



**UNIVERSITAT POLITÈCNICA DE CATALUNYA  
BARCELONATECH**

---

**Escola Tècnica Superior d'Enginyeria  
de Telecomunicació de Barcelona**

# **TESTBED PLATFORM FOR CYBERSECURITY USE CASES**

**A Master's Thesis**

**Submitted to the Faculty of the  
Escola Tècnica d'Enginyeria de Telecomunicació de  
Barcelona**

**Universitat Politècnica de Catalunya**

**by**

**Enric Ruhi Velasco**

**In partial fulfilment  
of the requirements for the degree of  
MASTER IN ADVANCED TELECOMMUNICATIONS  
TECHNOLOGIES (MATT)**

**Advisor: Josep Rafael Peguerols Valles**

**Barcelona, May 2020**





UNIVERSITAT POLITÈCNICA DE CATALUNYA  
BARCELONATECH

Escola Tècnica Superior d'Enginyeria  
de Telecomunicació de Barcelona



**Title of the thesis:** Testbed platform for Cybersecurity Use cases

**Author:** Enric Ruhi Velasco

**Advisor:** Josep Rafael Peguerols Valles

## **Abstract**

In the cybersecurity world, having not only theoretical but also practical experiences is crucial. Cybersecurity Use Cases (UCASES) is a subject from the master's degree in advanced telecommunication technologies (MATT) that provides the opportunity to see penetration testing and cybersecurity from a practical standpoint.

This thesis' objective is to provide a stable and up to date testbed for UCASES, composed of multiple virtual machines, vulnerable and non-vulnerable. In addition, it also contains a practical beginner's guide based on the guide provided by the lecturer of the subject, Josep Rafael Peguerols Valles, and a virtual machine with all tools the students will need to follow the guide and the subject UCASES.



UNIVERSITAT POLITÈCNICA DE CATALUNYA  
BARCELONATECH

Escola Tècnica Superior d'Enginyeria  
de Telecomunicació de Barcelona



*Aquest treball està dedicat als meus pares i la meva germana, els quals han estat al meu costat en tot moment, durant tot el màster, donant-me suport i recolzant les meves decisions.*

*Moltes Gracies!*



UNIVERSITAT POLITÈCNICA DE CATALUNYA  
BARCELONATECH

Escola Tècnica Superior d'Enginyeria  
de Telecomunicació de Barcelona



## **Acknowledgements**

First, I would like to thank my supervisor Josep Rafael Peguerols Valles. He had infinite patience with me and did far more for me than his role as supervisor required. Without him and his dedication and patience this project would have never been done.



## **Revision history and approval record**

Revision	Date	Purpose
0	09/04/2020	Document creation
1	07/05/2020	Document revision
2	11/05/2020	Document correction
3	15/05/2020	Document revision
4	18/05/2020	Document correction
5	22/05/2020	Document final revision

Written by:		Reviewed and approved by:	
Date	09/04/2020	Date	22/05/2020
Name	Enric Ruhi Velasco	Name	Josep Rafael Peguerols Valles
Position	Project Author	Position	Project Supervisor



## **Table of contents**

Abstract .....	1
Acknowledgements .....	3
Revision history and approval record .....	4
Table of contents .....	5
List of Figures .....	7
List of Tables .....	8
1. Introduction.....	9
1.1. Statement of the problem .....	9
1.2. Motivation .....	9
1.3. Solution .....	10
1.4. Objectives .....	10
1.5. Work plan, milestones and task diagram .....	10
2. Technology used or applied in this thesis .....	14
2.1. Virtual labs.....	14
2.1.1. HackTheBox & TryHackMe .....	14
2.2. Tools .....	15
2.2.1. Reconnaissance.....	15
2.2.2. Exploiting.....	17
3. Project development.....	18
3.1. Virtualization platform .....	18
3.1.1. Why this three requirements?.....	18
3.1.2. Choosing the platform .....	19
3.2. Updating UCASES .....	19
3.2.1. Tools .....	19
3.3. Virtual Machines.....	20
3.3.1. Attacker .....	20
3.3.2. Victims.....	21
4. Results .....	22
4.1. Scenario .....	22
4.2. Reconnaissance.....	23



4.2.1.	Recon-ng.....	23
4.2.2.	Nmap .....	25
4.2.3.	Scapy .....	26
4.2.4.	NSE.....	27
4.2.5.	Nessus .....	28
4.2.6.	Netcat.....	30
4.2.7.	Nikto.....	30
4.3.	Exploiting.....	30
4.3.1.	Metasploit.....	30
4.3.2.	Empire.....	32
5.	Budget.....	33
6.	Conclusions and future development.....	34
6.1.	Difficulties.....	34
6.2.	Testing .....	34
6.3.	Future developments.....	35
	Bibliography.....	36
	Glossary .....	38



## **List of Figures**

<i>Figure 1: Logo TryHackMe &amp; Hack the Box .....</i>	<i>15</i>
<i>Figure 2: Recon-ng ASCII when starting the framework .....</i>	<i>16</i>
<i>Figure 3: Scapy ASCII when starting the framework .....</i>	<i>16</i>
<i>Figure 4: Nessus logo .....</i>	<i>16</i>
<i>Figure 5: Metasploit logo .....</i>	<i>17</i>
<i>Figure 6: UCASES testbed network .....</i>	<i>23</i>
<i>Figure 7: Setting a NAMESERVER .....</i>	<i>24</i>
<i>Figure 8: Recon-ng database showing reverse-resolve findings .....</i>	<i>24</i>
<i>Figure 9: File of AV domains used by cache-snoop .....</i>	<i>25</i>
<i>Figure 10: Full TCP SYN scan output .....</i>	<i>26</i>
<i>Figure 11: OS fingerprinting on the vulnerable Windows 2008 machine. ....</i>	<i>26</i>
<i>Figure 12: TCPdump output of challenge #2 .....</i>	<i>27</i>
<i>Figure 13: Missing reset [R] response from target's ports 10 &amp; 11 that are "filtered" .....</i>	<i>27</i>
<i>Figure 14: Content of the "script.db" file .....</i>	<i>28</i>
<i>Figure 15: sshv1.nse finds SSH in port 23 .....</i>	<i>28</i>
<i>Figure 16: Nessus scan of UCASES target machines .....</i>	<i>29</i>
<i>Figure 17: Populated Metasploit services table .....</i>	<i>31</i>
<i>Figure 18: Hashdump of Windows credentials .....</i>	<i>31</i>
<i>Figure 19: Real time screen of Windows machine .....</i>	<i>32</i>



UNIVERSITAT POLITÈCNICA DE CATALUNYA  
BARCELONATECH

Escola Tècnica Superior d'Enginyeria  
de Telecomunicació de Barcelona



## **List of Tables**

<i>Table 1: Budget</i> .....	33
------------------------------	----



## **1. Introduction**

### **1.1. Statement of the problem**

The Universitat Politècnica de Catalunya (UPC) started, last year (2019-20); a new master called Master in Advanced Telecommunication Technologies (MATT) this master has multiple sub-tracks, one of them being cybersecurity.

UCASES is a recommended subject for the cybersecurity track in this master. Its objective is to introduce the students to the different penetration testing tools and its uses from a practical point of view. To summarize, the students have an “attacker” machine, with penetration testing tools, and through the semester, the students followed an extensive guide that explained how to use those tools. These tools, were used against “victim” machines that were installed locally on the student’s PC in a virtual environment.

After the first iteration of this subject, with feedback from the students, the lecturer spotted some problems. The ones that were more impactful for the students were the following.

First, the students had to put together their own virtual environment and VMs, that caused problems for some parts of the subject, since the OS the students used as “victim” machines were secure enough that some parts of the practices were straight up cut off by the victim OS security. That was solved at the time by tinkering with the OS security settings, but was inconsistent for some students, since not every student had the same version of the OS, and it took time off the lab work, and made the scenario feel too unrealistic.

Second, some of the labs were prepared for older versions of the tools, so some steps of the lab guide were outdated and were not working for the versions of the tools the students had installed in their “attacker” machine (LTS / latest). This made the students take too much time on a specific part of the subject. Either going back to a previous version of the tool that is not the latest/LTS or going through the documentation of the tool.

Finally, some of the labs, could be done on a random environment legally, but if they are done without any care, that might have unexpected outcomes for the target. For example, a scan of a target environment done by multiple students at the same time, even if unlikely, might cause DoS for the target environment. This could be a temptation if the students itself needs to, for example, open ports before doing a port scan to have some meaningful results and not all ports closed/filtered, again making the practices feel too unrealistic.

### **1.2. Motivation**

The previously stated problems might not seem important at first glance, but as a former student of the subject, I know all these problems pile up. We need to take into account that UCASES is an introduction to penetration testing tools, and most of the students will not have previous experience with them at all. Thus, having a consistent and stable set



up for the students is important since it will be the first experience with penetration testing for some of them. In addition, this subject contains a lot of information and the time constrain is a factor. Having problems with the testing environment or the guide can put a hindrance on the learning experience of the students. Finally, the cybersecurity world is in constant change, using old version of the tools does not set a good example neither a good experience for the students.

