Universidad de Oviedo
Universidá d'Uviéu
University of Oviedo

Escuela de Ingeniería Informática
Universidad de Oviedo

Trabajo de Desarrollo

POLICY-DRIVEN WINDOWS SECURITY AUTOMATION

DEGREE IN COMPUTER SOFTWARE ENGINEERING

FINAL DEGREE PROJECT

AUTHOR

Diego Ramírez Amandi

PROJECT SUPERVISOR

José Manuel Redondo López

July 2020

This document has been created based on the template developed by JOSÉ MANUEL REDONDO LÓPEZ. [1]

Declaración Responsable

El alumno: Diego Ramírez Amandi

Con DNI:

Y UO:

DECLARA

Que esta obra es completamente original y se han citado debidamente las fuentes utilizadas durante la realización de esta.

Y para que conste, lo firma en Oviedo, a 09 de Julio de 2021

Firmado:

Acknowledgements

To my family, who has always supported me. You have helped me push my limits and become a better person.

To my friends, who have always encouraged me to overcome any difficulties. Thanks for all the laughs and good moments.

And finally, to all the lecturers and professionals that have taught me so much over these four years, especially Jose Manuel Redondo, supervisor of this project.

Contents index

Figure Index

Table Index

Chapter 1. ¿WHAT IS THIS PROJECT?

1.1 RESUMEN

Este trabajo describe la aplicación que se ha creado para facilitar la implantación de medidas de seguridad emitidas por el Centro para la Seguridad de Internet (CIS), una asociación sin ánimo de lucro integrada por numerosas empresas y organismos estatales. El sistema desarrollado se centra en aseguración de servidores Windows Server, ofreciendo tanto una interfaz gráfica como por consola para facilitar su utilización en dispositivos con o sin entorno gráfico.

Gracias a esta aplicación, el usuario puede consultar información sobre los distintos controles de seguridad disponibles, indicar si lo desea los servidores a analizar y en base a esta información elegir que partes de la seguridad desea auditar, comprobando el estado actual y actualizando aquellas políticas de seguridad que no se ajustan a la norma propuesta por el CIS.

El sistema ha sido desarrollado a petición del director del trabajo, José Manuel Redondo López, el cual desea cubrir con esta aplicación una sección de aseguración automática en Windows enmarcada dentro de la asignatura de Seguridad de Sistemas Informáticos del Grado en Ingeniería Informática del Software.

1.2 PALABRAS CLAVE

Palabras clave: CIS, Banco de pruebas, Ansible, Políticas de seguridad, Servidores Windows, Automatización de la seguridad

1.3 ABSTRACT

This work describes an application created to ease the implementation of security policies suggested by the Center of Internet Security (CIS), a non-profit institution integrated by numerous enterprises and government agencies. The developed system focuses attention on the hardening of Windows Server systems, offering both a command-line and graphical user interface, allowing its use in systems with a graphical environment, but also in those without it.

Thanks to this application, the user can obtain information of the different security controls (also known as checks), specify the hosts to analyze, and based on the knowledge gathered decide which

aspects of the system security should be audited, checking the actual state and updating those policies that do not fulfill the specification emitted by CIS.

The system has been developed on request of the director of the project, José Manuel Redondo López, who wants to cover a topic regarding automatic Windows hardening in the subject Information Systems Security in the Software Engineering university degree.

1.4 KEYWORDS

Keywords: CIS, Benchmark, Ansible, Security policies, Windows Server, Security automation

Chapter 2. Information System Plan

PLANNING PHASE

ISP

2.1 ISP 1: INFORMATION SYSTEM PLAN START

2.1.1 ISP 1.1: ISP Need Analysis

When people try to make a system more secure, they often resort to blogs or web pages where the validity and reliability of information is not totally verified. Hopefully, there exists some agencies that provide you with the knowledge needed. One of these associations is CIS. They publish a document, called benchmark, that includes several security policies that are applicable to a determinate operating system. However, the automation of these policies comes at a very high economic cost, and the market does not offer many free-to-use alternatives, especially regarding Windows [2].

To ease the understanding about the topic, the document will refer to these security controls as "checks".

Due to this necessity, Professor José Manuel Redondo López from the Software Engineering School proposed me to develop an application to automate the application of this benchmark. The idea behind it is that people that want to harden a Windows Server could benefit from it and, with a click, obtain a more secure server within minutes.

In addition to this, this application could serve to teach a section about Windows hardening in the subject Information Systems Security of the Software Engineering degree, which is mainly focused on Linux systems.

2.1.2 ISP 1.2: ISP Scope Identification

The objectives of the system to develop should fulfil a set of requirements:

- CIS Benchmarks contain a large amount of information, including a description, rationale, how to audit, remediation, etc. for each security control included in the benchmark. Allowing the user to read this could ease the decision about what policies need to be applied for a specific set of servers, depending on the use they will have. The system that will be developed should offer this feature, including the most remarkable sections of each check.
- The benchmark includes a large number of checks to implement. However, it does not consider the final use of the system. For servers that contain sensible data, it might be interesting to apply as many security policies as possible, while if the server is used for non-crucial tasks, implementing some of them would be meaningless. For this reason, the application to be develop must allow the user to enable or disable each check separately to elaborate a custom configuration that fits the needs of the system.
- The usefulness of an automation system lies in the avoidance of repeating tasks that are identical. If a user wants to automate the configuration of various servers, it would be meaningless to configure and execute each one individually. The software that will be

developed must consider the addition of multiple hosts, so that a configuration can be applied to all of them in one run. This will significantly speed the process of configuring a large number of hosts.

- Servers in administration environments are often presented without a graphical user interface, allowing just inputs in a command line. For this reason, a tool that aims to automate a part of server configuration should provide a command line interface (CLI) that allows the user to interact with it. The application should include a CLI with menus that allow the user to navigate, enable/disable checks, add hosts, etc. However, as it is intended to be a teaching tool too, providing a GUI with responsible elements could ease the effort of students to understand key concepts of Windows Server security, as it is an Operating System that is not analyzed as deep as Linux. For this reason, the implementation of a Graphical User Interface is also required.
- The administration of Windows Server policies can be considerably challenging at low level for an unexperienced user. Both checking the state of a determinate register and changing its value require expert knowledge that a novice user might not have. In addition, the correct use of some configuration files, especially those that specify hosts require specific format that must be considered. The software developed must simplify all of these operations offering a simple interface where a user without expert knowledge can apply a configuration by just specifying the IP and credentials of the host.
- Windows Server, as every other operating system, is updated periodically receiving patches that often change and enhance security policies. Due to this, CIS updates its benchmark every year to fit the last changes. For this reason, it is necessary that the application developed is flexible and allows new checks to be introduced.
- CIS Benchmarks consists of a large number of checks, which are divided into sections. Each check has a unique number attached that identifies it in the document. If a user wants to customize the configuration, having a filter to search for specific configuration becomes a necessity. The software must allow filtering between sections and determined checks to improve its usability.

To sum up, the following strategic objectives to ensure the success of the project must be achieved:

- Show key information regarding the checks included.
- Allow the user to enable/disable each check.
- Allow configuration of multiple hosts.
- Implementation of a CLI and GUI.
- Automation of low-level operations, providing a simple interface.
- Open to updates. Flexible.
- Provide filters for checks.

2.1.3 ISP 1.3: Determination of Responsible Parties

The project director will be responsible for the validation of the objectives proposed along the development of the project.

The student will be in charge of creating and developing the software modules described, without the cooperation of other students.

2.2 ISP 2: DEFINITION AND ORGANIZATION OF THE ISP

2.2.1 ISP 2.1: Scope Specification

Taking into consideration the strategic objectives presented in the previous section, the project will be divided into the phases, containing each of them the following objectives:

Phase 1: Display Check information

CIS publishes freely a PDF document containing every check for a determined operating system. It also provides this document in other formats, such as JSON, XML, XCCDF, etc. However, these formats are only available at a high economic cost and only available freely for US students. For this reason, this phase will consist of transforming the PDF into a computer-readable format.

Each check includes the following parts [3]:

- Code: Identifier for the check inside the benchmark.
- Name: Title given to the check.
- Profile applicability: Configuration categories classified by CIS.
- Description: Basic information regarding the check.
- Rationale: Justification for its application.
- Audit: Explanation of what to do to check its status.
- Remediation: Explanation on the actions to take to solve any misconfiguration and fulfil the requirements of the check.
- Impact: Changes that applying the suggested configuration will generate.
- Default Value: Value given by default by the OS.
- References: Identifier for the check in CIS' database
- CIS Controls: Related checks.

From these, user will be able to check the following:

- Code
- Name
- Profile applicability
- Description
- Rationale
- Impact
- Audit
- Remediation
- Default Value

Objectives of this phase:

- Convert PDF file into a parsable format, in this case, JSON.
- Parse the sections mentioned into the application.

Phase 2: Automation of check application

A system will be developed to manipulate the registers needed in the hosts machines, the ones that will be analyzed, applying the statement "Check, and change only if needed". This means, the system will be able to detect bad configurations and act only on these, skipping others that are correctly set. This behavior will avoid overwriting unnecessary information, leading to better performance and less execution times.

The user will be able to apply the desired set of checks by just clicking a button, and the system will generate for him a recap of the execution, indicating any errors, checks applied, good and bad configurations and actions taken.

In addition to this, simple connection to host machines must be also implemented in order to check the validity of the results.

As the CIS Benchmark for Windows Server 2019 is considerably extensive, only the first section will be implemented, letting other sections for future versions.

Objectives of this phase:

- Automatize the application of checks.
- Ensure only bad configurations are overwritten.
- Manage basic connection between the administrator system (where the program will be executed) and a host (server where the configuration will be applied)

Phase 3: Hosts manipulation

The user should be able to specify one or more hosts for the application to deploy the checking process into them.

It is necessary for the hosts (nodes) to be able to communicate with the administrator PC. That is, host systems and the administrator machine must be connected to the same network.

A host must be defined using its IP and valid credentials of a privileged user to grant access.

Objectives of this phase:

- Allow user to add one or more hosts.
- Ensure a configuration can be applied to multiple hosts.

Phase 4: CLI Implementation

Systems without a graphical environment should be able to use the application. For this purpose, a fully interactive command-line interface must be implemented.

When the app is executed, the logo of the app and the main menu must show up, displaying the different features offered by the software. With the help of a keyboard, the user must be able to navigate the menus and use every functionality as if it was a graphical application.

In addition to this, status or log messages must be displayed to enhance the usability and feedback form the app.

Every feature commented before should be available through this interface.

Objectives of this phase:

- Develop a fully interactive command line interface.
- Implement log messages to improve usability.
- Display menus to ease the navigation using only keyboard inputs.
- Ensure correct error handling.

Phase 5: GUI Implementation

Users with a graphical environment should benefit from the possibility of executing the command-line interface mentioned before, but also have access to a Graphical User Interface that contains all the features already provided by the CLI.

The interface will be presented in a resizable window, allowing the user to adjust to the preferred screen resolution.

The GUI must include tooltips, dialogs and responsive elements that inform the user of the actions happening in the background.

Objectives of this phase:

- Develop a fully interactive graphical user interface.
- Include tooltips and dialogs to improve usability.

Chapter 3. ISP 7:
Definition of the Technological Architecture

Planning Phase

ISP

3.1 ISP 7.2: SELECTION OF THE TECHNOLOGICAL ARCHITECTURE

This will be the infrastructure selected to create the project:

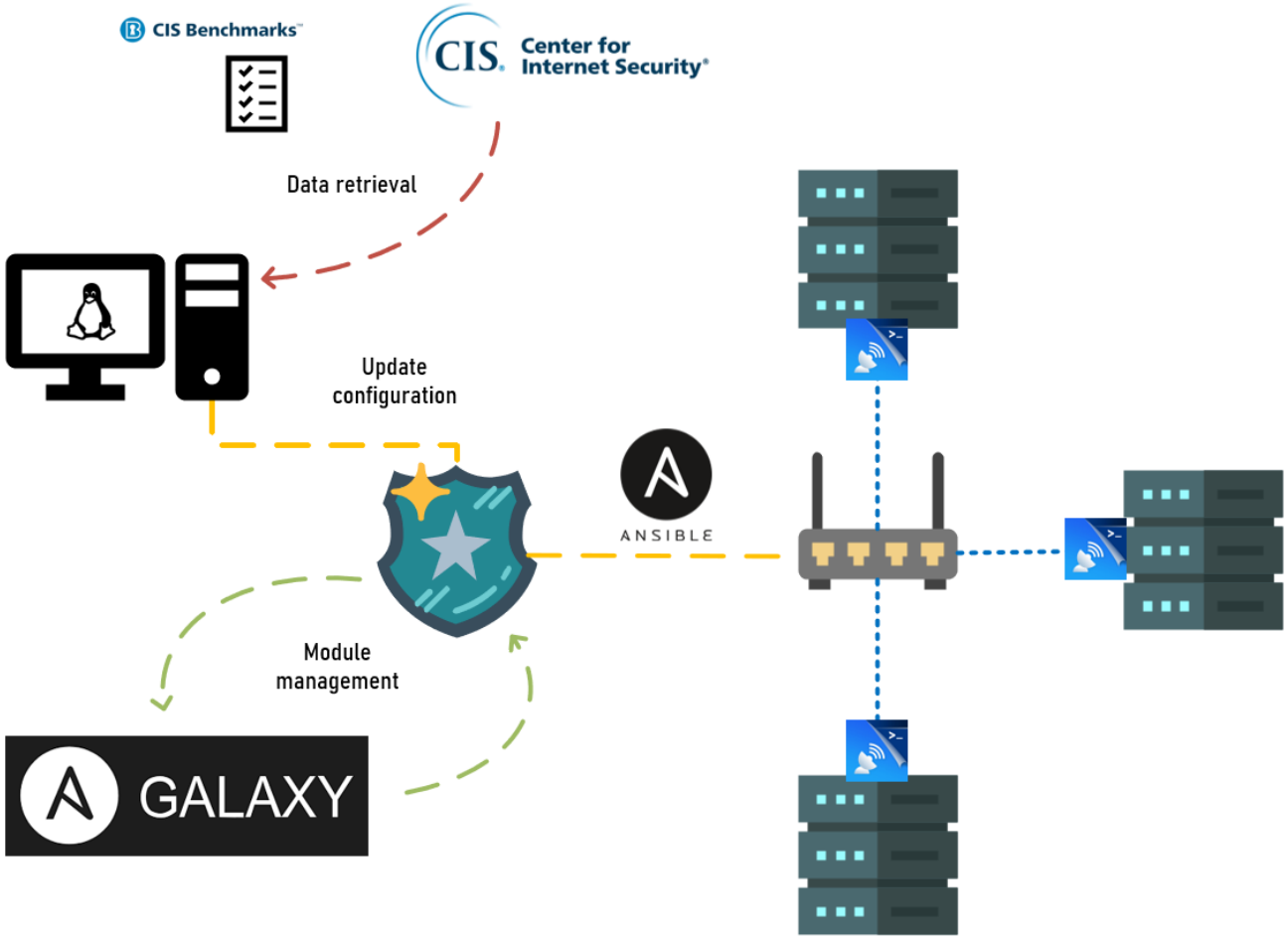


Figure 1: Technological Architecture

As it can be seen in the diagram , the objective is to create a system capable of running inside a Linux Machine that is able to apply a security configuration to a set of Windows Server Machines.

To achieve this, it must be able to process the data stored in the specific CIS Benchmark created for Windows Server distributions, offer the user the ability to create a configuration from it and then apply it with the help of Ansible. The application will connect to the hosts via WinRM protocol.

In addition to this, it must be able to gather the necessary modules from an internet repository: Ansible Galaxy. Therefore, the Linux machine requires of Internet access.

Chapter 4. System Feasibility Study

DEVELOPMENT PHASE

SFS

4.1 SFS 4, 5 y 6: Study and Assessment of Alternative Solutions and Selection of the Final Alternative

4.1.1 XCCDF

4.1.1.1 Description

This alternative takes advantage of two different technologies: XCCDF and OpenSCAP.

On the one hand, the eXtensible Configuration Checklist Description Format (XCCDF) is part of the Security Content Automation Protocol (SCAP) standard maintained by the National Institute of Standards and Technology (NIST). It is an XML based language designed to express, organize and manage security policies. [3]

XCCDF can be used to write security checklists and benchmarks. The specification supports information interchange, document generation, organizational and situational tailoring, automated compliance testing, and compliance scoring. In addition, it also defines a data model and format for storing results of benchmark testing. There are various versions available for free in the NIST page.

On the other hand, OpenSCAP is an open-source project composed by a set of tools designed to implement the SCAP standard. The OpenSCAP project provides numerous hardening guides developed by the open-source community designed to fit any need of a company.

The main tool of the OpenSCAP project is OpenSCAP Base [3]. It is capable of scanning host machines interpreting XCCDF Benchmarks.

The XCCDF specification can be downloaded for free from the NIST webpage. The last version published to this date is v1.2.

OpenSCAP Base can be also downloaded from OpenSCAP webpage. More documentation about the tool can be found here.

4.1.1.2 Advantages

- XCCDF is a well stablished standard for security checklist creation.
- XCCDF documents are expressed in XML. Therefore, it may be validated with an XML Schema-validating parser.
- Free usage license.

4.1.1.3 Disadvantages

There are some disadvantages here that must be considered:

- Although OpenSCAP has won some NIST certifications for its work, it is still an open-source project so problems might arise more frequently. For example, its Windows compatibility, introduced in version 1.3, is not completely developed, and often prompts error messages when running scans. In the following image, an error occurred while the alternative was being tested is shown:

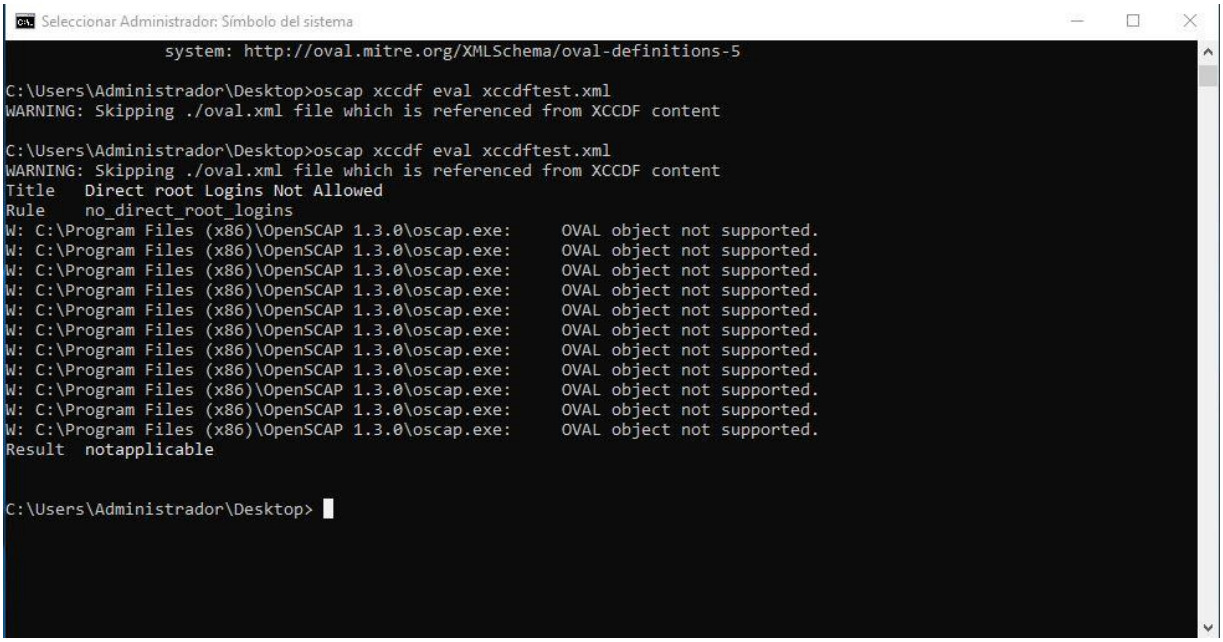


Figure 2: Errors in OpenSCAP Base execution for Windows

- In addition to this, XCCDF is designed mainly just to check registers and security policies and generate reports and scorings. However, it is not capable of carrying the fixing process. This should be a key aspect of the final product, so other alternatives must be used to cover this flaw.
- The lack of documentation on how to build efficient Benchmarks with XCCDF makes it more difficult to develop one without previous knowledge.

4.1.2 Ansible

4.1.2.1 Description

Ansible is an administration tool that allows the user to configure and verify systems. It is similar to other administration software like Chef, Puppet or Salt. However, Ansible is nowadays the most popular tool for these purposes.

This software automatizes the process of obtaining the server information before applying the desired tasks. It is property of RedHat, although it can be downloaded for free. However, it is only available for the moment in Linux systems. [5]

Ansible execution is idempotent, that is, tasks will always obtain the same result, no matter how many times they are run. This is achieved following a simple rule: "If a configuration is correct, don't do anything".

There are some key concepts that must be understood to comprehend Ansible functioning [4]:

- Inventory

File containing the connection parameters of the host where ansible will run the tasks.

- Task

Units of work that perform a determined action.

- Modules

Scripts that perform a set of actions. There are many different modules, available here, that do actions from installing software, updating registers, copying files, etc.

- Role

Set of vars, files, tasks, handlers, and other Ansible artifacts based on a known file structure.

- Playbook

Group of tasks that will be executed sequentially over one or more hosts.

Ansible supports different connection protocols such as SSH or WinRM, which makes it possible to work with both Linux and Windows systems.

The configuration files are written using YAML. This makes ansible files to be easily readable and closes the gap between human and machine. Tasks can be defined in a few lines of code using numerous modules implemented by the community.

The documentation for ansible can be read for the Ansible docs webpage.

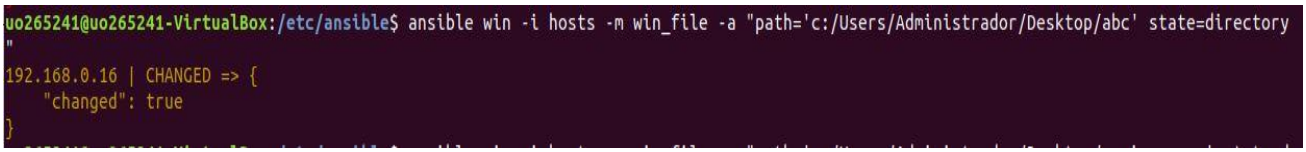


Figure 3: Example execution of Ansible

4.1.2.2 Advantages

There are numerous advantages for using Ansible:

- Idempotent tasks. Nothing is written twice; multiple executions do not affect the configuration.
- Well documented wiki. Every aspect of the tool is covered in its wiki. A large community attends any request.
- Checked and verified modules. You have the assurance that external code work as expected. Bugs are minimized.
- Configuration is not only revised, but also modified if wrong. This fits perfectly the needs of the application.
- Roles can be uploaded and shared. Ansible offers Ansible Galaxy, an open-source repository where given configurations can be shared.

4.1.2.3 Disadvantages

- It is only available in Linux systems.
- The RedHat support requires a paid license.
- It is a bit weaker in terms of system evaluation and scoring compared to XCCDF.

4.1.3 Conclusion

Due to the limitations of XCCDF and OpenSCAP alternative, and the bad performance offered in Windows Server, Ansible will be the option selected. Its versatility and ease to use, in addition to the ability to update configurations remotely will ease the development of the application and create a better final product.

Chapter 5. Planning and Management of the TFG

DEVELOPMENT PHASE

5.1 PROJECT PLANNING

5.1.1 Stakeholders Identification

There are two main stakeholders for this project:

- Developer
- Project director

5.1.2 OBS y PBS

Due to the organization and special characteristics of this project, the design of an OBS and PBS is not applicable.

5.1.3 Initial Planning. WBS

Code	Task	Duration	Start	Finish	Predecessors
1	**Project: Windows Server Security Automation**	**298 hours**	**mon 01/02/21**	**tue 29/06/21**	
1.1	**Initial planning**	**2 hours**	**mon 01/02/21**	**mon 01/02/21**	
1.1.1	Project presentation	1 hour	mon 01/02/21	mon 01/02/21	
1.1.2	Test laboratory creation	1 hour	mon 01/02/21	mon 01/02/21	3
1.2	**Analysis**	**54 hours**	**mon 01/02/21**	**thu 25/02/21**	
1.2.1	Initial analysis	4 hours	mon 01/02/21	tue 02/02/21	2
1.2.2	**Research**	**20 hours**	**wed 03/02/21**	**thu 11/02/21**	
1.2.2.1	Security Policies	10 hours	wed 03/02/21	mon 08/02/21	6
1.2.2.2	CIS Benchmarks	10 hours	mon 08/02/21	thu 11/02/21	8
1.2.3	**Alternative's study**	**30 hours**	**thu 11/02/21**	**thu 25/02/21**	
1.2.3.1	XCCDF Checklists	20 hours	thu 11/02/21	mon 22/02/21	7
1.2.3.2	Ansible automation	10 hours	mon 22/02/21	thu 25/02/21	11
1.3	**Development**	**178 hours**	**thu 25/02/21**	**fri 28/05/21**	

1.3.1	**Module: Ansible Manager**	**72 hours**	**thu 25/02/21**	**wed 31/03/21**	
1.3.1.1	Analysis	20 hours	thu 25/02/21	mon 08/03/21	5
1.3.1.2	Design	12 hours	mon 08/03/21	fri 12/03/21	15
1.3.1.3	Development	36 hours	fri 12/03/21	tue 30/03/21	16
1.3.1.4	Testing	4 hours	tue 30/03/21	wed 31/03/21	17
1.3.2	**Module: CLI**	**36 hours**	**wed 31/03/21**	**fri 16/04/21**	
1.3.2.1	Analysis	4 hours	wed 31/03/21	thu 01/04/21	14
1.3.2.2	Design	8 hours	fri 02/04/21	tue 06/04/21	20
1.3.2.3	Development	20 hours	tue 06/04/21	thu 15/04/21	21
1.3.2.4	Testing	4 hours	thu 15/04/21	fri 16/04/21	22
1.3.3	**Module: GUI**	**70 hours**	**fri 16/04/21**	**fri 28/05/21**	
1.3.3.1	Analysis	8 hours	fri 16/04/21	wed 21/04/21	19
1.3.3.2	Design	16 hours	wed 21/04/21	wed 28/04/21	25
1.3.3.3	Development	42 hours	wed 28/04/21	wed 26/05/21	26
1.3.3.4	Testing	4 hours	wed 26/05/21	fri 28/05/21	27
1.4	**Deployment**	**4 hours**	**fri 28/05/21**	**tue 01/06/21**	
1.4.1	**Executable creation**	**4 hours**	**fri 28/05/21**	**tue 01/06/21**	
1.4.1.1	CLI	2 hours	fri 28/05/21	mon 31/05/21	13
1.4.1.2	GUI	2 hours	mon 31/05/21	tue 01/06/21	31
1.5	**Training**	**4 hours**	**tue 01/06/21**	**wed 02/06/21**	
1.5.1	User manual creation	4 hours	tue 01/06/21	wed 02/06/21	29
1.6	**Organization**	**2 hours**	**mon 05/04/21**	**mon 05/04/21**	
1.6.1	Periodical meeting 1	2 hours	mon 05/04/21	mon 05/04/21	
1.7	**Documentation**	**56 hours**	**wed 02/06/21**	**tue 29/06/21**	
1.7.1	Structure analysis	4 hours	wed 02/06/21	thu 03/06/21	33
1.7.2	Write documentation	52 hours	fri 04/06/21	tue 29/06/21	38

Table 1: WBS

The WBS presented before can be represented by means of the Gantt Chart that follows:

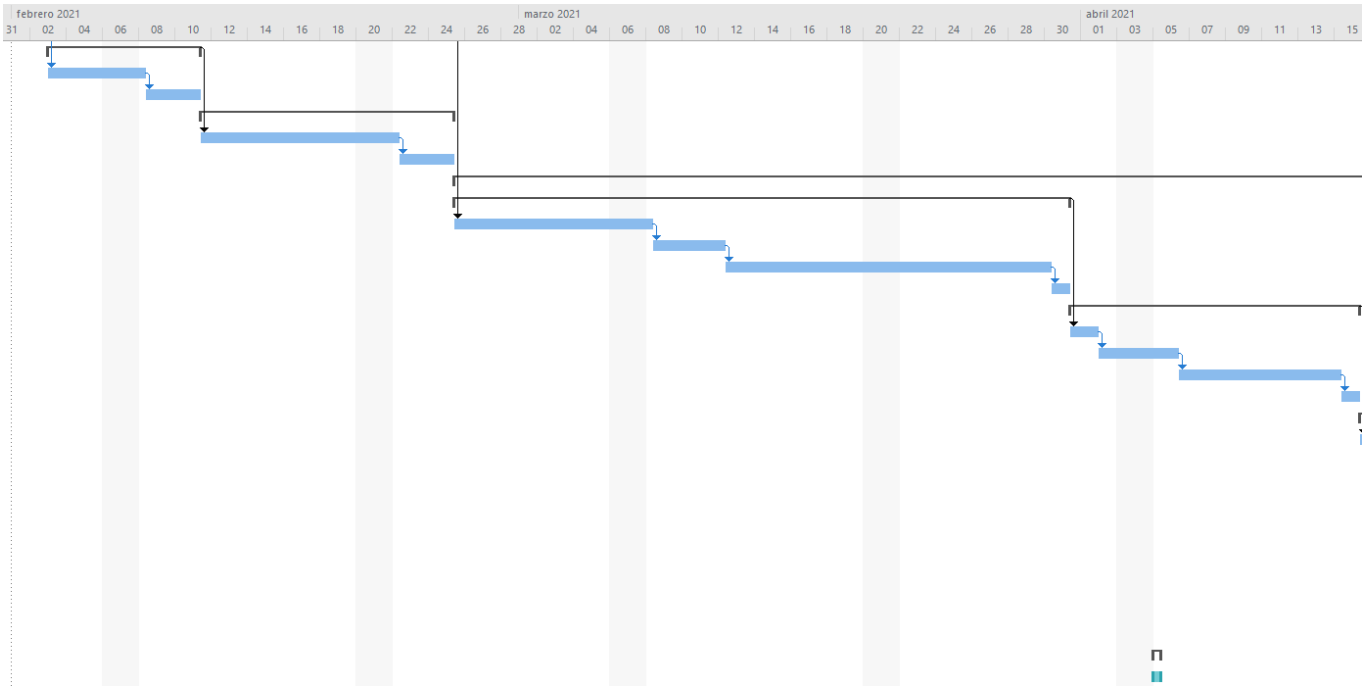


Figure 4: Gantt Chart (01/02/2021 – 15/04/2021)

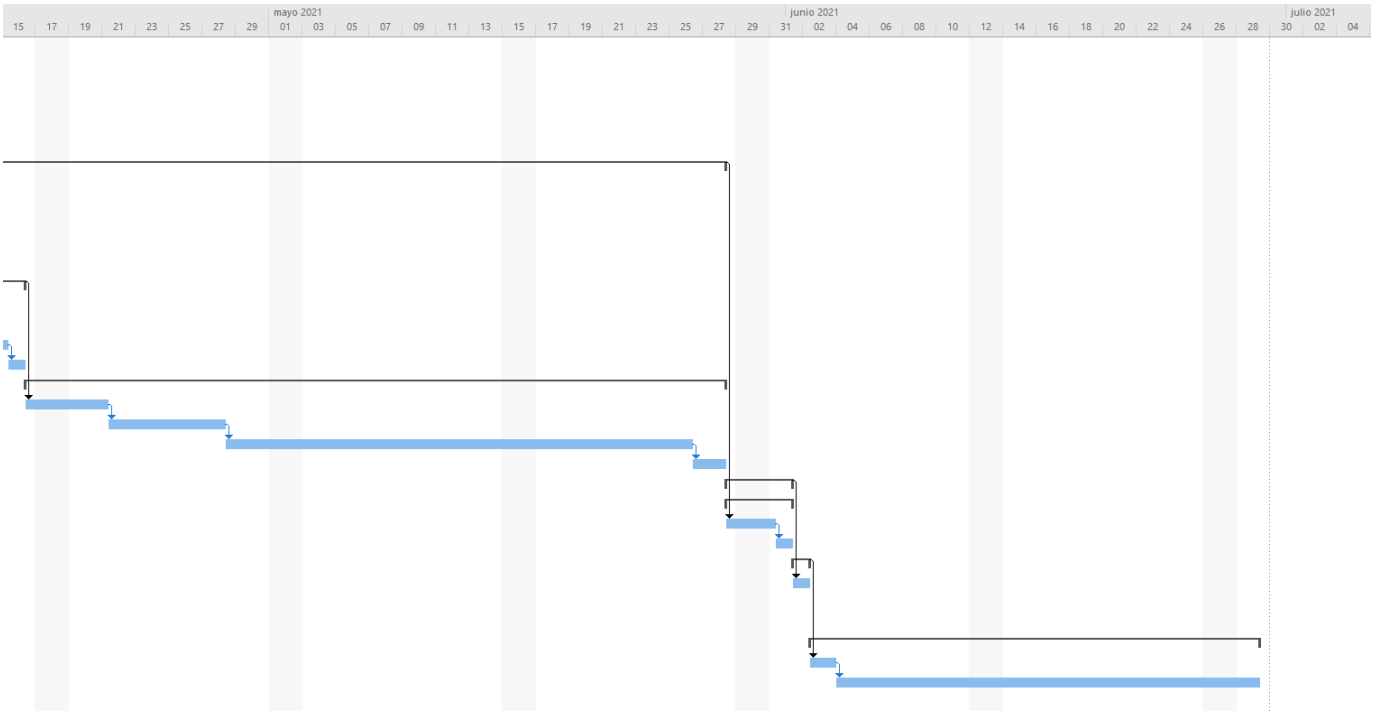


Figure 5: Gantt Chart (15/04/2021 – 29/06/2021)

5.1.4 Risks

5.1.4.1 Risks Management Plan

The Risk Management Plan is listed in the Anex I.

