Using a nested virtualization tool for teaching VPN fundamentals

Juan M. Orduña[1] · Carlos Pérez[1]

Abstract

A common problem of undergraduate courses about security and computer networks is the difficulty of providing practical exercises to students. Although different approaches have been followed to solve this problem, it is still an open issue. In this work, we first discuss several possible open-source simulation/emulation/virtualization tools that could be used in the design of a lab session for teaching Virtual Private Networks (VPN) fundamentals based on OpenVPN. Next, we describe how a hands-on lab about VPNs can be carried out with the selected virtualization tool, called NETinVM. The proposed lab starts installing the OpenVPN packages in client and server machines, as in a real-case scenario. Although in this case both client and servers are Kernel-based Virtual Machines (KVM) connected to networks inside NETinVM. The nested virtualization included in NETinVM allows to deploy all the required types of VPNs within the virtual machine, allowing the students to reproduce the lab without the need of a network infrastructure, and following exactly the same process required in a real scenario. The evaluation results show that the level of learning achieved by the students through this lab session seems to be high, and they consider this lab useful for their professional development, showing that virtualization technology can help in computer network education.

Keywords Nested virtualization tools · Computer networks lab · Network security · Problem-based learning · Virtual private networks

1 Introduction

A common problem of computer network and computer science courses is the difficulty of providing practical exercises to students. Although it has been proven that students learn more effectively from courses that provide for involvement in practical activities Sarkar [1]; Trabelsi and Alketbi [2]; O'Grady [3]; Carter [4], the main obstacle for developing such practical activities is the complexity and efforts needed to devote the required infrastructure to a single lab. In particular, practical exercises of some network advanced concepts, like firewalls, virtual lans (vlans) or virtual private networks (vpns) require either a physical

Juan M. Orduña and Carlos Pérez have contributed equally to this work.

✉ Juan M. Orduña
juan.orduna@uv.es

Carlos Pérez
carlos.perez@uv.es

[1] Departamento de Informática, Universidad de Valencia, Avda. Universidad, s/n, 46100 Burjassot, Valencia, Spain

infrastructure separated from the actual campus network (to avoid security issues), or simulation/emulation tools that faithfully reproduce a similar infrastructure. For these kind of courses and concepts, students may require multiple computer systems with complex configuration, but it is not feasible for higher education institutions to provide independent multiple computer systems to every student enrolled in the course Bhat et al. [5]. Even in the case of having such infrastructure, our experience of teaching computer networks and computer architecture labs during 30 years shows, many times the building and configuration of the real hardware of a network model (switches, routers, cabling, etc.) or the configuration of a simulation tool (OMNET++ Cogitative Software FZE [6], GEM5 Binkert et al. [7], etc.) is too complex and takes too much time of the lab session, leaving a short time for practising the advanced network concepts that were meant to be practised on that real hardware or simulator. This issue significantly reduces the effective learning of the lab sessions.

Although the advent of Software Defined Networks (SDNs) has allowed the development of tools like Cisco Packet Tracer Cisco Networking Academy [8], OMNET++ Cogitative Software FZE [6], or Mininet

Contributors [9] to simulate network equipment, those concepts exceeding the scope of the network, like SNMP, VPNs, certificates, RSA, etc. are quite complex to be practised with this kind of tools, since a virtual network infrastructure must be built and configured as a previous step to develop the lab session. The previous building and configuration of the virtual network infrastructure requires a huge effort from the professor. Also, some cloud-based virtualization tools for online teaching were proposed Salah [10]; Willems et al. [11]; Abraham [12]; Xu et al. [13]. Nevertheless, the use of cloud services adds some drawbacks to virtualization tools for teaching. drawbacks. First, the use of a given cloud infrastructure forces the user to learn and use a concrete technology and services, making the course dependent on a given service provider. Second, the number of students in a given course may require a cloud infrastructure size that exceeds the maximum size that the provider offers for free, increasing the cost of the course. Third, the use of cloud resources may add significant latencies that affect the interactivity of the exercises. Fourth, the use of cloud technologies like containers can greatly reduce the realism of the proposed activity, since the configuration of the network interfaces, disk and other elements are different from a real (or virtual) machine. Finally, the reproducibility and usability along time is seriously affected, since students are not guaranteed that the cloud infrastructure is accessible some time after the course finishes Son and Fitzgibbons [14]. These drawbacks also persist when then online access is provided by a university-owned infrastructure.

In order to solve these problems, in this work we propose an existing nested virtualization tool, called NETinVM, for teaching VPN fundamentals. NETinVM was developed Pérez et al. [15] some years ago. It is a tool based on nested virtualization (virtual machines and a network infrastructure inside a virtual machine) that includes a fully functional lab in a single virtual machine. This lab comprises three interconnected networks with several computers attached to each network, providing a portable and realistic scenario for teaching courses related to security, system administration and computer networks. With this tool, students can carry out complex exercises without changing the network and computer infrastructure at all, using the same commands and software that they would use in a real scenario. They can reproduce the results of the proposed exercises in a portable and autonomous way. The advantage of NETinVM over all the alternatives described above is that the required network infrastructure is already built, configured and fully operational in the NETinVM virtual machine, with no additional effort from the teacher to build and configure the network for the lab session. No network latencies can affect the interactivity of the exercises, the activity carried out is absolutely realistic,

since linux (virtual) machines are used to build and configure the VPN, and the reproducibility and usability is guaranteed because the tool is just a virtual machine that the student can save an execute in any computer. We first discuss several possible open-source simulation/emulation/virtualization tools that could be used for designing a lab session about Virtual Private Networks (VPN) fundamentals based on OpenVPN. Next, we show and explain the development of the proposed lab session with the selected virtualization tool, called NETinVM. Starting from scratch, the lab begins installing OpenVPN on the required KVM machines in NETinVM. Then, different VPN configurations are deployed and analyzed, ranging from simple, point-to-point VPN with no encryption nor authentication launched from the command line, to a client/server mode VPN using a Public Key Infrastructure (PKI) with X.509 certificates, and configured by means of a configuration file. In this way, not only VPN concepts are practised, but also security implementations like cipher and authentication algorithms, X.509 certificates, and firewall configuration using IPtables. Evaluation results show that the level of learning achieved by the students is high, and they consider this lab useful for their professional development, showing that virtualization technology can also help in computer network education.

The rest of the paper is organized as follows: Sect. 2 discusses several possible simulation, emulation or virtualization tools that could be used for the proposed VPN lab. Next, in order to make this paper self-contained, Sect. 3 describes the NETinVM features and configuration required to carry out the exercises described in the next sections. Then, Sect. 4 detail the session lab and how it is implemented using NETinVM. Next, Sect. 5 shows the measurement of the learning impact of this activity on students, starting from a survey provided anonymously in the academic year 2022–2023. Finally, Sect. 6 shows some conclusions remarks and future work to be done.

2 Network simulation tools

Last years a lot of software tools, both open-source as well as commercial ones, have become available for simulating a network infrastructure. Among the most extended commercial tools for building and simulating network infrastructure we can cite Cisco Packet Tracer Cisco Networking Academy [8], Riverbed modeler Riverbed Technology [16] (previously called Opnet), or Boson NetSim Boson, LLC [17]. Among the most extended open-source tools for network simulation we can find OMNET++ Cogitative Software FZE [6], GNS3 SolarWinds Worldwide [18], or Mininet Contributors [9], among others.

Since the purpose of this work paper is to provide readers with a way of learning about VPNs with no additional resources or funding, we discarded commercial tools like Riverbed or Cisco Packet Tracer for this lab. Although these tools have academic licenses, the temporal validity of these licenses could limit the utility of this work. Additionally, they do not completely fit the proposed lab session. For example, Cisco Packet Tracer (at least in its 7.3.1 release) only allows ms-dos commands in the prompt utility for the computers attached to the network infrastructure.