### **1.3. Solution**

The solution implemented in this project is a penetration testing testbed for the subject Cybersecurity Use cases (UCASES). The testbed must be easily installable by the students, maintainable and updateable for the lecturer of the subject.

The testbed must contain the necessary “victim” VMs to be able to perform the lab exercises for the subject.

In addition, we will make a guide explaining the basics of the tools, how to use them (theory) and where and how to test them in practice (e.g. URLs, VMs). Moreover, it will contain a small installation guide of the testbed and solutions for possible problems the students might encounter.

Furthermore, an attacker VM will be provided, with all the necessary tools for the students to do the practices.

This project is not a continuation of any other project or thesis. However, we used the first iteration of the UCASES subject as a starting point.

### **1.4. Objectives**

The main objectives of this project can be summarized as follows:

- Create a stable testbed for the UCASES subject.
- Provide an “attacker” VM with all the necessary tools to carry out the subject.
- Write an updated lab guide on how to perform all the practices with the updated tools.

### **1.5. Work plan, milestones and task diagram**

The project was divided in five main work packages (WP) with their corresponding internal tasks.

The first package consisted in obtaining the “attacker” OS that will be used for all the UCASES labs and installing / updating the tools to their latest / LTS if necessary.

The next two packages are the main part of this project and they encompass the updating part of the project. From the updated tools from the “attacker” OS, create the new guide and set up the necessary VMs to execute the lab exercises.

The fourth package consisted in doing tests and double-checking of the work done on WP: P2 and WP: P3.

Finally, the fifth task consisted in writing the necessary documentation for the UPC.



Testbed platform for Cybersecurity Use cases	<b>WP ref:</b> P1
<b>Major constituent:</b> Kali Linux SO preparation	
<b>Short Description:</b> Prepare an up to date kali Linux VM to test and choose the tools that will be used for the project	<b>Start date:</b> 10/01/2020 <b>End date:</b> 12/01/2020
<b>Internal task T1:</b> Download the latest LTS Kali Linux version ISO for Virtual Box. <b>Internal task T2:</b> Update and upgrade the tools. <b>Internal task T3:</b> Export updated state as a VM for the students.	

Testbed platform for Cybersecurity Use cases	<b>WP ref:</b> P2
<b>Major constituent:</b> Lab update	
<b>Short Description:</b> Using the material provided on the first iteration of UCASES as reference, update practices and environments.	<b>Start date:</b> 12/01/2020 <b>End date:</b> 23/03/2020
<b>Internal task T1:</b> Recon-ng. <b>Internal task T2:</b> NMAP. <b>Internal task T3:</b> Scapy & TCP dump. <b>Internal task T4:</b> NSE. <b>Internal task T5:</b> Nessus. <b>Internal task T6:</b> Netcat. <b>Internal task T7:</b> Set up VMs and its network to be able to do the reconnaissance lab. <b>Internal task T8:</b> Msfconsole. <b>Internal task T9:</b> Meterpreter. <b>Internal task T10:</b> Metasploit. <b>Internal task T11:</b> Empire. <b>Internal task T12:</b> Set up VMs and its network to be able to do the reconnaissance lab.	



Testbed platform for Cybersecurity Use cases	<b>WP ref:</b> P3
<b>Major constituent:</b> UCASES Guide	
<b>Short Description:</b> Write the guide for the students to be able to start using and practicing with the tools from the work package <b>P2</b>	<b>Start date:</b> 20/01/2020 <b>End date:</b> 29/03/2020
<b>Internal task T1:</b> Write basic explanation about the tools. <b>Internal task T2:</b> Introduction guide for the tools. <b>Internal task T3:</b> How to test and start playing with the tools.	

Testbed platform for Cybersecurity Use cases	<b>WP ref:</b> P4
<b>Major constituent:</b> Global Testing	
<b>Short Description:</b> Test work package <b>P3</b> guides to make sure they make sense and keep working as we update or change the OS.	<b>Start date:</b> 20/01/2020 <b>End date:</b> 29/03/2020
<b>Internal task T1:</b> Quick overview test of previous working functionalities. <b>Internal task T2:</b> Full extensive test at the end of <b>P3</b> .	

Testbed platform for Cybersecurity Use cases	<b>WP ref:</b> P5
<b>Major constituent:</b> Documentation	
<b>Short Description:</b> Write this document	<b>Start date:</b> 13/01/2020 <b>End date:</b> 01/04/2020
<b>Internal task T1:</b> Write the TFM document to present to the UPC and the tribunal. <b>Internal task T2:</b> Other documentation.	



WP#	Task#	Short Title	Milestone / Deliverable	Date (Week)
P2	T3	Kali Linux ready	Attacker OS that will be used for the TFM is available.	2
P3	T7	Reconnaissance lab	Finished the reconnaissance "chapter" update (VMs & tools)	6
P4	T12	Exploiting lab	Finished the exploiting "chapter" update (VMs & tools)	11
P5	T3	UCASES Guide	Finished writing UCASES guide	12
P6	T2	Testing	Final testing	12
P5	T2	TFM Documentation	End of TFM	13



## **2. Technology used or applied in this thesis**

The goal for this chapter is to present different options and already existing products, available at the time, which would solve the problem tackled in this project. This chapter will present them, see what characteristics they have and see why we discarded them and decided to build our own, custom, testbed. In addition, we will introduce the tools used in the guide, and therefore in UCASES. Note that, in this chapter, we will not see what we do with those tools in the guide / UCASES. That information can be found in chapter 4.

### **2.1. Virtual labs**

First, mention that we discarded any paid environment right away. Many of these penetration-testing environments offer a vulnerable environment to do reconnaissance on, and it contains vulnerable virtual machines to infiltrate and exploit. Some of them cover the contents of UCASES; offer a guide, support and a certificate if you complete them. However, the fact that there is a need for a paid subscription, per user in most cases, makes these options (e.g. pentesterlab [1], offensive security proving grounds [2]) not viable to solve the problem at hand.

On the other hand, there are some free and interesting options to choose from, but since there are a lot of options, we will look at the most promising ones.

#### **2.1.1. HackTheBox & TryHackMe**

Both of these environments, HackTheBox [3] and TryHackMe [4] (See logos in *Figure 1*), have both, a paid and a free version. Even if the free version has limitations is still a good penetration environment for novices. It has systematic exercises, with hints to solve them and a form to put the answer to make sure it is correct. Moreover, it has a discussion forum and discord if the user has any problems with the exercise. On top of that, there are write-ups available with the different solutions provided by other users.

Even if these are great options for novices to start, and have a lot of good material, both of them had the same problems.

First, both of them are missing some part of the UCASES course. Even if both of these sites cover far more than the subject itself, most of it is advanced stuff out of the scope of the subject. UCASES is an introductory subject and one of its objectives is to introduce the students to many tools without going too deep into any of them.

Second, and most important, the lecturer would not have control over anything inside those environments. If the subject UCASES uses a third party, it needs to be constant. Most of this environments change from one year to another, and this can be a major problem for the responsible, since some part of the subject might break from one semester to another without notice, or even mid semester.



Figure 1: Logo TryHackMe & Hack the Box

## 2.2. Tools

The tools used in UCASES can be separated in the two following main groups:

- Reconnaissance
- Exploiting

First, we will see the reconnaissance tools. These tools are used, as his name indicates, for scouting and gathering information about a target usually using information publically available on the internet, commonly known as Open-source intelligence or OSINT. It is important to note, that even if no laws specifically prohibits the actions done with this tools, such as port scans, doing reconnaissance without permission of the target can cause trouble and even carry a lawsuit.

Second, we will see exploiting tools. With these tools, the objective is usually to gain control over a target machine and actively manipulate it to achieve some goal. Unlike reconnaissance, the use of these tools on any third party machine without written permission is strictly illegal and is what most people consider “hacking”.

### 2.2.1. Reconnaissance

The reconnaissance tools used in UCASES are the following:

- **Recon-ng:** *“Web Reconnaissance framework written in Python. Complete with independent modules, database interaction, built in convenience functions, interactive help, and command completion, Recon-ng provides a powerful environment in which open source web-based reconnaissance can be conducted quickly and thoroughly.” [5].*

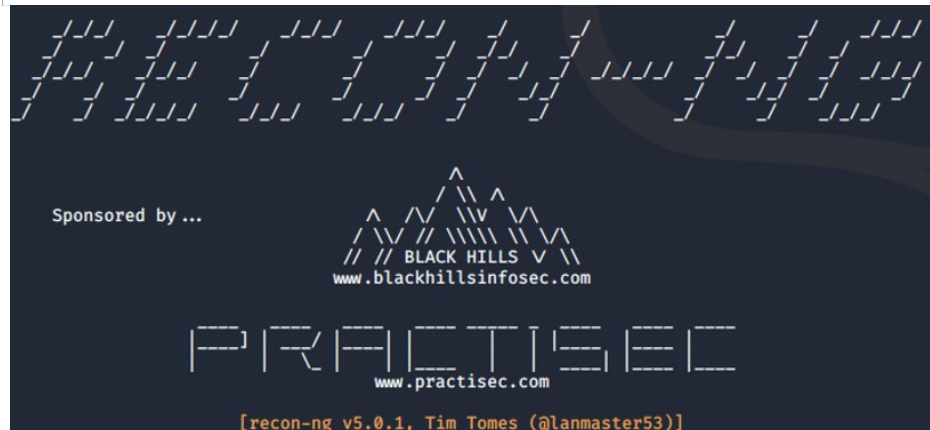


Figure 2: Recon-ng ASCII when starting the framework

- **Nmap:** “Nmap (“Network Mapper”) is a free and open source utility for network discovery and security auditing.” [6]
- **NSE:** “The Nmap Scripting Engine (NSE) is one of Nmap's most powerful and flexible features. It allows users to write (and share) simple scripts to automate a wide variety of networking tasks.” [7]
- **Scapy:** “Scapy is a Python program that enables the user to send, sniff and dissect and forge network packets. This capability allows construction of tools that can probe, scan or attack networks.” [8]

