Lab 1.2. TCP Advanced Concepts

Objectives

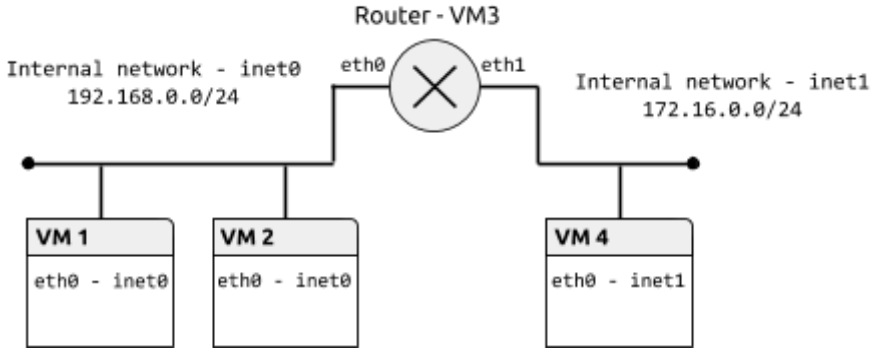
In this lab, we will study how TCP works. Also, we will see some parameters to tune the behaviour of TCP applications. Finally, we will learn how to configure NAT with `iptables`.

Contents

Environment Preparation

We will configure the network topology shown in Figure 1. We will use IPv4 addresses.

Figure 1: Network topology and addressing for this lab.



The content of the topology configuration file should be the following:

```
netprefix inet
machine 1 0 0
machine 2 0 0
machine 3 0 0 1 1
machine 4 0 1
```

Finally, configure the network interfaces of all machines of the network according to the following table. After configuring all machines, check the connectivity with the `ping` command.

Machine	IPv4 Address	Comments
VM1	`192.168.0.1/24`	Add Router as the default router
VM2	`192.168.0.2/24`	Add Router as the default router
VM3 - Router	`192.168.0.3/24 (eth0)` `172.16.0.3/24 (eth1)`	Enable packet forwarding
VM4	`172.16.0.1/24`	Add Router as the default router

TCP Connection Status

In this part we will use the Netcat tool (nc command), which allows reading from and writing to network connections. Netcat is very useful to research and debug the behaviour of a network at the transport layer, as it allows the user to specify many connection parameters. Moreover, we will use the `netstat` tool to see the status of network connections.

Exercise 1. Consult the man pages for `nc` and `netstat`. In particular check the following options of netstat: `-a`, `-l`, `-t`, `-n` and `-o`. Try some of the options for both commands to familiarize with their behaviour.

Exercise 2. (LISTEN) Start a TCP server in VM1 on port 7777 using command `nc -l 7777`. Check the connection status of the server with command `netstat -ltn`. foto1

Exercise 3. (ESTABLISHED) In VM2, start a client connection to the server started in the previous exercise using command `nc 192.168.0.1 7777`.

- Check the connection status and identify its parameters (IP address and port) using the `netstat` command. foto2
- Restart the server in VM1 using command `nc -l 192.168.0.1 7777`. Check that the connection is not possible from VM1 using `localhost` as destination address. Observe the difference with the previous command (without the IP address) using `netstat`. foto3
- Start the server and interchange a single character with the client. With the help of `wireshark`, observe the messages interchanged (specially, sequence numbers, acknowledgement numbers and TCP flags) and calculate how many bytes (and messages) were needed. foto4

Exercise 4. (TIMEWAIT) Close the connection in the server (with Ctrl+C) and check the connection status with `netstat`. Use `netstat` with option `-o` to determine the value of the TIMEWAIT timer.

Exercise 5. (SYN-SENT and SYN-RCVD) The `iptables` command can filter packets depending on TCP flags of segments (option `--tcp-flags`):

- Set a rule to filter connections on the server (VM1) in a way that puts the client in SYN-SENT state. Check the result with the `netstat` command in the client (VM2).
- Set a rule to filter connections on the client (VM2) in a way that puts the server in SYN-RCVD state. Check that this rule also puts the server in LAST-ACK state after closing the connection (Ctrl+C) in the client.
- Using `netstat` with option `-o` determine how many retransmissions are done and how often.

Note: The rule must be restrictive enough to only affect connections to the server. After each exercise, remove all filtering rules.

