



Universidad de Oviedo  
*Universidá d'Uviéu*  
University of Oviedo



Escuela de  
Ingeniería  
Informática  
Universidad de Oviedo

# *DEVELOPMENT OF BASE MACHINES FOR EXISTING INFRASTRUCTURES WITH ENHANCED SAFETY USING INTERNATIONAL STANDARDS*

**BACHELOR OF SOFTWARE ENGINEERING**

**END OF DEGREE PROJECT**

**AUTHOR**

María Flórez Miranda

**DIRECTOR**

José Manuel Redondo López

**July 2022**



***Este documento ha sido creado basándose en la plantilla elaborada por JOSÉ MANUEL REDONDO LÓPEZ. [1] [2]***



# Acknowledgements

*Quiero agradecer a mi familia y a mis amigos por estar siempre ahí animándome por complicadas que se vuelvan las cosas; a mis compañeros de clase, que hicieron que estos años de Universidad fueran los mejores hasta la fecha; y a mi tutor, José Manuel Redondo, por depositar tantísima confianza en mí y en mi trabajo y por su inestimable ayuda en todo momento del proyecto*

# Abstract

Although the need of applying good cybersecurity practices in daily life has become an extremely important issue in the recent years, it is not uncommon to find that organizations focus their efforts on securing certain areas of their infrastructure to the detriment of others, generating security holes in the latter. A clear example are networks, which tend to be strongly protected, and rightly so, against external and internal threats; however, the machines connected to these networks usually receive few or no improvements at all to their security configuration at its most basic level, since it is a tedious and mainly manual process, which requires expert knowledge and whose result depends on the sources followed to carry out this securitization.

This project has sought to attack this issue and offer a solution that allows increasing the base security of Linux and Windows machines while requiring the least possible manual intervention. To this end, recommendations from national and international standards, validated and widely used in the Cybersecurity field, have been followed. The ultimate goal is to deploy already secured base machines in an automated way, with the possibility of integrating them in existing infrastructures, as well as to provide organizations a way to increase the security of their machines by means of an unattended process, all while ensuring a certain level of quality provided by the followed standards.

Despite having been born out of necessity and not being commissioned per se, the main stakeholders of this project, the School of Computing Engineering and the University of Oviedo, have seriously considered making use of what has been developed to increase the security of their computer equipment in future academic years. Likewise, the project will be used as support material for the "Administration of Web Servers" subject of the Master's Degree in Web Engineering, starting next year.

# Resumen

Pese a que la necesidad de aplicar buenas prácticas de ciberseguridad en la vida diaria se ha convertido en un asunto de extrema importancia en los últimos años, no es extraño encontrar que las organizaciones centran sus esfuerzos en la securización de ciertas áreas de su infraestructura en detrimento de otras, generando agujeros de seguridad en estas últimas. Un claro ejemplo son las redes, que tienden a protegerse fuertemente, y con razón, ante amenazas externas e internas; sin embargo, las propias máquinas conectadas a esas redes suelen recibir pocas o nulas mejoras de su configuración de seguridad en su nivel más básico, ya que es un proceso tedioso y principalmente manual, que requiere de conocimiento experto y cuyo resultado depende de las fuentes seguidas para realizar dicha securización.

Este proyecto ha buscado atacar este problema y ofrecer una solución que permita incrementar la seguridad base de máquinas Linux y Windows requiriendo para ello la menor intervención manual



posible. Para ello, se han seguido las recomendaciones de estándares nacionales e internacionales, validados y de amplio recorrido en el campo de la ciberseguridad. El objetivo final es el de desplegar de manera automatizada máquinas base ya securizadas, con la posibilidad de integrarlas en infraestructuras existentes, así como otorgar a las organizaciones una forma de incrementar la seguridad de sus máquinas en un proceso desatendido, todo ello asegurando el nivel de calidad proporcionado por los estándares seguidos.

A pesar de haber nacido de una necesidad y no de un encargo per se, los principales interesados en este proyecto, la Escuela de Ingeniería Informática de Oviedo y la Universidad de Oviedo, han considerado seriamente hacer uso de lo desarrollado para incrementar la seguridad de sus equipos informáticos en futuros cursos académicos. Asimismo, el proyecto será usado como material de apoyo para la docencia de la asignatura Administración de Servidores Web del Máster en Ingeniería Web a partir del próximo año.

## Keywords

Cybersecurity, automation, hardening, unattended installation, Infrastructure as Code

## Palabras clave

Ciberseguridad, automatización, securización, instalación desatendida, Infraestructura como Código

# Table of Contents

<b>Chapter 1: Information System Planning .....</b>	<b>15</b>
1.1 PSI 1: Information System Planning Kick-Off .....	16
1.1.1 PSI 1.1: Analysis of the Necessity of the PSI .....	16
1.1.2 PSI 1.2: Identification of the objectives and scope of the PSI .....	17
1.1.3 PSI 1.3: Delegation of Responsibilities.....	18
1.2 PSI 2: Definition and Organization of the PSI.....	18
1.2.1 PSI 2.1: Specification of the Scope and Reach of the PSI.....	18
1.2.2 PSI 2.2: PSI Organization .....	21
1.3 PSI 3: Study of relevant information .....	23
1.3.1 PSI 3.1: Study of the current situation.....	23
1.3.2 PSI 3.2: Theoretical Concepts .....	26
<b>Chapter 2: Technological Architecture Definition .....</b>	<b>31</b>
2.1 PSI 7.1: Identification of Technological Infrastructure Needs .....	32
2.1.1 Alternatives for automation technologies to provision machines .....	32
2.2 PSI 7.2: Technological Architecture Selection.....	36
2.2.1 Selection of provisioning technology.....	36
2.2.2 Machine provisioning using Ansible and SSH .....	36
<b>Chapter 3: System Feasibility Study .....</b>	<b>39</b>
3.1 EVS 4, 5 y 6: Study and Valuation of the Solution Alternatives and Selection of the Final Alternative.....	40
3.1.1 Alternatives for security standards on which to base the hardening .....	40
3.1.2 Alternatives for Windows Operating System installed.....	42
3.1.3 Alternatives for automating infrastructure deployment.....	44
3.1.4 Alternatives for auditing tools .....	47
3.2 Study of Additional Tools .....	50
3.2.1 Tools for automated installation and generation of Vagrant boxes: Packer .....	50
<b>Chapter 4: Planning and Management of the End of Degree Project .....</b>	<b>51</b>
4.1 Project Planning .....	52
4.1.1 Identification of Stakeholders.....	52



4.1.2	Initial Planning. WBS .....	52
4.1.3	Risks .....	53
4.1.4	Initial Budget .....	56
4.2	Project Closure .....	57
4.2.1	Final Planning .....	57
4.2.2	Final Budget .....	58
4.2.3	Learnt Lessons.....	58
<b>Chapter 5: Analysis of the Information System .....</b>		<b>59</b>
5.1	ASI 1: System Definition .....	60
5.1.1	Determination of the System's Scope .....	60
5.2	Study of Automated Machine Creation and Unattended Installations .....	61
5.3	Analysis of the Available Benchmarks and Security Good Practices .....	62
5.3.1	CIS Microsoft Windows 10 Enterprise Benchmark, v1.12.0 .....	62
5.3.2	CIS Ubuntu Linux 18.04 LTS Benchmark, v2.1.0 .....	67
5.3.3	General good security practices .....	69
5.4	Analysis of Hardening Script Sources .....	70
5.4.1	Cyber Ansible .....	70
5.5	Integration between Hardening Sources .....	71
5.6	ASI 2: Establishment of Requirements.....	71
5.6.1	Acquisition of the System's Requirements .....	71
5.7	Specification of the Testing and Auditing Plan .....	74
5.7.1	Task monitoring .....	74
5.7.2	Auditing Plan .....	75
5.7.3	Tests to be manually performed.....	76
<b>Chapter 6: Design of the Information System .....</b>		<b>79</b>
6.1	Design of the Infrastructure Deployment Process.....	80
6.1.1	Steps.....	80
6.1.2	Hardening code to be deployed .....	82
6.2	DSI 5: Design of the Architecture of the System Modules.....	82
6.2.1	Deployment Diagrams .....	82
6.2.2	Package Diagrams .....	82
6.3	Technical Specification of the Testing and Auditing Plan .....	84

<b>Chapter 7: Construction of the Information System .....</b>	<b>85</b>
7.1 CSI 1: Preparation of the Generation and Construction Environment .....	86
7.1.1 Followed standards and regulations.....	86
7.1.2 Programming Languages.....	86
7.1.3 Tools and programs used for development and auditing .....	87
7.1.4 Operating Systems .....	88
7.1.5 Libraries used and consulted for analysis and development .....	89
7.1.6 Official tutorials and learning materials .....	90
7.2 Infrastructure Construction .....	91
7.3 Implemented Hardening Tasks .....	91
7.3.1 Detail of the hardening tasks .....	91
7.3.2 Results of the execution of the hardening tasks .....	92
7.4 Execution of System Audits.....	93
7.4.1 Windows 10 Education Virtual Machine audits.....	94
7.4.2 Ubuntu Linux Virtual Machine audits .....	103
7.4.3 Windows 10 real machines .....	105
7.4.4 Results' summary.....	111
7.4.5 Analysis of the results .....	113
7.5 Execution of Additional Tests.....	113
7.6 CSI 6: Elaboración de los Manuales de Usuario .....	115
7.6.1 Installation and Execution Manual .....	115
7.6.2 Programmer's Manual .....	121
<b>Chapter 8: Conclusions and Extensions .....</b>	<b>125</b>
8.1 Conclusions.....	126
8.2 Extensions .....	127
<b>Annexes.....</b>	<b>129</b>
Risk Management Planning.....	130
Methodology .....	130
Risk categories .....	130
Probability and Impact Matrix.....	130
Budget Planning .....	131
Personnel.....	131

Licenses.....	132
Material Resources.....	132
Indirect costs .....	132
Initial Budget Summary .....	133
Final Budget Summary.....	133
References and Bibliography .....	134
Delivered Contents.....	137
Contents.....	137

# Index of Figures

FIGURE 1. SPECIFIC FEATURES OF THE SYSTEM WHERE THE DEVELOPMENT WILL TAKE PLACE .....	23
FIGURE 2. RESULTS OF THE CLARA ANALYSIS FOR THE W10 PRO 2021-2022 MACHINE .....	25
FIGURE 3. RESULTS OF THE CLARA ANALYSIS FOR THE W10 PRO 2022-2023 MACHINE .....	25
FIGURE 4. SUMMARY OF THE FINAL RESULTS OF THE CIS-CAT LITE ANALYSIS FOR L1 AND L2 ENVIRONMENTS FOR THE W10 PRO MACHINES (2021-2022 AND 2022-2023) .....	25
FIGURE 5. ANSIBLE'S BASIC INFRASTRUCTURE .....	33
FIGURE 6. PUPPET'S BASIC INFRASTRUCTURE .....	34
FIGURE 7. CHEF'S BASIC INFRASTRUCTURE .....	35
FIGURE 8. SSH CONNECTION SETUP FLOW .....	37
FIGURE 9. GANTT DIAGRAM FOR THE INITIAL PLANNING .....	52
TABLE 2. LIST OF IDENTIFIED RISKS .....	56
FIGURE 10. GANTT DIAGRAM FOR THE FINAL PLANNING .....	58
FIGURE 11. SIMPLIFIED ARCHITECTURE TO BE DEPLOYED .....	82
FIGURE 12. PACKAGE DIAGRAM OF THE WHOLE DIRECTORY STRUCTURE (SIMPLIFIED) .....	83
FIGURE 13. PACKAGE DIAGRAM FOR THE PROVISIONERS' PACKAGES .....	84
FIGURE 14. W10 EDUCATION PRE-HARDENING AUDIT RESULTS WITH CLARA FOR A LOW CATEGORY SYSTEM .....	94
FIGURE 15. W10 EDUCATION PRE-HARDENING AUDIT RESULTS WITH CLARA FOR A MEDIUM CATEGORY SYSTEM .....	95
FIGURE 16. W10 EDUCATION PRE-HARDENING AUDIT RESULTS WITH CLARA FOR A HIGH CATEGORY SYSTEM .....	95
FIGURE 17. W10 EDUCATION PRE-HARDENING AUDIT RESULTS WITH CIS-CAT LITE FOR L1 PROFILE .....	96
FIGURE 18. W10 EDUCATION PRE-HARDENING AUDIT RESULTS WITH CIS-CAT LITE FOR L2 PROFILE .....	96
FIGURE 19. W10 EDUCATION POST-HARDENING (L1) AUDIT RESULTS WITH CLARA FOR A LOW CATEGORY SYSTEM .....	97
FIGURE 20. W10 EDUCATION POST-HARDENING (L1) AUDIT RESULTS WITH CLARA FOR A MEDIUM CATEGORY SYSTEM .....	98
FIGURE 21. W10 EDUCATION POST-HARDENING (L1) AUDIT RESULTS WITH CLARA FOR A HIGH CATEGORY SYSTEM .....	99
FIGURE 22. W10 EDUCATION POST-HARDENING (L1) AUDIT RESULTS WITH CIS-CAT LITE FOR L1 PROFILE .....	99
FIGURE 23. W10 EDUCATION POST-HARDENING (L1) AUDIT RESULTS WITH CIS-CAT LITE FOR L2 PROFILE .....	100
FIGURE 24. W10 EDUCATION POST-HARDENING (L1 + EXTRA L2) AUDIT RESULTS WITH CLARA FOR A LOW CATEGORY SYSTEM .....	100
FIGURE 25. W10 EDUCATION POST-HARDENING (L1 + EXTRA L2) AUDIT RESULTS WITH CLARA FOR A MEDIUM CATEGORY SYSTEM .....	101
FIGURE 26. W10 EDUCATION POST-HARDENING (L1 + EXTRA L2) AUDIT RESULTS WITH CLARA FOR A HIGH CATEGORY SYSTEM .....	102
FIGURE 27. W10 EDUCATION POST-HARDENING (L1 + EXTRA L2) AUDIT RESULTS WITH CIS-CAT LITE FOR L1 PROFILE .....	102
FIGURE 28. W10 EDUCATION POST-HARDENING (L1 + EXTRA L2) AUDIT RESULTS WITH CIS-CAT LITE FOR L2 PROFILE .....	103
FIGURE 29. UBUNTU PRE-HARDENING MACHINE AUDIT SUMMARY WITH LYNIS .....	103
FIGURE 30. UBUNTU POST-HARDENING MACHINE AUDIT SUMMARY WITH LYNIS .....	104
FIGURE 31. W10 PRO (2021-2022 ACADEMIC YEAR IMAGE) AUDIT RESULTS WITH CLARA FOR A LOW CATEGORY SYSTEM .....	105
FIGURE 32. W10 PRO (2021-2022 ACADEMIC YEAR IMAGE) AUDIT RESULTS WITH CLARA FOR A MEDIUM CATEGORY SYSTEM .....	106
FIGURE 33. W10 PRO (2021-2022 ACADEMIC YEAR IMAGE) AUDIT RESULTS WITH CLARA FOR A HIGH CATEGORY SYSTEM .....	107
FIGURE 34. W10 PRO (2021-2022 ACADEMIC YEAR IMAGE) AUDIT RESULTS WITH CIS-CAT LITE FOR L1 PROFILE .....	107
FIGURE 35. W10 PRO (2021-2022 ACADEMIC YEAR IMAGE) AUDIT RESULTS WITH CIS-CAT LITE FOR L2 PROFILE .....	108
FIGURE 36. W10 PRO (2022-2023 ACADEMIC YEAR IMAGE) AUDIT RESULTS WITH CLARA FOR A LOW CATEGORY SYSTEM .....	108

FIGURE 37. W10 PRO (2022-2023 ACADEMIC YEAR IMAGE) AUDIT RESULTS WITH CLARA FOR A MEDIUM CATEGORY SYSTEM.....	109
FIGURE 38. W10 PRO (2022-2023 ACADEMIC YEAR IMAGE) AUDIT RESULTS WITH CLARA FOR A HIGH CATEGORY SYSTEM.....	110
FIGURE 39. W10 PRO (2022-2023 ACADEMIC YEAR IMAGE) AUDIT RESULTS WITH CIS-CAT LITE FOR L1 PROFILE .....	110
FIGURE 40. W10 PRO (2022-2023 ACADEMIC YEAR IMAGE) AUDIT RESULTS WITH CIS-CAT LITE FOR L2 PROFILE .....	111
FIGURE 41. VAGRANT'S DOWNLOADS PAGE .....	115
FIGURE 42. VIRTUALBOX DOWNLOADS PAGE .....	116
FIGURE 43. DETAIL OF THE FOLDERS FOR EACH OF THE WINDOWS CUSTOM BOXES .....	117
FIGURE 44. EXAMPLE OF A RUN FOR ADDING A CUSTOM BOX TO THE LOCAL CACHE OF BOXES.....	117
FIGURE 45. EXAMPLE OF THE WINDOWS MACHINE AND THE CONTROLLER BOOTING UP.....	119
FIGURE 46. STARTING THE ANSIBLE PROVISIONING .....	120
FIGURE 47. A FINISHED ANSIBLE RUN.....	120
FIGURE 48. EXAMPLE OF A PLAYBOOK WITH SEVERAL ROLES .....	121
FIGURE 49. EXAMPLE OF VARIABLES FOR PASSWORD SETTINGS.....	122
FIGURE 50. RISK CATEGORIES ACCORDING TO PMBOK.....	130
FIGURE 51. PROBABILITY AND RISK MATRIX .....	131

## Index of Tables

TABLE 1. WORKTEAMS AND USERS ASSOCIATED TO EACH TEAM .....	22
TABLE 3. SUMMARY OF THE INITIAL INTERNAL BUDGET .....	57
TABLE 4. SUMMARY OF THE FINAL INTERNAL BUDGET .....	58
TABLE 5. CIS MICROSOFT WINDOWS 10 ENTERPRISE BENCHMARK'S SECTIONS' SUMMARY .....	64
TABLE 6. CIS UBUNTU LINUX 18.04 LTS BENCHMARK SECTIONS' SUMMARY .....	67
TABLE 7. WINDOWS 10 (EDUCATION AND PRO) COMPLIANCE SCORES WITH CLARA.....	112
TABLE 8. WINDOWS 10 (EDUCATION AND PRO) COMPLIANCE SCORES WITH CIS-CAT LITE .....	112
TABLE 9. UBUNTU LINUX COMPLIANCE SCORES WITH LYNIS.....	113
TABLE 10. PARAMETERS THAT CAN BE CONFIGURED FOR THE WINDOWS 10 MACHINE.....	123
TABLE 11. STRUCTURE OF THE CONTENTS UPLOADED .....	137
TABLE 12. PROJECT DIRECTORY STRUCTURE .....	137

# CHAPTER 1: INFORMATION SYSTEM PLANNING

**PLANNING PHASE**

**PSI**



## 1.1 PSI 1: INFORMATION SYSTEM PLANNING KICK-OFF

---

### 1.1.1 PSI 1.1: Analysis of the Necessity of the PSI

#### 1.1.1.1 PSI 1.1.1 Project Justification

Windows and Linux machines are often installed and then left as they come, with minimal or no additional security configurations and little control over what gets installed on them. This is not totally users' and administrators' fault: by default, both operating systems come with poor protection mechanisms, and hardening them usually takes time and effort, which not everyone and not every organization is willing to spend on this task. It becomes an especially tedious process for machines that are installed manually, and thus, systems stay with as little security measures as they came with.

Another common scenario, which may be derived from having very permissive protection measures in place, is that the machines get mistreated by installing software that will then stay regardless of its frequency of use or actual usefulness, because it's costly to clean the system. This situation is highly inadequate as well, since organizations could be unaware of all the programs and services installed on their machines, losing track of critical security updates for said software.

However, even if organizations decide to harden their machines, this is more often than not done without taking into account current security standards, or by following knowledge sources that are not verified or can not be fully trusted, leading to misconfigured systems that do not comply

The University of Oviedo is one of these organizations that aims at improving its overall security, and even though it has taken significant steps towards doing so, the efforts have been mainly directed at securing its networking infrastructure. For example, endpoint machines, the ones used by students and teaching staff in labs and classrooms, are left with a very basic configuration, insufficient for complying with current international security standards; additionally, the lack of automatization in deployments only makes this problem harder to solve. Despite the challenge, this educational institution is currently pursuing the ENS (Esquema Nacional de Seguridad) Certification from Spain, a set of rules and other security measures based on the ISO/IEC 27000 family of standards.

The need of addressing these issues in an efficient manner motivated this project's proposal by José Manuel Redondo López, on behalf of the University's Cybersecurity Committee. As such, the project must offer a way of allowing for the automated creation and hardening of machines that could be used in both physical and virtual labs of the various schools of the University of Oviedo, all following official international security standards that ultimately could lead to easily obtaining recognized security certifications, both nationally and internationally.



## 1.1.2 PSI 1.2: Identification of the objectives and scope of the PSI

The overall objective of this project, as it is briefly described in section *PSI 1.1: Analysis of the Necessity of the PSI*, is to provide a means of automatically deploying machines that have been hardened in an automated way following international security recommendations, guidelines and standards, so that the University of Oviedo can improve its security and become closer to achieving certifications. The key term is “automated”, as the main point of this project is to ease a tedious and error prone set of processes so that it becomes as unattended as possible, benefiting from the advantages Infrastructure as Code and automating tools provide for deploying new architecture.

As of now, the University uses Windows 10 as the main operating system for the computers of all its networks, both for classrooms and physical labs. These Windows 10 machines have a very basic security configuration, posing a huge security risk.

In addition to this, some schools belonging to this educational institution, particularly those that offer Engineering degrees, use Linux machines in the labs of certain subjects. Even though the objective of some of the labs is to learn how to install a Linux distribution from scratch, or how to make basic machines more secure, others would certainly benefit from having already hardened machines. It is also common that research teams from the University use for their work Linux machines that should be hardened from the beginning. And not only that: this institution’s Computing Department deploys Ubuntu machines as part of an infrastructure with virtual machines, which as in the last case, need to be hardened manually. So, the problem with these systems affects not only students, but professors and underlying technological architecture too.

Given what was commented in section *PSI 1.1: Analysis of the Necessity of the PSI*, three different scenarios, that are present in the University and this project needs to address, have been identified:

1. Machines that are left as they come by default because organizations (in this case, in the field of academics, though may apply to other areas) do not want to invest in securing them
2. Machines whose installation can not be automated
3. Machines that are hardened in an incorrect way, not following any kind of official standard

Thus, considering these three far from ideal situations, which can be tackled as separate problems but are essential parts of the global aim of this project, a set of intermediate goals have been defined:

1. **Automate the creation of secured machines following an Infrastructure as Code approach:** nowadays, the University sets up new machines by first configuring a “base” machine that contains all the software and configuration considered necessary, and then cloning said machine to replicate it in the rest of the computers. The process is not automated, needing to be performed manually, and changes are not easy to be made. Thus, this project will provide a way of creating, configuring and deploying new base machines, both for Windows and for Linux systems.
2. **Automate the installation of operating systems:** installing an operating system from scratch is often a long and tedious process in which a user must be in front of the computer to provide





the setup process with the information it needs, typically related to basic settings. It is rare for an educational network to have many different types of systems; settings are generally shared among computers with the same purpose, and don't usually change from one year to the next one. This project will analyze how to automate a Windows 10 installation from a basic OS ISO, and following the study, run an unattended installation and provide the resulting product to be used for setting up future University machines.

3. **Harden machines correctly following international security standards, and perform audit analysis on them:** as it has been mentioned, machines are usually badly hardened or not secured at all. The third objective of the project, which closely intertwines with the one exposed in the first place, will be to thoroughly examine a set of internationally recognized and validated security guidelines first, and then implement said guidelines on Windows and Linux machines, in an automated way. These hardened machines, along with all the implemented recommendations, will be provided as part of this project, so that users can not only deploy the base, secured machines, but also reuse all the code used to harden them to improve the security of already existing infrastructure. Additionally, using verified audit tools based on official security benchmarks, this project will test the machines before and after they are hardened, to assess the level of security a base machine has against the score it could get after implementing the security measurements described in this objective. This will be done along the phase, to check that the recommendations actually improve the machine's security.

### 1.1.3 PSI 1.3: Delegation of Responsibilities

- The **project's author**, that is, the student developing this project, will be in charge of creating the base machines, hardening them so that they are in line with recognized security standards, and testing them using reliable audit tools. All these tasks will be developed autonomously, without establishing contact with other colleagues that may be developing a similar project in parallel
- The **project's director** will be responsible for validating how the general and intermediate objectives of the project are being achieved while the project is being developed

## 1.2 PSI 2: DEFINITION AND ORGANIZATION OF THE PSI

---

### 1.2.1 PSI 2.1: Specification of the Scope and Reach of the PSI

Based on the objectives defined in section *PSI 1.2: Identification of the objectives and scope of the PSI*, the project is divided into the following phases, aligned with the intermediate objectives:



### ***1.2.1.1 Phase 1: Automate the creation of secured machines following an Infrastructure as Code approach***

Right now, the common client's scenario is that machines are manually created, configuring by hand all the necessary parameters and software; these base machines are then cloned onto whatever computers that need this specific image. Moreover, virtual machines used for labs follow a similar approach, so distributing them, making changes on their configuration, or ensuring they are consistent among all the users that use them could not be an easy task. This first phase of the project aims to address this issue, automating the creation and deployment of machines, following an Infrastructure as Code approach that will also ease the distribution of the machines.

In this first phase, different tools that allow the automated management of machines will be considered and discussed, as choosing the ideal one will be critical in other phases of the project. The ultimate goal of the phase will be to provide the user with base machines using an Infrastructure as Code approach, so they can then be used to integrate whatever security recommendations deemed appropriate.

The scope of this phase will include the automated deploying of virtual machines, addressing one of the problems discussed above. This could then, in the future, be extrapolated to physical machines, though further investigations must be carried out.

Objectives of the phase:

- Provide a means of automating the creation of an Ubuntu Linux machine
- Provide a means of automating the creation of a Windows machine
- Study the provision of Vagrant-managed machines using Ansible

### ***1.2.1.2 Phase 2: Automate the installation of operating systems***

Windows 10 installations are normally done manually, needing constant user input for the various prompts that appear along the setup process; this is what happens in the University of Oviedo for configuring new base machines. The first thing in this second phase of the project will be to study how to automate the installation of Windows 10, and how to export the resulting product into a format that can be used with the tools mentioned in Phase 1. After this study, an unattended installation will be run to set up base machines, after which they will be exported in the format required by the tool chosen for managing the infrastructure (first phase). Resources so that anyone interested can perform the same process or even export the resulting machine to other formats will be provided as well.

This scope of this phase will be reduced to automating the installation process of Windows 10, more specifically in its Education edition. This is mainly due to the student in charge of this project's access to official Windows 10 Education (ES and US) editions of this operating system, since they are made available to all University's students through an agreement with Microsoft. More detail into this decision can be found in *EVS 4, 5 y 6: Study and Valuation of the Solution Alternatives and Selection*



of the Final Alternative. Regarding the Linux machines that will be used in other phases of the project, fortunately official base machines, with minimal configuration but already with their installation complete, are easy to download and use in projects such as this.

Objectives of the phase:

- Automate the installation of a Windows 10 Education machine, preferably the latest-available version of the operating system
- Provide pre-configured configurable files and resources for creating custom Windows 10 unattended setups
- Provide custom base Windows 10 Education machines in the specific format required by the tool chosen in Phase 1

### ***1.2.1.3 Phase 3: Harden machines correctly following international security standards, and perform audit analysis on them***

Base Windows 10 and Ubuntu Linux machines will be hardened following security recommendations from official and validated sources. The chosen guidelines will be thoroughly examined to understand the impact they will have on the systems and the fitness for the client.

After checking the policies that will be implemented, that is, the recommendations that want to be applied for each of the machines, a way to automate the application of said recommendations will be devised. This automation will be aligned with the Infrastructure as Code approach followed in this whole project.

Additionally, the set of hardening recommendations will be provided to the client in such a way that they are not coupled to the specific provisioned machines; the code has been abstracted, so administrators and other interested parties will be able to extract it, adapt it and reuse it to harden existing infrastructures.

The machines will be audited pre and post hardening using verified and mature audit tools that are based on many official security benchmarks and guidelines. These tools will help assessing the default security levels the base machines have, against the intermediate and final scores the same machines get after the hardening process. Auditing will take place along the whole phase, to check that the recommendations actually improve the machine's security.

In this phase, only policies that comply with CIS Benchmarks will be included. They not only are detailed enough for the systems we want to focus on, but many of the policies they cover are also part of several distinct security benchmarks, mapping directly to other compliance guidelines. Thus, by covering CIS we are also covering many of the policies from other official benchmarks. More detail into their description, benefits and application scope can be found in section *PSI 3.2: Theoretical Concepts*.

Regarding auditing, there will be used both tools that are based on or directly test against CIS Benchmarks, as they are the policies that will be first implemented, and tools that allow checking the level of compliance with the ENS.

It is important to note that the hardening process will be done in an almost iterative way. Policies will be gradually implemented and configured in the machines, as this will allow tracking possible errors faster and check what remediations increase the security scores or render the machine useless. Thus, the audit and testing process will be performed almost at the same time as the implementation, though only the final reports will be provided as part of the documentation.

