

CYBERSECURITY AUTOMATION FOR AN INDUSTRY

4.0 GARMENT MANUFACTURING SYSTEM

H.H.D Kalhara

(IT18139440)

B.Sc. (Hons) Degree in Information Technology Specializing in Cyber
Security

Department of Computer Systems Engineering

Sri Lanka Institute of Information Technology
Sri Lanka

October 2021

CYBERSECURITY AUTOMATION FOR AN INDUSTRY

4.0 GARMENT MANUFACTURING SYSTEM

H.H.D Kalhara

(IT18139440)

Dissertation submitted in partial fulfilment of the requirement for the
Bachelor of Information Technology Specializing in Cyber Security


Department of Computer Systems Engineering

Sri Lanka Institute of Information Technology
Sri Lanka

October 2021

DECLARATION

I declare that this is my own work and this proposal does not incorporate without acknowledgement any material previously submitted for a degree or diploma in any other university or Institute of higher learning and to the best of my knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text. Also, I hereby grant to Sri Lanka Institute of Information Technology, the nonexclusive right to reproduce and distribute my dissertation, in whole or in part in print, electronic or other medium. I retain the right to use this content in whole or part in future works (such as articles or books).

Name	Student ID	Signature
H.H.D Kalhara	IT18139440	

The above candidates are carrying out research for the undergraduate

Dissertation under my supervision.

Signature of the supervisor:

.....

Prof. Pradeep Abeygunawardhana

Date:

Signature of the co-supervisor:

.....

Ms. Wellalage Sasini Nuwanthika

Date:

ABSTRACT

The digitalization of Industry 4.0 towards the Internet of Things (IoT) prioritizes productivity over security. Many studies show that IoT systems are used with insecure default configurations and publicly available hardcoded passwords, resulting in vulnerabilities such as insecure data, insecure communication and weak authentication to exist on the Operating System (OS) which leads to attacks such as industry espionage, industry sabotage, network-based attacks, and malware. A centralized automated security hardening tool, that can scan and apply security configurations in IoT Operating systems is the purposed solution for the mentioned problems. Several field visits and a risk assessment were conducted to identify security requirements and evaluate them for development of security measures utilized in the purposed Python and Ansible based security configuration management tool. Simultaneous security auditing and remediation of several devices, increased efficiency, reduced system downtime, reduced labor cost were the results from preformed tests using the tool and manual security hardening in a virtual environment consisting of Raspbian OS and Robot Operating Systems (ROS). The security measures applied using the tool can be customized based on organizational requirements and additional security measures can be added to the tool as the Industry 4.0 security requirements develops. Therefore, the security configuration management tool is a flexible, scalable, efficient, and reliable tool for automating security hardening on Industry 4.0 IoT systems and Cyber Physical Systems (CPS).

Key words – Cyber Physical Systems (CPS), Industrial Internet of Things (IIoT), Cyber security, Security hardening

ACKNOWLEDGEMENT

I'd like to express my gratitude to Prof. Pradeep Abeygunawardhana and Ms. Sasini Nuwanthika, our project supervisor and co-supervisor, for all their advice and time given to our research project, as well as for their direction and guiding through difficult topics and research gaps.

I'd like to thank our external supervisor, Chartered Eng. P.A. Gamini De Alwis, for guiding us in the right direction throughout our research project by sharing his manufacturing experience, and Dr. Darshi De Saram for sharing his knowledge and experience, as well as providing valuable feedback, which contributed to the success of our project.

I'd like to express my gratitude to Dr. Asela Kulatunga, Head of Department of Manufacturing and Industrial Engineering, University of Peradeniya, Sri Lanka, for allowing us to visit Peradeniya University to study CNC machine and robotic applications workflow, which served as a good starting point for our research. I'd like to express my gratitude to Mr. Wijeweera, owner of Wijeweera Knit Wear (Private) Limited, for allowing us to visit the garment factory during the initialization stage of our project to identify requirements.

Table of Contents

DECLARATION.....	i
ABSTRACT	ii
ACKNOWLEDGEMENT.....	iii
LIST OF FIGURES.....	v
LIST OF TABLES.....	v
LIST OF ABBREVIATIONS	vi
LIST OF APPENDICES.....	vii
1. INTRODUCTION	1
1.1 Background Review	1
1.2 Literature Review	5
1.3 Research Gap.....	10
1.4 Research Problem.....	11
1.4.1 Collaboration between different systems.....	11
1.4.2 Centralized security management.....	11
1.4.3 Secure communication	11
1.4.4 Insecure data	12
1.4.5 Initial cost.....	12
1.4.6 Absence of methodology to industry 4.0.....	12
1.5 Research Objectives.....	13
1.5.1 Main objectives.....	13
1.5.2 Specific objectives.....	13
2. METHODOLOGY	14
2.1 Methodology.....	14
2.1.1 Required CPS and IoT devices identification.....	16
2.2.2 IoT Security requirement analysis and evaluation.....	17
2.2.3 The security configuration management tool.....	19
2.2 Commercialization Aspects of the Product.....	34
2.3 Testing.....	35
3. RESULTS & DISCUSSION	38
3.1 Results.....	38
3.2 Research Findings	41
3.3 Discussion.....	42
4. CONCLUSION.....	45
REFERENCE LIST.....	46

APPENDICES	49
Appendix A: Gantt Chart	49
Appendix B: Security Rules List in Tested Security Profile	50

LIST OF FIGURES

Figure 1.1: Industrial revolution	2
Figure 1.2: Composition of Exports - 2019	5
Figure 2.1: Overall System Diagram	14
Figure 2.2: Individual Workflow Diagram	15
Figure 2.3: Heatmap of identified threats against ansible controller and IoT devices	18
Figure 2.4: Components inside security configuration management tool	19
Figure 2.5: Audit process	21
Figure 2.6: Audit role file structure	25
Figure 2.7: Remediate process	27
Figure 2.8: Remediate role file structure	29
Figure 2.9: Sample device details	30
Figure 2.10: Sample audit summary	30
Figure 2.11: Report function process	31
Figure 2.12: Sample report results table	33
Figure 2.13: Additional details about rule T_101	33
Figure 2.14: Network diagram of test scenarios	35

LIST OF TABLES

Table 1.1: Comparison of IaC configuration management platforms	9
Table 1.2: Comparison of Existing Tools	10
Table 2.1: Identified Threats Against Ansible Controller and IoT Devices	17
Table 2.2: List of Information in Security Profiles	20
Table 2.3: Sections Available in Audit Role Tasks	23

Table 2.4: Ansible Modules Used in Remediate Role Tasks	28
Table 2.5: List of Information in rules.yml	32
Table 2.6: Summary of Performed Tests	37
Table 3.1: The Results from Performed Audits	38
Table 3.2: The Results from Performed Remediations	38

LIST OF ABBREVIATIONS

Abbreviation	Description
CAD	Computer-Aided Design
CAM	Computer-Aided Manufacturing
CIA	Confidentiality, Integrity and Availability
CNC	Computer Numerical Control
CPPS	Cyber Physical Product Systems
CPS	Cyber Physical Systems
DDoS	Distributed Denial of Service
DoS	Denial of Service
DSL	Domain Specific Language
IaC	Infrastructure as Code
IDS	Intrusion Detection System
IIoT	Industrial Internet of Things
IoT	Internet of Things
IP	Internet Protocol
ISO	International Organization for Standardization
NC	Numerical Controller
OCTAVE	Operationally Critical threat Asset and Vulnerability Evaluation
OS	Operating System
OWASP	Open Web Application Security Project
PAM	Pluggable Authentication Modules
RFID	Radio-Frequency Identification

ROS	Robot Operating System
SCADA	Supervisory Control and Data Acquisition
SSH	Secure Shell
TCP	Transmission Control Protocol
YAML	YAML Ain't Markup Language
ZWPC	Zero-Waste Pattern Cutting

LIST OF APPENDICES

Appendix A: Gantt Chart	49
Appendix B: Security Rules List in Tested Security Profile	50

1. INTRODUCTION

1.1 Background Review

The invention of steam power ushered in the first industrial revolution, the greatest leap forward in productivity. Fabric was created using inventions such as the spinning machine and looms. The textile business began with the invention of the first mechanical sewing machine. The first industrial revolution was fueled by coal, water, and steam, whereas the second was fueled by electricity, gas, and oil. The sewing machine began to be created rapidly to meet high demands as a result of the revolution in the apparel industry. Partial automation with Programmable Management Systems kicked off the third industrial revolution, also known as the digital revolution. The digital revolution was a success due to advancement in microchips, programming, fiber optic links, and telecommunication facilities. At the Hannover Fair in 2011, the German Federal Government was the first to announce the fourth industrial revolution. The physical world is created in a virtual environment, and cyber physical systems (CPS) are linked and communicate with each other and with humans in real time to make decisions without human intervention, with the goal of developing new internet services and business models that provide efficiency, visibility, troubleshooting, flexibility, supervising, and, productivity while lowering costs. Value creation and business models will emerge from industrial processes that have integrated CPS and Internet of Things (IoT), as well as modular frameworks that can adapt to constantly changing requirements. Industry 4.0 garment manufacturing systems rely on IoT and other technologies like CPS, wireless sensor networks, Computer Vision, Machine learning, Data analytics, cyber security, cloud computing, systems engineering, augmented reality, and 3D printing to connect the digital and physical environments.