5.1.4.2 Risk identification

The following risks have been detected:

- Difficulties facing a new technology.
- Compatibility problems between ansible modules
- Tight schedules
- Inaccurate estimations
- Poor quality code
- Policy changes regarding CIS Benchmarks
- Incompatibilities between OSCAP and Windows
- Low flexibility
- Connection failures when applying a configuration.
- Unfamiliarity with the Windows Security field

5.1.4.3 Risk Register

ID	**Name**			
1	Difficulties facing a new technology			
Description				
The lack of experience using technologies, such as Ansible, YAML or WinRM can challenge the success of the project.				
Category				**Probability**
Technology				High
Budget	**Schedule**	**Scope**	**Quality**	**Impact**
Low	High	High	High	0,39

ID	**Name**			
2	Compatibility problems between ansible modules			
Description				
Due to the great number of modules supported by Ansible, most of them being developed by the community, incompatibilities might arise.				
Category				**Probability**
Technology				Low
Budget	**Schedule**	**Scope**	**Quality**	**Impact**
Nil	Medium	High	High	0,17

ID	Name			
3	Tight schedules			
Description				
The lack of time between study and work might affect the planification delaying certain phases of the project.				
Category				**Probability**
Planning				Medium
Budget	**Schedule**	**Scope**	**Quality**	**Impact**
Low	Critical	High	Medium	0,45

ID	Name			
4	Inaccurate estimations			
Description				
Mainly due to the investigation phase at the beginning of this project, it is possible that estimations might become scarce to research all the information needed.				
Category				**Probability**
Estimating				Medium
Budget	**Schedule**	**Scope**	**Quality**	**Impact**
Medium	Critical	Medium	Low	0,45

ID	Name			
5	Poor quality code			
Description				
A bad architectural design of the application might lead to "bad smells" in the code.				
Category				**Probability**
Quality				Low
Budget	**Schedule**	**Scope**	**Quality**	**Impact**
Nil	Nil	Nil	Critical	0,27

ID	Name			
6	Policy changes regarding CIS Benchmarks			
Description				
CIS might change their policies and limit the use of future versions of the benchmark.				
Category				**Probability**
Subcontractors and Suppliers				Very low
Budget	**Schedule**	**Scope**	**Quality**	**Impact**
Critical	Nil	Nil	Nil	0,09

ID	Name			
7	Incompatibilities between OSCAP and Windows			
Description				
Previous versions of OSCAP do not support Windows, and integration process is still in development.				
Category				**Probability**
Technology				Medium
Budget	**Schedule**	**Scope**	**Quality**	**Impact**
Low	Medium	High	Medium	0,28

ID	Name			
8	Low flexibility			
Description				
The unavailability of computer-readable Benchmark documents can lead to difficulties adding new sections of the benchmark.				
Category				**Probability**
Quality				Low
Budget	**Schedule**	**Scope**	**Quality**	**Impact**
Nil	Nil	Medium	Critical	0,27

ID	Name			
9	Connection failures when applying a configuration			
Description				
A connection failure between the administrator and host machine can make security configurations incomplete giving a false sense of security to the user.				
Category				**Probability**
Performance and Reliability				Low
Budget	**Schedule**	**Scope**	**Quality**	**Impact**
Nil	Low	Medium	Critical	0,27

ID	Name			
10	Unfamiliarity with the Windows Security field			
Description				
Developing a tool to increase the security of an OS without proper knowledge may increase the time taken by the analysis phase of the project.				
Category				**Probability**
Technology				High
Budget	**Schedule**	**Scope**	**Quality**	**Impact**
Low	High	Low	Medium	0,39

5.1.5 Initial Budget

5.1.5.1 Costs budget

For calculating the initial budget, only one developer will be considered. Attending to market, it is considered a price of 17€/hour:

L1	L2	Description	Quantity	Units	Price	Subtotal (2)	Total
1		Project: Windows Server Security Automation					5.100,00 €
	1.1	Initial planning	2	hours	17,00 €	34,00 €	
	1.2	Analysis	54	hours	17,00 €	918,00 €	
	1.3	Development	178	hours	17,00 €	3.026,00 €	
	1.4	Deployment	4	hours	17,00 €	68,00 €	
	1.5	Training	4	hours	17,00 €	68,00 €	
	1.6	Organization	2	hours	17,00 €	34,00 €	
	1.7	Documentation	56	hours	17,00 €	952,00 €	

Table 2: Initial cost budget

5.2 PROJECT EXECUTION

5.2.1 Plan tracking

Here are presented the corresponding baselines to the start, middle and close phases of the project:

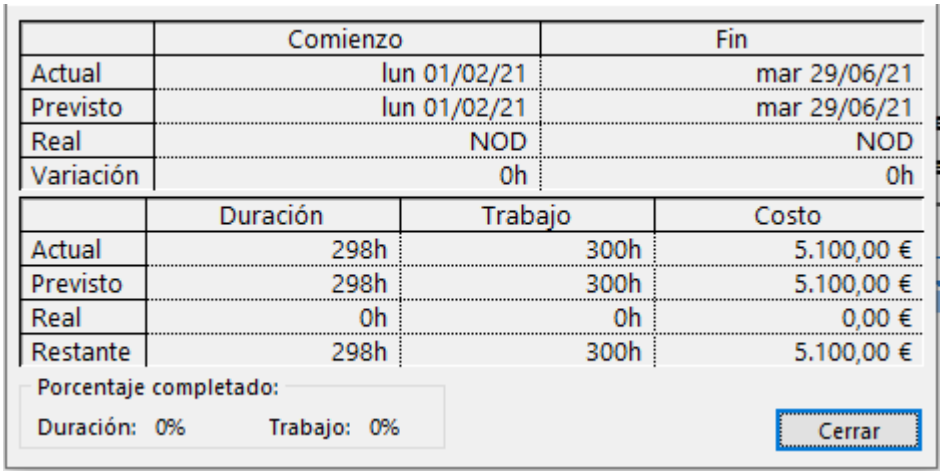


Figure 6: Initial project baseline

	Comienzo	Fin
Actual	lun 01/02/21	lun 12/07/21
Previsto	lun 01/02/21	mar 29/06/21
Real	lun 01/02/21	NOD
Variación	0h	28h

	Duración	Trabajo	Costo
Actual	326h	328h	5.576,00 €
Previsto	298h	300h	5.100,00 €
Real	155,05h	156h	2.652,00 €
Restante	170,95h	172h	2.924,00 €

Porcentaje completado:
Duración: 48% Trabajo: 48%

Cerrar

Figure 7: Mid-project baseline

	Comienzo	Fin
Actual	lun 01/02/21	lun 12/07/21
Previsto	lun 01/02/21	mar 29/06/21
Real	lun 01/02/21	lun 12/07/21
Variación	0h	28h

	Duración	Trabajo	Costo
Actual	326h	334h	5.678,00 €
Previsto	298h	300h	5.100,00 €
Real	326h	334h	5.678,00 €
Restante	0h	0h	0,00 €

Porcentaje completado:
Duración: 100% Trabajo: 100%

Cerrar

Figure 8: End project baseline

5.2.2 Project Incidence Logbook

DATE	ID	INCIDENCE	SOLUTION
01/02/2021	INC-1	Problems communicating both machines.	Laboratory setup completion is delayed (4h).
03/02/2021	INC-2	Great documentation found for Windows Security Policies	Duration of the security policies task is reduced (5h)
17/02/2021	INC-3	OSCAP for Windows. Execution problems.	XCCDF alternative research completion is delayed (30h).
25/02/2021	INC-4	Problems connecting via SSH with Ansible.	WinRM is proposed as a new alternative. More duration added to Ansible Manager-Analysis. (30h)

27/02/2021	INC-5	Ansible Manager analysis phase is increased due to lack of experience with the tool.	More duration added to Ansible Manager-Analysis. (40h)
01/05/2021	INC-6	Lack of time due to external activities leads to tasks in may being delayed.	Workdays are reduced from 4h to 2h during May.

5.2.3 Risks

ID	**Name**		
1	Difficulties facing a new technology		
Initial	**Budget**	**Schedule**	**Probability**
	Low	High	High
	Scope	**Quality**	**Impact**
	High	High	0,27
Occurred	YES		
New	**Budget**	**Schedule**	**Probability**
	Low	High	High
	Scope	**Quality**	**Impact**
	High	High	0,27
Comments			
Causes INC-4. Causes INC-5.			

ID	**Name**		
2			
Initial	**Budget**	**Schedule**	**Probability**
	Nil	Medium	Low
	Scope	**Quality**	**Impact**
	High	High	0,17
Occurred	NO		
New	**Budget**	**Schedule**	**Probability**
	Nil	Medium	Very low
	Scope	**Quality**	**Impact**
	High	High	0,06
Comments			
Once modules that will be used have been found, the risk of incompatibilities is drastically reduced.			

ID	Name		
3	Tight schedules		
Initial	**Budget**	**Schedule**	**Probability**
	Low	Critical	Medium
	Scope	**Quality**	**Impact**
	High	Medium	0,45
Occurred	YES		
New	**Budget**	**Schedule**	**Probability**
	Low	Critical	Medium
	Scope	**Quality**	**Impact**
	High	Medium	0,45
Comments			
Causes INC-6.			

ID	Name		
7	Incompatibilities between OSCAP and Windows		
Initial	**Budget**	**Schedule**	**Probability**
	Low	Medium	Medium
	Scope	**Quality**	**Impact**
	High	Medium	0,28
Occurred	YES		
New	**Budget**	**Schedule**	**Probability**
	Low	Medium	Medium
	Scope	**Quality**	**Impact**
	High	Medium	0,28
Comments			
Causes INC-3.			

ID	Name		
10	Unfamiliarity with the Windows Security field		
Initial	**Budget**	**Schedule**	**Probability**
	Low	High	High
	Scope	**Quality**	**Impact**
	Low	Medium	0,39
Occurred	YES		
New	**Budget**	**Schedule**	**Probability**
	Low	High	Low
	Scope	**Quality**	**Impact**
	Low	Medium	0,17
Comments			
Causes INC-2. Probability reduced.			

5.3 PROJECT TERMINATION

5.3.1 Final Planning

Code	Task	Duration	Start	Finish
1	**Project: Windows Server Security Automation**	**326 hours**	**mon 01/02/21**	**mon 12/07/21**
1.1	**Initial planning**	**5 hours**	**mon 01/02/21**	**tue 02/02/21**
1.1.1	Project presentation	1 hour	mon 01/02/21	mon 01/02/21
1.1.2	Test laboratory creation	4 hours	mon 01/02/21	tue 02/02/21
1.2	**Analysis**	**59 hours**	**tue 02/02/21**	**tue 02/03/21**
1.2.1	Initial analysis	4 hours	tue 02/02/21	wed 03/02/21
1.2.2	**Research**	**15 hours**	**thu 04/02/21**	**wed 10/02/21**
1.2.2.1	Security Policies	5 hours	thu 04/02/21	fri 05/02/21
1.2.2.2	CIS Benchmarks	10 hours	fri 05/02/21	wed 10/02/21
1.2.3	**Alternative's study**	**40 hours**	**thu 11/02/21**	**tue 02/03/21**
1.2.3.1	XCCDF Checklists	30 hours	thu 11/02/21	wed 24/02/21
1.2.3.2	Ansible automation	10 hours	thu 25/02/21	tue 02/03/21
1.3	**Development**	**198 hours**	**tue 02/03/21**	**fri 11/06/21**
1.3.1	**Module: Ansible Manager**	**92 hours**	**tue 02/03/21**	**tue 13/04/21**
1.3.1.1	**Analysis**	**40 hours**	**tue 02/03/21**	**fri 19/03/21**
1.3.1.1.1	Tool learning	10 hours	tue 02/03/21	fri 05/03/21
1.3.1.1.2	Console manipulation	4 hours	fri 05/03/21	mon 08/03/21
1.3.1.1.3	File manipulation	8 hours	tue 09/03/21	thu 11/03/21
1.3.1.1.4	Multiple task execution	8 hours	thu 11/03/21	tue 16/03/21
1.3.1.1.5	Role creation	8 hours	tue 16/03/21	thu 18/03/21
1.3.1.1.6	Repository sharing	2 hours	fri 19/03/21	fri 19/03/21
1.3.1.2	Design	12 hours	fri 19/03/21	thu 25/03/21
1.3.1.3	Development	36 hours	thu 25/03/21	mon 12/04/21
1.3.1.4	Testing	4 hours	mon 12/04/21	tue 13/04/21
1.3.2	**Module: CLI**	**36 hours**	**wed 14/04/21**	**thu 29/04/21**
1.3.2.1	Analysis	4 hours	wed 14/04/21	thu 15/04/21
1.3.2.2	Design	8 hours	thu 15/04/21	mon 19/04/21
1.3.2.3	Development	20 hours	tue 20/04/21	wed 28/04/21
1.3.2.4	Testing	4 hours	wed 28/04/21	thu 29/04/21
1.3.3	**Module: GUI**	**70 hours**	**fri 30/04/21**	**fri 11/06/21**
1.3.3.1	Analysis	8 hours	fri 30/04/21	wed 05/05/21
1.3.3.2	Design	16 hours	wed 05/05/21	mon 17/05/21
1.3.3.3	Development	42 hours	mon 17/05/21	wed 09/06/21
1.3.3.4	Testing	4 hours	thu 10/06/21	fri 11/06/21

1.4	**Deployment**	**4 hours**	**fri 11/06/21**	**mon 14/06/21**
1.4.1	**Executable creation**	**4 hours**	**fri 11/06/21**	**mon 14/06/21**
1.4.1.1	CLI	2 hours	fri 11/06/21	fri 11/06/21
1.4.1.2	GUI	2 hours	mon 14/06/21	mon 14/06/21
1.5	**Training**	**4 hours**	**mon 14/06/21**	**tue 15/06/21**
1.5.1	User manual creation	4 hours	mon 14/06/21	tue 15/06/21
1.6	**Organization**	**171 hours**	**mon 05/04/21**	**thu 01/07/21**
1.6.1	Periodical meeting 1	2 hours	mon 05/04/21	mon 05/04/21
1.6.2	Periodical meeting 2	3 hours	fri 11/06/21	fri 11/06/21
1.6.3	Periodical meeting 3	3 hours	thu 01/07/21	thu 01/07/21
1.7	**Documentation**	**56 hours**	**wed 16/06/21**	**mon 12/07/21**
1.7.1	Structure analysis	4 hours	wed 16/06/21	thu 17/06/21
1.7.2	Write documentation	52 hours	thu 17/06/21	mon 12/07/21

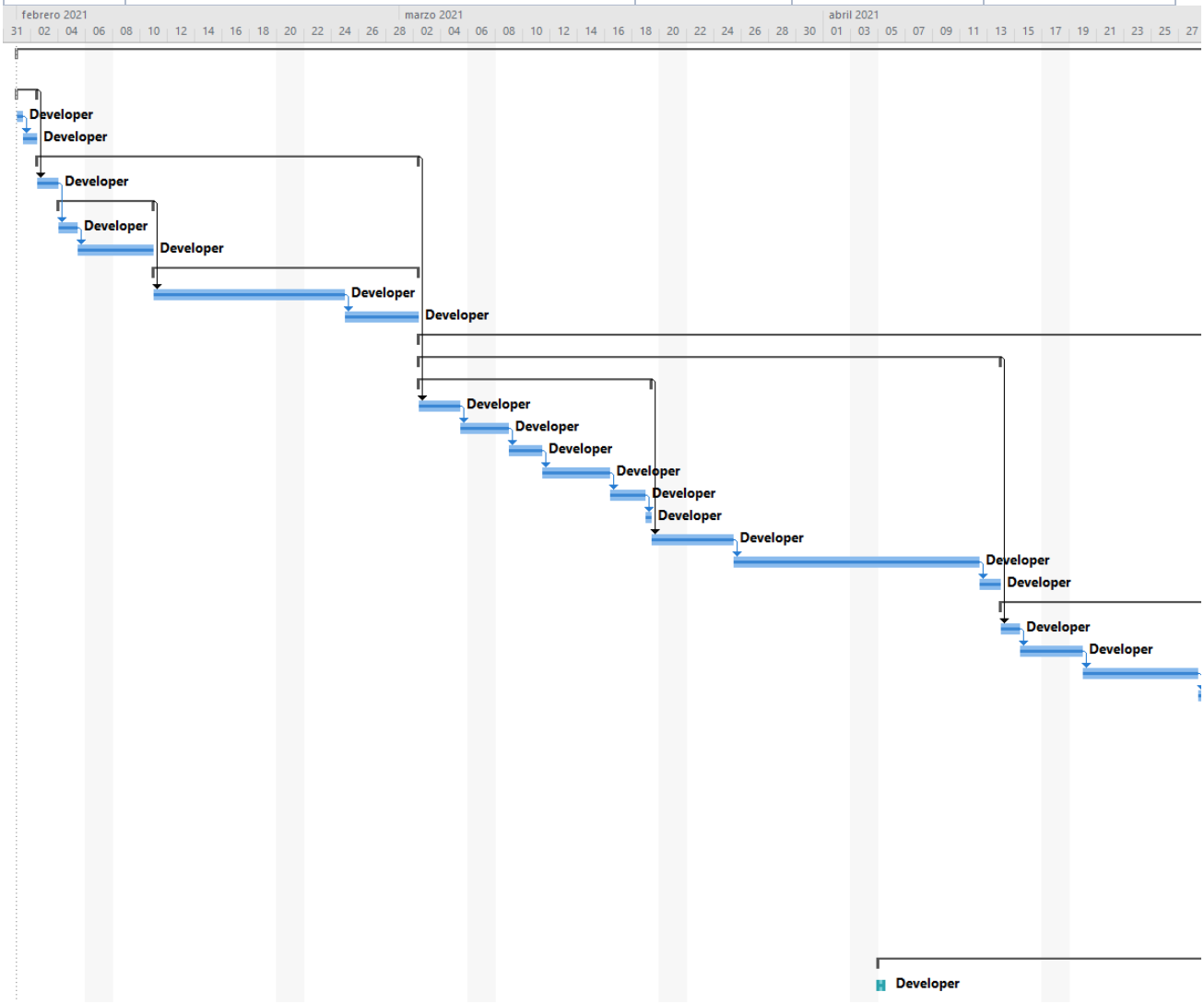


Figure 9: Final Plan. Gantt Diagram (01/02/2021-27/04/2021)

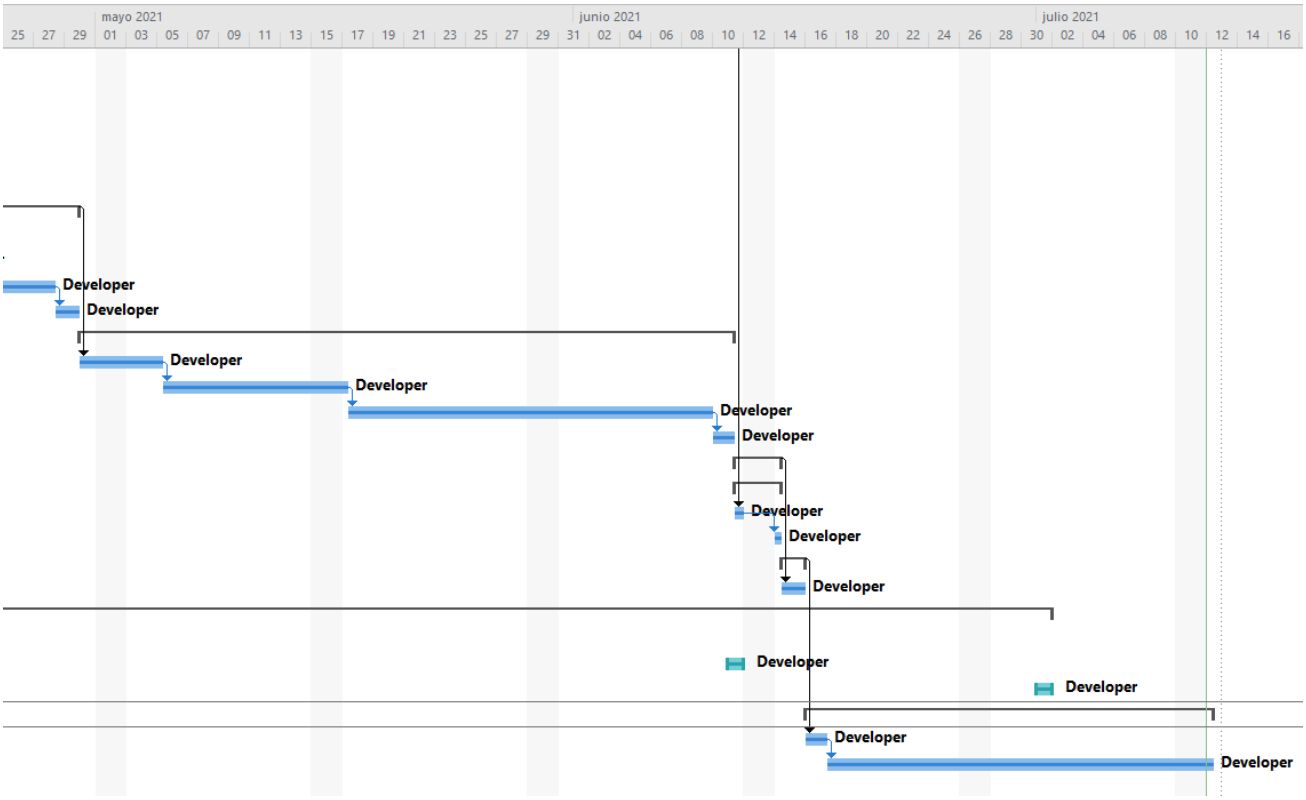


Figure 10: Final Plan. Gantt Diagram (27/04/2021-12/07/2021)

5.3.2 Final Risks Report

No more costs are identified or modified. Therefore, any query can be solved by means of sections 5.1.4 and 5.2.3.

5.3.3 Final Cost Budget

L1	L2	Description	Quantity	Units	Price	Subtotal (2)	Total
1		Project: Windows Server Security Automation					5.678,00 €
	1.1	Initial planning	5	hours	17,00 €	85,00 €	
	1.2	Analysis	59	hours	17,00 €	1.003,00 €	
	1.3	Development	198	hours	17,00 €	3.366,00 €	
	1.4	Deployment	4	hours	17,00 €	68,00 €	
	1.5	Training	4	hours	17,00 €	68,00 €	
	1.6	Organization	8	hours	17,00 €	136,00 €	
	1.7	Documentation	56	hours	17,00 €	952,00 €	

Chapter 6. Information System Analysis

DEVELOPMENT PHASE

ISA

6.1 ISA 1: SYSTEM DEFINITION

6.1.1 Scope Determination of the System

The following project will consist on the creation of an automatic system to apply multiple checks covered by the Windows Server 2019 CIS Benchmark to one or more machines running this operating system.

Due to the large number of checks given by CIS, the aim of the project will be to design an application capable to do it. For this reason, only a few, the first section of the document, which contains password policies will be implemented in the application. However, the system designed should admit the addition of new sections without the need of a redesign.

As explained, the following sections will only cover the design and development of the modules included in the application.

6.2 ISA 2: ESTABLISHMENT OF REQUIREMENTS

6.2.1 System Requirements Collection

Functional requirements

RFBenchmark.1 The system will allow the user to visualize, for each check, the available information in the CIS Benchmark. The system will show the following fields:

RFBenchmark.1.1 Code of the check.

RFBenchmark.1.2 Name of the check.

RFBenchmark.1.3 Profile applicability of the check.

RFBenchmark.1.4 Description of the check.

RFBenchmark.1.5 Rationale of the check.

RFBenchmark.1.6 Impact of the check.

RFBenchmark.1.7 Audit of the check.

RFBenchmark.1.8 Remediation of the check.

RFBenchmark.1.9 Default value of the check.

RFAnsible.1 The system will allow the user to download the necessary Ansible modules to perform the checks.

RFAnsible.1.1 The system will alert the user.

RFAnsible.2 The system will allow the user to connect to the host machine.

RFAnsible.2.1 The system will ask the user for access credentials.

RFAnsible.2.1.1 The user will ask for the name of the account in the host machine.

RFAnsible.2.1.2 The user will ask for the password of the previously specified account name in the host machine.

RFAnsible.2.2 The system will allow the connection to multiple machines.

RFAnsible.2.2.1 The credentials will be asked following the procedure of RFAnsible.2.1

RFAnsible.2.2.1.1 There must exist an account with those credentials to stablish a successful connection.

RFAnsible.2.2.1.2 The user must be able to abandon the creation of a custom host at any given time.

RFAnsible.3 The system will inform the user of the changes made in the hosts.

RFAnsible.3.1 The system will give information about any problems in the connection to the host.

RFAnsible.3.2 The system will assign a final status of each check.

RFAnsible.3.3 The system will not overwrite registers that were already correctly configured.

RFConsole.1 The system will include a colorized command line interface.

RFConsole.1.1 The system will offer interactive menus to navigate through the features of the application.

RFConsole.1.1 The system must inform the user of the option selected.

RFConsole.1.2 The system will allow the user to display the list of tasks.

RFConsole.1.2.1 The list will show the following information:

RFConsole.1.2.1.1 The code of the check.

RFConsole.1.2.1.2 The name of the check.