Objectives of the phase:

- Study CIS Benchmarks’ policies for Windows 10 and Ubuntu Linux, analyzing what policies should be considered to be applied
- Apply hardening policies recommended by CIS for the Windows 10 Education machine, following their CIS Benchmarks
- Apply hardening policies recommended by CIS for the Ubuntu Linux machine, following their CIS Benchmarks
- Provide the code for the automated hardening decoupled from the machines, so users can reuse said code to harden already existing machines
- Export the hardened machines in a format fit for virtualization so that they can be used as new base machines, though hardened
- Test the developed Linux and Windows machines before and after hardening, using ENS and CIS based audit tools, and compare their results against each other for each machine and against the ones obtained from the School of Computer Engineering’s computers

### 1.2.2 PSI 2.2: PSI Organization

After explaining the objectives of each phase of the project and the overall objective of the project, the following work teams have been established:

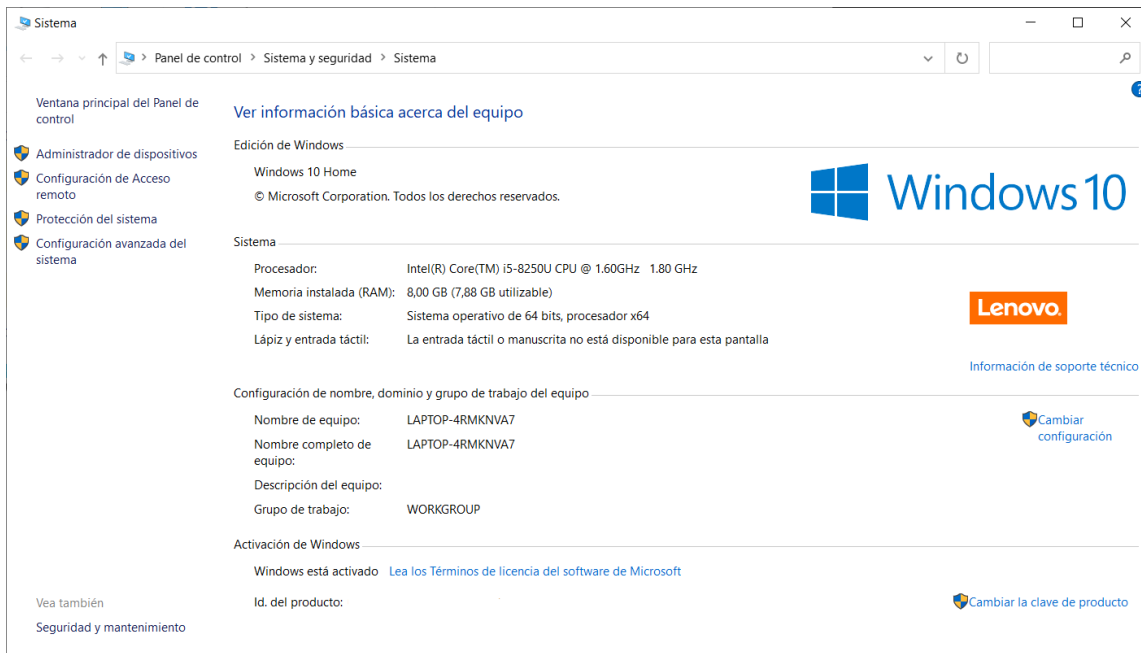
USER	PROFILE	TASKS/RESPONSIBILITIES
<b>REVIEWING TEAM</b>		
Project Director	Computer Scientist, Cybersecurity Professor and member of the University’s Cybersecurity Committee	Review that the intermediate objectives of the project have been/are being achieved Review the progress made by the student in charge of developing the project
<b>ANALYSIS AND DEVELOPMENT TEAM</b>		



Student	Computer Science Student (Software Engineering)	Carry out all the phases of the project and its intermediate objectives, including: <ul style="list-style-type: none"> <li>• Studying and analyzing security benchmarks and recommendations to harden the machines</li> <li>• Automating the deployment of base and hardened machines</li> <li>• Provisioning base machines with security recommendations, in an iterative way to continuously improve their security</li> <li>• Automating the installation of a Windows 10 operating system</li> </ul>
<b>TESTING AND AUDITING TEAM</b>		
Student	Computer Science Student (Software Engineering)	Ensure remediations are correctly applied and no unexpected errors during the provisioning process appear Carry out audits over the default and the hardened machines Document findings and possible failures, and analyze reports' results
School of Computer Engineering's scholars	Computer Scientists	Provide admin access to the School of Computer Engineering's computers to carry out auditing tests

**Table 1. Workteams and users associated to each team**

Regarding the materials for carrying out the project, the student will use a laptop computer for developing and testing the project, whose specific features will be detailed below. No other physical infrastructure will be needed during the development, as a virtualization platform will be used for deploying the machines.



**Figure 1. Specific features of the system where the development will take place**

Additionally, for having more in-depth and first-hand knowledge of the current security status of the machines used in the University, the student will request the School of Computing Engineering to let her test the images of the machines they have been using during the past academic year. For that, the school will need to provide a computer or computers with said image to be able to perform an audit.

## 1.3 PSI 3: STUDY OF RELEVANT INFORMATION

### 1.3.1 PSI 3.1: Study of the current situation

The outline of the client's current situation has already been established in section PSI 1.1: Analysis of the necessity of the PSI, but to fully understand the situation, talks with those responsible for the management and maintenance of computer systems in the University of Oviedo were held. Additionally, several tests to assess the security level of the machines in the University, more specifically in the School of Computer Engineering, were performed, to check first-hand the security regular computers have.

#### 1.3.1.1 University of Oviedo's current infrastructure and security measures

After gathering information from networks and systems administrators, the following was disclosed:

- The University's network is protected by two perimetral firewalls from two different manufacturers, thus complying with what's required by the ENS



- A VPN is used for connecting to internal services from outside the University's network
- 2FA has been recently made compulsory for every single access to services
- Networks are segmented, so that the ones for regular classrooms and physical labs (first group) are isolated from the management ones, which are also segmented
- Backups are done periodically for a large number of assets, including virtual machine copies of essential services in the cloud
- Redundancy for everything
- There exists an authentication domain for allowing users belonging to said domain (mainly students) to authenticate in any of the University's schools
- Users registered in the classrooms' and physical labs' computers are not local administrators. On the other hand, Teaching and Research staff manage their machines themselves and are not part of the domain, so in that case each user is responsible for the security and usage of their system

With this information, it can be concluded that the efforts aimed at improving the University's security are mainly directed towards networking, paying little attention to the individual machines.

### *1.3.1.2 On-premises machine tests using audit tools*

Regarding the on-site tests, they allowed to have a better insight of the actual situation of some of the machines used by the University, which is the focus of this project. They were carried out for both the machines that were used during the 2021-2022 academic year, and the ones that will be configured for the 2022-2023 academic year; details for the tools used in these analysis can be found in *Alternatives for auditing tools*.

The machines' operating system is Windows 10 Pro in both cases, in contrast with the one used in this project's development, which is Windows 10 Education. Nevertheless, though base scores after running the auditing processes may differ from one Windows version to the other, the general security guidelines still stand. A summary of the results is pictured below.

The analysis performed with the first of the tools, used to measure the percentage of compliance with the ENS, evaluates three categories of machines, from those that are in environments where security is not so crucial (BAJA or LOW) to those that operate in high security environments (ALTA or HIGH). The results the machine used during the 2021-2022 academic year obtained scores of 29,71%, 31,24% and 36,44% for the LOW, MEDIUM and HIGH categories respectively. On the other hand, the new, 2022-2023 academic year machine obtained better results: 50,02%, 50,64%, and 53,94% for the same categories.

### Centro Criptológico Nacional

Nombre del sistema: DESKTOP-02IGJTS  
Organización: Uniovi  
Unidad: Equipos Escuela 2021-2022  
Categoría del sistema: BAJA

Auditado por María  
Informes generados el día 28/06/2022 15:20:33 UTC  
Versión de CLARA: 2.0

Datos del sistema	
Análisis ENS	
Resultados	
Valor de criticidad	Cumplimiento (29,71%)

### Centro Criptológico Nacional

Nombre del sistema: DESKTOP-02IGJTS  
Organización: Uniovi  
Unidad: Equipos Escuela 2021-2022  
Categoría del sistema: MEDIA

Auditado por María  
Informes generados el día 28/06/2022 15:21:48 UTC  
Versión de CLARA: 2.0

Datos del sistema	
Análisis ENS	
Resultados	
Valor de criticidad	Cumplimiento (31,24%)

### Centro Criptológico Nacional

Nombre del sistema: DESKTOP-02IGJTS  
Organización: Uniovi  
Unidad: Equipos Escuela 2021-2022  
Categoría del sistema: ALTA

Auditado por María  
Informes generados el día 28/06/2022 15:19:27 UTC  
Versión de CLARA: 2.0

Datos del sistema	
Análisis ENS	
Resultados	
Valor de criticidad	Cumplimiento (36,44%)

Figure 2. Results of the CLARA analysis for the W10 Pro 2021-2022 machine

### Centro Criptológico Nacional

Nombre del sistema: DESKTOP-02IGJTS  
Organización: Uniovi  
Unidad: Equipos Escuela 2022-2023  
Categoría del sistema: BAJA

Auditado por María  
Informes generados el día 28/06/2022 15:08:23 UTC  
Versión de CLARA: 2.0

Datos del sistema	
Análisis ENS	
Resultados	
Valor de criticidad	Cumplimiento (50,02%)

### Centro Criptológico Nacional

Nombre del sistema: DESKTOP-02IGJTS  
Organización: Uniovi  
Unidad: Equipos Escuela 2022-2023  
Categoría del sistema: MEDIA

Auditado por María  
Informes generados el día 28/06/2022 15:06:50 UTC  
Versión de CLARA: 2.0

Datos del sistema	
Análisis ENS	
Resultados	
Valor de criticidad	Cumplimiento (50,64%)

### Centro Criptológico Nacional

Nombre del sistema: DESKTOP-02IGJTS  
Organización: Uniovi  
Unidad: Equipos Escuela 2022-2023  
Categoría del sistema: ALTA

Auditado por María  
Informes generados el día 28/06/2022 15:04:37 UTC  
Versión de CLARA: 2.0

Datos del sistema	
Análisis ENS	
Resultados	
Valor de criticidad	Cumplimiento (53,94%)

Figure 3. Results of the CLARA analysis for the W10 Pro 2022-2023 machine

The other tool, that checks compliance with CIS Benchmarks for general and high security environments, also produced low results for both machines, which have been simplified in a single figure as they obtained the same scores.

## Summary

Description	Tests				Scoring			
	Pass	Fail	Error	Unkn.	Man.	Score	Max	Percent
Total	85	258	0	3	1	85.0	346.0	25%
Total	89	350	0	3	1	89.0	442.0	20%

Figure 4. Summary of the final results of the CIS-CAT Lite analysis for L1 and L2 environments for the W10 Pro machines (2021-2022 and 2022-2023)

The full reports have been provided as part of the additional documentation, and more detail on how these audits were planned and carried out has been provided in sections *Specification of the Testing and Auditing Plan* and *Execution of System Audits*. Results obtained will be explained more thoroughly in the latter section.





## 1.3.2 PSI 3.2: Theoretical Concepts

This project deals with some novel technologies and programming techniques, as well as security standards which may not be known to non-specialized users. The present section has tried explaining some of the most relevant theoretical concepts so that any user can best understand the whole document.

### 1.3.2.1 *Infrastructure as Code (IaC)*

Infrastructure as Code is a process which consists of creating, managing and provisioning computer infrastructure using code in the form of machine-readable definition files, substituting with this having to do the full process manually (through physical hardware, configuration tools...). This approach has many clear and attractive benefits, and as businesses and organizations start choosing virtualization, containers and cloud computing over physical hardware, IaC's usage is becoming more widespread.

As the infrastructure specifications are included in plain configuration files, distributing and editing these configurations becomes easier than if performed in a manual way. This also ensures that the given environment will be prepared the same each time, improving consistency and reducing the number of possible errors.

Nowadays, it's not uncommon that an organization's infrastructure changes, needing to grow or even to get some of its parts dismantled or modified. Infrastructure as Code offers a way in which keeping track of the current infrastructure becomes easier to manage, and so modifications can be applied faster, with less errors, and less costly. Additionally, documentation and version control for these files is generally more straightforward, again due to their file nature.

Some examples of tools that help in the process of building Infrastructure as Code are Chef, Puppet, Ansible or Terraform. Vagrant is also considered part of this group, though it is specifically aimed at creating virtual machine environments and not whole infrastructures.

### 1.3.2.2 *Unattended installation*

An unattended installation refers to the process of setting up and deploying a software, such as an operating system, without requiring any manual intervention from the user. This not only leverages the installation process, which usually requires users to be present until the software has finished getting set up, but also makes possible performing simultaneous installations on several machines, saving deployment time. Nowadays, many Microsoft products, including their operating systems, support this kind of installation.

In the case of performing an unattended Windows installation, the process will use a special XML answer file (typically called Unattend.xml or Autounattend.xml) that contains all the necessary user input for filling the dialog boxes that are presented to the user during a normal setup process, as well



as some additional configuration data. Though users can write these files manually, or reuse an already existing one, Microsoft has made available to the public an application that makes it easier to create such files.

This kind of installations have been heavily popularized in the last years, especially since the outbreak of cloud-related operations and, though different to what this project intends to do, an interesting approach is cloud-init, a technology that provides cloud images for Linux and FreeBSD operating systems (Ubuntu, CentOS, RedHat...).

### *1.3.2.3 Center for Internet Security (CIS), CIS Benchmarks and CIS's Critical Security Controls*

The **Center for Internet Security (CIS)** is an international organization whose main purpose is to increase awareness of cybersecurity threats and promote good security practices among individuals, businesses and governments. CIS is in charge of developing and validating sets of recommendations based on best cybersecurity practices, in a joint effort between their vast number of security experts and other well-known companies; and they even provide computing environments that are already hardened, following their own security guidelines.

CIS makes available for anyone interested a set of hardening guides for a wide variety of products, from OSs to network devices and other pieces of software; these are called **CIS Benchmarks**. These documents, versioned and updated to adapt to new versions of the products they cover, recommend configuration values and other technical security controls and good practices that can be applied to said products. CIS Benchmarks are guaranteed to be reliable and high-quality, as they are created, validated and used by hundreds of renowned companies, as well as the US Government. Each benchmark is complete for the product it covers, specifying all the elements that should be configured to achieve whatever level of security the organization requires; and the recommendations they contain are described in full detail, including information on what should be fixed, how to do it and why, and the impact it could have on the system to modify the current configuration.

Each security control present in the benchmarks will have a certain impact on the overall security score the machine achieves once audited. Some of them do not affect this score, though its application is still recommendable, but other controls increase it if present or configured adequately. The ratio between the number of these two types of recommendations depends on the benchmark being considered.

Apart from affecting the security score of the machines, the controls can be classified according to usage profiles and levels. On the one hand, usage profiles refer to the functionality the product is supposed to fulfil; on the other hand, security levels, which are normally two (Level 1, for general use in corporate environments, and Level 2, for high security environments) but can be extended with specific ones, pertain to the environment where the machine that implements the control is supposed to operate.



The security controls in a CIS Benchmark can be grouped into **CIS Critical Security Controls**; both sets of measures, though called similarly, should not be mistaken for one another. CIS Critical Security Controls represent the best cybersecurity practices recognized worldwide (20 in v7 and 18 in v8, the last version as of the time of this project's development), and are subdivided into subcontrols. Each of the 20 or 18 controls can be divided into three different categories (Basic, Foundational and Organizational), and they model from basic cybersecurity operations to advanced ones, from Data Protection to Malware Defenses or even Penetration Testing. Its ultimate goal is to be able to set up all the security mechanisms that a company, no matter the size, could be benefitted from.

The main difference between the CIS Benchmarks and the CIS Critical Security Controls, apart from the scope, is that the former are designed for specific products, whereas the latter contain the same controls and subcontrols always, no matter the product the benchmark is targeted to.

#### *1.3.2.4 Spanish Centro Criptológico Nacional (CCN, National Cryptologic Center)*

The Centro Criptológico Nacional (CCN), or National Cryptologic Center, created in 2004, is the Spanish institution tasked with coordinating the activities of any of the Public Administration entities that use encryption procedures or resources, as well as providing training for any of their staff that specializes in the cryptology field. It is also responsible for ensuring the security of Information Technologies, or IT, in the scope of the Public Administration, remaining aware and informing of the acquisition of any cryptology material. In summary, it is the national organization that dedicates to cybersecurity, from risk and threat mitigation to promotion of good practices.

The CCN has developed a series of security solutions, such as audit tools or malware analyzers, among many others. It also promotes cybersecurity training and good practices through their own learning platform.

#### *1.3.2.5 Spanish Esquema Nacional de Seguridad (ENS, National Security Schema) and CCN-STIC Guides*

The Spanish **Esquema Nacional de Seguridad (ENS)**, or National Security Schema, is a set of guidelines, requirements and principles that helps creating and maintaining all the necessary security conditions for the correct usage of electronic means. These guidelines, if applied, ensure the security of systems, data, communications and other electronic services. It also takes into consideration different categories of information systems, from "Low" (sometimes referred to as "Basic" in the documentation) to "High", which reflect how serious the consequences of a security incident affecting the given system could be over the organization's activities, individuals or resources.

For public entities, the contents of the ENS allow satisfying the security requirements Public Administrations in Spain should comply with; and for regular citizens, it means that the public entities they relate with have all the necessary security conditions that ensure both their rights and their information are adequately protected. Spanish public universities are among the organizations that must apply the ENS, though only a few are certified.

The ENS is regulated by several Spanish Royal Decrees and rules, and though it does not have a direct correlation to CIS Benchmarks, being specifically for national implementation, it can be mapped to the well-known ISO 27001 security controls, establishing a bridge between the two sets of recommendations if necessary.

The **CCN-STIC Series**, which are sets of rules, recommendations and other instructions from the Spanish Centro Criptológico Nacional, publish free security guides aimed at helping entities configure their software for achieving compliance with the ENS. They provide complete and detailed explanations on what and how things need to be configured and why.

### 1.3.2.6 Domain-Specific Language (DSL)

A Domain-Specific Language or DSL is the opposite to a regular, general-purpose language such as Java, as it is a computer language specifically designed for a particular purpose, problem or domain. There is a great variety of DSLs, that can be generally divided into three categories: markup, modelling or specification and programming languages; in addition to this, there exist both textual and graphical DSLs, though the latter are more uncommon. Some examples include HTML, CSS or even regular expressions.

According to Martin Fowler and other authors [3], DSLs can also be categorized into Internal and External ones. Internal DSLs take advantage of a host language and use it in particular ways that give the appearance of coding in another, specific language; in contrast, External DSLs are independent, have a custom syntax, and specific parsers are written to be able to process them. Some of the languages in this last category can be encoded in data structure representations, like YAML; Ansible, one of the Infrastructure as Code tools mentioned previously, uses this approach.

### 1.3.2.7 Virtualization and Hypervisors

**Virtualization** is a technology that allows creating IT services (individual machines, servers, etc.) by means of mimicking features and resources traditionally associated to physical hardware. This works by using a physical host whose capabilities are then distributed among all the different services being simulated.

**Hypervisors** are one of the technologies that enable virtualization. They are a special kind of software that help separating the host's physical resources from all the virtualized environments, as they are responsible for dividing these resources and assigning them to each virtual guest. There exist two main possibilities when installing a hypervisor: it can either be installed on top of the operating system, or onto the hardware layer directly. The former is the preferred alternative when the host is a regular desktop or a laptop.

Nowadays, virtualization is heavily used, being such a widespread technology among enterprises that it is more and more common to make use of specialized management software for handling all the virtualized infrastructure.



# CHAPTER 2: TECHNOLOGICAL ARCHITECTURE DEFINITION

**PLANNING PHASE**

**PSI**



## 2.1 PSI 7.1: IDENTIFICATION OF TECHNOLOGICAL INFRASTRUCTURE NEEDS

---

Nowadays, as it has been mentioned before, the University performs the operations for installing and configuring its machines manually; and with the current approach, if its systems needed hardening, it would be done in the same way. This method takes time and could be error-prone, specially when securing the machines, as many guidelines require modifying low-level settings.

Thus, one of the first things that should be discussed is *how* to connect to and provision the machines in general, not only the ones specifically deployed in this project, but also those that are already deployed but need hardening. Different technologies that may serve this purpose will be evaluated, considering the infrastructure each of them needs; the ideal scenario is that the machines to be provisioned stay as clean as possible, with no additional software to be installed on them, no matter if they are virtualized environments or not. The infrastructure should remain simple in any of these cases.

### 2.1.1 Alternatives for automation technologies to provision machines

#### 2.1.1.1 Ansible

Ansible is an infrastructure managing tool, designed for the automated configuration and provisioning of local and remote machines, all following an Infrastructure as Code approach. This software is currently maintained by RedHat.

For configuring a machine using Ansible, code must be written in a special DSL language based on Python and encoded in the form of YAML files. It uses modules for most of its tasks, which can perform a wide variety of commands, from installing new software to managing system-specific features. Tasks and all the information needed to run them are grouped into playbooks, though when these files become too large, they can be reorganized into roles or collections, in a directory-like structure. Ansible will then use SSH to connect to the servers or machines and execute all the configured tasks.

##### 2.1.1.1.1 Advantages

Though not a purely technical detail, Ansible is backed by RedHat, one of the most important open-source enterprises in the world. Also, if choosing *Vagrant* as the tool to manage machines, it must be noted that it has direct support with Ansible, and includes several provisioners to either execute Ansible remotely or locally; configuration is also easy, requiring few parameters, though additional information can be specified if needed.

Compared to a similar product which will be seen below called Chef, Ansible is faster and easier to set up; it's also easier to manage because it uses a DSL disguised as YAML for its configurations and playbooks. To finish, Ansible's source of truth are the deployed playbooks, while Chef's one is a server that has uploaded cookbooks (the equivalent of playbooks): in this regard, the approach of the first technology makes more sense.

A really positive thing about Ansible is that it does not require any additional software on the controlled machines; it can be installed only in a control node, which is the machine from which all the other nodes will be managed, and where all the code will be placed. The only technology it needs to connect to the controlled nodes is SSH.

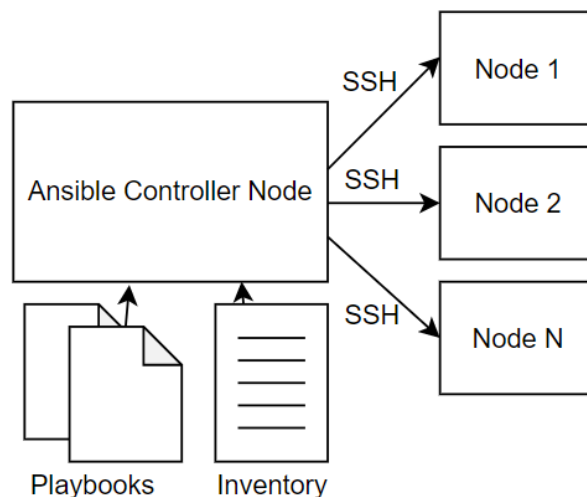
In general, Ansible is simpler to use and understand than other similar technologies, while remaining a powerful provisioning tool.

#### 2.1.1.1.2 Disadvantages

Compared to other technologies, Ansible's playbooks can be considered lists of commands to be run sequentially. This can make reusing tasks difficult, as well as running them in certain order, and though it can be more or less worked around with the usage of roles and specific groups of commands called handlers, it is not as flexible as with other tools.

Additionally, a huge drawback is that Ansible can not run on Windows hosts natively, though it can still provision Windows remote machines.

#### 2.1.1.1.3 Integration with the rest of the infrastructure



**Figure 5. Ansible's basic infrastructure**

- Ansible's official site: <https://www.ansible.com/>



### 2.1.1.2 Puppet

Puppet is an infrastructure automation tool similar to the others described in this section, maintained by a private company called Puppet, Inc. Puppet uses its own Ruby-based DSL, called PuppetDSL, which is also used to write manifests and modules through which the configuration of provisioned machines can be managed.

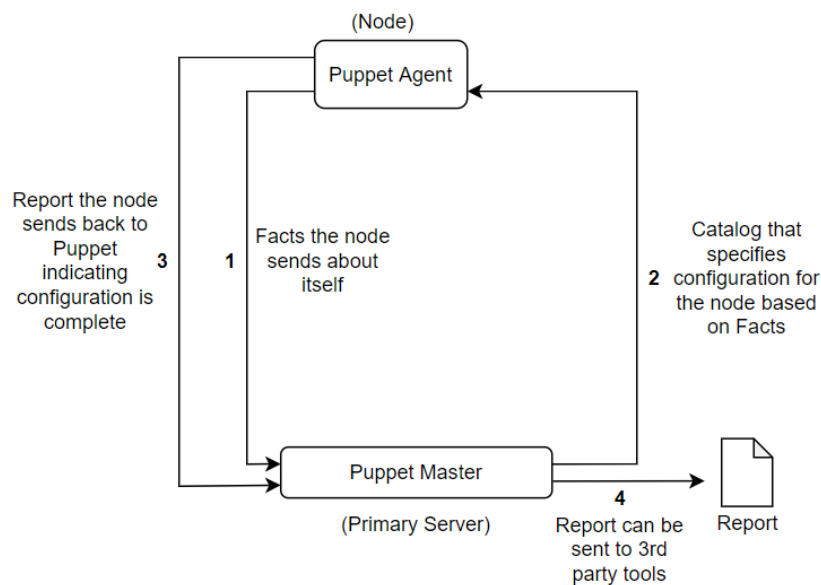
#### 2.1.1.2.1 Advantages

Puppet is a bit more flexible than other solutions, like Ansible, as Puppet's modules and manifests are reusable and can be run in any order.

#### 2.1.1.2.2 Disadvantages

The main disadvantage Puppet has is that it needs to have a special client in the machines that need to be controlled, which can be a huge problem if the organizations do not want or want to install additional software in said machines. The language in which the configuration files must be written is based on Ruby, so users might need to have at least some programming background in that language. Additionally, Puppet seems to have substantial differences between versions, which makes it harder for users to use it consistently.

#### 2.1.1.2.3 Integration with the rest of the infrastructure



**Figure 6. Puppet's basic infrastructure**

- Puppet's official site: <https://puppet.com/>

### 2.1.1.3 Chef

Chef is an open-source technology similar to Ansible, as it is also an automation software for configuring environments using an Infrastructure as Code approach. Chef groups administration tasks into their own sets of files, called cookbooks and recipes, in contrast to Ansible's playbooks and tasks.

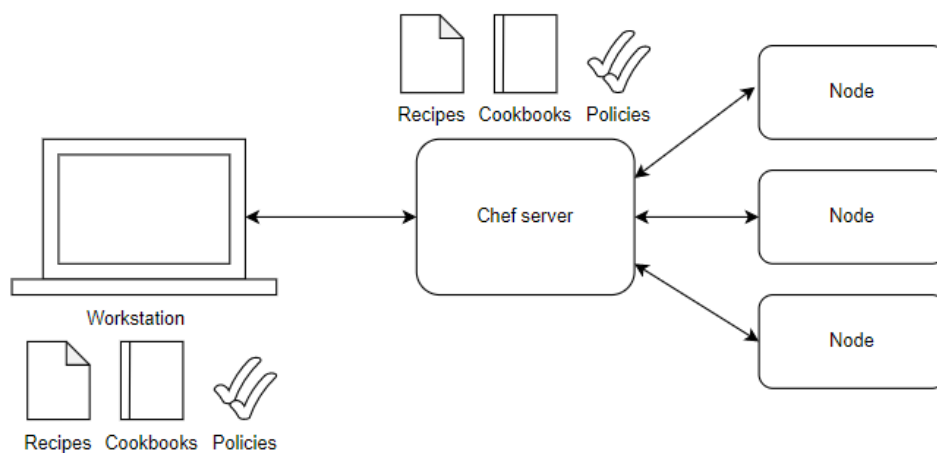
#### 2.1.1.3.1 Advantages

It runs on a wide variety of platforms, including Windows, and other Unix-based distributions, as well as many Cloud platforms. The tool is complex to manage and creating the configuration files (cookbooks) requires programming knowledge, but in turn allows handling complex tasks, which can be a great advantage over other similar tools that offer more limited functionality.

#### 2.1.1.3.2 Disadvantages

Chef runs in a master-client architecture similar to Puppet, thus needing a client to be installed in the controlled machines. It has also been mentioned that it makes use of a dedicated server where the cookbooks must be uploaded, so its architecture is more complex and the configuration files could get outdated more easily.

#### 2.1.1.3.3 Integration with the rest of the infrastructure



**Figure 7. Chef's basic infrastructure**

- Chef's official site: <https://www.chef.io/>



## 2.2 PSI 7.2: TECHNOLOGICAL ARCHITECTURE SELECTION

---

### 2.2.1 Selection of provisioning technology

After considering each of the three alternatives and how they would need to be included in the current architecture, **Ansible** was the tool chosen for automating the machine provisioning process. Though more limited than other alternatives, it offers all the needed functionality for carrying out the project, being its learning process shorter than for other tools. The lack of additional software has also been a determining factor when choosing Ansible, though its incompatibility to run on Windows hosts makes it compulsory to include an additional node in the infrastructure to act as a controller. This could be considered an advantage, as the node itself can be created following an Infrastructure as Code approach and distributed and reused easily where it is needed.

Ansible is also really simple to deploy, as it only requires SSH to connect to the target machines. Thanks to this, any machine that fulfils this condition can be intervened, so it is easy to adapt to different scenarios. However, in the case of Windows provisioning specifically, it must be noted that access to the controlled machine's PowerShell is typically required for running tasks, as well as full administrative privileges in many cases.

### 2.2.2 Machine provisioning using Ansible and SSH

#### 2.2.2.1 *Infrastructure considerations for running Ansible*

As it has already been mentioned, Ansible does not run on Windows hosts such as the machine used for development. Thus, an intermediate machine with all the necessary settings and code for executing Ansible, that is compatible with the tool, must be set up as part of the infrastructure. This machine will be considered a **controller** node, as it will be the one which will connect to all the other machines in the network that need to be provisioned and “control” them by means of Ansible.

Though it could be considered a drawback that Ansible does not operate on the Windows host, the approach proposed above can actually fit better into the overall architecture of the system and the purpose of the project, for a number of reasons.

First and foremost, the controller node can be created and managed as a piece of Infrastructure as Code. This means that it will be easier to reuse and adapt to different environments, serving for multiple scenarios no matter the host, apart from having all the benefits discussed in *PSI 3.2: Theoretical Concepts*.

Design-wise, having a separate machine for managing and running Ansible makes more sense than executing it from the given host. It will encapsulate everything Ansible-related, from roles and playbooks to the tool itself or any configuration files needed to manage the controlled nodes. In

conjunction with an IaC approach, the controller could be packaged and introduced in any environment that requires Ansible, with minimal changes.

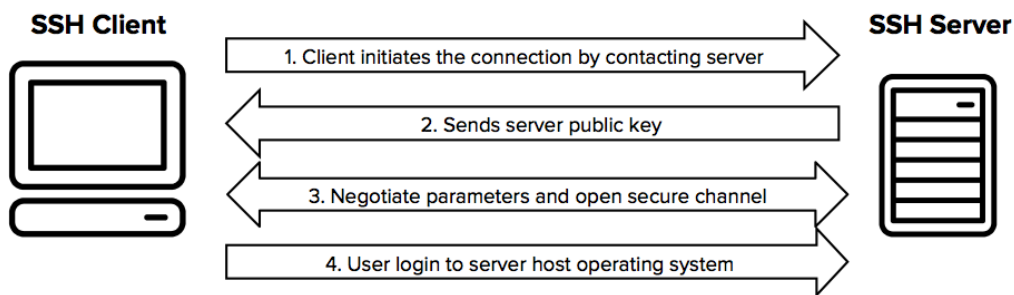
Finally, it adapts well to system restrictions, as no additional software must be installed in the controlling user's machine (the host) nor in the remote machines.

### 2.2.2.2 SSH

The Secure SHell protocol, commonly referred to as SSH, is a method for accessing remotely to servers or machines via a secure channel. SSH protects integrity and security of communications with robust encryption and offers various strong authentication mechanisms, so it can be used to establish secure connections over insecure networks. SSH was designed to provide a secure alternative to traditional, unprotected remote communications protocols like Telnet or FTP.