Regarding the open-source available tools, all of them could be used for lab teaching Virtual Private Networks (VPN) fundamentals based on OpenVPN. Nevertheless, the advantage of NETinVM Pérez et al. [15] over these tools is that the required network infrastructure is already built, configured and fully operational in the NETinVM virtual machine, with no additional effort from the teacher to build and configure the network for the lab session. Unlike the case of OMNET++ or Mininet, the teacher does not need to program (in C++, ned or python) nor configure the hosts and routers in order to have a infrastructure on which the lab session can be carried out.

Additionally, NETinVM uses nested virtualization instead of simulation, which allows the deployment of other VPN technologies like Wireguard VPN tunnels Jason A. Donenfeld [19] just by exactly following the documentation for its installation on Linux systems. In this sense, we completely carried out the process described in the guide "How to Install Wireguard VPN on Debian 11" Howtoforge Linux tutorials [20]: we installed wireguard on two hosts of NETinVM, configured them and connected a wireguard client to a server just following the exact commands shown in the tutorial. This NETinVM feature significantly facilitates the design of open exercises to explore new solutions and keep up to date with new technologies. Moreover, since NETinVM uses off-the-shelf components to build networks and computers, it can also be used in many other network-related learning scenarios beyond VPNs, like construction and analysis of firewalls with iptables or nftables, network monitoring using SNMP, or traffic analysis using TCPDUMP and/or Wireshark, etc.. Although some of the functionality can be attained by using containers instead of nested virtual machines, doing so would limit the usefulness of NETinVM. For example, partitioning a disk or configuring IPTABLES can't be done inside a container. Furthermore, if the objective is to show how a VPN server can be deployed in a container, this can be also done in NETinVM, since containers can run inside the nested virtual machines. Nevertheless, the use of containers can reduce the realism of the proposed lab session in this work, since the configuration of the network interfaces, disks and other elements are different from a real (or virtual) machine.

Due to all these reasons, we selected NETinVM as the tool for the proposed lab session.

3 Background

NETinVM is a single VMware virtual machine image that contains, ready to run, a series of Kernel Virtual Machine (KVM) virtual machines which, when started, conform a whole computer network inside the VMware virtual machine. Hence the name NETinVM, an acronym for NETwork in Virtual Machine. Each of the KVM machines can be independently run. There are some pre-configured scripts to run and stop the desired KVM machines, and a backup of the whole system can be saved or retrieved at any moment. For a detailed description, full documentation and tutorials, the NETinVM web page can be consulted Pérez and Pérez [21]. When started, the KVM machines form a computer network named "example.net" whose logic structure is shown in Fig. 1. It consists of an "external" network to simulate Internet. Between the external network and the corporative network there is a firewall, consisting of a station called "fw". The firewall is directly attached to a perimetric network (called DMZ) and the internal network, each one with several computers. This structure allows to simulate different situations, including the building and configuration of VPNs between the external and internal networks.

The physical structure of NETinVM is shown in Fig. 2. NETinVM is pre-configured to create three interconnected virtual networks, playing the role of the corporate, perimeter and external networks of an organization. These
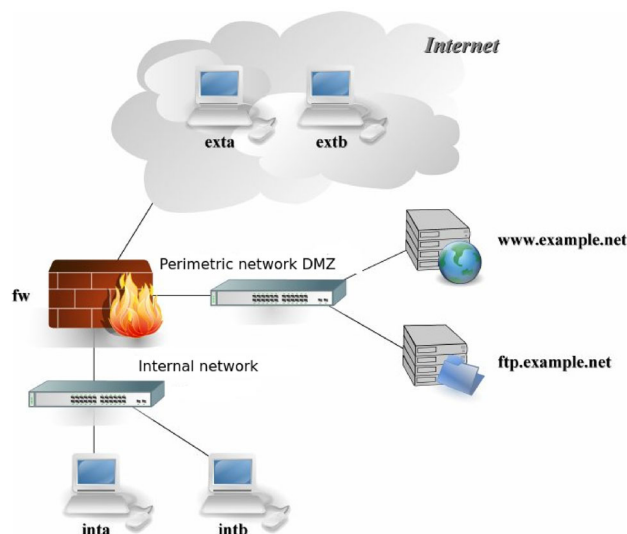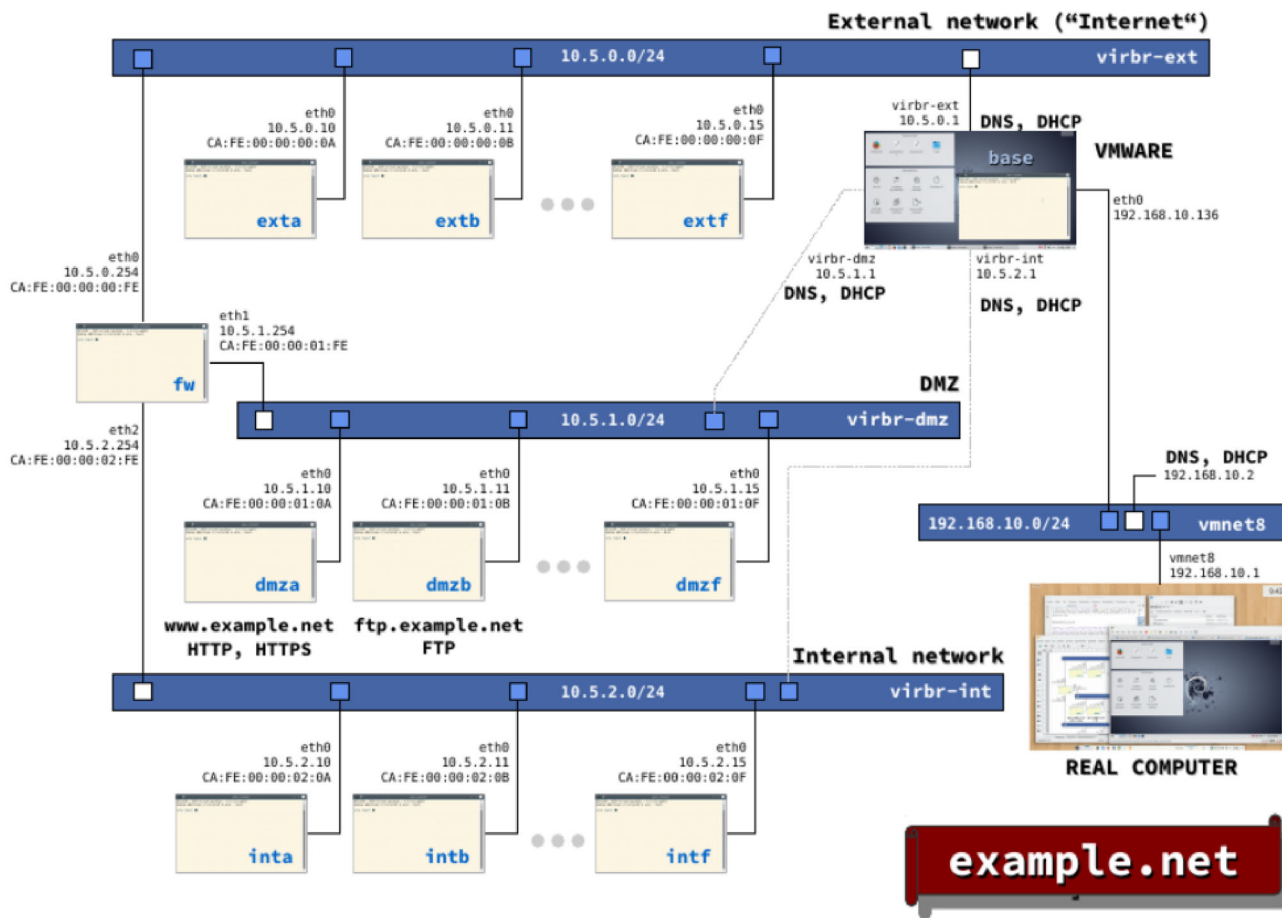

Fig. 1 Logical structure of NETinVM

Fig. 2 General structure of NETinVM

networks are named "int" (for internal network), "dmz" (for DMZ or demilitarized zone, which is often used as a synonym for perimeter network) and "ext" (for external network). One of the KVM machines, "fw" (for firewall), interconnects the three networks providing communication and packet filtering, as shown in Fig. 2. The rest of KVMs have a single network interface connected to the network they are named after, as follows (where X can be from "a" to "f"): *intX* KVMs are connected to the internal network. These machines only offer the SSH service. *dmzX* KVMs are connected to the perimeter network (DMZ). They are conceived as bastion nodes. In this network there are two machines with alias. "dmza" has the alias "www.example.net" and it provides HTTP and HTTPS services; "dmzb" has the alias "ftp.example.net" and it offers FTP. Finally, *extX* KVMs are connected to networks that are external to the organization (e.g., "Internet"). These three networks are connected through *base* to VMware's "vmnet8" (NAT) virtual network, which allows the connection of KVM to external (real) networks.