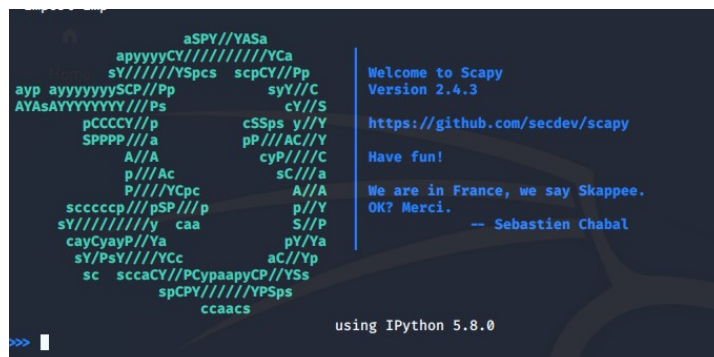


Figure 3: Scapy ASCII when starting the framework

- **Nessus:** “network vulnerability scanner that uses the Common Vulnerabilities and Exposures architecture for easy cross-linking between compliant security tools.” [9]



Figure 4: Nessus logo



- **Netcat:** “Computer networking utility for reading from and writing to network connections using TCP or UDP.” [10]
- **Nikto:** “Nikto is a web server assessment tool. It is designed to find various default and insecure files, configurations and programs on any type of web server.” [11]

### 2.2.2. Exploiting

The exploiting tools used in UCASES are the following:

- **Metasploit:** “Penetration testing platform that enables you to find, exploit, and validate vulnerabilities. It provides the infrastructure, content, and tools to perform penetration tests and extensive security auditing” [12]



Figure 5: Metasploit logo

- **Empire:** “Empire is a pure PowerShell post-exploitation agent built on cryptologically-secure communications and a flexible architecture.” [13]



### 3. Project development

In this section it is covered the process and the decision-making made during the development of the testbed, as well as other minor details that might be of interest and utility. This project is divided in three main blocks:

- **Attacker machine:** OS for the virtual machine that the students will use to perform the labs.
- **Victim machines:** Multiple OS, vulnerable and not, that will constitute the testbed and the ones the students will target during the labs.
- **UCASES guide / tool upgrade:** Development of an extensive guide for the subject, with the latest versions of the tools used, and the necessary information to start using them.

Moreover, a question that rose pretty early was “What virtualization platform will we use?” this question, even if is not one of the main blocks per se, is an impactful decision needed to be made almost at the start of the project.

#### 3.1. Virtualization platform

This was the first major decision taken for this project. Selecting a friendly, free and active virtualization platform was important. One of our objectives was to make the testbed easy to use & install for the students, and easy to use or modify and update for anyone. For this reason, we kept in mind that the testbed we developed in this project checked the following requirements.

- **Easy to use:** User friendly, we want a virtualization platform that has almost no learning curve and that is intuitive to manage.
- **Free & active:** As everything on this project, the software must be free to prevent extra cost for the students or the university. It also needs to be active since an active community and dev team is usually a signal of stability for the software.
- **Should be able to “evolve”:** We kept in mind that this testbed should not be a final product, others should be able to modify it and change it without major issues.

##### 3.1.1. Why this three requirements?

The first requirement was straightforward; we did not want the students to spend half a month, or more, of a four-month course learning how to use a virtualization platform, or anyone who would want to expand or update the testbed to find a stiff learning curve only to understand the virtualization software.

Next, as mentioned previously, if a software has an active community and development team, usually means long-term stability. We did not want to do all the set up for the testbed and that within two years that setup (VM networking, ISO files, etc.), see that any of it was no longer supported with all the problems for the lecturer of UCASES, or anyone who would like to update the platform, that the deprecation of the software might carry.



Moreover, is easier to find help and to solve any problem a student might encounter, that we did not expect or foresee, and therefore, the solution would not be available in the guide of this project.

In addition, as we said before, the students should not pay anything to study this subject (aside from its enrolment fee), nor the university should incur in extra expenses.

Furthermore, we did not even consider using “non-official/illegal” source to obtain proprietary software, and make the students look for them.

Finally, the cybersecurity world evolves fast, tools and software can change drastically within a year so, that is the reason for the point “Should be able to evolve”. Any one should be able to take the testbed, guide and the concepts used for this project, and be able to add or change anything to keep it up to date or expanding it. Adding a VM or changing a part of the guide to fit a change on a tool/concept or even adding a new tool, should not be an issue to anyone. Change in this world is inevitable and constant so one of this project’s objectives was to make the change easy.

### 3.1.2. Choosing the platform

Once we had the requirements for the virtualization platform, we just needed to look for the most used and popular virtualization platforms, since we wanted an active community and support.

On searching, we found two main candidates as the “most popular”: VMware, Virtual Box. There were others (e.g. Hyper-V [14] (Windows only), Parallels [15] (Mac)) but the OS limitations, the usability (Not friendly for new users) or that they were simply not free discarded them for our project.

Is important to note, that the option of using Docker [16] containers, with Docker instead of VMs was considered. It was tempting to go deep inside this technology and it looked like a promising option for creating a testbed with containers instead of virtual machines, but it clashes with one of the “requirements” of this project. It would not be easy to use, at least not at first; Docker has a learning curve with it, which could be a problem for the students and for anyone trying to expand the project. Even if Docker is a powerful and, once you have some experience with it, easy to use tool. It takes some starting learning curve that would take valuable time from the actual subject UCASES.

This left the two major products at the time, Oracle’s VirtualBox [17] and VMware [18]. Since this project was not about a benchmarking of virtualization products and VirtualBox and VMware Workstation (free) are similar, we opted for the 100% free and open source tool VirtualBox.

## 3.2. Updating UCASES

### 3.2.1. Tools

To be able to decide what VMs we would use for our testbed, first, we needed to decide what tools we wanted to use on UCASES. This was similar to the first iteration done of the subject, but a check was needed to see if any of the tools have been deprecated. The tools will be divided in two main groups, reconnaissance and exploiting.



The reconnaissance part will not affect the VMs themselves, but their ports and the network they are in to.

The reconnaissance tools that UCASES will dip on are the following: Recon-ng, Nmap, Scapy, NSE, Nessus, Netcat and Nikto.

The exploiting actively will interact with the VM and try gain control of it and use its features such as shell, webcam, etc.

The exploiting tools UCASES features are two: Metasploit and Empire.

The process of updating the tools was straightforward. First, we updated all the tools to the latest/LTS version, making sure all the tools were not deprecated or had any major issues. Next, we read the changelogs of the updates and the latest documentation, when available. After that, we proceeded to redo the entire UCASES subject (labs) with the new tools and taking notes of the changes. After that, we did a third iteration of the labs, knowing the changes beforehand. On this iteration, we wrote the guide, trying to think out of the box to try to spot possible problems the students might face (e.g. tool not on the PATH) and giving solutions to them. Finally, once the guide was done, a final iteration was done for each part and one globally, to make sure there was nothing amiss.

### **3.3. Virtual Machines**

#### **3.3.1. Attacker**

To be able to do the UCASES subject the students need a machine with, at least, all the tools they would use during the course. Even if all the tools are available for almost any Linux distribution, we choose Kali Linux as the attacker machine. We will now look at the reasons that lead to that decision.

Kali Linux [19] has been, and still is, the penetration testing OS by default for most of the community and there are reasons for that.

All the tools the average penetrating tester might need are already pre-installed. On the past Kali might have been a harsh OS for people used to Windows or MAC since the GUI was not a focus for its developers nor was a priority for the users of the OS. This has changed drastically the last years. The GUI of Kali Linux has improved greatly, the tools are sorted by utility and since the last versions, and it has an “undercover” mode that makes it similar to windows 10. That might be useful for some users so the change is not that hard, and it is a nice touch since Kali is heavily related to black hat hacking, due to people misinformation, so this makes it easier to use it without problems.

One other option was to use Ubuntu 18.04 [20] LTS “Bionic Beaver”, the latest LTS. This would have been the option if Kali did not update the GUI on the latest versions; there was no major benefit on using this OS over Kali. However, there is a disadvantage; unlike Kali Linux, the tools are not pre-installed on Ubuntu. Manually installing all the tools we need for this machine would be a time waste without any reward for doing it.

In addition, we considered other penetration testing distributions similar to Kali, such as BackBox [21], Parrot Security OS [22], BlackArch [23] and more. The ones that stood out were BlackBox and Parrot Security.