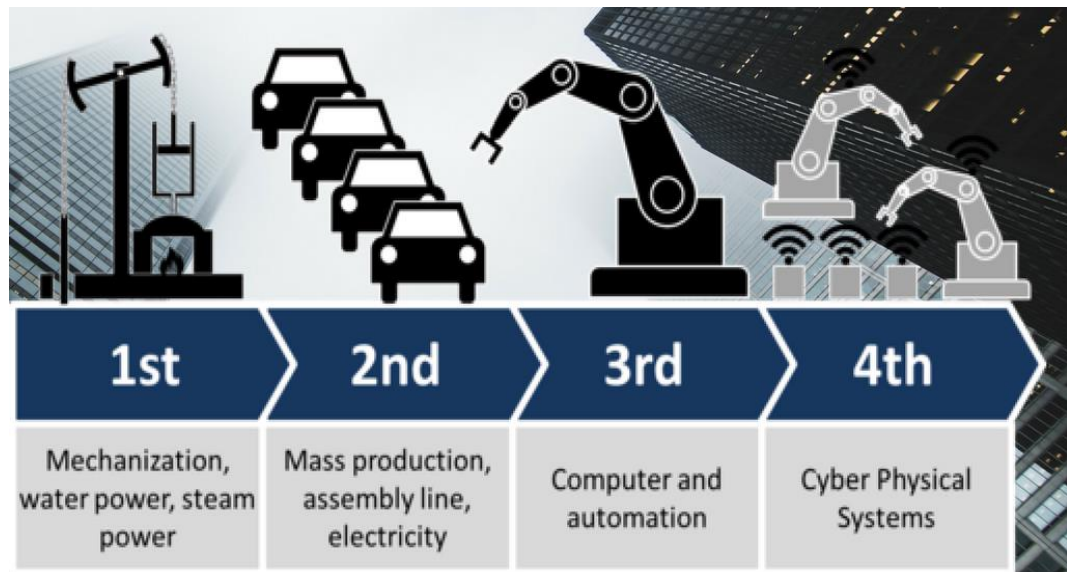


Figure 1.1: Industrial revolution

Industry 4.0's objective is to develop I-IoT for the integration of network technologies and smart computing into production for automation. IoT refers to the interconnection of computers and computer-related devices in order to improve intelligence, productivity, and safety. While the IoT refers to a network of interconnected computing devices that communicate with one another and with users in real time, it is most usually utilized for consumer purposes, I-IoT is used for industrial purposes such as manufacturing. Unlike IoT, I-IoT includes a wider range of network technologies, command and control, service requirements, and smart devices.

Since the beginning of the first revolution, the garment industry has grown to be an important part of the global manufacturing industries. Textiles have a long history, and they continue to evolve because to their high flexibility to new technologies such as Industrial Internet of Things (IIoT). The apparel fashion market has been growing extremely competitive. As a result, industry-wide integrated technologies are fast evolving, enabling for new manufacturing processes. Industry 4.0 enables features such as flexibility, scalability, customer satisfaction, management, and accessibility, all of which are important in the garment industry. Nonetheless, the majority of automation systems are centered on the apparel industry.

In a garment and apparel factory, the fundamental flow of production processes comprises designing the product according to marketing demands and consumer needs and trends, selecting acceptable clothing material, and assembling the product. Creating layers using clothing materials, cutting shapes with minimal material waste, different sewing operations, finishing, product inspection, packaging, storing, and distribution.

Cutting is a key process in garment manufacturing process that has direct impact to the profit because of the material wastage issue, availability of materials, cost of materials, and labor dependency of the process. Die cutters, first introduced in the 1900s, improved cutting efficiency and quality. Numerical Controller (NC) machines, introduced in the 1940s, enabled continuous cutting, allowing to boost production, increase flexibility and enhance material utilization. The digital revolution given rise to Computer Numerically Controlled (CNC) machinery. Cutting has become the most technologically advanced area in the apparel industry as a result of this improvement. There are a variety of cutting technologies available, including lasers, plasma, ultrasound, computer-controlled knives, and markers. Since the development of the first fully automated cutting system, existing cutting technologies have improved in terms of productivity, adaptability, and pattern matching capabilities. Cutting devices, such as CNC machines, are part of the industry 4.0 revolution, which bring forth solutions to address problems such as labor-intensive problems, waste, and increased cost.

In the fourth industrial revolution, the concept of digitalization and integration has been highlighted, and CNC plays a critical role in automating cutting process in garment manufacturing systems. The CNC serves as a hub through which vital information flows. CNC controllers in Industry 4.0 should be able to support interconnectivity with sensors, and cloud servers. The transition from traditional hardware-based controllers to smart automation software architecture is a difficulty. Security issues develop as a result of modern industry 4.0 manufactures heavily prioritizing functionality over security. Increased economic losses, loss of production, and even death could result from lack of security. IoT relies on the connection between the digital and analog components. As a result, industrial sabotage

and industrial espionage have skyrocketed in recent years. Security should be a key factor for the growth of industry 4.0 due to levels of awareness, vulnerabilities in existing measures, and planning for future issues are vital. therefore, security must play a key role in the development of Industry 4.0. If industry 4.0 manufacturing automation developers could discover cyber security requirements that have not yet been fully captured in automation and design and develop systems that address all security aspects in automation, the developing automation systems would be possibly risk-free and secure.

For industry 4.0 automated manufacturing systems, CPS play a critical role in cyber security. The base of an industry 4.0 should allow garment manufacturing automation, including CNCs, to provide the enhanced performance while maintaining security.

1.2 Literature Review

Central Bank Annual Report 2019 Textiles and garments account for 46.9% of industrial exports in Sri Lanka, whereas textiles and textile articles account for 14.6 percent of imported intermediate goods [1]. Adapting Industry 4.0 concepts and establishing textile 4.0 manufacturing facilities will enhance productivity, which will have an impact on export rates.