The default gateway for the internal and perimeter networks (machines "intX" and "dmzX") is "fw", and for "fw" is the IP address of "base" in the "ext" network. The machines on the external network ("extX") have "base" as the default gateway, and "fw" as the gateway to access the perimeter and internal networks. Therefore, the traffic among KVM machines of the three networks always goes through "fw", while the traffic directed to machines outside "base" goes through "fw" if and only if it comes from the internal or the perimeter networks. All the traffic to the outside world goes through "base", which has IP forwarding and NAT enabled and routes back external traffic through the "ext" network. Communications between "base" and any KVM machine are carried out directly, without passing through "fw" (provided that the IP of "base" corresponding to the network of the KVM machine is used). This arrangement is convenient because it allows access from "base" to all KVM machines using SSH, regardless of the configuration of routing and packet filtering in "fw". The KVM machines can communicate to

each other via standard network protocols. All KVM machines have the SSH service enabled by default and there are bastion nodes offering HTTP and FTP services, but any other standard IP service can be also configured (NFS, SMTP,...).

Since the first paper about NETinVM was published Pérez et al. [15], significant implementation changes have been added to NETinVM in successive updates, although the basic design remains unchanged. The most relevant updates are the use of KVM (Kernel Virtual Machines) to provide nested virtualization, the use of OVS (Open Virtual Switch) for building virtual networks, and the use of Libvirt (with "virt-manager" and "virsh") to manage virtual machines, storage and networking in "base". Furthermore, since "virt-manager" provides each machine with a graphical console, now it is possible to selectively start a LXDE desktop in the KVM machines Carlos Pérez and David Pérez [22]. Since all of these new components are mainstream, NETinVM can now be used to teach and learn about virtual machine management (in addition to its traditional use for learning about security, operating systems and computer networking).

Using this virtual infrastructure, the next section describe a hands-on lab session that guides the student through the setting up and configuration of different VPNs.

4 VPN lab session

In this section, we describe the main features of the lab session about VPNs designed with the nested virtualization tool NETinVM, highlighting the exercises and questions that the NETinVM features allow to be asked to ensure the students understand the mechanisms underlying the openvpn commands.

4.1 Methodology

The lab session proposed in this work was carried out first time in the 2022–2023 academic year in the official course of Architecture of Computer Networks, a course in the third year of the Degree of Computer Engineering at University of Valencia (Spain). The course is part of several network courses, and it focuses on the higher layers of the TCP/IP protocol stack, as well as security issues in networks. The course includes 4 mandatory, on-site labs which students must attend, under the guidance, support and supervision of an instructor (professor). Each lab group has between 16 and 18 students, grouped in pairs.

The proposed lab session is scheduled just after half of the semester. Since the beginning of the course, all the material of the lab session is available to students, as if they want to carry out the lab session in advance. This material

includes a link to download NETinVM, and two documents: first, a short introduction to VPNs, including the VPN classification accepted by the Spanish Centro Criptológico Nacional, depending on the Ministry of Defense, Spain Centro Criptológico Nacional [23], and a brief explanation of OpenVPN internals Crist and Keijser [24]. Second, a session guide with the exercises to be carried out to install and configure a VPN within the NETinVM infrastructure.

The time scheduled for all the lab sessions is two hours and a half. The lab session starts with a written, very short exam (10 min) which students must take individually. This mini-exam asks questions about the concepts an steps explained in the two documents cited above, and its purpose is to ensure that students have previously studied them and they understand what they are going to do. Next, the students must follow the lab session guide, carrying out all the exercises and answering all the questions asked in that document. They must upload a report with all the exercises and answers at the end of the lab session. The activities indicated in the lab sessions guide are described in the next subsections, and are the same required to set up a real VPN on a real infrastructure.

Although in our case the lab session was carried out onsite, and additional advantage of using NETinVM is that the same lab session can be carried out online. In that case, videoconferencing tools like Zoom or Teams for tutoring sessions, since the student can share his/her screen and even give up the control of the NETinVM machine to the instructor. Also, remote connections using SSH can be done. In this case the students must connect through SSH to the instructor machine (it can be a virtual machine) and redirect ports to allow the SSH connection of the instructor.

For the sake of shortness, we have not included in this section all the commands used in the lab session. Those readers interested in the detailed commands for implementing in NETinVM both an OpenVPN-based VPN and a PKI infrastructure can check appendix 1.

Since the OpenVPN packages required for this lab are not installed by default in the KVM machines, we must first install them in all the machines involved in a VPN. In particular, these packages should be installed in the NETinVM machines extc, intc, and inta.

4.2 Point-to-point VPN using clear text tunnel

Once the required software is installed, the first step will consist in installing a point-to-point VPN using a clear text tunnel between the machines intc (acting as a client) and extc (acting as the server). We can directly follow the examples given in the OpenVPN reference book "Mastering OpenVPN" Crist and Keijser [24].

4.2.1 Setting up the VPN from the command line

In this case the setting of the VPN is done as a single command on two KVM machines. If everything has gone fine, the output on the client side should be something similar to the one shown in Fig. 3.

Figure 3 shows how the tun0 interface is opened and the connection is established. Using this output, some questions may be asked to the students:

- Which virtual interfaces is using the VPN?
- Which sockets (local and remote) are being used? UDP or TCP sockets?
- Which cipher algorithm is being used? Why?

At this point, students are asked to carry out the next exercise (we will denote as "Exercise A"): open new terminals on both machines to ping each other, and use the Wireshark tool in the NETinVM desktop to capture the traffic between both sides of the connection. In order to make students analyze the IP packet encapsulation in any of the captured frames, the next questions may be asked:

- Analyze the first fame captured. Have the IP addresses changed?
- Is there any VPN tunnel?
- Where can you see (if possible) the original IP addresses 10.200.0.2 and 10.200.0.1?

Figure 4 illustrates this exercise. It shows a snapshot of the Wireshark application, where a packet sent from intc (IP address 10.5.2.12) to extc (10.5.0.12) is selected. The HMAC field of the OpenVPN protocol, shown in the intermediate frame of the snapshot, includes the IP addresses used in the tunnel (0a c8 00 02 and 0a c8 00 01). Also, the lower part of the snapshot shows the hexadecimal values of the whole frame, and it can be seen that the encapsulated IP packet shows the source and destination addresses of the tunnel in hexadecimal format.


```
>  📌                        intc - console1              ∨ ∧ ⊗
root@intc:~# openvpn --ifconfig 10.200.0.2 10.200.0.1 --dev tun --remote  extc.
example.net
Fri Sep  9 16:55:30 2022 disabling NCP mode (--ncp-disable) because not in P2MP
client or server mode
Fri Sep  9 16:55:30 2022 OpenVPN 2.4.7 x86_64-pc-linux-gnu [SSL (OpenSSL)] [LZO]
[LZ4] [EPOLL] [PKCS11] [MH/PKTINFO] [AEAD] built on Apr 28 2021
Fri Sep  9 16:55:30 2022 library versions: OpenSSL 1.1.1d  10 Sep 2019, LZO 2.10
Fri Sep  9 16:55:30 2022 ****** WARNING ******: All encryption and authenticat
ion features disabled -- All data will be tunnelled as clear text and will not b
e protected against man-in-the-middle changes. PLEASE DO RECONSIDER THIS CONFIGU
RATION!
Fri Sep  9 16:55:30 2022 TUN/TAP device tun0 opened
Fri Sep  9 16:55:30 2022 /sbin/ip link set dev tun0 up mtu 1500
Fri Sep  9 16:55:30 2022 /sbin/ip addr add dev tun0 local 10.200.0.2 peer 10.200
.0.1
Fri Sep  9 16:55:30 2022 TCP/UDP: Preserving recently used remote address: [AF_I
NET]10.5.0.12:1194
Fri Sep  9 16:55:30 2022 UDP link local (bound): [AF_INET][undef]:1194
Fri Sep  9 16:55:30 2022 UDP link remote: [AF_INET]10.5.0.12:1194
Fri Sep  9 16:55:31 2022 Peer Connection Initiated with [AF_INET]10.5.0.12:1194
Fri Sep  9 16:55:31 2022 WARNING: this configuration may cache passwords in memo
ry -- use the auth-nocache option to prevent this
Fri Sep  9 16:55:31 2022 Initialization Sequence Completed
```

