on paper or in a file on your local machine). At any time you can use the `iptables-save` command, to print out the rules you created so far. Note that saving them to a file on the virtual host might not be a good idea, as the complete host can fail.

3.1 Ping and the Internet Control Message Protocol (ICMP)

The `ping` tool is usually used to test the reachability of a host that implements the Internet protocol (IP). It operates by sending packets using the Internet Control Message Protocol (ICMP) of the type 'echo request' to the host whose reachability is to be tested, and processing the response from that host (if any).

In Listing 1, you can see a sample output of this utility testing the web server of KTH (`ping www.kth.se`) . The results show three ICMP requests (which were successful as the time implies that there was a response) and a statistical summary of the response packets received.

```
PING www.kth.se (130.237.32.143) 56(84) bytes of data.
64 bytes from 130.237.32.143: icmp_req=1 ttl=254 time=10.4 ms
64 bytes from 130.237.32.143: icmp_req=2 ttl=254 time=10.4 ms
64 bytes from 130.237.32.143: icmp_req=3 ttl=254 time=9.31 ms
--- www.kth.se ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2002ms
rtt min/avg/max/mdev = 9.315/10.064/10.478/0.543 ms
```

Listing 1: Sample output in the command line of the ping utility

3.1.1 Blocking ICMP requests

`ping` the internal host `inside-host` from the external host `outside-host`. What happens? Now execute the following command on the firewall host:
`iptables -A FORWARD -p icmp --icmp-type echo-request -j DROP`

`ping` the host again. What happens and why?

Check how the output from `iptables -vL` changed.

To remove the rule from the chain, first list the rules again:
`iptables --line-numbers -L FORWARD`
There you can find the line number of the rule you just added, and then remove it with:
`iptables -D FORWARD line_number`

3.1.2 Rejecting ICMP Requests

Now, create a new rule which REJECTS all ICMP echo-requests from the external network to your internal network. Note that iptables rules only match if all the conditions in the rule are true.

Specifying Source and Destination

For simplicity, we have only one internal and one external host in the lab network setup. However, when specifying the source and destination for iptables rules, try to be more general than specifying only single IP addresses as source or destination. You can accomplish this either by using an IP address range (e. g., `10.#.0.0/20`) for the `-s` and `-d` option, or by specifying the involved interface with `-i` and `-o`.

Now, verify that

- you can `ping` from the internal host to the external,
- the external host cannot `ping` the internal, and
- the firewall can `ping` both hosts.

Can you `ping` from the external host to the **internal interface** `eth0` on the **firewall host** (note: this question is often misunderstood, please read it carefully)? Why/why not?

Can this have any security implications?

What is the difference between rejecting and dropping traffic?

What are the advantages of rejecting and what are the advantages of dropping?

3.2 Logging and Limits

One important tasks for a firewall is to log rejected or dropped packets, making it easier to trace attacks. A rule can be created with the jump target NFLOG to save information to the system's ulog. We use NFLOG instead of LOG, since our virtualized system has some contraints and therefore iptables cannot log directly to the syslog system but ulog.
Create a rule that logs all rejected ping messages. Make sure the string "`Ping rejected by <your name>:`" is written in the log message. Check the ulog so the traffic is saved. This can easily be done with:
`less +F /var/log/ulog/syslogemu.log`
Notice, that you cannot use `tail -f` for displaying the ulog. Furthermore, `less +F` can be quit by first pressing Ctrl+C to interrupt and then pressing q to quit.

The module `limit` can be used to limit how often a rule can be triggered. Use the `limit` module to make sure no more than 5 pings each minute are saved to the system log.

Milestone
Report your progress to a lab assistant.

3.3 Building a firewall

You will now build a simple firewall for your network. Before you start, make sure that there are no rules left from the previous assignments: flush all rules by running
`iptables -F`
(not specifying a chain removes all rules from all chains).

Order of rules

For each step, think about the order of the rules: Is it enough to append a new rule at the end of a chain, or should it go before a certain other rule, in order to work as intended? If you use the `-I` argument without specifying a number, the rule will be placed at the top of the chain.