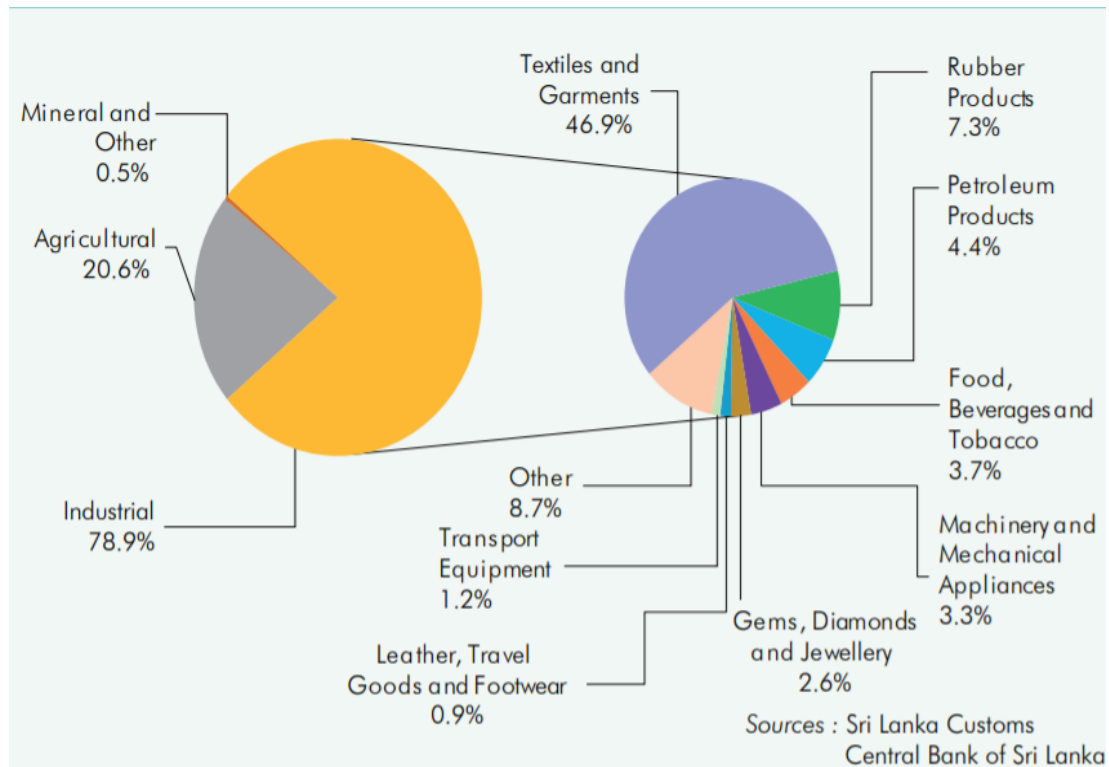


Figure 1.2: Composition of Exports - 2019

Knitting, cutting, stitching, finishing, product quality checking, packing, storing, and distribution are just a few of the processes that are involved in garment production. Scarp fabrics are produced in the textile industry at a rate of 60 billion square meters per year [2]. The majority of scarp fabrics are created during cutting operations. To address this issue, Abu Sadat Muhammad Sayem and Shreshta Ramkalaon [2] developed Zero-Waste Pattern Cutting (ZWPC). With the use of Computer-Aided Design (CAD) and Computer-Aided Manufacturing (CAM) applications, the modern textile industry uses CNC-based cutting machines to automate the cutting process. Cutting processes in Industry 4.0 can be automated using a CPS with a smart CNC, IoT devices and sensors.

Many studies have revealed that CPS and IoT have the same underlying architecture. CPS have high level of integration and collaboration between physical and electronic components. CPS architecture is composed of two main layers. The physical layer takes sensed data, acts on commands received from the cyber layer, and sends acquired data back to the cyber layer. While the cyber layer processes received data from physical layer and issue commands back to the physical layer [4].

When it comes to cyber-attacks against CPS, the Stuxnet worm is a malware that specifically targets CPS. Rather than collecting, altering, or erasing information, the purpose of this modernized cyber warfare weapon is to physically destroy its target [5]. The Stuxnet worm was effective in infecting the Iranian nuclear plant at Natanz in 2010 and gaining access to the facility's operating systems via four zero-day vulnerabilities. Stuxnet accomplished its purpose by interrupting power to centrifuges, bringing an end to Iran's nuclear program [6].

The Stuxnet worm highlighted that the Confidentiality, Integrity, and Availability (CIA) triad is insufficient for CPS security. According to Yosef Ashibani and Qusay H. Mahmoud's CPS security analysis [7], the following aspects should be considered when safeguarding CPS.

Securing access to devices

As previously stated, CPS incorporates a variety of devices including IoT devices, sensors, CNCs, and Radio-frequency identifications (RFIDs). Because some of these devices have poorly built or configured authentication mechanisms, protecting access to them is difficult [8]. Therefore, acquiring unauthorized access to CPS devices is easier than gaining unauthorized access to computer systems. According to the Open Web Application Security Project (OWASP) CPS and IoT devices have hardcoded passwords or weak guessable passwords that can be brute forced quickly, and some of these passwords are publicly available [9]. According to OWASP, these devices lack physical security, which enables side channel attacks. By conducting a side channel attack that examines the devices' electrical signals, attackers can predict or replicate the actions performed by these devices [10].

Securing data transmissions

CPS, like other network devices, is susceptible to Denial of Service (DoS) attacks, which disrupt network activity and render CPS unusable [11]. According to OWASP, CPS devices contain unsecured network services, which are redundant services that run on the devices and are exposed to the internet [9], allowing attackers to manipulate these devices remotely. Some CPS devices cannot be called comprehensive computer systems because to their limited data processing and storage capabilities [12]. Furthermore, the majority of CPS device manufacturers prioritize functionality over reliability and security. As a result, malware, Distributed Denial of Service (DDoS) attacks, eavesdropping, and unauthorized access are more capable of affecting CPS devices [7].

Securing applications

The primary security concern with the application layer is privacy. Other security concerns on the application layer have a low probability of occurring. Data such as fashion design templates and employee information can be deemed private data in the context of garment manufacturing system. Loss of privacy can result in legal repercussions such as fines and penalties. As a result, programs that employ CPS must be secured [13]. According to OWASP, these applications do not provide adequate privacy protection [9].

Securing data storage

The majority of CPS devices have low resources. these nodes cannot protect data at rest [11]. Sensors and other CPS devices have limited memory and processing power. As a result, lightweight ciphers are required to encrypt data at rest [14]. Lightweight cipher is not used in the most software-based systems. These softwares are therefore incompatible with CPS devices.

Securing actuation

Because CPSs are capable of physical interactions, any commands given to a CPS device should come from trustworthy and authorized sources. Securing actuation prevents CPS from acting in response to rogue feedbacks and attackers unauthorized commands [12]. If the CPSs are operated through the internet, the following attack

types are possible: eavesdropping, Man in the Middle attacks, spoofing, replay attacks, and compromised keys. As a result, security must be implemented throughout the system [11].

As shown by OWASP [9], insecure default settings, a lack of update, and device management after deployment in production can result in the security vulnerabilities described above.

Due to the immense number of devices, manually securing them individually is time intensive. In order to save time, Infrastructure as Code (IaC) must be introduced. DevOps employs IaC as one of its primary strategies for automating logic for deploying, configuring, and updating devices [15]. Securing CPS devices necessitates the use of IaC's configuration and update management features. As a result, configuration management IaC platforms like Chef, Puppet, Ansible, and SaltStack [16] are ideal candidates for this purpose.

Puppet is a master-agent architecture opensource configuration management tool. Puppet master operates on a Linux/Unix server, whereas puppet agents operate on Windows, Linux, and Unix client devices. Puppet requires signed certificates on both the master and the agents. For configuration management, Puppet uses the Puppet DSL (Domain Specific Language) [17].

Chef is master-agent architecture opensource configuration management tool. Chef resembles Puppet in many ways. Chef master operates on a Linux/Unix server, whereas Chef agents operate on Linux/Unix/Windows client devices. Chef includes a workstation component that allows it to test all its configurations. To manage configurations, Chef employs Ruby as a Domain Specific Language (DSL) [18].

RedHat developed Ansible, an opensource configuration management software. Ansible is built on a client-server model, with the server being a Unix or Linux system and clients being Linux/Unix/Windows machines. Secure Shell (SSH) connection between the server and the clients is required. Ansible necessitate the installation of a specific agent on client devices. Ansible manages configurations with "YAML Ain't Markup Language" (YAML) [19]. YAML is a human-readable data-serialization language, not a markup language, as its name implies.

SaltStack is a master-client architecture opensource configuration management tool. Salt master is a server that runs Linux/Unix, while Salt minion (agent) is a client that runs Windows/Linux/Unix. SaltStack manages configurations with YAML. [20].

Table 1.1: Comparison of IaC configuration management platforms

	Chef	Ansible	SaltStack	Puppet
Code	Opensource	Opensource	Opensource	Opensource
Language	Ruby	YAML	YAML	Puppet DSL
Ease of Setup	Difficult	Easy	Difficult	Difficult

1.3 Research Gap

OpenSCAP and CIS-CAT Pro are two popular industrial tools for security configurations and system hardening. However, these tools are designed for server hardening, and none of them are capable of hardening CPS device-based operating systems such as Raspbian Operating System (OS) and Robot OS (ROS). To automate security configuration and minimize system downtime, the proposed tool is primarily focused on CPS or IoT devices operating system hardening, which includes the above-mentioned operating systems as well as Ubuntu Linux. A comparison of operating system configuration capability between existing tools and the proposed tool is shown in the table below.