Fig. 3 Output in the client side (*intc* machine)

Next, the following exercise (which we will denote as "Exercise B"): using nc application is proposed: start Wireshark and capture the traffic in the "mirror-int" interface.[1] Since the VPN connection keeps busy console1 of both extc and intc,[2] we should login in console2 of these machines. We will login as root, and in console2 of both extc and intc we will use nc to generate messages to be transmitted through the tunnel.

4.2.2 Setting up the VPN from a configuration file

The next step in the lab session consist of introducing the student to the syntax of the main options that can be included in a configuration file, following the examples shown in Crist and Keijser [24]. The session guide should explain how each option specified in the command line can also be specified in a configuration file.

As an exercise to practise, the students are asked to repeat the last exercise but now using configuration files instead of the command line in both server and client.

At this point, the session guide should introduce the daemon and log options, explaining that the implementation as a daemon prevents any output to directed to the console. Also, it is important to highlight the use of the ps aux | grep concatenated commands in order to see whether the VPN is still being executed or not. As an illustrative example, Fig. 5 shows the configuration file which students must write on the client side.

4.3 Point-to-point VPN using pre-shared keys

The session guide explains that in this case the secret key should be generated and copied to both ends of the connection (using SCP, for example). The command option --genkey and the level of debugging --verb are explained. In order to copy the generated key, the shared directory among all the KVM machines in NETinVM is used[3]

At this point, the session guide asks the students to capture a snapshot with the output generated by the command issued in the server side. An example (with increased verbosity) is shown in Fig. 6

---

[1] "mirror-int" is a virtual Ethernet interface in "base" that is connected to a mirror port preconfigured in the "int" switch to make it easy to capture and analyze traffic in the internal network.

[2] "consoleX" is a terminal directly attached to a virtual serial port in the virtual machine. This avoids that this interface generates network traffic, making it easier to analyze captured network traffic.

[3] This shared folder is implemented using a "virtualfs" filesystem. "virtualf" is a Linux kernel's standard component that lets guest Linux systems access to a directory in its host's filesystem. This access has the advantage (over SMB, NFS, or any other network filesystem) that it doesn't generate any network packets (simplifying the analysis of network traffic).

Fig. 4 Wireshark capture showing the encapsulated IP addresses in the IP payload field
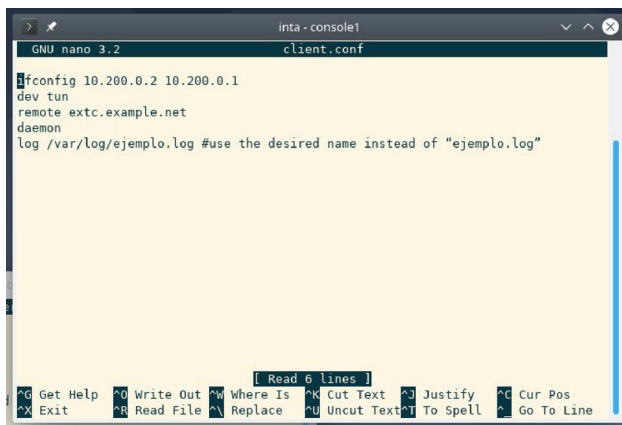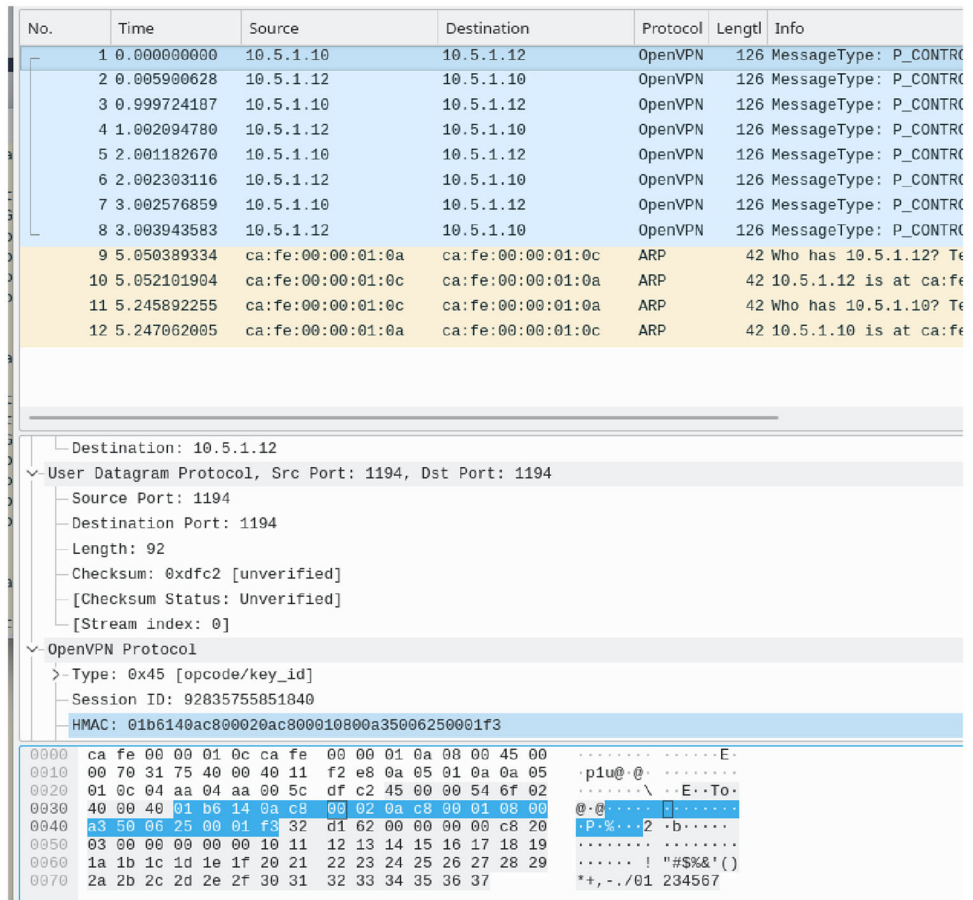
Fig. 5 Client configuration file

Also, students are asked to repeat the exercises A and B of section A.1.1 to see how a sniffer cannot capture data in clear text now, neither the tunnel IP addresses nor the text sent by the nc application.
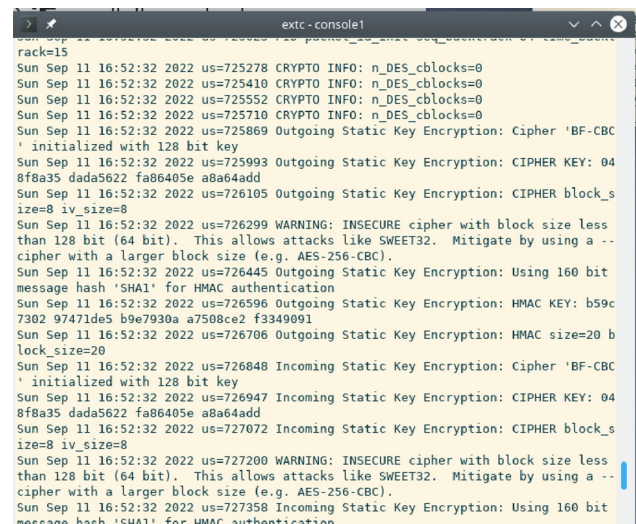
Fig. 6 Output delivered by the server to the openvpn command when used with the –secret and increased verbosity

4.4 PKI deployment. Client/server VPN

The last part of the lab session is devoted to the client/server mode, where the server is a single OpenVPN process to

which multiple clients can connect. Each authorized and authenticated client is assigned an IP address from an address pool managed by the server. Clients cannot communicate among them directly. Instead, all the traffic flows through the server (unless the "client-to-client" option is used and the appropriate rules are set in the firewall and/or routers).