3.3.1 Default Policy

Each chain has a default policy target that details what to do with any packet that does not match any rules present in the chain. Set all chains to the policy target DROP.
Verify that you cannot send any data through or to the firewall (e. g., using `ping`).
Default policy targets can only be set to ACCEPT or DROP but not to REJECT as it is an extension target. If you would want to REJECT all packages that do not match a rule in the INPUT chain, how would you do it?

3.3.2 Network Permissions

Now it is time to start allowing some carefully chosen traffic through the firewall.

- Create a rule that allows all traffic originating from the internal network arriving to the internal interface (`eth0`) of the firewall host.
- Create a rule that allows all traffic originating from the firewall to reach the internal network.

Make sure you now can reach the internal network from the firewall, and the firewall from the internal network.

3.3.3 Permitting a Service

SSH (Secure shell) is a common service for remote management of firewalls. Create rules which allow a host from the external network to connect to the firewall with SSH. SSH runs on port 22 and uses only TCP.
Make sure that the ssh daemon *sshd* is running on all hosts by executing `service ssh restart`

Source and destination ports

Note that the well-known port number for a service only applies for the machine where the service is running (server), not for machines that connect to the service (clients). So if a client machine sends a packet to a specific service on a server machine, the destination port will be the well-known port number for that service but the source port can be any number. Accordingly, when the server machine sends an answer packet from the service back to the client machine, this packet will have the well-known port number as source port but the destination port will be the number chosen by the client machine.

Verify that,

- you **can** connect to the firewall with SSH from the other hosts,
- you can**not** connect directly from your external host to the internal host with ssh, and
- you **can** connect from your external host to the internal host if you first connect to the firewall with ssh and then connect from there with ssh to the internal host (note: think about which username you should use when logging in to the hosts with ssh).

What kind of security advantage does a setup with an SSH terminal server offer?

What kind of security disadvantage does a setup with an SSH terminal server introduce?

See: `http://www.fail2ban.org`
How does the framework `fail2ban` mitigate this disadvantage?

3.3.4 Stateful Filtering

In most cases we want to allow hosts on the internal network to connect to the external network, e. g., the Internet. However, we do not want hosts on the Internet to be able to connect to hosts on the internal network. For some protocols, such as TCP this can be done *stateless*, due to the three way handshake needed to create a connection: By blocking the initial SYN packet in one direction we can prevent the establishment of connections in that direction while still allowing connections that were established in the other direction to send packets both ways (as they will never send an initial SYN packet in the direction that we blocked).
However, stateless filtering breaks a large number of protocols, for example UDP based protocols cannot easily be allowed through in only one direction. Therefore, we need a *stateful* firewall to properly handle it. Some protocols, such as FTP, also break if connections from the Internet are completely denied.
Examine the module `state`. Use this module to create a match that allows the hosts on the inside of the firewall to establish connections to the outside. Allow all data that belongs to these connections through the firewall. Keep blocking all other connection attempts from the outside.

3.3.5 Opening Ports

Sometimes, you want computers on the outside of the network to have access to a specific service on the inside. Your task now is therefore to add rules to your firewall, so that external computers on the outside can reach the *echo* service (port 7) on hosts on the inside both on UDP and TCP.

The echo service (provided by the extended internet service deamon `xinetd`) is a legacy testing protocol that simply replies with the same data that is sent to it. Usually it is disabled by default, but on your hosts it is activated (you can check this by running `lsof -i` and look for lines that end in "echo (LISTEN)" or similar). To test your firewall rules, connect to the echo service that is running on your internal host, **from the external host** by running
`telnet IPADDRESS 7`
on the external host and observe that all data you send is echoed back to you. Notice that you can quit `telnet` by pressing Ctrl+] (or Ctrl+5) and then typing `quit` in the telnet prompt.

3.3.6 Blocking Ports

Sometimes you do not want your *internal* users to be able to connect to the Internet on a specific port at all. One commonly blocked port is 135 which is used for Windows file sharing[2]. Make sure your firewall blocks all traffic on port 135 (both TCP and UDP) from the computers on the internal network.