RFConsole.1.2.1.3 The status of the check. Each check can be:

RFConsole.1.2.1.3.1 Enabled.

RFConsole.1.2.1.3.2 Disabled.

RFConsole.1.3. The system will allow checks to be toggled.

RFConsole.1.3.1 The user will specify the code of the check to be toggled.

RFConsole.1.3.1.1 The system will accept regular expressions to perform multiple toggles of checks.

RFConsole.1.4. The system will allow the user to specify the code of a task to check available information. The information displayed here is gathered in RFBenchmark.1.

RFGraphical.1 The system will include a graphical user interface (GUI).

RFGraphical.1.1 The GUI will allow the user to resize the window to the desired resolution.

RFGraphical.1.2 The GUI will allow the user to resize the window to the desired resolution.

RFGraphical.1.3 The features included in the GUI must be the same as the ones included in the command-line interface. They are specified in RFConsole.1.

Non-functional requirements

RNF.1. The application must be run in a Linux system.

RNF.2. The system must have Ansible installed.

RNF.3. The system GUI must include tooltips for buttons.

RNF.4. The user must know how to use regular expressions to be able to select multiple checks

6.2.2 System Actors Identification

Primary actors

- Ansible Galaxy: Online repository where Ansible modules are stored. [8] When it receives the required command, it is able to download the specified modules to the user's machine.
- User: Main actor of the application. The application requires the user to input orders to serve the features included.
- Ansible playbook: Ansible module that applies a given configuration to one or more host machines.

Secondary actors

- Host: It represents the machine or machines in were ansible playbook will run the necessary tasks.

6.2.3 Use Case Specification

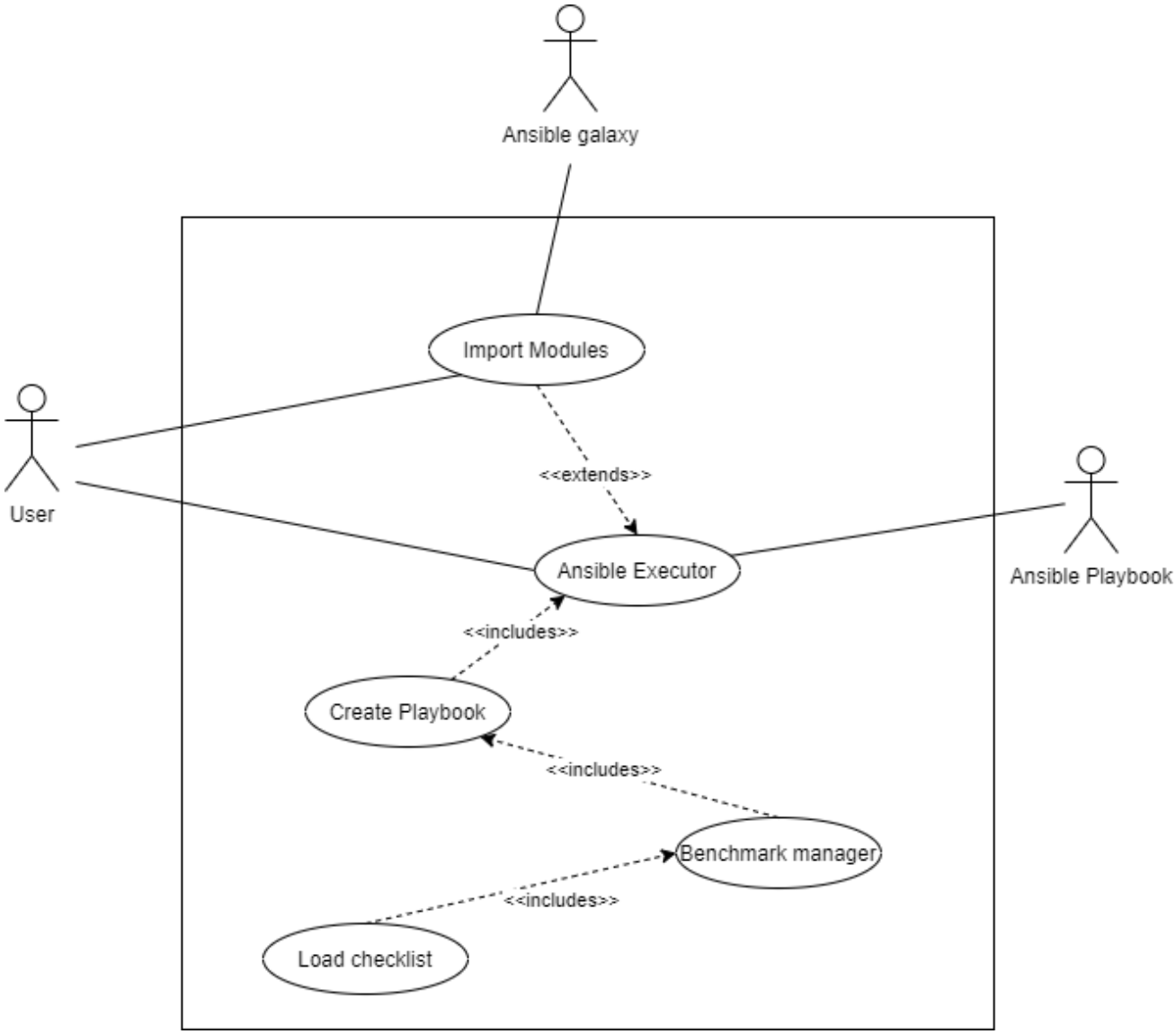


Figure 11: Use case 1. Default Ansible execution

Use case name
Default Ansible execution
Description
The user aims to apply a default configuration to the default host specified in Ansible's inventory file. For it to work, user must import the necessary modules from Ansible galaxy. In this execution, all CIS checks will be executed.

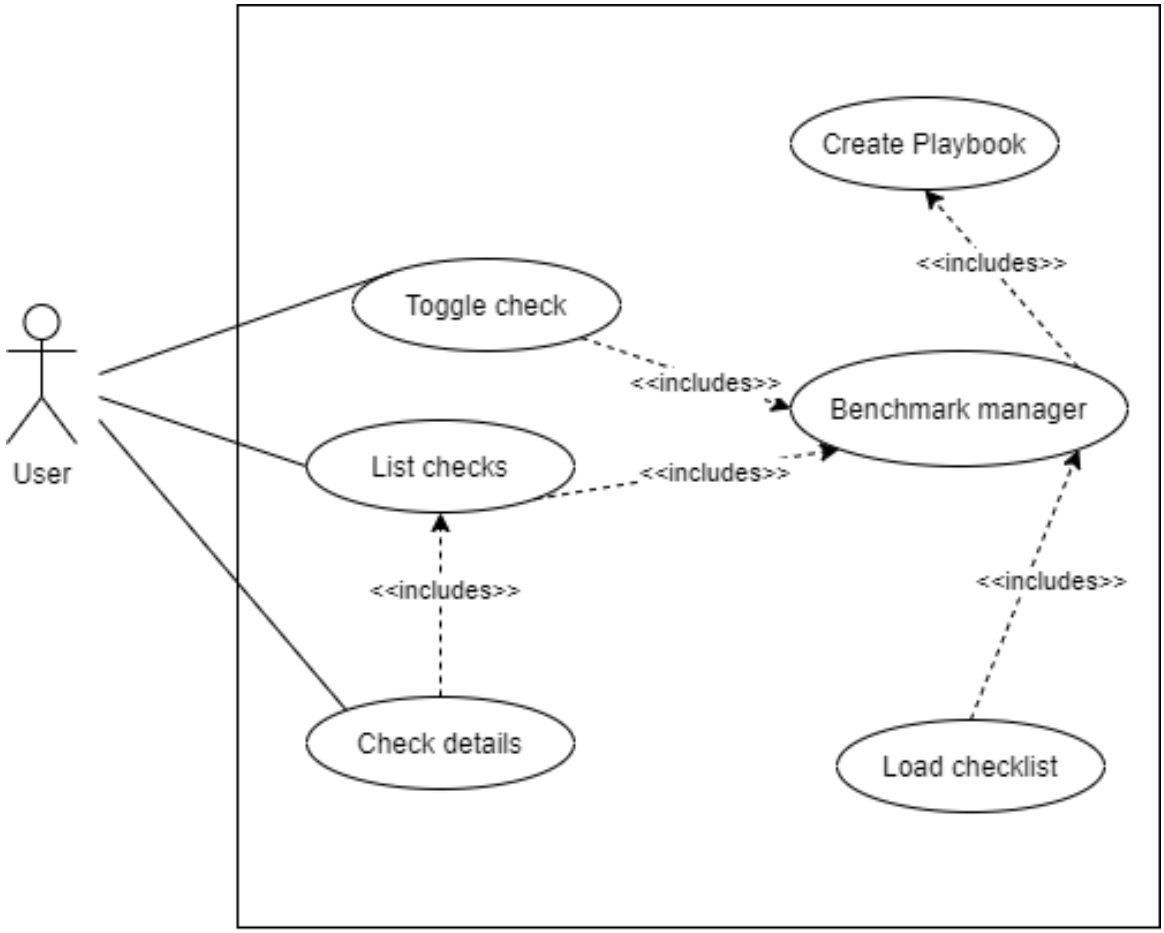


Figure 12: Use case 2. User creates a custom configuration.

Use case name
Creation of a custom configuration
Description
The user wants to create a new configuration. He can choose to list all available checks, show all the available information of a concrete check and, if needed, toggle the state for one or more at a time.

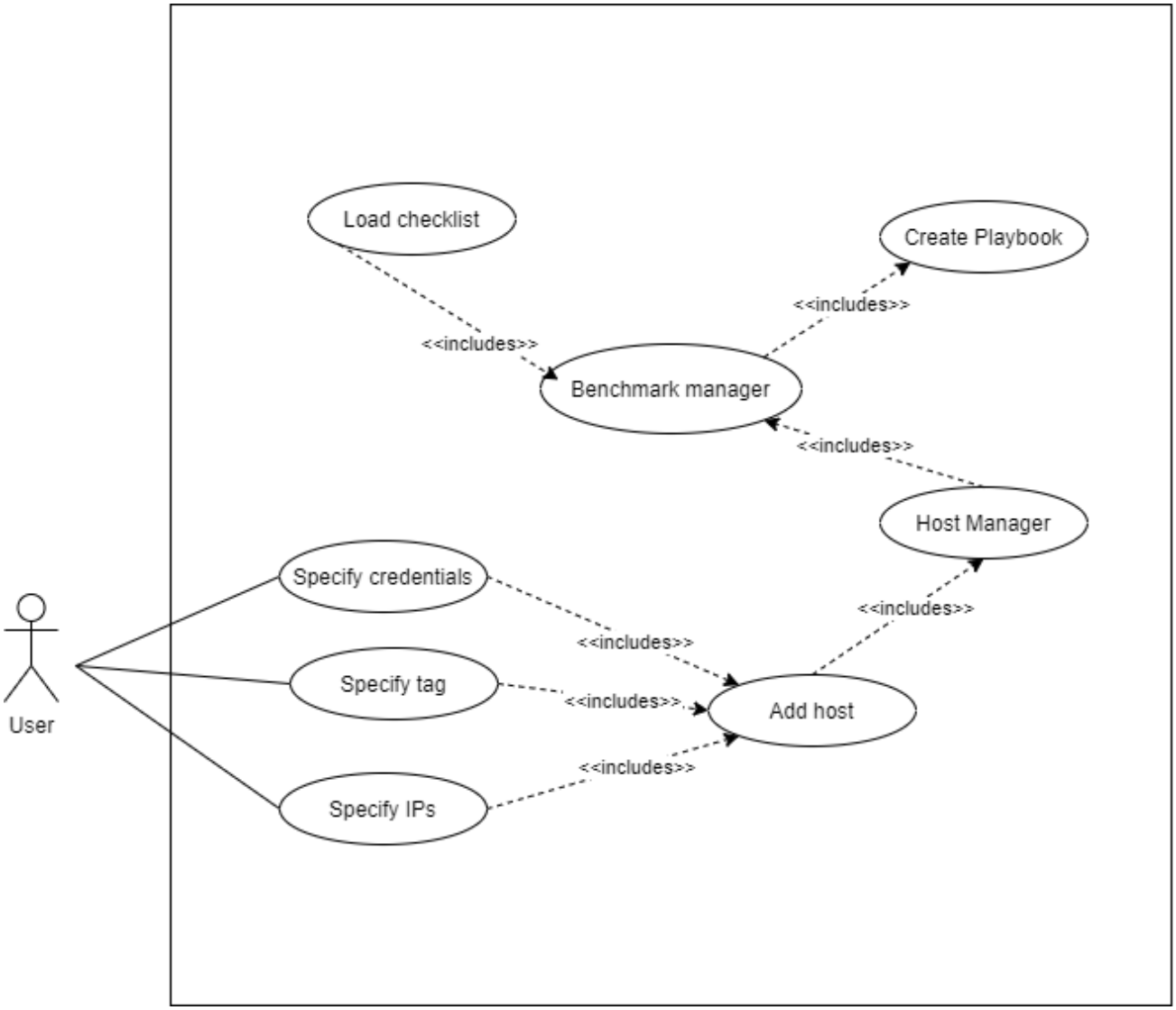


Figure 13: Specify a host

Use case name
Specify a host
Description
The user will create a custom ansible inventory including a set of credentials (username/password), one of more IPs that identify the hosts in the network and a tag to encapsulate all the information.

6.3 ISA 3: ANALYSIS SUBSYSTEMS IDENTIFICATION

6.3.1 Subsystems Description

Subsystem: Benchmark manager

This is the main subsystem of the application. It is in charge of parsing and modeling into objects the content of the CIS Benchmark, managing the list of checks. It allows the user to search for checks, enable and disable them and show specific details of each one.

Subsystem: Host manager

The host manager is responsible of registering new hosts inputted by the user. It stores a group of IPs with a set of credentials to access its core. Additionally, it also specifies connection configurations such as the protocol to be used.

Subsystem: Command executor

This subsystem focuses on the execution of Ansible commands. It manages the import of all necessary modules from Ansible Galaxy and, based on the information given by the host and benchmark managers, executes the checks specified by the user over the set of hosts retrieved by the host manager.

Subsystem: CLI

This module develops its main functionality serving a set of text menus that allow the user to communicate with the application by means of keyboard inputs.

Subsystem: GUI

The GUI subsystem is the responsible for showing and managing graphical user interface. It is necessary for registering all actions and events produced in the app.

6.3.2 Description of the Interfaces between Subsystems

The subsystems previously described will communicate within each other locally.

6.4 ISA 4: USE CASE ANALYSIS

6.4.1 Use Case 1: Ansible Execution

ANSIBLE EXECUTION	
PRECONDITIONS	Default inventory file is defined. Ansible is installed in the machine.
POSTCONDITIONS	As the user has not specified any custom configuration, all controls must be applied. The recap must show that no task was skipped. In addition to this, as no host is specified, the system will order ansible to use the default inventory file and apply the configuration to the hosts registered there.
ACTORS	User Ansible Galaxy Ansible Playbook
DESCRIPTION	This case represents the execution of a default configuration. The user will first import all necessary modules ordering to do it to the system. After that, the user can execute a default configuration over the machines included in the inventory file. After Ansible has finished its execution, a report containing the results of the analysis will be executed.
VARIATIONS	1. Host file defined If a host file has been previously defined in the application, the system will ignore the default Ansible inventory and instead use the custom one. This means that the execution will be done over the IPs registered there. 2. Custom configuration Modifications in the state of one or more checks will result in a different output as the tasks assigned to these will appear as skipped, that is, the system will not report back its current value or make any changes.
EXCEPTIONS	If the connection between the local machine and the host is lost while changes are being applied, only the registers where the configuration was already changed will be checked. However, if the connection is not available at the start, the host will be marked as unreachable, and the system will inform the user.
NOTES	If the credentials are incorrect for the hosts defined, the connection will also be refused.

Table 3: Use Case 1 Description

6.4.2 Use Case 2: Creation of a Custom Configuration

CREATION OF A CUSTOM CONFIGURATION	
PRECONDITIONS	Benchmark must have loaded the checklist.
POSTCONDITIONS	After the operation, the user can list all the checks and see that the ones that were selected have been toggled. These changes will persist until the application is closed.
ACTORS	User
DESCRIPTION	This case represents the set of actions that a user can input to manage and modify the benchmark. There are three main actions, list, toggle and show details. To toggle a check, one or more codes must be introduced by the user. If the code matches at least one of the checks registered in the benchmark, the status will be switched from ENABLED to DISABLED and vice versa. To show check details, the user must introduce the exact code of the desired control.
VARIATIONS	1. Introduce a regular expression to toggle If a special character is introduced, the system will consider it as a regular expression while evaluating it against all check codes.
EXCEPTIONS	If multiple codes are introduced when obtaining details, the query operation will be aborted.
NOTES	If no toggling action is preformed, all the checks will be enabled by default.

Table 4: Use Case 2 Description

6.4.3 Use Case 3: Host Specification

SPECIFY A HOST	
PRECONDITIONS	None
POSTCONDITIONS	After the operation, a new Inventory file will be created including a group with the IPs introduced by the user. This will be the new inventory used by ansible until the application is stopped.
ACTORS	User
DESCRIPTION	This use case represents the creation of a new Inventory file. It will contain a group of IPs under the given tag, and it will associate the
VARIATIONS	1. Registration of multiple IPs The field asking for IPs must be able to process as many IPs as the user wants to register. The order in which they are provided by the user will mark the order of execution for Ansible.
EXCEPTIONS	1. If the process is stopped before the confirmation of creation, the data introduced by the user will be lost. The interface will reset its values. 2. If any of the IPs given by the user does not meet the IPv4 format, it will be rejected and the system, which will alert of the problem. 3. If multiple IPs are registered, the hosts that they represent must have an account with the same username and password as the credentials specified in the application. 4. If the tag is not formed by the abecedary letters, both capital or lower case, the system will reject the value and ask the user for a new one, as it must require Ansible specifications.
NOTES	-

Table 5: Use Case 3 Description

6.4.3.1 Annex: Robustness Diagrams

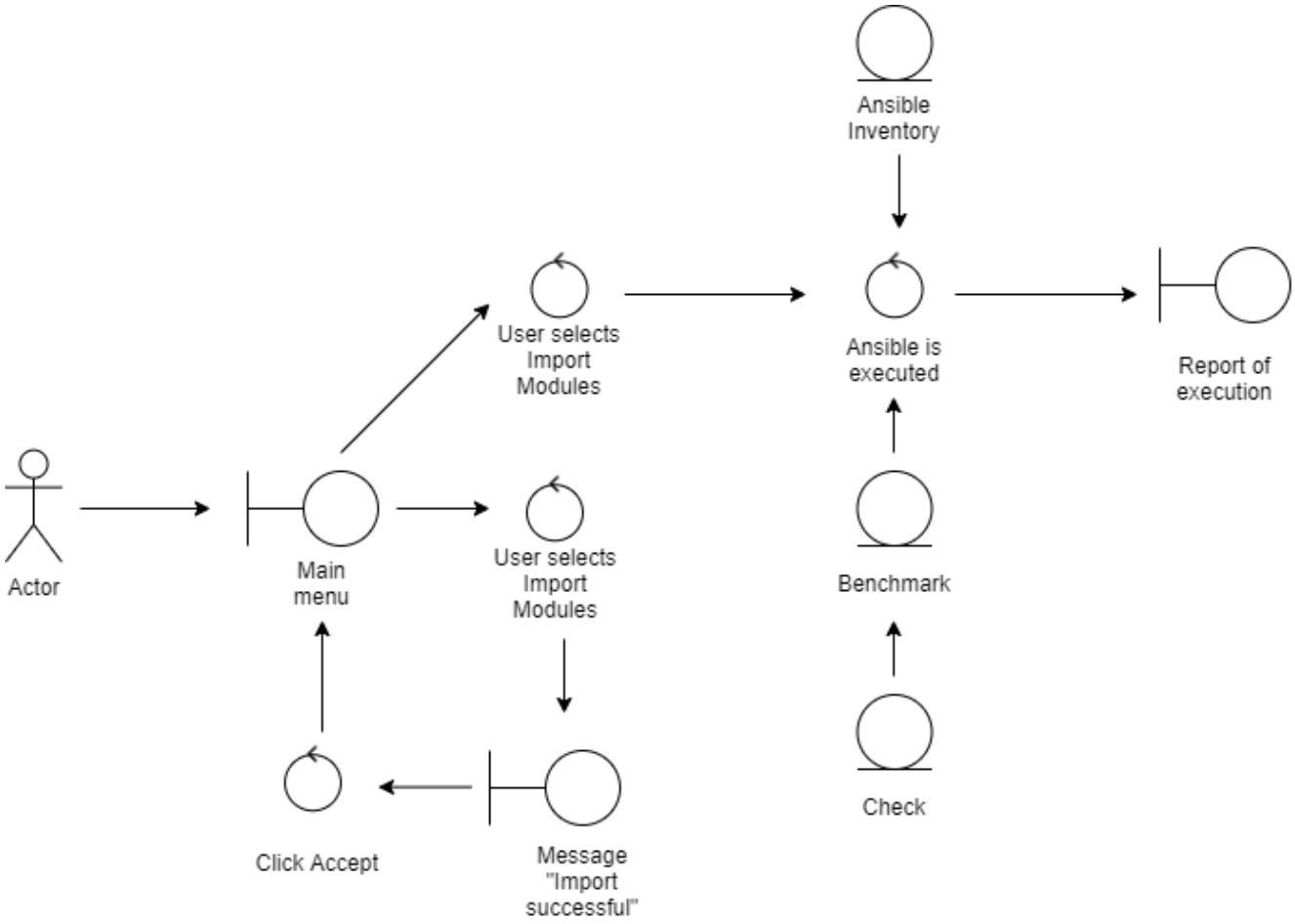


Figure 14: Robustness diagram for Use Case 1. Default Ansible execution

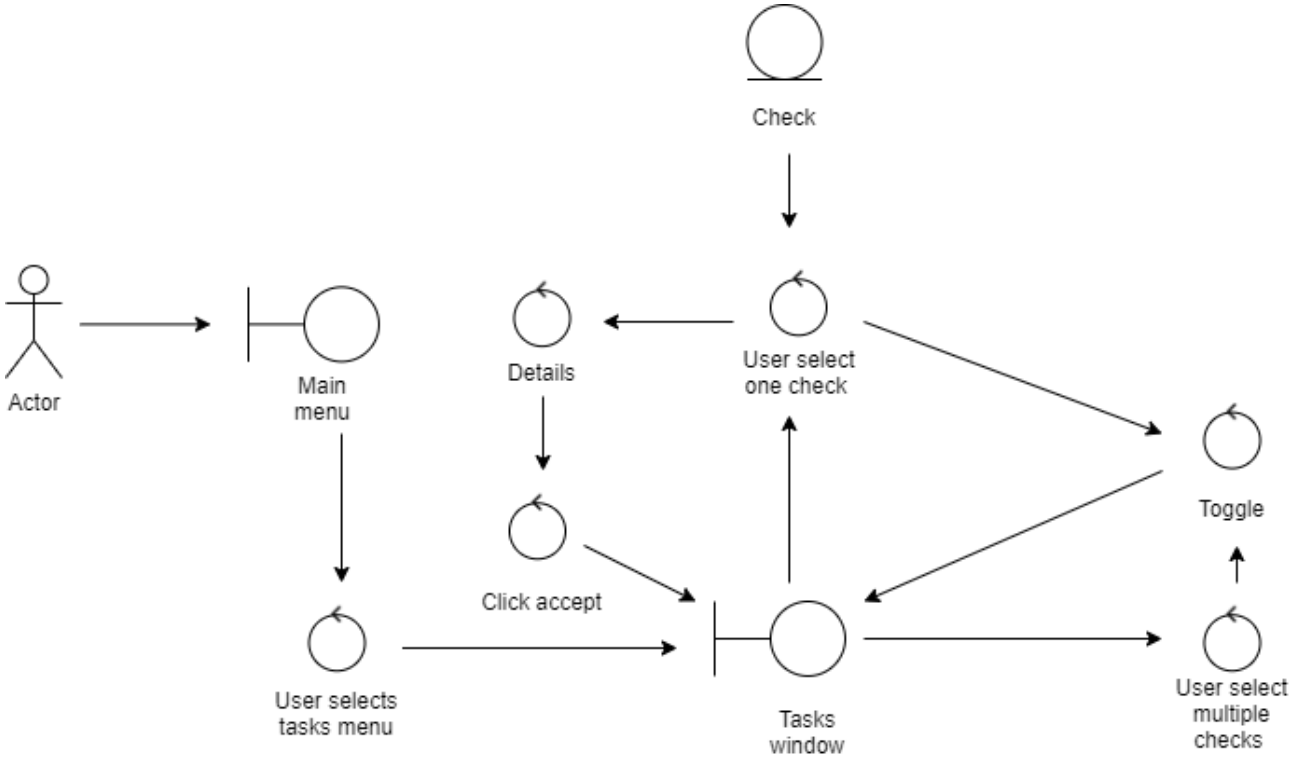


Figure 15: Robustness diagram for Use Case 2: Creation of a custom configuration.

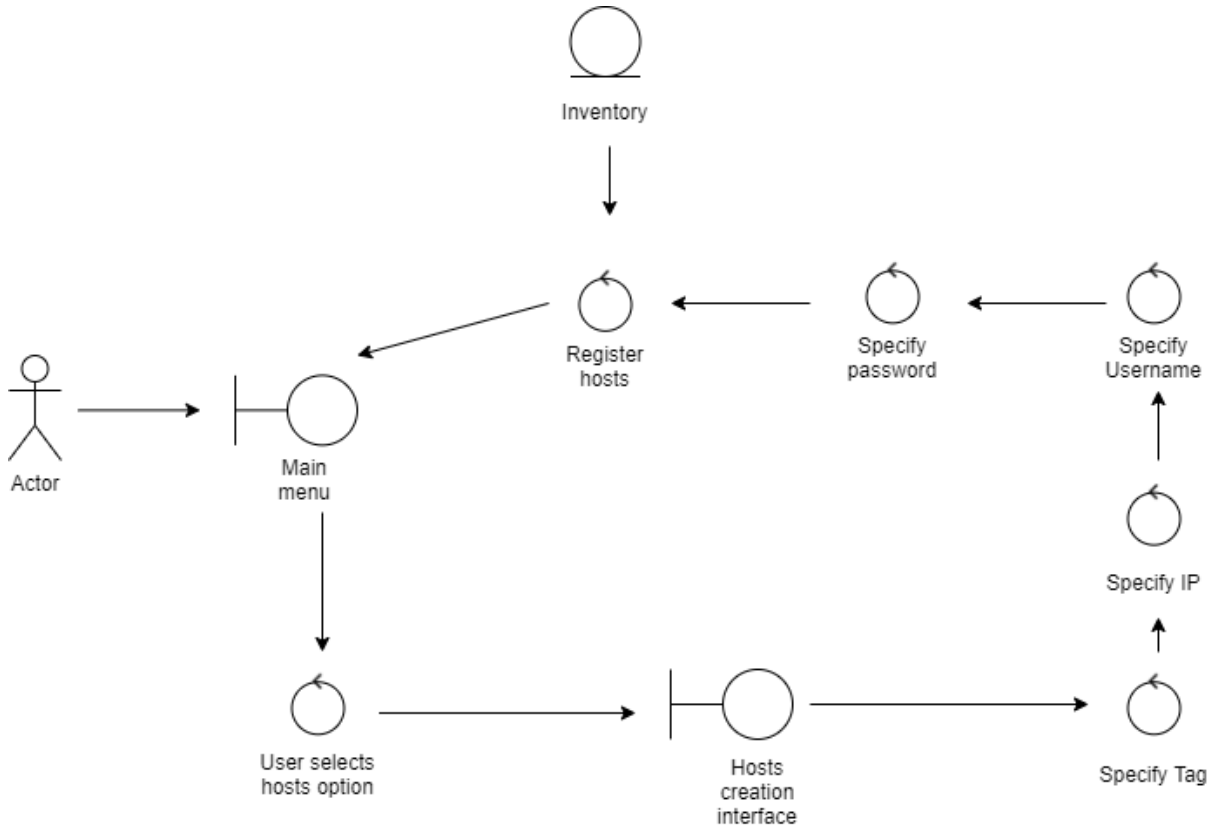


Figure 16: Robustness diagram for Use Case 3: Specify a Host

6.5 ISA 5: CLASS ANALYSIS

6.5.1 Class Diagram

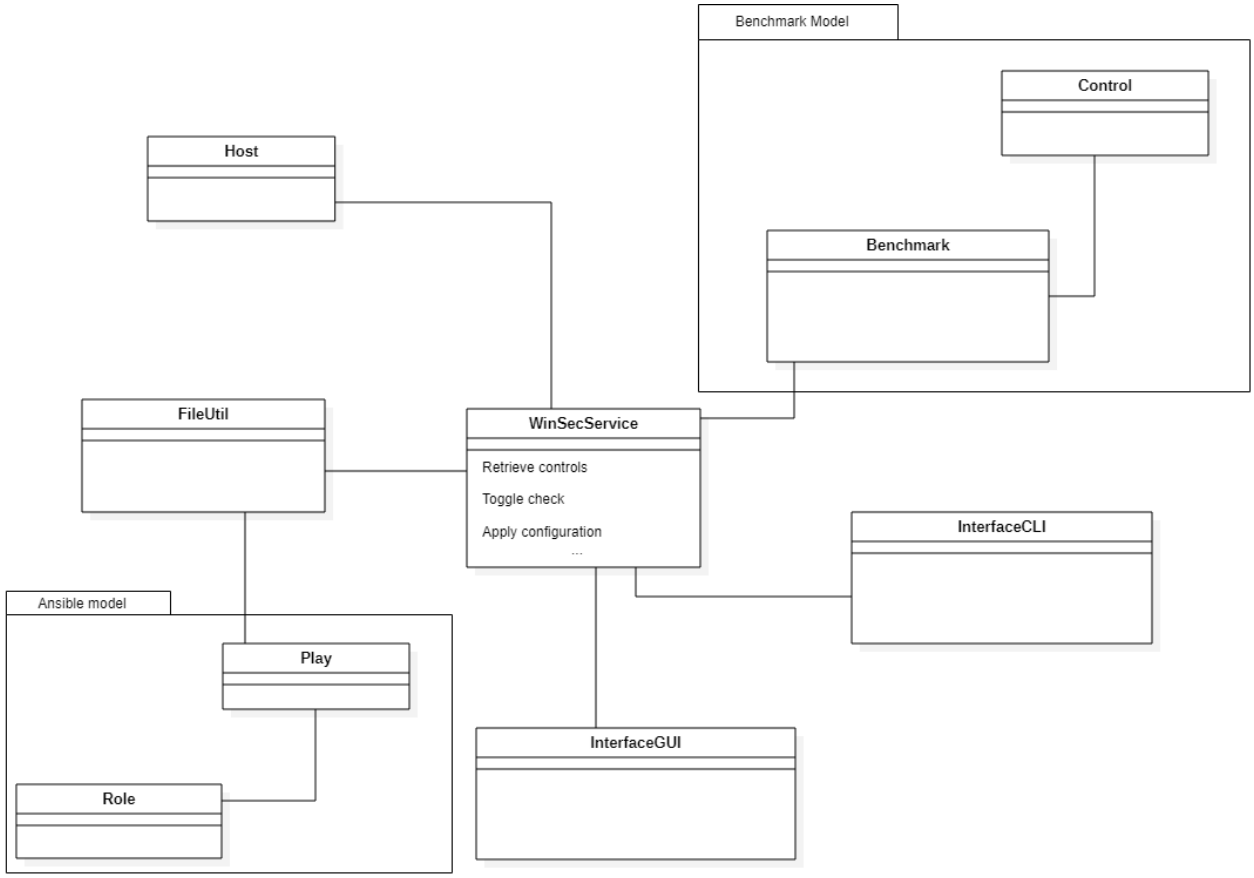


Figure 17:Base class diagram

6.5.2 Class Description

6.5.2.1 Subsystem 1

Name
Ansible Model
Description
This represents the structure needed by ansible to create and execute a playbook. A playbook (“Play”) will contain one or more roles.
Responsibilities
Represent the structure of a valid Ansible playbook. It must also consider a flag that will indicate which tasks need to be executed.