In this part, the lab session will try to emulate the university network with NETinVM. The VPN server will be extc, one client will be installed in intc, and another client will be installed in inta, and from both machines we will access the http server through the VPN tunnel as if these machines had another IP addresses. Also, the need for establishing a PKI infrastructure is explained. In this session we use the easy-rsa package for generating the items required to set up the PKI, which are:

- A private key and a X.509 certificate for the Certifying Authority (CA) who signs documents.
- A private key and a X.509 signed certificate for the server.
- A private key and a X.509 signed certificate *for each* client.
- A Diffie-Hellman group for the server

In NETinVM, the shared directory ∼/shared is used to copy the files that should be created in the server (extc) and then copied to each client. Also, we use the file ∼/easyrsa/vars.example as a template to configure the information to be specified in the certificates. Using the Easy-RSA−3.0.7 package, installed in extc, the students are able to practise the whole process of generating the PKI for the server (extc) and the two clients (intc and inta). Files can be copied between machines using the aforementioned shared directory.

Once the PKI infrastructure has been set up, the guide explain the configuration file for the server:

Listing 1 Server configuration file
```
proto udp
port 1194
dev tun
server 10.200.0.0  255.255.255.0
topology subnet
persist-key
persist-tun
keepalive 10 60
dh /etc/openvpn/keys/dh.pem
ca /etc/openvpn/keys/ca.crt
cert /etc/openvpn/keys/servidor.crt
key /etc/openvpn/keys/servidor.key
tls-auth /etc/openvpn/keys/ta.key 0
user nobody
group nogroup \# use 'group nogroup' on Ubuntu
verb 3
daemon
log-append /var/log/openvpn.log
```

and the one for the clients:

Listing 2 Client configuration file
```
client
proto udp
remote extc.example.net
port 1194
dev tun
nobind
remote-cert-tls server
tls-auth /etc/openvpn/keys/ta.key 1
ca /etc/openvpn/keys/ca.crt
cert /etc/openvpn/keys/client2.crt
key /etc/openvpn/keys/client2.key
```

It must be noted that the files "client2.key" and "client2.crt" should be used in inta, but these files should be "client1.key" and "client1.crt" in intc.

In order to test the configuration, it is required that Wireshark is installed on extc. The client/server exercise proposed to the students in the session guide consists of starting the LXDE desktop in extc (the session guide references section "Taking advantage of KVM and libvirt" of NETinVM documentation Carlos Pérez and David Pérez [22]), and two instances of Wireshark Sharpe et al. [25]. The students should start one of the instances to capture the traffic traversing the "tun0" interface, while the other should capture the traffic traversing the "eth0" interface. Then, students should issue ping commands to the VPN server (ping -c 3 10.200.0.1) from the intc console, and compare the frames captured by both interfaces, explaining the differences. Figure 7 shows the different frames displayed by each interface.

The final step is to enable IP routing in the server (extc), allowing the clients (inta and intc) to use the VPN tunnel to access other machines in the server network (10.5.0.0/24). The exercise asks the student to write down the configuration file for a client/server VPN analogous to the one of the University of Valencia, where the server emulating vpn.uv.es is extc, and there are two clients emulating the laptop of both students at their homes: inta and intc. However, in this case the tunnel will use the range of IP addresses 10.5.0.48/28, allowing the use of the IP addresses 10.5.0.49−10.5.0.62. The PKI infrastructure should assign the client1 certificate to inta, which will establish the VPN in first place, and the client2 certificate to intc, which will establish the VPN in second place. The students are asked to execute in extc the command `tcpdump -i tun0 -n` and then execute in intc the command `ping -c 4 10.5.0.50`

The students should show the configuration files in both server and clients, and they are asked to detail and explain the differences existing between the output of the ping in intc and the output of the tcpdump in extc. The output in both machines are shown in Fig. 8. This figure shows how
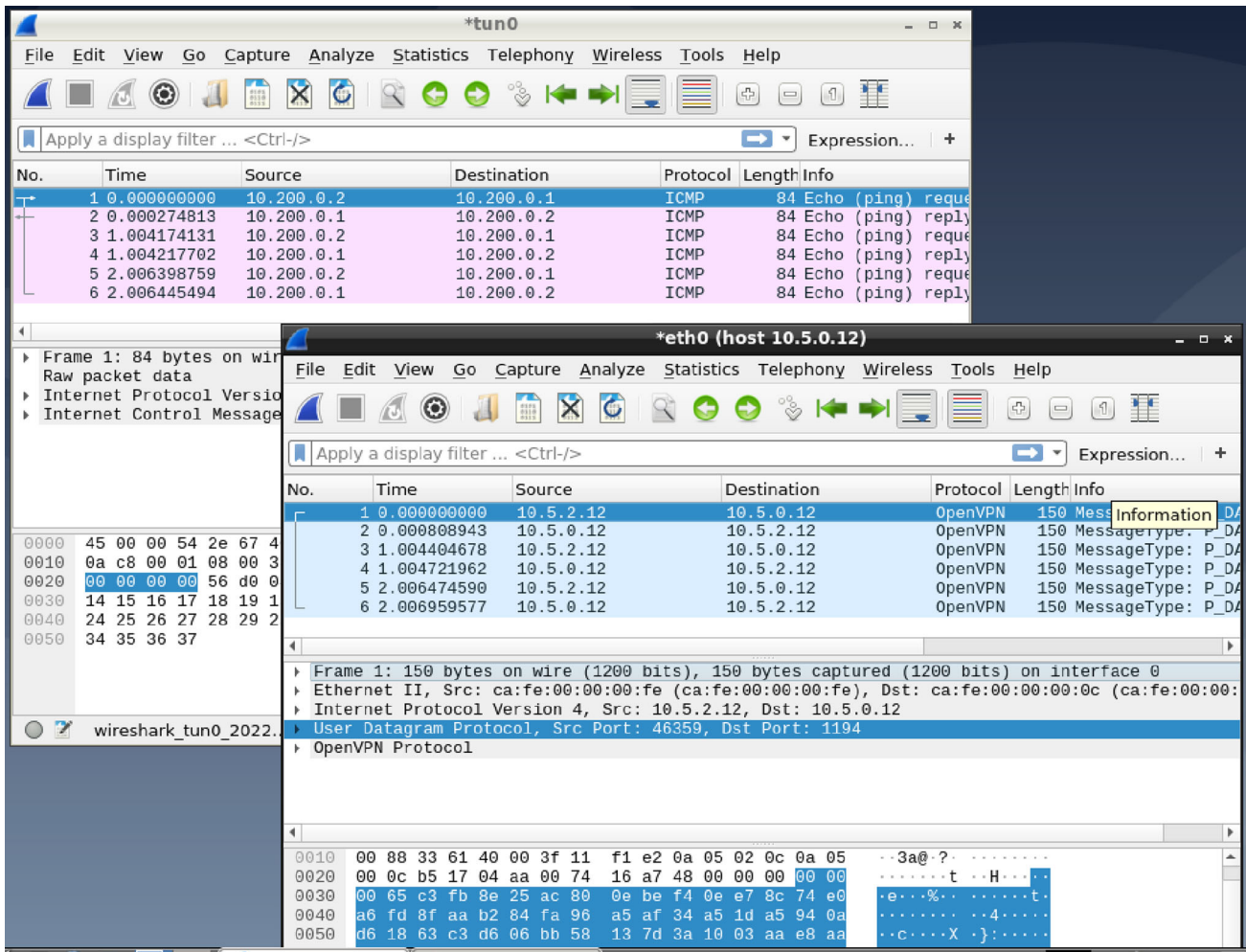
Fig. 7 Frames captured by tun0 and eth0 interfaces at extc machine when pinging from intc to the VPN tunnel address 10.200.0.1
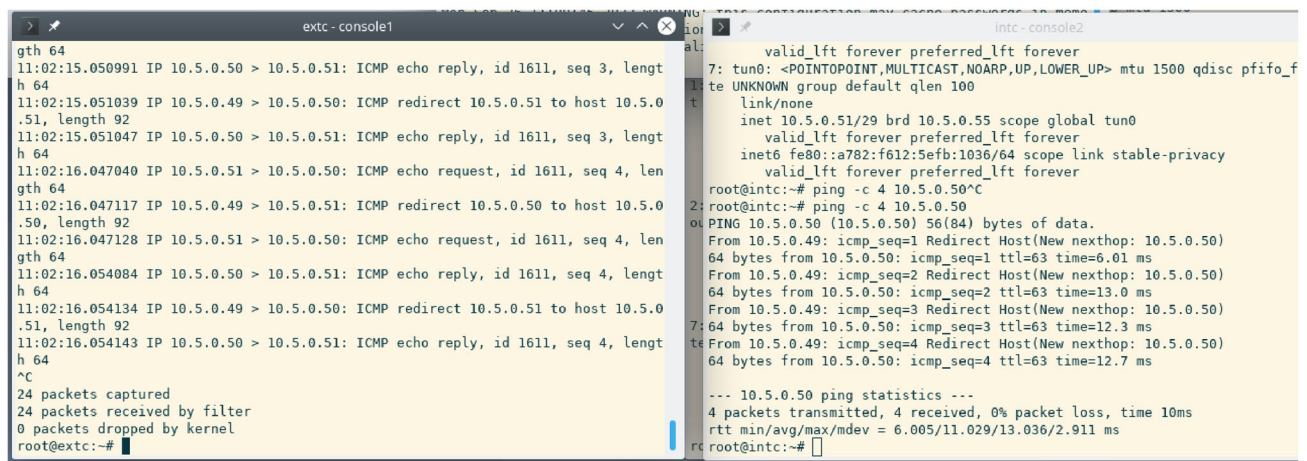