3.3.7 Verifying Your Setup

Verify your setup in a systematic manner, making sure all rules work correctly. To test a certain rule, you can listen to a port with `netcat` and try to connect to it using `telnet`:
On the receiving host run: `nc -l PORT`
On the sending host run: `telnet IPADDRESS PORT`
Furthermore, use the counters in the output of `iptables -vL` to make sure the packets match the rules you expect.
At this point you should have the following rules active, in this order of priority:

- Connections on **port 135** from the inside hosts are blocked.
- Connections on the **SSH port** are allowed to the firewall host from the outside.
- Connections on the **echo port** are allowed to the internal hosts from the outside.
- Connections **directly to and from the firewall** are allowed from the internal networks.
- Connections **from the inside** are allowed out.
- Connections **from the outside** are blocked.

Milestone
Report your progress to a lab assistant.

3.3.8 Defending Against SSH Brute-force Attacks

You have allowed connections to the SSH port (hence, its service) of the firewall from any host on the outside network. There is nothing that would avoid a brute-force attack or a Distributed Denial of Service (DDoS) attack because there are no limitations on the amount of times that one particular host can try a username and a password (besides the ones that the SSH server implementation might have internally). So you are going to put in practice the knowledge you have acquired about stateful filtering and logging to prevent a host from the outside network to execute a brute force attack.

Let's start by creating a new chain called `SSH` that is supposed to handle all SSH traffic. After creating the chain, you need to modify the rule in the INPUT chain that allowed connections to the SSH port on the firewall host from the outside network: send all SSH packages to the 'SSH' chain instead of directly accepting them.

[2] Actually there are more ports involved (135-139 on older Windows machines and 445 on more recent ones) but in this lab it is enough to block 135 as one example.

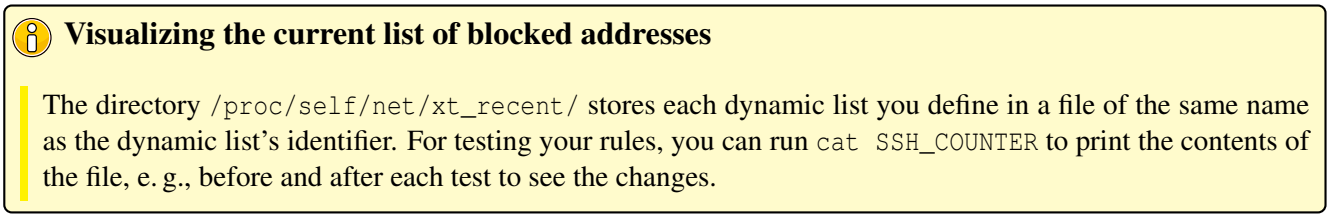
In the 'SSH' chain, first add a rule that accepts all packages from established connections, so that all the following rules only apply for new connection attempts (and don't slow down legitimate, established connections).
Now, have a look at the documentation of the module `recent`. It allows you to create a dynamic list of IP addresses for later use, such as counting the amount of times a particular IP address tries to start a new connection on a port.
Your next step, with the knowledge of the module `recent`, is to create the necessary rules in the 'SSH' chain to catch those hosts that are potentially dangerous. We will assume that anyone trying to log in **more than three times in a row within 30 seconds** from the same host is potentially malicious and should be slowed down.
We want not only to block, but also to log suspicious packages. But in case of a brute-force attack, we don't want the logfile to be flooded. So from each suspicious IP address we will block the fourth, fifth and all further connection attempts (in 30 seconds), but only log the fourth connection attempt.

Create a first rule so that the source address of the packet is added to a custom dynamic list named `SSH_COUNTER`
Then, add a second rule so that the fifth (and any further) packet attempting to establish an SSH connection and coming from the same source in a time period of 30 seconds gets dropped.
In order to log suspicious connection attempts, create two rules so that one of them logs the fourth SSH connection from the same host address (use the prefix "SSH brute force attacker: "), and the other one blocks the package.
Note that in order to avoid getting our log full of entries for every malicious attacker we implemented the previous rule such that the fifth attempt and any other subsequent one within the specified time gets dropped directly without being logged (and only the fourth attempt will be logged and blocked).
Finally, you have to accept all packets that have not been stopped so far. Why do you need to do this?