Proposed attributes
roleName: Name of the role ***flag:*** String containing the tasks to be executed.
Proposed Methods
-

6.5.2.2 Subsystem 2

Name
Benchmark model
Description
This represents the structure defined by CIS to structure its Benchmarks. A Benchmark is formed of a set of controls that can be switched on or off creating custom configurations.
Responsibilities
Represent the structure of the CIS Benchmark. Manage controls and allow the user to create custom configurations.
Proposed attributes
controls: Set of controls of the Benchmark.
Proposed Methods
toggle(id): Switches the state of a control. ***listControls():*** Retrieves the list of controls.

6.6 ISA 8: USER INTERFACES DEFINITION

6.6.1 Interface Description

6.6.1.1 Command-line interface (CLI)

The Command-line Interface consist of a menu offering the different features of the system. To encapsulate the customization of the benchmark, all check related functionality will be held in a secondary menu.

Each time the user introduces some input and presses the ENTER key, the system will inform the of its choice in a log format that can be consulted in Fig.19. As it can also be noticed, incorrect data will generate messages with an ERROR tag, while those that reflect a change in the application will contain the SUCCESS tag. Colors will be used in these messages to alert the user of every meaningful action.

The user will be able to navigate the menu by introducing the number associated to the option. Every time the software requires for a user input, the system will display a message with an arrow, as shown in Fig.19.

Each feature will automatically display its output and after that, the last menu were the user was situated will be printed again.

Each time the app is run, the ASCII logo of the app will be displayed (Fig.20).

6.6.1.2 Graphical User interface (GUI)

The Graphical User Interface is composed of a side menu where the user can navigate through the different windows, and the main working zone displayed to the right. Every window will maintain this basic design:

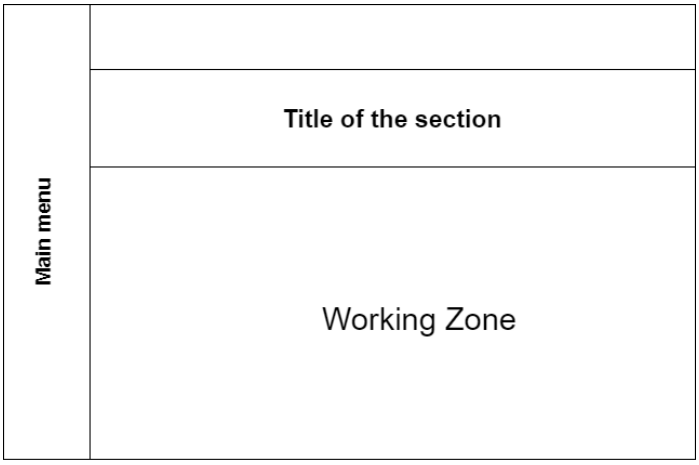


Figure 18: Basic GUI Schema

6.6.2 Interface Look Definition

6.6.2.1 Command-line interface (CLI)

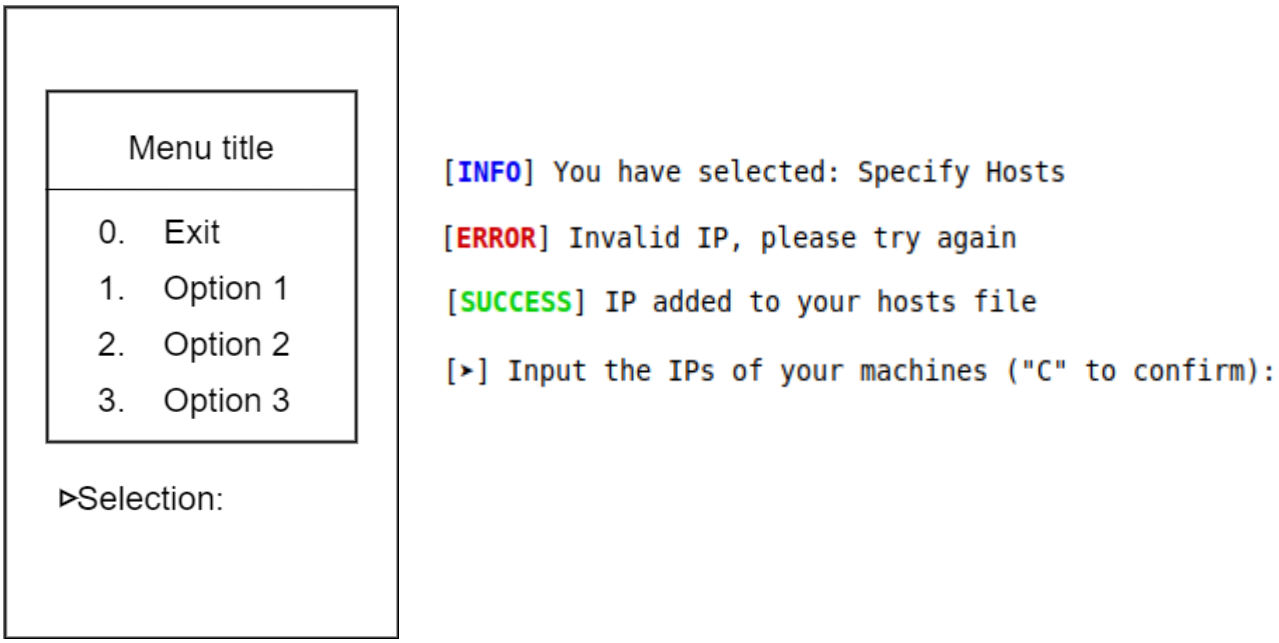


Figure 19: Menu and system message prototypes

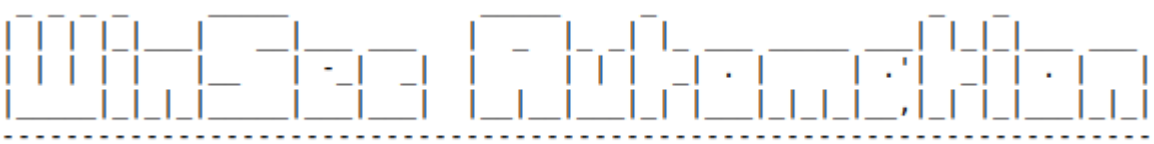


Figure 20: Menu logo for the application

6.6.2.2 Graphical User Interface

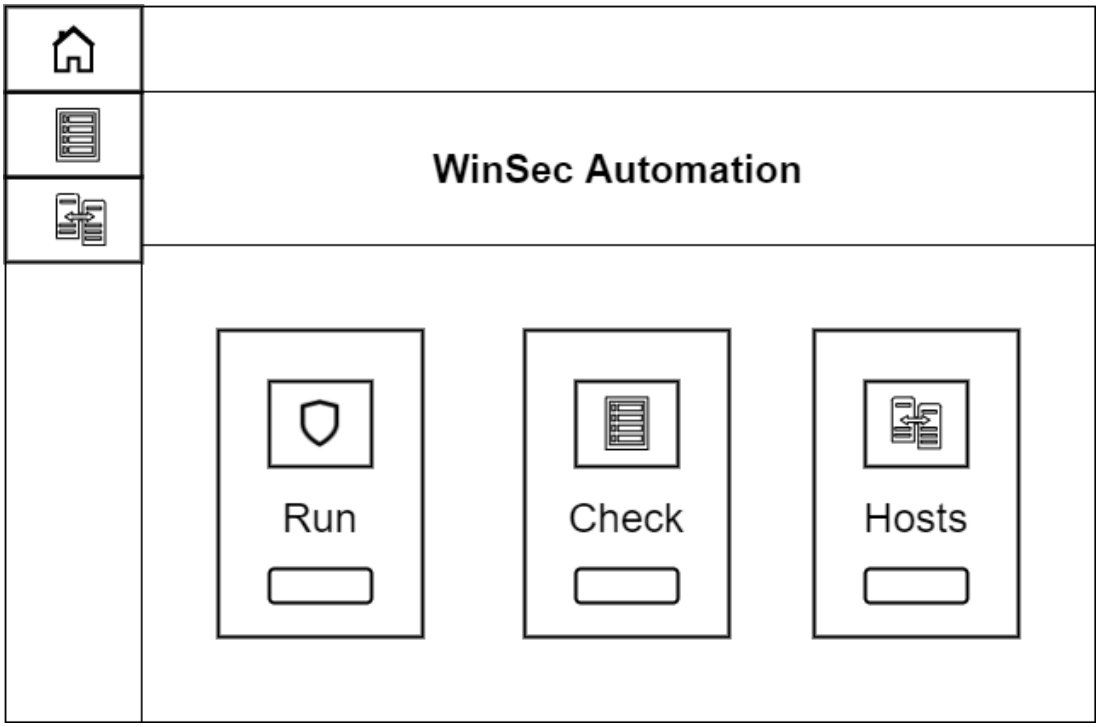


Figure 21: Main Window

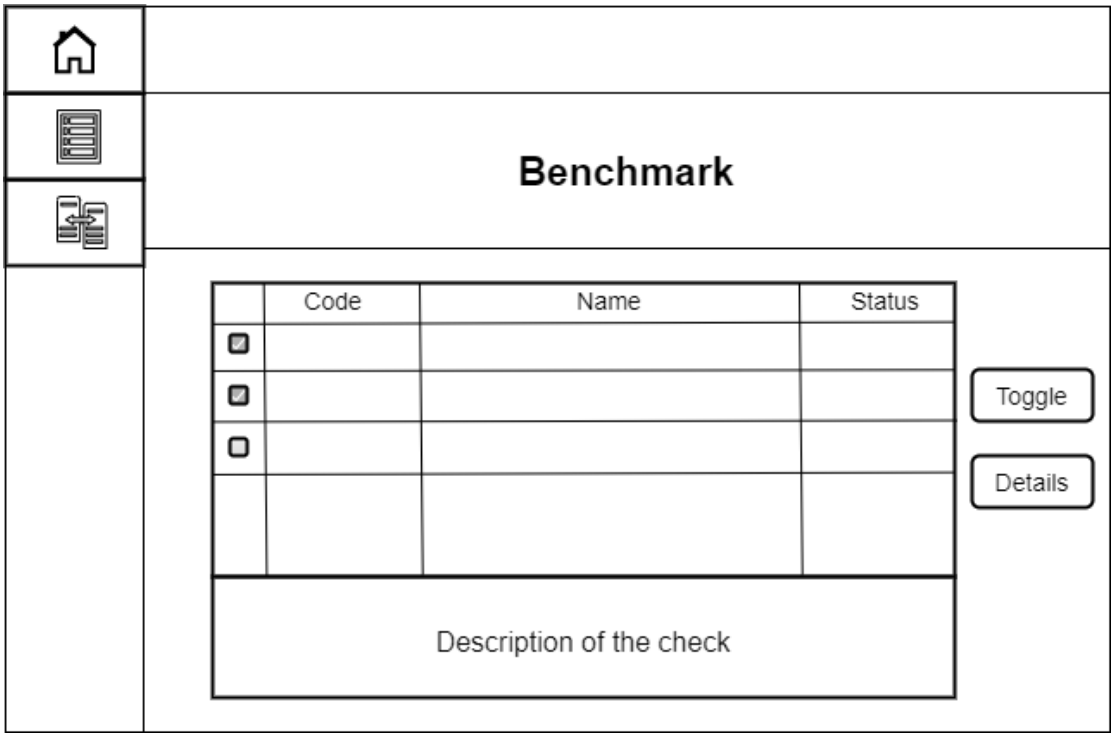


Figure 22: Benchmark Window

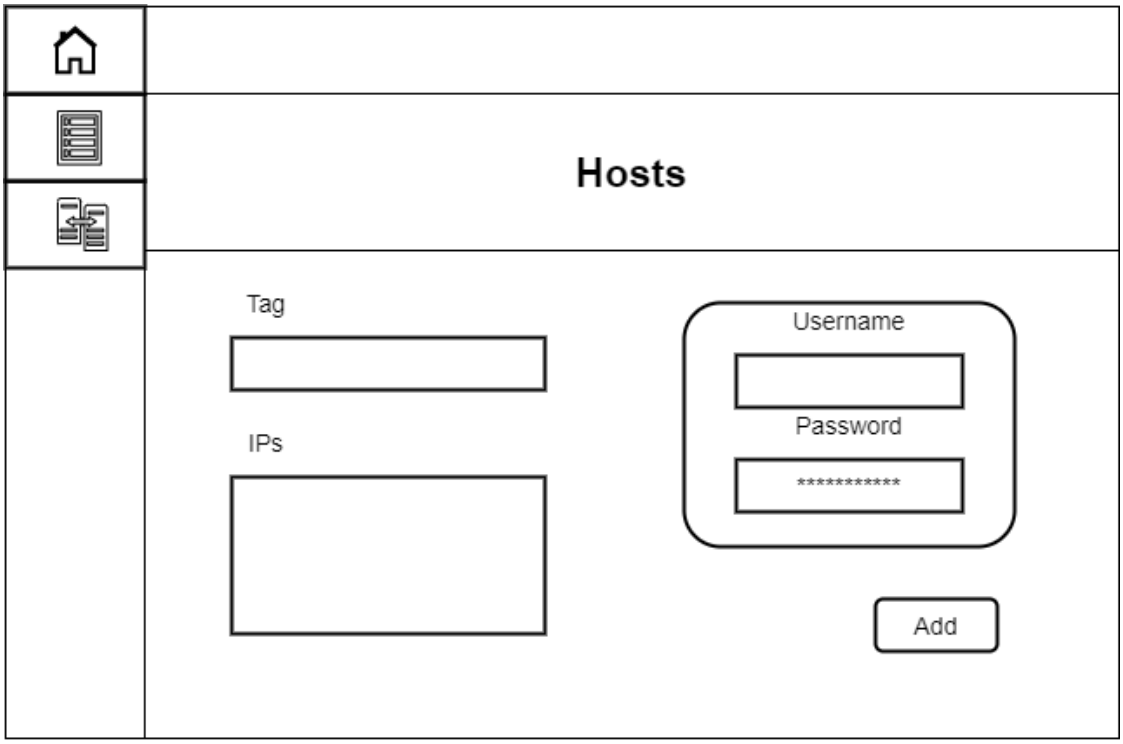


Figure 23: Hosts Window

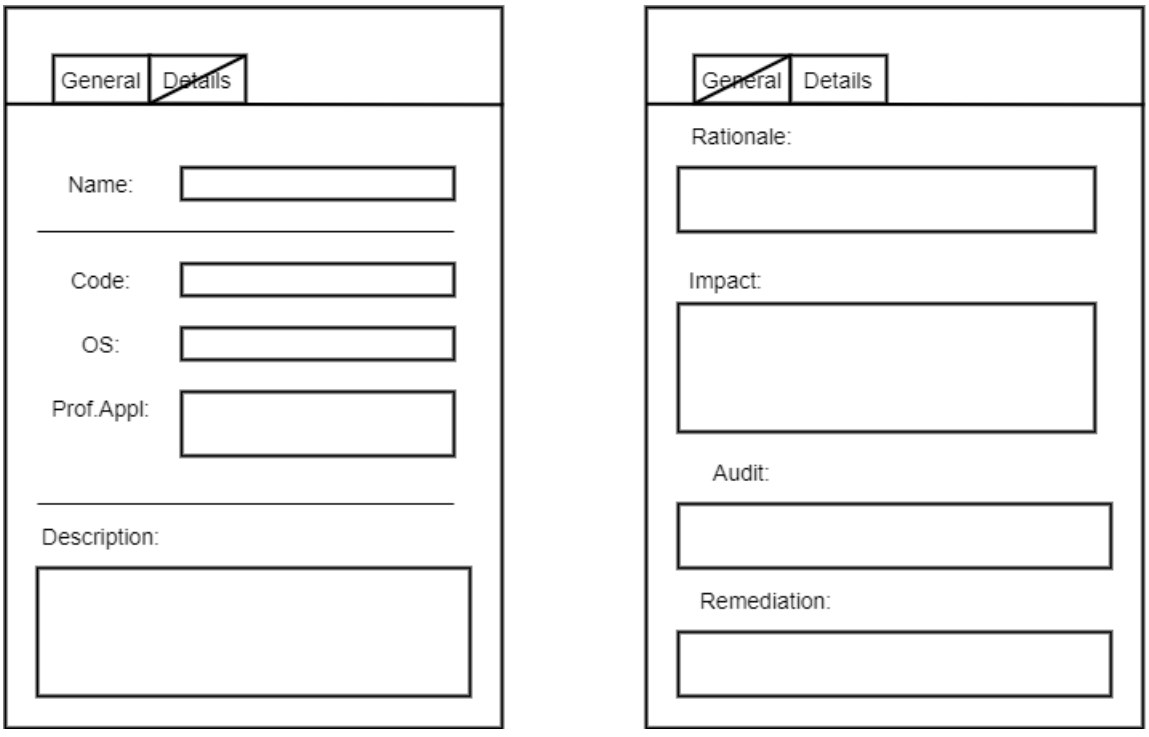


Figure 24: Details View

Figure 25: Results window (while Ansible is executing)

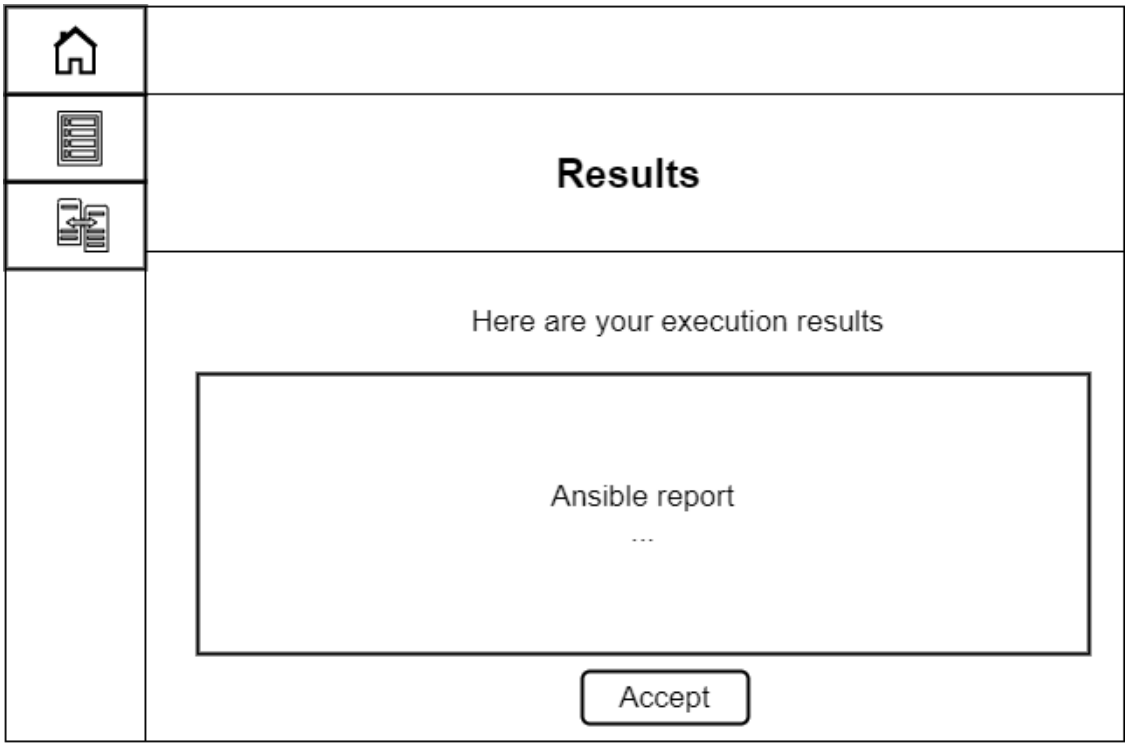


Figure 26: Results Window (with Ansible Report)

6.6.3 Interface Behavior Description

Firstly, in the case of the CLI menus only accept the numbers appearing in the tables. In any other value is introduced, the system will reject it and ask again.

Besides that, as both the GUI and CLI require the same input validation, data validation will be covered in relation to both interfaces.

When adding a host, the tag must be a set of alphabetical characters, that is from A to Z both capital and lower case. For Ips, they must meet the IPv4 standard.

6.6.4 Navigation Diagram

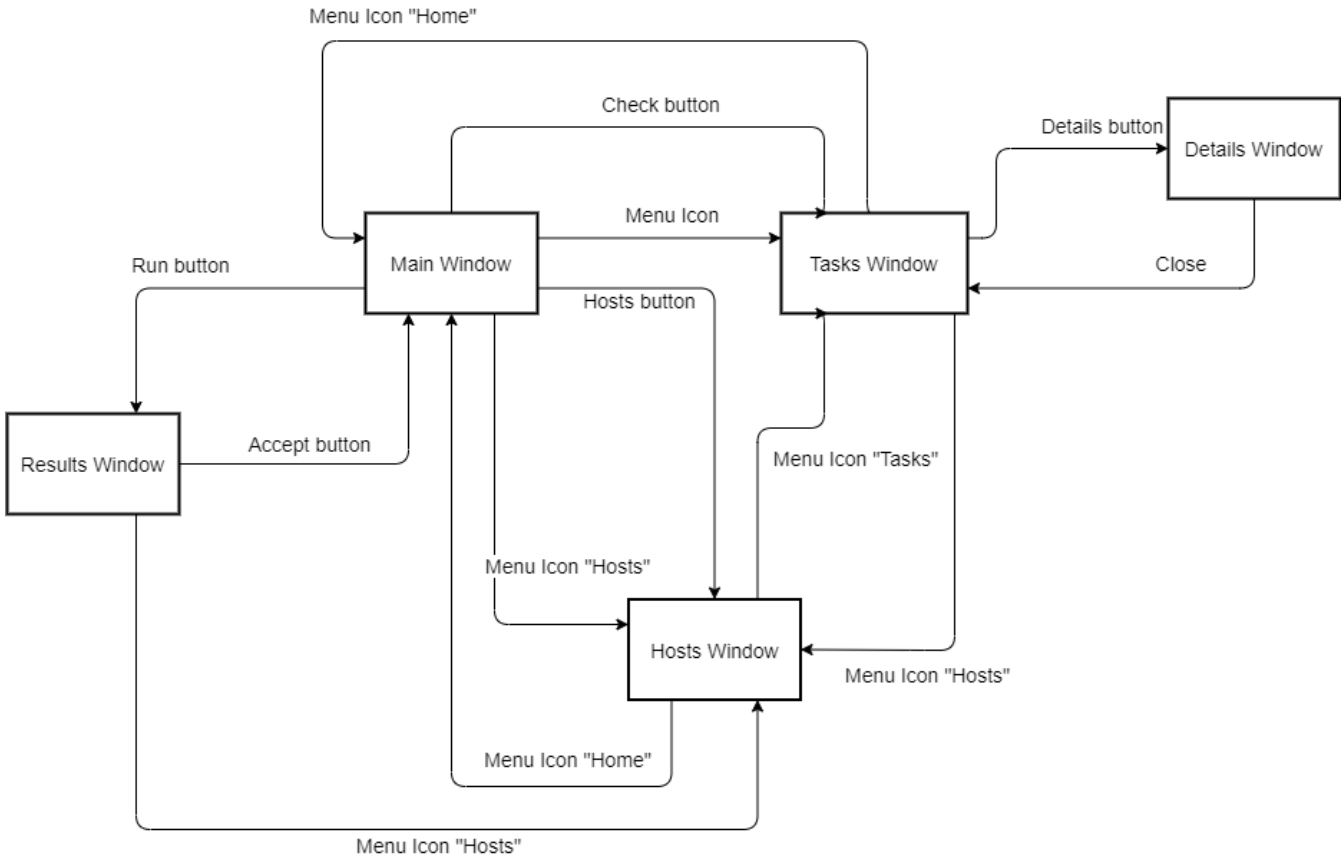


Figure 27: Navigation diagram

6.7 ISA 10: TEST PLAN SPECIFICATION

Use Case 1: Ansible Execution	
Test	**Expected result**
Default configuration. Host misconfigured. Default inventory.	Ansible will apply all the available checks to the machine Windows Server machine. All values will be updated.
Test	**Expected result**
Default configuration. Host misconfigured. Custom inventory.	Ansible will apply all the available checks to the machine Windows Server machine. All values will be updated.
Test	**Expected result**
Default configuration. Unreachable host.	Ansible will mark the new added host as unreachable. No checks will be executed.
Test	**Expected result**
Default configuration. Host configured. Custom inventory.	Ansible will apply all the available checks to the machine previously specified. All checks should appear with state “ok”.
Test	**Expected result**
Custom configuration. Host misconfigured. Custom inventory.	Ansible will change the registers corresponding to the enabled checks. Disabled checks will appear as skipped.
Test	**Expected result**
Custom configuration. Host misconfigured. Custom inventory. Double execution.	Ansible will change the registers corresponding to the checks enabled in the first execution. For the second execution, they will appear as “ok”. Disabled checks will appear as skipped in both executions.

Table 6: Use Case 1 Test Plan

Use Case 2: Creation of a custom configuration	
Test	**Expected result**
List all checks	All controls must appear in a table, being all of them enabled. The description box should contain the description of the first check.
Test	**Expected result**
Toggle checks	All controls must appear in a table, those that have been toggled must be DISABLED. The description box should contain the description of the first check.
Test	**Expected result**
Show details	The Details Window must appear containing the information for the check selected.
Test	**Expected result**
Search checks (unique)	Only the control specified is shown.
Test	**Expected result**
Search checks (multiple)	Only the checks matching the regular expression are shown. The description box should contain the description of the first displayed check.

Table 7: Use Case 2 Test Plan

Use Case 3: Specify a host	
Test	**Expected result**
Add valid host (single IP)	A message indicating the host has been added pops out of the window. The user is redirected to the home window after closing the dialog.
Test	**Expected result**
Add invalid host (tag)	A message indicating the tag name is invalid pops out of the window. The user is not redirected after closing the dialog.
Test	**Expected result**
Add invalid host (IP)	A message indicating the IP value is invalid pops out of the window. The user is not redirected after closing the dialog.
Test	**Expected result**
Add invalid host (empty fields)	A message indicating the tag name is invalid pops out of the window. The user is not redirected after closing the dialog.
Test	**Expected result**
Add valid host (multiple IPs)	A message indicating the host has been added pops out of the window. The user is redirected to the home window after closing the dialog.