Table 1.2: Comparison of Existing Tools

Operating System	OpenSCAP	CIS-CAT Pro	Proposed tool
Ubuntu Linux	✗	✓	✓
Robot OS	✗	✗	✓
Raspbian OS	✗	✗	✓

1.4 Research Problem

Smart computing is combined with technologies such as IoT, cognitive computing, machine learning, and data analytics are used when automating a manual or semi-automated system towards Industry 4.0. When creating an industry 4.0 automated system, most system developers are unaware of the cyber security challenges. The goal of the study is to find applications of cyber security needs that have not yet been fully addressed in automation.

The development of the secure network environment is the main challenge in industry 4.0.

Industry 4.0's network environment is developed using a Cyber Physical Product Systems (CPPS) platform. developing the CPPS platform has become a difficult project due to following limitations,

1.4.1 Collaboration between different systems

A collaborative model between computer systems and physical devices is required For transmitting data, storing data, recording data, decision-making, and corrective and preventive action, [21].

1.4.2 Centralized security management

A central management system such as Supervisory Control and Data Acquisition (SCADA) can be utilized to apply security configurations/updates to physical devices and monitor physical devices [22]. A security model to provide security configurations and updates to CPS will enhance efficiency in managing security. In addition to the CPS modeling language, software, physical device environments, hardware platforms, and other functional and non-functional factors must be considered when creating a security model for CPS [21].

1.4.3 Secure communication

CPS that utilize SCADA systems communicate with the internet using TCP/IP protocols, which are known to be vulnerable [23].

1.4.4 Insecure data

When it comes to Industry 4.0, manufacturing companies often overlook data security. IoT-based CPSs, which are connected to many embedded sensors and communication devices, pose a substantial risk due to the increase in data utilization and the increased probability of system breaches [24].

1.4.5 Initial cost

Industry 4.0 transformation is not an instant process, but it will lead to end-to-end integration in a manufacturing organization, allowing it to plan and execute engineering according to business goals with considerable initial interest in cost and time [25].

1.4.6 Absence of methodology to industry 4.0

Lack of a dynamic fundamental arrangement to facilitate with the transformation to Industry 4.0 [26].

1.5 Research Objectives

1.5.1 Main objectives

The main goal of this research component is to develop a solution for automating security configurations for CPS devices. By default, these CPS devices have vulnerable configurations, and their life cycles are shorter than that of computer systems. As a result, system administrators must often analyze current security configurations and apply security configurations. Manual security configuration is inefficient and result in more system downtime. To reduce system downtime, the idea is to create a program that can automate and manage security configurations.

1.5.2 Specific objectives

1.5.2.1 Audit security configurations

A mechanism to audit security configurations is included in this suggested tool to ensure that security configurations are appropriately configured in CPS devices. The audit function will be written in bash and python, with ansible delivering the scripts to the CPS devices as ansible tasks. The audit function saves the results as a comma-separated values (csv).

1.5.2.2 Centralized device configuration management

The suggested tool will be installed on a Linux-based ansible controller. Using SSH keypairs or password-based authentication, SSH connections are established between ansible controller and CPS devices. The ansible controller's goal is to centralize the configuration management of CPS devices.

1.5.2.3 Generate audit reports

There is a function in this suggested tool to generate reports depending on audit results. The following information is included in the report: security configuration name, severity, description, and rationale. As a result, these audit reports can demonstrate applied security measures on a system to the users and management.

2. METHODOLOGY

2.1 Methodology

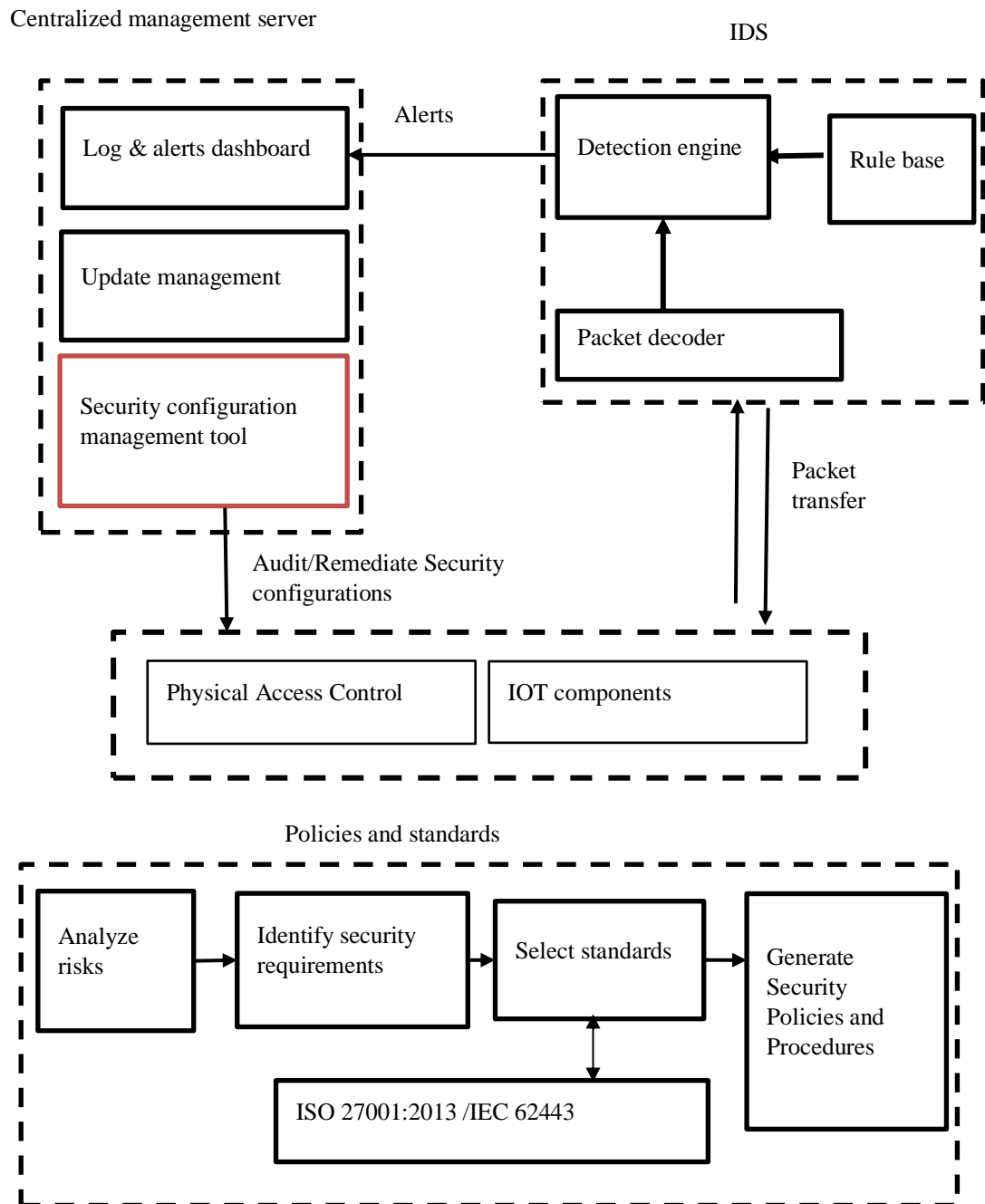


Figure 2.1: Overall System Diagram

Figure 2.1 Overall system diagram illustrates how my component the security configuration management tool fits in to our overall security solution of industry 4.0

garment manufacturing system. The security configuration management tool is installed in a centralized management server called ‘ansible controller’ from there the tool audit and remediate IoT devices or CPS devices connected to the network.

Below Individual workflow was followed by me during this research.