SSH uses a client-server architecture model, meaning that an SSH client is the first to establish the connection to an SSH server, which will make use of the protocol to accept the remote connection initiated by the client. One of the most common authentication methods, and the one typically used for automation, is public-key authentication, based on public-key cryptography. This mechanism uses a cryptographic pair of keys, a public and a private one; the public key will be placed on the server, to authorize and allow the owner of the corresponding private key to access this server.

Once the SSH client and the SSH server have established a connection, a negotiation between the two takes place, to agree on a series of communication parameters. Any transmitted data between the two will, from then on, be encrypted and protected using the encryption algorithm agreed on during the negotiation phase, like AES (Advanced Encryption Standard) or SHA-2 (Standard Hashing Algorithm), among others.



**Figure 8. SSH connection setup flow**

#### 2.2.2.2.1 How SSH Works for Ansible

Ansible only requires SSH to connect to the remote machines from the control node; in fact, it uses by default OpenSSH, a suite of utilities for secure networking based on SSH. Ansible will use this protocol to try to connect to the controlled nodes with the same username used for the controller; in case the user does not exist on any of them, another username can be specified. Another possibility Ansible offers, which might be the desired option if some tasks to be executed require root privileges



or other user's permissions, is privilege escalation; in fact, many Windows tasks require such user rights to be able to run and complete successfully.

Additionally, usage of SSH keys to connect to the remote systems is encouraged, as it eliminates the need for using password authentication; in fact, Ansible assumes by default that this is the authentication mechanism.

# CHAPTER 3: SYSTEM FEASIBILITY STUDY

**DEVELOPMENT PHASE**

**EVS**



The current approach in the University of Oviedo, as it has been mentioned before in section PSI 1.1: Analysis of the necessity of the PSI, is that machines are deployed as clones, so would be perfectly viable to install and harden the base machine in an automated way, and then clone it in every system where it needs to be replicated. Automated hardening of Cloud-based deployments of Linux and Windows machines is also common nowadays, and so in this section, technologies that serve both purposes will be discussed, closing the gap between the Cloud and a traditional on-premises infrastructure, while constructing a feasible and adequate system for the current needs, all through software.

## 3.1 EVS 4, 5 Y 6: STUDY AND VALUATION OF THE SOLUTION ALTERNATIVES AND SELECTION OF THE FINAL ALTERNATIVE

---

### 3.1.1 Alternatives for security standards on which to base the hardening

#### 3.1.1.1 CIS and CIS Benchmarks

##### 3.1.1.1.1 Description

CIS Benchmarks have already been described in the PSI 3.2: Theoretical Concepts section, but its advantages and disadvantages will be further explored below to make a more informed decision on why choosing these recommendations.

##### 3.1.1.1.2 Advantages

CIS Benchmarks are validated by experts and backed and used by well-known companies and the US Government. They are complete, meaning they cover all the important security aspects for the product they describe. Also, one of its most attractive characteristics is that they are fully explained, describing to the user what is wrong or should be remediated, how to fix it and why. They also explain possible consequences of applying said changes, so organizations can study whether the recommendation applies to their case or should be skipped.

Benchmarks are updated, maintaining a detailed list of changes from version to version. Errata are corrected quickly, and guides get adapted to new versions of the products they deal with.

They are international, meaning that if a machine complies with a certain security level according to the benchmarks, it could be introduced in environments that require certain official security certifications. In Spain's case, complying with some of the recommendations mean to also comply with the ENS. Related to this, CIS Benchmarks can be mapped to ISO 27001 controls, which are some of the most important internationally.

### 3.1.1.1.3 Disadvantages

CIS Benchmarks only cover part of the market's most used products. In the case of Windows 10 desktop, only benchmarks for the Enterprise versions are available. If the intention was to provision other kinds of Windows 10 workstations, security experts should consider that remediations shown might need to be adapted to fit the needs of the system, apart from the organization's.

Another thing that is important to be considered in Spain specifically is that they do not have direct correlation to the ENS, and some of the recommended values and configurations might clash with the ones demanded by the ENS. So, for achieving compliance with the ENS, certain compromises with which the organization will have to agree might need to be made.

One more drawback is that, depending on the product, they can be really extensive, so adapting them to specific functionalities or features can be tedious or require help from expert administrators.

Finally, official, complete tools for testing and automating the compliance with CIS Benchmarks are not free; the only free tool offers very limited functionality, and is only available for few OSs.

- CIS official site: <https://www.cisecurity.org/>
- CIS Benchmarks section: <https://www.cisecurity.org/cis-benchmarks/>

### 3.1.1.2 ENS and CCN-STIC Guides

#### 3.1.1.2.1 Description

As it was mentioned in the *PSI 3.2: Theoretical Concepts* section, the Spanish Centro Criptológico Nacional publishes free security guides, the CCN-STIC ones, that organizations can follow to implement controls for complying with the ENS. Organizations from the Spanish Public Administration field must comply with the ENS by law, and in those cases, it might be a priority to focus on studying these guides and applying its controls.

#### 3.1.1.2.2 Advantages

As it has been briefly mentioned, complying with the ENS is compulsory for many Spanish Public Administration entities, and Public Universities are within the application scope of the ENS. One of the long-term goals of the University of Oviedo is to be ENS-compliant and get certified, so using the CCN-STIC Guides could shorten the path towards directly achieving that goal.

#### 3.1.1.2.3 Disadvantages

The ENS is only of national application. Even though it may map to a security standard such as the ISO/IEC 27001, it is regulated by Spanish rules and decrees that adapt European legislation, so it is not valid for international scenarios a priori.

- ENS official site (in Spanish): <https://ens.ccn.cni.es/es/>





- CCN-STIC Guides: <https://www.ccn-cert.cni.es/guias/guias-series-ccn-stic/800-guia-esquema-nacional-de-seguridad.html>

### 3.1.1.3 Decision

For this project, **CIS Benchmarks** have been chosen as the best option, as it was already mentioned in section *PSI 2.1: Specification of the Scope and Reach of the PSI*. They have many attractive advantages, and though they can not be directly mapped to the ENS, they can be first related to the ISO/IEC 27001 controls. Another important remark is that, because they are international recommendations, hardened machines could better adapt to other environments apart from the national scenario.

## 3.1.2 Alternatives for Windows Operating System installed

### 3.1.2.1 Windows 10 Education

#### 3.1.2.1.1 Description

Windows 10 Education is a variant of Windows 10 Enterprise specially aimed at the educational sector, offering specific, pre-determined features that increase security while targeting basic necessities for schools, universities and other academic institutions. Certain services are disabled by default, like Cortana or Microsoft Store recommendations; and it comes with pre-installed software such as Microsoft Word or Microsoft Excel among others.

#### 3.1.2.1.2 Advantages

Some institutions like the University of Oviedo have special agreements with Microsoft that make licenses available for their software to their students, including several versions of their operating systems. Education versions are the ones available, both for Windows 10 and Windows 11, for free.

Also, as it has been seen in *Alternatives for security standards on which to base the hardening*, CIS Benchmarks specifically, are available, in the case of Windows 10 desktop, only for its Enterprise versions. As Windows 10 Education is based on Enterprise, many of the recommendations of the Benchmarks can still be applied directly to this system.

#### 3.1.2.1.3 Disadvantages

Windows 10 Education, though intended for students and educational institutions, is not the operating system that the University of Oviedo uses, so any security or compliance scores obtained by auditing the machines this project will deploy will not be completely accurate to the real scenario.



### 3.1.2.2 Windows 10 Pro

#### 3.1.2.2.1 Description

Windows 10 Pro has special features that are specifically suited for businesses and enterprises while retaining or expanding all the characteristics of the basic Windows 10 Home version, the most common distribution for casual users. One thing to consider is that a Pro license can be upgraded to an Enterprise one, as Windows Enterprise is intended to be a variant or improvement and not a proper, standalone version of the operating system.

#### 3.1.2.2.2 Advantages

Windows 10 Pro is the version of the operating system that University of Oviedo's computers have installed, at least the ones used by the regular machines. Using this OS would replicate the real scenario and allow getting more accurate data on how the hardening would translate to the University's systems.

#### 3.1.2.2.3 Disadvantages

No license is provided by the University of Oviedo, so a special license should be purchased if it is the version intended to be used. Official Windows 10 Pro licenses can cost up to 260€ for a single-use code, which will be wasted in simulating a system. Also, no specific CIS Benchmarks are published for the Windows 10 Pro versions, so policies should need careful study from expert administrators before implementation.

- Windows 10 Pro official section within the Microsoft Site: <https://www.microsoft.com/es-es/d/windows-10-pro/df77x4d43rkt?activetab=pivot:informaci%C3%B3ngeneral>

### 3.1.2.3 Decision

For this project, it has been decided that a **Windows 10 Education** ISO will be used for the Windows machines that will be deployed; to be more precise, the version used will be the latest one when the development of this project started, **Windows 10 Education Release 21H2**. The student carrying out the project can access the license for free, and given that it is specifically meant to be used in educational environments, it should contain all the programs needed to cover basic school-related needs, while allowing for installing whatever additional software each specific organization might need for its students. Also, as Windows 10 Education is based on Enterprise, even if there are no specific CIS Benchmarks for this variant, the policies described by said guidelines should still be valid for it; explicit mentions to Windows 10 Education systems are presents throughout the documents.



## 3.1.3 Alternatives for automating infrastructure deployment

### 3.1.3.1 Vagrant

#### 3.1.3.1.1 Description

Vagrant is a tool designed by HashiCorp for automating the management of virtual machines, specially those that serve as development environments, following an Infrastructure as Code approach. It is directed towards locally deploying small infrastructures with few machines, which can be easily created in special files called Vagrantfiles and configured (or provisioned) using provisioners.

Vagrant supports all the most used virtualization platforms, such as VirtualBox, VMWare, Hyper-V, Docker or even AWS. Regarding proper machine tuning, it can make use of a wide variety of provisioning tools like Ansible, Chef, Puppet or the standard shell scripts to install software and configure other settings.

The tool is written in Ruby and its files use this syntax, though it is not necessary to know the language to prepare configurations given its simplicity.

#### 3.1.3.1.2 Advantages

Vagrant is specifically designed and optimized for the automated deployment of few local machines, making it ideal for the objectives of this project. Its Infrastructure as Code approach and high-level properties allow adding new features and changing existing ones easily, all in an automated way. It also has straightforward integration with Ansible and other similar provisioning mechanisms.

As it has already been mentioned, Vagrant has an extensive catalogue of boxes, though it also supports custom-created boxes if no available machine adapts to your needs, which in this project is indispensable for the Windows machines.

Finally, even though it offers many advanced features for machine management, the basics are easy to grasp, so even non-advanced users can start deploying their machines quickly, and those who want to expand their knowledge can focus on what they really need.

#### 3.1.3.1.3 Disadvantages

Communication between machines that get deployed is not always straightforward. The easiest way to deploy machines in the same network capable of seeing each other is to place them in the same Vagrantfile, which can increase dependency.

- Vagrant's official site: <https://www.vagrantup.com/>



### 3.1.3.2 Terraform

#### 3.1.3.2.1 Description

Terraform is an Infrastructure as Code tool developed by HashiCorp, used for creating, managing, and deploying both basic and complex infrastructures of machines and services (servers, databases, firewall policies or even Kubernetes clusters), all automated. Its flexibility allows describing local or remote architectures, as big and complex as they might be, and it is intended for handling the initial deployment and any subsequent modifications done to the infrastructure along its lifetime. Terraform is extensively used in businesses to manage the cloud computing resources they may have with different providers, as it integrates seamlessly with AWS, Microsoft Azure or Google Cloud.

#### 3.1.3.2.2 Advantages

Terraform is a very powerful tool with which users can easily manage complex and extensive infrastructures describing them as code, which makes it easier to document and track changes and new features. It has a very detailed documentation, so new users can easily start using it for their infrastructures.

#### 3.1.3.2.3 Disadvantages

Even though it is possible to manage virtual machines with Terraform, it is very basic and offers a reduced set of features for VMs in comparison to other alternatives such as Vagrant. Additionally, it is not focused on deploying development environments, which in the case of this project is a major drawback.

- Terraform's official site: <https://www.terraform.io/>

### 3.1.3.3 Docker

#### 3.1.3.3.1 Description

Docker is a technology developed by Docker, Inc. mainly for creating applications' containers. Each container provides a low resource-consumption virtual environment, and includes a filesystem with everything the application needs to run, so each time it will be executed in the same way, no matter where. This mechanism is widely used nowadays for distributing applications.

#### 3.1.3.3.2 Advantages

Docker can mimic typical virtual machines while consuming significantly less resources and being much smaller, which is a significant benefit if the host does not have many resources to spend. Docker containers are also faster to get started than a regular machine to get booted.



### 3.1.3.3.3 Disadvantages

Docker has some limitations when it tries to act as a virtual machine. Because containers use the host's kernel, some operating systems can not be installed on top of another. The tool itself also has compatibility problems with some versions of OSs like Windows.

- Docker's official site: <https://www.docker.com/>

### 3.1.3.4 Virtualization software's CLI Tools

#### 3.1.3.4.1 Description

Many virtualization software products such as VirtualBox include command line tools with which the lifecycle of their machines can be managed. Though not platform-agnostic, many people still use these tools to develop their own automation mechanisms; even Vagrant makes use of these utilities for some of its internal operations.

#### 3.1.3.4.2 Advantages

In principle, no additional software is needed apart from the virtualization platform.

#### 3.1.3.4.3 Disadvantages

Using the virtualization software's specific tools means the code that is being written will be tied to a specific product entirely. Also, the version of the virtualization platform affects the command-line utilities, as they often change with new versions, so distributing workflows with other users might be impossible depending on the version each user has, or need manual tweaking in order to work. Finally, other IaC tools seen previously can achieve more functionality while keeping the automation consistent for every user.

### 3.1.3.5 Decision

**Vagrant** has been chosen as the software to automate the infrastructure deployment, mainly because of its advanced features, the support for different providers and its easy use and distribution of the necessary code for creating machines. In comparison to CLI tools, it clearly offers many more features while also integrating some of said tools in the core of its technology. It is also better suited for deploying local machines, offering many more capabilities in that regard compared to Terraform; in this project, only a few local machines will be deployed at once, so Vagrant is the best choice between the two. Regarding Docker, it has already been established that it presents some problems when trying to operate with some operating systems, so it was discarded almost from the start.

## 3.1.4 Alternatives for auditing tools

### 3.1.4.1 CLARA

#### 3.1.4.1.1 Description

CLARA is a tool developed by the CCN that allows performing analysis on the security features a system has. The compliance analysis CLARA performs is based on the security rules provided by several CCN's Guides's security templates. These security templates are applicable in a vast number of varied scenarios with different application scopes, so they, alongside with the security rules themselves, have been designed to define general security guidelines that ensure systems to be compliant with the minimum security requirements established by the ENS. However, each organization must consider that the predefined templates might need to get adapted to their operational needs.

#### 3.1.4.1.2 Advantages

First and foremost, CLARA is a free and very lightweight tool, and it does not need to be installed in the machine to be audited (though it needs administrator rights to be run). The reports it outputs state the ENS compliance score and all the values it checks with the actual and the expected results, so knowing what is not configured correctly according to these national regulations is generally fast to look up. The audit can also be adapted to the category of the system to be audited, so it will be better fitted to the environment where the system will operate.

Another advantage is that it is mainly focused on auditing Microsoft Windows systems, both desktop and server versions, so it supports almost all the latest Windows operating systems.

#### 3.1.4.1.3 Disadvantages

Some information present in the reports CLARA outputs might not be clear enough to know what to fix, specially because these reports are fully in Spanish and some values can be hard to find in the official documentation. Also, some of the results that, according to CLARA, are not correct, comply with the recommended configuration from other guidelines, such as the CIS Benchmarks. CLARA also does not seem to be totally updated for the latest Windows 10 releases, as the way to apply some of the remediations has been updated with these upgrades and this tool doesn't reflect the changes, while CIS Benchmarks do reflect them.

A huge drawback is that only few Linux systems are supported, being CentOS and RedHat the only distributions for which the tool is available.

- CLARA's official site: <https://www.ccn-cert.cni.es/soluciones-seguridad/clara.html>



### 3.1.4.2 Lynis

#### 3.1.4.2.1 Description

Currently maintained by independent software company CISOfy, Lynis is a well-known and mature security tool for systems that run Unix-based operating systems, including, but not limited to, Linux, macOS and Solaris. The tool performs extensive scans of the given system, and so it can support compliance testing and system hardening.

Lynis can serve multiple different purposes, given its flexibility: from simple security auditing to compliance testing and system hardening, but also penetration testing or vulnerability detection, among others.

When scanning, Lynis uses and tests only those components that it is able to find in the system, thus meaning that installing other tools is not necessary, and that Lynis can run with almost zero dependencies. This means that this tool will always perform analysis that are fitted to the system it is running on, while keeping it clean from other unnecessary tools. Additionally, the tests that get performed can be tuned, disabling them if they're too strict or even adding self-created ones.

An important thing that differences Lynis from other similar tools is that it will not automatically harden the machine, but rather perform in-depth security scans, without polluting the system and without risking breaking it. It will help users understand the level of security of their environment and discover any possible vulnerabilities, and will allow them to decide later what kind of security level is deemed appropriate for their system.

Lynis is currently free to use; the project is open source with GPL license and was made available in 2007. It has gathered a big community of users over the years, and it is now being used by different profiles, from individuals to multinationals, businesses and even government departments.

#### 3.1.4.2.2 Advantages

Lynis is one of the most renowned auditing tools in the market, at least for Unix-based OSs, as it uses a wide range of sources to perform tests, such as CIS Benchmarks, NIST, NSA, OpenSCAP or other guides and recommendations from various vendors. Scans are easy and fast to run and output a detailed report on every security configuration that is checked, informing on whether their value is correct according to the sources or not, as well as any security recommendations for further hardening the system.

The tool is available for many Unix-based distributions, covering, among others, Linux Ubuntu, which makes it an ideal tool for auditing at least some of the machines of this project.

#### 3.1.4.2.3 Disadvantages

The tool needs to be installed in the machine, which despite not needing additional software to be run, could be forbidden by the organization. It is also terminal-based, so users not accustomed to working without GUI could find Lynis's usage troublesome.



- Lynis's official site: <https://cisofy.com/lynis/>

### 3.1.4.3 CIS-CAT Lite

#### 3.1.4.3.1 Description

CIS-CAT Lite is an assessment tool developed by CIS, and the free version of the more advanced solution CIS-CAT Pro. It tests against their benchmarks and controls, scoring the system being audited with a compliance score that ranks from 0 to 100. This free version is very limited, with very few benchmarks available, though it allows testing different environment levels for each one. Otherwise, the results each audit produces have the same level of detail as with the CIS-CAT Pro version of the tool.

#### 3.1.4.3.2 Advantages

CIS-CAT Lite is specifically targeted to systems that are implementing CIS Benchmarks' recommendations and other CIS controls, so for the scope of this project is one of the tools that are better suited for the audit part. The reports each audit outputs are highly detailed, with all the different controls broken down and scored both globally and separately, and linked to their corresponding entry in the benchmark, which gets replicated in the report. Thus, locating which control passes and which fails and finding the recommended value is straightforward and quicker than resorting to the benchmark document itself.

#### 3.1.4.3.3 Disadvantages

As it has already been mentioned, the Lite version is very limited; it can only test compliance using the benchmarks for the last versions of Microsoft Windows 10, Ubuntu Linux and Google Chrome. For the scope of the project, the options it offers are only fit for testing the Windows machine. Another disadvantage is that it needs Java to be installed and configured to run, which will require to add that specific dependency to the Linux machine.

- CIS-CAT Lite official site: <https://learn.cisecurity.org/cis-cat-lite>

### 3.1.4.4 Decision

It was decided that all the three tools will be used, but each in the machine that suits them best. **Lynis** will be used for auditing the Ubuntu Linux machine, since it is specifically targeted to these systems, whereas Windows 10 Education will be audited with both the **CLARA** tool and the **CIS-CAT Lite** one.





## 3.2 STUDY OF ADDITIONAL TOOLS

---

### 3.2.1 Tools for automated installation and generation of Vagrant boxes: Packer

Once Vagrant was chosen as the final candidate for managing the infrastructure to be deployed in the project, a way to generate Vagrant boxes from freshly installed machines was investigated. Given the constraint that the result should be compatible with Vagrant, the decision to use **Packer**, a tool developed by HashiCorp as well, was almost immediate.

Packer allows creating custom boxes for Vagrant, in an automated way; it also integrates well with unattended installations. This tool uses a template in either Hashicorp Configuration Language (new versions) or plain JSON (old versions) to specify all the needed configuration for the image or images that will be built, including the ISO to use, any files that must be executed or used when the machine creation completes (such as unattended installation files), or the specific provider for which the box will be generated (VirtualBox, VMware, etc.).

# CHAPTER 4: PLANNING AND MANAGEMENT OF THE END OF DEGREE PROJECT

**DEVELOPMENT PHASE**



## 4.1 PROJECT PLANNING

### 4.1.1 Identification of Stakeholders

The stakeholders of this project have been already identified in previous sections (see *PSI 1.3: Delegation of Responsibilities* and *PSI 2.2: PSI Organization*), with their corresponding responsibilities during the lifecycle of said project, so they will not be shown in this section.

### 4.1.2 Initial Planning. WBS

One thing worth noting is that this project underwent some radical changes right before starting it that affected not only its scope but also its objectives. Thus, the project's starting date was a bit later than desired.

Also, the student was working full-time when the planning phase took place; as such, the planning was made with this information in mind. It was calculated that, on average, she would be able to dedicate to the project around 4 hours a day, seven days a week.

It must be noted that some tasks overlap, like the ones belonging to the development and the auditing phases. This has a reason: the student has tried to organize the audits and other testing-related activities constant along the project, so that continuous improvements to the hardening code specifically can be made.

In total, the project was thought to be starting mid-February and last 93 days, so it could end before the start of July. This meant that, if the average hours spent on the project a day were 4, it would last a total of 372 hours.

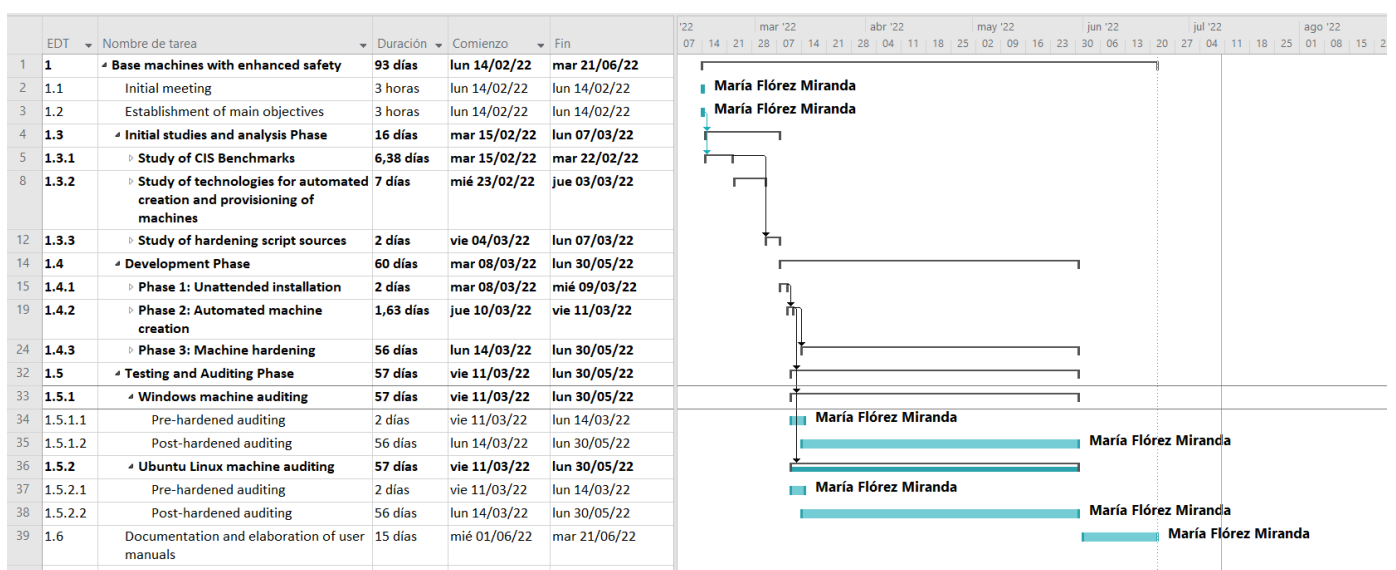


Figure 9. Gantt diagram for the Initial Planning

## 4.1.3 Risks

### 4.1.3.1 Risk Management Planning

The Risk Management Planning has been detailed in the Annexes, so it will not be shown here.

### 4.1.3.2 Identification of Risks

Despite the nature of the project, which makes it difficult to identify risks in a normal way, the following possible risks have been identified, as well as the corresponding category and response in accordance to the Risk Management Planning:

ID	Short description	Description	Category	Response	Probability
1	<b>Failure when running the hardening scripts for old Windows releases</b>	Some remediations may change or become invalid from one version of the operating system to another, as Windows specifically has made some changes to the naming of internal registry keys and other settings. This may make the remediations unable to run or fail	External, Technical	MITIGATE, by trying to keep the systems updated, which is actually a recommended security mechanism	Low
2	<b>No possibility of automating the installation of a given OS</b>	There may exist the possibility of being unable to automate the installation of a given OS, because it does not provide the mechanism to do so	Technical	ASSUME the risk, there may be no possibility other than to install the OS by hand	Low
3	<b>Lack of specialized knowledge of the user trying to execute the code</b>	If the user does not have enough technical knowledge on the system, he or she may apply some recommendations	External, Technical	TRANSFER the risk by finding someone who has the necessary technical knowledge to be	Medium



		that restrict access to critical services		able to discern what recommendations can be applied and what can not. Or MITIGATE it by instructing the user on what recommendations could potentially break the system	
4	<b>Lack of consensus between the organization and the user applying the remediations</b>	It is extremely important, specially for some high security recommendations, that both the users that apply said recommendations and the organizations are totally aligned. If not, critical services may become unable to operate	External, Organizational	MITIGATE the risk by specifically indicating what recommendations need approval from different departments	Medium
5	<b>Changes in CIS standards' recommendations</b>	Because CIS is constantly trying to adapt to new features and fixing any obsolete value or errata on their benchmarks, some recommendations that have already been adapted into code might have to be changed to reflect those modifications	External, Technical	MITIGATE the risk by staying ahead of possible new releases of both the products and the CIS Benchmarks, and checking the latter's registry of changes	Low in the case of products that have reached their end of life, Medium in the case of products that are still getting releases



6	<b>Collision between CIS and ENS standards</b>	Some recommendations from CIS directly clash or may directly clash with what other standards such as the ENS establish	External	MITIGATE the risk by reaching consensus with the organization on what security standard has more priority to follow	Medium
7	<b>Provider-specific code changes from version to version of the virtualization provider</b>	Some virtualization providers, like VirtualBox, change their internal functions from version to version, so settings configured in a determined way using their internal tools may fail in the future if another version of the provider is used, such as when creating new boxes.	External, Technical	MITIGATE the risk by staying alert of possible new releases of the software, as well as all the changes introduced in new versions of the product	Low
8	<b>No possibility of deploying the Ansible controller on the same network as the machine to be hardened</b>	Some networks may be very restricted and deploying a new machine on the network could not be possible	External, organizational, technical	ASSUME the risk by contacting the organization beforehand, explaining the conditions under which the provisioner must run and reaching a solution	Low
9	<b>No administrative privileges on the machine to be provisioned</b>	The user trying to connect to the machine to be provisioned does not have administrative privileges, so tasks	External, Technical	MITIGATE the risk by contacting the organization or the system's administrator and reaching an agreement on	Medium



			may fail or running them could directly be impossible		what user to use or whether to grant them special privileges for the amount of time the provisioning process lasts	
10	<b>Services that allow access to the machine are restricted</b>	<b>that remote the are</b>	It could be possible that the machine intended to be hardened restricts some type of communication service, such as Windows with WinRM. In this case, it would be impossible to connect to the machine remotely to provision it	External, Technical	MITIGATE the risk by contacting the organization or the system’s administrator and reaching an agreement to temporarily enable services that allow remote management of the corresponding machine	Medium

**Table 2. List of identified risks**

It must be noted that, though these risks are related to the project and could originate from it, some of them fail under the user and the organization’s responsibility, so it should be their task to consider how to respond to these risks happening.

#### 4.1.4 Initial Budget

First things first, this project has been planned with the intention of having almost zero cost regarding materials and licenses. This budget intends to show how much the project would cost in a real scenario, if all assets were quantified in regard to pricing, from personnel to licenses.

Because of the nature of this project, only the internal budget has been considered; there is no actual real client, and because of that, there is a lot of information lacking to make a representative budget.

#### 4.1.4.1 Internal Budget

In this section, only a summary of the internal budget will be shown; the full budget will be present in the Annexes.

Budget Summary		
Item	Description	Total cost
1	Personnel	13.000,00 €
2	Licenses	260,00 €
3	Material Resources	7,81 €
4	Indirect Costs	2.625,00 €
TOTAL		15.892,81 €

**Table 3. Summary of the initial internal budget**

## 4.2 PROJECT CLOSURE

### 4.2.1 Final Planning

The final planning significantly diverted from the original one. First and foremost, the project uses many new technologies that required investigation and a constant process of trial and error. Because of that, some phases ended up taking much more time than expected, such as the unattended installation. On the other hand, some phases of the project were started in parallel, in contrast with what had been planned: it was the case of the implementation of hardening tasks, since at the same time the Cyber Ansible code was being checked and adapted into the project's scripts, new recommendations were being added simultaneously, to complement them.

However, in the end the project managed to last until the same day planned beforehand, effectively amounting to the same number of hours, though some tasks had their planned starting or ending date shifted.