We choose Kali over BlackBox for multiple reasons. First, even if they are quite similar, Kali has a bigger community and a big Bug Bounty program [24], BlackBox has none, which makes Kali a better option on the long run since both of those things note a long-term software/product.

For Parrot Security OS what made us discard it was that it is relatively new. Since we said earlier that the cybersecurity world is in constant change and there are always new tools, this might seem like a poor or hypocritical decision. Why would we update to later version of tools and then discard a new OS?

The reason for this is that the tools we use are stable and have been used for a long time; they are not new, just updated. On the other hand, brand new products on cybersecurity come and go as fast as the field evolves. This means that a new OS might disappear if not successful or something happens. We are not saying that Parrot OS is bad or is going to fail, is just that Kali is a safer option with almost no drawbacks.

In conclusion, even after a broad search for a better option of the classic Kali Linux, no OS had any major benefit to justify the security and stability that Kali provides. To summarize, Kali Linux was the better OS for this project.

### 3.3.2. Victims

The “victim” or target machines are the VMs that will form the testbed itself, these machines will be used as test subject for the tools for the length of the course. We need two kinds of VM, one group that is vulnerable and exploitable, and another that is relatively secure so the students can see the difference from one to the other on some parts of UCASES.

Almost all of the modern or up to date operating systems already have a good AV (e.g. Windows defender) or defence mechanisms. Therefore, any modern OS such as Windows 10 will do the trick for the secure operating systems in our environment. Since it will be used for introduction to penetration testing, none of the attacks or exploits used should be able to affect them, without intentionally disabling the security that is.

For this testbed, we will be using Windows 10 as a secure machine since it is the most commonly used OS and can be a good example that, even if the scenario we provide in the tested makes it look easy to scan the target environment or take control of a machine. In reality, it is not that easy at all. In fact, most modern operating systems will simply not let you even download a file with a payload in it, even less execute it.

On the other hand, our vulnerable machines will be two a windows 2008 and an Ubuntu 14.04. To be more precise, we will use the Metasploitable 3 [25] VMs. As said earlier, UCASES introduces many tools but it does not dives deep in any one. We choose this machine because it brings the opportunity to explore more than what the subject covers for metasploit, since these machines have been built to use metasploit on. There are free guides and videos available online using metasploit to break into these machines, which makes it a good option for the students to expand the skills and knowledge obtained on UCASES.



## 4. Results

In this section, we will do an overview of the final version of the guide. We will describe the scenario from the testbed, explain how we used all the tools mentioned in previous chapters, and see what specific parts of the tools we used, since most, if not all of them, are extensive and deep enough to have a subject for themselves. As we did for other chapters, we will separate the tools in two groups: reconnaissance and exploiting. Note that this is an overview of what is done in UCASES, for a more detailed explanation on how UCASES uses the tools and for what purposes, please look at the full guide provided at the annexes.

### 4.1. Scenario

The final scenario of the testbed is composed of two vulnerable machines (Windows 2008, Ubuntu 14.04); one secure machine (Windows 10) and an attacker machine (Kali Linux). All of these machines will be run inside a virtualization platform, in our case, VirtualBox and they will be group together in a NAT network. Using the NAT network option for networking on the virtualization platform will make it possible that the machines can see each other and have access to internet without port forwarding via a “fake” internal network that connects all the machines together. This is important since we need visibility between the attacker and the target machines in most of the labs, and access to the internet in some small parts of some practices. We can see a diagram of what the scenario looks like in the *Figure 6*. Is important to note that with this configuration, the machines will have access to each other, the host and the internet, but not the host nor LAN will have access to them without a setup of port forwarding. This is not a problem for this project, but it is important to be aware of it.

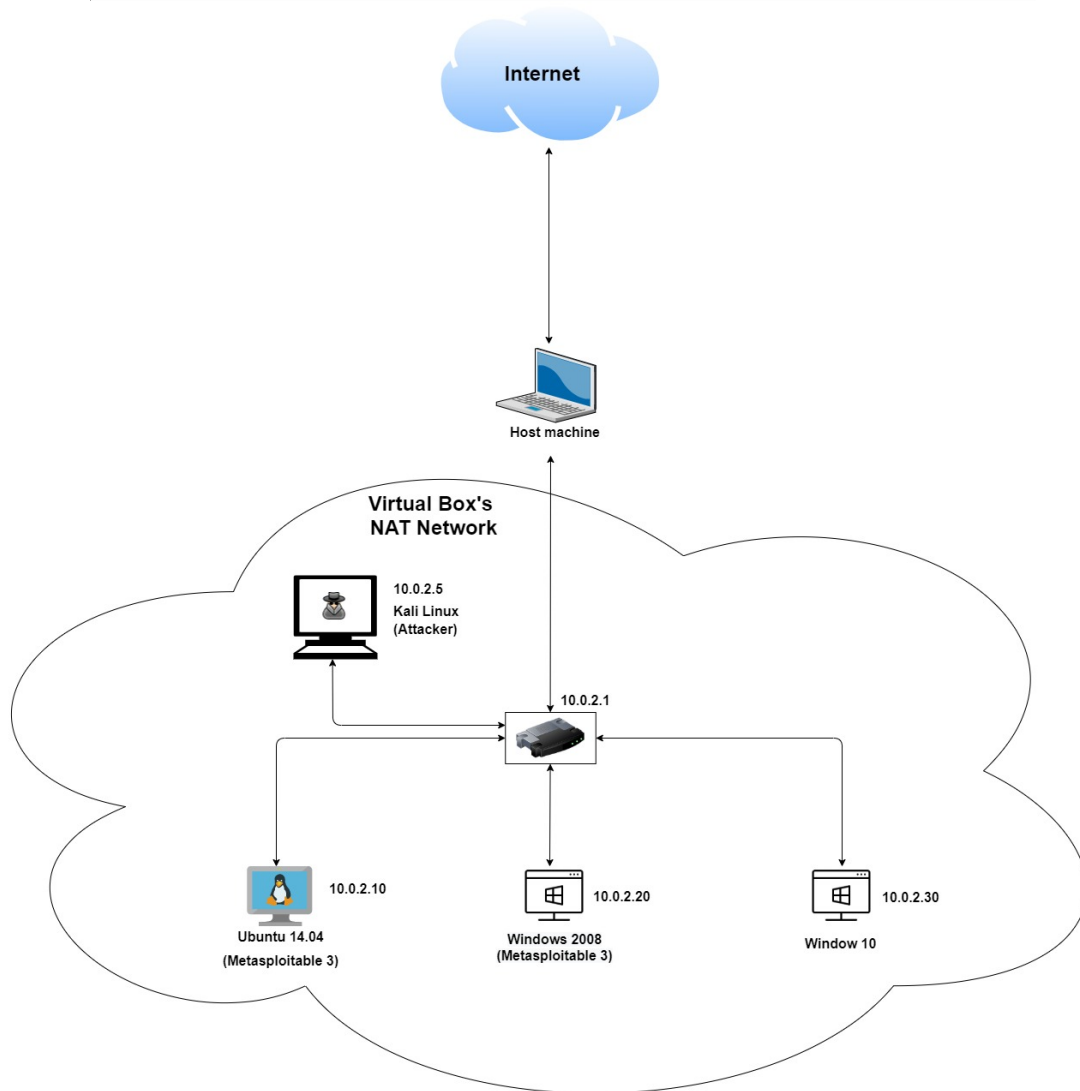


Figure 6: UCASES testbed network

## 4.2. Reconnaissance

### 4.2.1. Recon-ng

Recon-ng is a vast tool. It is composed of modules and each module does a specific job. The objective of this lab is for the students to gain familiarity with recon-ng and its user interface, be able to navigate within its framework and see how potent it actually is to gather information. We take a look at two specific modules "reverse-resolve" and "cache-snoop" that could be used for a task of gathering information for a penetration tester. It also exemplifies how to manage the recon-ng database and its default files. This lab is one of the few parts of the UCASES that does not use the testbed itself but a public network.