Fig. 8 Output of the tcpdump command (in extc) and the ping command (in intc) showing the redirection carried out by the VPN

the redirect messages are shown in both client and server, and the differences in the number of messages exchanged.

5 Learning impact

Traditional Lecture-Based Learning (LBL), where the teacher makes an oral presentation intended to present the main concepts of the course, is usually complemented in Computer Science education with exercises or labs to be carried out by the students Dehbozorgi et al. [26]. This is the case for a Computer Networks course at University of Valencia. This is a mandatory course scheduled in the third year of the Degree of Computer Engineering. The course is part of several network courses, and it focuses on the higher layers of the TCP/IP protocol stack, as well as security issues in networks. Since one of the goals of this course is to provide the students with practical skills in computer networks, we have extended the traditional LBL model with the lab session described in Sect. 4, introducing students to the details of VPNs. NETinVM is run locally by students during the lab session, with the on-site guidance and support of the professor, although it could also be made available online. In fact, we have used NETinVM in other online courses. When this approach is used, the student can share his screen and, if necessary, even let the instructor control his copy of NETinVM.

The feasibility of the proposed LBL activity (the lab session) fully relies on NETinVM, since the set up of different VPNs would significantly affect the configuration of the actual network infrastructure of the University, taking into account that 45 students were enrolled in the course. Therefore, we asked the students to evaluate the activity, instead of the tool. Concretely, we made an anonymous and voluntary survey, asking the students (grouped by teams formed by two or three students) to evaluate the proposed activity, carried out as the 4th lab session. In particular, we asked two questions with button answers, where the students selected one among the four lab sessions carried out during the course. Also, a third question asked for a mark (ranging between 0 and 10 points, being 10 the best possible one) to evaluate the activities carried out in the proposed lab session. Nevertheless, only 23, 17 or 24 students answered the different questions of the survey. Since the necessary number of user answers in studies focused on human-computer interaction should be no less than 40 Hornbaek [27], no statistically significant studies can be performed on the answers of the 23 students who answered the survey (it was a voluntary, anonymous survey). Nevertheless, some general ideas can be extracted.

The first question of the survey was in which of the lab sessions of the course did the student think that he/she had learned the most about the topic of the session. Table 1 shows the results for this question, both in absolute values and in terms of percentage. This table shows that almost 40% of the students considered that this lab session had provided them with the highest level of learning over the rest of lab sessions.

The other question asked the students which lab session they thought it was the most useful one for their professional activity. Table 2 shows the results corresponding to this question, which was answered by 17 students. Again, the proposed lab session was selected by more than 40% of the students.

Finally, the students were requested to evaluate the activities carried out in the proposed lab session by selecting a mark ranging between 0 and 10 points, being 10 the best possible one. Table 3 shows the results of the survey. Each column shows each of the possible values (ranging from 0 to 10) answered, and the only row in the table shows, for each of these values, the number of students who answered that value. A total of 24 students answered this question. Around a 74% of the students marked the activity with 7 points or higher, and the average mark is 7,17. These results show that students at least do not consider that the tool used in the lab session reduces the potential learning, as a real network infrastructure often does. Therefore, we can conclude that the proposed lab session is a feasible way of introducing students in actual but complex computer network concepts like VPNs.

6 Conclusions and future work

In this work, we have proposed an implementation of a lab session about Virtual Private Networks based on NETinVM. A fully guided lab text is delivered to students with the exercises to be carried out to install and configure a VPN within the NETinVM infrastructure. Different VPN configurations are deployed and analyzed, ranging from simple, point-to-point VPN with no encryption nor authentication launched from the command line, to a client/server mode VPN using a Public Key Infrastructure (PKI) with X.509 certificates. The procedure to install and configure OpenVPN in NETinVM is exactly the same as the one required in a real world scenario with physical computers and networks. This means that the NETinVM infrastructure is well suited to autonomous student work and self-learning activities. Although nested virtualization components are generally available, setting up the virtual infrastructure necessary to perform activities similar to the one presented in this paper would require a significant effort, even for people with the adequate computer skills.

Evaluation results show that the level of learning achieved by the students through this lab session seems to

Table 1 Evaluation of the learning achieved in each of the four lab sessions

	Possible values (lab session number)			
	Lab sess. 1	Lab sess. 2	Lab sess. 3	Lab sess. 4
Number of answers	5 (22%)	2 (9%)	7 (30%)	9 (39%)

Table 2 Perception of the lab sessions usefulness

	Possible values (lab session number)			
	Lab sess. 1	Lab sess. 2	Lab sess. 3	Lab sess. 4
Number of answers	1 (30%)	0 (0%)	6 (26%)	10 (43%)

Table 3 Evaluation of the activity provided by the students

	Marks obtained (in the range from 0 to 10)									
	0	2	4	5	6	7	8	9	10	Avg.
Number of answers	1	1	2	1	2	5	8	1	3	7.17

be higher than in the other lab sessions of the course, and they consider this lab useful for their professional development. Also, the average evaluation of the lab session is high. These results show that students at least do not consider that the tool used in the lab session is an impairment that prevents or reduce the potential learning, like a real network infrastructure often is. Therefore, we can conclude that the proposed lab session is a feasible way of introducing students in actual but complex computer network concepts like VPNs.

As a future work to be done, we plan to carry out the same lab in other courses the next term, extending the survey to more students for collecting a statistically significant number of answers which allow a complete human-computer study.

We think this experiment suggests several interesting topics of future work to be done. First of all, we plan to to carry out the same lab in other courses the next term, extending the survey to more students for collecting a statistically significant number of answers which allow a complete human-computer interaction study.

We also plan to test other complex network scenarios, such as other VPN services (eg: WireGuard), LAN segmentation with VLANs, network filtering with NFTABLES or load balancing using reverse proxies.

Although not directly related with networks, we also plan to tackle system administration scenarios, such has using containers, remote backup of KVM virtual machines, live migration of KVMs, network filesystems such as NFS or testing advanced features of BTRFS.