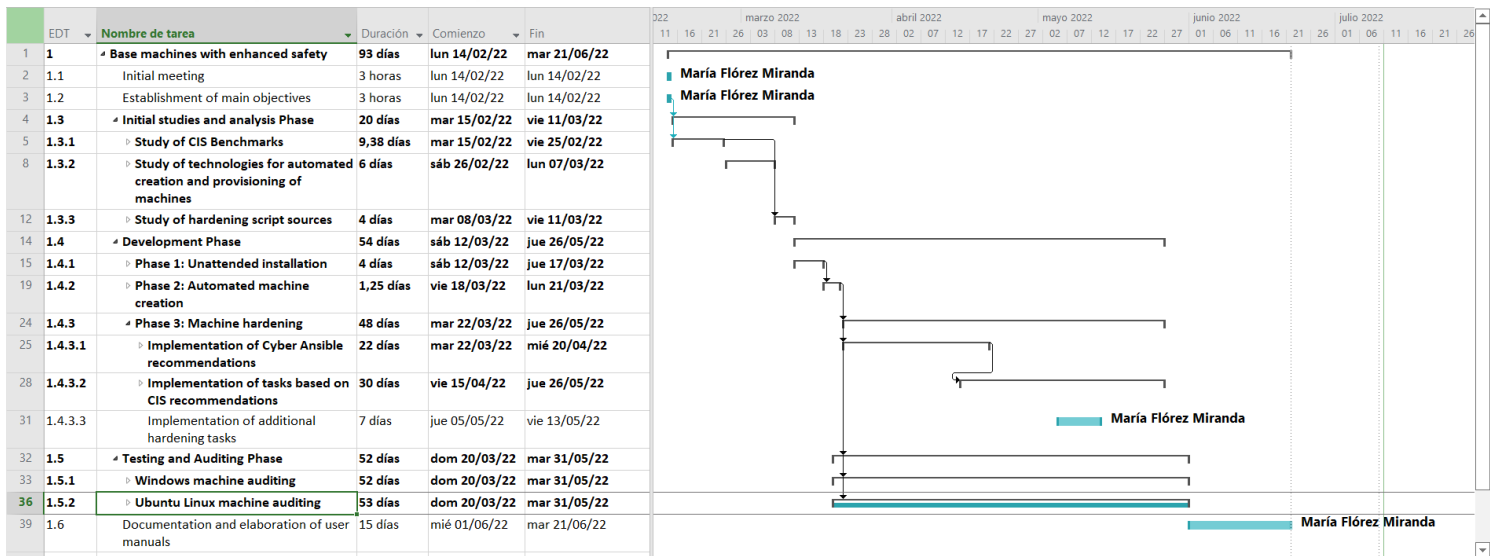


Figure 10. Gantt Diagram for the Final Planning

## 4.2.2 Final Budget

### 4.2.2.1 Final Internal Budget

The final, real budget of the project turned out to be less than expected, as no license for the Windows machine had to be purchased. Otherwise, the rest of the items were the same as the initial in the initial budget:

Initial Budget Summary		
Item	Description	Total cost
1	Personnel	13.000,00 €
2	Licenses	0,00 €
3	Material Resources	7,81 €
4	Indirect Costs	2.625,00 €
TOTAL		15.632,81 €

Table 4. Summary of the final internal budget

### 4.2.3 Learnt Lessons

It was found that organizing a project as big as this, with so many new technologies, can be very challenging. It is difficult to estimate some tasks which make use of knowledge the student does not even have prior to them starting. In this project's case, the unattended installation of a Windows 10 machine, one of the core parts, turned out to be one of the most difficult to fulfil, and as such the whole planning had to be moved to adapt.



# CHAPTER 5: ANALYSIS OF THE INFORMATION SYSTEM

**DEVELOPMENT PHASE**

**ASI**



The first thing that needs to be addressed beforehand is that this is not a typical software development project. Not only because it does not intend to develop a proper application or software module, but infrastructure; but also, because some of the phases have required carefully examining security guidelines that have already been tested and validated. For all these reasons, some of the sections a usual memory would include have been left out, reformulated, or merged to better adapt to the project it documents.

This analysis of the information system thus includes specific sections for discussing the security guidelines studied for making sure the machines this project deploys are hardened following the standards, as well explaining the different sources from where hardening scripts are taken from and how all these sources intend to be integrated.

## 5.1 ASI 1: SYSTEM DEFINITION

---

### 5.1.1 Determination of the System's Scope

As it has already been explained in previous sections, this project will study and provide a way to solve three main problems: automating the installation of operating systems, automating the creation of (preferably secured) machines, and improving the security of existing infrastructure using validated sources. Though addressing all three issues can translate into deploying real-life architecture, this project must be considered as a first iteration, with a reduced scope that has allowed testing all these intermediate objectives but that could be extended and implemented in a real infrastructure in the future.

First and foremost, the main intention of the project is to be run on the local machine of the user. The machines that will be developed, and afterwards, hardened, will be deployed as virtual machines with the user's computer acting as their host, so in principle anyone interested must have the full code in their machines. However, apart from the project as a whole being Infrastructure as Code, and therefore, of easy distribution, even in parts, both base machines and their secured counterparts will be provided, as well as the original scripts used to export them, so anyone can replicate the process and obtain their own hardened machines.

Another important thing to mention is that hardening real machines is out of the scope of the project: it will be limited to the virtualized environments. Existing infrastructure, that is, the University's computers, will be relegated to auditing, to test their current security status so that they can later be compared to the secured machines. This decision has also been influenced by the need of the organization to check first what configurations and services are being changed. Nevertheless, future plans on applying the results of this project on real machines have already been considered (see *Extensions*).

It is also worth noting that:

- Regarding **machine creation and deployment automation**, Vagrant supports a number of virtualization technologies, but only machines for VirtualBox will be generated and tested. This virtualization software is freely available, offers rich functionality and runs on many different operating systems, which is why it has become one of the most popular and used products of this kind; the other alternatives (Docker, Hyper-V, etc.) all present drawbacks and are even incompatible with certain machines.
- In the case of the **unattended installation**, it will be limited to Windows 10 Education, as it has been mentioned in section *1.2.1PSI 2.1: Specification of the Scope and Reach of the PSI*, and further discussed in *Alternatives for Windows Operating System installed*.
- For the **hardening process**, the scope will be limited to applying as many controls of the CIS Benchmarks' guidelines as possible, and as long as they make sense, particularly for Level 1. However, there exists the intention of further exploring both the CIS Benchmarks themselves and other sets of national and international standards. More information on this can be seen in *Extensions*.

## 5.2 STUDY OF AUTOMATED MACHINE CREATION AND UNATTENDED INSTALLATIONS

---

Infrastructure creation with Vagrant is done by making use of Vagrantfiles, special files in which developers include all the needed configurations for deploying said infrastructure. Probably the most important elements that must be specified in these files are boxes: they represent packaged Vagrant environments and, as such, will indicate what machine or machines should be brought up. The most common way of working with Vagrant is by choosing from a catalogue an already existing box, particularly base ones, as they include only the bare minimum to function; however, it could be possible that none of them adapt to the infrastructure's needs. In those cases, the best approach is to create a custom box.

As it was found when investigating how Vagrant works and the possibilities for deploying Windows machines, their public box catalogue does not contain any Windows 10 official box, in contrast with other operating systems such as Unix-based ones. Using a third-party, user-developed Windows environment was considered as a possibility for this project, but it was soon discarded in favour of creating a custom box for the following reasons:

1. Installing and creating a custom box allows the student to configure a base machine tailored to the needs of this particular project, including what version of the operating system to use
2. Using a third-party box implies trusting how the author has configured the machine and what software has he or she installed on it. For a project focused on education needs, it can be dangerous security-wise



3. Most boxes are old and have outdated Windows 10 versions installed, as opposed to creating a custom box with the latest available version of the operating system. This helps reduce security issues
4. The Windows machine is intended to replicate as much as possible the real environment, or at least model an educational scenario. In this regard, no Windows 10 Education boxes were found in the catalogue

Further investigation also showed that, using a special, Vagrant-related tool called Packer (see *Tools for automated installation and generation of Vagrant boxes: Packer*), it was possible to combine the steps of performing an automated, unattended installation of Windows, and creating a custom base box by exporting the machine resulted from the installation. This not only allows addressing some of the issues at once, but will also ensure that the environment used for hardening really represents the common scenario where the machine is left with as little security mechanisms as it came with when installed.

## 5.3 ANALYSIS OF THE AVAILABLE BENCHMARKS AND SECURITY GOOD PRACTICES

---

For understanding what security measures should be implemented, a study on the guidelines that will be used as a reference needs to be carried out. This section is dedicated to analyzing how the benchmarks are organized and what they contain to be able to make an informed decision on the recommendations that can or should be applied.

### 5.3.1 CIS Microsoft Windows 10 Enterprise Benchmark, v1.12.0

The CIS Microsoft Windows 10 Enterprise Benchmark, version 1.12.0, was the most updated version of this specific guide at the beginning of this project's development. The benchmark is based on the latest Windows 10 Enterprise release, 21H2, and though the configurations were tested against this specific version of the operating system, it is intended to work for all Windows 10 versions, both current and older.

#### 5.3.1.1 Profiles

The Benchmark was found to include, apart from the default configuration profiles for security levels Level 1 and Level 2, two additional profiles that extend the original ones. Thus, the four main, distinguishable security profiles, that can then be combined, are as follow:

- **Level 1 (L1) - Corporate/Enterprise Environment (general use):** the most basic profile, intended to act as the starting point for the majority of organizations. In principle, the

recommendations included as part of this first level provide clear security benefits while maintaining the functionality of the system or systems as much as possible.

- **Level 2 (L2) - High Security/Sensitive Data Environment (limited functionality):** this second profile is intended to extend Level 1, since the recommendations included are designed to be applied in environments where a high level of security must be accomplished, even at the expense of sacrificing services' functionality; this means that, for complying with L2, L1 settings need to be applied as well. Implementation of any of the recommendations should first be considered carefully and in conjunction with the organization, as performance and utility of resources can be negatively affected, and access to them, restricted.
- **BitLocker (BL):** BitLocker is a volume encryption tool that Microsoft offers for both the Professional and Enterprise versions of the Windows operating system; this software allows encrypting and thus protecting hard drives from data theft. Thus, the BitLocker-related controls pertaining to this profile are intended to be applied if the organization has this tool deployed.
- **Next Generation Windows Security (NG):** this profile, as hinted by its name, contains recommendations for environments where the newest configuration and hardware are available, and so includes advanced security features that could be incompatible with some systems. Because of this, implementation requires careful testing beforehand. Nevertheless, these controls are strongly advised to be applied if possible, because of its impact on the overall security of the systems.

### 5.3.1.2 Sections

This guide is divided into 19 main sections grouping recommendations that refer to the same set of policies; the schema is directly based on how Windows arranges the equivalent security policies. Because of this, some sections do not include any content, appearing just to keep the CIS document's structure consistent with Windows's policy organization.

For the reader's ease, the following table roughly summarizes all the different sections of the document, with a brief description of their overall purpose.

Section number	Name	Description
1	Account Policies	Recommendations for account policies related to passwords and account lockouts
2	Local Policies	Recommendations for local policies, related to assignment of user rights and configuration of local security options for accounts and various services
3	Event Log	Blank; included to keep consistency
4	Restricted Groups	Blank; included to keep consistency

5	System Services	Recommendations for configuring different system services, (Xbox-related, Bluetooth-related, Remote Management-related, etc.)
6	Registry	Blank; included to keep consistency
7	File System	Blank; included to keep consistency
8	Wired Network (IEEE 802.3) Policies	Blank; included to keep consistency
9	Windows Defender Firewall with Advanced Security	Recommendations for configuring the default Windows Firewall for the three offered profiles (Domain, Private, Public)
10	Network List Manager Policies	Blank; included to keep consistency
11	Wireless Network (IEEE 802.11 Policies)	Blank; included to keep consistency
12	Public Key Policies	Blank; included to keep consistency
13	Software Restriction Policies	Blank; included to keep consistency
14	Network Access Protection NAP Client Configuration	Blank; included to keep consistency
15	Application Control Policies	Blank; included to keep consistency
16	IP Security Policies	Blank; included to keep consistency
17	Advanced Audit Policy Configurations	Recommendations for configuring audit-related policies that allow monitoring different types of events (Logon/Logoff, account management, changes to policies, system events...)
18	Administrative Templates (Computer)	Recommendations for Group Policy Administrative Templates, system-based
19	Administrative Templates (User)	Recommendations for Group Policy Administrative Templates, user-based

**Table 5. CIS Microsoft Windows 10 Enterprise Benchmark's sections' summary**

#### 5.3.1.2.1 Section 1: Account Policies

This first section includes some basic security recommendations for password and account lockout policies; in fact, all controls are Level 1, so their implementation is encouraged, preferably at domain level.

Password Policy recommendations deal with essential settings such as the minimum password length or its maximum age, and though most of them are already set up for domain members, standalone workstations lack almost all recommended values for their local accounts. On the other hand,



Account Lockout Policy recommendations deal with settings that affect what happens to user accounts if they get blocked; neither domain members nor standalone workstations have these values configured by default, leaving systems vulnerable to Denial of Service and brute-force attacks.

#### 5.3.1.2.2 Section 2: Local Policies

The second section of the document includes recommendations related to Local Policies; these Local Policies include three main sections, being the most critical the User Rights Assignment and the Security Options. The third one relates to a general Audit Policy that, even from official Microsoft documentation, is advised to be left unused in favour of other audit policies (see *Section 17: Advanced Audit Policy Configurations*).

User Rights Assignment's recommendations deal with features that control what users or groups have certain privileges and/or rights on a device, whereas the Security Options ones help configure varied specific security settings. Some of the recommendations for both subsections may need to be considered or parameterized to adjust to the organization's security policies, or even discussed with its legal/human resources representatives.

#### 5.3.1.2.3 Section 5: System services

This section includes recommendations for various system services, such as Internet Connection Sharing (which allows turning a computer into an Internet router), OpenSSH SSH Server, or Windows Remote Management (WinRM). Many of the recommendations included in this section are Level 2, as they involve restricting or disabling services that could be crucial to the system's or the organization's normal operation. A perfect example is WinRM, which is needed for remotely managing the machine via Ansible, among other uses.

#### 5.3.1.2.4 Section 9: Windows Defender Firewall with Advanced Security

This ninth section contains recommendations to configure Windows's default firewall, for three different profiles: Domain, Private and Public. Each of these profiles corresponds to a specific network scenario: Domain is for networks where the host authenticates to domain controllers, whereas Private and Public profiles are applied to private and public networks respectively. This last profile is the default one. All recommendations for the three profiles are Level 1.

Basic firewall configuration, such as whether it is enabled or not, are already set by default with the recommended values by CIS; however, some more advanced settings, which could be considered as good practices, would need to be specifically set to comply with the recommendation.

#### 5.3.1.2.5 Section 17: Advanced Audit Policy Configurations

This section of the document comprises recommendations for configuring audit-related policies that allow and manage the logging of events. The settings discussed in this section provide better and more specific control over the audit process, in comparison to other Audit policies (such as the ones under Local Policies).





The recommendations present here could be subject to the organization's requirements, security policies or legal obligations.

#### 5.3.1.2.6 Section 18: Administrative Templates (Computer)

This section of the guide includes recommendations based on the Group Policy Administrative Templates (ADMX), specifically for the computer side. This part of the document contains a large number of items with different security configuration profiles and a wide variety of application scopes; as such, special attention has been put into choosing the settings that will be considered for implementation based on the recommendations.

#### 5.3.1.2.7 Section 19: Administrative Templates (User)

The final section of the document includes, too, recommendations based on the Group Policy Administrative Templates (ADMX), but for users; however, it includes significantly less items than *Section 18: Administrative Templates (Computer)*.

### 5.3.1.3 Recommendations that will not be implemented

While this could be considered as part of the definition of the project's scope because it limits its reach, choosing the recommendations that will and will not be implemented could only be possible after carefully examining the Benchmark in this analysis phase. From this study, the following decisions were taken.

First, only those recommendations marked as "Automated" will be implemented, as they can be set via Ansible, and so "Manual" controls will be omitted. It does not make sense in a project whose intention is to automate the hardening process as much as possible to consider configuration settings that can only be set manually, as there is no reproducible code to be provided to future users of the system.

Additionally, only some of the security policies from Level 2 will be implemented, as long as they contain security measures that could be useful to the organization while not impeding remote management (due to Vagrant and Ansible) or heavily affect the system's functionalities. However, applying the remediations for this specific high security environment will be left to the user, with some control mechanism to let them enable these features or not.

Regarding the additional configuration profiles, policies specific to them will be omitted as well. On the one hand, BitLocker-related recommendations are intended for organizations that choose to use this volume encryption software, so consensus with the business side should be reached first. On the other hand, the Next Generation Windows Security profile is intended to be used in environments with advanced features and so its implementation should also be thoroughly tested and discussed with the organization. In any case, the guide clearly specifies that these two profiles are to be considered optional add-ons to the default levels, and therefore should have less priority.

Finally, some of the policies that do not fail under any of the categories mentioned in this section (meaning that they should be considered for implementation) will not be fully applied or not applied at all; some of them require expert administrative knowledge, while others depend on the organization where they will be implemented, for legal, technical, or business reasons.

### 5.3.2 CIS Ubuntu Linux 18.04 LTS Benchmark, v2.1.0

The CIS Ubuntu Linux 18.04 LTS Benchmark, version 2.1.0, was the latest available version of this guide when this project started.

#### 5.3.2.1 Profiles

The guide incorporates profiles for 2 different types of machines (servers and workstations) and the 2 common security levels (Level 1 and Level 2). As what happened with the Windows Benchmark's profiles, items considered of Level 1 are of basic application, providing substantial security benefits while keeping services and general activity of machines (either servers or workstations) as unaffected as possible. On the other hand, Level 2 recommendations should be applied carefully and are only really recommended for high-security environments, as they could seriously alter normal operation of the machine whose type is affected by this security level.

#### 5.3.2.2 Sections

The benchmark document is divided into 6 main sections, each for a main configuration area that should be covered. They can be seen in the following table:

Section number	Name	Description
1	Initial Setup	Initial configurations applicable to all systems
2	Services	Recommendations for disabling or restricting services
3	Network Configuration	Recommendations related to the network configuration and how to set it up securely
4	Logging and auditing	Recommendations for enabling logging and auditing on the system
5	Access, Authentication and Authorization	Recommendations related to configuring access, authentication and authorization features (SSH, passwords...)
6	System Maintenance	Maintenance recommendations

**Table 6. CIS Ubuntu Linux 18.04 LTS Benchmark sections' summary**



#### 5.3.2.2.1 Section 1: Initial Setup

In the first section, recommendations for first configuring the system are listed, as they are easier being performed during the initial setup and become harder to implement once the system has been in use. In general, CIS advises that all systems should apply these security items.

The section guides users to safely perform basic system configuration, such as configuring the filesystem, setting up software updates or enabling secure boot settings.

#### 5.3.2.2.2 Section 2: Services

In this second section, CIS provides a list of services that can be disabled or deleted safely without disrupting normal operation of the system in most of the cases, as well as some services that should have special security configuration. The recommendations should be carefully examined individually to really ensure that modifying the settings of these services or removing them from the system does not affect it in any way.

#### 5.3.2.2.3 Section 3: Network Configuration

The third section includes recommendations for securely setting up the network configuration of the system. The section serves as a guide for disabling uncommon or unused network protocols and devices, enabling network parameters with recommended values, and correctly configuring the system's firewall (UncomplicatedFirewall, nftables or iptables).

#### 5.3.2.2.4 Section 4: Logging and auditing

This section contains recommendations for configuring logging services, both rsyslog and journald, which are intended to be used together. Obviously, these recommendations only apply if the mentioned services are installed or meant to be installed, otherwise they can be omitted. However, it is recommended to have some sort of advanced, properly configured logging service.

#### 5.3.2.2.5 Section 5: Access, Authentication and Authorization

This fifth section includes recommendations for everything related to access, authentication and authorization processes, including SSH Server and password-related configurations, as well as setting up sudo and time-based job schedulers.

#### 5.3.2.2.6 Section 6: System Maintenance

This section contains special maintenance recommendations that are expected to be checked regularly. However, though many of them include automated ways to perform the audits, the remediations are not quick, being in some cases partially manual, partially automated, and sometimes requiring the system administrators or maintainers to analyze the output of the audit processes to take the appropriate measures to fix any discrepancies in each situation.



### 5.3.2.3 Recommendations that will not be implemented

First, as this benchmark is significantly simpler than the Windows one, as many recommendations as possible will be applied, if they make sense for the kind of machine being hardened.

However, as with Windows security items, only recommendations that can be audited and remediated in an automated way, from beginning to end, will be implemented. Also, any Level 2 recommendation will be studied, to check whether they affect any critical system functionality or, on the contrary, could suppose a clear security benefit; if the case is the latter, then they will be applied, as long as their full automation is possible.

Finally, System Maintenance recommendations will not be considered. From careful examination of the remediation process for each task, it can be deduced that they are to be performed by regular maintainers of the system, such as system administrators, as they are the ones which will have a better insight of the infrastructure; the tasks are also intended to be performed over time, and not only during the first setup of the machine. Additionally, some of them may require special knowledge of the environment or even require adapting remediations to site policies.

## 5.3.3 General good security practices

Apart from all the recommendations contained in the benchmarks, there are a series of good security practices that system administrators and other security experts can consider important or even critical for certain organizations. They have been discussed beforehand with some security advisors and will be implemented and tested on the Windows machine.

### 5.3.3.1 Windows PowerShell

Standard, non-administrator users should not be able to access PowerShell to its full capability for command or script execution, or at least be denied elevation requests. As such, an approach for fully disabling PowerShell automatically for the Users group will be implemented.

It is important to note that only two PowerShell-specific recommendations appear in the Windows 10 Education Benchmark, one for logging commands, functions or scripts invoked through this shell, and another for transcribing their input and output. No recommendation referencing to directly blocking non-administrator users from performing certain operations was found in the guides.

### 5.3.3.2 Change the Vagrant user password

The default, well-known default Vagrant user's password should be changed by a more robust one, that should also comply with the password policy in place. This change should not affect the SSH connection between the Ansible controller or the host and the machine, as for these communications a public-key authentication mechanism will be in place.



### 5.3.3.3 *Manage local administrators*

In some organizations' machines, it could be interesting to manage what users should be included or kept out of the local administrators' group, to avoid having additional users with administrative privileges. A way to manage the list of local administrators of the system being hardened will be provided.

## 5.4 ANALYSIS OF HARDENING SCRIPT SOURCES

---

### 5.4.1 Cyber Ansible

The project being developed is not the first dealing with automated system hardening based on international security guidelines; in fact, Carlos Lacasa's *Cyber Ansible* partially tackles this issue, providing Ansible playbooks for Windows and Linux systems that implement many cybersecurity-related tasks. Among them, there can be found roles that group together security recommendations based on different international standards and sources.

Careful examination of these scripts is vital in this phase of the project, as it is third-party code.

#### 5.4.1.1 *Compliance scripts for Windows systems*

For Windows systems, security configurations based on recommendations from CIS, GSA (the US General Services Administration), ACSC (the Australian Cyber Security Center) and STIG (Security Technical Implementation Guide) can be found.

Cyber Ansible's Windows compliance scripts make use of some Ansible modules specifically designed for manipulating security policies and system registry values. In addition to that, configuration values for some of the implemented recommendations have been externalized to a variables' file, so that they can be easily modified if needed without having to touch the proper Ansible code.

Finally, Cyber Ansible can be used to provision either English or Spanish-based machines, as it loads the appropriate configuration files depending on the system locale.

#### 5.4.1.2 *Compliance scripts for Linux systems*

In the case of Linux, Cyber Ansible explicitly provides hardening scripts only for OpenSCAP, SOX and STIG-based recommendations. However, as it happens with Windows, many of the recommendations can be reused with minor configuration changes to be CIS-compatible, as they modify the same settings.



## 5.5 INTEGRATION BETWEEN HARDENING SOURCES

---

After analyzing the Cyber Ansible scripts, it could be concluded that some of them could be reused, though determined configuration values would need to be changed and parameterized to adapt to the ones recommended by CIS Benchmarks. So, this project's hardening phase will have several additional chores for integrating the different sources of knowledge:

- Examine all Cyber Ansible compliance scripts, for all sources, and identify the tasks and/or files that can be adapted to comply with CIS (or that directly comply with CIS)
- Reorganize said tasks and/or files into a CIS Benchmarks-based directory structure, for easier access to them when following the guides. Identify them with their CIS identifiers
- Modify any non-CIS-compliant setting inherited from Cyber Ansible to adjust to the version of the benchmarks being used
- Add support for missing recommendations, adapting the remediations that were agreed to be implemented into Ansible tasks
- Include in-code explanations for those tasks that could be particularly critical or that need to be configured differently than recommended by CIS Benchmarks for continued Vagrant or Ansible management
- Provide explicit support for Windows 10 Education machines, as they are not considered in Cyber Ansible
- Parameterize all variables that do not have a fixed, recommended value, but rather a range of possible values or even values that should be discussed with the organization beforehand (e.g. warning messages, log file sizes or password-related configurations). Keep support for different languages
- Make all Level 2 implemented tasks (at least for Windows) conditional, so that their execution can be activated or deactivated depending on the environment where the hardening will take place

Additionally, some basic security configurations (PowerShell access control, local administrators' configuration or system updates) will be implemented. Separate code for these settings will be created, to clearly distinguish between them and CIS-specific tasks.

## 5.6 ASI 2: ESTABLISHMENT OF REQUIREMENTS

---

### 5.6.1 Acquisition of the System's Requirements

As this project's main goal is to automate as much as possible the whole machine's installation-creation-hardening process, little to no manual intervention of the user is required. As such, the



requirements that have been acquired are mainly non-functional ones, with some exceptions for each of the specific phases.

### 5.6.1.1 Functional requirements

#### Automated Machine Creation (AMC-FR)

- AMC-FR-1. The system must deploy machines for the following operating systems:
  - AMC-FR-1.1. Ubuntu Linux
  - AMC-FR-1.2. Windows 10
- AMC-FR-2. The machines must be deployed using VirtualBox as the virtualization platform

#### Automated Machine Hardening (AMH-FR)

- AMH-FR-1. The system must be able to deploy a controller machine that connects via Ansible to the following base machines:
  - AMH-FR-1.1. Ubuntu Linux
  - AMH-FR-1.2. Windows 10

### 5.6.1.2 Non-functional requirements

The following groups of non-functional requirements were identified:

#### Compliance (COMP-NFR)

- COMP-NFR-1. The new code developed for hardening based on the CIS Benchmarks must comply with CIS recommendations
  - COMP-NFR-1.1. If a recommended setting impedes communication between the machine being provisioned and the machine executing Ansible:
    - COMP-NFR-1.1.1. The provided value will be such that it does not interfere with the machines' communication, even if it does not comply with CIS benchmarks
    - COMP-NFR-1.1.2. The following information must be provided to the user:
      - COMP-NFR-1.1.2.1. An explanation about why the CIS recommendation has not been followed
      - COMP-NFR-1.1.2.2. An indication of what the CIS-recommended value is
- COMP-NFR-2. The third-party code used for hardening must comply with CIS recommendations
  - COMP-NFR-2.1. The code must be verified to check the settings comply with the ones specified in the CIS Benchmarks
  - COMP-NFR-2.2. If a setting is not CIS-compliant, it must be modified to comply with the corresponding recommendation, unless COMP-NFR-2.3 happens



- COMP-NFR-2.3. If a recommended setting impedes communication between the machine being provisioned and the machine executing Ansible, the following information must be provided to the user:
  - COMP-NFR-2.3.1. An explanation about why the CIS recommendation has not been followed
  - COMP-NFR-2.3.2. An indication of what the CIS-recommended value is

### **Open Source (OPSRC-NFR)**

- OPSRC-NFR-1. The hardening policies' third-party code used in the project must be Open Source

### **Reusability (REU-NFR)**

- REU-NFR-1. The code used for hardening must be independent from the machine that will execute said code, allowing it to be reused in different applications

### **Extensibility (EXT-NFR)**

- EXT-NFR.1. The code used for hardening must be developed in such a way that allows adding new tasks without having to modify existing ones

### **Flexibility (FLEX-NFR)**

- FLEX-NFR-1. The code used for hardening must be developed in such a way that supports adapting to new requirements
- FLEX-NFR-2. CIS recommendations' values that are not fixed in the guides must be parameterized by means of variables with descriptive names
  - FLEX-NFR-2.1. The variable must be put in an external configuration file
  - FLEX-NFR-2.2. The variable must be set with a default value that complies with the specific CIS recommendation

### **Security (SEC-NFR)**

- SEC-NFR-1. Communications between machines must be secured
  - SEC-NFR-1.1. The information shared between the Ansible controller and the controlled machines must travel through an SSH encrypted channel
    - SEC-NFR-1.1.1. The SSH authentication method must use public-key encryption

### **Internationalization and Localization (INT-NFR)**

- INT-NFR-1. The hardening code developed for Windows must support the following Operating System Locales:
  - INT-NFR-1.1. Spanish
  - INT-NFR-1.2. English





- INT-NFR-2. The hardening code developed for Windows must allow supporting new Operating System Locales

## 5.7 SPECIFICATION OF THE TESTING AND AUDITING PLAN

---

As it has already been established at the beginning of this chapter, this project is not a regular development one. Because of this, the testing plan is also different from what could be expected from such projects.

First of all, the initial phases of the project that involve the automated machine creation and unattended installation can not really be tested, apart from checking the machines are set up following the indicated configuration. The tools used for this part of the project have not been developed by the student and already incorporate mechanisms to protect from incorrect settings; configuring either invalid values or values not expected by the tools or the system being configured will prevent the machines from getting set up or even booting.

The hardening phase also has some constraints. CIS Benchmarks have been carefully crafted and are constructed following an audit-remediate approach: this means that for each security control they consider, they include the way to check the current control's value (how to audit it), and how to configure it to comply with what CIS recommends (the remediation). Because of this, tests are embedded in the deployment itself. It is also important to remember that these guides and all their controls have been devised, tested and validated by internationally-recognized organizations; if the values this project configured for the security settings of the machines didn't adjust to the recommendations, then they wouldn't be complying with the benchmarks. As such, it does not make sense planning and creating tests that attempt to set non-complying configurations, because their correct or recommended values have already been established.

However, this does not mean there is no way of checking that the recommendations are in place and affect the overall security of the machines. For this purpose, audits using the tools specified in section *EVS 4, 5 y 6: Study and Valuation of the Solution Alternatives and Selection of the Final Alternative* will be carried out. Additionally, careful monitoring of the results of the executed Ansible tasks will take place, to ensure that any failure gets detected and corrected.

### 5.7.1 Task monitoring

All runs of Ansible tasks will be monitored and the logs, exported to text files to detect any possible failure. Failed tasks will be examined and remediated if possible; those that fail will be recorded and an alternative solution, if possible, will be proposed.



## 5.7.2 Auditing Plan

Auditing will take place throughout most of the duration of the project, and will be specially focused on the virtualized environments being deployed by the student. However, some audits over real infrastructure will be carried out as well, as was previously mentioned in *PSI 3.1: Study of the current situation*. The plan followed in both cases will be detailed below.

### 5.7.2.1 Virtualized environments' auditing

#### 5.7.2.1.1 Windows 10 Education machines

The Windows virtual machines will be audited pre-hardening for checking the default security score against CIS Benchmarks and ENS. From this point on, as security controls get applied, new intermediate audits will be carried out, though only the final reports for the completely hardened machines will be taken into consideration for contrasting their scores against the initial ones.