First, the lab starts with an introduction on how to navigate through the different options of recon-ng, set a “nameserver”, use the Linux shell from within recon-ng itself and search for modules.

```
[recon-ng][default] > options set NAMESERVER 147.83.2.3
NAMESERVER => 147.83.2.3
[recon-ng][default] > █
```

Figure 7: Setting a NAMESERVER

Next, we use “reverse-resolve” that takes a block of IP addresses and sends PTR lookups to a DNS server to determine which of those IP addresses resolve into names. With this part, the students see how to define a netblock, select and use a module, and how recon-ng stores the information gathered from running the module in to its internal database. In this part of the lab, the students can see how to use this module and how it store its information to the recon-ng’s database. We can see the output of that database after the scan in *Figure 8*.

```
[recon-ng][default][reverse_resolve] > show hosts
```

rowid	host	ip_address	region	country	latitude	longitude	module
1	leaf-r.upc.edu	147.83.2.1					reverse_resolve
2	helicon2.upc.edu	147.83.2.2					reverse_resolve
3	backus.upc.edu	147.83.2.3					reverse_resolve
4	backus.upcnet.es	147.83.2.3					reverse_resolve
5	migracio-shaula-rang2.upc.es	147.83.2.4					reverse_resolve
6	peter3.upc.edu	147.83.2.5					reverse_resolve
7	helicon1.upc.edu	147.83.2.6					reverse_resolve
259	peter15.upc.edu	147.83.2.249					reverse_resolve
260	lamp.upc.edu	147.83.2.251					reverse_resolve
261	connect-rise.eu	147.83.2.251					reverse_resolve
262	tcn.upc.edu	147.83.2.251					reverse_resolve
263	event.upc.edu	147.83.2.251					reverse_resolve
264	smtp-proves.upc.edu	147.83.2.252					reverse_resolve
265	ackerman3.upc.es	147.83.2.253					reverse_resolve
266	bus-soatemp-nat.upc.es	147.83.2.254					reverse_resolve

```
[*] 266 rows returned
[recon-ng][default][reverse_resolve] > █
```

Figure 8: Recon-ng database showing reverse-resolve findings

After that, we use “cache-snoop”, which does a DNS cache snooping against a target and associates this records with antivirus update sites, to give the possible AV the target is using. In this section, the students can see the results that give the tool, but more important, they see that this specific tool uses a file of AV domains (see output in *Figure 9*) to run and note that this file could be modified to fit the needs. This happens with other modules and tools and it is an important thing see.

```
[recon-ng][default][cache_snoop] > shell cat /home/kali/.recon-ng/data/av_domains.lst
[*] Command: cat /home/kali/.recon-ng/data/av_domains.lst
dnl-01.geo.kaspersky.com
download797.avast.com
downloads2.kaspersky-labs.com
es-latest-3.sophos.com/update
es-web-2.sophos.com
es-web-2.sophos.com.edgesuite.net
es-web.sophos.com
es-web.sophos.com.edgesuite.net
forefrontdl.microsoft.com
guru.avg.com
liveupdate.symantec.com
liveupdate.symantecliveupdate.com
osce8-p.activeupdate.trendmicro.com
update.nai.com
update.symantec.com
www.dnl-01.geo.kaspersky.com
www.download797.avast.com
www.downloads2.kaspersky-labs.com
www.es-latest-3.sophos.com/update
www.es-web-2.sophos.com
www.es-web-2.sophos.com.edgesuite.net
www.es-web.sophos.com
www.es-web.sophos.com.edgesuite.net
www.forefrontdl.microsoft.com
www.guru.avg.com
www.liveupdate.symantec.com
www.liveupdate.symantecliveupdate.com
www.osce8-p.activeupdate.trendmicro.com
www.update.nai.com
www.update.symantec.com
[recon-ng][default][cache_snoop] > █
```

Figure 9: File of AV domains used by cache-snoop

#### 4.2.2. Nmap

Nmap is an extremely useful tool used for network discovery. This lab is separated in two parts. The objective of this first part is to see some of the options that Nmap provides, such as scan through a network range, port range, different kinds of scans (TCP & UDP) and analyse what happens when the packets contains bad checksums. The second part's objective is to see the OS fingerprinting option of Nmap, how accurate it is, and see the file that feature uses to decide what the target's OS is.

In this first part of the lab, the students will use tcpdump to analyse the traffic and see what is happening "behind the scenes". The students will use the Windows 10 machine and one of the Metasploitable ones, this way they will be able to see the difference in timing and results when the traffic is or is not filtered. In this practice, they also will see how to store the information of the nmap scans. We can see an example of a full scan of all port in *Figure 10*.

PORT	STATE	SERVICE
21/tcp	open	ftp
22/tcp	open	ssh
80/tcp	open	http
135/tcp	open	msrpc
139/tcp	open	netbios-ssn
445/tcp	open	microsoft-ds
3306/tcp	open	mysql
3389/tcp	open	ms-wbt-server
4848/tcp	open	appserv-http
7676/tcp	open	imqbrokerd
8009/tcp	open	ajp13
8022/tcp	open	oa-system
8031/tcp	open	unknown
8080/tcp	open	http-proxy
8181/tcp	open	intermapper
8383/tcp	open	m2mservices
8443/tcp	open	https-alt
9200/tcp	open	wap-wsp
49152/tcp	open	unknown
49153/tcp	open	unknown
49154/tcp	open	unknown
49155/tcp	open	unknown
49158/tcp	open	unknown

Figure 10: Full TCP SYN scan output

On the second part of the lab, focuses on OS fingerprinting, and the students will be able to see how Nmap can deduce what OS has the target if enough information is provided, and how can this be prevented if the traffic is filtered. They will also look at the files Nmap uses to use as its default behaviour (e.g., what ports will be scanned if no port is specified). We can see an example of the output of the OS fingerprinting in *Figure 11*.

```
Nmap scan report for 10.0.2.5
Host is up (0.00034s latency).
Not shown: 1018 closed ports
PORT      STATE SERVICE
21/tcp    open  ftp
22/tcp    open  ssh
80/tcp    open  http
135/tcp   open  msrpc
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
MAC Address: 08:00:27:EE:7D:70 (Oracle VirtualBox virtual NIC)
Device type: general purpose
Running: Microsoft Windows 7|2008|8.1
OS CPE: cpe:/o:microsoft:windows_7:- cpe:/o:microsoft:windows_7::sp1 cpe:/o:microsoft:windows_server_2008::sp1 cpe:/o:microsoft:windows_server_2008:r2 cpe:/o:microsoft:windows_8 cpe:/o:microsoft:windows_8.1
OS details: Microsoft Windows 7 SP0 - SP1, Windows Server 2008 SP1, Windows Server 2008 R2, Windows 8, or Windows 8.1 Update 1
Network Distance: 1 hop
```

Figure 11: OS fingerprinting on the vulnerable Windows 2008 machine.

#### 4.2.3. Scapy

Scapy is a packet manipulation program. It is able to create packets of a wide range of protocols, send and capture them, proving, unit testing, attacks, network discovery and more.

In this lab, we want the students to solve different “challenges” with Scapy. The objective is to browse the documentation to find the options necessary to create specific packets. We also ask them to use tcpdump commands to sniff those packets.

This lab’s objective is not only to familiarize the students with the concept of constructing specific packets, that could be useful in penetration testing, but also to grow more used to browsing the documentation of Scapy & tcpdump to figure out a way to solve an “obstacle” using a specific tool. This skill is highly useful on penetration testing since there are many tools with an extensive documentation.

This lab consist in four challenges.

First, the students will see the default Scapy behaviour and will be able to compose a simple command.

Second, some complexity is added making a loop on the packet sending and adding a payload that they will be able to see on the tcpdump, as we can see in the *Figure 12* with the payload "UCASES\_Challenge".

```

0.936125 IP 10.0.2.4 > 10.0.2.5: ICMP echo request, id 0, seq 0, length 24
 0x0000: 4500 002c 0001 0000 4001 62c8 0a00 0204  E.,....@.b.....
 0x0010: 0a00 0205 0800 1b31 0000 0000 5543 4153  ....1....UCAS
 0x0020: 4553 5f43 6861 6c6c 656e 6765             ES_Challenge
0.936479 IP 10.0.2.5 > 10.0.2.4: ICMP echo reply, id 0, seq 0, length 24
 0x0000: 4500 002c 0130 0000 8001 2199 0a00 0205  E.,..0....!.....
 0x0010: 0a00 0204 0000 2331 0000 0000 5543 4153  ....#1....UCAS
 0x0020: 4553 5f43 6861 6c6c 656e 6765 0000       ES_Challenge..

```

Figure 12: TCPdump output of challenge #2

Third, the students will recreate an old DoS attack, parched long ago, that consisted in sending a packet with the same IP and port of the recipient.

Fourth, using Scapy and tcpdump, we dig a little bit more in the state "filtered" seen in previous Nmap labs. We can see in the *Figure 13* that the port 10 & 11 are filtered not providing response and port 5347 is open providing it.

```

tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
10:24:56.739991 IP 10.0.2.4.20 > 10.0.2.15.10: Flags [S], seq 0, win 8192, length 0
10:24:56.741306 IP 10.0.2.4.20 > 10.0.2.15.11: Flags [S], seq 0, win 8192, length 0
10:24:56.742413 IP 10.0.2.4.20 > 10.0.2.15.5357: Flags [S], seq 0, win 8192, length 0
10:24:56.742807 IP 10.0.2.15.5357 > 10.0.2.4.20: Flags [S.], seq 497712914, ack 1, win 65392, options [mss 1460], length 0
10:24:56.743142 IP 10.0.2.4.20 > 10.0.2.15.5357: Flags [R], seq 1, win 0, length 0

```

Figure 13: Missing reset [R] response from target's ports 10 & 11 that are "filtered"

#### 4.2.4. NSE

Nmap Scripting Engine (NSE) is one of the most powerful features of Nmap. It allows the use of scripts within Nmap. We can create our custom scripts or use the ones already created by other people to automatize process that are repetitive or done multiple times with this tool. The objective of this lab is to take a peek at the scripts already present in Nmap, use some of them to see what they do (robots.txt.nse, nbstat.nse), look at some SMB scripts and finally look at the source code of sshv1.nse, and see why Nmap does a scan when we ask for it to execute a script.