We are also performing security exercises based on the instructor's remote access to the students' NETinVM (with

express consent, of course). And lastly we are testing ways of hosting large numbers of NETinVM instances on a personal computer using design options similar to those used to accommodate a handful of nested KVM machines in a virtual machine with only 4 GB of RAM, such as NETinVM itself.

Appendix 1 Detailed VPN Lab session

Point-to-point VPN using clear text tunnel

Since the OpenVPN packages required for this lab are not installed by default in the KVM machines, we must first install them in all the machines involved in a VPN. We must enter as root in the console of the machine, and execute the commands `#apt-get update` and `#apt install openvpn`. Since this lab uses the NETinVM machines extc, intc, and inta, these commands should be executed in these three machines to install openvpn in all of them.

Once the required software is installed, the first step will consist in installing a point-to-point VPN using a clear text tunnel between the machines intc (acting as a client) and extc (acting as the server). We can directly follow the examples given in the OpenVPN reference book "Mastering OpenVPN" Crist and Keijser [24].

Setting up the VPN from the command line

In this case the setting of the VPN is done as a single command on each machine:

1. Start the server side in listening mode:
   ```
   root@extc: ~# openvpn --ifconfig
   10.200.0.1
   10.200.0.2 --dev tun
   ```
2. Start the client side:
   ```
   root@intc: ~# openvpn --ifconfig
   10.200.0.2
   ```

```
    10.200.0.1   --dev   tun   --remote
  extc.example.net
```

If everything has gone fine, the output on the client side should be something similar to the one shown in Fig. 3, which shows how the tun0 interface is opened and the connection is established.

Next, the following exercise (which we will denote as "Exercise B") using nc application is proposed: start Wireshark and capture the traffic in the "mirror-int" interface.[4] Since the VPN connection keeps busy console1 of both extc and intc,[5] we should login in console2 of these machines. We will login as root, and in console2 of extc we will type

```
root@extc: # nc -l -p 31000
```

In console2 of intc (client side) we will type first

```
root@intc: # nc 10.200.0.1 31000
```

Next, we will type any message we want to transmit through the tunnel, for example:

```
root@intc: # nc 10.200.0.1 31000 hi, from
the client side
```

Setting up the VPN from a configuration file

The next step in the lab session consist of introducing the student to the syntax of the main options that can be included in a configuration file, following the examples shown in Crist and Keijser [24]. The session guide should explain that each option specified in the command line with the syntax

```
--<option> <option arguments>
```

can also be specified in a configuration file using the syntax

```
<some option> <option-arguments>
```

that is, removing the two dashes before the option of the command line.

The invocation of the configuration file from the command line is done using the arguments

```
--config <conf_file_name>
```

when invoking openvpn.

As an exercise to practise, the students are asked to repeat the last exercise but now using configuration files instead of the command line in both server and client.

At this point, the session guide should introduce the daemon and log options, explaining that the implementation as a daemon prevents any output to directed to the console. Also, it is important to highlight the use of the ps

aux | grep concatenated commands in order to see whether the VPN is still being executed or not. As an illustrative example, Fig. 5 shows the configuration file which students must write on the client side.

Point-to-point VPN using pre-shared keys

The session guide explains that in this case the secret key should be generated and copied to both ends of the connection (using SCP, for example). The command

```
root@extc: # openvpn --genkey --secret
secret.key
```

is explained. Also it is explained that if OpenVPN starts with an augmented level of debugging (greater or equal than –verb 7) then the keys are printed when program is started.

At this point, an example using NETinVM is shown. In order to copy the generated key, the shared directory among all the KVM machines in NETinVM is used[6]: in extc, the command

```
root@extc: # cp secret.key./shared
```

while in intc we will copy the key to the working directory:

```
root@intc: #  cp./shared/secret.key  /
root
```

And we start the OpenVPN daemon in both ends. In the server the command is:

```
root@extc: #    openvpn   --ifconfig
10.200.0.1 10.200.0.2 --dev tun --secret
secret.key --verb 7
```

while in the customer is:

```
root@intc: #    openvpn   --ifconfig
10.200.0.2
  10.200.0.1 --dev tun --secret secret.key
  --remote extc.example.net
```

At this point, the session guide asks the students to capture a snapshot with the output generated by the command issued in the server side.

PKI deployment. Client/server VPN

The last part of the lab session is devoted to the client/server mode, where the server is a single OpenVPN process to which multiple clients can connect. Each authorized and authenticated client is assigned an IP address from an address pool managed by the server. Clients cannot communicate among them directly. Instead, all the traffic flows

---

[4] "mirror-int" is a virtual Ethernet interface in "base" that is connected to a mirror port preconfigured in the "int" switch to make it easy to capture and analyze traffic in the internal network.

[5] "consoleX" is a terminal directly attached to a virtual serial port in the virtual machine. This avoids that this interface generates network traffic, making it easier to analyze captured network traffic.

[6] This shared folder is implemented using a "virtualfs" filesystem. "virtualf" is a Linux kernel's standard component that lets guest Linux systems access to a directory in its host's filesystem. This access has the advantage (over SMB, NFS, or any other network filesystem) that it doesn't generate any network packets (simplifying the analysis of network traffic).

through the server (unless the "client-to-client" option is used and the appropriate rules are set in the firewall and/or routers).

In this part, the lab session will try to emulate the university network with NETinVM. The VPN server will be extc, one client will be installed in intc, and another client will be installed in inta, and from both machines we will access the http server through the VPN tunnel as if these machines had another IP addresses. Also, the need for establishing a PKI infrastructure is explained. In this session we use the easy-rsa package, which should be installed in extc. Thus, the whole process of installing the package and generating the following items is detailed in the session guide:

- Download the correct version from GitHub:
  ```
  root@extc: # wget -P
  /https://github.com/OpenVPN/easy-
  rsa/releases/download/v3.0.7/EasyRSA
  −3.0.7.tgz
  ```
- Extract the tarball and rename the directory:
  ```
  root@extc:  #  tar  xvf  EasyRSA
  −3.0.7.tgz
  root@extc: # mv  ~/EasyRSA −3.0.7/
  ~/easyrsa/
  ```
- In OpenVPN we need to create:

  - A private key and a X.509 certificate for the Certifying Authority (CA) who signs documents.
  - A private key and a X.509 signed certificate for the server.
  - A private key and a X.509 signed certificate *for each* client.
  - A Diffie-Hellman group for the server

The EasyRSA−3.0.7 package, installed in extc, allows the students to practise the whole process of generating a PKI for the server as well as two clients, that are installed in intc and inta. The initialization of the environment is started with the command

```
extc: ~/easyrsa#./easyrsa init-pki
```

This command creates a directory called "pki". At this point, the Diffie-Hellman module should be generated:

```
extc: ~/easyrsa#./easyrsa gen-dh
```

which generates a file called dh.pem in the pki/ directory.

Next, we should generate the public and private keys for the Certification Authority (CA). When using this command, we should use the "nopass" option to avoid the need of providing a password each time the private key is used. The exact command is

```
extc:   ~/easyrsa#./easyrsa   build-ca
nopass
```

which generates the public key (ca.crt file) and the private key (ca.key) of the CA. Next, we will create a private key for the server, and a X.509 certificate signing request (CSR) including its public key:

```
extc:   ~/easyrsa#./easyrsa   gen-req
servidor nopass
```

OpenVPN also needs the server certificate, digitally signed by the CA. In order to get it, the guide explains the required command:

```
extc:   ~/easyrsa#./easyrsa   sign-req
server servidor
```

This command leaves the signed certificate in the file /root/easyrsa/pki/issued/servidor.crt. The next step consists of creating the private key and the X.509 digital certificate for each client, and signing the latter with the CA created before with the commands

```
extc: /easyrsa#./easyrsa gen-req cli-
entX nopass
extc: /easyrsa#./easyrsa sign-req cli-
ent clientX
extc: /easyrsa#./easyrsa sign-req cli-
ent clientX
```

Finally, OpenVPN needs a symmetric key for TLS, which can be generated in the server with the command

```
extc:   ~/easyrsa# openvpn --genkey --
secret ta.key
```

At this point, the files required for the server (dh.pem, ca.crt, servidor.key, servidor.crt, and ta.key) should be copied to the directory /etc/openvpn/keys in the server (extc), and for each client (intc and inta machines) the required files (clientX.key, ca.crt, and ta.key) should also be copied (through the use of the "shared" directory) to the directory /etc/openvpn/keys of intc (X=1) and inta (X=2).