Table 8: Use Case 3 Test Plan

Chapter 7. Information System Design

DEVELOPMENT PHASE

ISD

7.1 ISD 3: REAL USE CASES DESIGN

7.1.1 Use Case 1: Ansible Execution

7.1.1.1 Sequence Diagrams

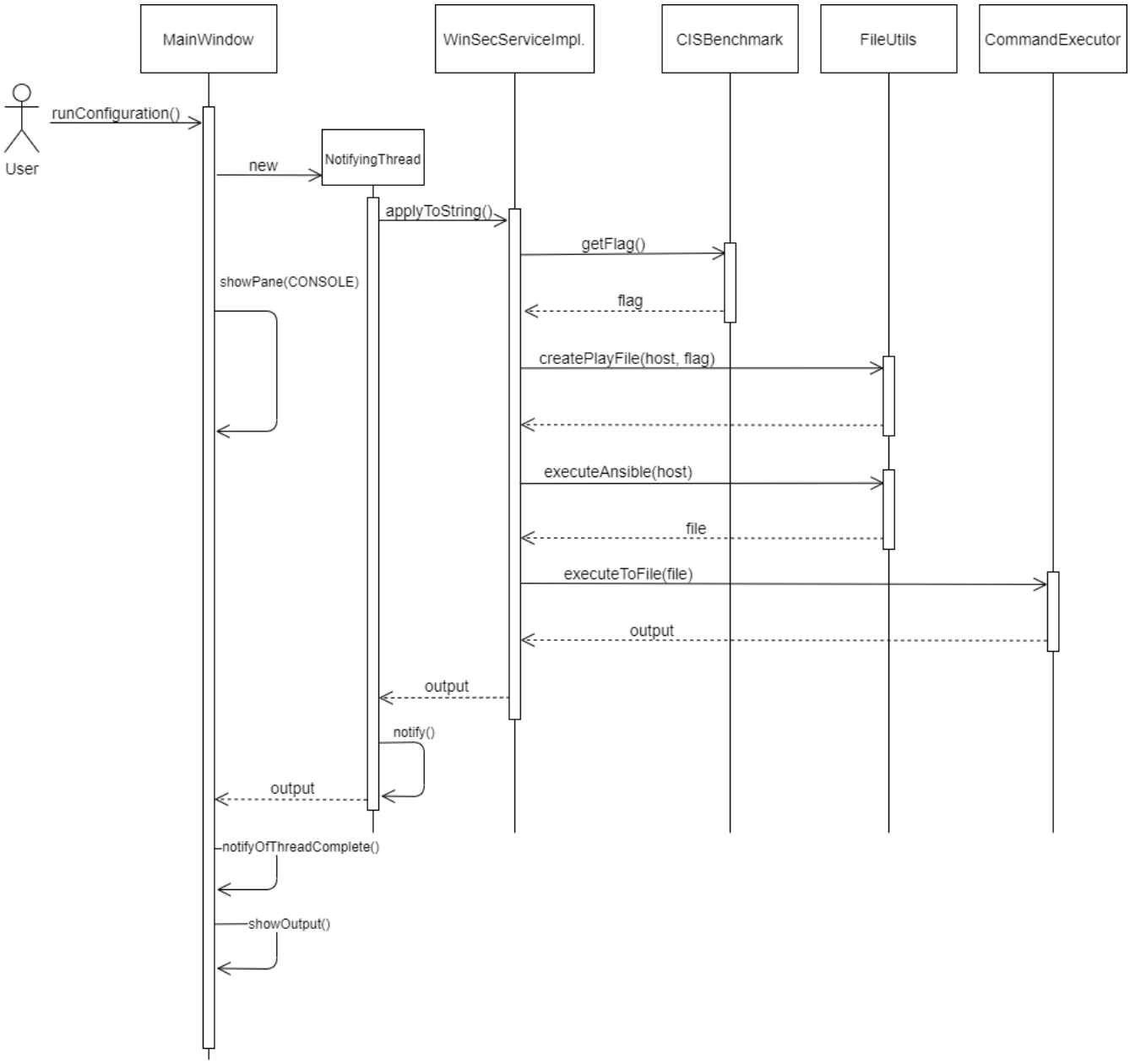


Figure 28: Sequence diagram. Ansible execution

7.1.2 Use Case 2: Creation of a Custom Configuration

7.1.2.1 Sequence Diagrams

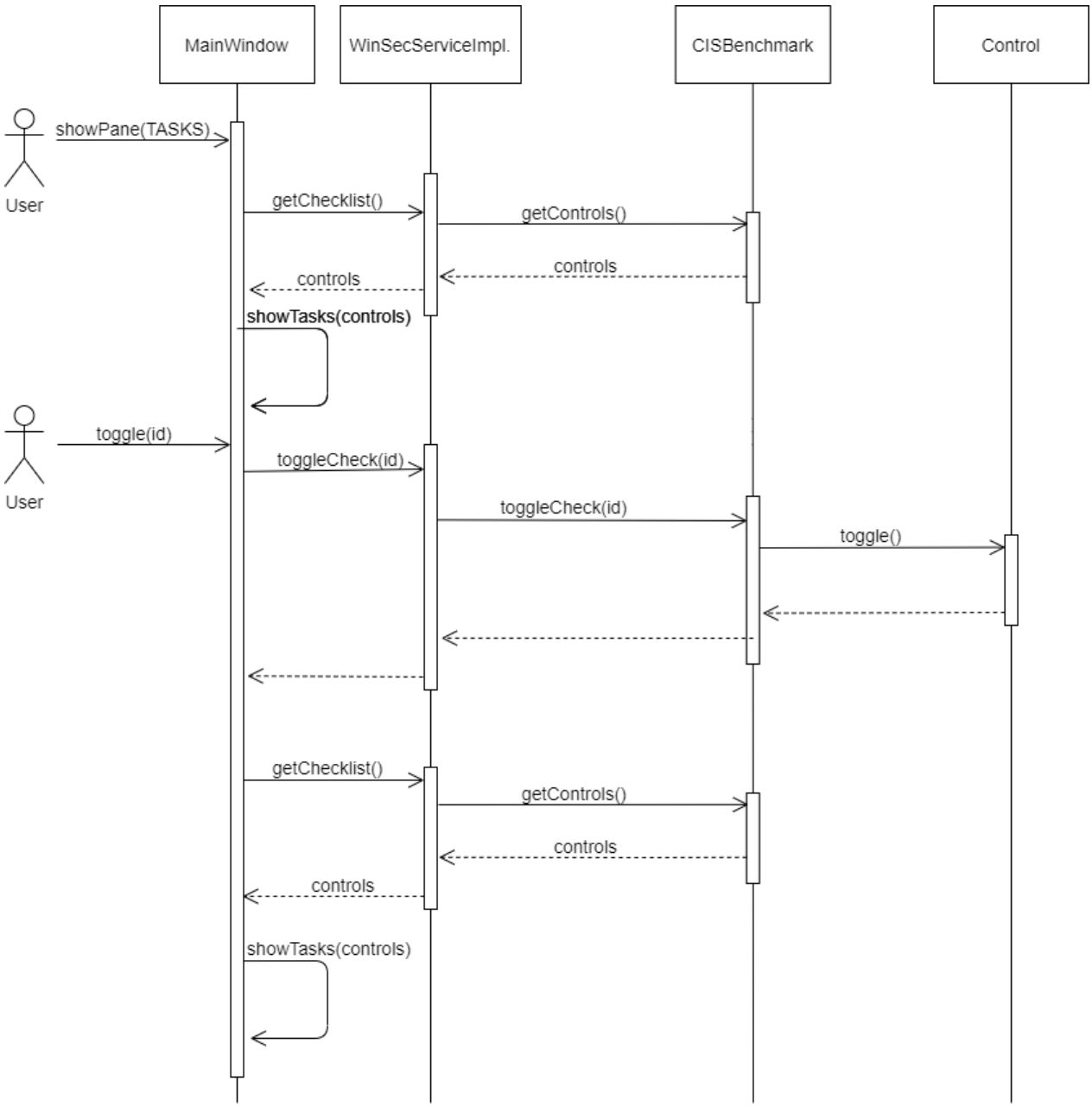


Figure 29: Sequence diagram. Custom configuration

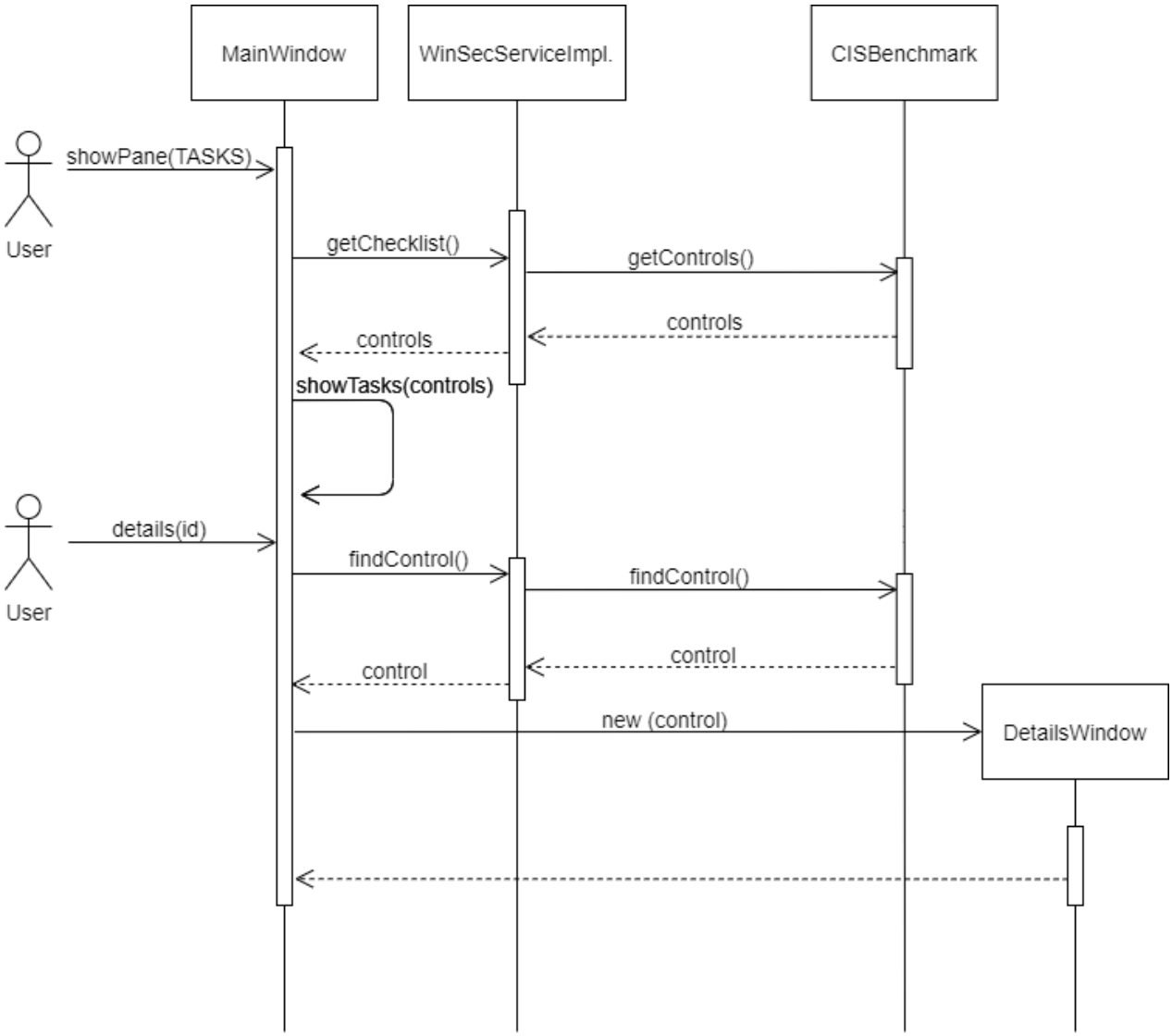


Figure 30: Sequence diagram. Show details

7.1.3 Use Case 3: Host Specification

7.1.3.1 Sequence Diagrams

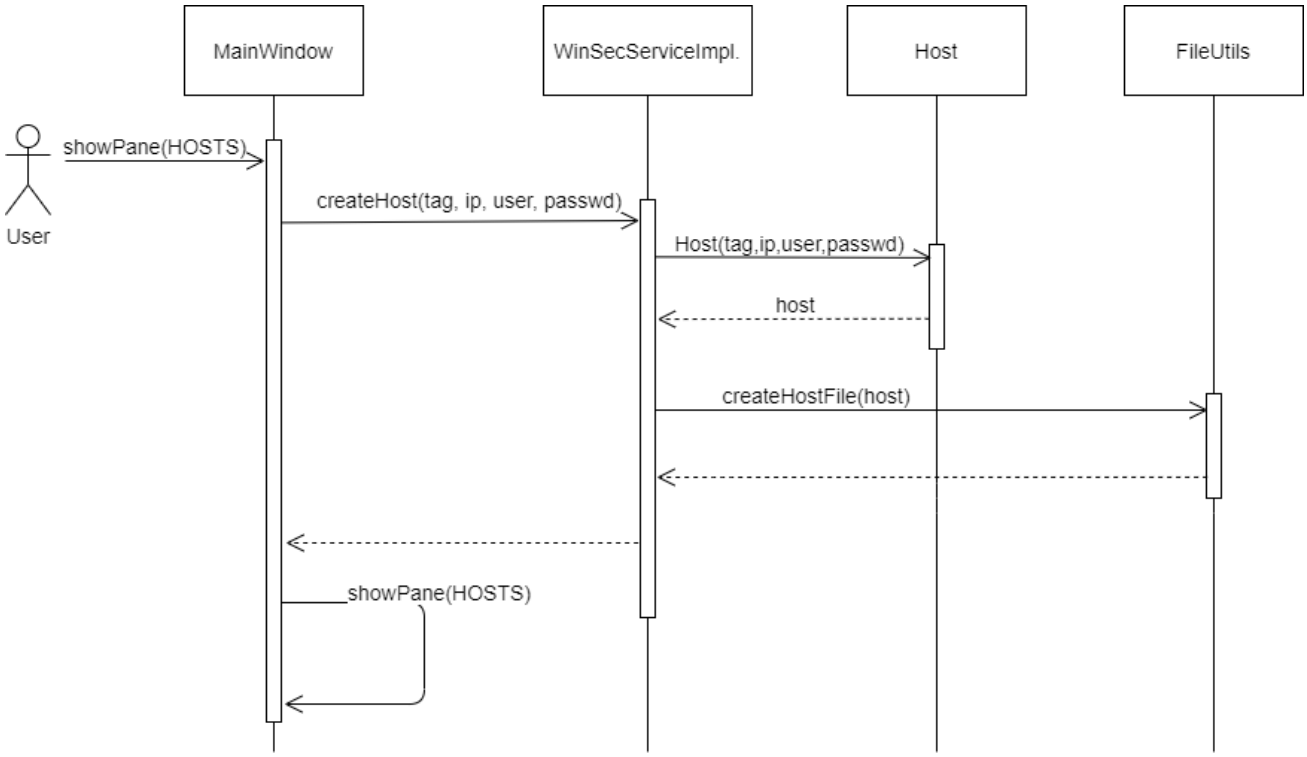


Figure 31: Sequence diagram. Host creation

7.2 ISD 4: CLASS DESIGN

7.2.1 Class Diagram

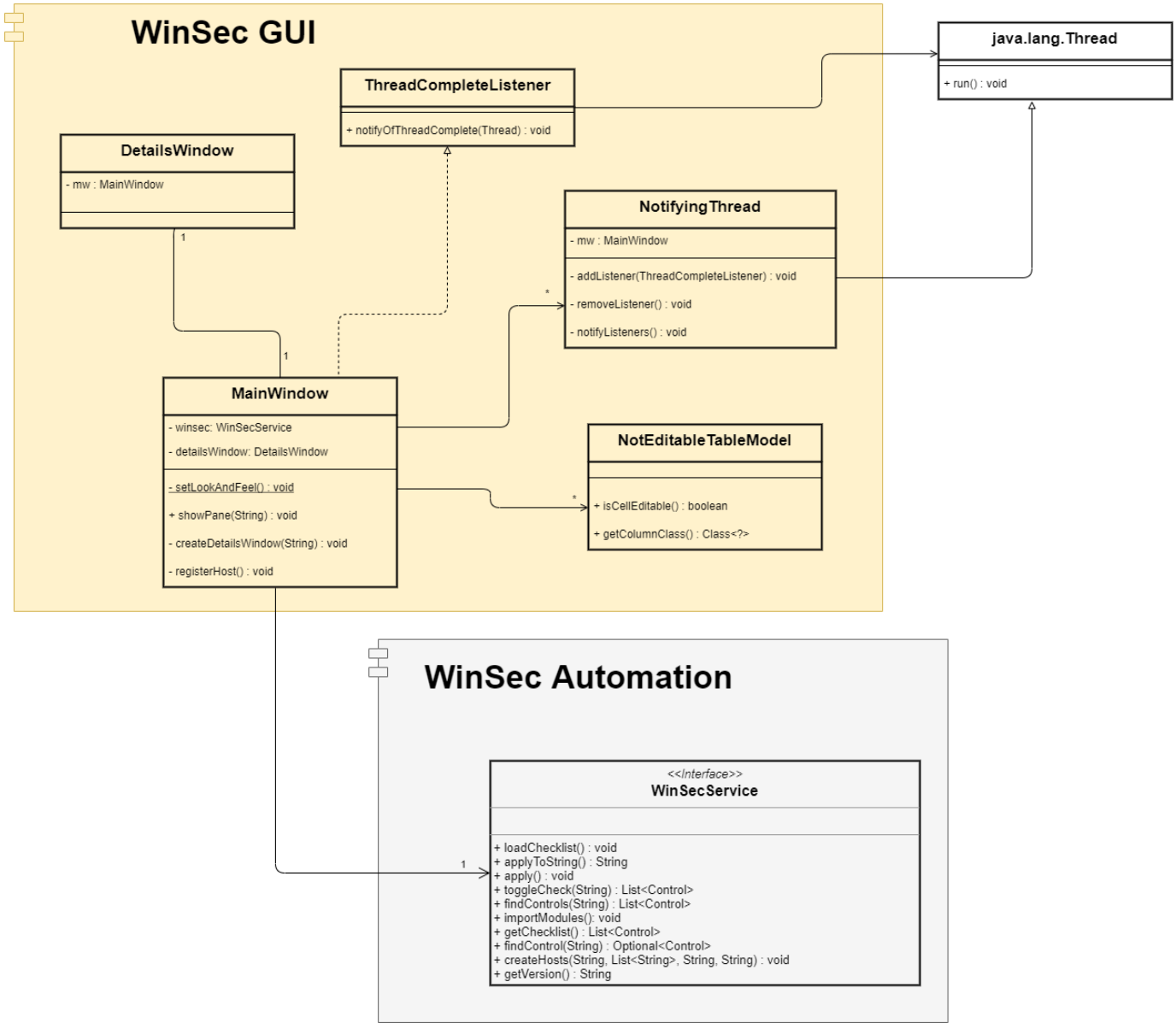


Figure 32: Class diagram. WinSec GUI

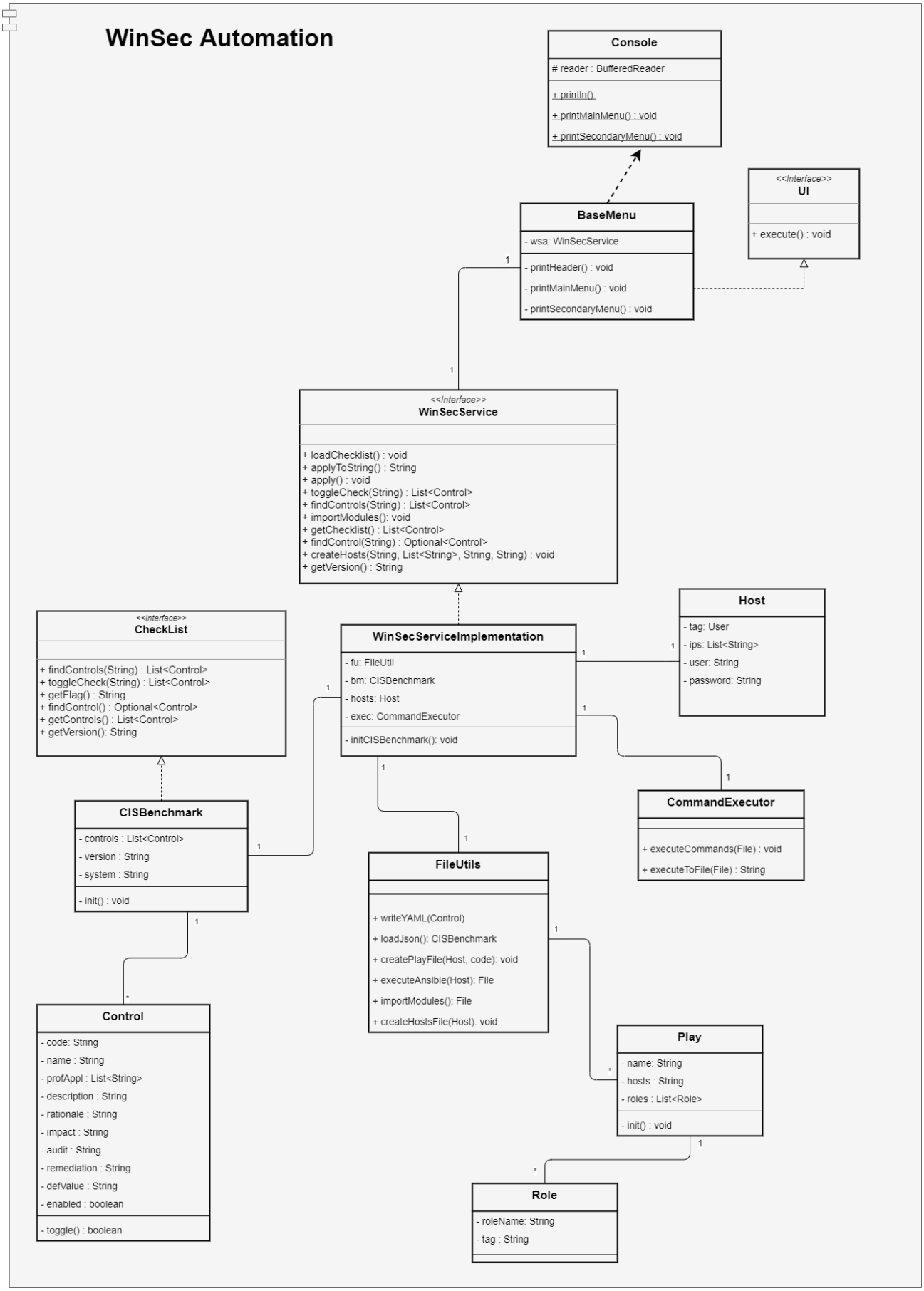


Figure 33: Class Diagram. WinSec Automation

7.3 ISD 5: System Modules Architecture Design

7.3.1 ISD 5.1 System Modules Design

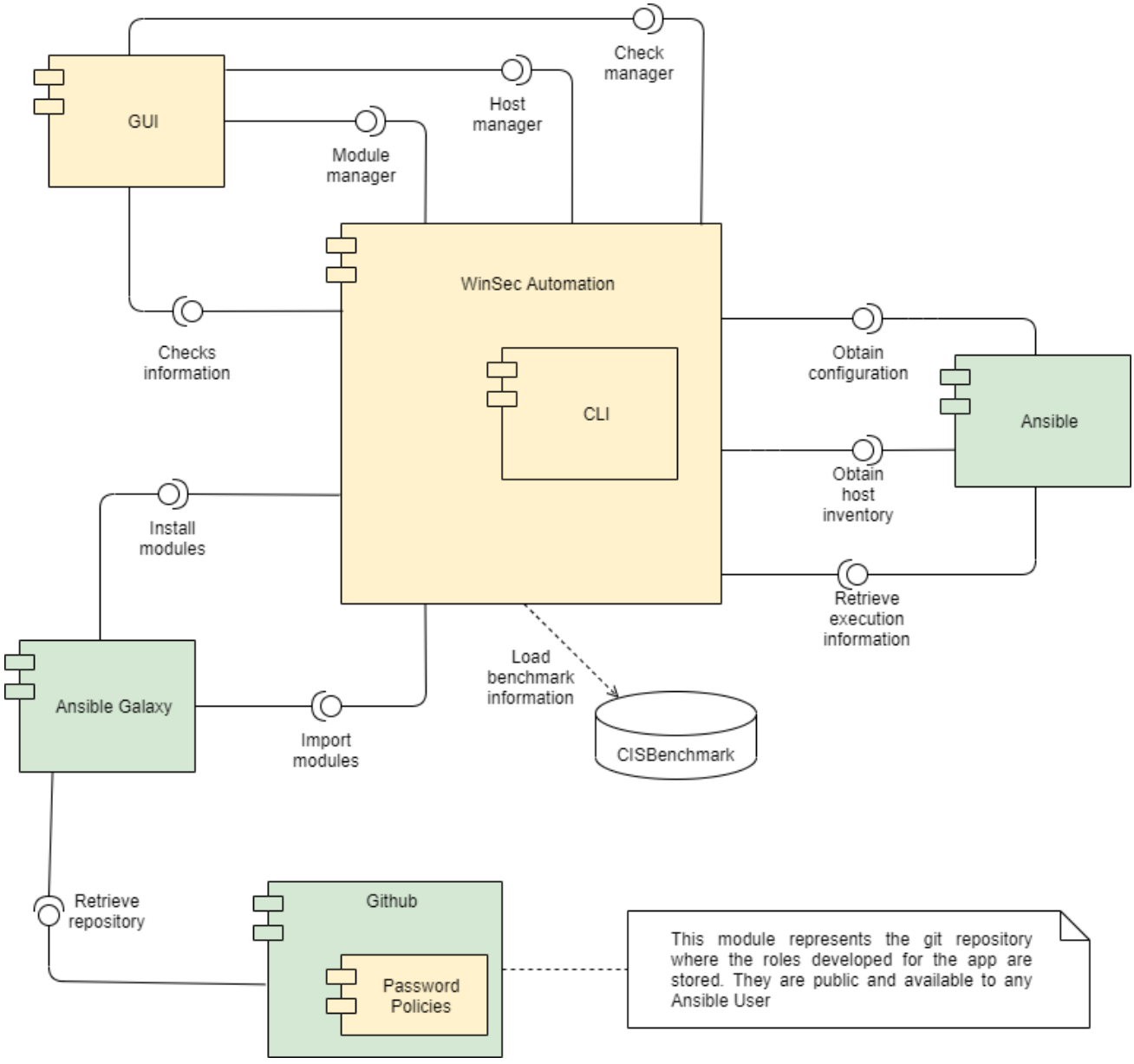


Figure 34: Component diagram

7.3.2 ISD 5.3 User Interface Revision

The final look of the application will be presented:

Figure 35: GUI-Main menu

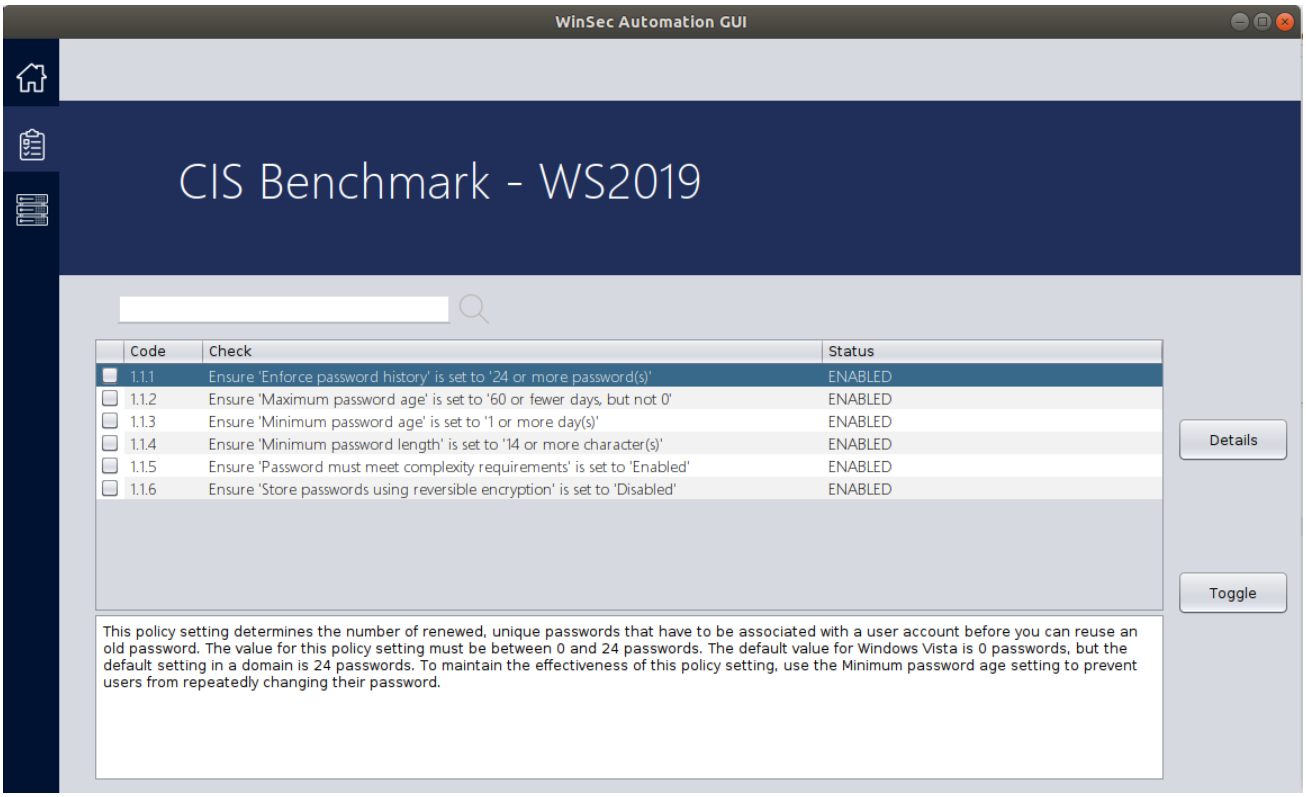


Figure 36: GUI-Tasks menu

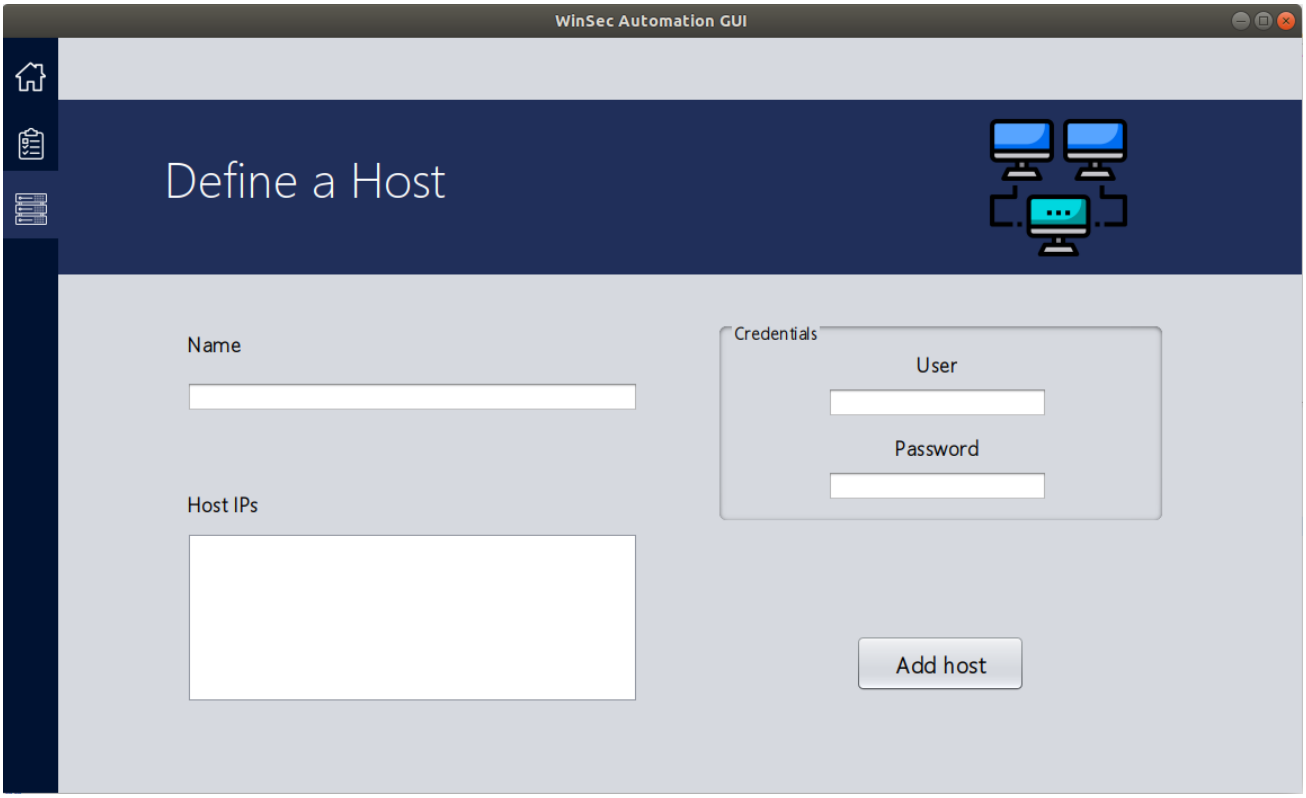


Figure 38: GUI-Host's menu

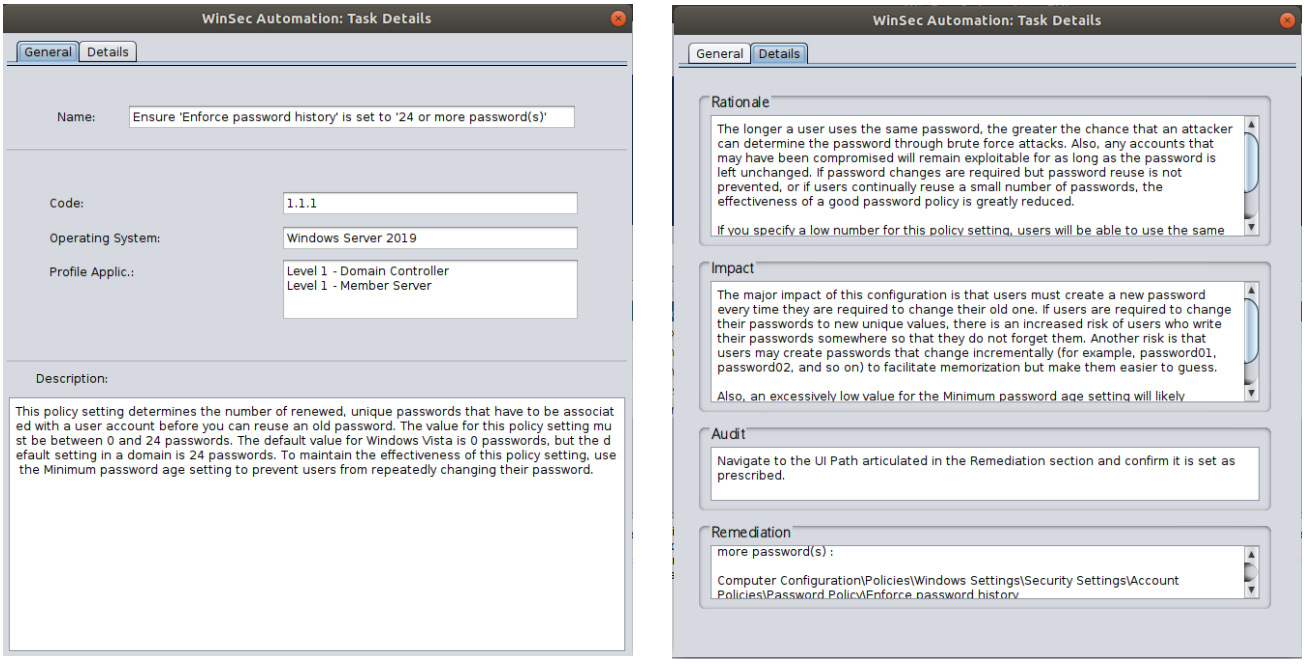


Figure 37: GUI-Details Window

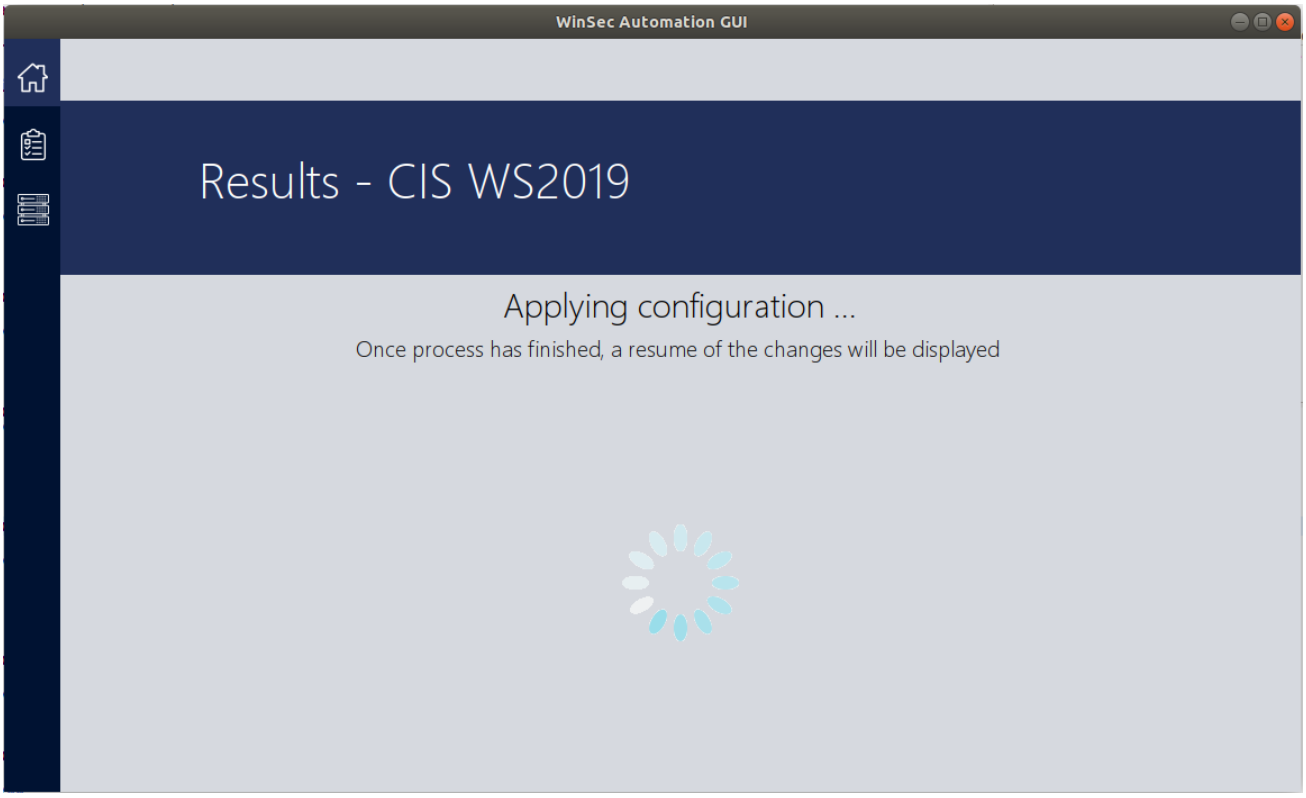


Figure 39: GUI-Applying configuration

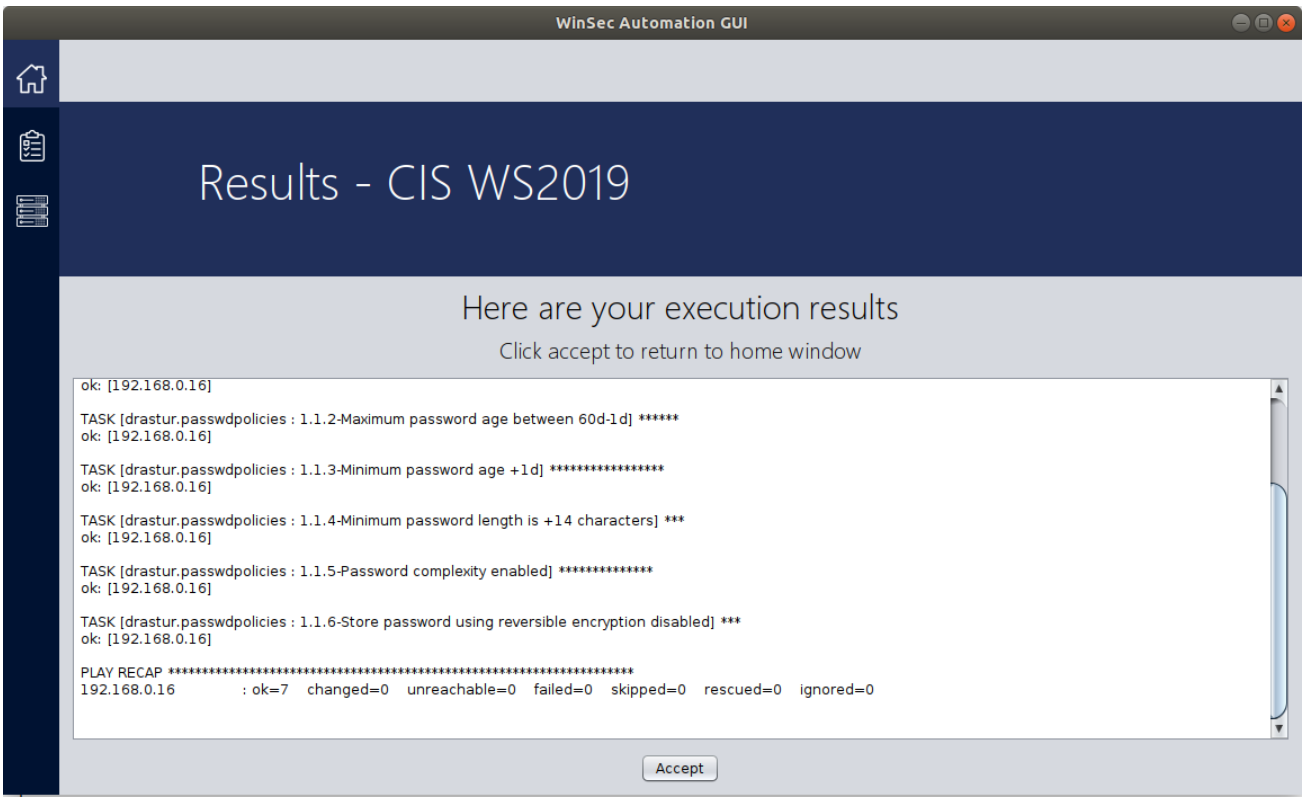


Figure 40: GUI-Results window

7.4 ISD 10: TECHNICAL SPECIFICATION OF THE TEST PLAN

7.4.1 Integration and System Testing

The system will be tested using two Virtual Machines (virtualized using VirtualBox 6.1.22):

- Windows Server 2019: It will act as a host where the configuration will be changed. Bridged connection.
- Ubuntu LTS 18.04.1: It will be the machine in which the application will be executed. NAT connection.

Use Case 1: Default Ansible Execution	
Test	**Procedure**
Default configuration. Host misconfigured. Default inventory.	Open the app and execute the default configuration. All registers in the host machine are misconfigured (Default inventory contains valid credentials for Windows Server VM).
	Expected result
	Ansible will apply all the available checks to the machine Windows Server machine. All values will be updated.
Test	**Procedure**
Default configuration. Host misconfigured. Custom inventory.	Open the app, add a new reachable host (tag: “win”, Windows VM credentials), and execute the default configuration. All registers in the host machine are misconfigured.
	Expected result
	Ansible will apply all the available checks to the machine Windows Server machine. All values will be updated.
Test	**Procedure**
Default configuration. Unreachable host.	Open the app, add a non-reachable host. Apply the default configuration.
	Expected result
	Ansible will mark the new added host as unreachable. No checks will be executed.
Test	**Procedure**
Default configuration. Host configured. Custom inventory.	Open the app, add the Windows host. Some registers must be correctly set. Apply the default configuration.
	Expected result
	Ansible will apply all the available checks to the machine previously specified. All checks should appear with state “ok”.

Test	Procedure
Custom configuration. Host misconfigured. Custom inventory.	Open the app, add the Windows host, who must have all registers misconfigured. Apply a custom configuration where only the first three checks are enabled (1.1.1-1.1.3).
	Expected result
	Ansible will change the registers corresponding to 1.1.1 to 1.1.3. Checks from 1.1.4 to 1.1.6 will appear as skipped.
Test	**Procedure**
Custom configuration. Host misconfigured. Custom inventory. Double execution.	Open the app, add the Windows host, who must have all registers misconfigured. Apply a custom configuration where only the first three checks are enabled (1.1.1-1.1.3). Once finished, apply the same configuration again.
	Expected result
	Ansible will change the registers corresponding to 1.1.1 to 1.1.3 in the first execution. For the second execution, they will appear as "ok". Checks from 1.1.4 to 1.1.6 will appear as skipped in both executions.

Table 9: Use Case 1 Test Plan Technical Specification

Use Case 2: Creation of a custom configuration	
Test	**Procedure**
List all checks	Open the app and navigate to the task's menu. **Expected result** All controls (1.1.1 to 1.1.6) must appear in a table, being all of them enabled. The description box should contain the description of the first check.
Test	**Procedure**
Toggle checks	Open the app, navigate to the task's menu, introduce regexp "1.1.1|1.1.2|1.1.3" in the search bar and press enter. Check all controls that appear (1.1.1 to 1.1.3) and press Toggle. Clean search bar data and press enter. **Expected result** All controls (1.1.1 to 1.1.6) must appear in a table, 1.1.1 to 1.1.3 DISABLED. The description box should contain the description of the first check.
Test	**Procedure**
Show details	Open the app, navigate to the task's menu, select control 1.1.2 and click Details. **Expected result** The Details Window must appear containing the information for the check 1.1.2.
Test	**Procedure**
Search checks (unique)	Open the app, navigate to the task's menu, search introducing "1.1.3" and pressing enter. **Expected result** Only 1.1.3 must appear in the table. The description box should contain the description of 1.1.3 check.
Test	**Procedure**
Search checks (multiple)	Open the app, navigate to the task's menu, search introducing "2$|1.1.1" (regular expression) and pressing enter. **Expected result** Checks 1.1.1 and 1.1.3 must appear in the table. The description box should contain the description of 1.1.1 check.

Table 10: Use Case 2 Test Plan Technical Specification

Use Case 3: Specify a host	
Test	**Procedure**
Add valid host (single IP)	Open the app and navigate to the host menu. Introduce “test” for tag, IP “1.1.1.1”, user “test” and password “test” and press “Add”.
	Expected result
	A message indicating the host has been added pops out of the window. The user is redirected to the home window after closing the dialog.
Test	**Procedure**
Add invalid host (tag)	Open the app and navigate to the host menu. Introduce a string containing numbers for the tag field and press “Add”.
	Expected result
	A message indicating the tag name is invalid pops out of the window. The user is not redirected after closing the dialog.
Test	**Procedure**
Add invalid host (IP)	Open the app and navigate to the host menu. Introduce “test” for tag, IP “1.1.1.257”, user “test” and password “test” and press “Add”.
	Expected result
	A message indicating the IP value is invalid pops out of the window. The user is not redirected after closing the dialog.
Test	**Procedure**
Add invalid host (empty fields)	Open the app and navigate to the host menu. Press “Add”.
	Expected result
	A message indicating the tag name is invalid pops out of the window. The user is not redirected after closing the dialog.
Test	**Procedure**
Add valid host (multiple IPs)	Open the app and navigate to the host menu. Introduce “test” for tag, IP “1.1.1.1” and “1.1.1.2”, user “test” and password “test” and press “Add”.
	Expected result
	A message indicating the host has been added pops out of the window. The user is redirected to the home window after closing the dialog.

Table 11: Use Case 3 Test Plan Technical Specification

Chapter 8. INFORMATION SYSTEM CONSTRUCTION

DEVELOPMENT PHASE

ISC

8.1 ISC 1: GENERATION AND CONSTRUCTION ENVIRONMENT PREPARATION

8.1.1 Standards and Rules followed

There are two main standards that have been actively consulted during the development of the project. A brief resume regarding both will be now exposed:

8.1.1.1 SCAP

The Security Content Automation Protocol (SCAP) is a standard formed by a set of specifications created by the National Institute of Standards and Technology (NIST) that aims to unify the format and terminology used to manipulate security configurations and fails. [7]

Specially during the investigation phase that took place at the start of the project, the XCCDF specification, part of the SCAP standard, was intensively reviewed, mainly caused by the lack of any other documentation regarding this topic. The specification for the language can be downloaded from the NIST webpage.

8.1.1.2 CIS Benchmarks

These files, created by the Center of Internet Security, are the only best-practice security configuration guides both developed and accepted by government, business, industry, and academia.

Each CIS Benchmarks is specifically designed for a determinate operating system. The list of operating systems supported can be checked here. They can be downloaded for free in PDF format after proper application.

8.1.2 Programming Languages

8.1.2.1 Java

Java is the main programming language used for the development of this project. There are several reasons for this choice:

- It is an object-oriented language that allows the developer to create reusable code that is both easy to understand and maintain.
- Multithreading capability. Java allows the programmer to perform multiple tasks at the same time. This is almost mandatory when creating responsive graphical user interfaces.

- Its wide use and maturity. Being one of the most used programming languages means there is a large community behind that provides well needed support. In addition to this, the great number of tutorials and documentation available make it an excellent choice for this project.

Figure 41: Java

8.1.2.2 YAML

YAML, acronym for *YAML Ain't Markup Language* is the specification used by Ansible to create any configuration.

It is a data-serialization language with a simple format, which makes it easy to read and understand. It is based on key value pairs separated by a colon which can be used to create structures such as lists, define attributes, etc. [10]

Figure 42: YAML

8.1.2.3 JSON

JSON, as YAML, is a data-serialization language used widely to interchange data. Its format gives it two advantages: it is easy to read by humans, but also for machines to parse. Originally, it was created based on the standard ECMAScript for the JavaScript programming language. [10]

Inside the developed system, it is used to store all the information about the CIS Benchmark.

Figure 43: JSON

8.1.2.4 INI

INI is the default format for Ansible Inventory definition. It consists of a text-based file, containing key value pairs defining different configuration properties, grouped over sections defined between square brackets. [10]

This format has been used since MS-DOS operating system, and is still widely spread for configuration purposes.

8.1.3 Tools and Programs Used for Development

8.1.3.1 Eclipse

Eclipse is an integrated development environment (IDE) used to create Java applications. The Eclipse SDK, which can be downloaded from the official Eclipse Foundation webpage, is a free open-source software containing the previously mentioned IDE.

Eclipse offers a simple interface that allows the developer to navigate projects and files, execute applications and debug code. One of its most useful features is its marketplace.

This site allows you to download plugins from an immense catalog that add all kinds of new features, from code generation to code highlighting among many others.

Specifically, two plugins have been added to the base IDE to develop this project:

- **WindowBuilder**: This plugin generates the code for the Swing components that composing the GUI. It renders a graphical view of the window, allowing the programmer to add new components to it using its mouse. Once added, properties for the component can also be modified.
- **ANSI Escape in Console:** This plugin allows the Eclipse console to read ANSII codes, allowing the developer to create colorized output.

8.1.3.2 Apache Maven

Apache Maven, or just Maven, is a software designed to manage and build projects created by Jason van Zyl. It is also able to manage project dependencies, adding numerous features to classical libraries. [13]

All Maven configuration is confined in one XML file, the POM (Project Object Model).

8.1.3.3 Git

Git is one of the most used version control systems. It is a free open-source software that allows its users to keep track of every change made in one or more files efficiently.

Thanks to its integration with Eclipse via the preinstalled plugin Eclipse EGit, projects can be synchronized and stored in repositories directly from the IDE.

To store the projects online, GitHub is also used. It is a platform aimed to host mainly projects and files by means of git.

8.1.3.4 Diagrams.net

Diagrams.net, previously known as Draw.io, is a tool for drawing and designing all different kinds of diagrams and visuals. [14]

It allows a very flexible configuration for any component added to the canvas, with multiple properties that can be completely configured. It also includes many different export formats, from PNG to XML.

There exists both a Desktop and online version of the tool. Both can be used for free.

8.2 ISC 2: COMPONENT AND PROCEDURES CODE GENERATION

The only code (JavaDoc) commented Will be the one included in the Interface WinSecServices, as it acts as the controller and only way of communication between the view and the model:

Package com.winsec.service

Interface WinSecService

```
public interface WinSecService
```

loadChecklist

```
void loadChecklist()
```

Loads the checklist into objects

applyToString

```
java.lang.String applyToString()
```

Based on the controls checked, it performs the supervision and remediation of any bad configuration

Returns:

String The output of the Ansible execution

apply

```
void apply()
```

Based on the controls checked, it performs the supervision and remediation of any bad configuration. Prints the info in the predefined console.

toggleCheck

```
java.util.List<com.winsec.model.Control> toggleCheck(java.lang.String id)
```

Changes the state of a check.

Parameters:

`id` - Code identifier for the check to be switched.

Returns:

Returns the controls that have been switched.

findControls

```
java.util.List<com.winsec.model.Control> findControls(java.lang.String id)
```

Returns a list of controls matching a given pattern.

Parameters:

`id` - Code identifier for the check.

Returns:

A list of controls matching a given pattern.

importModules

```
void importModules()
```

Imports/updates all needed modules from Ansible Galaxy

getChecklist

```
java.util.List<com.winsec.model.Control> getChecklist()
```

Retrieves the full list of checks defined in the checklist.

Returns:
The list of checks

findControl

```
java.util.Optional<com.winsec.model.Control> findControl(java.lang.String id)
```

Searches for the first check who's code matches the given one.

Parameters:
`id` - Identifier for the control

Returns:
The first control matching the id provided, encapsulated in an Optional, or an empty optional if not found.

createHost

```
void createHost(java.lang.String tag,
                java.util.List<java.lang.String> ips,
                java.lang.String user,
                java.lang.String password)
```

Creates a host for Ansible execution with the given data.

Parameters:
`tag` - Identifier for the group of machines

`ips` - IPv4 address of each of the machines

`user` - User name of an existing account in the machines

`password` - Password for the user name account specified.

getVersion

```
java.lang.String getVersion()
```

Retrieves the checklist version for the benchmark

Returns:
The checklist version

getSystem

```
java.lang.String getSystem()
```

Retrieves the OS for the benchmark

Returns:
The OS version

8.3 ISC 4: INTEGRATION TESTS EXECUTION

Use Case 1: Default Ansible Execution	
Test	**Expected result**
Default configuration. Host misconfigured. Default inventory.	Ansible will apply all the available checks to the machine Windows Server machine. All values will be updated.
	Obtained result
	As expected in the test specification
Test	**Expected result**
Default configuration. Host misconfigured. Custom inventory.	Ansible will apply all the available checks to the machine Windows Server machine. All values will be updated.
	Obtained result
	As expected in the test specification
Test	**Expected result**
Default configuration. Unreachable host.	Ansible will mark the new added host as unreachable. No checks will be executed.
	Obtained result
	As expected in the test specification
Test	**Expected result**
Default configuration. Host configured. Custom inventory.	Ansible will apply all the available checks to the machine previously specified. All checks should appear with state “ok”.
	Obtained result
	As expected in the test specification
Test	**Expected result**
Custom configuration. Host misconfigured. Custom inventory.	Ansible will change the registers corresponding to 1.1.1 to 1.1.3. Checks from 1.1.4 to 1.1.6 will appear as skipped.
	Obtained result
	As expected in the test specification
Test	**Expected result**
Custom configuration. Host misconfigured. Custom inventory. Double execution.	Ansible will change the registers corresponding to 1.1.1 to 1.1.3 in the first execution. For the second execution, they will appear as “ok”. Checks from 1.1.4 to 1.1.6 will appear as skipped in both executions.
	Obtained result
	As expected in the test specification

Table 12: Use Case 1 Test Execution

Use Case 2: Creation of a custom configuration	
Test	**Expected result**
List all checks	All controls (1.1.1 to 1.1.6) must appear in a table, being all of them enabled. The description box should contain the description of the first check.
	Obtained result
	As expected in the test specification
Test	**Expected result**
Toggle checks	All controls (1.1.1 to 1.1.6) must appear in a table, 1.1.1 to 1.1.3 DISABLED. The description box should contain the description of the first check.
	Obtained result
	As expected in the test specification
Test	**Expected result**
Show details	The Details Window must appear containing the information for the check 1.1.2.
	Obtained result
	As expected in the test specification
Test	**Expected result**
Search checks (unique)	Only 1.1.3 must appear in the table. The description box should contain the description of 1.1.3 check.
	Obtained result
	As expected in the test specification
Test	**Expected result**
Search checks (multiple)	Checks 1.1.1 and 1.1.3 must appear in the table. The description box should contain the description of 1.1.1 check.
	Obtained result
	As expected in the test specification

Table 13: Use Case 2 Test Execution

Use Case 3: Specify a host	
Test	**Expected result**
Add valid host (single IP)	A message indicating the host has been added pops out of the window. The user is redirected to the home window after closing the dialog.
	Obtained result
	As expected in the test specification
Test	**Expected result**
Add invalid host (tag)	A message indicating the tag name is invalid pops out of the window. The user is not redirected after closing the dialog.
	Obtained result
	As expected in the test specification
Test	**Expected result**
Add invalid host (IP)	A message indicating the IP value is invalid pops out of the window. The user is not redirected after closing the dialog.
	Obtained result
	As expected in the test specification
Test	**Expected result**
Add invalid host (empty fields)	A message indicating the tag name is invalid pops out of the window. The user is not redirected after closing the dialog.
	Obtained result
	As expected in the test specification
Test	**Expected result**
Add valid host (multiple IPs)	A message indicating the host has been added pops out of the window. The user is redirected to the home window after closing the dialog.
	Obtained result
	As expected in the test specification

Table 14: Use Case 3 Test Execution

8.4 ISC 6: USER MANUALS ELABORATION

8.4.1 Installation Manual

This section is covered by Appendix: Installation Manual.

8.4.2 Execution Manual

Now that all prerequisites have been fulfilled, both applications (the GUI and CLI version) can be run from a command line by means of the following command:

```
java -jar WinSecGUI.jar

java -jar WinsSecCLI.jar
```

To stop the application, there are different alternatives depending on the interface selected:

For the GUI application, the recommended close operation will be to click on the default close button of the window.

On the CLI application, the expected close operation would be to input a 0 from the main menu, that is, select the Exit feature from the menu. However, as any command-line interface, the process can also be interrupted pressing CTRL + C.

8.4.3 User Manual

The User Manual is included in the *Annex: User Manual.*

8.4.4 Programmer Manual

This section will cover the information needed to improve and expand the functionality of the app.

Before any development is made, it is highly recommended that anyone aiming to do an enhancement to this project revises the documentation mentioned in the sections above that analyze in detail the structure and processes held in the application.

This application is divided into two projects: WinSecAutomation and WinSec_GUI.

The first one contains the logic and model of the application, as well as the command-line interface. The second focuses on the graphical elements that compose the GUI.

The application follows a layered architecture mixed with an MVC. This means that there are three well differentiated parts: the view, the controller and the model.