First, the students will look at the file "script.db" (see the start of the file in *Figure 14*) within the Nmap files to see that there are all the scripts listed, and that those scripts have been categorized. We also will use the "wc" (Word Count) command to see how many scripts are there for a specific category.

```

1 Entry { filename = "acarsd-info.nse", categories = { "discovery", "safe", } }
2 Entry { filename = "address-info.nse", categories = { "default", "safe", } }
3 Entry { filename = "afp-brute.nse", categories = { "brute", "intrusive", } }
4 Entry { filename = "afp-ls.nse", categories = { "discovery", "safe", } }
5 Entry { filename = "afp-path-vuln.nse", categories = { "exploit", "intrusive", "vuln", } }
6 Entry { filename = "afp-serverinfo.nse", categories = { "default", "discovery", "safe", } }
7 Entry { filename = "afp-showmount.nse", categories = { "discovery", "safe", } }
8 Entry { filename = "ajp-auth.nse", categories = { "auth", "default", "safe", } }
9 Entry { filename = "ajp-brute.nse", categories = { "brute", "intrusive", } }
10 Entry { filename = "ajp-headers.nse", categories = { "discovery", "safe", } }
11 Entry { filename = "ajp-methods.nse", categories = { "default", "safe", } }
12 Entry { filename = "ajp-request.nse", categories = { "discovery", "safe", } }
13 Entry { filename = "allseeingeys-info.nse", categories = { "discovery", "safe", "version", } }
14 Entry { filename = "amqp-info.nse", categories = { "default", "discovery", "safe", "version", } }
15 Entry { filename = "asn-query.nse", categories = { "discovery", "external", "safe", } }
16 Entry { filename = "auth-owners.nse", categories = { "default", "safe", } }
17 Entry { filename = "auth-spoof.nse", categories = { "malware", "safe", } }
18 Entry { filename = "backorifice-brute.nse", categories = { "brute", "intrusive", } }
19 Entry { filename = "backorifice-info.nse", categories = { "default", "discovery", "safe", } }
20 Entry { filename = "bacnet-info.nse", categories = { "discovery", "version", } }
21 Entry { filename = "banner.nse", categories = { "discovery", "safe", } }
22 Entry { filename = "bitcoin-getaddr.nse", categories = { "discovery", "safe", } }

```

Figure 14: Content of the "script.db" file

Then, they will use the "http-robots.txt.nse" script to pull the robots.txt file from any website. This file is used to tell web crawlers to ignore specific directories or pages on a website that the owner does not want listed on search engines. It can be a good source of information for attackers, but is important to note that this file is not a security measure. It can be accessed and read by anyone with access to the document root of the web server.

After that, we will use "nbstat.nse" script, this script works like the "nbtstat.exe" command included in some versions of Windows, which pulls NetBIOS-over-TCP statistics, including machine names, MAC addresses and usernames. Using this with our vulnerable machines as target will return the MAC information of the machine and more. The students will be able to check it out easily with "ipconfig -all" (windows) or "ip a" (Linux).

Next, the students will take a look at all the smb scripts and the different types Nmap already have (vuln, enum, etc.).

Finally, the students will use "sslv1.nse" specifying the port 22, and answer the question "What happens if we are running SSH in a different port?" To answer this, they will look at the source code of the script, start an ssh daemon in port 23 (or any other except 22), and see that indeed the script finds the port 23 with ssh, as we can see in the Figure 15, and that's because the port scan Nmap does before running the script.

```

Not shown: 998 closed ports
PORT      STATE SERVICE      VERSION
23/tcp    open  ssh          OpenSSH 8.1p1 Debian 5 (protocol 2.0)

```

Figure 15: sslv1.nse finds SSH in port 23

#### 4.2.5. Nessus

Nessus is a vulnerability scanner. Unlike most of other practices, this one do not require the use of the terminal, aside for the installation of the package if needed. The objective of this lab is to see what Nessus is capable of and to see what a scan in to our environment machines retrieves. Also it the students will familiarize themselves with its



GUI. Nessus is a proprietary software, and it has paid versions, with many features, but it also has an “essentials” version targeted for students and cybersecurity novices that will allow us to test it and what it is capable to do on a small scale.

In this lab, the students will see, first, how to create a Nessus policy, a custom template defining what actions are performed during a Nessus scan. The students will, at least see, systematically most of what a Nessus policy configuration offers. From basic settings of a user authentication credentials, plugin selection and details, SSH configurations, password guessing, brute forcing of users, etc. After that, they will use that policy to launch a full scan to our three target machines (Windows 10, Linux 14.04 and Windows 2008), we can see the result of the scan in the *Figure 16*, and they will explore the results and see how that scan can be exported.

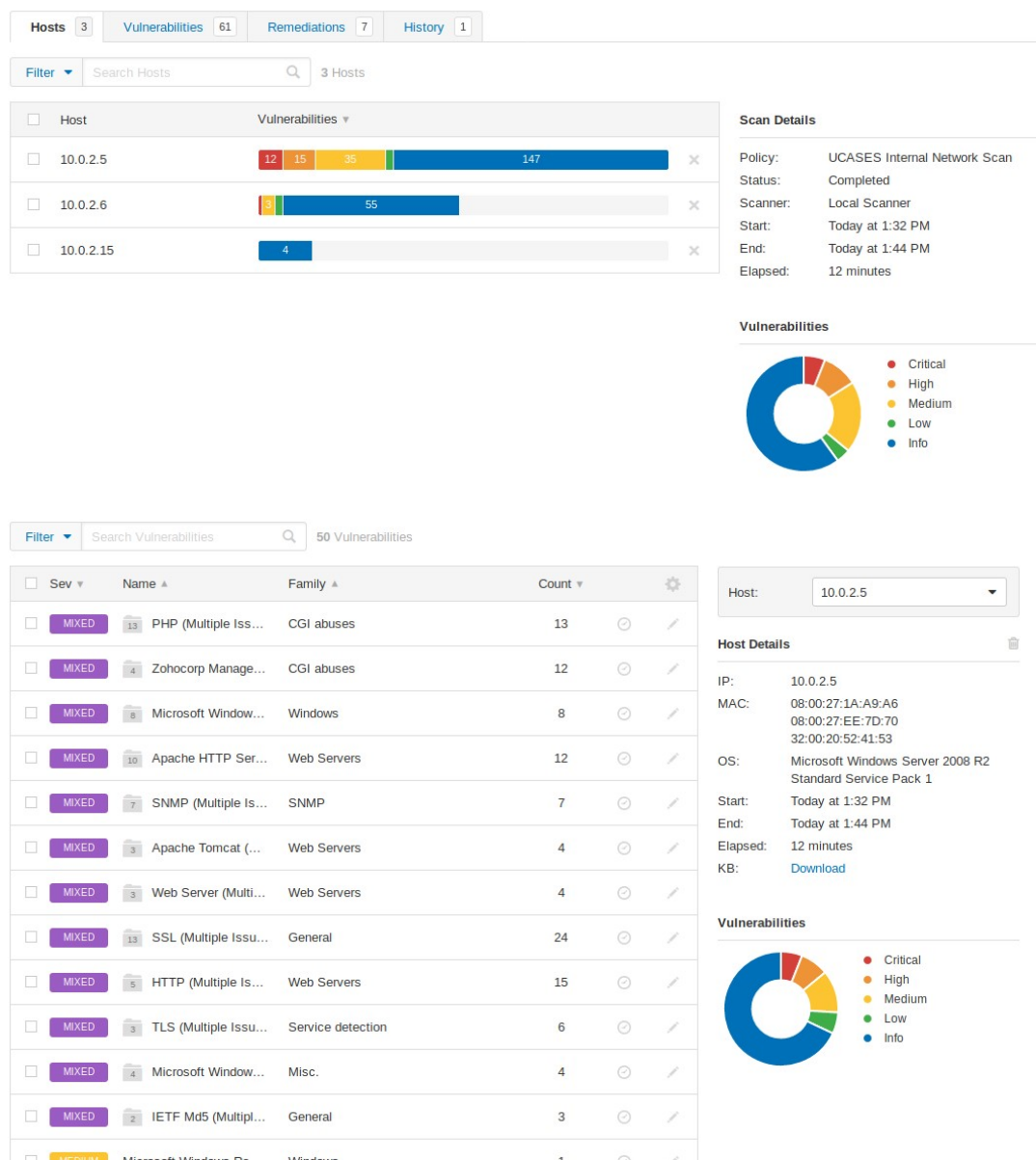


Figure 16: Nessus scan of UCASES target machines



#### 4.2.6. Netcat

The objective of this lab is to show the students Netcat, introduce them to the command if they never used it, and to see some different examples and cases of Netcat. This is a short lab in comparison to others, but we deemed important to see the tool working, for the people who never used it before, and to play a little with it with different scenarios (e.g. Chat between two machines, port scanning, etc.).

In this lab, first, the students will use Netcat between two machines to simulate a simple chat. Since Netcat can be installed on Windows, it can be done with any of the machines. After that, the students will gather information from connection strings from clients, for example Netcat on the port 22 of the target will reveal a SSH service. Then, using the browser to access a port where Netcat is listening to capture different User-Agent strings for different browsers. Finally, we will use the command line together with Netcat, to “create” a monitoring tool to verify if a service is still running.