The tools that will be used for auditing will be CLARA and CIS-CAT Lite, already explained in *Alternatives for auditing tools*:

- **CLARA**, which checks compliance with the ENS, allows selecting the category to which the system being audited belongs. After checking the list of public entities that have already been certified as ENS-compliant, it was found that the universities which appear on the list have information systems that either belong to the BASIC or MEDIUM categories; however, this project's Windows machines will also be tested for the HIGH category. This will ensure having the full picture in case the machine was to be introduced in a high security environment, or if the developed Ansible code needs to provision systems in such environments.
- **CIS-CAT Lite**, specific for checking compliance against CIS-related controls, allows running audits based on the CIS Microsoft Windows 10 Enterprise Benchmark v1.12.0, the same version taken as reference for applying the recommendations, for both Levels 1 and 2. The main focus of the audits will be placed on compliance for Level 1, but they will take place for the two security levels, as some L2 recommendations will be implemented.

Thus, the full list of scenarios that will be audited for the Windows machine is as follows:

1. Pre-hardened Windows 10 Education machine:
  - a. CLARA tool
    - i. System category "Low"
    - ii. System category "Medium"
    - iii. System category "High"
  - b. CIS-CAT tool
    - i. Level 1 (L1) - Corporate/Enterprise Environment (general use)
    - ii. Level 2 (L2) - High Security/Sensitive Data Environment (limited functionality)
2. Hardened Windows 10 Education machine:
  - a. CLARA tool



- i. System category “Low”
- ii. System category “Medium”
- iii. System category “High”
- b. CIS-CAT tool
  - i. Level 1 (L1) - Corporate/Enterprise Environment (general use)
  - ii. Level 2 (L2) - High Security/Sensitive Data Environment (limited functionality)

#### 5.7.2.1.2 Ubuntu Linux machines

As with the Windows machines, the Ubuntu ones will be audited pre- and post-hardening. In this case, the tool for auditing will be Lynis (described on *Alternatives for auditing tools*), whose score is partially based on CIS controls. No specific security levels or environment needs to be set for the audit, so only the following scenarios will be considered:

1. Pre-hardened Ubuntu Linux machine, using the Lynis tool
2. Hardened Ubuntu Linux machine, using the Lynis tool

#### 5.7.2.2 On-premises auditing

With the permission of the School of Computer Engineering’s scholars, the ones responsible for managing some of the computer equipment and software, it was agreed that some audits will take place to test the security of the machines used in the 2021-2022 academic year. Additionally, the base image that will be cloned onto the machines for the next academic year, 2022-2023, will also be audited. The scenarios planned for each system’s audits are the same described for the virtual Windows machine (see *Virtualized environments’ auditing*), though for obvious reasons these machines will not be hardened.

### 5.7.3 Tests to be manually performed

Finally, there do exist some parts of the hardening process for the Windows 10 Education machine that can be tested following a more conventional approach, though the process will need to remain manual, that is, by modifying configuration values and/or testing the behaviour by hand. However, this should be the way to check the result of applying these security settings, since they intend to be implemented adapting good security practices and, by themselves, do not appear in the CIS Benchmarks.

All tests pertaining to the default machine user refer to the user set up as default during the installation process, which will be the same with which Ansible will try to connect to said Windows 10 machine.



### 5.7.3.1 PowerShell access restriction

This test will check the restriction mechanism devised for controlling access of non-administrators to the Windows PowerShell.

PowerShell Access Restriction (Windows 10)	
Test	Expected Result
Indicate PowerShell needs to be disabled (restricted access)	Non-admin users will not be able to access PowerShell through any of the executables

### 5.7.3.2 Configure local administrators

This test will check the results of executing the task for keeping only indicated users as administrators of the local system, with emphasis on the default machine user.

Configure local administrators (Windows 10)	
Test	Expected Result
Take out default machine user from being Local Administrator	The default machine user will be taken out of the 'Administrators' group and will lose administrative privileges

### 5.7.3.3 Local administrators' password change

This test will check the results of executing the task for changing the passwords for the local administrators of the system, with emphasis on the default machine user.

Local administrators' password change (Windows 10)	
Test	Expected Result
Change password for the default machine user	The password of the default machine user will be changed to the new one



# CHAPTER 6: DESIGN OF THE INFORMATION SYSTEM

**DEVELOPMENT PHASE**

**DSI**



An important remark that must be made beforehand is that this project mainly deploys scripts that have been validated and verified against international standards in an organized and controlled way; as such, some diagrams that would apply to a regular Object-Oriented Programming project, such as Classes Diagrams, will not be shown in this chapter. Additionally, certain sections have been omitted and/or reformulated to better explain the design process of this project.

## 6.1 DESIGN OF THE INFRASTRUCTURE DEPLOYMENT PROCESS

---

This section will explain all the steps needed for deploying the final infrastructure. Though they individually fulfil the planned intermediate objectives, and the resulting product or products of each of the steps could be valuable on its/their own, the final goal of the project can only be achieved following them in the order presented below. One thing worth noting is that this order doesn't adjust exactly to the phases explained in section *PSI 2.1: Specification of the Scope and Reach of the PSI*; this is due to the relative independence of the intermediate goals.

### 6.1.1 Steps

#### 6.1.1.1 Step 1: Automated Unattended Installation

One thing that needs to be clarified is that, though the project is indeed automating the installation of a Windows machine, it is doing it through Packer, whose final purpose is creating virtual machine images. It could be argued that this does not equate automating a real installation; however, the process would be very similar, if not easier, on a real machine, as Packer adds some specific extra steps for building the final images. For this part, several items must be prepared beforehand.

As a basic step, Packer is an executable, not a software that has to be installed in the machine. As such, the executable will be placed in the root directory where all the rest of the files referenced below will also be present.

Additionally, an ISO of the operating system to be installed is needed, preferably one belonging to the user as the retail key needs to be known, as well as its checksum. In the case of this project, an officially licensed Windows 10 Education, release 21H2 ISO will be the one used.

Packer templates are also required, as they are the ones containing the configuration for building the boxes for different providers; as it was decided in an earlier analysis phase, only data for creating VirtualBox-specific Vagrant boxes will be considered. These templates can be in either HCL or JSON format, but the latter will be the chosen one as it is easier to read and manipulate: in the end, it will be a collection of key-value pairs identifying the specific setting and its value.

Finally, as the intention is to perform a totally unattended installation, special XML files (Autounattend.xml) for filling the installation form will be mandatory. These files must then be

referenced in the corresponding template to be made available right when the machine first boots; the Windows install will automatically look for and load them if they exist. Special attention needs to be put into the `Autounattend.xml`, as it will be where the machine locale, the locale of the user input, the product key or the default user will be specified.

After everything mentioned above is correctly configured, Packer will be executed and the machine will start being built automatically, requiring no manual intervention; the process will be monitored to check if any problem arises. If it succeeds, a box artifact will be outputted, with the machine installed and set up as requested. This artifact will then be imported into Vagrant's local cache of boxes, so it becomes available to be used with that tool in the next steps.

### 6.1.1.2 Step 2: Automated machines' deployment

Machines will be created and deployed with a basic first configuration via code, using Vagrantfiles for the main infrastructure. Additionally, the needed auditing tools will be copied to or installed into the corresponding machine, so that they become available without requiring the user to set them up manually.

The infrastructure, detailed in *DSI 5: Design of the Architecture of the System Modules*, will deploy in the student's computer using VirtualBox and Vagrant to manage the machines, and will vary depending on whether the specific machine is intended to be hardened or not, since in this last case an extra node acting as an Ansible controller needs to be set up as well.

### 6.1.1.3 Step 3: Machine hardening

Machine hardening, though it could be contemplated as an independent process, is in this project closely tied to the previous step, as the infrastructure will be fully virtual and managed via Vagrant and Vagrantfiles. Two machines will be deployed in each case: the base machine to be hardened, which will be either the Windows 10 Education or the Ubuntu Linux 18.04 one, and an Ansible controller, in charge of provisioning the base machine with the hardening Ansible scripts. Both machines will be connected to the same network, so that the controller can access via SSH to the controlled node. As Vagrant presents some problems when trying to make two machines deployed using two different Vagrantfiles communicate, machines to be hardened will be created in the same file as the Ansible controller, and both will be placed in the same internal network to be able to see each other. The deployed infrastructure is detailed in *DSI 5: Design of the Architecture of the System Modules*.

The Ansible controller, which will be built as a plain Ubuntu machine, will have all the needed code (playbooks and/or roles) copied to it, as well as a static inventory in which the machines to be provisioned will be specified. Ansible will then be installed onto the controller node automatically using dedicated Vagrant provisioners, and the hardening code will be run to provision whatever machines have been specified in the inventory. Once the tasks finish running, the resulting hardened environments can be packaged so that they can be deployed wherever they are needed.



## 6.1.2 Hardening code to be deployed

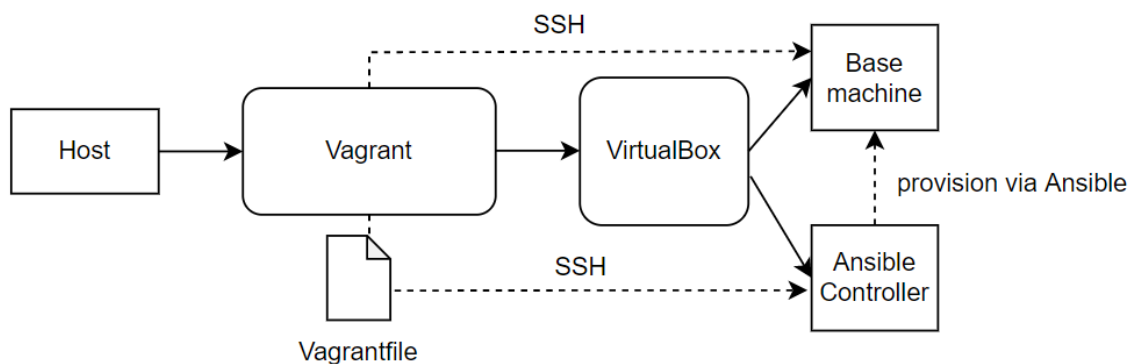
The hardening code will be designed and then implemented following the guidelines specified in *Integration between Hardening Sources* and the limitations found in section *Analysis of the Available Benchmarks and Security Good Practices*. It will include Cyber Ansible, CIS-compliant tasks, as well as code specifically developed for adapting CIS recommendations; all this code will have its “audit” section from the benchmarks checked to ensure they can be detected in automated audits. Additionally, some non-CIS related, hardening tasks will be implemented.

A diagram showing the exact way in which the hardening scripts will be organized can be seen in *Provisioners*; CIS-related tasks belonging to the same section according to the benchmarks will be grouped in the same file, while being clearly separated from other hardening tasks.

## 6.2 DSI 5: DESIGN OF THE ARCHITECTURE OF THE SYSTEM MODULES

### 6.2.1 Deployment Diagrams

The deployment of the full structure, considering Vagrant as the manager for the Virtual Machines and VirtualBox as the virtualization provider, would be like this:



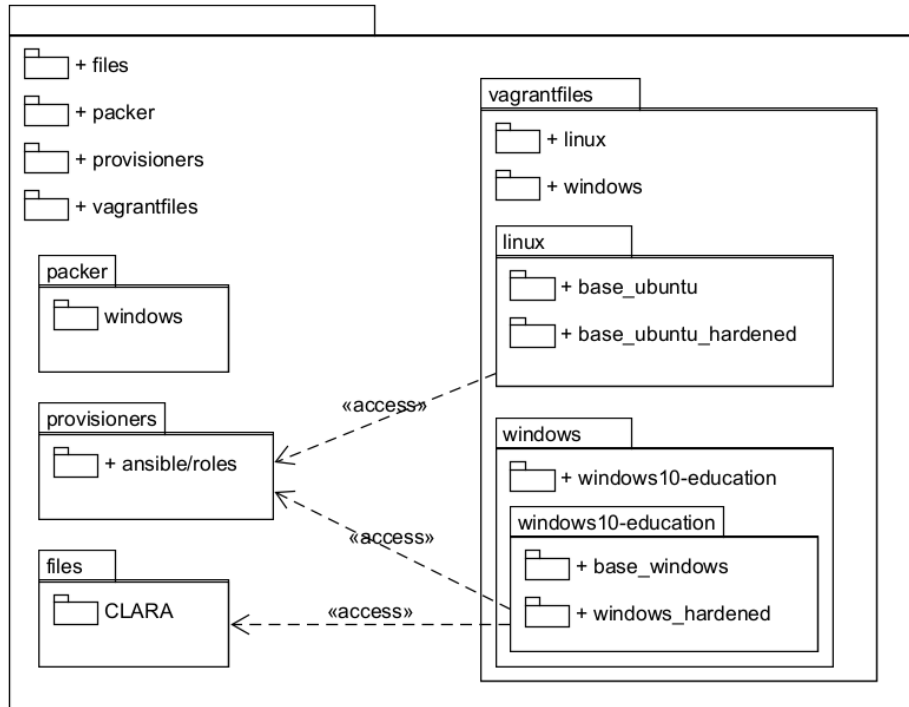
**Figure 11. Simplified architecture to be deployed**

### 6.2.2 Package Diagrams

The following diagrams illustrate how the project is structured physically in the different directories, to help understand how the scripts and other additional files have been distributed. The most important directories have been included as independent diagrams as well to better show its internal structure.

### 6.2.2.1 Full directory structure

This diagram illustrates the whole physical package architecture of the project and how the different packages are related. It must be noted that the directory packer is designed to be included in the architecture because it will contain all the necessary code (minus the packer executable itself) for building the custom Windows 10 box, so that any user will be able to replicate the process.



**Figure 12. Package diagram of the whole directory structure (simplified)**

### 6.2.2.2 Provisioners

This diagram shows how all the shared provisioning code has been arranged. The structure has been deliberately designed to allow extension, making it possible to add different provisioners without having to touch any existing code, or implement new Ansible roles for either supported or new types of machines.

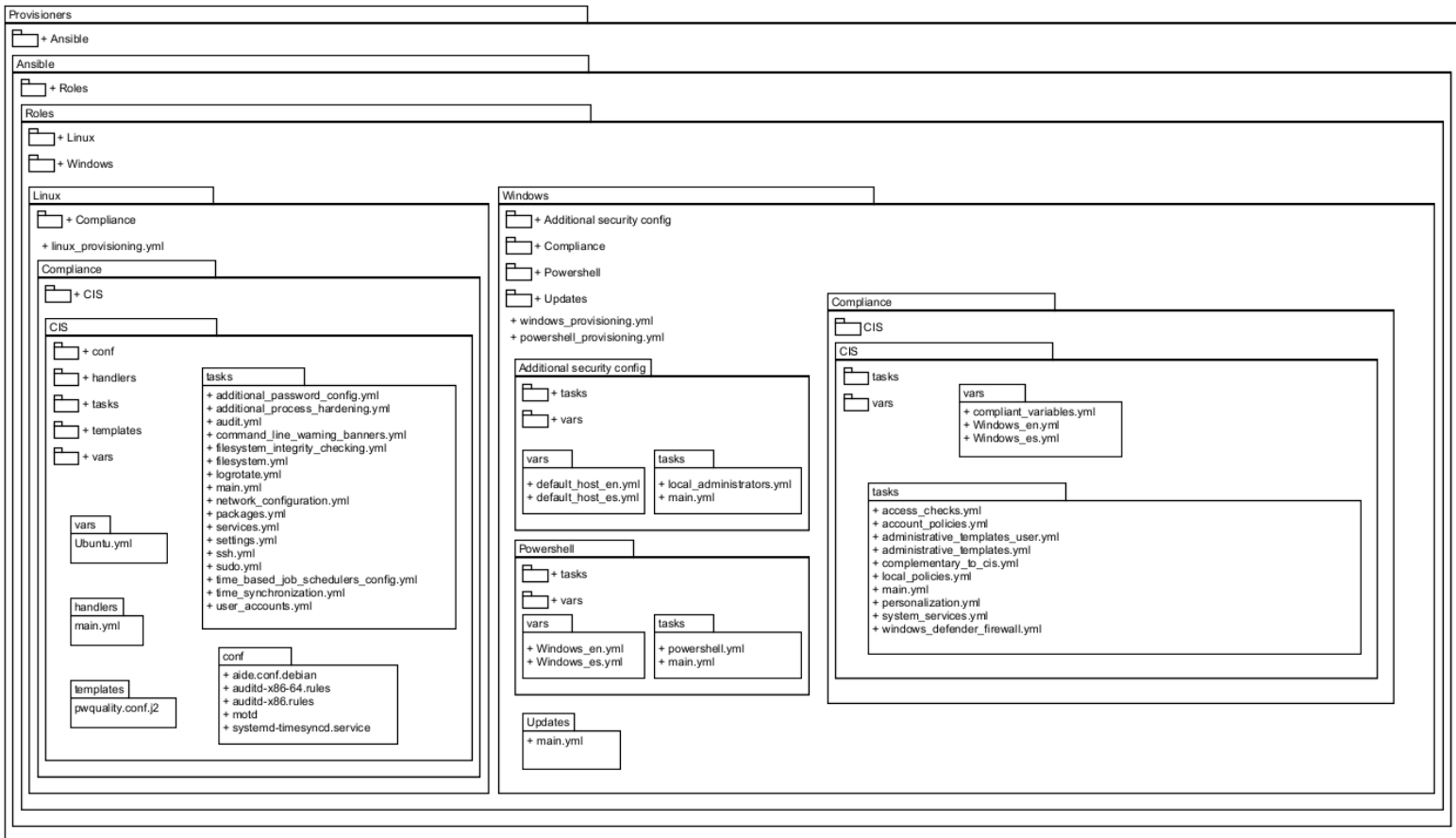


Figure 13. Package diagram for the Provisioners' packages

## 6.3 TECHNICAL SPECIFICATION OF THE TESTING AND AUDITING PLAN

The testing and auditing will be carried out as described in section *Specification of the Testing and Auditing Plan*. As it was already commented, monitoring and auditing will take place during the whole hardening phase, using all three auditing tools (CLARA, CIS-CAT Lite, Lynis), and scores will be recorded and compared from one execution to another to ensure that tasks improve the machines' security according to the standards.

# CHAPTER 7: CONSTRUCTION OF THE INFORMATION SYSTEM

**DEVELOPMENT PHASE**

**CSI**



Because of the nature of this project, the construction phase had to be adapted as well, to fit the objectives of the project. Some sections have been adapted or fused, so they can better reflect the work done.

## 7.1 CSI 1: PREPARATION OF THE GENERATION AND CONSTRUCTION ENVIRONMENT

---

### 7.1.1 Followed standards and regulations

The purpose of this project made it compulsory to follow CIS recommendations from CIS Benchmarks, which are in themselves internationally recognized security standards. Additionally, the ENS has been taken into consideration for performing audits and adding some additional hardening policies.

### 7.1.2 Programming Languages

In the case of Ansible, even though it is written in Python and its code will be transformed to this language on the remote machines to execute the tasks, it uses its own DSL encoded in YAML files. So, all the code developed specifically for Ansible has been written in this DSL.

Regarding Vagrant, it is written in Ruby, an interpreted, general-purpose, multi-paradigm language; because of this, it has been possible to include some slightly more advanced Ruby code in certain Vagrantfiles to externalize basic configuration settings to a YAML file whose data can then be loaded, instead of having the values directly hardcoded in the specific Vagrantfile. The rest of the code for managing Vagrant keeps its Ruby syntax, but is fairly simple and mainly uses variable assignment.

Additionally, shell scripts for Linux and batch files for Windows have been written. These files, containing system-specific commands, were used to add some additional configuration to the base machines, as well as for making it easier for administrators and others interested in the project to execute the necessary commands to deploy the infrastructure, since neither Vagrant nor Packer have a graphical user interface.

Finally, though less important, Autounattend files for the Windows machine's unattended installation have been modified and extended using XML.



## 7.1.3 Tools and programs used for development and auditing

### 7.1.3.1 Visual Studio Code v.1.68.1

Visual Studio Code is a flexible and lightweight code editor, compatible with a large number of programming languages. It includes syntax highlighting for many of them, as well as debugging, code refactoring and intelligent code completion, though these features are not available for the full catalogue of supported languages. However, its base capabilities can be extended through both official and user-developed plugins.

Visual Studio Code, in its latest available version to the date of this document, has been the chosen IDE for developing the code of this project. Thanks to its versatility and custom extensions for Vagrant, YAML and Ansible, it has allowed for easily editing Ansible playbooks, Vagrantfiles, shell scripts and other configuration files.

Website: <https://code.visualstudio.com/>

### 7.1.3.2 Vagrant 2.2.19

Vagrant was used throughout the entire project for managing all the virtual machines that were deployed. A more detailed explanation on this tool can be found in section *Alternatives for automating infrastructure deployment*.

Website: <https://www.vagrantup.com/>

### 7.1.3.3 Packer 1.8.2

Packer was used to generate the Windows custom boxes that could later be used with Vagrant. More information on this tool can be found in *Study of Additional Tools*.

Website: <https://www.packer.io/>

### 7.1.3.4 Ansible

Ansible, as it was explained in [previous sections], was used for machine provisioning, more specifically, for the hardening process. It was not manually set up in the controller nodes, as Vagrant supports straightforward integration with this technology and its Ansible provisioners detect if the tool is present in the machine, automatically trying to install it otherwise. Thus, the version used in this project is the one Vagrant considers appropriate (typically the latest one) each time the controller boots up.

Website: <https://www.ansible.com/>



### 7.1.3.5 Lynis

Lynis was the tool chosen for auditing the security level of the Ubuntu Linux machines. More information on this utility can be found in *Alternatives for auditing tools*.

Website: <https://cisofy.com/lynis/>

### 7.1.3.6 CLARA, v2.0

CLARA was one of the auditing tools used for testing the security level of the Windows machines throughout the project, both the virtualized environments and the real University's machines. A more detailed explanation on it can be found in *Alternatives for auditing tools*.

Website: <https://www.ccn-cert.cni.es/soluciones-seguridad/clara.html>

### 7.1.3.7 CIS-CAT Lite, v4.18.0

CIS-CAT Lite was the second tool for auditing the security level of the Windows machines; for a more detailed explanation on the utility and what it can audit against, see *Alternatives for auditing tools*.

For downloading this tool, users must first register in the CIS site. Thus, no direct link to the downloads page has been provided.

Website: <https://learn.cisecurity.org/cis-cat-lite>

## 7.1.4 Operating Systems

### 7.1.4.1 Windows 10 Education, Release 21H2

As discussed in section *Alternatives for Windows Operating System installed*, this was the specific version of the Windows 10 operating system used for creating the Windows 10 base Vagrant boxes and subsequently deploying the machines.

### 7.1.4.2 Ubuntu Linux 18.04 LTS (64-bit)

Ubuntu Linux 18.04 LTS was the chosen version of this specific Linux distribution. Though not the latest Ubuntu version (it was released in 2018), it is a stable release that still gets support. Not only that, but HashiCorp, the organization behind Vagrant, has published a special Ubuntu 18.04 64-bit base box, very basic but that can be extended if needed; because of its small size and high optimization, it is ideal for quick deployments.

Website: <https://releases.ubuntu.com/18.04/>

Link to the specific Vagrant box: <https://app.vagrantup.com/hashicorp/boxes/bionic64>



## 7.1.5 Libraries used and consulted for analysis and development

### 7.1.5.1 *Cyber Ansible*

Cyber Ansible is a project maintained by Carlos Lacasa that compiles Ansible playbooks related to Cybersecurity, for both Windows and Linux machines. It includes tasks for configuring general protective measures, as well as hardening rules based on recommendations by reliable international sources like GSA, ACSC, OpenSCAP or CIS.

Cyber Ansible has been extremely useful for having a basic understanding of how to implement CIS policies for the two kinds of machines that were dealt with. Compliance tasks have been carefully examined; those that could be reused have been included in this project, adapting them to be CIS-compliant. Nevertheless, it has been a good starting point for writing new tasks that security recommendations for both Windows and Linux.

It is currently available in GitHub.

Author: Carlos Lacasa (GitHub: <https://github.com/carloslacasa>)

Link to repository: <https://github.com/carloslacasa/cyber-ansible>

### 7.1.5.2 *SSI Infrastructure Automation Materials*

This repository, crafted by José Manuel Redondo López, teacher in the Information Systems Security course from the Software Engineering degree of the University of Oviedo, and director of this project, contains several different scripts and materials for the automated deployment of varied infrastructure. It includes materials for several distributions, that can be deployed using Docker or Vagrant. One of the machines present in this repository, “Magic the Hardening”, which is an already hardened Ubuntu instance, has been reused in this project for testing to which extent its security could be further increased. This project has been extremely useful material for learning how to write more complex Vagrantfiles.

It is currently available in GitHub, and the project still gets regular updates.

Author: José Manuel Redondo López (GitHub: <https://github.com/jose-r-lopez>)

Link to repository: [https://github.com/jose-r-lopez/SSI Infrastructure Automation Materials](https://github.com/jose-r-lopez/SSI-Infrastructure-Automation-Materials)

### 7.1.5.3 *Packer Windows*

Packer Windows is a project that compiles several Windows templates fit for creating Vagrant boxes via Packer. It includes templates for Windows Desktop and Server boxes, and for several virtualization platforms, including VirtualBox.





This project is currently available in GitHub, though it has been archived by its owner and is read-only. However, the templates can still be used, tuning them to adapt to the user's or developers' needs.

Author: Joe Fitzgerald (GitHub: <https://github.com/joefitzgerald>)

Link to repository: <https://github.com/joefitzgerald/packer-windows>

#### 7.1.5.4 Windows 10 Vagrant Box

Similarly to the Packer Windows project mentioned above, it is a repository that includes Windows templates for generating custom Windows boxes that can then be used with Vagrant. However, unlike Packer Windows, only templates and resources for Windows 10 are available, though its detailed documentation and scripts were useful to generate the custom box that ended up being used in this project.

It is currently available in GitHub, though the project does not seem to be updated. However, as in the case of Packer Windows, the templates and files can still be used, provided they are tuned to the user's needs.

Author: Jeff Skinner Box (GitHub: <https://github.com/jeffskinnerbox>)

Link to repository: <https://github.com/jeffskinnerbox/Windows-10-Vagrant-Box>

### 7.1.6 Official tutorials and learning materials

As this project mainly uses technology that is not taught in the Software Engineering degree or is just briefly mentioned, several official tutorials for these technologies, as well as documentation from the Master's degree on Web Engineering, were used to better understand the different concepts to handle.

#### 7.1.6.1 Vagrant tutorials for getting started

The official Vagrant site offers tutorials for helping users grasp the first, basic concepts of Vagrant. They provide a step-by-step tutorial for creating a user's first development environment, including networking and provisioning, as well as additional tutorials for creating custom boxes with Packer and working with different virtualization providers.

Vagrant's First Steps: A tutorial: <https://learn.hashicorp.com/vagrant>



### 7.1.6.2 *Introducción a Ansible (Introduction to Ansible) presentation, from the Administration of Web Servers course, Master's Degree in Web Engineering (University of Oviedo)*

This presentation has been provided by the author himself, who has allowed it to be consulted and referenced as support material.

Author: José Manuel Redondo López

### 7.1.6.3 *Topic 4: Security Policies and Automated Hardening presentation, from the Information Systems Security course, Software Engineering Degree (University of Oviedo)*

This presentation has been provided by the author himself, who has allowed it to be consulted and referenced as support material.

Author: José Manuel Redondo López

## 7.2 INFRASTRUCTURE CONSTRUCTION

---

All the machines used in this project have been developed via code following the structure and specifications designed in *Chapter 6: Design of the Information System*.

## 7.3 IMPLEMENTED HARDENING TASKS

---

### 7.3.1 Detail of the hardening tasks

All the implemented tasks have a descriptive name indicating its purpose; in the case of CIS-based hardening tasks, the name includes the corresponding CIS recommendation identifier, as well as its full title according to the benchmarks, when possible. Tasks that need to clarify any additional information (e.g., included recommendations, possible values, important remarks, etc.) include code comments.

The tasks have been organized following the structure defined in *DSI 5: Design of the Architecture of the System Modules*, in the same order in which they appear in the followed guidelines, so they are easier to locate.



## 7.3.2 Results of the execution of the hardening tasks

Though the full construction process of the information system includes automating the installation and deployment of machines from code, the most vital part comprises the implementation and execution of Ansible scripts with specific tasks to harden said machines. As many of these hardening tasks directly translate CIS Benchmarks' recommendations into code, ensuring the remediations are correctly applied in those cases can be checked by means of specific audits. However, examining first the Ansible execution logs to find any failing task can prove useful for determining the reason of its failure and how it can be fixed, in case a solution exists.

An important thing to note is that, unless specified otherwise, if an Ansible task fails, the whole workflow stops. This can be useful in some cases, as it allows quick detection of the failing task, but can be a problem if the failure is acceptable. For cases like these, Ansible can be configured to ignore errors, so it will continue running tasks against the host; however, this scenario comes at a cost, as the recap of the run will not register any failed tasks. As such, logs will have to be carefully examined from the beginning of the execution to check if there have been unexpected results at any point of the run.

This "ignore errors" scenario is the one chosen for the project; the reasoning behind this is that as many security recommendations as possible should be run and applied to the machine being hardened. However, for each system the logs were checked and failures, recorded and solved unless an acceptable reason for it to fail could be given.

### 7.3.2.1 Windows 10 Education hardening tasks' execution summary

When running the hardening tasks for the Windows machine, it was found that none of them failed, unless restrictions to the communication between Ansible and the machine being provisioned were made (restriction to PowerShell access, reduced privileges, etc.; more information on *Execution of Additional Tests*). Logs were examined when provisioning both an English and a Spanish-based machine, so it could be guaranteed that in the end the tasks were correctly parameterized for the two locales.

### 7.3.2.2 Ubuntu Linux 18.04 hardening tasks' execution summary

Contrary to what happened to the Windows machine, some of the executed tasks for the Linux machine failed; the failures were recorded in the following table, as well as the explanation for solving them or not.