At this point the newly created 'SSH' chain should contain the following rules, in this order of priority:

- A rule accepting all packets from already established connections.
- A rule adding IP addresses to a dynamic list.
- Every fifth (and higher) packet from the same source host address within 30 seconds is dropped.
- Every fourth packet from the same source host address within 30 seconds is logged.
- Every fourth packet from the same source host address within 30 seconds is dropped.
- Anything that has not been stopped so far is accepted.

Test your rules, paying attention to the counters with `iptables -vL` before and after connecting via ssh. If you implemented the rules correctly, you should notice that you have to wait a few seconds from the moment you start trying to connect for the fourth time in 30 seconds from the external host.



You have now a set of rules that would minimize the impact of an attack on your system but there is still a way for a powerful attacker to carry out a successful attack. Can you speculate how and why?

Milestone
Report your progress to a lab assistant.

3.3.9 Building Your Own Firewall

Consider your home network, the services you offer and the service you require from the Internet. Which protocols do you actually use? Which can you block? When you compare these requirements to the rules you just created in this lab, which ones would you keep, which ones not and which additional rules would you add. You may note them down in abbreviated form (which port/protocol, what to do with it, why).

rules to keep:

rules not to keep:

additional rules:

Milestone
Report your progress to a lab assistant.

4 nmap: Detecting Server Capabilities

nmap is a free open source utility commonly used for network exploration and security auditing. nmap can determine what hosts are available on the network, what services (application name and version) those hosts are offering, what operating systems (and OS versions) they are running, what type of packet filters/firewalls are in use, and dozens of other characteristics.To view the nmap help, run: `man nmap`

Do not use nmap outside the lab

In this second part of the lab, you will switch from the perspective of protecting a network with a firewall to the other side and explore ways to attack a system using the network mapper nmap. You can safely explore this tool inside this lab environment. Outside the lab, however, remember never to use nmap on a system if you do not have the explicit consent of the system's owner. Even basic scans on one IP address might be considered as attack and have potentially legal implications.

4.1 Enumeration

You will now use nmap to enumerate your outside network from the external host (`outside-host`). Try to locate the IP address of the unknown server that is connected to the network (see Appendix A). Note that the scan can take up to 5 minutes, so you might want to test your command on a smaller portion of the network before scanning the whole subnet. Also make sure to use options that speed up the scanning, otherwise the scan will take a very long time (up to one hour with some standard settings). Have a look at the `--min-rate` option for that purpose.

Saving time

Since the scan will take some time, pipe the output from nmap to a file and put the process in the background so you can continue your work. Use the `>` operator to pipe the output to a file and the `&` operator to send the process to background:

```
nmap [arguments for nmap] > [output_file] &
```

In case you have trouble to speed up the search and do not find the server, you may ask an instructor in the lab room for the server IP corresponding to your group number. In this way you can continue with the fingerprinting and service search.

Which arguments did you use to locate the server?

nmap

What is the server's address, how long time did the scan take and how many addresses did you scan?

server address:

time: 49.04 s **number of scanned addresses:**

If you get an error message while scanning: Make sure you are root. If you still can't find the server, verify your routing and netmask settings.

4.2 Service Discovery

Now that you have found the server's IP address, try to gather more information, such as the services running on the server. Scan for both TCP and UDP services. Futhermore, try to find out the version number of the discovered services. For the UDP scan you might want to look into the `-F` option.

What arguments did you use for TCP discovery?

nmap

What arguments did you use for UDP discovery?

nmap

UDP discovery is much slower than TCP discovery. Why?

What is the difference between *open*, *closed*, *filtered* and *unfiltered* ports?

open:

closed:

filtered:

unfiltered:

Which services did you find? (If you found about 7 open TCP ports and about 3 open UDP ports you are fine. You might not be able to find version information for all services.)

TCP/UDP	Port	Service	Version
TCP	7	echo	NA
TCP	22	ssh	Protocol 2.0
TCP	53	domain	dnsmasq 2.68
TCP	80	http	apache httpd 2.4.7
TCP	5222	xmpp-client	ejabberd protocol 1.0
TCP	5269	xmpp-server	ejabberd
TCP	5280	xmpp-bosh?	
UDP	7	echo	
UDP	53	domain	dnsmasq 2.68
UDP	123	ntp	NTP v4 (unsynchronized)

4.3 OS discovery

Finally, try to guess the operation system of the server by using the appropriate nmap operations.