The view, both for the CLI and GUI, ask the controller for all the information required to display. In this case, the interface WinSecService acts a façade that contains all the possible actions that can be performed by the model. Any future additions must respect this architecture.

To finish, there is one last aspect to consider before starting to code. The data source for this app is a JSON file situated in the resources folder of the WinSecAutomation project. The structure is the same as the one included in the CIS Benchmark. It is read by the FileUtils class, which loads the information into the model when the app is started. In case it is desired to add new checks, they must be added as a new item inside the “controls” array included in the benchmark. Each new control must contain the same attributes as the ones already implemented.

In addition to this, a new Ansible role must be executed with the expected behavior for those new checks. To achieve conditional execution, tasks must contain a when clause that compares the code of the check to the flag passed to the role. If needed, the repository for the role covering section 1.1 of the CIS Benchmark can be consulted at Drastur/passwdPolicies.

Chapter 9. Conclusions and Extensions

9.1 CONCLUSIONS

Windows Security is vital for both individuals and enterprises. It is, without any doubt, the most extended operating system for servers worldwide, which makes thoroughly difficult to understand the lack of free tools to automatize its security.

The system developed successfully achieves to fill this gap by providing complete automation of both audit and remediation of numerous security controls, defined and revised by reputable enterprises.

Alongside this, the tool created aims to teach non expert users of the importance of security in any server, and particularly in Windows Servers. Although this will only be achieved once it is included in a teaching room, the satisfaction of the client, supervisor of this project, supposes a complete success of the project.

The development of the project, despite some setbacks produced by the lack of experience on this topic have set me to discover numerous features of Windows Security Policies, server administration and process automation that will certainly help me in the future.

9.2 EXTENSIONS

The main extension that can be made to this project is the addition of new security controls. As this application has been developed to demonstrate that an automation of Windows Server security is possible, creating many different controls was not a priority.

However, the system has been developed taking this extension into account to ease the addition of any new security controls to the application. There are some steps that must be followed:

Firstly, the developer wanting to create new controls must implement them using Ansible. It is highly recommended to create them following the CIS Benchmark structure, that is, implementing all controls in a section to then pack them in an Ansible role. For each control added, there will be an Ansible task responsible for checking and modifying the necessary values in the host.

The second step is to publish it via Ansible Galaxy. This can be easily done by pushing the Ansible role to a GitHub public repository and then linking it from the Ansible Galaxy site.

Once finished, the developer must create a new JSON object containing the information included in the CIS Benchmark document in the "*CISBenchmark.json*" file.

Finally, the last step will be to add the name of the new role created to the *ROLES* list defined in the class *Play*.

By creating new roles, not only the creator will benefit from the inclusion of new security controls but also anyone wanting to download the role. These new roles created by the community could be freely shared inside the GitHub repository of the project.

APPENDICES

RISK MANAGEMENT PLAN

This section will cover the necessary processes to perform the risk management of the project. The methodology applied is the one specified in the PMBOK [3].

The risk management can be divided into six different phases:

1. Risk management planning
2. Risk identification
3. Qualitative Risk Analysis
4. Quantitative Risk Analysis
5. Risk Response planification
6. Risk monitoring and control

Risk management planning

Picking a suitable methodology is key to perform a successful risk management. In this case, as mentioned before, the PMBOK methodology will be the one followed.

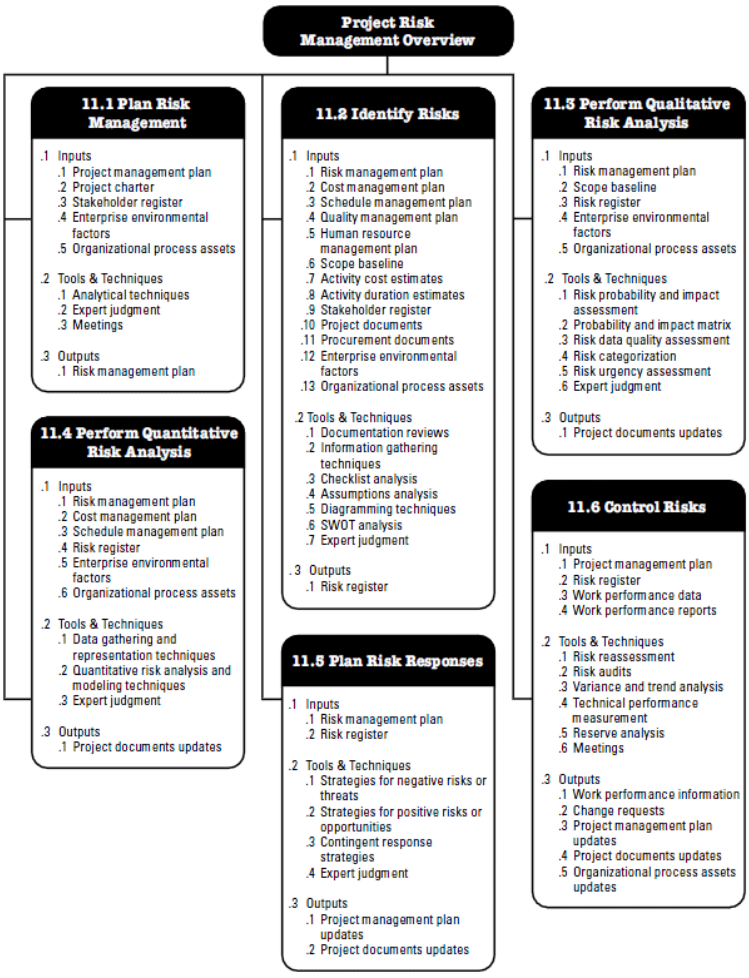


Figure 44: Risk management planning (PMBOK)

Risk Identification

To identify all registered risks, a common risk gathering technique has been used: brainstorming.

The PMBOK defines brainstorming as a general data gathering and creativity technique that can be used to identify risks, ideas, or solutions to issues by using a group of team members or subject-matter experts. Typically, a brainstorming session is structured so that each participant's ideas are recorded for later analysis.

The list of risks obtained will be classified using the following structure:

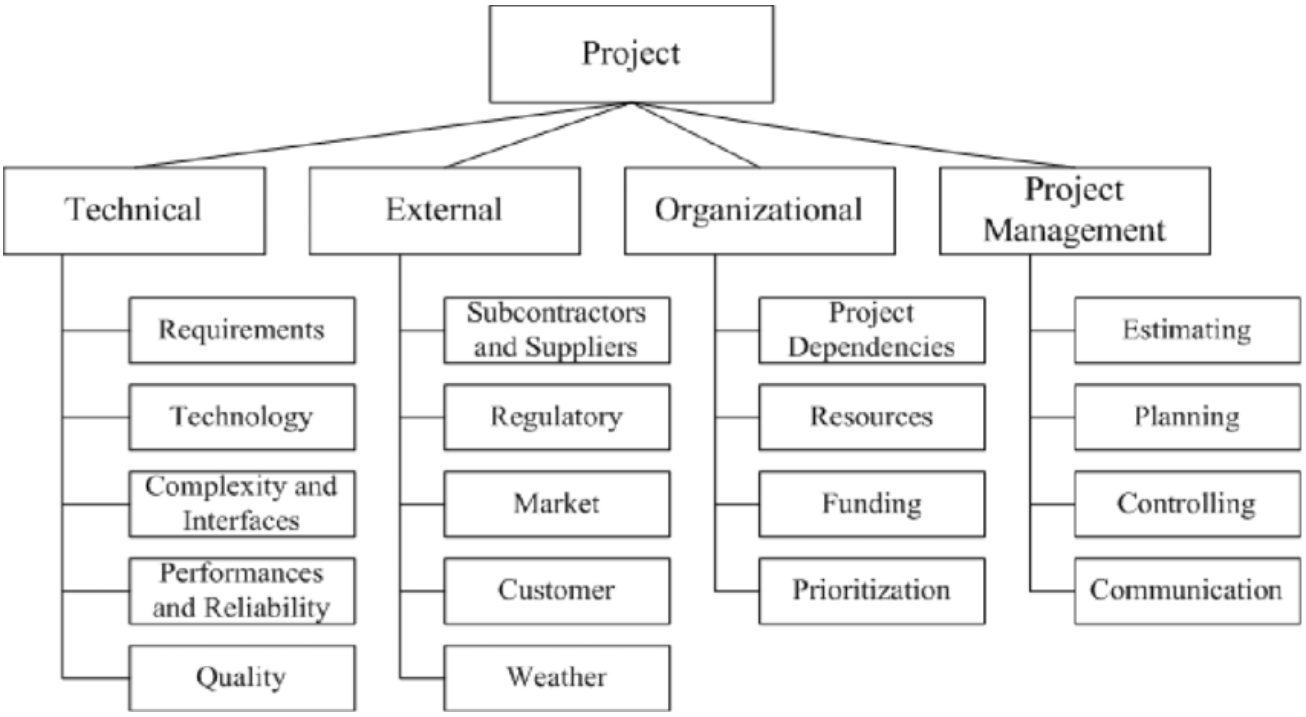


Figure 45: Project Risk Breakdown Structure (PBS)[PMBOK]

Risk Analysis

This stage determines which is the estimated probability and impact of each risk. PBMOK defines two analysis phases:

- **Qualitative analysis:** This phase consists of assessing the probability of occurrence of the risks as well as the estimated impact on the project. Thanks to this phase, each risk obtains a rating for its likelihood (Very high, high, medium, low, or very low). Impact is then measured considering budget, project schedule, scope, and product quality (Critical, High, Medium, Low and Nil).
- **Quantitative analysis:** In this case, from the estimations obtained in the previous phase numeric values are assigned to help defining policies, strategies, and reserves. From these

values, and making use of the Probability and Impact Matrix, a prioritized list of risk is obtained from the one attained in the previous phase.

Probability	Threats Risk Score = Probability x Impact					Opportunities High (RED) / Med (YEL) / Low (GRN)				
0.90 Very Likely	0.05	0.09	0.18	0.38	0.72	High	High	High	Med	Low
0.70 Likely	0.04	0.07	0.14	0.28	0.56	High	High	Med	Med	Low
0.50 Possible	0.03	0.05	0.10	0.12	0.40	High	High	Med	Low	Low
0.30 Unlikely	0.02	0.03	0.06	0.12	0.24	High	Med	Med	Low	Low
0.10 Very Unlikely	0.01	0.01	0.02	0.04	0.08	Med	Low	Low	Low	Low
	0.05	0.10	0.20	0.40	0.80	Very High	High	Med.	Low	Very Low
	Example Impact Definitions – May Be Tailored to Each Project Objective Impact on an Objective (e.g. Cost, Schedule, Scope, Quality)									

Figure 46: Probability and Impact Matrix

Risk Response Planification

Once the product from the analysis phase is obtained, the strategies that will be followed must be defined.

There are different strategies to follow when facing threats:

- **Avoid**

Under this strategy, all the project team works to eliminate all possible triggers for the risk, often needing to reschedule some parts of the project. For this reason, it is only feasible to opt for this technique with risks that affect secondary activities that do not depend on the main ones.

- **Transfer**

This strategy is based on extracting the risk and transferring it to a third party. This is mostly used on very unlikely risks that have a huge impact on the project.

- **Mitigate**

This strategy aims to minimize the probability of occurrence of the risk. This can be achieved monitoring and preventing, as far as possible, the triggers that produce it. This will lead to its avoidance or, in case it occurs, it will minimize its effects on the project.

- **Accept**

Risk acceptance is a strategy whereby the project team acknowledge to assume the existence of the risk and learn to cope with it. For this reason, when this strategy is selected no concrete actions are performed.

Risk monitoring and control

Finally, this phase aims to control the response to risks during the project. Techniques such as Risk reevaluation will be applied to update the Risk management plan and keep it up to date.

Prerequisites

To be able to successfully run both applications (GUI and CLI), the following prerequisites should be met:

- Java 8 or higher installed
- Linux environment
- Ansible installed

If Ansible is not installed in the machine, execute the following commands on your machine (requires Python 2.7/Python 3.5 or higher):

```
$ sudo apt update
$ sudo apt install ansible
```

For Java, check if it is installed running:

```
java -version
```

If Java 8 is installed, the output of the command should look like this:

```
openjdk version "1.8.0_292"
OpenJDK Runtime Environment (build 1.8.0_292-8u292-b10-0ubuntu1~18.04-b10)
OpenJDK 64-Bit Server VM (build 25.292-b10, mixed mode)
```

Otherwise, be sure to install it. Information about Java 8 installation can be found here.

Execution (JAR files)

There are two .jar files included (WinSecGUI.jar and WinSecCLI.jar). Now that all prerequisites have been fulfilled, both applications (the GUI and CLI version) can be run from a command line by means of the following command:

```
$ java -jar WinSecGUI.jar
$ java -jar WinsSecCLI.jar
```

To stop the application, there are different alternatives depending on the interface selected:

For the GUI application, the recommended close operation will be to click on the default close button of the window.

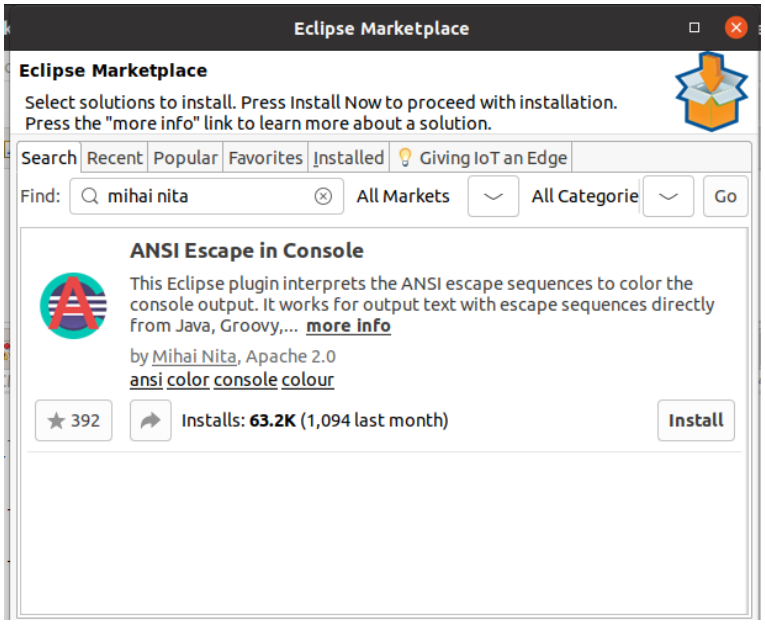
On the CLI application, the expected close operation would be to input a 0 from the main menu, that is, select the Exit feature from the menu. However, as any command-line interface, the process can also be interrupted pressing CTRL + C.

Execution (from Eclipse)

The first step is to import both projects (winsecautomation and winsec_GUI) to Eclipse.

To run the CLI version, click on the project and then Run As>Java Application. In the Select Java Application window, select **CMDAPP-com.winsec.ui** and press OK**.**

The Command-line Interface uses ANSI Escape codes to colorize the output, so installing the Eclipse extension *ANSI Escape in Console* is strongly recommended.



To run the GUI version, click on the project and then Run As>Java Application. In the Select Java Application window, select **MainWindow-com.gui.winsec** and press OK**.**

Laboratory

To fully test the potential of the app, an Ubuntu and a Windows Server 2019 machine are available to download from the following link:

WS2019

https://unioviedo-my.sharepoint.com/:u:/g/personal/uo265241_uniovi_es/EdR3gga9gbdFs-_SmEUD11kBFh5ZNwsKMmblJPT6ATqrSg?e=i5nhJe

Ubuntu 20.04

https://unioviedo-my.sharepoint.com/:u:/g/personal/uo265241_uniovi_es/EfyNcMZ0kh9HoY6NVmXE_qQBHUGz7gi91fk4oBWTLDoaNw?e=dzTYew

Credentials

Ubuntu	Windows Server
Username -> "User"	Username -> "Administrador"
Password-> "test123…"	Password-> "test123…"

You can import these two machines using VirtualBox.

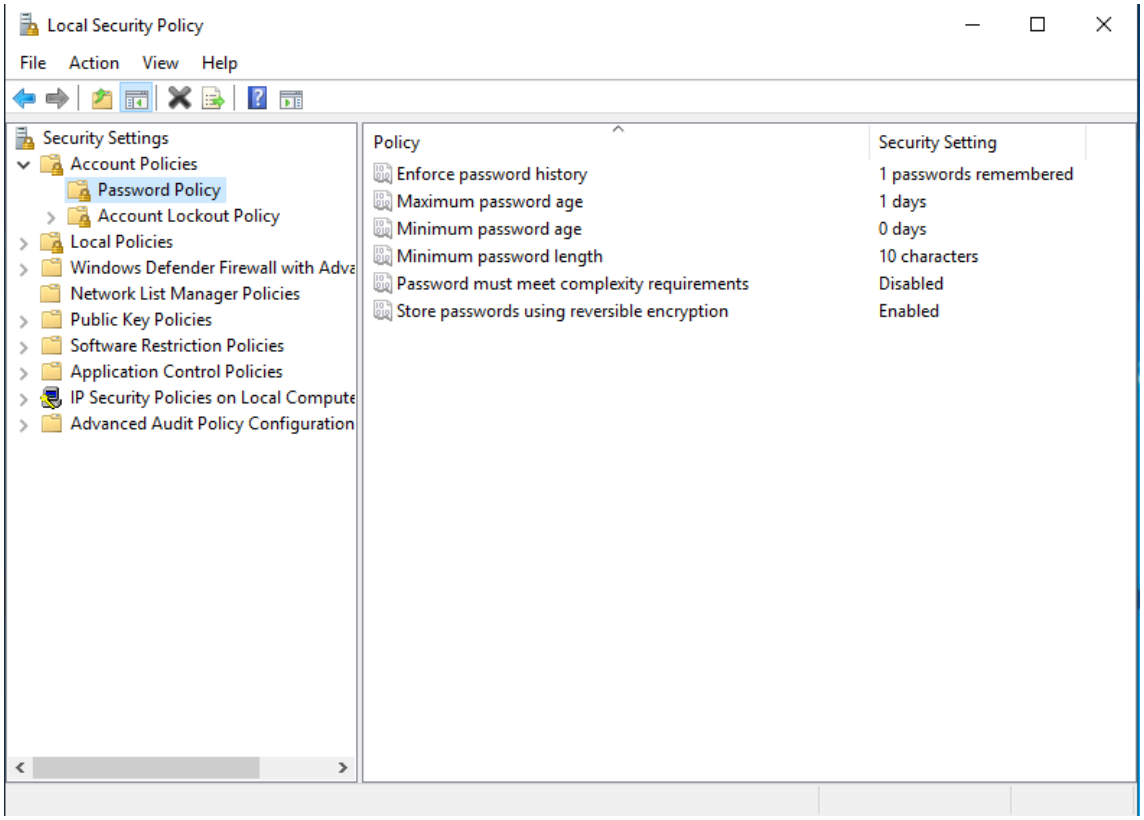
The Ubuntu machine has all prerequisites already installed and contains both the source code and JAR files of both applications.

To run the JAR files, open a terminal in the Desktop folder for *User* and executing the commands explained above.

To check the source code, open Eclipse by clicking on the link included in the Desktop.

The second machine provided, running Windows Server 2019, will be the one used as host. The only needed action for this machine is to retrieve its IP. This can be obtained running ipconfig from a PowerShell window.

This machine has its password policies misconfigured on purpose:

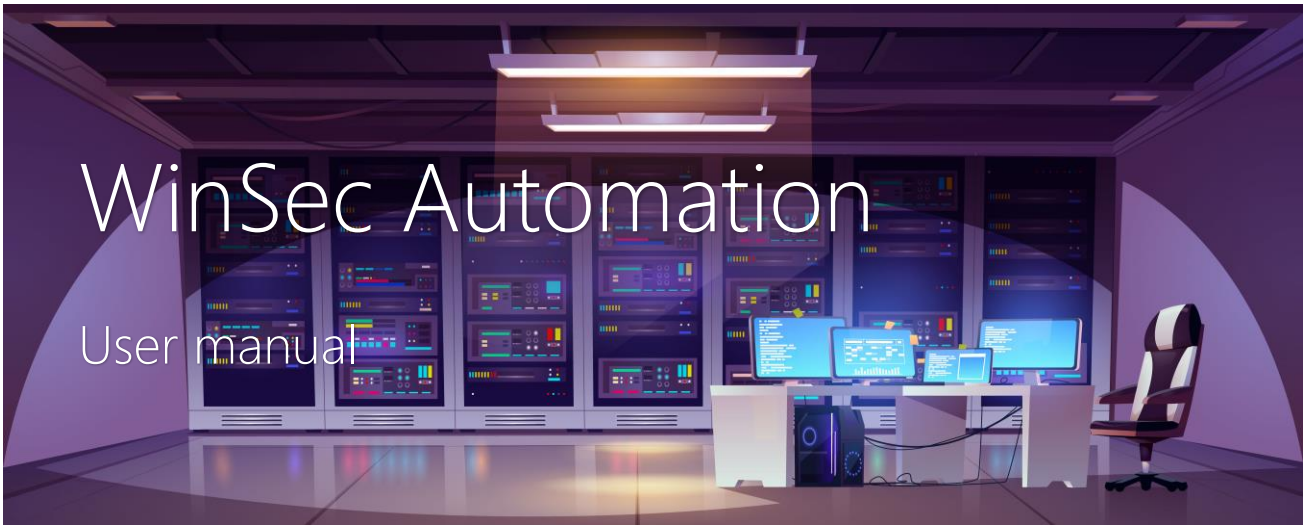


They can be checked by writing in the search bar “Security policy”>Local Security Policy>Account Policies>Password Policy.

Now, you will be able to update this configuration from the Ubuntu machine. Just add a new host, containing the IP of the Windows Server, and the credentials for the “Administrador” account.

Once execution has finished, you will be able to verify the changes by closing the Local Security Window and opening it again. The values shown for each policy should correspond to the ones recommended by CIS.

USER MANUAL



This guide aims to show all available features of the application. As the application offers the user two different interfaces, both will be explained separately.

Command Line Interface

1. *Start the application*

Before executing the tool, be sure this software installed in your machine:

- Python 2.7/3.5 (or older)
- Ansible (Community version)
- Java 8

Once all prerequisites are met, the application can be run executing in a command line:

```
java -jar WinsSecCLI.jar
```

The application will start, and the ASCII logo as well as the main menu will be printed:

The menus of the application allow the user to select the desired option by introducing the number that appears next to it and pressing ENTER key.

2. Import modules

If this is your first time executing the app, importing the necessary Ansible modules is a must. However, this can be easily done selecting option ***2. Import Modules*** from the Main menu.

This will make the application order Ansible to download them. However, if they are already installed, a warning message will appear, and no further action will be performed.

```
[➤] Selection: 2
[INFO] You have selected: Import Modules

[INFO] Importing modules...
[WARNING]: - drastur.passwdpolicies (master) is already installed - use --force to change version to unspecified
```

3. Run default configuration

If hosts are already set in your default Ansible inventory, it is possible to run all checks over all systems specified in that file. To run Ansible, select option ***3. Manage Tasks*** in the Main menu. This will display the tasks menu:

```
[INFO] You have selected: Display tasks menu

 _____________________________
|         Tasks menu          |
|_____________________________|
|  0. Exit                    |
|  1. List tasks              |
|  2. Toggle task             |
|  3. Show task details       |
|  4. Apply configuration     |
 _____________________________

[➤] Selection:
```

As it can be seen on the image, once the Tasks menu is displayed, we can select option ***4. Apply configuration.*** From this moment. Ansible will take control and start printing the output of each check that has been applied.

```
[➤] Selection: 4
[INFO] You have selected: Apply changes


PLAY [WinSecAutomation] ********************************************************************************************************************************************************

TASK [Gathering Facts] *********************************************************************************************************************************************************
ok: [192.168.0.16]

TASK [drastur.passwdpolicies : 1.1.1-Password history size is 24] ***************************************************************************************************
changed: [192.168.0.16]

TASK [drastur.passwdpolicies : 1.1.2-Maximum password age between 60d-1d] *******************************************************************************************
ok: [192.168.0.16]

TASK [drastur.passwdpolicies : 1.1.3-Minimum password age +1d] ******************************************************************************************************
ok: [192.168.0.16]

TASK [drastur.passwdpolicies : 1.1.4-Minimum password length is +14 characters] *************************************************************************************
ok: [192.168.0.16]

TASK [drastur.passwdpolicies : 1.1.5-Password complexity enabled] ***************************************************************************************************
ok: [192.168.0.16]

TASK [drastur.passwdpolicies : 1.1.6-Store password using reversible encryption disabled] ***************************************************************************
ok: [192.168.0.16]

PLAY RECAP *********************************************************************************************************************************************************
192.168.0.16               : ok=7    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

4. *Add a host*

If we want to execute a certain configuration but we do not know how to modify the Ansible default Inventory, or in case we want to specify it manually in the application, we can take advantage of the option 1. Specify Hosts in the Main menu.

Once it is selected, the software will ask the user for four different fields:

- **Tag**: This is an identifier for the set of hosts that will be later specified. It must be a word not containing numbers, spaces or special symbols.
- **IPs**: The user will be able to specify as many different IPs as it needs. These must follow IPv4 format specification. Once finished, the user must enter the character “C” and press ENTER to confirm.
- **User**: Username of an account that exist in all specified machines with administrator privileges.
- **Password**: Password credentials for the username previously introduced.

An example execution of this feature can be seen here:

```
[➤] Selection: 1
[INFO] You have selected: Specify Hosts

------ Ansible Inventory creation: Add your host ("Q" to quit) ------
[➤] Input a tag for your group of hosts: tutorial
[➤] Input the IPs of your machines ("C" to confirm): 192.168.1.1
[SUCCESS] IP added to your hosts file
[➤] Input the IPs of your machines ("C" to confirm): 12.12.12.12
[SUCCESS] IP added to your hosts file
[➤] Input the IPs of your machines ("C" to confirm): 1.1111.1.1
[ERROR] Invalid IP, please try again
[➤] Input the IPs of your machines ("C" to confirm): C
[➤] Input the name of the user: Administrator
[➤] Input the password for the user:
[SUCCESS] Ansible Inventory updated. Hosts added
```

From now on, these will be the hosts targeted by the application until it is stopped.

5. Create a custom configuration

Although applying all possible consoles could be just fine, depending on the future usage of the configured systems a customization of the base checklist might come handy.

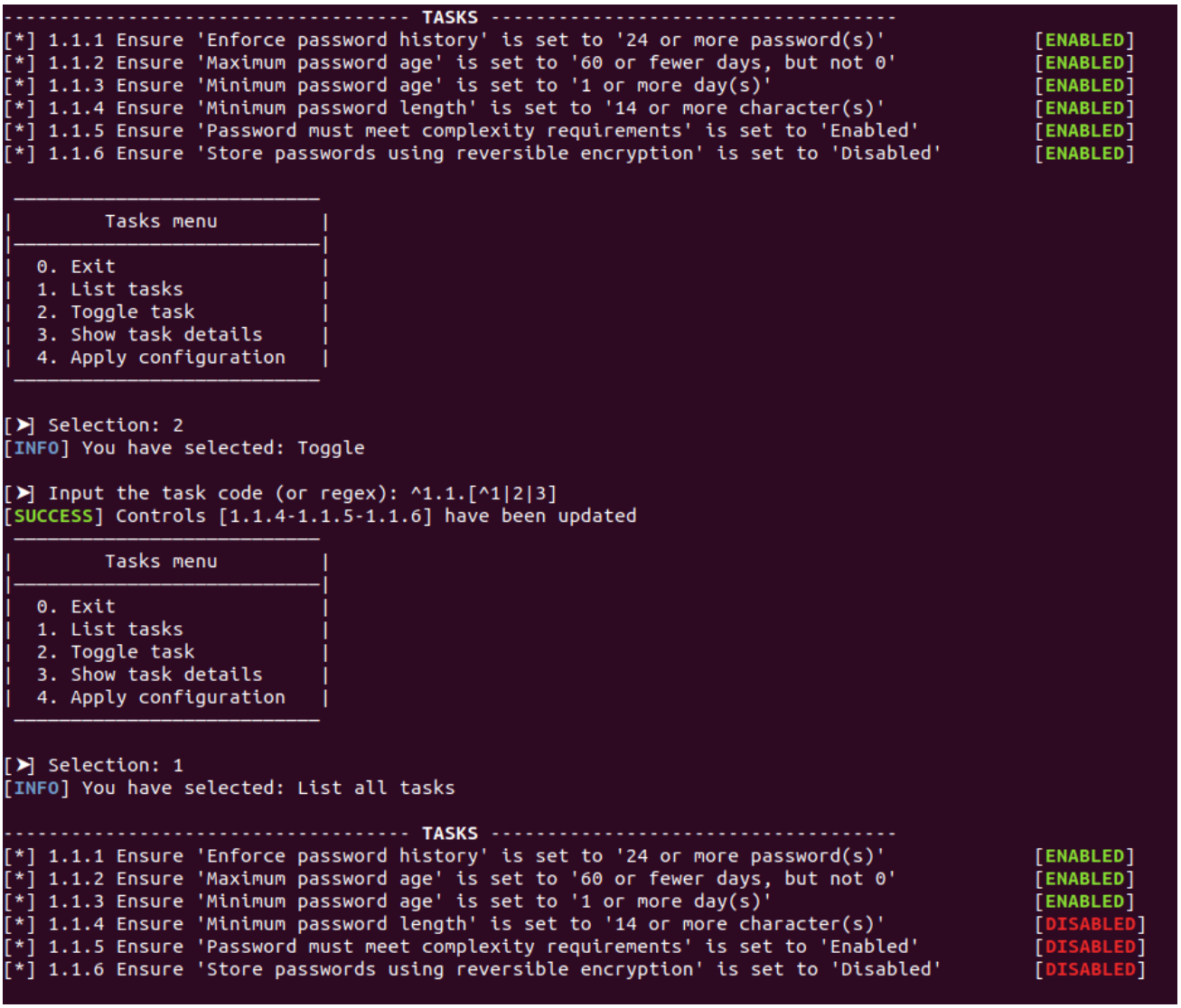
By default, every time the app is run all checks will be enabled by default. However, if the user desires to toggle off some of them he can make use of the option ***2. Toggle task*** in the Tasks menu**.** It will ask the user for the code of the task to be switched and will change its status. This means that an ENABLED task will become DISABLED and vice versa.

In addition to this, it is recommended to use the ***1. List task*** option to keep track on the status of all checks at any given time.

All disabled tasks will appear as **skipped** when Ansible is executed.