Task Description	Reason of failure	¿Acceptable?
Ensure systemd-timesyncd is installed	No package matching 'systemd-	It depends; systemd-timesyncd is a daemon for system clock synchronization, and the CIS benchmark explicitly establishes that if either chrony or ntp are



	timesyncd' is available	used instead, this daemon should be stopped and the section, skipped. Thus, failing to find the package could have no special impact. However, it was found that the daemon became inactive and eventually dead in the Ubuntu machine; the reason for this could not be found as it requires having more specialized administrative knowledge
Make sure services not required by CIS standard are stopped (item=ntp)	Could not find the requested service ntp: host	It depends on whether this service is meant to substitute systemd-timesyncd or chrony for time synchronization.
Make sure services not required by CIS standard are stopped (item=cups)	Could not find the requested service cups: host	Yes, as this task is supposed to look for this service and deactivate or stop it, so in the end the result will be more or less the same. In fact, the CIS Benchmark recommends removing completely CUPS from the system, specially for Servers (for Workstations it is an L2 recommendation). Not having it installed in the system means complying with the recommended value from the start

## 7.4 EXECUTION OF SYSTEM AUDITS

Once the hardening tasks started being implemented and executed over the Windows and Ubuntu machines, audits began taking place. Though the continuous auditing process helped to iteratively improve the machines' security levels, the most important audit reports were the ones performed before and after the Ansible provisioning, and are the ones provided as part of the documentation.

In this section of the document, only the most relevant parts of the reports will be shown and analyzed, as they are lengthy and can be better examined separately. For that regard, all the referenced reports have been made available for consulting.

To make it easier for the reader, tables summarizing the results will also be presented (see *Results' summary*).



## 7.4.1 Windows 10 Education Virtual Machine audits

### 7.4.1.1 Pre-hardening (base machine)

#### 7.4.1.1.1 Audit results with CLARA

#### Centro Criptológico Nacional



Nombre del sistema: VAGRANT-10  
Organización: Uniovi  
Unidad: Audit  
Categoría del sistema: BAJA

Auditado por Maria  
Informes generados el día 25/06/2022 15:45:33 UTC  
Versión de CLARA: 2.0  
0418a6c4-5044-4055-9945-164721870953-09504833-81c9-4368-a235-3c96ac32396c-2f74

Datos del sistema		Ocultar todo
Análisis ENS		Ocultar
Resultados		
Valor de criticidad	Cumplimiento (56,12%)	
OP.ACC.4 - Proceso de gestión de derechos de acceso (60%)		Mostrar
OP.ACC.5 - Mecanismos de autenticación (50%)		Mostrar
OP.ACC.6 - Acceso local (50%)		Mostrar
OP.EXP.2 - Configuración de seguridad (39,62%)		Mostrar
OP.EXP.5 - Gestión de cambios (100%)		Mostrar
OP.EXP.6 - Protección frente a código dañino (33,33%)		Mostrar
OP.EXP.8 - Registro de actividad de los usuarios (100%)		Mostrar
MP.EQ.2 - Bloqueo de puesto de trabajo (0%)		Mostrar
MP.COM.3 - Protección de la autenticidad y de la integridad (70%)		Mostrar
Directivas del sistema de ficheros (100%)		Ocultar
No hay datos relevantes que mostrar en esta sección.		
Directivas de servicios del sistema (90,51%)		Mostrar

0418a6c4-5044-4055-9945-164721870953-09504833-81c9-4368-a235-3c96ac32396c-2f74

Figure 14. W10 Education pre-hardening audit results with CLARA for a LOW category system



### Centro Criptológico Nacional

Nombre del sistema: VAGRANT-10  
Organización: Uniovi  
Unidad: Audit  
Categoría del sistema: MEDIA

Auditado por María  
Informe generado el día 25/06/2022 15:41:52 UTC  
Versión de CLARA: 2.0  
0418a014-0004-4005-9965-16472182983-09094683-8149-4488-a235-3636a32396c-2974

Mostrar todo

Datos del sistema		Mostrar
Análisis ENS		Ocultar
Resultados		
Valor de criticidad	Cumplimiento (53,42%)	
OP.ACC.4 - Proceso de gestión de derechos de acceso (60%)		Mostrar
OP.ACC.5 - Mecanismos de autenticación (33,33%)		Mostrar
OP.ACC.6 - Acceso local (25%)		Mostrar
OP.EXP.2 - Configuración de seguridad (36%)		Mostrar
OP.EXP.5 - Gestión de cambios (100%)		Mostrar
OP.EXP.6 - Protección frente a código dañino (33,33%)		Mostrar
OP.EXP.8 - Registro de actividad de los usuarios (100%)		Mostrar
OP.EXP.10 - Protección de los registros de actividad (100%)		Mostrar
MP.EQ.2 - Bloqueo de puesto de trabajo (0%)		Mostrar
MP.COM.3 - Protección de la autenticidad y de la integridad (66,67%)		Mostrar
Directivas del sistema de ficheros (100%)		Mostrar
Directivas de servicios del sistema (89,78%)		Mostrar

0418a014-0004-4005-9965-16472182983-09094683-8149-4488-a235-3636a32396c-2974

Figure 15. W10 Education pre-hardening audit results with CLARA for a MEDIUM category system

### Centro Criptológico Nacional

Nombre del sistema: VAGRANT-10  
Organización: Uniovi  
Unidad: Audit  
Categoría del sistema: ALTA

Auditado por María  
Informe generado el día 25/06/2022 15:29:53 UTC  
Versión de CLARA: 2.0  
0418a014-0004-4005-9965-16472182983-09094683-8149-4488-a235-3636a32396c-2974

Mostrar todo

Datos del sistema		Mostrar
Análisis ENS		Ocultar
Resultados		
Valor de criticidad	Cumplimiento (56,4%)	
OP.ACC.4 - Proceso de gestión de derechos de acceso (60%)		Mostrar
OP.ACC.5 - Mecanismos de autenticación (31,25%)		Mostrar
OP.ACC.6 - Acceso local (25%)		Mostrar
OP.EXP.2 - Configuración de seguridad (35,43%)		Mostrar
OP.EXP.5 - Gestión de cambios (100%)		Mostrar
OP.EXP.6 - Protección frente a código dañino (33,33%)		Mostrar
OP.EXP.8 - Registro de actividad de los usuarios (88,89%)		Mostrar
OP.EXP.10 - Protección de los registros de actividad (75%)		Mostrar
MP.EQ.2 - Bloqueo de puesto de trabajo (0%)		Mostrar
MP.EQ.3 - Protección de equipos informáticos (100%) (*)		Mostrar
MP.COM.3 - Protección de la autenticidad y de la integridad (53,33%)		Mostrar
Directivas del sistema de ficheros (100%)		Mostrar
Directivas de servicios del sistema (89,42%)		Mostrar

0418a014-0004-4005-9965-16472182983-09094683-8149-4488-a235-3636a32396c-2974

Figure 16. W10 Education pre-hardening audit results with CLARA for a HIGH category system

#### 7.4.1.1.2 Audit results with CIS-CAT Lite

### Summary

Description	Tests				Scoring			
	Pass	Fail	Error	Unkn.	Man.	Score	Max	Percent
<b>1 Account Policies</b>	3	5	0	2	0	3.0	10.0	30%
1.1 Password Policy	1	4	0	2	0	1.0	7.0	14%
1.2 Account Lockout Policy	2	1	0	0	0	2.0	3.0	67%
<b>2 Local Policies</b>	54	43	0	1	1	54.0	98.0	55%
2.1 Audit Policy	0	0	0	0	0	0.0	0.0	0%
2.2 User Rights Assignment	23	14	0	0	0	23.0	37.0	62%
19.7.47.1 Networking	0	0	0	0	0	0.0	0.0	0%
19.7.47.2 Playback	0	0	0	0	0	0.0	0.0	0%
<b>Total</b>	<b>81</b>	<b>262</b>	<b>0</b>	<b>3</b>	<b>1</b>	<b>81.0</b>	<b>346.0</b>	<b>23%</b>

**Note:** Actual scores are subject to rounding errors. The sum of these values may not result in the exact overall score.

### Profiles

This benchmark contains 10 profiles. The **Level 1 (L1) - Corporate/Enterprise Environment (general use)** profile was used for this assessment.

*Figure 17. W10 Education pre-hardening audit results with CIS-CAT Lite for L1 profile*

### Summary

Description	Tests				Scoring			
	Pass	Fail	Error	Unkn.	Man.	Score	Max	Percent
<b>1 Account Policies</b>	3	5	0	2	0	3.0	10.0	30%
1.1 Password Policy	1	4	0	2	0	1.0	7.0	14%
1.2 Account Lockout Policy	2	1	0	0	0	2.0	3.0	67%
<b>2 Local Policies</b>	54	48	0	1	1	54.0	103.0	52%
2.1 Audit Policy	0	0	0	0	0	0.0	0.0	0%
2.2 User Rights Assignment	23	16	0	0	0	23.0	39.0	59%
2.3 Security Options	31	32	0	1	1	31.0	64.0	48%
19.7.47.1 Networking	0	0	0	0	0	0.0	0.0	0%
19.7.47.2 Playback	0	1	0	0	0	0.0	1.0	0%
<b>Total</b>	<b>85</b>	<b>354</b>	<b>0</b>	<b>3</b>	<b>1</b>	<b>85.0</b>	<b>442.0</b>	<b>19%</b>

**Note:** Actual scores are subject to rounding errors. The sum of these values may not result in the exact overall score.

### Profiles

This benchmark contains 10 profiles. The **Level 2 (L2) - High Security/Sensitive Data Environment (limited functionality)** profile was used for this assessment.

*Figure 18. W10 Education pre-hardening audit results with CIS-CAT Lite for L2 profile*



## 7.4.1.2 Post-hardening (only L1 tasks)

### 7.4.1.2.1 Audit results with CLARA

#### Centro Criptológico Nacional



Nombre del sistema: VAGRANT-10  
Organización: Uniovi  
Unidad: Audit  
Categoría del sistema: BAJA

Auditado por Maria  
Informes generados el día 25/06/2022 17:24:17 UTC  
Versión de CLARA: 2.0  
0413ba14-0564-405d-996c-16472182953-09504803-81c9-4d68-e235-38766c3229e1-3274

Datos del sistema		Mostrar todo
Análisis ENS		Mostrar
Resultados		Ocultar
Valor de criticidad	Cumplimiento (84,43%)	
OP.ACC.4 - Proceso de gestión de derechos de acceso (100%)		Mostrar
OP.ACC.5 - Mecanismos de autenticación (100%)		Mostrar
OP.ACC.6 - Acceso local (100%)		Mostrar
OP.EXP.2 - Configuración de seguridad (45,28%)		Mostrar
OP.EXP.5 - Gestión de cambios (100%)		Mostrar
OP.EXP.6 - Protección frente a código dañado (66,67%)		Mostrar
OP.EXP.8 - Registro de actividad de los usuarios (100%)		Mostrar
MP.FQ.2 - Bloqueo de puesto de trabajo (75%)		Mostrar
MP.COM.3 - Protección de la autenticidad y de la integridad (90%)		Mostrar
Directivas del sistema de ficheros (100%)		Mostrar
Directivas de servicios del sistema (93,43%)		Mostrar

0413ba14-0564-405d-996c-16472182953-09504803-81c9-4d68-e235-38766c3229e1-3274

**Figure 19. W10 Education post-hardening (L1) audit results with CLARA for a LOW category system**





## Centro Criptológico Nacional

Nombre del sistema: VAGRANT-10  
Organización: Uniovi  
Unidad: Audit  
Categoría del sistema: MEDIA

Auditado por Maria  
Informes generados el día 25/06/2022 17:21:04 UTC  
Versión de CLARA: 2.0  
0413ba14-5004-4030-99e5-164721823f53-09504803-81c0-4368-4235-36558632394e1-2f74

Datos del sistema		Mostrar todo
Análisis ENS		Ocultar
Resultados		
Valor de criticidad	Cumplimiento (77,56%)	
OP.ACC.4 - Proceso de gestión de derechos de acceso (100%)		Mostrar
OP.ACC.5 - Mecanismos de autenticación (66,67%)		Mostrar
OP.ACC.6 - Acceso local (66,67%)		Mostrar
OP.EXP.2 - Configuración de seguridad (44%)		Mostrar
OP.EXP.5 - Gestión de cambios (100%)		Mostrar
OP.EXP.6 - Protección frente a código dañino (66,67%)		Mostrar
OP.EXP.8 - Registro de actividad de los usuarios (100%)		Mostrar
OP.EXP.10 - Protección de los registros de actividad (100%)		Mostrar
MP.EQ.2 - Bloqueo de puesto de trabajo (83,33%)		Mostrar
MP.COM.3 - Protección de la autenticidad y de la integridad (91,67%)		Mostrar
Directivas del sistema de ficheros (100%)		Mostrar
Directivas de servicios del sistema (92,7%)		Mostrar

0413ba14-5004-4030-99e5-164721823f53-09504803-81c0-4368-4235-36558632394e1-2f74

**Figure 20. W10 Education post-hardening (L1) audit results with CLARA for a MEDIUM category system**



Centro Criptológico Nacional

Nombre del sistema: VAGRANT-10  
Organización: Uniovi  
Unidad: Audit  
Categoría del sistema: ALTA

Auditado por Maria  
Informes generados el día 25/06/2022 17:17:11 UTC  
Versión de CLARA: 2.0  
0413ba14-1044-405b-99a5-1a471182953-0504603-81c9-43a8-a235-3b5b6a3239e1-3f74

Datos del sistema		Mostrar todo
Análisis ENS		Ocultar
Resultados		
Valor de criticidad	Cumplimiento (77,11%)	
OP.ACC.4 - Proceso de gestión de derechos de acceso (90%)		Mostrar
OP.ACC.5 - Mecanismos de autenticación (62,5%)		Mostrar
OP.ACC.6 - Acceso local (66,67%)		Mostrar
OP.EXP.2 - Configuración de seguridad (44,09%)		Mostrar
OP.EXP.5 - Gestión de cambios (100%)		Mostrar
OP.EXP.6 - Protección frente a código dañino (66,67%)		Mostrar
OP.EXP.8 - Registro de actividad de los usuarios (88,89%)		Mostrar
OP.EXP.10 - Protección de los registros de actividad (75%)		Mostrar
MP.EQ.2 - Bloqueo de puesto de trabajo (83,33%)		Mostrar
MP.EQ.3 - Protección de equipos informáticos (100%) (*)		Mostrar
MP.COM.3 - Protección de la autenticidad y de la integridad (100%)		Mostrar
Directivas del sistema de ficheros (100%)		Mostrar
Directivas de servicios del sistema (92,34%)		Mostrar

0413ba14-1044-405b-99a5-1a471182953-0504603-81c9-43a8-a235-3b5b6a3239e1-3f74

Figure 21. W10 Education post-hardening (L1) audit results with CLARA for a HIGH category system

7.4.1.2.2 Audit results with CIS-CAT Lite

Summary

Description	Tests				Scoring			
	Pass	Fail	Error	Unkn.	Man.	Score	Max	Percent
<b>1 Account Policies</b>	8	0	0	2	0	8.0	10.0	80%
1.1 Password Policy	5	0	0	2	0	5.0	7.0	71%
1.2 Account Lockout Policy	3	0	0	0	0	3.0	3.0	100%
<b>2 Local Policies</b>	93	4	0	1	1	93.0	98.0	95%
2.1 Audit Policy	0	0	0	0	0	0.0	0.0	0%
19.7.47.1 Networking	0	0	0	0	0	0.0	0.0	0%
19.7.47.2 Playback	0	0	0	0	0	0.0	0.0	0%
<b>Total</b>	<b>193</b>	<b>150</b>	<b>0</b>	<b>3</b>	<b>1</b>	<b>193.0</b>	<b>346.0</b>	<b>56%</b>

Note: Actual scores are subject to rounding errors. The sum of these values may not result in the exact overall score.

Profiles

This benchmark contains 10 profiles. The Level 1 (L1) - Corporate/Enterprise Environment (general use) profile was used for this assessment.

Figure 22. W10 Education post-hardening (L1) audit results with CIS-CAT Lite for L1 profile



## Summary

Description	Tests				Scoring			
	Pass	Fail	Error	Unkn	Man. Score	Max	Percent	
<b>1 Account Policies</b>	<b>8</b>	<b>0</b>	<b>0</b>	<b>2</b>	<b>0</b>	<b>8.0</b>	<b>10.0</b>	<b>80%</b>
1.1 Password Policy	5	0	0	2	0	5.0	7.0	71%
1.2 Account Lockout Policy	3	0	0	0	0	3.0	3.0	100%
<b>2 Local Policies</b>	<b>94</b>	<b>8</b>	<b>0</b>	<b>1</b>	<b>1</b>	<b>94.0</b>	<b>103.0</b>	<b>91%</b>
2.1 Audit Policy	0	0	0	0	0	0.0	0.0	0%
19.7.47.1 Networking	0	0	0	0	0	0.0	0.0	0%
19.7.47.2 Playback	0	1	0	0	0	0.0	1.0	0%
<b>Total</b>	<b>199</b>	<b>240</b>	<b>0</b>	<b>3</b>	<b>1</b>	<b>199.0</b>	<b>442.0</b>	<b>45%</b>

**Note:** Actual scores are subject to rounding errors. The sum of these values may not result in the exact overall score.

## Profiles

This benchmark contains 10 profiles. The **Level 2 (L2) - High Security/Sensitive Data Environment (limited functionality)** profile was used for this assessment.

*Figure 23. W10 Education post-hardening (L1) audit results with CIS-CAT Lite for L2 profile*

### 7.4.1.3 Post-hardening (L1 tasks + extra L2 tasks)

#### 7.4.1.3.1 Audit results with CLARA

## Centro Criptológico Nacional

Nombre del sistema: VAGRANT-10  
Organización: Uniovi  
Unidad: Audit  
Categoría del sistema: BAJA



Auditado por María  
Informes generados el día 25/06/2022 03:52:52 UTC  
Versión de CLARA: 2.0  
0418ba14-0364-40b5-89a5-16472182953-09504803-81c9-4d48-a235-3b3bc3239a3-2f74

Mostrar todo

Datos del sistema		Mostrar
Análisis ENS		Ocultar
<b>Resultados</b>		
Valor de criticidad	Cumplimiento (84,56%)	
OP.ACC.4 - Proceso de gestión de derechos de acceso (100%)		Mostrar
OP.ACC.5 - Mecanismos de autenticación (100%)		Mostrar
OP.ACC.6 - Acceso local (100%)		Mostrar
OP.EXP.2 - Configuración de seguridad (46,23%)		Mostrar
OP.EXP.5 - Gestión de cambios (100%)		Mostrar
OP.EXP.6 - Protección frente a código dañino (66,67%)		Mostrar
OP.EXP.8 - Registro de actividad de los usuarios (100%)		Mostrar
MP.EQ.2 - Bloqueo de puesto de trabajo (75%)		Mostrar
MP.COM.3 - Protección de la autenticidad y de la integridad (90%)		Mostrar
Directivas del sistema de ficheros (100%)		Mostrar
Directivas de servicios del sistema (94,53%)		Mostrar

0418ba14-0364-40b5-89a5-16472182953-09504803-81c9-4d48-a235-3b3bc3239a3-2f74

*Figure 24. W10 Education post-hardening (L1 + extra L2) audit results with CLARA for a LOW category system*



## Centro Criptológico Nacional

Nombre del sistema: VAGRANT-10  
Organización: Uniovi  
Unidad: Audit  
Categoría del sistema: MEDIA

Auditado por María  
Informe generado el día 25/06/2022 03:49:19 UTC  
Versión de CLARA: 2.0

0418ae14-5041-40f0-99e5-16472182953-09504803-81c9-46e8-e235-365bd3239e1-2f74

Mostrar todo

Datos del sistema		Mostrar
Análisis ENS		Ocultar
Resultados		
Valor de criticidad	Cumplimiento (78,48%)	
OP.ACC.4 - Proceso de gestión de derechos de acceso (100%)		Mostrar
OP.ACC.5 - Mecanismos de autenticación (73,33%)		Mostrar
OP.ACC.6 - Acceso local (66,67%)		Mostrar
OP.EXP.2 - Configuración de seguridad (44,8%)		Mostrar
OP.EXP.5 - Gestión de cambios (100%)		Mostrar
OP.EXP.6 - Protección frente a código dañino (66,67%)		Mostrar
OP.EXP.8 - Registro de actividad de los usuarios (100%)		Mostrar
OP.EXP.10 - Protección de los registros de actividad (100%)		Mostrar
MP.EQ.2 - Bloqueo de puesto de trabajo (83,33%)		Mostrar
MP.COM.3 - Protección de la autenticidad y de la integridad (91,67%)		Mostrar
Directivas del sistema de ficheros (100%)		Mostrar
Directivas de servicios del sistema (93,8%)		Mostrar

0418ae14-5041-40f0-99e5-16472182953-09504803-81c9-46e8-e235-365bd3239e1-2f74

**Figure 25. W10 Education post-hardening (L1 + extra L2) audit results with CLARA for a MEDIUM category system**



## Centro Criptológico Nacional

Nombre del sistema: VAGRANT-10  
Organización: Uniovi  
Unidad: Audit  
Categoría del sistema: ALTA

Auditado por Maria  
Informes generados el día 25/06/2022 03:45:33 UTC  
Versión de CLARA: 2.0  
0418ae14-5064-40f0-99e5-16472182953-09504003-81c9-4d68-a235-365bd3239a1-2f74

Mostrar todo

Datos del sistema		Mostrar
Análisis ENS		Ocultar
Resultados		
Valor de criticidad	Cumplimiento (77,79%)	
OP.ACC.4 - Proceso de gestión de derechos de acceso (90%)		Mostrar
OP.ACC.5 - Mecanismos de autenticación (68,75%)		Mostrar
OP.ACC.6 - Acceso local (66,67%)		Mostrar
OP.EXP.2 - Configuración de seguridad (44,09%)		Mostrar
OP.EXP.5 - Gestión de cambios (100%)		Mostrar
OP.EXP.6 - Protección frente a código dañino (66,67%)		Mostrar
OP.EXP.8 - Registro de actividad de los usuarios (88,89%)		Mostrar
OP.EXP.10 - Protección de los registros de actividad (75%)		Mostrar
MP.EQ.2 - Bloqueo de puesto de trabajo (83,33%)		Mostrar
MP.EQ.3 - Protección de equipos informáticos (100%) (*)		Mostrar
MP.COM.3 - Protección de la autenticidad y de la integridad (100%)		Mostrar
Directivas del sistema de ficheros (100%)		Mostrar
Directivas de servicios del sistema (93,07%)		Mostrar

0418ae14-5064-40f0-99e5-16472182953-09504003-81c9-4d68-a235-365bd3239a1-2f74

**Figure 26. W10 Education post-hardening (L1 + extra L2) audit results with CLARA for a HIGH category system**

### 7.4.1.3.2 Audit results with CIS-CAT Lite

#### Summary

Description	Tests				Scoring			
	Pass	Fail	Error	Unkn	Man.	Score	Max	Percent
<b>1 Account Policies</b>	8	0	0	2	0	8.0	10.0	80%
1.1 Password Policy	5	0	0	2	0	5.0	7.0	71%
1.2 Account Lockout Policy	3	0	0	0	0	3.0	3.0	100%
<b>2 Local Policies</b>	93	4	0	1	1	93.0	98.0	95%
2.1 Audit Policy	0	0	0	0	0	0.0	0.0	0%
19.7.47.1 Networking	0	0	0	0	0	0.0	0.0	0%
19.7.47.2 Playback	0	0	0	0	0	0.0	0.0	0%
<b>Total</b>	<b>193</b>	<b>150</b>	<b>0</b>	<b>3</b>	<b>1</b>	<b>193.0</b>	<b>346.0</b>	<b>56%</b>

**Note:** Actual scores are subject to rounding errors. The sum of these values may not result in the exact overall score.

#### Profiles

This benchmark contains 10 profiles. The **Level 1 (L1) - Corporate/Enterprise Environment (general use)** profile was used for this assessment.

**Figure 27. W10 Education post-hardening (L1 + extra L2) audit results with CIS-CAT Lite for L1 profile**

## Summary

Description	Tests					Scoring		
	Pass	Fail	Error	Unkn	Man.	Score	Max	Percent
<b>1 Account Policies</b>	<b>8</b>	<b>0</b>	<b>0</b>	<b>2</b>	<b>0</b>	<b>8.0</b>	<b>10.0</b>	<b>80%</b>
1.1 Password Policy	5	0	0	2	0	5.0	7.0	71%
1.2 Account Lockout Policy	3	0	0	0	0	3.0	3.0	100%
<b>2 Local Policies</b>	<b>96</b>	<b>6</b>	<b>0</b>	<b>1</b>	<b>1</b>	<b>96.0</b>	<b>103.0</b>	<b>93%</b>
2.1 Audit Policy	0	0	0	0	0	0.0	0.0	0%
19.7.47.1 Networking	0	0	0	0	0	0.0	0.0	0%
19.7.47.2 Playback	0	1	0	0	0	0.0	1.0	0%
<b>Total</b>	<b>204</b>	<b>235</b>	<b>0</b>	<b>3</b>	<b>1</b>	<b>204.0</b>	<b>442.0</b>	<b>46%</b>

**Note:** Actual scores are subject to rounding errors. The sum of these values may not result in the exact overall score.

## Profiles

This benchmark contains 10 profiles. The **Level 2 (L2) - High Security/Sensitive Data Environment (limited functionality)** profile was used for this assessment.

*Figure 28. W10 Education post-hardening (L1 + extra L2) audit results with CIS-CAT Lite for L2 profile*

## 7.4.2 Ubuntu Linux Virtual Machine audits

### 7.4.2.1 Pre-hardening

```

Hardening index : 56 [#####          ]
Tests performed : 252
Plugins enabled : 0

Components:
- Firewall           [U]
- Malware scanner    [X]

Scan mode:
Normal [U] Forensics [ ] Integration [ ] Pentest [ ]

Lynis modules:
- Compliance status [?]
- Security audit    [U]
- Vulnerability scan [U]

Files:
- Test and debug information : /var/log/lynis.log
- Report data                : /var/log/lynis-report.dat

=====

Lynis 3.0.8

Auditing, system hardening, and compliance for UNIX-based systems
(Linux, macOS, BSD, and others)

2007-2021, CISofy - https://cisofy.com/lynis/
Enterprise support available (compliance, plugins, interface and tools)

=====

[TIP]: Enhance Lynis audits by adding your settings to custom.prf (see /etc/lynis/default.prf for
all settings)

vagrant@vagrant:/$ _

```

*Figure 29. Ubuntu pre-hardening machine audit summary with Lynis*

### 7.4.2.2 Post-hardening

```
Hardening index : 82 [##### ]
Tests performed : 258
Plugins enabled : 0

Components:
- Firewall           [U]
- Malware scanner    [X]

Scan mode:
Normal [U] Forensics [ ] Integration [ ] Pentest [ ]

Lynis modules:
- Compliance status [?]
- Security audit     [U]
- Vulnerability scan [U]

Files:
- Test and debug information : /var/log/lynis.log
- Report data                : /var/log/lynis-report.dat

=====

Lynis 3.0.8

Auditing, system hardening, and compliance for UNIX-based systems
(Linux, macOS, BSD, and others)

2007-2021, CISOfy - https://cisofy.com/lynis/
Enterprise support available (compliance, plugins, interface and tools)

=====

[TIP]: Enhance Lynis audits by adding your settings to custom.prf (see /etc/lynis/default.prf for
all settings)

vagrant@vagrant:~$ _
```

Figure 30. Ubuntu post-hardening machine audit summary with Lynis



## 7.4.3 Windows 10 real machines

### 7.4.3.1 2021-2022 academic year machine

#### 7.4.3.1.1 Audit results with CLARA



#### Centro Criptológico Nacional

Nombre del sistema: DESKTOP-02IGJT5  
Organización: Uniovi  
Unidad: Equipos Escuela 2021-2022  
Categoría del sistema: BAJA

Auditado por María  
Informes generados el día 28/06/2022 15:23:33 UTC  
Versión de CLARA: 2.0  
0413ba14-5064-4020-99e5-16472182953-09504803-81c9-46a8-e235-365b6c3239a1-2f74

Mostrar todo

Datos del sistema		Mostrar
Análisis ENS		Ocultar
Resultados		
Valor de criticidad	Cumplimiento (29,71%)	
OP.ACC.4 - Proceso de gestión de derechos de acceso (70%)		Mostrar
OP.ACC.5 - Mecanismos de autenticación (50%)		Mostrar
OP.ACC.6 - Acceso local (0%)		Mostrar
OP.EXP.2 - Configuración de seguridad (38,68%)		Mostrar
OP.EXP.5 - Gestión de cambios (0%)		Mostrar
OP.EXP.6 - Protección frente a código dañino (33,33%)		Mostrar
OP.EXP.8 - Registro de actividad de los usuarios (100%)		Mostrar
MP.EQ.2 - Bloqueo de puesto de trabajo (0%)		Mostrar
MP.COM.3 - Protección de la autenticidad y de la integridad (70%)		Mostrar
Directivas del sistema de ficheros (100%)		Mostrar
Directivas de servicios del sistema (90,88%)		Mostrar

0413ba14-5064-4020-99e5-16472182953-09504803-81c9-46a8-e235-365b6c3239a1-2f74

**Figure 31. W10 Pro (2021-2022 academic year image) audit results with CLARA for a LOW category system**





## Centro Criptológico Nacional

Nombre del sistema: DESKTOP-02IGJT5  
Organización: Uniovi  
Unidad: Equipos Escuela 2021-2022  
Categoría del sistema: MEDIA

Auditado por María  
Informes generados el día 28/06/2022 15:21:48 UTC  
Versión de CLARA: 2.0  
0418ae14-5044-4005-99e5-16472182953-09504803-81c9-4d68-a235-365bdc3239e1-2F74

Mostrar todo

Datos del sistema		Mostrar
Análisis ENS		Ocultar
Resultados		
Valor de criticidad	Cumplimiento (31,24%)	
OP.ACC.4 - Proceso de gestión de derechos de acceso (70%)		Mostrar
OP.ACC.5 - Mecanismos de autenticación (33,33%)		Mostrar
OP.ACC.6 - Acceso local (0%)		Mostrar
OP.EXP.2 - Configuración de seguridad (33,6%)		Mostrar
OP.EXP.5 - Gestión de cambios (0%)		Mostrar
OP.EXP.6 - Protección frente a código dañado (33,33%)		Mostrar
OP.EXP.8 - Registro de actividad de los usuarios (100%)		Mostrar
OP.EXP.10 - Protección de los registros de actividad (100%)		Mostrar
MP.EQ.2 - Bloqueo de puesto de trabajo (0%)		Mostrar
MP.COM.3 - Protección de la autenticidad y de la integridad (66,67%)		Mostrar
Directivas del sistema de ficheros (100%)		Mostrar
Directivas de servicios del sistema (90,15%)		Mostrar

0418ae14-5044-4005-99e5-16472182953-09504803-81c9-4d68-a235-365bdc3239e1-2F74

**Figure 32. W10 Pro (2021-2022 academic year image) audit results with CLARA for a MEDIUM category system**



## Centro Criptológico Nacional

Nombre del sistema: DESKTOP-02IGJT5  
Organización: Uniovi  
Unidad: Equipos Escuela 2021-2022  
Categoría del sistema: ALTA

Auditado por María  
Informes generados el día 28/06/2022 15:19:27 UTC  
Versión de CLARA: 2.0  
0418ae14-50d1-40f0-99e5-1647218f2953-0950403-81c9-4de8-a235-3b5bd3239e1-2f74

Datos del sistema		Mostrar todo
Análisis ENS		Ocultar
Resultados		
Valor de criticidad	Cumplimiento (36,44%)	
OP.ACC.4 - Proceso de gestión de derechos de acceso (70%)		Mostrar
OP.ACC.5 - Mecanismos de autenticación (31,25%)		Mostrar
OP.ACC.6 - Acceso local (0%)		Mostrar
OP.EXP.2 - Configuración de seguridad (33,07%)		Mostrar
OP.EXP.5 - Gestión de cambios (0%)		Mostrar
OP.EXP.6 - Protección frente a código dañino (33,33%)		Mostrar
OP.EXP.8 - Registro de actividad de los usuarios (88,89%)		Mostrar
OP.EXP.10 - Protección de los registros de actividad (75%)		Mostrar
MP.EQ.2 - Bloqueo de puesto de trabajo (0%)		Mostrar
MP.EQ.3 - Protección de equipos informáticos (100%) (*)		Mostrar
MP.COM.3 - Protección de la autenticidad y de la integridad (53,33%)		Mostrar
Directivas del sistema de ficheros (100%)		Mostrar
Directivas de servicios del sistema (89,78%)		Mostrar

Figure 33. W10 Pro (2021-2022 academic year image) audit results with CLARA for a HIGH category system

### 7.4.3.1.2 Audit results with CIS-CAT Lite

#### Summary

Description	Tests				Scoring			
	Pass	Fail	Error	Unkn.	Man.	Score	Max	Percent
<b>1 Account Policies</b>	3	5	0	2	0	3.0	10.0	30%
1.1 Password Policy	1	4	0	2	0	1.0	7.0	14%
1.2 Account Lockout Policy	2	1	0	0	0	2.0	3.0	67%
<b>2 Local Policies</b>	60	37	0	1	1	60.0	98.0	61%
2.1 Audit Policy	0	0	0	0	0	0.0	0.0	0%
19.7.47.1 Networking	0	0	0	0	0	0.0	0.0	0%
19.7.47.2 Playback	0	0	0	0	0	0.0	0.0	0%
<b>Total</b>	<b>85</b>	<b>258</b>	<b>0</b>	<b>3</b>	<b>1</b>	<b>85.0</b>	<b>346.0</b>	<b>25%</b>

**Note:** Actual scores are subject to rounding errors. The sum of these values may not result in the exact overall score.

