Lecturer
Prof. Dr. Christian Tschudin (christian.tschudin@unibas.ch)

Tutors
Carlos Andrés Tejera (carlos.tejera@stud.unibas.ch)
Andreas Wassmer (andreas.wassmer@unibas.ch)

Uploaded on
Thu., 2 March 2022

Deadline
Wed., 16 March 2022
Upload on ADAM

Target/Idea

In a first exercise, some basic handling of the tools of the trait shall be trained. Starting with the handling of and working with virtual machines, followed by the working in a Linux and pure terminal environment, VMs should be a common tool and a matter of quick setup if needed. The presentation of 2 created VMs will be part of the assignment.
In case of questions and getting stuck, the tutors can also be contacted outside of the exercise lessons via e-mail anytime.

VM - Quickly Explained

A Virtual-Machine (VM) is a simulated computer system with its own processing unit (typically a CPU) and its own memory. Besides the flexible simulation of multiple elements (like memory-controllers), the CPU architecture remains often the same as the one of the original running host system. For the graphics, commonly a VGA-passthrough is used and in some cases a generic GPU front with passthrough. Yet, there are even more flexible simulation programs that also emulate (and thus translate) different architectures or specific hardware, and/or allow CPU/GPU-passthrough for specific functionality (like CUDA).
VM in general can have multiple purposes, often used as emulation environment of a specific hardware, portable/sharable work-environment, secure testing-environment or as part of the virtualisation to split and assign resources of server-systems (Cloud-Services like "systems as a service").

VM Setup

For the VM setup, we recommend using **Oracle Virtual Box** (https://www.virtualbox.org/) for everyone using a x86 system as an intuitive simple interface. For user of Microsoft and Apple ARM devices (like devices with M1 Apple Metal chip), we recommend **QEMU** (https://www.qemu.org/) or **UTM** (https://mac.getutm.app/). With the latter one a GUI app using QEmu internally and it can be used to run x86 VMs on ARM devices and it generally simplifies the usage of QEmu. QEmu itself is a terminal-based VM Emulator that can also be used to emulate any other kind of device (like a RaspberryPi). Hence, this CLI-based tool is especially interesting for developers despite having more complex handling compared to common VMs.

For this purpose here, we recommend the following Linux distributions :

- Alpine (https://alpinelinux.org/downloads/) for terminal-based micro-Linux
- Debian/Fedora (https://getfedora.org/, https://www.debian.org/distrib/) for graphic-interface-OS

Alternatively, other distributions (like Fedora) work as well, or Mini-Distros like AntiX, Rufus, Etcher and such alike (partially with graphic-interfaces).
Note : To remove any overlap with other courses using VM, we ask you not to install Ubuntu, otherwise this would be an equivalent alternative.

For the VM configuration, we recommend **2 GB RAM with 10 GB of hard disk limit for VMs with graphic interface** and $<$ **1 GB RAM with 5 GB of hard disk limit for terminal-based VMs**. Note that using too little hard disk can end in bottlenecks, while more hard disk creates high data-blocks. Analogously for RAM, using too little can hinder the usefulness of the VM, while more RAM clutters up the host's RAM. The guest-OS is then installed to this former hard disk, which shows up as editable memory file on the actual executing host computer. As such it is only virtual and we henceforth call it virtual hard disk. For the installation source, it is mounted/connected via a virtual device (like a virtual DVD-player) with a loaded disk image (iso, dmg, ...). Analoguely, the virtual hard disks can be mounted and unmounted like images, yet use other writable formats like "qcow2" (QEmu), "vdi" (VirtualDisk), "vhd" and many more.

You can find detailed instructions for VirtualBox on the web

https://www.howtoforge.com/tutorial/debian-minimal-server/.

For QEMU, you find some detailed instructions here :

https://www.howtoforge.com/tutorial/debian-minimal-server/
or
https://pqsec.org/2021/01/05/intall-debian-ubuntu-in-qemu.html.

For UTM, there is a Guide to be found on the website itself :

https://getutm.app/guide/

Additional knowledge useful for follow-up exercises :

- **File exchange** : Here, we recommend a shared folder and/or bi-directional Drag-and-Drop functionality with graphic-interface-guest-OS. Other methods for file exchange will be discussed in future exercises.

- **Duplication** of VMs and running them concurrently. Hint: Here, conflicts can arise due to identical identifiers or absolute linking!

Exercise 1 - VMs (4 Points)

You will setup and create your own two VMs, one with a GUI (Desktop environment) and another one without, and present both in the exercise sessions. In each one, a given set of commands will be executed as proof of concept. Please, also mention any problems or difficulty you had achieving this task combined with which OS and hardware you are using.

Note : To avoid any overlap with other courses using VMs, you should be able to roughly justify your choices and steps taken to creating the VMs. Furthermore, we ask you not to use Ubuntu as Guest-OS here.

a) Linux with graphic-interface

The larger image with a running web-browser (web-access is not necessary).

b) Linux without graphic-interface

The smaller image with a terminal-based editor (VIM, nano,...)

Exercise 2 - Experience with VMs (2 Points)

Answer the questions regarding the usage of VMs.

(a) Do both of your VMs have internet access? What steps did you have to take or did you had to install any special software/driver during the installation of the VM app?

(b) Duplicate one of your VMs and run them concurrently. Depending on the VM-app and Hypervisor you have chosen, what files did you have to copy? What actions did you have to take to let them run concurrently? Write down all steps.

Exercise 3 - Linux (2 Points)

Answer the questions regarding the usage of Linux distributions.

(a) What means "sudo" , "sudoer-list", "su" and for what is it used?

(b) Some software installations require administrative access, some do not. Why and what is the consequence?

(c) What is the purpose of a package manager? Who provides the content / lists?

(d) Which command would you use to create a file in the terminal?

(e) What does the command "cat"? How can you figure that out?

(f) What is "piping" and what is its purpose? Give one example of how you could use that.

(g) Some programs (e.g. scripts) need more time to be executed and block the terminal session until it is done. How could you avoid that?

(h) How to determine in a terminal the file type of a file without file extension (like ".pdf", ".txt")? (Hint: file header)

(i) What is the purpose of the "touch" command?

(j) Is there a build in manual / info-text to commands? How can be access it? (Hint : There is an external call and a command-internal "option".)