```
---------------------------------- TASKS ----------------------------------
[*] 1.1.1 Ensure 'Enforce password history' is set to '24 or more password(s)'          [ENABLED]
[*] 1.1.2 Ensure 'Maximum password age' is set to '60 or fewer days, but not 0'         [ENABLED]
[*] 1.1.3 Ensure 'Minimum password age' is set to '1 or more day(s)'                    [ENABLED]
[*] 1.1.4 Ensure 'Minimum password length' is set to '14 or more character(s)'          [ENABLED]
[*] 1.1.5 Ensure 'Password must meet complexity requirements' is set to 'Enabled'       [ENABLED]
[*] 1.1.6 Ensure 'Store passwords using reversible encryption' is set to 'Disabled'     [ENABLED]

 ____________________________
|        Tasks menu          |
|____________________________|
|  0. Exit                   |
|  1. List tasks             |
|  2. Toggle task            |
|  3. Show task details      |
|  4. Apply configuration    |
 ____________________________

[➤] Selection: 2
[INFO] You have selected: Toggle

[➤] Input the task code (or regex): 1.1.1
[SUCCESS] Control [1.1.1] has been updated
 ____________________________
|        Tasks menu          |
|____________________________|
|  0. Exit                   |
|  1. List tasks             |
|  2. Toggle task            |
|  3. Show task details      |
|  4. Apply configuration    |
 ____________________________

[➤] Selection: 1
[INFO] You have selected: List all tasks

---------------------------------- TASKS ----------------------------------
[*] 1.1.1 Ensure 'Enforce password history' is set to '24 or more password(s)'          [DISABLED]
[*] 1.1.2 Ensure 'Maximum password age' is set to '60 or fewer days, but not 0'         [ENABLED]
[*] 1.1.3 Ensure 'Minimum password age' is set to '1 or more day(s)'                    [ENABLED]
[*] 1.1.4 Ensure 'Minimum password length' is set to '14 or more character(s)'          [ENABLED]
[*] 1.1.5 Ensure 'Password must meet complexity requirements' is set to 'Enabled'       [ENABLED]
[*] 1.1.6 Ensure 'Store passwords using reversible encryption' is set to 'Disabled'     [ENABLED]
```

One very interesting feature of this function is to perform multiple toggles at once. For this, regular expressions can be created and introduced to select one or more than one task. In order to create these patterns, the following documentation can be checked. In the following image, all checks starting by 1.1. whose 3rd number is not 1, 2 or 3 will be toggled (In this case, 1.1.4, 1.1.5 and 1.1.6).



6. Display check information

If the user wants to obtain more information regarding each check, whether if guided by curiosity or aiming to evaluate if the check suits its needs, they can make use of the option ***3. Show task details*** in the Tasks menu.

This will print the different sections associated to the check by CIS.

Graphical User Interface

1. Start the application

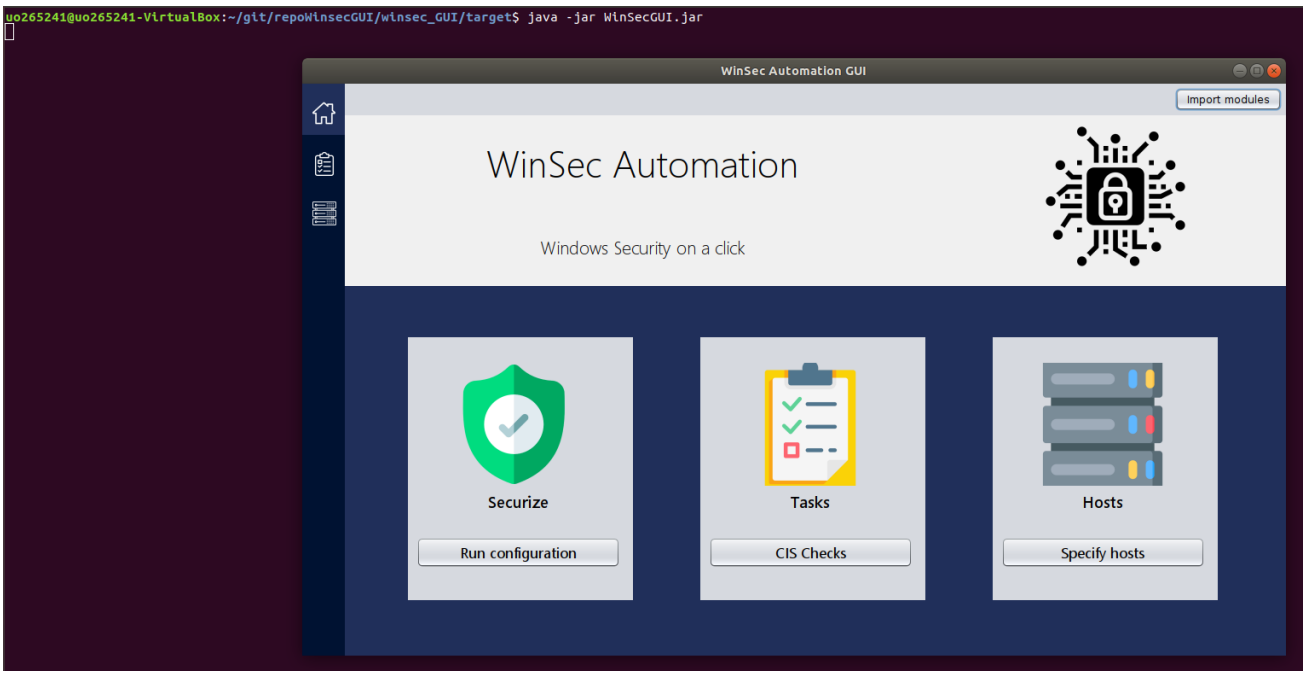
As in the case of the CLI, some prerequisites must be fulfilled to execute the application. Before executing the tool, be sure this software installed in your machine:

- ☐ Python 2.7/3.5 (or older)
- ☐ Ansible (Community version)
- ☐ Java 8

Once all prerequisites are met, the application can be run executing in a command line the following command:

```
java -jar WinsSecGUI.jar
```

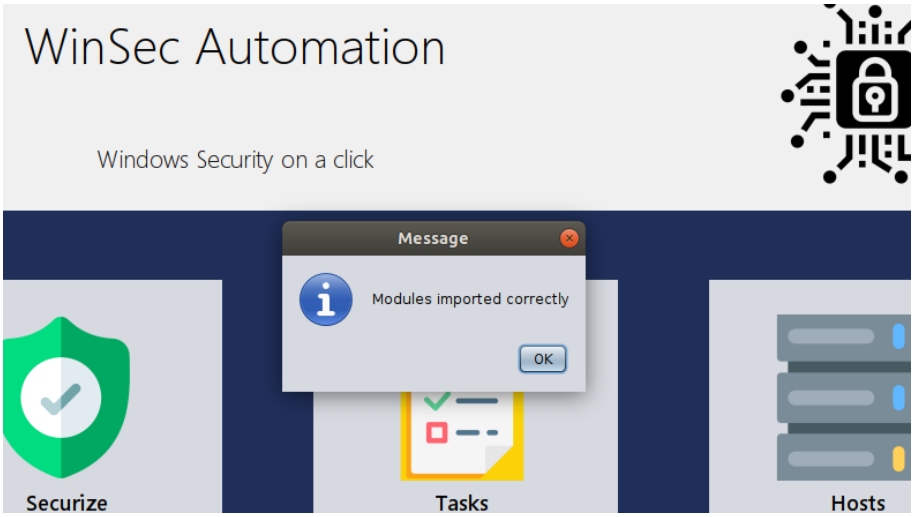
The main window will show up offering the user different options:



2. Import modules

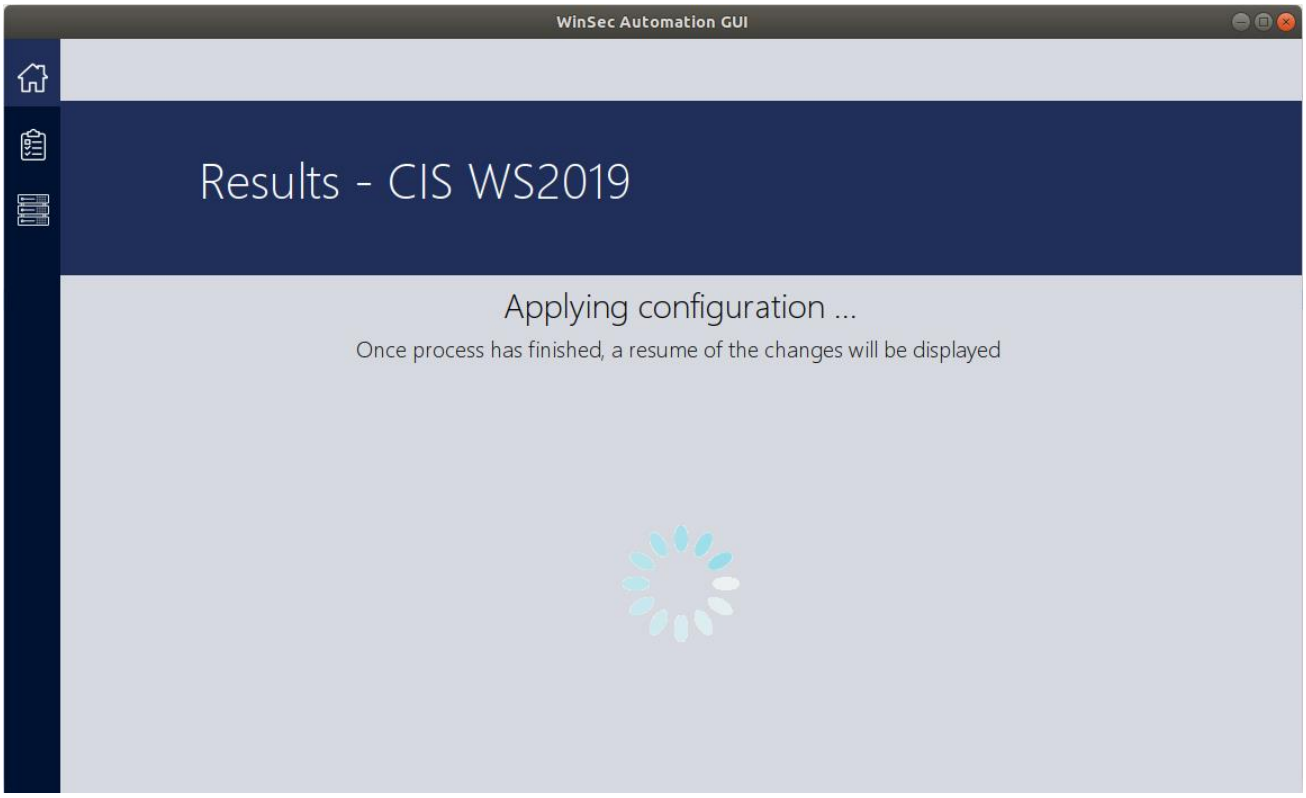
If it is the first time running the application, Ansible modules must be imported by means of the button on the top right corner.

Once the modules have been imported, a message indicating that the action was completed successfully will appear:

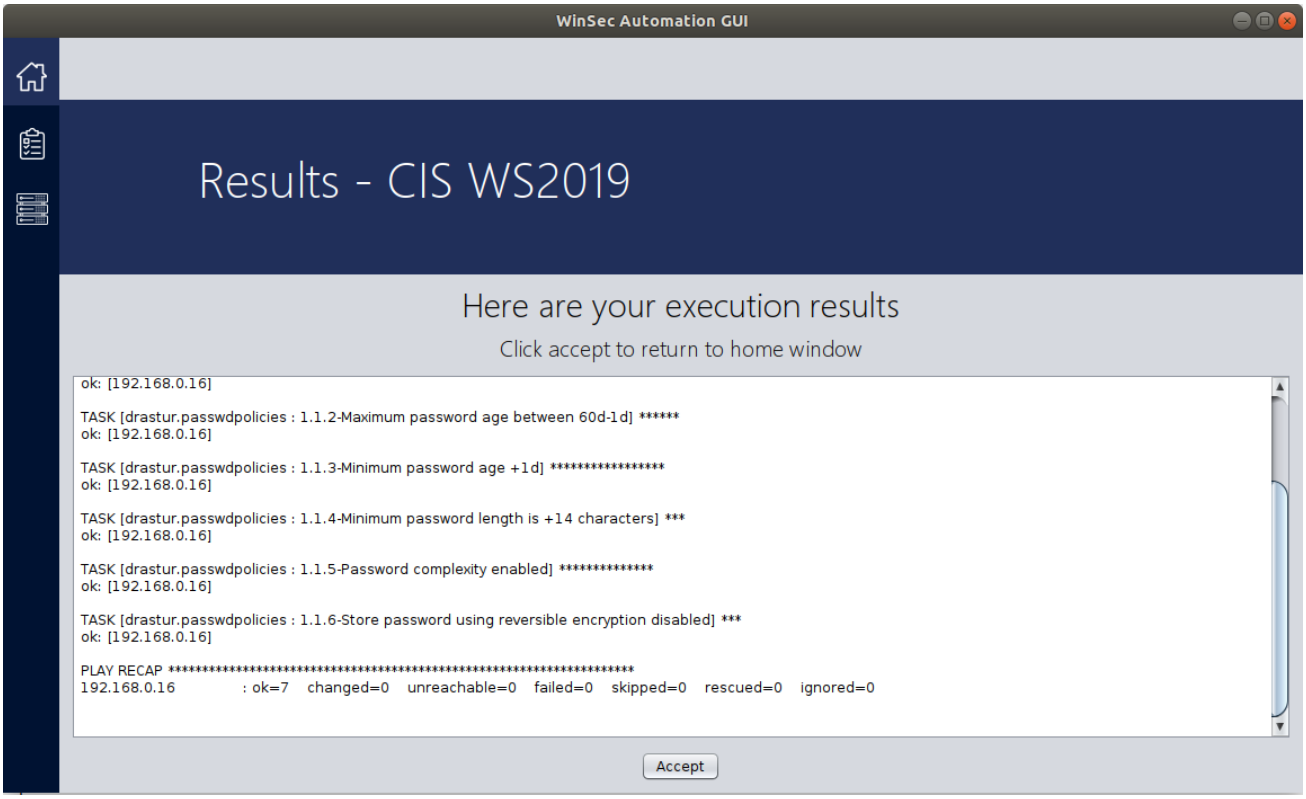


3. *Run default configuration*

To run the default configuration, that is, apply all checks to the hosts predefined in the default Ansible inventory, the user must click on the button Run configuration. This will initiate Ansible and update the content of the window. The following window will appear:

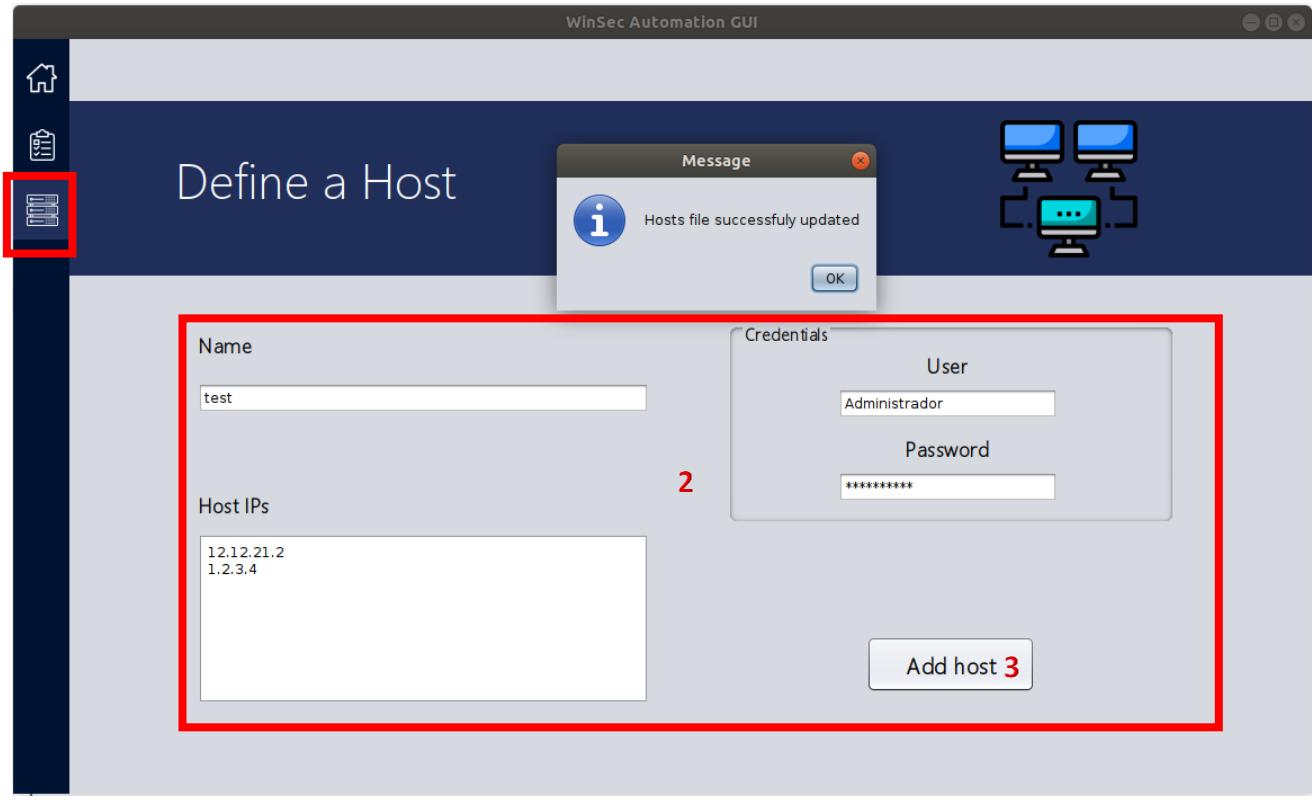


Once Ansible has completed its execution, the window will be automatically updated and the recap of the execution will be shown, as it can be seen here:



4. *Add a host*

The operations required to create a custom inventory are minimal. If you want to create one, just navigate to the Hosts menu[1], introduce the required data[2] and click on the button “Add host”[3]. If all the data introduced is valid, the application will inform you that a new inventory file was created.



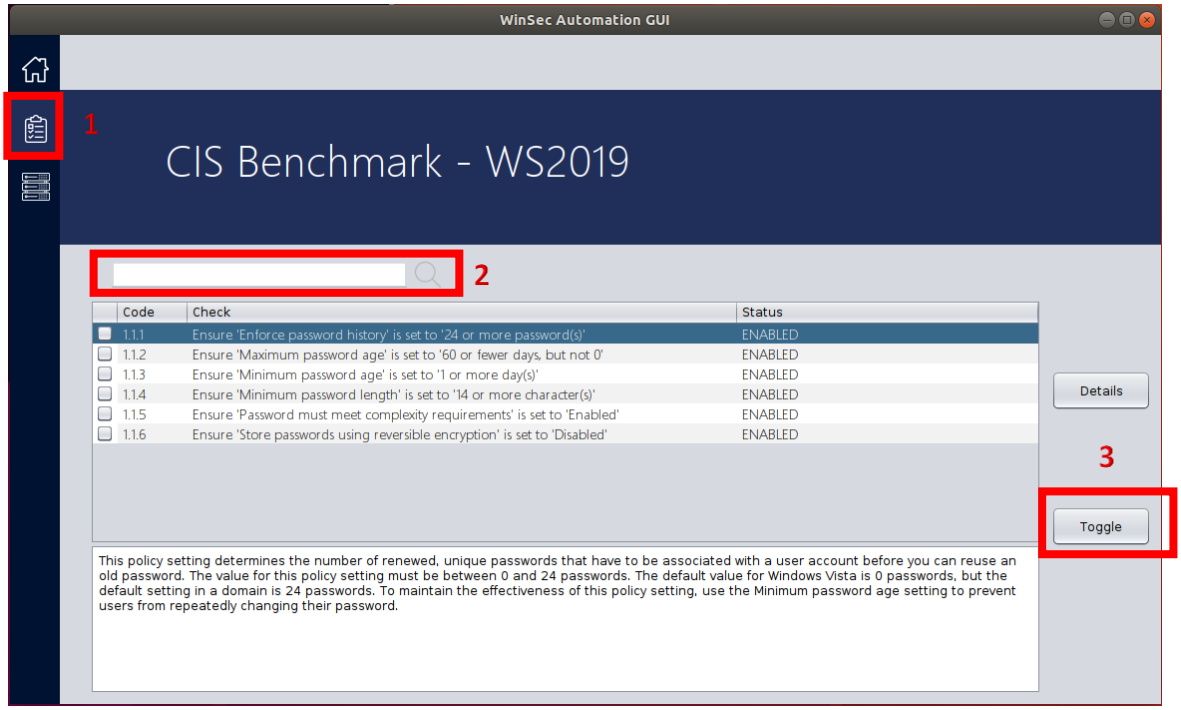
5. Create a custom configuration

Although running all checks might be a viable option for many users, some of them will require a customization of the base list that fits their needs. To do this, navigate to the tasks section [1] using the menu to the left, or the button CIS Checks in the Home window.

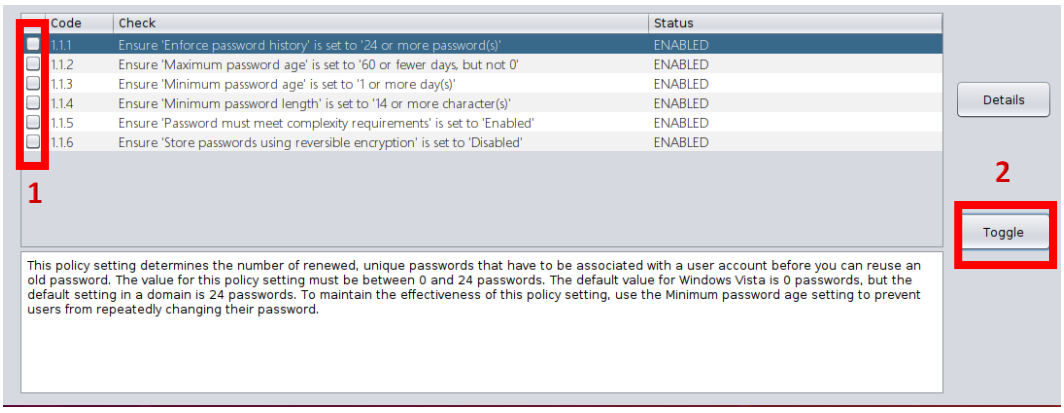
The tasks' view will display all available check in a sortable table, showing in the bottom the description of the selected task.

This view also includes a search bar [2], which filters the data according to the code value. It also allows the user to input regular expressions, therefore enabling the query of multiple checks at once.

However, if your aim is to customize the list the most important element here is the toggle button [3].



As it can be seen on the following image, each check has an associated checkbox [1] that the user can interact with. By clicking on it, the user can select one or more checks and then press Toggle [2] to switch their status between ENABLED and DISABLED:



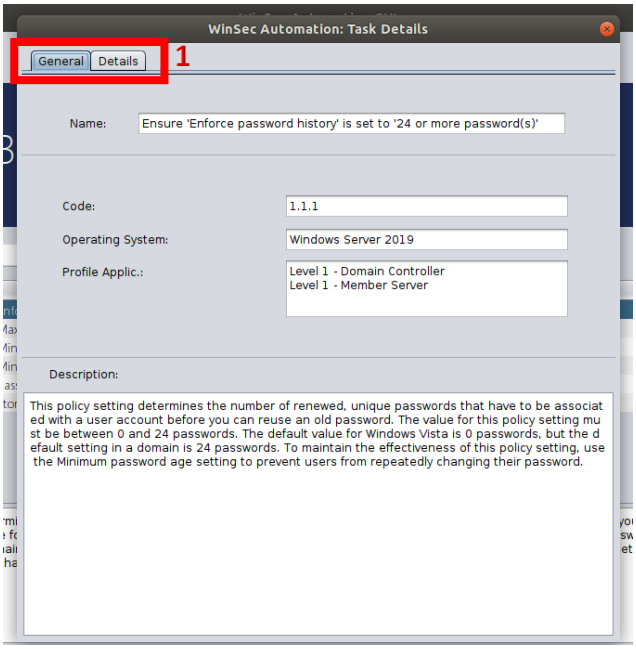
6. Display check information

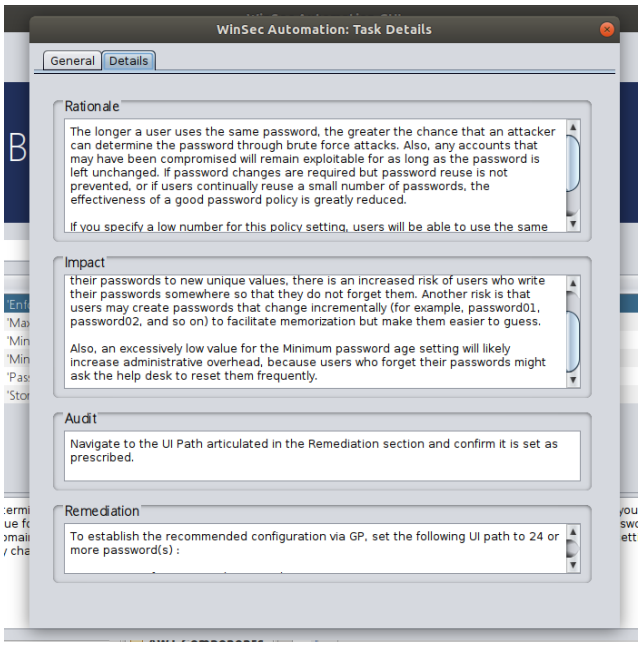
If you want to investigate a wider explanation of the check, the Details button in the tasks view might come handy.

This button will show the information available for the check selected in the tasks table. This selection can be modified by clicking the desired check or moving with the arrow keys UP and DOWN. Once the Details button is clicked, a new window will show up.

The Details window contains two tabs [1], General and Details:

- The General tab includes basic information about the check such as the name, code, operating system where it can be applied, etc.
- On the other hand, the Details tab displays the denser information such as the rationale, impact, etc.





CONTENT DELIVERED IN THE APPENDICES

Contents

Directory	Content
./	Contains a README.txt file explaining this whole structure.
./winsecautomation	Contains the whole directory structure of the project, which includes the CLI and business/model of the application.
./winsec_GUI	Contains the whole directory structure of the GUI project.
./installation	Contains a useful Installation and QuickStart guide, as well as a demo laboratory to execute the application.
./jar	Contains the executables for the two versions of the application (GUI and CLI).
./documentation	Documentation of the project (this file).
./machines	Download links for the machines of the demo laboratory.
./ansible	Contains a link to the Ansible role developed.

BIBLIOGRAPHIC REFERENCES

[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.

[2] "CIS Benchmarks," CIS, [Online]. Available: https://learn.cisecurity.org/benchmarks. [Accessed 20 3 2021].

[3] C. Security, «CIS Microsoft Windows Server 2019 Benchmark v1.2.1,» CIS Benchmarks, 2021.

[4] "XCCDF - The Extensible Configuration Checklist Description Format," CIS, [Online]. Available: https://csrc.nist.gov/projects/security-content-automation-protocol/specifications/xccdf. [Accessed 21 3 2021].

[5] "OpenSCAP Base," [Online]. Available: https://www.open-scap.org/tools/openscap-base/. [Accessed 15 02 2021].

[6] "How Ansible Works," Red Hat, [Online]. Available: https://www.ansible.com/overview/how-ansible-works. [Accessed 30 3 2021].

[7] «Ansible Documentation,» [En línea]. Available: https://docs.ansible.com/ansible/latest/index.html. [Último acceso: 12 4 2021].

[8] «Galaxy Documentation - Ansible,» [En línea]. Available: https://galaxy.ansible.com/docs/. [Último acceso: 20 04 2021].

[9] "Security Content Automation Protocol," [Online]. Available: https://csrc.nist.gov/projects/security-content-automation-protocol. [Accessed 30 3 2021].

[10] "YAML Syntax," [Online]. Available: https://docs.ansible.com/ansible/latest/reference_appendices/YAMLSyntax.html. [Accessed 21 05 2021].

[11] "JSON," [Online]. Available: https://www.json.org/json-en.html. [Accessed 7 6 2021].

[12] Techopedia, "What is an INI file?," [Online]. Available: https://www.techopedia.com/definition/24302/ini-file. [Accessed 7 6 2021].

[13] Apache, "Apache Maven Project," [Online]. Available: https://maven.apache.org/. [Accessed 12 4 2021].

[14] "Diagrams.net," [Online]. Available: https://www.diagrams.net/integrations. [Accessed 15 06 2021].

[15] Project Management Institute, Inc., A Guide to the Project Management Body of Knowledge (PMBOK® Guide) Fifth Edition, 2013.