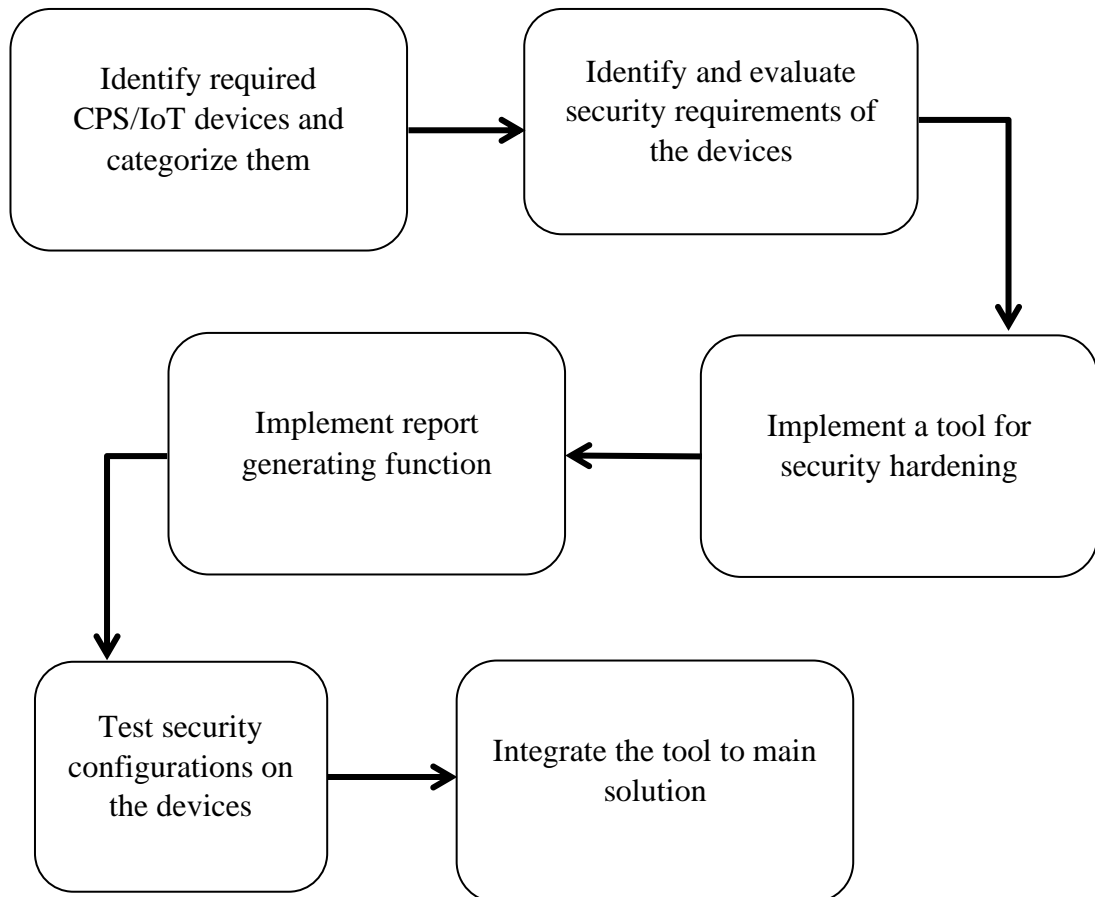


Figure 2.2: Individual Workflow Diagram

2.1.1 Required CPS and IoT devices identification

A field visit is taken to Wijeweera Knit Wear (Private) Limited garment manufacturing factory to identify garment manufacturing processes. We acquired knowledge about how CPS devices can be integrated into manufacturing processes. For an example CNC machines are used in cutting process.

To research security aspects of CNC and IoT devices, a field visit is taken to Department of Manufacturing and Industrial Engineering, University of Peradeniya, Sri Lanka. Understanding of engineering aspects of CNC machines and current security measures applied to these devices are the main learning outcomes from the conducted field visits.

The obtain knowledge is applied to create security rules required to harden these IoT systems.

2.2.2 IoT Security requirement analysis and evaluation

The OCTAVE-Allegro risk assessment methodology was used to identify and evaluate risks to the purposed overall system. A critical assets identification is done to identify critical assets in the purposed system. Ansible controller and IoT devices were identified as critical assets in the system. Since ansible controller and IoT devices are the key components in my purposed solution, I have conducted risk assessment on those devices.

Several threats to the ansible controller and IoT devices were identified. Evaluations were conducted on identified threats to measure their potential impact and risk to the system. The threats that have significant impact and risk to the system is listed in Table 2.1. Figure 2.3 contains a heatmap of threats shown in Table 2.1.

Table 2.1: Identified Threats Against Ansible Controller and IoT Devices

ID	Name
A1	Unauthorized access to ansible controller via network - Ansible controller
A2	Disclosure of access credentials - Ansible controller
A3	Denial of service attacks - Ansible controller
A4	Unauthorized access to ansible controller via console - Ansible controller
A5	Ransomware - Ansible controller
A6	Spyware - Ansible controller
A7	Power outage - Ansible controller
A8	Trojan - Ansible controller
A9	Overloading system storage - Ansible controller
B1	Overheat - IoT
B2	Ransomware - IoT
B3	Unauthorized access to ansible controller via network - IoT
B4	Disclosure of access credentials - IoT
B5	Denial of service attacks - IoT
B6	Spyware - IoT
B7	Power outage - IoT
B8	Overloading system storage - IoT

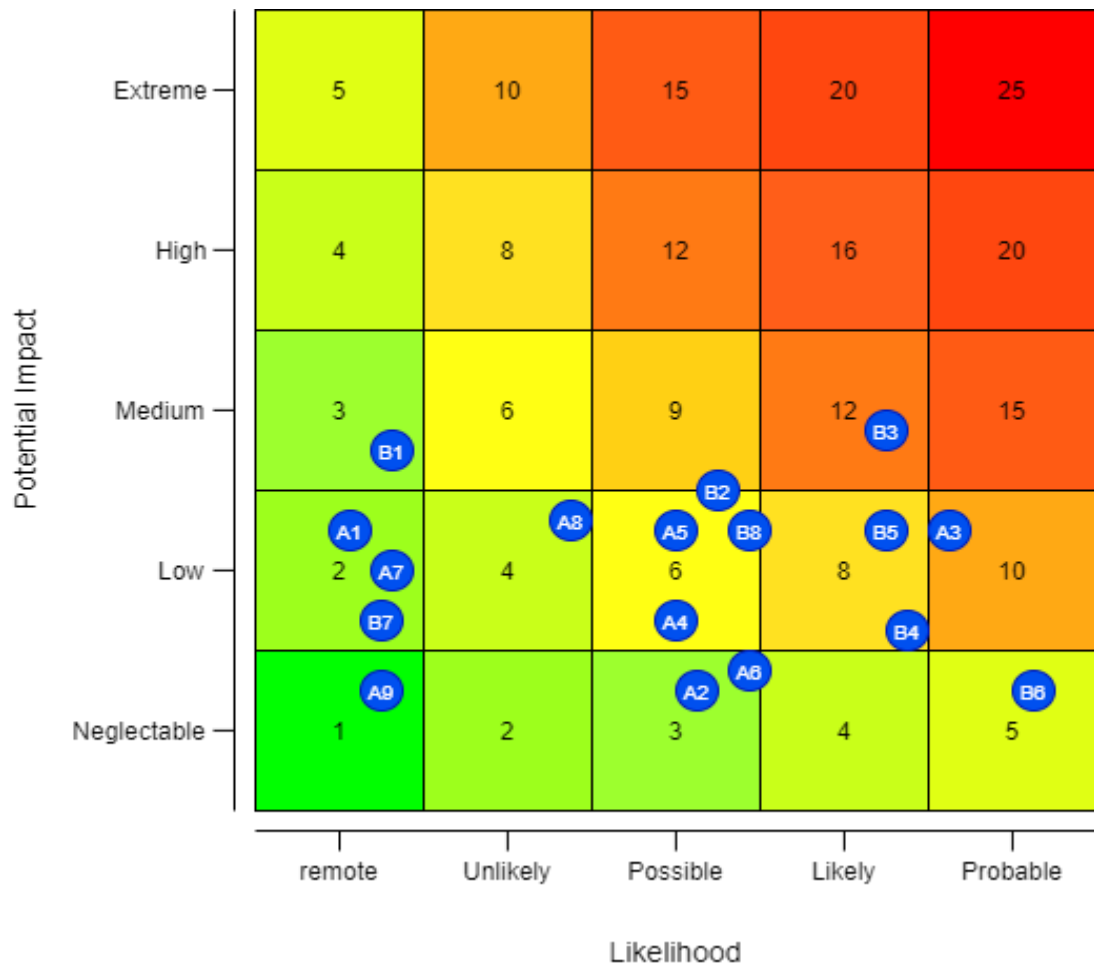


Figure 2.3: Heatmap of identified threats against ansible controller and IoT devices

Technical security measures to mitigate these identified threats will be included in the security configuration management tool as security rules.