#### Profiles

This benchmark contains 10 profiles. The **Level 1 (L1) - Corporate/Enterprise Environment (general use)** profile was used for this assessment.

Figure 34. W10 Pro (2021-2022 academic year image) audit results with CIS-CAT Lite for L1 profile



## Summary

Description	Tests					Scoring		
	Pass	Fail	Error	Unkn	Man.	Score	Max	Percent
<b>1 Account Policies</b>	3	5	0	2	0	3.0	10.0	30%
1.1 Password Policy	1	4	0	2	0	1.0	7.0	14%
1.2 Account Lockout Policy	2	1	0	0	0	2.0	3.0	67%
<b>2 Local Policies</b>	60	42	0	1	1	60.0	103.0	58%
19.7.47.1 Networking	0	0	0	0	0	0.0	0.0	0%
19.7.47.2 Playback	0	1	0	0	0	0.0	1.0	0%
<b>Total</b>	<b>89</b>	<b>350</b>	<b>0</b>	<b>3</b>	<b>1</b>	<b>89.0</b>	<b>442.0</b>	<b>20%</b>

**Note:** Actual scores are subject to rounding errors. The sum of these values may not result in the exact overall score.

## Profiles

This benchmark contains 10 profiles. The **Level 2 (L2) - High Security/Sensitive Data Environment (limited functionality)** profile was used for this assessment.

*Figure 35. W10 Pro (2021-2022 academic year image) audit results with CIS-CAT Lite for L2 profile*

### 7.4.3.2 2022-2023 academic year machine

#### 7.4.3.2.1 Audit results with CLARA

## Centro Criptológico Nacional

Nombre del sistema: DESKTOP-02IGJT5  
Organización: Uniovi  
Unidad: Equipos Escuela 2022-2023  
Categoría del sistema: BAJA



centro criptológico nacional

Auditado por Maria  
Informe generado el día 28/06/2022 15:08:23 UTC  
Versión de CLARA: 2.0  
0413ae14-5064-405d-99d5-164721182f93-09504603-81c9-4668-a235-365b6c3239a1-2f74

Mostrar todo

Datos del sistema		Mostrar
Análisis ENS		Ocultar
Resultados		
Valor de criticidad	Cumplimiento (50,02%)	
OP.ACC.4 - Proceso de gestión de derechos de acceso (70%)		Mostrar
OP.ACC.5 - Mecanismos de autenticación (50%)		Mostrar
OP.ACC.6 - Acceso local (0%)		Mostrar
OP.EXP.2 - Configuración de seguridad (38,68%)		Mostrar
OP.EXP.5 - Gestión de cambios (100%)		Mostrar
OP.EXP.6 - Protección frente a código dañino (33,33%)		Mostrar
OP.EXP.8 - Registro de actividad de los usuarios (100%)		Mostrar
MP.EQ.2 - Bloqueo de puesto de trabajo (0%)		Mostrar
MP.COM.3 - Protección de la autenticidad y de la integridad (70%)		Mostrar
Directivas del sistema de ficheros (100%)		Mostrar
Directivas de servicios del sistema (90,51%)		Mostrar

0413ae14-5064-405d-99d5-164721182f93-09504603-81c9-4668-a235-365b6c3239a1-2f74

*Figure 36. W10 Pro (2022-2023 academic year image) audit results with CLARA for a LOW category system*



## Centro Criptológico Nacional

Nombre del sistema: DESKTOP-02IGJT5  
Organización: Uniovi  
Unidad: Equipos Escuela 2022-2023  
Categoría del sistema: MEDIA

Auditado por Maria  
Informes generados el día 28/06/2022 15:06:50 UTC  
Versión de CLARA: 2.0  
0418a14-5064-402b-99a5-164721182953-09504803-81c9-4468-4235-3b5b6c3239e1-3274

Mostrar todo

Datos del sistema		Mostrar
Análisis ENS		Ocultar
Resultados		
Valor de criticidad	Cumplimiento (50,64%)	
OP.ACC.4 - Proceso de gestión de derechos de acceso (70%)		Mostrar
OP.ACC.5 - Mecanismos de autenticación (33,33%)		Mostrar
OP.ACC.6 - Acceso local (0%)		Mostrar
OP.EXP.2 - Configuración de seguridad (33,6%)		Mostrar
OP.EXP.5 - Gestión de cambios (100%)		Mostrar
OP.EXP.6 - Protección frente a código dañino (33,33%)		Mostrar
OP.EXP.8 - Registro de actividad de los usuarios (100%)		Mostrar
OP.EXP.10 - Protección de los registros de actividad (100%)		Mostrar
MP.EQ.2 - Bloqueo de puesto de trabajo (0%)		Mostrar
MP.COM.3 - Protección de la autenticidad y de la integridad (66,67%)		Mostrar
Directivas del sistema de ficheros (100%)		Mostrar
Directivas de servicios del sistema (89,78%)		Mostrar

0418a14-5064-402b-99a5-164721182953-09504803-81c9-4468-4235-3b5b6c3239e1-3274

**Figure 37. W10 Pro (2022-2023 academic year image) audit results with CLARA for a MEDIUM category system**



## Centro Criptológico Nacional

Nombre del sistema: DESKTOP-02IGJT5  
Organización: Uniovi  
Unidad: Equipos Escuela 2022-2023  
Categoría del sistema: ALTA

Auditado por María  
Informes generados el día 28/06/2022 15:04:37 UTC  
Versión de CLARA: 2.0  
0418ae14-5084-4095-99e5-16472182953-09504033-81c9-46e8-a235-3636dc3239e1-2f74

Datos del sistema		Mostrar todo
Análisis ENS		Ocultar
Resultados		
Valor de criticidad	Cumplimiento (53,94%)	
OP.ACC.4 - Proceso de gestión de derechos de acceso (70%)		Mostrar
OP.ACC.5 - Mecanismos de autenticación (31,25%)		Mostrar
OP.ACC.6 - Acceso local (0%)		Mostrar
OP.EXP.2 - Configuración de seguridad (33,07%)		Mostrar
OP.EXP.5 - Gestión de cambios (100%)		Mostrar
OP.EXP.6 - Protección frente a código dañino (33,33%)		Mostrar
OP.EXP.8 - Registro de actividad de los usuarios (88,89%)		Mostrar
OP.EXP.10 - Protección de los registros de actividad (75%)		Mostrar
MP.EQ.2 - Bloqueo de puesto de trabajo (0%)		Mostrar
MP.EQ.3 - Protección de equipos informáticos (100%) (*)		Mostrar
MP.COM.3 - Protección de la autenticidad y de la integridad (53,33%)		Mostrar
Directivas del sistema de ficheros (100%)		Mostrar
Directivas de servicios del sistema (89,78%)		Mostrar

0418ae14-5084-4095-99e5-16472182953-09504033-81c9-46e8-a235-3636dc3239e1-2f74

**Figure 38. W10 Pro (2022-2023 academic year image) audit results with CLARA for a HIGH category system**

### 7.4.3.2.2 Audit results with CIS-CAT Lite

#### Summary

Description	Tests				Scoring			
	Pass	Fail	Error	Unkn.	Man.	Score	Max	Percent
<b>1 Account Policies</b>	3	5	0	2	0	3.0	10.0	30%
1.1 Password Policy	1	4	0	2	0	1.0	7.0	14%
1.2 Account Lockout Policy	2	1	0	0	0	2.0	3.0	67%
<b>2 Local Policies</b>	60	37	0	1	1	60.0	98.0	61%
19.7.47.1 Networking	0	0	0	0	0	0.0	0.0	0%
19.7.47.2 Playback	0	0	0	0	0	0.0	0.0	0%
<b>Total</b>	<b>85</b>	<b>258</b>	<b>0</b>	<b>3</b>	<b>1</b>	<b>85.0</b>	<b>346.0</b>	<b>25%</b>

**Note:** Actual scores are subject to rounding errors. The sum of these values may not result in the exact overall score.

#### Profiles

This benchmark contains 10 profiles. The **Level 1 (L1) - Corporate/Enterprise Environment (general use)** profile was used for this assessment.

**Figure 39. W10 Pro (2022-2023 academic year image) audit results with CIS-CAT Lite for L1 profile**

## Summary

Description	Tests				Scoring			
	Pass	Fail	Error	Unkn	Man. Score	Max	Percent	
<b>1 Account Policies</b>	3	5	0	2	0	3.0	10.0	30%
1.1 Password Policy	1	4	0	2	0	1.0	7.0	14%
1.2 Account Lockout Policy	2	1	0	0	0	2.0	3.0	67%
<b>2 Local Policies</b>	60	42	0	1	1	60.0	103.0	58%
19.7.47.1 Networking	0	0	0	0	0	0.0	0.0	0%
19.7.47.2 Playback	0	1	0	0	0	0.0	1.0	0%
<b>Total</b>	<b>89</b>	<b>350</b>	<b>0</b>	<b>3</b>	<b>1</b>	<b>89.0</b>	<b>442.0</b>	<b>20%</b>

**Note:** Actual scores are subject to rounding errors. The sum of these values may not result in the exact overall score.

## Profiles

This benchmark contains 10 profiles. The **Level 2 (L2) - High Security/Sensitive Data Environment (limited functionality)** profile was used for this assessment.

**Figure 40. W10 Pro (2022-2023 academic year image) audit results with CIS-CAT Lite for L2 profile**

## 7.4.4 Results' summary

Since comparing the results can be difficult just by looking at the reports individually, some summary tables were produced. It has been explicitly indicated whether the machines have been provisioned or not; the ones that have are those that result from the implementation of the work developed in this project.

### 7.4.4.1 Windows 10 (Education and Pro machines)

The following tables summarize the results obtained with CLARA and CIS-CAT Lite, respectively. As it can be seen, the tables include the “intended operational environment level” for the machines; this is meant to refer to the environment category in which each of them, given their current provisioning level, are intended to be operated. This information has been deliberately left blank for the University machines, as the student did not want to make any uninformed guesses.

CLARA AUDIT RESULTS						
Operating System	Version	CIS compliance information			System category	COMPLIANCE SCORE (%)
		Provisioned?	Provisioning level	Intended operational environment level		
Windows 10 Education	21H2	No	-	L1	LOW (Baja)	56,12
Windows 10 Education	21H2	No	-	L1	MEDIUM (Media)	53,42
Windows 10 Education	21H2	No	-	L1	HIGH (Alta)	56,40
<b>Windows 10 Education</b>	<b>21H2</b>	<b>Yes</b>	<b>L1 tasks only</b>	<b>L1</b>	<b>LOW (Baja)</b>	<b>84,43</b>
<b>Windows 10 Education</b>	<b>21H2</b>	<b>Yes</b>	<b>L1 tasks only</b>	<b>L1</b>	<b>MEDIUM (Media)</b>	<b>77,56</b>
<b>Windows 10 Education</b>	<b>21H2</b>	<b>Yes</b>	<b>L1 tasks only</b>	<b>L1</b>	<b>HIGH (Alta)</b>	<b>77,11</b>
<b>Windows 10 Education</b>	<b>21H2</b>	<b>Yes</b>	<b>L1 tasks + L2 tasks</b>	<b>L2</b>	<b>LOW (Baja)</b>	<b>84,56</b>
<b>Windows 10 Education</b>	<b>21H2</b>	<b>Yes</b>	<b>L1 tasks + L2 tasks</b>	<b>L2</b>	<b>MEDIUM (Media)</b>	<b>78,48</b>
<b>Windows 10 Education</b>	<b>21H2</b>	<b>Yes</b>	<b>L1 tasks + L2 tasks</b>	<b>L2</b>	<b>HIGH (Alta)</b>	<b>77,79</b>



Windows 10 Pro ('21-'22)	20H2	No	-	-	LOW (Baja)	29,71
Windows 10 Pro ('21-'22)	20H2	No	-	-	MEDIUM (Media)	31,24
Windows 10 Pro ('21-'22)	20H2	No	-	-	HIGH (Alta)	36,44
Windows 10 Pro ('22-'23)	21H2	No	-	-	LOW (Baja)	50,02
Windows 10 Pro ('22-'23)	21H2	No	-	-	MEDIUM (Media)	50,64
Windows 10 Pro ('22-'23)	21H2	No	-	-	HIGH (Alta)	53,94

**Table 7. Windows 10 (Education and Pro) compliance scores with CLARA**

CIS-CAT Lite AUDIT RESULTS						
Operating System	Version	CIS compliance information			Assessment	SCORE (%)
		Provisioned?	Provisioning level	Intended operational environment level		
Windows 10 Education	21H2	No	-	L1	Windows 10 Enterprise Benchmark v1.12.0, Level 1 (L1) Corporate/Enterprise Environment (general use)	23,00
Windows 10 Education	21H2	No	-	L1	Windows 10 Enterprise Benchmark v1.12.0, Level 2 (L2) High Security/Sensitive Data Environment (limited functionality)	19,00
Windows 10 Education	21H2	Yes	L1	L1	Windows 10 Enterprise Benchmark v1.12.0, Level 1 (L1) Corporate/Enterprise Environment (general use)	56,00
Windows 10 Education	21H2	Yes	L1	L1	Windows 10 Enterprise Benchmark v1.12.0, Level 2 (L2) High Security/Sensitive Data Environment (limited functionality)	45,00
Windows 10 Education	21H2	Yes	L2	L2	Windows 10 Enterprise Benchmark v1.12.0, Level 1 (L1) Corporate/Enterprise Environment (general use)	56,00
Windows 10 Education	21H2	Yes	L2	L2	Windows 10 Enterprise Benchmark v1.12.0, Level 2 (L2) High Security/Sensitive Data Environment (limited functionality)	46,00
Windows 10 Pro ('21-'22)	20H2	No	-	-	Windows 10 Enterprise Benchmark v1.12.0, Level 1 (L1) Corporate/Enterprise Environment (general use)	25,00
Windows 10 Pro ('21-'22)	20H2	No	-	-	Windows 10 Enterprise Benchmark v1.12.0, Level 2 (L2) High Security/Sensitive Data Environment (limited functionality)	20,00
Windows 10 Pro ('22-'23)	21H2	No	-	-	Windows 10 Enterprise Benchmark v1.12.0, Level 1 (L1) Corporate/Enterprise Environment (general use)	25,00
Windows 10 Pro ('22-'23)	21H2	No	-	-	Windows 10 Enterprise Benchmark v1.12.0, Level 2 (L2) High Security/Sensitive Data Environment (limited functionality)	20,00

**Table 8. Windows 10 (Education and Pro) compliance scores with CIS-CAT Lite**

### 7.4.4.2 Ubuntu Linux

LYNIS AUDIT RESULTS			
Operating System	Version	CIS compliance information	COMPLIANCE SCORE (%)
		Provisioned?	
Ubuntu Linux	18.04	No	56,00
<b>Ubuntu Linux</b>	<b>18.04</b>	<b>Yes</b>	<b>82,00</b>

**Table 9. Ubuntu Linux compliance scores with Lynis**

### 7.4.5 Analysis of the results

As it can be concluded from the obtained results, the scores have improved dramatically from the ones obtained by the base machines to the hardened ones, even if they have their security improved just by applying Level 1 recommendations; for example, the Linux machine increments its security score in 30 points (from an average 56 to the much greater 82), whilst the Windows machine is able to more than duplicate its CIS compliance score (from 23% to 56%) and jump to over 75% consistently against the ENS, even in high security environments.

Also, it must be noted that some of the implemented tasks had to have their recommended values modified or their application restricted because they posed an issue for the communication between the controller and the controlled node; this means that results can potentially be better, though with the planned infrastructure it was not possible to check it.

However, there is still room for improvement, particularly for achieving CIS compliance; even though the scores with CIS-CAT Lite were more than doubled, there are still a high number of basic security recommendations that can be applied in the future, and that can boost those results. Nevertheless, this would require a detailed study for checking no vital feature or service gets deactivated or removed.

## 7.5 EXECUTION OF ADDITIONAL TESTS

PowerShell Access Restriction (Windows 10)			
Test	Actual Result	¿Expected == Actual?	Actions taken/Comments
<b>Indicate PowerShell needs to</b>	PowerShell gets disabled for all users, instead of	No	As PowerShell access is needed for Windows provisioning via Ansible, tasks that try to run after this will fail, because the user for





**be disabled (restricted access)**

for standard, non-administrator users only.

connecting to the machine loses access to this resource.

This may be happening because by default local administrators belong to both the Administrators and the Users group, and by extent get blocked from being able to access PowerShell. Tests for removing the default local administrator user from the Users group to bypass this issue were carried out, though this didn't prove to be a solution to the problem.

A conditional variable for running or not the specific Ansible playbook was placed in the Vagrantfile, to control whether the controller should execute this code or not. This is just a workaround, though. The best way to restrict access to the PowerShell executables is via GUI

#### Configure local administrators (Windows 10)

Test	Actual Result	¿Expected==Actual?	Actions taken/Comments
<b>Take out default machine user from being Local Administrator</b>	The default machine user gets taken out of the 'Administrators' group and loses administrative privileges	Yes	As administrative privileges are needed for Windows provisioning via Ansible, tasks running after this will fail

#### Local administrators' password change (Windows 10)

Test	Actual Result	¿Expected==Actual?	Actions taken/Comments
<b>Change password for the default machine user</b>	The password of the default machine user is changed to the new one	Yes	SSH communications are still possible because no password authentication is used

## 7.6 CSI 6: ELABORACIÓN DE LOS MANUALES DE USUARIO

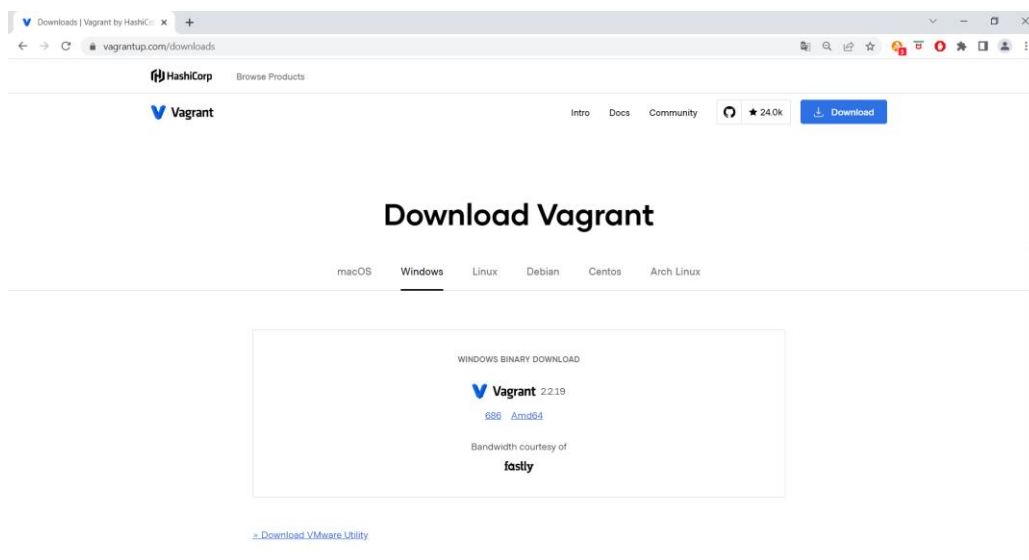
### 7.6.1 Installation and Execution Manual

This project is comprised of several different parts, and not everything could be included in the project repository for size reasons. This manual intends to explain in detail how to get the project up and running, from the initial hardware and software requirements to the actual process of hardening the provided machines.

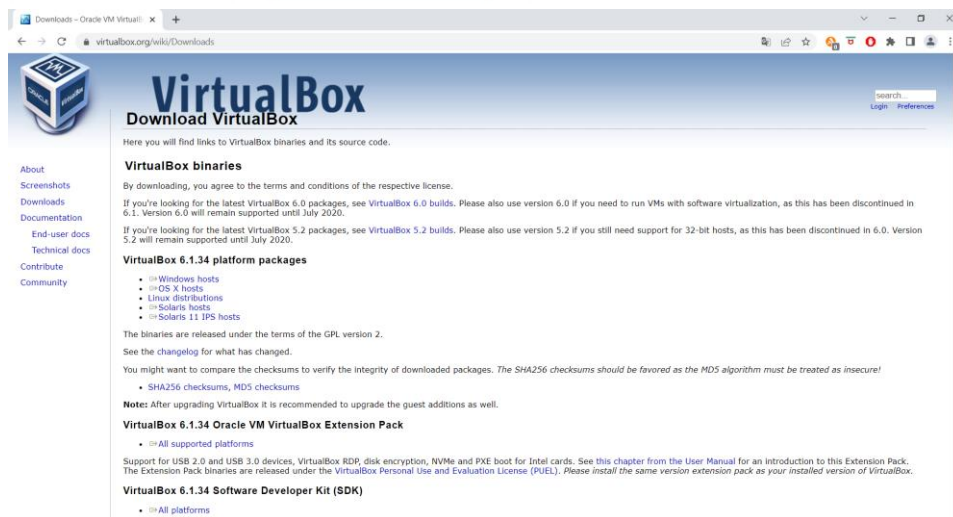
#### 7.6.1.1 Initial requirements

##### 7.6.1.1.1 Software requirements

For executing this project, the user will need to have installed in his or her computer, at least, Vagrant and VirtualBox. Both are available for several different platforms, so users will simply need to navigate to the official downloads site (<https://www.vagrantup.com/downloads> and <https://www.virtualbox.org/wiki/Downloads> respectively) and choose the version that fits their operating system. For some systems, it is possible to install the tools using a package manager, in which case both sites have detailed explanations on how to do so; for the rest, it is as simple as following the steps of the corresponding setup wizard.



*Figure 41. Vagrant's downloads page*



**Figure 42. VirtualBox downloads page**

### 7.6.1.1.2 Hardware requirements

Some of the machines deployed as part of this project have a considerable size, particularly the Windows 10 ones. It is crucial to check that the computer where the project will be deployed has enough available memory.

For reference, the Windows machines require at least, when booting, 30 GB of memory. Additionally, the custom base box they are brought up against can take up to 15 GB. It is recommended that, if not enough memory is available on the host, an external drive is used to store the machines.

Regarding other settings, such as the video memory or the maximum RAM the machine will use, they can be configured by modifying the parameters in the Vagrantfiles or the external configuration files (see *Modifying the machines' configuration*) to fit the host's specific needs or limitations.

### 7.6.1.2 Get the boxes for the machines

The Ubuntu Linux machines use a box from Vagrant's public catalogue; as such, once one of these machines is booted up, Vagrant will automatically try to find the box in the user's local cache of boxes and, if it isn't found, it will download it from the catalogue.

However, the custom Windows boxes this project uses have not been uploaded to any catalogue, so they will need to be generated and added to the local cache manually. All the necessary files for creating the boxes from scratch, minus the packer.exe executable, can be found in the /packer folder of the project; however, it can be a tedious process, so the machines have been generated and can be found in the following link: [VAGRANT BOXES](#). Each of the directories present in the shared folder contain custom Vagrant boxes for Spanish and English machines; the desired box can be downloaded to the host.

Mis archivos > TFG > VAGRANT BOXES

Nombre	Modificado	Modificado por	Tamaño de archi...	Compartir
Windows 10 Education Span...	16 de junio	María Flórez Miranda	2 elementos	Compartido
Windows 10 Education English	18 de junio	María Flórez Miranda	2 elementos	Compartido
Windows 10 Education English - old	18 de junio	María Flórez Miranda	1 elemento	Compartido

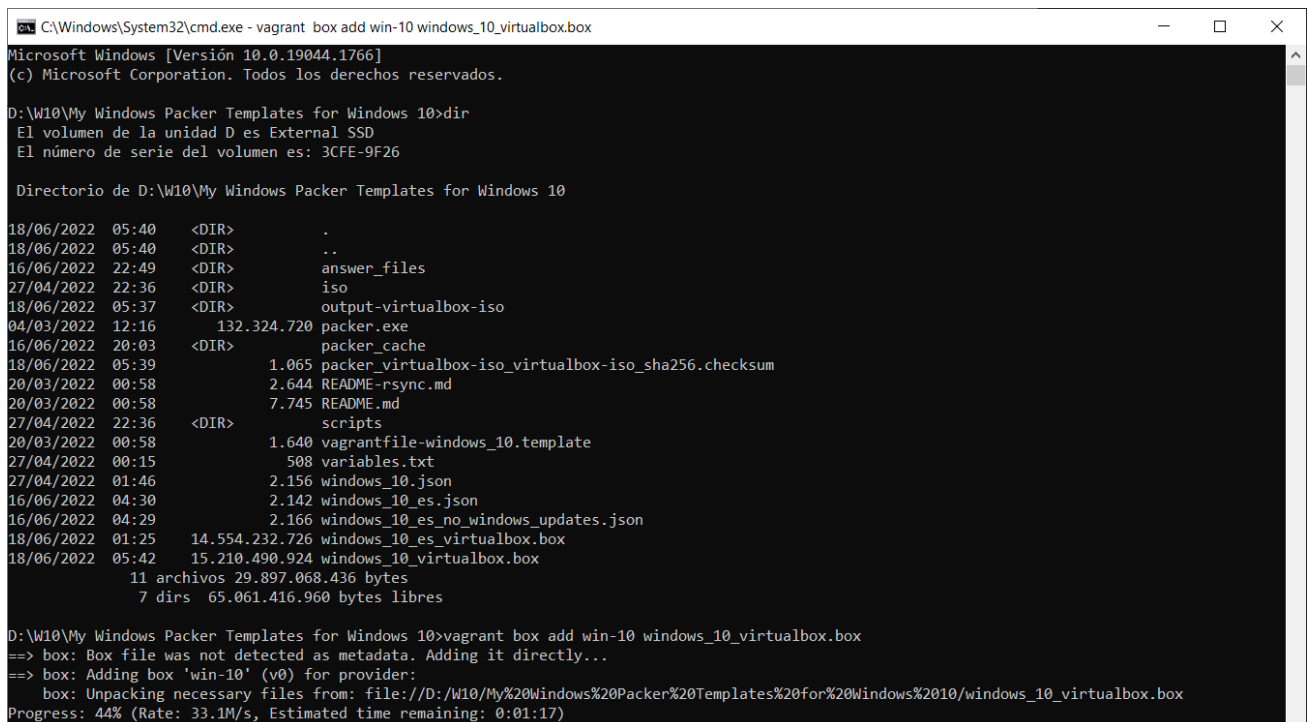
**Figure 43. Detail of the folders for each of the Windows custom boxes**

Once the box has been downloaded, it needs to be added to the local cache of boxes. For doing this, the terminal application needs to be opened in the same directory where the box is; then, run the following command:

```
vagrant box add <name we want to give to the box> <path of the box>
```

The process can take some time to complete; also, the memory of the host will be seen decreasing rapidly. It is not something to worry about, as when the box finishes being added to the cache, the memory will stabilize again.

An example of the command being run can be seen below:



```
C:\Windows\System32\cmd.exe - vagrant box add win-10 windows_10_virtualbox.box
Microsoft Windows [Versión 10.0.19044.1766]
(c) Microsoft Corporation. Todos los derechos reservados.

D:\W10\My Windows Packer Templates for Windows 10>dir
El volumen de la unidad D es External SSD
El número de serie del volumen es: 3CFE-9F26

Directorio de D:\W10\My Windows Packer Templates for Windows 10

18/06/2022 05:40 <DIR> .
18/06/2022 05:40 <DIR> ..
16/06/2022 22:49 <DIR> answer_files
27/04/2022 22:36 <DIR> iso
18/06/2022 05:37 <DIR> output-virtualbox-iso
04/03/2022 12:16 132.324.720 packer.exe
16/06/2022 20:03 <DIR> packer_cache
18/06/2022 05:39 1.065 packer_virtualbox-iso_virtualbox-iso_sha256.checksum
20/03/2022 00:58 2.644 README-rsync.md
20/03/2022 00:58 7.745 README.md
27/04/2022 22:36 <DIR> scripts
20/03/2022 00:58 1.640 vagrantfile-windows_10.template
27/04/2022 00:15 508 variables.txt
27/04/2022 01:46 2.156 windows_10.json
16/06/2022 04:30 2.142 windows_10_es.json
16/06/2022 04:29 2.166 windows_10_es_no_windows_updates.json
18/06/2022 01:25 14.554.232.726 windows_10_es_virtualbox.box
18/06/2022 05:42 15.210.490.924 windows_10_virtualbox.box
11 archivos 29.897.068.436 bytes
7 dirs 65.061.416.960 bytes libres

D:\W10\My Windows Packer Templates for Windows 10>vagrant box add win-10 windows_10_virtualbox.box
==> box: Box file was not detected as metadata. Adding it directly...
==> box: Adding box 'win-10' (v0) for provider:
    box: Unpacking necessary files from: file:///D:/W10/My%20Windows%20Packer%20Templates%20for%20Windows%2010/windows_10_virtualbox.box
Progress: 44% (Rate: 33.1M/s, Estimated time remaining: 0:01:17)
```

**Figure 44. Example of a run for adding a custom box to the local cache of boxes**

Once the process completes, the box will be available in the local cache and available for Vagrant.