What arguments did you use:

nmap

What operation system did nmap consider to be most likely? (If you do not get any text that makes sense to you, try again using the extra option `--fuzzy`)

What information is used by nmap, to guess the operation system?

Milestone
Report your progress to a lab assistant.

5 Cleanup

In theory it is enough to shutdown the virtual hosts and the VM, or simply stop the VM, and log off the lab computer. But since you imported the VM into a temporary disk location on your lab computer, you should delete the VM from the VirtualBox inventory additionally though. You find the corresponding menu options in the graphical user interface of VirtualBox. This also cleans up your user profile and prevent stale data and issues later.

You are done with the lab, yeah!

6 History

Version	**Contribution**	**Author** (Affiliation)	**Contact**
1.0	First development	Pehr Söderman (ICT/KTH)	pehrs@kth.se
2.0	Adaptation for HT2012	Benjamin Greschbach (CSC/KTH)	bgre@kth.se
3.0	Adaptation for HT2013	Guillermo Rodríguez Cano (CSC/KTH)	gurc@csc.kth.se
3.1	New exercise for HT2013	Guillermo Rodríguez Cano (CSC/KTH)	gurc@csc.kth.se
3.2	Updating for HT2015	Benjamin Greschbach (CSC/KTH)	bgre@csc.kth.se
4.0	VirtualBox for HT2017	Andreas Lindner (CSC/KTH)	andili@kth.se

A Lab Network Map

The network map below shows the connections and information about the addressing of the hosts in the lab network. Remember to replace the placeholder # with your according group value. Furthermore, you have to concatenate the network prefix with the host part to obtain the IP address for a certain interface. The length of the network prefix is given by the network mask[3]. For example, if your group number # is 42, then `eth0` of the external host should get the address 10.42.0.2 (10.42.0.0/20 being the network prefix and .0.2 the host part), `eth0` of the firewall host should get 192.168.42.1, and so on.

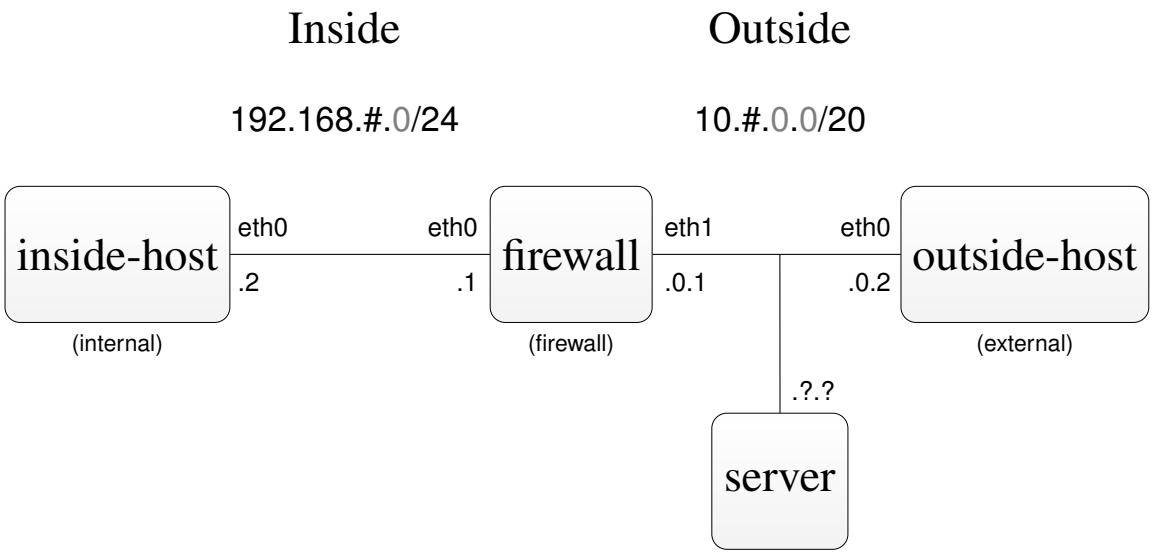


Figure 1: Lab network map.