2.2.3 The security configuration management tool

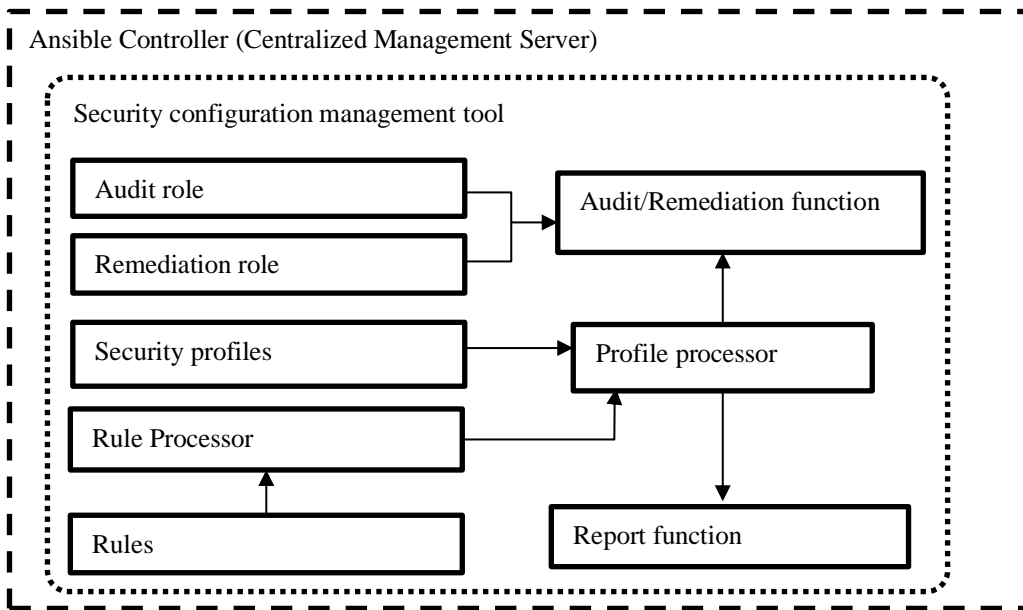


Figure 2.4: Components inside security configuration management tool

The Security configuration management tool is mainly written in Python3, YAML, and Ansible. The tool only provides command line interaction to the user and it only supports Linux because Ansible does not support Windows as the host. The tool provides following main functionalities to the users,

- Audit selected group of IoT devices
- Remediate selected group of IoT devices,
- Generate audit reports

In addition to mentioned main functionalities the tool has several sub functionalities to provide details to the users about available security profiles and rules. The Components in Figure 2.4 are used to implement the functionalities. Implementation of main functionalities are described in later part of this document.

2.2.3.1 Audit function

The audit function in the security configuration management tool is used for scanning compliance of selected IoT devices based on security rules stated in the security profile. The audit function takes security profile id and host group as inputs. The tool requires active SSH connections and administrative access to the selected IoT devices otherwise the audit will fail. As mentioned before Ansible requires SSH connections to communicate between host and clients. Ansible requires administrative access to IoT devices to perform certain actions for example. Accessing bootloader configurations for auditing requires administrative access because these configurations are saved in grub file which can only be accessed by the root user.

The main objective of using security profile is to define which rules to be audited or remediated using the tool. These profiles can be custom tailored based on level of security required by the organization. The security profiles are written in YAML and contains following information shown on the Table 2.2.

Table 2.2: List of Information in Security Profiles

Label	Description
ID	Security profile ID
Name	Security profile name
Category	Security profile category
Applicable hosts	Security profile is applicable to which type of IoT devices
Target system	Name of the Operating system used in IoT devices
Target system version	Version of the Operating system used in IoT devices
Description	Description about the security profile
Version	Version of the security profile
Rules	Contains the rule ids of the selected rule set

The host group defines group IoT devices to be audited. Ansible uses host group to deliver audit commands to client IoT devices. Host group is saved in a file called inventory located in the tool's root directory. The users can define IoT device host name, IoT device IP and group them based on requirements. The users must provide

access credential to client IoT devices in the inventory otherwise Ansible cannot establish SSH connection with IoT devices.

Once valid inputs are provided audit function calls profile processor to process the selected security profile and load its rules to create a YAML file called 'extra_vars.yml'. While loading rules, Profile processor will call Rule processor to cross validate mentioned rules are stated in 'rules.yml'. Then the audit function will create an Ansible playbook using audit playbook template that contains audit role combined with given host group. Then the audit function runs the playbook with previously created 'extra_vars.yml'. Finally, the audit function calls report function to generate reports based on audit findings. The explained audit process is illustrated in Figure 2.5.

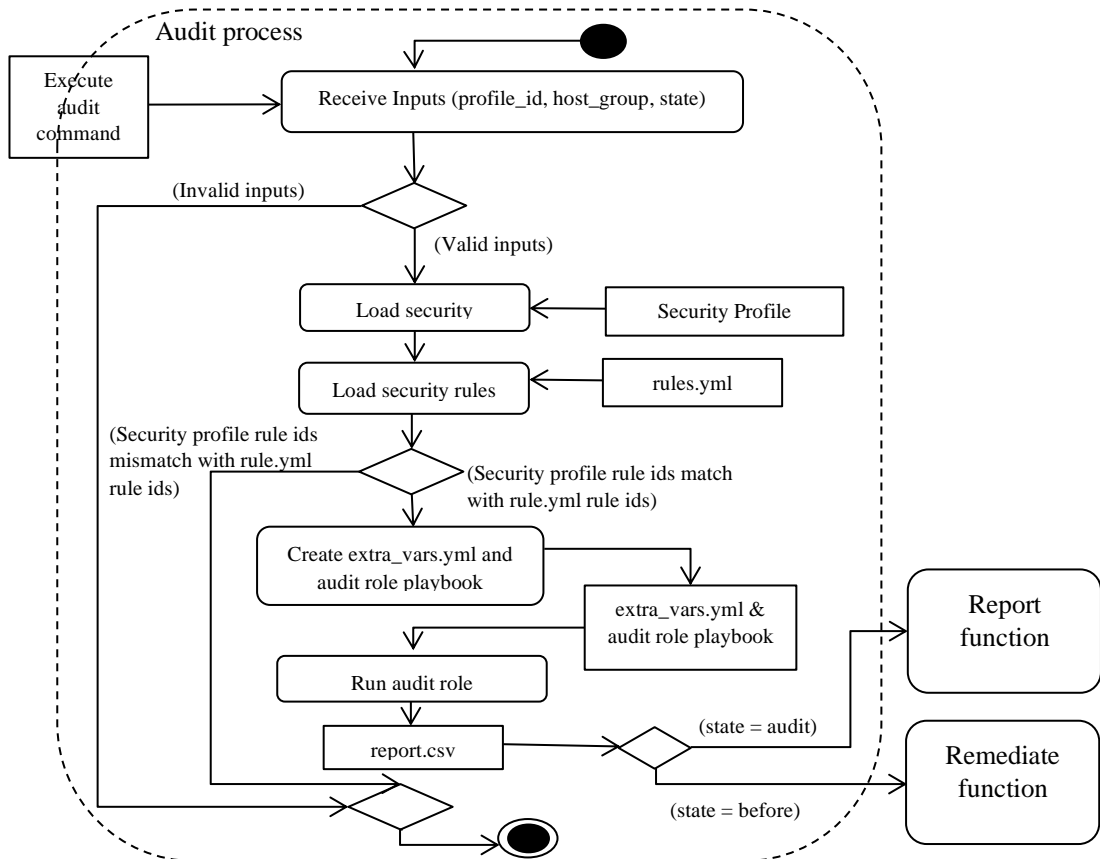


Figure 2.5: Audit process

Audit role is an Ansible role, created to deliver audit commands to client IoT devices via SSH. The commands to audit selected rules are converted into tasks inside the audit role. These tasks contain following ansible modules,

- File – create and delete .csv audit reports
- Lineinfile – insert audit findings to the .csv audit reports
- Shell – audit current system configurations

Once audit role is executed, it will access ‘main.yml’ stored in tasks directory and executes tasks mentioned in ‘main.yml’ sequentially. The first task is to create an audit report on home directory of the user which ansible use to access the system. Report is a .csv file named after device host name and device IP address as shown below. Ansible uses ‘ansible_facts’ to gather information about system such as host name and IP address.