Once the PKI infrastructure has been set up, the guide explain the configuration file for the server, as shown in listing 1, and also for the clients, as shown in listing 2.

It must be noted that the files "client2.key" and "client2.crt" should be used in inta, but these files should be "client1.key" and "client1.crt" in intc.

In order to carry out the next exercise, it is required that Wireshark is installed on extc,

```
root@extc: ~# apt-get update
root@extc: ~# apt install wireshark
```

The client/server exercise proposed to the students in the session guide consists of starting the LXDE desktop in extc (the session guide references section "Taking advantage of KVM and libvirt" of NETinVM documentation Carlos Pérez and David Pérez [22]), and two instances of Wireshark Sharpe et al. [25]. The students should start one of the instances to capture the traffic traversing the "tun0" interface, while the other should capture the traffic traversing the "eth0" interface. Then, students should issue ping commands to the VPN server (ping -c 3 10.200.0.1)

from the intc console, and compare the frames captured by both interfaces, explaining the differences.

The final exercise in the lab session requires the activation of routing in the server (extc) with the command

```
root@extc:     ~#     sysctl     -w
net.ipv4.ip_forward=1
```

The exercise asks the student to write down the configuration file for a client/server VPN analogous to the one of the University of Valencia, where the server emulating vpn.uv.es is extc, and there are two clients emulating the laptop of both students at their homes: inta and intc. However, in this case the tunnel will use the range of IP addresses 10.5.0.48/28, allowing the use of the IP addresses 10.5.0.49−10.5.0.62. The PKI infrastructure should assign the client1 certificate to inta, which will establish the VPN in first place, and the client2 certificate to intc, which will establish the VPN in second place. The students are asked to execute the command

```
root@extc: ~ # tcpdump -i tun0 -n
```

and then execute the command

```
root@intc: ~ # ping -c 4 10.5.0.50
```

The students should show the configuration files in both server and clients, and they are asked to detail and explain the differences existing between the output of the ping in intc and the output of the tcpdump in extc. The output in both machines are shown in Fig. 8. This figure shows how the redirect messages are shown in both client and server, and the differences in the number of messages exchanged.

Declarations

Conflict of interest The authors declare no competing interests.

References

1. Sarkar, N.: Teaching computer networking fundamentals using practical laboratory exercises. Educ. IEEE Trans. **49**(2), 285–291 (2006)
2. Trabelsi, Z., Alketbi, L.: Using network packet generators and snort rules for teaching denial of service attacks. In: Proceeding of 18th ACM Conference on Innovation and Technology in Computer Science Education. ACM, New York, pp 285–290 (2013)
3. O'Grady, M.J.: Practical problem-based learning in computing education. Trans. Comput. Educ. **12**(3), 10:1-10:16 (2012). https://doi.org/10.1145/2275597.2275599
4. Carter, J.: (ed) ITiCSE '13: Proceedings of the 18th ACM Conference on Innovation and Technology in Computer Science Education. ACM, New York (2013)
5. Bhat, A.Z., Mir, A.W., Pandey, J., et al.: Nested virtualization, an implementation scenario for higher education institutions, solution, and performance issues. In: 2022 10th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO), pp 1–5 (2022), https://doi.org/10.1109/ICRITO56286.2022.9965019
6. Cogitative Software FZE: OMNeT++ Discrete Event Simulator. Accessed 14 May 2023 (2023), URL https://omnetpp.org/
7. Binkert, N., Beckmann, B., Black, G., et al.: The gem5 simulator. SIGARCH Comput. Archit. News **39**(2), 1–7 (2011). https://doi.org/10.1145/2024716.2024718
8. Cisco Networking Academy: Cisco Packet Tracer. Accessed 14 May 2023 (2023), URL https://www.netacad.com/courses/packet-tracer
9. Contributors, M.P.: Mininet: An Instant Virtual Network on your Laptop (or other PC). Accessed 24 May 2023 (2023), URL https://mininet.org/
10. Salah, K.: Harnessing the cloud for teaching cybersecurity. In: Proc. of the 45th ACM Technical Symposium on Computer Science Education. ACM, New York pp 529–534 (2014), https://doi.org/10.1145/2538862.2538880
11. Willems, C., Klingbeil, T., Radvilaviciusyz, L., et al.: A distributed virtual laboratory architecture for cybersecurity training. In: IEEE proceedings of the 6th International Conference on Internet Technology and Secured Transactions. IEEE, Los Alamitos, pp 408–415 (2011)
12. Abraham, S.: Virtual learning tools in cyber security education. In: 16th Annual NY State CyberSecurity Conference, pp. 408–415. IEEE, Los Alamitos (2013)
13. Xu, L., Huang, D., Tsai, W.T.: Cloud-based virtual laboratory for network security education. Educ. IEEE Trans. **57**(3), 145–150 (2014). https://doi.org/10.1109/TE.2013.2282285
14. Son CJoon; Irrechukwu, Fitzgibbons, P.: A comparison of virtual lab solutions for online cybersecurity education. Commun. Int. Inf. Manag. Assoc. **12**(4), 81–96 (2012)
15. Pérez, C., Orduña, J.M., Soriano, F.: A nested virtualization tool for information technology practical education. SpringerPlus (2016). https://doi.org/10.1186/s40064-016-2041-8
16. Riverbed Technology: Riverbed modeler. Accessed 14 May 2023 (2023), URL https://support.riverbed.com/content/support/software/steelcentral-npm/modeler-index.html
17. Boson, L.L.C.: NetSim network simulator. Accessed 24 May (2023), URL https://www.boson.com/netsim-cisco-network-simulator
18. SolarWinds Worldwide: GNS: The software that empowers network professionals. Accessed 14 May 2023 (2023), URL https://gns3.com/

19. Jason A. Donenfeld: Wireguard: fast, modern, secure VPN tunnel. Accessed 25 May 2023 (2023), URL https://www.wireguard.com/

20. Howtoforge Linux tutorials: How to install wireguard vpn on debian 11. Accessed 25 May 2023 (2023), URL https://www.howtoforge.com/how-to-install-wireguard-vpn-on-debian-11/

21. Pérez, C., Pérez, D.: Netinvm: A tool for teaching and learning about systems, networks and security. Accessed 10 Nov 2022 (2020), URL http://www.netinvm.org

22. Carlos Pérez, David Pérez.: Netinvm: A tool for teaching and learning about systems, networks and security. Accessed 15 Dec 2022 (2016), URL https://informatica.uv.es/~carlos/docencia/netinvm/netinvm.html#taking-advantage-of-kvm-and-libvirt-or-how-to-use-the-lxde-desktop-in-the-kvms

23. Centro Criptológico Nacional.: Guía de seguridad de las tic ccn-stic-836 - seguridad en redes privadas virtuales (vpn). (2022), URL https://www.ccn-cert.cni.es/

24. Crist, E., Keijser, J.: Mastering OpenVPN. Community experience distilled, Packt Publishing, (2015) URL https://books.google.es/books?id=5VUqjgEACAAJ

25. Sharpe, R., Warnicke, E., Lamping, U.: Wireshark user's guide. Accessed 15 Dec 2022 (2020), URL https://www.wireshark.org/docs/wsug_html_chunked/

26. Dehbozorgi, N., MacNeil, S., Maher, M.L., et al.: A comparison of lecture-based and active learning design patterns in cs education. In: 2018 IEEE Frontiers in Education Conference (FIE), pp 1–8 (2018), https://doi.org/10.1109/FIE.2018.8659339

27. Hornbaek, K.: Some whys and hows of experiments in human-computer interaction. Found. Trends Hum.-Comput. Interact. **5**(4), 299–373 (2013). https://doi.org/10.1561/1100000043

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.