#### 4.2.7. Nikto

Nikto is a vulnerability scanner. It is used “behind the scenes” as plugins for modern scanners, one example of a scanner that uses Nikto is Nessus, the scanner we saw in point 4.2.5 this chapter.

The objective of this lab is to introduce Nikto to the students, in this lab we will not enter in much detail in any part of the tool, but just see the basics on how to use the tool and do some basic scans. Since Nikto, as we said before, is used in the majority of automated scans, we thought important to at least introduce the tool and see how its basics works.

First, the students will do a quick check of the Nikto documentation provided inside the terminal itself (classics “-help” flag and “man” command). After that, they will perform a basic scan of a target host with only the IP and see the results and the time it takes. Next, they will be more specific in what the scan targets, such as ports and protocols, and see that the scan is faster. Then, the students will combine Nmap in a “greppable” format with Nikto to see how the two tools could be combined. Finally, we will introduce the mutate and test (-T) flags briefly so the students can see that we only scratch the surface of the tool.

### 4.3. Exploiting

#### 4.3.1. Metasploit

Metasploit is a well-known and extensive penetration-testing platform with build in utilities and commands, such as nmap. This lab is one of the longest in UCASES, and it is separated in two main parts, reconnaissance and exploiting.

The objective of this lab is to perform an attack on our Windows 2008 Metasploitable machine, being the final objective to gain a shell with root access on the said machine. The students will see a small mock example of what is a penetration testing attack. They will start gathering information using knowledge of previous labs. Then, with that information will look for an attack vector exploiting the vulnerable services the machine has. From there, using Metasploit, they will try to introduce a malicious payload to exploit those vulnerabilities thus completing the attack.

First, the student will populate the Metasploit database, using the build in nmap command (db\_nmap). This information would be among the lines of: what ports are open, what services are running on those ports, what version of the services, etc. We can see an example on the *Figure: 17*. To do this part they will need to review what was explain on the nmap lab.

host	port	proto	name	state	info
10.0.2.15	21	tcp	ftp	open	Microsoft ftpd
10.0.2.15	22	tcp	ssh	open	OpenSSH 7.1 protocol 2.0
10.0.2.15	80	tcp	http	open	Microsoft IIS httpd 7.5
10.0.2.15	4848	tcp	ssl/appserv-http	open	
10.0.2.15	5985	tcp	http	open	Microsoft HTTPAPI httpd 2.0 SSDP/UPnP
10.0.2.15	8020	tcp	http	open	Apache httpd
10.0.2.15	8022	tcp	http	open	Apache Tomcat/Coyote JSP engine 1.1
10.0.2.15	8027	tcp		open	
10.0.2.15	8080	tcp	http	open	Sun GlassFish Open Source Edition 4.0
10.0.2.15	8282	tcp	http	open	Apache Tomcat/Coyote JSP engine 1.1
10.0.2.15	8383	tcp	ssl/http	open	Apache httpd
10.0.2.15	8484	tcp	http	open	Jetty winstone-2.8
10.0.2.15	8585	tcp	http	open	Apache httpd 2.2.21 (Win64) PHP/5.3.10 DAV/2
10.0.2.15	9200	tcp	wap-wsp	open	
10.0.2.15	49153	tcp	msrpc	open	Microsoft Windows RPC
10.0.2.15	49154	tcp	msrpc	open	Microsoft Windows RPC

Figure 17: Populated Metasploit services table

Once the reconnaissance part is done, the objective will be to gain root access to the target, to do so the students will try to introduce a malicious payload into the target using the information obtained previously. There are many ways to gain root access in the vulnerable machine, but the guide explains a non-direct way using two vulnerable services. First, the students will exploit a non-root service providing them a shell with local privileges and a payload in the Windows file system. From there, the students will need to find another vector to gain direct root access or, as it explained in the guide, exploiting another service with root access, but only in Java context, and from there executing the payload left previously thus gaining root access on a system context. Finally, the students will explore what they can do with the root access for example "hashdump" to output the Windows password hashes, as we can see on *Figure 18* or "screenshar" to have real time feed of the victim screen as shown in *Figure 19*.

```
meterpreter > hashdump
Administrator:500:aad3b435b51404eeaad3b435b51404ee:e02bc503339d51f71d913c245d35b50b:::
anakin_skywalker:1011:aad3b435b51404eeaad3b435b51404ee:c706f83a7b17a0230e55cde2f3de94fa:::
artoo_detoo:1007:aad3b435b51404eeaad3b435b51404ee:fac6aada8b7afc418b3afea63b7577b4:::
ben_kenobi:1009:aad3b435b51404eeaad3b435b51404ee:4fb77d816bce7aeee80d7c2e5e55c859:::
boba_fett:1014:aad3b435b51404eeaad3b435b51404ee:d60f9a4859da4feadaf160e97d200dc9:::
chewbacca:1017:aad3b435b51404eeaad3b435b51404ee:e7200536327ee731c7fe136af4575ed8:::
c_three_pio:1008:aad3b435b51404eeaad3b435b51404ee:0fd2eb40c4aa690171ba066c037397ee:::
darth_vader:1010:aad3b435b51404eeaad3b435b51404ee:b73a851f8ecff7acafbaa4a806aea3e0:::
greedo:1016:aad3b435b51404eeaad3b435b51404ee:ce269c6b7d9e2f1522b44686b49082db:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
han_solo:1006:aad3b435b51404eeaad3b435b51404ee:33ed98c5969d05a7c15c25c99e3ef951:::
jabba_hutt:1015:aad3b435b51404eeaad3b435b51404ee:93ec4eaa63d63565f37fe7f28d99ce76:::
jarjar_binks:1012:aad3b435b51404eeaad3b435b51404ee:ec1dcd52077e75aef4a1930b0917c4d4:::
kylo_ren:1018:aad3b435b51404eeaad3b435b51404ee:74c0a3dd06613d3240331e94ae18b001:::
lando_calrissian:1013:aad3b435b51404eeaad3b435b51404ee:62708455898f2d7db11cfb670042a53f:::
leia_organa:1004:aad3b435b51404eeaad3b435b51404ee:8ae6a810ce203621cf9cfa6f21f14028:::
luke_skywalker:1005:aad3b435b51404eeaad3b435b51404ee:481e6150bde6998ed22b0e9bac82005a:::
sshd:1001:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
sshd_server:1002:aad3b435b51404eeaad3b435b51404ee:8d0a16cfc061c3359db455d00ec27035:::
vagrant:1000:aad3b435b51404eeaad3b435b51404ee:e02bc503339d51f71d913c245d35b50b:::
meterpreter >
```

Figure 18: Hashdump of Windows credentials

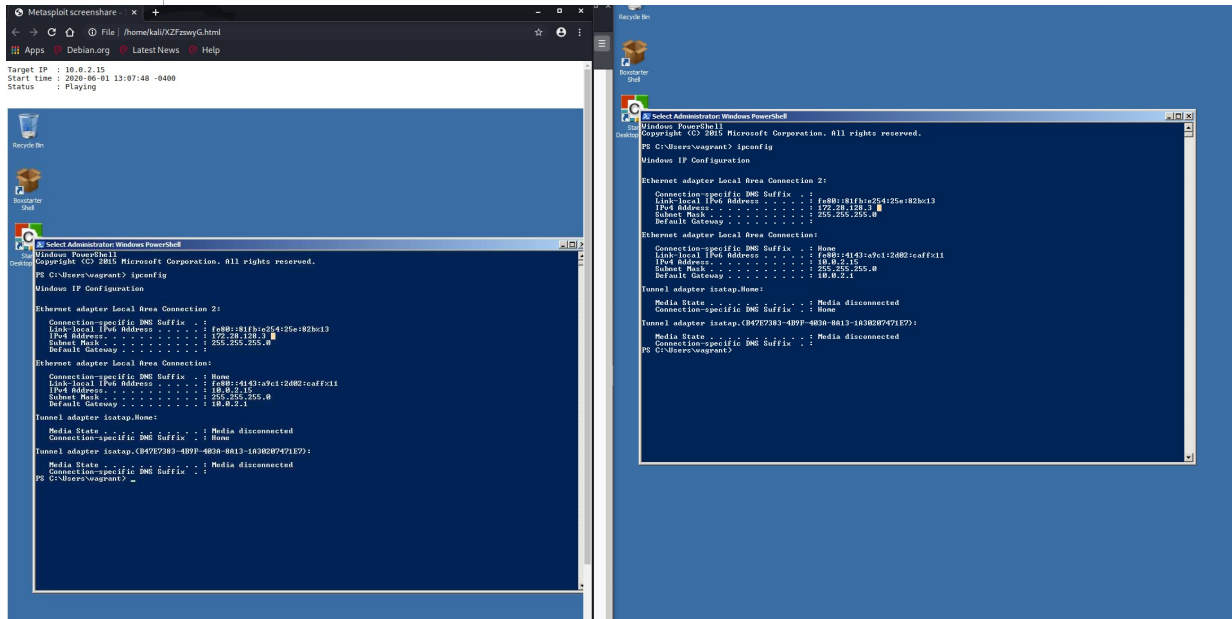


Figure 19: Real time screen of Windows machine

### 4.3.2. Empire

Empire is similar to Metasploit, but specific to PowerShell. The objective of this lab is for the students to see some of the Empire features on a basic level. Contrary to the Metasploit lab, this lab will not pay attention on the exploitation process; will introduce the malicious file forcefully on a non-real scenario, focusing on the objective on seeing some of the Empire modules.