‘Device hostname’_‘Device IP address’_Audit_report.csv

For an example, if a device has its host name as ‘IOTA’ and device IP address is 10.0.0.2 the name of the created audit report is shown below.

IOTA_10.0.0.2_Audit_reprot.csv

Then tasks in sections executes in sequential order to audit security rules and write findings to the audit report. Once the tasks in sections are complete, the report is fetched to ansible controller and the report in client is deleted.

The audit tasks are divided into 6 main sections to reduce complexity of the audit role. Dividing task into sections allows future addition of tasks to audit security rules making the tool more scalable and flexible for improvements. The contents of 6 main sections and described in Table 2.3. The audit tasks in sections have shell modules to audit current configuration of the client system and the audit findings are written the audit report.

Table 2.3: Sections Available in Audit Role Tasks

Section ID	Name	Description	Content
Section 1	Initial setup	Contains rules that check correct initial system settings	<ul style="list-style-type: none"> • Filesystem configuration • Sudo configuration • Filesystem integrity checking • Secure boot settings • Process hardening • Mandatory access control • Warning banners
Section 2	Services	Contains rules that check availability of services	<ul style="list-style-type: none"> • Time synchronization services • Special purpose services • Service clients
Section 3	Network Configuration	Contains rules that check network configurations and firewall configurations	<ul style="list-style-type: none"> • Network parameters • Uncommon protocols • Firewall configuration
Section 4	Logging and Auditing	Contains rules that check logging, log monitoring, and auditing	<ul style="list-style-type: none"> • System accounting • Data retention • Logging • Journald configuration
Section 5	Access, Authentication and Authorization	Contains rules that check access, authentication and authorization	<ul style="list-style-type: none"> • Cron configuration • SSH configuration • PAM configuration

			<ul style="list-style-type: none"> • User accounts and environments
Section 6	System maintenance	Contains rules that check system maintenance	<ul style="list-style-type: none"> • System file permissions • User and group settings

The following format is used to write audit findings on the audit report,

'Rule_id', 'Result'

'Rule_id' is same as rule_id stated in rules.yml and 'Result' is a binary value. If audit found system is complaint with audited security rule the value of 'Result' is 1 and If system is not complaint with audited security rule the value of 'Result' is 0. For an example, if a security rule with 'Rule_id' T_101 is audited and system is complaint with the audit rule the result written in audit report is shown below.

T_101,1

The file structure of the audit role is illustrated in Figure 2.6. In audit role, each section has several tasks. An example is shown in section_1. 'main.yml' under defaults has several variables that gets overridden by 'extra_vars.yml'. These variables are used by remediation function and they will be discussed on later part of this document. 'main.yml' in vars directory contains dynamic variables to store the audit report path and the audit report name

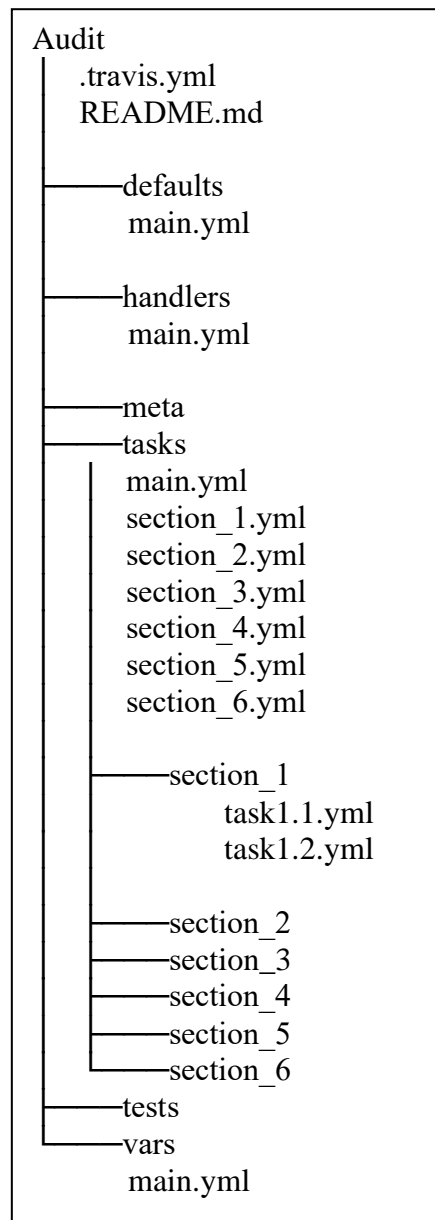


Figure 2.6: Audit role file structure

2.2.3.2 Remediate function

The remediate function in the security configuration management tool is used for reconfiguring non complaint systems to secure them. Similar to audit function, remediation takes security profile id and host group as inputs. The tool requires active SSH connections and administrative access to the selected IoT devices otherwise the remediate function will fail because remediating most of the security rules requires administrative privileges.

Once valid inputs are provided remediate function calls audit function by passing received inputs and an extra input called 'state'. By default, the value of 'state' variable in the audit function is set to 'Audit' however in this scenario remediate function overrides 'state' value as 'Before'. As mentioned before audit function will call profile processor to create a YAML file called 'extra_vars.yml'. The value of 'state' is also added to 'extra_vars.yml'. Then the audit function will create an Ansible playbook and run the playbook with previously created 'extra_vars.yml'. Once the audit is finished, the remediate function will create an Ansible playbook using remediate playbook template that contains remediate role combined with given host group. Then the remediate function runs the playbook with previously created 'extra_vars.yml'. Once the remediation is finished, the remediation function calls audit function by passing received inputs and value of 'state' as 'After'. The value of 'state' is also added to 'extra_vars.yml'. Finally, two audit reports of each device are combined using 'dataframes' in 'pandas' Python library and report function is called to generate reports based on the combined audit reports. The explained remediate process is illustrated in Figure 2.7.