Exercise 6. Try a connection to a closed port in the server (e.g. 7778) and, using `wireshark`, observe the TCP segments interchanged, specially the TCP flags.

Introduction to the Security of TCP

Different aspects of TCP can be used to compromise system security. In this part, we are going to study two of them: TCP SYN flood DoS attacks and port scanning techniques.

Exercise 1. A TCP SYN flood attack consists in saturating a server with the massive sending of SYN messages.

- (Cliente VM2) To prevent that the attacker replied the SYN+ACK message from the server with a RST message (that would free the connection), we will block SYN+ACK messages in the attacker

with `iptables`.
- (Client VM2) To send TCP segments with the parameters of interest we will use the `hping3` command (consult the `man` page). In this case, we will send SYN messages to port 22 of the server (`ssh`) as fast as possible (*flood*).
- (Server VM1) Analyze the behaviour of the machine, in terms of received packets. Check if it is possible to connect to the `ssh` service.

Repeat the attack disabling the SYN cookies mechanism in the server with the `sysctl` command (`net.ipv4.tcp_syncookies` parameter).

Exercise 2. (CONNECT technique) Netcat can scan ports using the CONNECT technique, which tries to establish a connection with a given port. Depending on the answer (SYN+ACK or RST), it is possible to determine if there is a process listening.

- (Server VM1) Start a server in port 7777.
- (Client VM2) Scan the port rank 7775-7780 using `nc`. In this case, use options for exploration (`-z`) and verbose output (`-v`). **Note:** The version of `nc` installed in the VM does not support port ranges. Therefore, you have to do this manually or you can include the command to scan one port in a `for` loop.
- Observe the packets interchanged with `wireshark`.

Optional. The `nmap` tool can perform different port scan techniques, using more efficient strategies. This techniques (SYN stealth, ACK stealth, FIN-ACK stealth…) are faster than the previous one and are based on the operation of TCP. Consult the `man` page of `nmap` (section PORT SCANNING TECHNIQUES) and use different techniques to scan server ports. Observe with `wireshark` the messages interchanged.

TCP Options and Parameters

The behaviour of a TCP connection can be controlled with several options that are included in the header of SYN messages and can be configured by the operating system. The following table includes some of these options and their associated kernel parameters:

Option TCP	Kernel parameter	Purpose	Default value
Window scaling	`net.ipv4.tcp_window_scaling`		
Timestamps	`net.ipv4.tcp_timestamps`		
Selective ACKs	`net.ipv4.tcp_sack`		

Exercise 1. Using the `sysctl` command and the recommended bibliography, complete the previous table.

Exercise 2. Start a server on port 7777 and make a connection from the client VM. Using `wireshark`, study the value of the options interchanged during the connection. Vary some of the previous parameters (e.g. don't use selective ACKs) and observe the result in a new connection.

The following kernel parameters can be used to configure the keepalive timer:

Kernel parameter	Purpose	Default value
`net.ipv4.tcp_keepalive_time`		
`net.ipv4.tcp_keepalive_probes`		

`net.ipv4.tcp_keepalive_intvl`		

Exercise 3. Using the `sysctl` command and the recommended bibliography, complete the previous table.

Network Address Translation (NAT) and Port Forwarding

In this part, we will assume that the network connection in Router (VM3) with VM4 is public and can't route traffic from `192.168.0.0/24`. Also, we will suppose that Router's IP address is dynamic.

Exercise 1. Configure dynamic address translation in Router :

- (VM3 - Router) Using the `iptables` command, configure Router to perform SNAT (masquerade) on interface `eth1`.
- (VM1) Check the connectivity between VM1 and VM4 with the `ping` command.
- (VM4 and VM1) Using `wireshark` determine the source and destination IP address of the ICMP Echo Request and Reply messages in both networks. Which parameter is used, instead of source port, to match Echo replies with Echo requests? Check the output of command `conntrack -L` or, alternatively, file `/proc/net/nf_conntrack`.

Exercise 2. Configure access to a server in the private network:

- (VM1) Start a Netcat server on port 7777.
- (VM3 - Router) Using `iptables`, forward connections (DNAT) from Router's port 80 to VM1's port 7777.
- (VM4) Connect to port 80 of Router with `nc` and check the result in VM1. Analyze the traffic with `wireshark`, specially source and destination ports and IP addresses in both networks.