B FAQ – Common Problems and Fixes

I can't ping between `inside-host` and `outside-host` (but between direct neighbors). Check that `route add default gw ADDRESS` was issued WITHOUT specifying a netmaks after the neighbor's address. You have to take the interface down (`ifconfig eth`$\frac{0}{1}$ `down`) and up again (`ifconfig eth`$\frac{0}{1}$ `up`), after issuing the correct command.

Telnet got stuck, how can I exit from it? Issue Ctrl+C on the listening netcat `nc` instance. Ctrl+] can be used to exit telnet as usual.

I can't login with ssh when connecting from one virtual host to another. Make sure to use the username `student` (the only user account on the virtual hosts) when connecting with ssh: `ssh student@IPADDRESS` Then the passphrase `time2work` will work.

I've set up the correct rules, but ping does not work (no rules match). Note that `ping` uses the ICMP protocol, so it's neither using TCP nor UDP. If your rules match only for TCP and UDP, ping traffic is not affacted by them.

[3]The netmask is written right behind the network address in standard slash-notation: /24 stands for a netmask of 24 bits (which is equal to 255.255.255.0 in legacy notation). Remember that an IPv4 address has 32 bits, so each number in the dot-notation represents 8 bits.

C Flow Diagram of iptables

The following graph shows a simplified and brief scheme of the iptables flow graph. Each chain contains the corresponding tables for the processing of the packets. There are five predefined chains, namely, PREROUTING, FORWARD, INPUT, OUTPUT and POSTROUTING, but not all chains have all the tables (e.g., the INPUT chain only contains the 'mangle' and the 'filter' tables). Note that predefined chains have a default policy (e.g., ACCEPT) which is always applied to the packet traversing the chain when it does not match any of the rules defined for that chain.

Packets start at a given chain, but typically they come either as 'inbound traffic' (that is, from the network card) or from a 'local process', and after traversing the tables of each chain, they will get out of iptables as 'outbound traffic'. Note that depending on the rules the processing of the packets might not be sequential, as the graph shows, and the packets may jump to another chain or even discarded/dropped, but in any case, every packet coming or leaving iptables will traverse one chain at least.

For this lab the only table that you will be using is the one focused on packet filtering, 'filter'.