### 7.6.1.3 Executing the project

This project contains several Vagrantfiles, all within their respective folder inside the /vagrantfiles directory:

- linux/debian/ubuntu contains the following machines:
  - base\_ubuntu: a basic Ubuntu Linux 18.04 machine without hardening
  - base\_ubuntu\_hardened: a basic Ubuntu Linux 18.04 machine and an Ansible controller for running the hardening code against the Ubuntu machine
  - magic\_the\_hardening: an already hardened machine (Ubuntu Linux 18.04) and an Ansible controller for further hardening the machine. This machine is a modification of the original “Magic The Hardening” machine that can be found in the following repository: [https://github.com/jose-lopez/SSI\\_Infraestructure\\_Automation\\_Materials](https://github.com/jose-lopez/SSI_Infraestructure_Automation_Materials). Its author has granted permission to use it.
- windows/windows10-education contains the following machines:
  - base\_windows: a base Windows 10 Education machine without hardening
  - windows\_hardened: a base Windows 10 Education machine and an Ansible controller for running the hardening code against it

For booting up a specific machine or set of machines, simply navigate to its directory, open the terminal there and execute the following command:

```
vagrant up
```

This will bring up all machines declared in the Vagrantfile, in a sequential order. If a specific machine is wanted to boot, its name must be specified as follows when calling the command: `vagrant up <machine name or id>`.

```
C:\Windows\System32\cmd.exe - vagrant up
ed>dir
El volumen de la unidad C es Windows-SSD
El número de serie del volumen es: 4486-085D

Directorio de C:\Users\maria\Desktop\TFG\REPO\step-by-step-2\vagrant-ansible-hardening\vagrantfiles\windows\windows10-education\windows_hardened

25/06/2022  20:29    <DIR>          .
25/06/2022  20:29    <DIR>          ..
10/06/2022  02:27    <DIR>          .vagrant
10/06/2022  00:57             1.702 insecure_private_key
25/06/2022  21:12             78.045 log_vagrant_up.txt
11/06/2022  02:35    <DIR>          provisioning
05/07/2022  01:28             910 README.md
25/06/2022  06:05    <DIR>          reports
19/06/2022  02:24             3.185 Vagrantfile
19/06/2022  23:47             55 vagrant_package_windows_hardened_box.bat
23/06/2022  01:45             55 vagrant_package_windows_hardened_box.sh
10/07/2022  01:31             146 windows_machine.yml
25/06/2022  05:57    <DIR>          windows_shared
              7 archivos          84.098 bytes
              6 dirs  21.998.825.472 bytes libres

C:\Users\maria\Desktop\TFG\REPO\step-by-step-2\vagrant-ansible-hardening\vagrantfiles\windows\windows10-education\windows_harden
ed>vagrant up
Bringing machine 'windows10' up with 'virtualbox' provider...
Bringing machine 'controller' up with 'virtualbox' provider...
==> windows10: Importing base box 'win-10'...
==> windows10: Matching MAC address for NAT networking...
==> windows10: Setting the name of the VM: windows10
==> windows10: Clearing any previously set network interfaces...
==> windows10: Preparing network interfaces based on configuration...
windows10: Adapter 1: nat
windows10: Adapter 2: hostonly
==> windows10: Forwarding ports...
windows10: 3389 (guest) => 3389 (host) (adapter 1)
windows10: 22 (guest) => 2222 (host) (adapter 1)
windows10: 5985 (guest) => 55985 (host) (adapter 1)
windows10: 5986 (guest) => 55986 (host) (adapter 1)
==> windows10: Running 'pre-boot' VM customizations...
==> windows10: Booting VM...
```

**Figure 45. Example of the Windows machine and the controller booting up**

In the case of the machines that get hardened, once the controller boots, the Ansible provisioner will start running, and logs will start appearing in the terminal, detailing the tasks that are running and their results. Do not touch this terminal nor interrupt the process; once it finishes, control of the command line will be regained. An example of how an Ansible run is started and then completed can be seen below:

```
==> controller: Detected mount group ID within mount options. (gid: 1000 guestpath: /vagrant)
==> controller: Running provisioner: file...
controller: ../../../../provisioners/ansible/roles/windows => /vagrant/provisioning/roles/windows
==> controller: Running provisioner: ansible_local...
controller: Installing Ansible...
controller: Running ansible-playbook...
cd /vagrant && PYTHONUNBUFFERED=1 ANSIBLE_NOCOLOR=true ANSIBLE_CONFIG='provisioning/ansible.cfg' ansible-playbook --limit="windows"
--inventory-file=provisioning/inventory -v provisioning/roles/windows/windows_provisioning.yml
Using /vagrant/provisioning/ansible.cfg as config file

PLAY [Provisioning roles for Windows workstations and servers] *****

TASK [Gathering Facts] *****
ok: [windows10]

TASK [compliance/cis : Determine Language settings] *****
changed: [windows10] => {"changed": true, "cmd": "Get-WinSystemLocale | select name", "delta": "0:00:01.249907", "end": "2022-07-10 01:46:25.498530", "rc": 0, "start": "2022-07-10 01:46:24.248622", "stderr": "", "stderr_lines": [], "stdout": "\r\nName \r\n-- \r\nen-US\r\n\r\n\r\n", "stdout_lines": [""], "Name ", "---- ", "en-US", "", ""}]

TASK [compliance/cis : Include variables for OS version (ES)] *****
skipping: [windows10] => (item=/vagrant/provisioning/roles/windows/compliance/cis/vars/Windows_es.yml) => {"ansible_loop_var": "item", "changed": false, "item": "/vagrant/provisioning/roles/windows/compliance/cis/vars/Windows_es.yml", "skip_reason": "Conditional result was False"}
```

Figure 46. Starting the Ansible provisioning

```
changed: [windows10] => {"changed": true, "description": "Local administrator account", "fullname": "vagrant", "groups": [{"name": "Administrators", "path": "WinNT://WORKGROUP/VAGRANT-10/Administrators"}, {"name": "Users", "path": "WinNT://WORKGROUP/VAGRANT-10/Users"}], "name": "vagrant", "password_expired": false, "password_never_expires": true, "path": "WinNT://WORKGROUP/VAGRANT-10/vagrant", "sid": "S-1-5-21-308831880-3784153787-4016506613-1000", "state": "present", "user_cannot_change_password": false}

PLAY RECAP *****
windows10 : ok=219 changed=146 unreachable=0 failed=0 skipped=7 rescued=0 ignored=0
```

Figure 47. A finished Ansible run

#### 7.6.1.4 Packaging the machines

Once the machines are completely set up, there exists the possibility of exporting them so they can be used as “base” boxes for creating new machines. This can be done via the following command:

```
vagrant package --output <new name for the box>.box <name of the machine to export>
```

As the result of this is a box, it can then be added to the local cache as before (check *Get the boxes for the machines*) or uploaded to a box catalogue.

#### 7.6.1.5 Stopping and destroying machines

To stop and shut down a specific machine, run the following command:

```
vagrant halt [name or id of the machine]
```

This will make Vagrant attempt to first gracefully shut down the machine; however, if it fails, then the machine will be powered-off.

Finally, to remove the machine or machines from the host, run the following command:

```
vagrant destroy
```

This will stop all running machines and destroy every resource created along the machine's set up process. To destroy only a specific machine, its name must be specified when calling the command: `vagrant destroy <machine name or id>`.

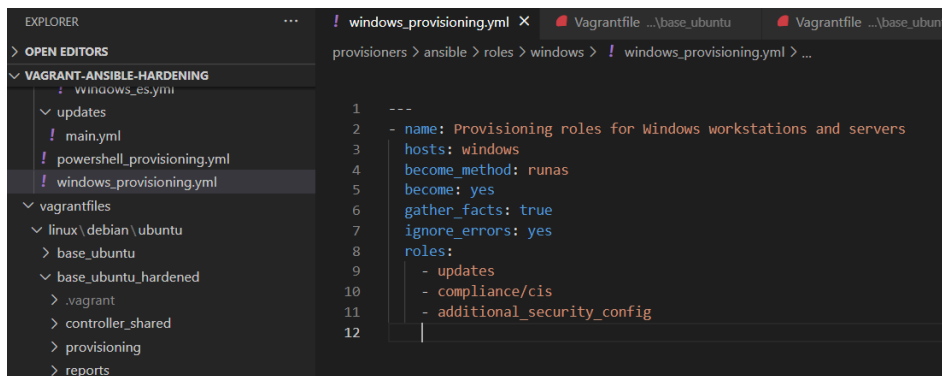
## 7.6.2 Programmer's Manual

This manual is thought for programmers who want to extend or modify some items of this system.

### 7.6.2.1 Adding or modifying new provisioning Ansible tasks

For extending the Ansible code or modifying it, the programmer must have at least some basic knowledge of Ansible.

There are two different folders within the `/provisioners/ansible/roles` directory of the project, which is the one containing all Ansible code. Each of the folders is expected to include all the scripts for the roles of each type of machine. For adding new tasks or modifying the ones that already exist, locate the `.yaml` file where they will or are placed; if a new `.yaml` file needs to be created, it must be referenced in `main.yaml` (`tasks/main.yaml` or simply `main.yaml`) as this will be the file Ansible will look for when provisioning. Additionally, if a new whole role is created, you will need to include it in the playbook to be called, such as in this example from `windows_provisioning.yaml`, which references all the roles that are executed by default:



```
1 ---
2 - name: Provisioning roles for Windows workstations and servers
3   hosts: windows
4   become_method: runas
5   become: yes
6   gather_facts: true
7   ignore_errors: yes
8   roles:
9     - updates
10    - compliance/cis
11    - additional_security_config
```

**Figure 48.** Example of a playbook with several roles

It is recommended that all variables that are subject to change are placed in external files inside the `vars` folder of each role; this also makes it easier to modify any current value, as well as to reuse the same value in several places. An example showing how some of the variables in one of these files appear can be seen below:



```
81 # PAM, password and account settings
82 minlen: 14
83 minclass: 4
84 retry: 3
85 lcredit: -1
86 ucredit: -1
87 dcredit: -1
88 ocredit: -1
89 difok: 5
90 pwd_remember: 5
91 pwd_max_days: 365
92 pwd_min_days: 1
93 pwd_warn_age: 7
94 max_failed_login_attempts: 5
95 unlock_timeout: 600
96 user_inactivity: 30
97 default_umask: '027'
```

**Figure 49. Example of variables for password settings**

Finally, it is worth noting that it is not mandatory to follow the structure this project uses for placing the files, just the recommended one. However, in those cases the programmer must make sure he or she is configuring the path of the files correctly when invoking the playbook.

### 7.6.2.2 Modifying the machines' configuration

Vagrant allows including variables in the Vagrantfiles as a means of storing values that can then be used elsewhere by referencing their name; they are particularly useful for parameterizing the configuration of a box. A step further is to externalize all the variables, which keeps the Vagrantfile clean of data that is subject to change depending on the environment needs.

In the project, some machines have been created following the external .yml file approach. The Windows 10 machine that gets hardened has some basic configuration for booting up placed in a windows\_machine.yml file in the root of the /vagrantfiles/windows/windows10-education/windows\_hardened folder; adding or modifying values is straightforward. A summary can be seen below:

Name	Description	Example
name	The name identifying the machine	windows10 ubuntu
box	The specific box against which the machine will be brought up, which can either be from the local cache or from a public catalogue	win-10 hashicorp/bionic64
ram	The amount of RAM (in MB) for the machine	4096
cores	The number of cores for the machine	2
ip	The IP of the machine; in the case of the hardened Windows 10, it is for the internal network	172.17.177.21
disable_powershell	Conditional flag for running or not the playbooks for restricting access to PowerShell	False True



vram	The amount of Video Memory for the machine (in MB). Configure it considering if the machine will have a GUI or not	128
------	--	-----

**Table 10. Parameters that can be configured for the Windows 10 machine**



# CHAPTER 8: CONCLUSIONS AND EXTENSIONS





## 8.1 CONCLUSIONS

---

The execution and completion of this project allowed arriving at some important conclusions.

First, the objectives of the project, both the intermediate and the final ones, were achieved. It was possible to completely automate the installation of a Windows machine, which in itself can be a challenge, both when creating the necessary files and when setting up the machine. Being able to export the result of this installation process so that it could then be used as the base for any Windows machine was also a huge milestone; in fact, at the beginning it was not clear if this whole objective could be really achieved or not.

Another core part of this project, the hardening phase, can also be considered quite successful. First of all, it was proven that both Windows and Linux machines can have their security dramatically improved and their compliance level against official, international standards increased; and this can be done in a fully automated way. Not only that, but all the recommendations that have been adapted into code can be easily extended, modified, and distributed so that they can be used for different, varied environments. It is not difficult to arrive to the conclusion that, from this, the next steps would be to try and replicate what has been done over virtual machines on real, physical ones.

On a personal level, I feel very glad that the project took such a turn to become what it is today. Despite all the issues that emerged along the way, the end result was worth the effort, and what is most important, I feel motivated to keep improving and evolving the current project, and continue learning about all the new technologies and standards I was able to start working with. I can undoubtedly say that the whole journey this project has entailed has made me grow in experience, in curiosity and in self-discipline.

I've also been really interested in Cybersecurity in the past years, and having the possibility to contribute with my work to this field, even if only a little, has been extremely rewarding. This has further reinforced my determination to keep expanding my knowledge in this area, and eventually become a professional dedicated to this specific sector. Finally, the encouragement from the director of this project and the potential this work has have made me proud of the produced output, even if it only represents a small step towards something that could be much bigger in the future.



## 8.2 EXTENSIONS

---

One of the possible extensions of this project is to implement more CIS controls, by first studying the benchmark guides more thoroughly. This project implements a fairly vast set of the controls, that have proven to greatly improve the compliance level with respect to ENS and CIS; however, some of the benchmark's remediations have been left out for the moment, particularly those pertaining to Windows. The reason for this is that it requires expert knowledge on administration, as well as the consensus of those organizations where the policies are intended to be applied, as some of these policies may not align to the organizations' needs.

Apart from implementing more CIS controls, it could be interesting to study and include policies from other benchmarks or sources that could complement the CIS ones. However, this should be done carefully, ensuring they do not clash with one another. If the University seeks to be compliant with the ENS, a good starting point could be considering what ENS-specific controls could be implemented to achieve higher compliance scores and ultimately obtain the certification.

In addition to making more policies available, one of the key points this project aims to tackle in the future is the operating system support. For the moment, Windows-wise the only version that is *directly* supported is Windows 10 Education, since it was the ISO available for testing. As the University's computers use Windows 10 Pro, it could be interesting to extend the support for Pro, though this would need some careful study as well to check if any of the controls needs to be adapted for this OS version.

Finally, one future goal of the project is to be able to implement hardened machines in physical labs, either by generating the images beforehand with all security recommendations in place, or by provisioning the already installed machines with all the recommended settings. Though some talks on this matter have already taken place, doing this will require some more time and expert administrator knowledge, to make sure no vital service or feature is affected and all other changes comply with the organization's policies.



# ANNEXES





## RISK MANAGEMENT PLANNING

During the project planning, it was considered that some risks could appear during the whole lifecycle of the project; as such, the following Risk Management Planning was devised, to ensure the risks are mitigated as much as possible.

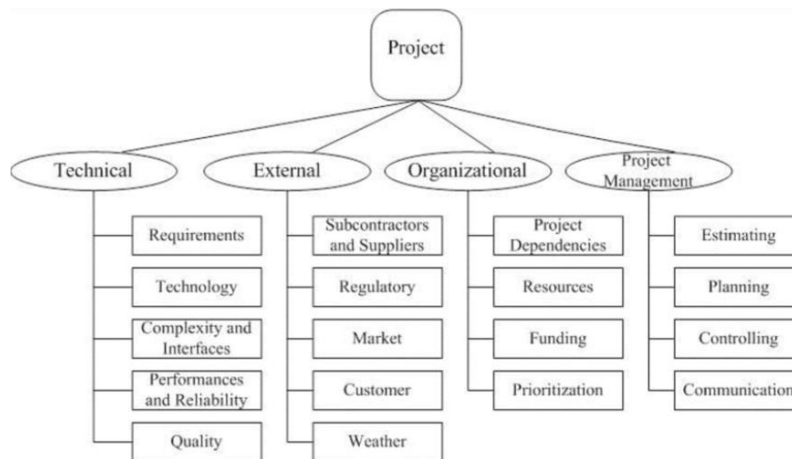
### Methodology

The proposed methodology for this project is the BOEHM model, in which risk management gets divided into two main phases:

1. Risk valuation: identification, analysis and prioritization of the system's risks
2. Risk control: the strategies that will be applied to the risks, along with monitorization

### Risk categories

The risk categories used are the ones belonging to PMBOK. This structure, which decomposes the risks in a hierarchical way, is known as Risk Breakdown Structure, and could be adapted to the structure of this project if needed.



**Figure 50. Risk categories according to PMBOK**

### Probability and Impact Matrix

The Probability and Impact Matrix used for prioritizing the risks of the project is the following:

<b>Probability</b>	<b>Very High</b>	<b>0,90</b>	0,05	0,14	0,27	0,50	0,81
	<b>High</b>	<b>0,70</b>	0,04	0,11	0,21	0,39	0,63
	<b>Medium</b>	<b>0,50</b>	0,03	0,08	0,15	0,28	0,45
	<b>Low</b>	<b>0,30</b>	0,02	0,05	0,09	0,17	0,27
	<b>Very Low</b>	<b>0,10</b>	0,01	0,02	0,03	0,06	0,09
			<b>0,05</b>	<b>0,15</b>	<b>0,30</b>	<b>0,55</b>	<b>0,90</b>
			<b>Negligible</b>	<b>Low</b>	<b>Medium</b>	<b>High</b>	<b>Critical</b>
			<b>0,05</b>	<b>0,15</b>	<b>0,30</b>	<b>0,55</b>	<b>0,90</b>
			<b>Impact</b>				

*Figure 51. Probability and Risk Matrix*

## BUDGET PLANNING

As it has already been said, this project was successfully planned and completed without having any direct cost, as all licenses and technologies used were sought to be free of charge or open source. However, this section shows the full budget breakdown if all assets were to be quantified, as what would happen if this project would need to be put into production and/or developed by dedicated teams.

### Personnel

The roles identified for carrying out this project are the ones shown in the table below:

Personnel			
Role	Months	Salary/month	Total cost
Analyst	1	2.000,00 €	2.000,00 €
Programmer	4	1.500,00 €	6.000,00 €
Auditor	2	2.500,00 €	5.000,00 €
<b>TOTAL</b>			<b>13.000,00 €</b>

These will be the roles the student identifies with during the different phases of the project, which in some cases will be more than one at the same time, such as during the Machine Hardening phase. In this point in time, the student will be performing both roles simultaneously, as that specific part of the project requires the developer to receive constant feedback from the audits.

For the final execution of the project, though the planning suffered modifications, the roles undertaken by the student remained the same, for the same duration of time.

## Licenses

These are the licenses that have been used in this project; as it can be seen, only the Windows 10 license has been considered, as the rest of the software used is free of charge, including the IDEs and other tools.

Licenses				
Software	Quantity	Price	Usage time / months	Total cost
Windows 10 Enterprise/Education	1	260,00 €		260,00 €
TOTAL				260,00 €

## Material Resources

These are the material resources that have been used for developing this project, from the initial planning phases to the auditing ones. The amortization factor has been calculated taking this into account, as the total hours considered are the total hours of the duration of this project.

Material Resources							
Description	Units	Lifespan	Type	Amortization factor (%) (hours project/lifespan hours)	Price	Total price	Total after amortization
Lenovo Laptop	1	5 years	amortization	0,85%	799,00 €	799,00 €	6,79 €
External SSD Drive	1	5 years	amortization	0,85%	70,00 €	70,00 €	0,59 €
USB Drives	2	7 years	amortization	0,61%	10,00 €	20,00 €	0,12 €
External HDD Drive	1	7 years	amortization	0,61%	50,00 €	50,00 €	0,30 €
TOTAL							7,81 €

## Indirect costs

These are the indirect costs of the project, which shows all the indirect expenses a project like this generates. All the prices have been calculated taking into account real values during the time period this project was in development.

Indirect costs			
Description	Unit (months)	Price/month	Total cost
Electricity	5	80,00 €	400,00 €
Internet	5	30,00 €	150,00 €
Office space	5	400,00 €	2.000,00 €
Office supplies	5	15,00 €	75,00 €
TOTAL			2.625,00 €

## Initial Budget Summary

After considering the cost of each of the individual items, the initial budget can be summarized as follows:

Initial Budget Summary		
Item	Description	Total cost
1	Personnel	13.000,00 €
2	Licenses	260,00 €
3	Material Resources	7,81 €
4	Indirect Costs	2.625,00 €
TOTAL		15.892,81 €

## Final Budget Summary

Regarding the final budget, it changed from what was considered at the beginning of the project, since no licenses had to be purchased. As such, the corresponding item was taken out of the summary, whose table can be seen below:

Initial Budget Summary		
Item	Description	Total cost
1	Personnel	13.000,00 €
2	Licenses	0,00 €
3	Material Resources	7,81 €
4	Indirect Costs	2.625,00 €
TOTAL		15.632,81 €



## REFERENCES AND BIBLIOGRAPHY

---

- [1] J. M. Redondo, «Documentos-modelo para Trabajos de Fin de Grado/Master de la Escuela de Informática de Oviedo,» 17 6 2019. [En línea]. Available: [https://www.researchgate.net/publication/327882831\\_Plantilla\\_de\\_Proyectos\\_de\\_Fin\\_de\\_Carrera\\_de\\_la\\_Escuela\\_de\\_Informatica\\_de\\_Oviedo](https://www.researchgate.net/publication/327882831_Plantilla_de_Proyectos_de_Fin_de_Carrera_de_la_Escuela_de_Informatica_de_Oviedo).
- [2] J. Redondo, «Creación y evaluación de plantillas para trabajos de fin de grado como buena práctica docente.,» *Revista de Innovación y Buenas Prácticas Docentes*, p. pp, 2020.
- [3] M. Fowler, «Domain-Specific Languages Guide,» 28 August 2019. [En línea]. Available: [https://martinfowler.com/dsl.html#:~:text=A%20Domain%2DSpecific%20Language%20\(DSL,as%20computing%20has%20been%20done](https://martinfowler.com/dsl.html#:~:text=A%20Domain%2DSpecific%20Language%20(DSL,as%20computing%20has%20been%20done). [Último acceso: 30 June 2022].
- [4] Red Hat, «What is Infrastructure as Code?,» 11 May 2022. [En línea]. Available: <https://www.redhat.com/en/topics/automation/what-is-infrastructure-as-code-iac>. [Último acceso: 26 June 2022].
- [5] Red Hat, «What is Virtualization?,» 2 March 2018. [En línea]. Available: <https://www.redhat.com/en/topics/virtualization/what-is-virtualization>. [Último acceso: 30 June 2022].
- [6] Centro Criptológico Nacional, «ENS - FAQ,» 2022. [En línea]. Available: <https://ens.ccn.cni.es/es/esquema-nacional-de-seguridad-ens/faq-ens>. [Último acceso: 2 July 2022].
- [7] O. Ramírez, «Provisioning Virtual Machines with Vagrant and Ansible,» 6 December 2020. [En línea]. Available: <https://orlando-ramirez.com/2020/12/06/provisioning-virtual-machines-with-ansible-and-vagrant/>. [Último acceso: 23 March 2022].
- [8] Centro Criptológico Nacional, «CLARA,» [En línea]. Available: <https://www.ccn-cert.cni.es/soluciones-seguridad/clara.html>. [Último acceso: 7 July 2022].
- [9] Microsoft Corporation, «Unattended Windows Setup Reference (full),» 24 June 2021. [En línea]. Available: <https://docs.microsoft.com/en-us/windows-hardware/customize/desktop/unattend/>. [Último acceso: 4 April 2022].
- [10] Centro Criptológico Nacional, «Guías CCN-STIC,» [En línea]. Available: <https://www.ccn.cni.es/index.php/es/menu-guias-ccn-stic-es>. [Último acceso: 28 June 2022].
- [11] HashiCorp, «Vagrant Documentation,» [En línea]. Available: <https://www.vagrantup.com/docs>. [Último acceso: 9 July 2022].
- [12] Microsoft Corporation, «Windows Security documentation,» [En línea]. Available: <https://docs.microsoft.com/en-us/windows/security/>. [Último acceso: 1 July 2022].



- [13] Microsoft Corporation, «System Access Policies,» 14 February 2019. [En línea]. Available: [https://docs.microsoft.com/en-us/openspecs/windows\\_protocols/ms-gpsb/d9bcb85c-67be-49cc-90ea-d2bd50873417](https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-gpsb/d9bcb85c-67be-49cc-90ea-d2bd50873417). [Último acceso: 10 July 2022].
- [14] HashiCorp, «Unattended Installation for Windows,» [En línea]. Available: [https://www.packer.io/guides/automatic-operating-system-installs/autounattend\\_windows](https://www.packer.io/guides/automatic-operating-system-installs/autounattend_windows). [Último acceso: 30 April 2022].
- [15] HashiCorp, «Packer Documentation,» [En línea]. Available: <https://www.packer.io/docs>. [Último acceso: 9 July 2022].
- [16] RedHat, community, «Ansible Documentation (full),» [En línea]. Available: <https://docs.ansible.com/ansible/latest/index.html>. [Último acceso: 8 July 2022].
- [17] «Group Policy Administrative Templates Catalog (Microsoft),» [En línea]. Available: <https://admx.help/>. [Último acceso: 10 July 2022].
- [18] J. Terra, «Ansible vs Chef: What's the Difference?,» 2022 June 13. [En línea]. Available: <https://www.simplilearn.com/ansible-vs-chef-differences-article>. [Último acceso: 1 July 2022].
- [19] Oracle Corporation, «Chapter 8. VBoxManage (from Oracle VM VirtualBox User Manual),» 2022. [En línea]. Available: <https://www.virtualbox.org/manual/ch08.html>. [Último acceso: 5 July 2022].
- [20] Oracle Corporation, «Oracle VM VirtualBox User Manual,» 2022. [En línea]. Available: <https://www.virtualbox.org/manual/>. [Último acceso: 9 July 2022].
- [21] CISOfy, «Lynis,» [En línea]. Available: <https://cisofy.com/lynis/>. [Último acceso: 10 June 2022].
- [22] R. Velasco, «Diferentes ediciones de Windows 10: Home vs Pro vs Enterprise vs Education,» 22 November 2021. [En línea]. Available: <https://www.softzone.es/windows/como-se-hace/windows-10-home-vs-pro-vs-enterprise-vs-education/>. [Último acceso: 1 July 2022].
- [23] K. Sudhakar, «How to Create a Vagrant Box from an Existing Box,» 2 June 2021. [En línea]. Available: <https://www.linuxshelltips.com/create-vagrant-box-using-existing-box/>. [Último acceso: 30 June 2022].
- [24] J. Redondo, Introducción a Ansible (PPT).
- [25] J. Redondo, Topic 4: Security Policies and Automated Hardening (PPT).
- [26] Microsoft Corporation, «Windows Firewall Profiles,» 31 May 2018. [En línea]. Available: <https://docs.microsoft.com/en-us/previous-versions/windows/desktop/ics/windows-firewall-profiles>. [Último acceso: 7 July 2022].
- [27] Centro Criptológico Nacional, «Esquema Nacional de Seguridad - Preguntas Frecuentes,» 2022. [En línea]. Available: <https://www.ccn-cert.cni.es/publico/dmpublidocuments/ENS-FAQ.pdf>. [Último acceso: 2 July 2022].



- 
- [28] Center for Internet Security, «CIS Microsoft Windows 10 Enterprise Benchmark v1.12.0,» 2022.
- [29] Center for Internet Security, «CIS Ubuntu Linux 18.04 LTS Benchmark v2.1.0,» 2021.
- [30] T. Ylonen y C. Lonvick, «The Secure Shell (SSH) Protocol Architecture,» January 2006. [En línea]. Available: <https://datatracker.ietf.org/doc/html/rfc4251>. [Último acceso: 2 July 2022].
- [31] Centro Criptológico Nacional, «Sector Público Certificado (ENS),» [En línea]. Available: <https://ens.ccn.cni.es/es/certificacion/sector-publico>. [Último acceso: 5 July 2022].
- [32] E. Krout, «Ansible vs Puppet: Which is right for you?,» 7 May 2021. [En línea]. Available: <https://acloudguru.com/blog/engineering/ansible-vs-puppet-which-is-right-for-you>. [Último acceso: 2 July 2022].



## DELIVERED CONTENTS

### Contents

The following table shows the structure of the contents that have been uploaded, so it is easier to locate each of the different parts.

The present document will be uploaded on its own, but there will be some additional files uploaded in separate ZIP files.

Directory	Contents
<i>vagrant-ansible-hardening-main</i>	This folder contains the structure of all the developed project
<i>documentation</i>	It contains all the documentation associated to the project, in more detail than what could be included in this document, as well as README.txt files indicating additional resource links

**Table 11. Structure of the contents uploaded**

### *Directory structure for the developed project*

In this section, the structure of the developed project will be detailed. It has already been shown in the form of a diagram (see *Package Diagrams*), but here, a more detailed explanation on the contents of each of the folders will be given. The project can also be found in the following GitHub repository: <https://github.com/Yori1999/vagrant-ansible-hardening.git>.

Directory	Contents
<i>./ Root directory</i>	Contains all the following folders and a README explaining the general purpose of the project
<i>./files</i>	Contains the CLARA files. This should not be removed, as Vagrant’s provisioners will try to locate this folder and copy its contents to the Windows machines
<i>./packer</i>	Contains all the files used for creating the Windows boxes using Packer, minus the packer.exe itself
<i>./provisioning</i>	This is one of the most important directories, as it contains all the Ansible code for provisioning the machines
<i>./vagrantfiles</i>	Contains all the different Vagrantfiles for creating each machine or set of machines, as well as some specific configurations for each of them

**Table 12. Project directory structure**