First, the students select and set up an http listener, a process that will listen for a connection from the target machine. Then, they will create a stager to run an agent via PowerShell out of a ".bat" file. After that, the students will introduce that file into the target system, even if the malicious file is detected. Next, the student will test multiple information gather modules.

First, "situational\_awareness/host/winenum" that without admin access will pull some valuable information.

Second, "privesc/powerup/allchecks" that will look for privilege escalation.

Third, the students will try "credentials/powerdump" but it will need admin access. To gain the elevated privileges they will use "privesc/ask", that will prompt the user with a pop up to ask for admin (Most users would click "Yes" without a second hesitation). Once the students have admin, they will dump the hashed passwords of the target.

Finally, the students will end with some fun exploring modules from the "trollsploit" family.



## 5. Budget

This project was developed using open source tools such as virtual box, Kali Linux OS, Metasploitable OS, etc. This means that those tools have no cost for our project since they are completely free and publically available. Any tools that are not open source a free version is used so no additional cost is added from those tools either.

We will consider the salary of an engineer as approximately 32.000 €/year working 40 hour/week and 4 weeks/month. That makes the hourly salary of about 25 €/hour. The duration of this project is 12 ECTS credits that makes about 300 hours.

Concept	Cost
Engineer	7000 €

*Table 1: Budget*



## 6. Conclusions and future development

This thesis has been able to develop a suitable testbed for the MATT's subject UCASES, This testbed contains four virtual machines used on different labs of the subject, from reconnaissance analysis, to penetration testing and exploiting.

In addition, this thesis also provides a VM with all the necessary tools that the students would need for UCASES, with the latest versions at the time.

Finally, it also contains a detailed guide for the subject that contains all the necessary information to be able to perform the mandatory tasks to study UCASES. This includes a systematic guide on how to set up the required scenario for the UCASES' labs, how to solve the most common problems the students might encounter, and finally, a complete explanation introducing the tools and some guided exercises to see how those tools work.

### 6.1. Difficulties

There have been two main difficulties, we did not foresee, when developing this project, and they were the following:

- Too many options at each step
- Missing or loose documentation.

These problems did not break the planning or made us make major changes to the thesis. The issue they caused was a heavy time impact on the project.

The first drawback we faced was having too many options. Starting this project, we realized right away that we would need to make some decisions, what we did not anticipate is the vast amount of options, this decisions had. One example is the attacker OS, we have seen the options we balanced at the end, but there are plenty of operating systems that can be used for this project and that we looked at (e.g. Fedora, Gentoo Linux, tails, Pentoo [based on Gentoo], etc.) and this are just a couple of them. As we can see, even if you discard some of them quickly, going through, and looking at them, consumes an absurd amount of time.

The second obstacle we dealt with was the loose or missing documentation on some of the tools. UCASES uses many tools along its semester, and not all of these tools' documentation is clear or easily found. Moreover, we wanted to do specific labs, with specific steps to introduce the tool, so many tutorials were of no use. An example of that issue was Recon-ng, which most of the documentation or tutorials found were outdated since the last major version change. This leads to first, look for a reliable source to start (such as the help command) and go from there, leading to a trial and error to find what we want for each specific part of the lab.

### 6.2. Testing

The testing done for the project is straightforward. UCASES can be divided in multiple labs, and the guide has been developed as such. Once the testbed was running we updated the labs separately, systematically and making sure, no problems were found



along the way. When the entire guide was finished, we did all the labs over again from a clean testbed and Kali, making sure all worked and the commands were correct. If any problem was found, (e.g. tool not included to path), we added the solution to the guide to make sure that if a student bumps on that problem, he can easily solve it.

### 6.3. **Future developments**

This project could be just a starting point to creating a virtual environment within the UPC's servers. This route of development creates two main proposals to expand this project.

First, this testbed could be deployed within the university servers to create a full virtual environment so the students only need to install Kali Linux or any other attacker machine they want and attack the virtual environment within the UPC. Even if this task might not merit a master thesis, the creation of a continuous integration platform to be able to modify, add, reset, etc. the VMs does. This would make the process of modifying the virtual environment much more easy and safe by the lecturer. Having automated tests to make sure any change done to the environment does not break any existing lab is a priceless feature to have.

Second, UCASES is not the only subject within the cybersecurity track from MATT that would benefit from a testbed, for example the subject Network Security (NS), had some practices that were done within the student's computer. That carried some problems due a different configurations and VMs.

To summarize, here are the proposals to expand and continue this project:

- Deploy the testbed to UPC servers and create a CI platform to be able to modify the testbed easily and securely.
- Expand the testbed to encompass multiple MATT subjects to standardize the student's environments to prevent inconsistency problems and, if necessary update the labs.



## **Bibliography**

- [1] «TryHackMe,» [Online]. Available: <https://tryhackme.com/>. [Accessed: 02/2020].
- [2] «Pentesterlab,» [Online]. Available: <https://pentesterlab.com/>. [Accessed: 02 /020].
- [3] «HackTheBox,» [Online]. Available: <https://www.hackthebox.eu/>. [Accessed: 02/2020].
- [4] «Offensive-security.com,» [Online]. Available: <https://www.offensive-security.com/labs/>. [Accessed: 02/2020].
- [5] «Tools.kali.org,» [Online]. Available: <https://tools.kali.org/information-gathering/recon-ng>. [Accessed: 01/2020].
- [6] «Nmap.org,» [Online]. Available: <https://nmap.org/>. [Accessed: 01/2020].
- [7] «Nmap.org,» [Online]. Available: <https://nmap.org/book/nse.html>. [Accessed: 02/2020].
- [8] «Scapy.readthedocs.io,» [Online]. Available: <https://scapy.readthedocs.io/en/latest/introduction.html>. [Accessed: 02/2020].
- [9] «SearchNetworking,» [Online]. Available: <https://searchnetworking.techtarget.com/definition/Nessus>. [Accessed: 02/2020].
- [10] «En.wikipedia.org,» [Online]. Available: <https://en.wikipedia.org/wiki/Netcat>. [Accessed: 02/2020].
- [11] «Cirt.net,» [Online]. Available: <https://cirt.net/nikto2-docs/introduction.html>. [Accessed: 03/2020].
- [12] «Tools.kali.org,» [Online]. Available: <https://tools.kali.org/exploitation-tools/metasploit-framework>. [Accessed: 03/2020].
- [13] «GitHub,» EmpireProject/Empire, [Online]. Available: <https://github.com/EmpireProject/Empire/wiki>. [Accessed: 03/2020].
- [14] «Docs.microsoft.com,» microsoft, [Online]. Available: <https://docs.microsoft.com/es-es/virtualization/hyper-v-on-windows/about/>. [Accessed: 01/2020].
- [15] «Parallels.com,» [Online]. Available: <https://www.parallels.com/es/>. [Accessed: 02/2020].
- [16] «Docker,» [Online]. Available: <https://www.docker.com/>. [Accessed: 02/2020].



- [17] «Virtualbox.org,» Oracle, [Online]. Available: <https://www.virtualbox.org/>. [Accessed: 02/2020].
- [18] «VMware,» [Online]. Available: <https://www.vmware.com/>. [Accessed: 02/2020].
- [19] «Kali.org,» [Online]. Available: <https://www.kali.org/>. [Accessed: 02/2020].
- [20] «Releases.ubuntu.com,» [Online]. Available: <https://releases.ubuntu.com/18.04.4/>. [Accessed: 02/2020].
- [21] «BackBox.org,» [Online]. Available: <https://www.backbox.org/>. [Accessed: 02/2020].
- [22] «Parrotsec.org,» [Online]. Available: <https://www.parrotsec.org/>. [Accessed: 02/2020].
- [23] «Blackarch.org,» [Online]. Available: <https://blackarch.org/>. [Accessed: 02 2020].
- [24] «Offensive-security.com,» [Online]. Available: <https://www.offensive-security.com/bug-bounty-program/>. [Accessed: 02/2020].
- [25] «GitHub,» rapid7, [Online]. Available: <https://github.com/rapid7/metasploitable3>. [Accessed: 02/2020].



UNIVERSITAT POLITÈCNICA DE CATALUNYA  
BARCELONATECH

Escola Tècnica Superior d'Enginyeria  
de Telecomunicació de Barcelona



## **Glossary**

**AV** – Antivirus

**CI** – Continuous Integration

**CtF** – Capture the Flag

**DoS** – Denial of Service

**GUI** – Graphical User Interface

**NSE** – Nmap Scripting Engine

**OS** – Operative System

**UPC** - Universitat Politècnica de Catalunya

**UCASES** – Cybersecurity Use Cases (UPC subject)

**VM** – Virtual Machine