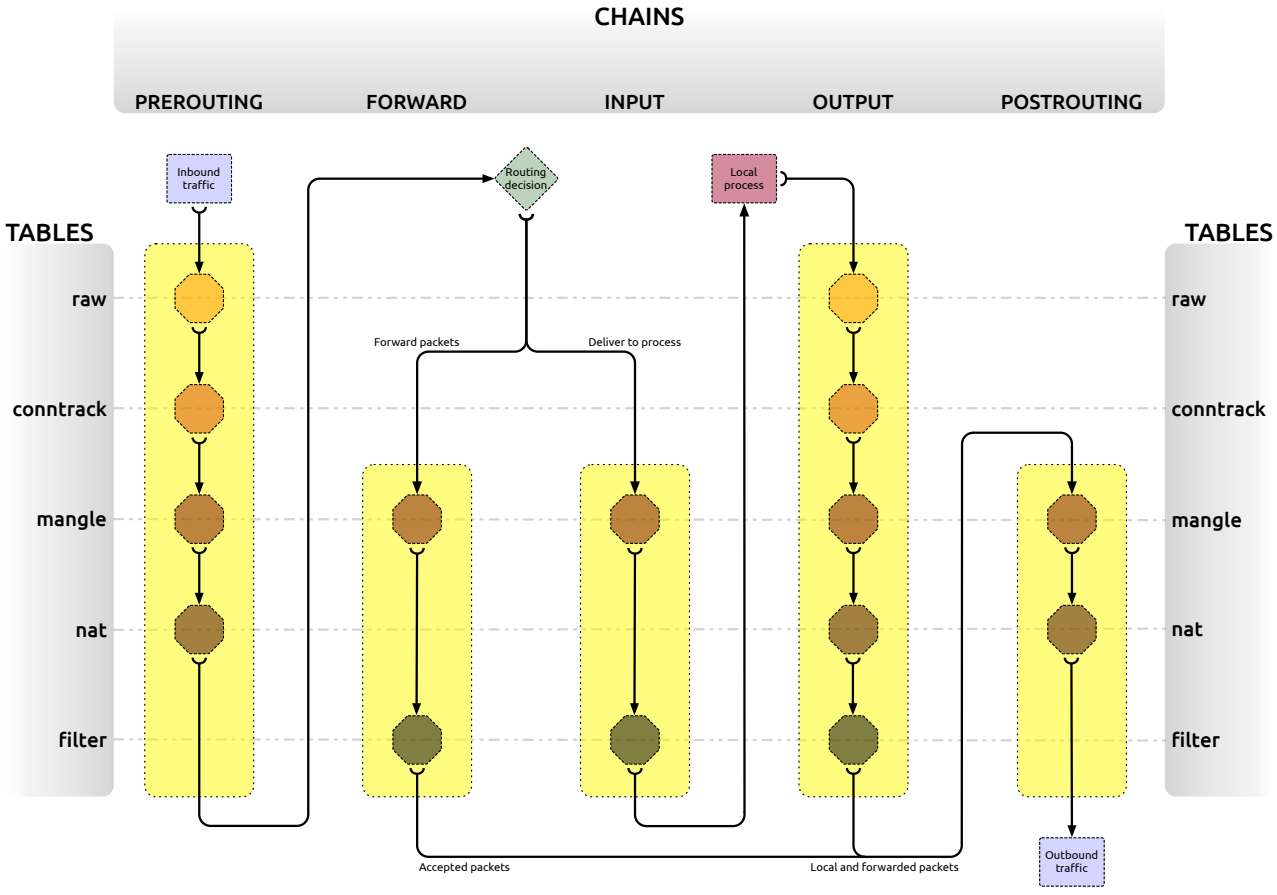


Figure 2: iptables flow graph

D Command Line Basics – in case you are not familiar with the shell

D.1 Manual Pager

You get a reference manual for almost all command line programs by issuing `man PROGRAMNAME`, e. g., `man iptables`. It uses the same shortcuts as the file viewer `less`, which provides the following actions: Pressing Q will exit the viewer (quit). To scroll up or down half a page press Ctrl+U or Ctrl+D (or Space/Shift+Space for whole page scrolling). `/keyword` starts a search for "keyword" (only downwards from current position). Pressing N or Shift+N will get you to the next or previous occurence of the keyword. G or Shift+G will goto the beginning or end of the file.

D.2 Viewing a File

Use `ls` to list the files in the current directory (`cd ..` and `cd DIRECTORY` to change the directory). For viewing a file there are several possibilities, `less FILENAME` being one of the more convenient ones (see Section D.1 above for navigation shortcuts). If you want to read a file continuously (because you expect data to be written to it by another program while reading it) you can use `tailf FILENAME`, which you have to exit by pressing Ctr+C.

D.3 Editing a File

There are several file editors you can use on the command line, but if you are not familiar with `vim` or `emacs`, the best choice is probably to use `nano`. You open a file for editing by calling `nano FILENAME`. Then you can edit the file and move around with arrow keys. All available commands are displayed at the bottom, where ^ denotes the Ctrl key, so you can save the file ("WriteOut") with Ctrl+O and exit nano with Ctrl+X.

D.4 Background Jobs

You can use the `&` character after any command that may take more time, to start it running in background (so that you can continue to use the terminal while the command is executed).
For example `nmap -sP 10.0.0.0/20 > scanresults.txt &`
Alternatively, you can press Ctrl+Z (suspend), which stops the current job and sends it to the background. To see all jobs were sent to the background, run `jobs` which also prints out a number for each job. `bg JOBNUMBER` or `fg JOBNUMBER` can be used to either continue a job in the background or foreground. Pressing Ctrl+C while a job is running in foreground will kill (exit) it.

D.5 Clipboard – Copy+Paste

Note that Ctrl+C on the command line is used to kill the current program. Most terminals support Ctrl+Shift+C (after selecting lines with the mouse) and Ctrl+Shift+V as shortcuts for copy and paste to and from the system clipboard. Using the middle mouse button to paste selected text is also supported frequently.