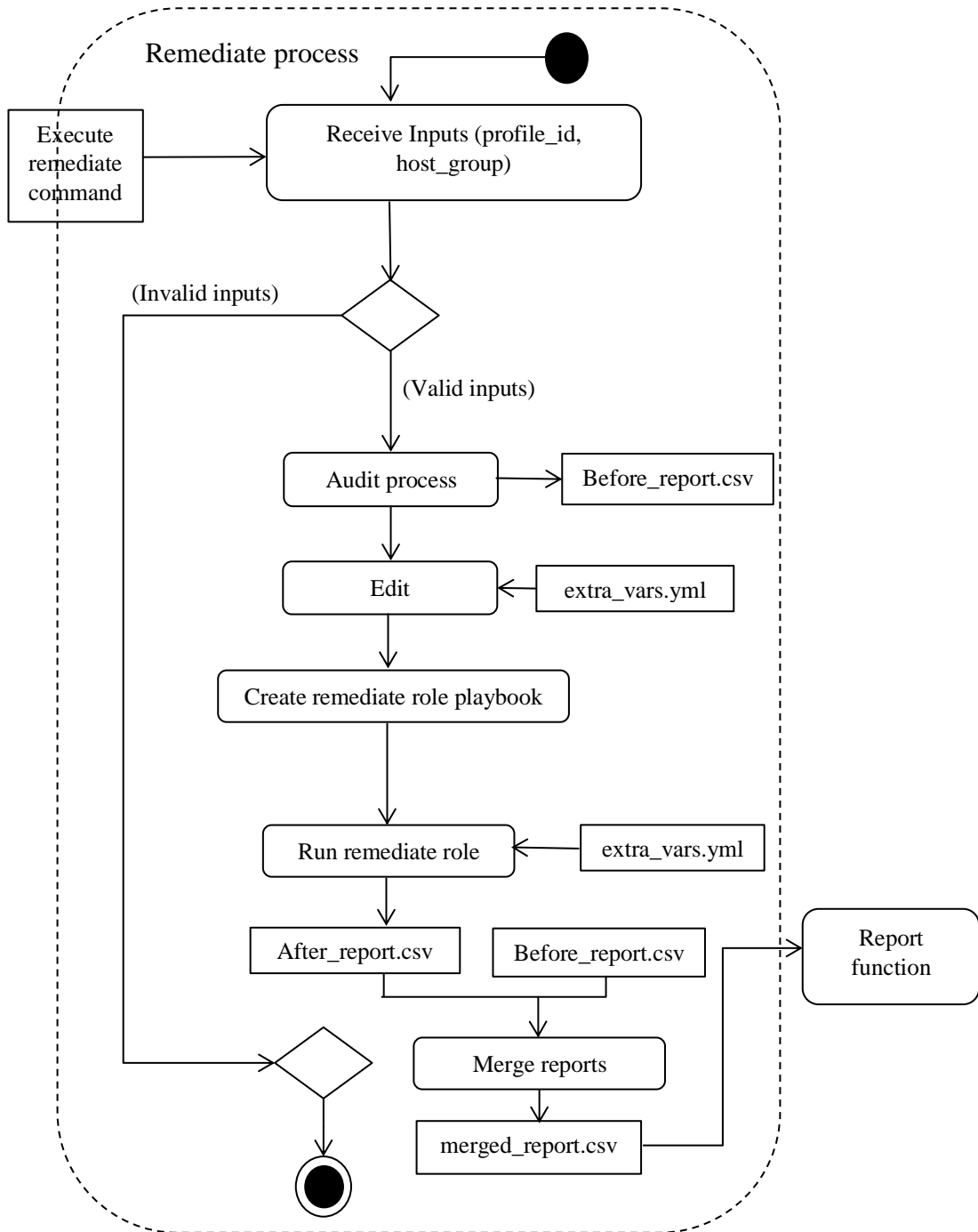


Figure 2.7: Remediate process

As mentioned before, report is a .csv file named after device host name and device IP address as shown below. However, 'Audit' is replaced by 'state' in remediation to uniquely identify two reports.

'Device hostname' _ 'Device IP address' _ 'State' _report.csv

For an example, if a device has its host name as ‘IOTA’ and device IP address is 10.0.0.2 the names of the created audit reports during remediation are shown below.

IOTA_10.0.0.2_Before_reprot.csv, IOTA_10.0.0.2_After_reprot.csv

Once remediate role is executed, it will access ‘main.yml’ stored in tasks directory and executes tasks mentioned in ‘main.yml’ sequentially. The remediate tasks are divided into 6 main sections to reduce complexity of the remediate role. Dividing task into sections allows future addition of tasks to remediate security rules making the tool more scalable and flexible for improvements. The contents of 6 main sections and described in Table 2.3. Remediate role use variety of Ansible modules to configure systems. Used Ansible modules and their purpose is described in Table 2.4. More information about these modules can be found in Ansible documentation [18].

Table 2.4: Ansible Modules Used in Remediate Role Tasks

Module name	Purpose of use
lineinfile	Edit configuration files
modprobe	Load/unload kernel modules
mount	Mount/unmount file systems
systemd	Enable/disable/start/stop/restart services
apt	Install/remove packages
file	Create/delete files and edit file permissions
sysctl	Configure kernel parameters
shell	Execute shell commands
copy	Copy files from ansible controller to client devices
ufw	Configure firewall
pamd	Configure Pluggable Authentication Modules (PAM)

The file structure of remediate role is similar to audit role file structure shown in Figure 2.6. However, new directory called files is added to store login banners. Modified file structure shown in Figure 2.8.

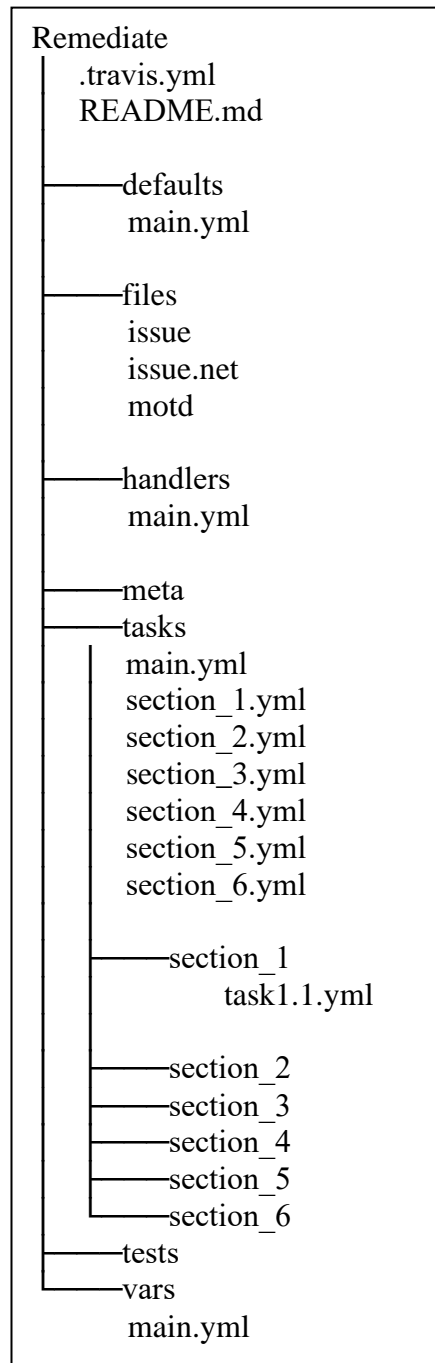


Figure 2.8: Remediate role file structure

2.2.3.3 Report function

Report function is used to generate comprehensive reports at can be viewed by users and organization management. Each report contains device details such as device host name, device IP, device host group, report created date, and time. Reports also contain graphs to illustrate summary about total rules scanned during audit. The graphs are used to illustrate total rules scanned their pass/fail ratio, high severity rules pass/fail ratio, medium severity rules pass/fail ratio, and low severity rules pass/fail ratio. Sample device details are shown in Figure 2.9 and sample summary about totals rules scanned during audit shown in Figure 2.10.

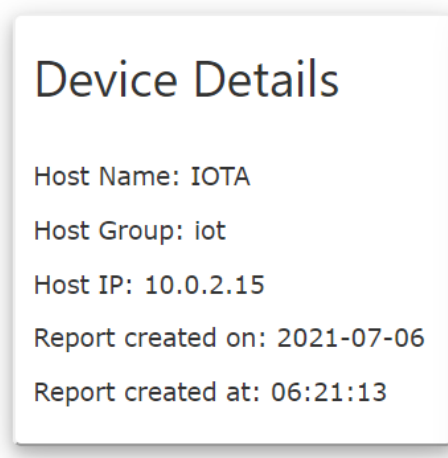


Figure 2.9: Sample device details

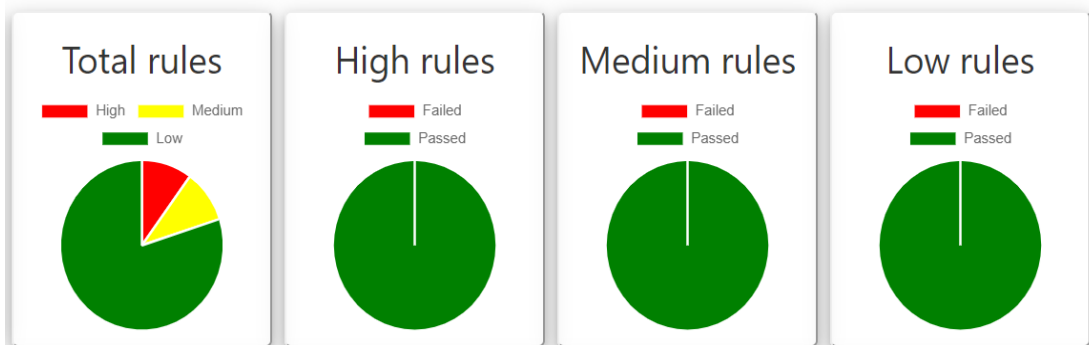


Figure 2.10: Sample audit summary

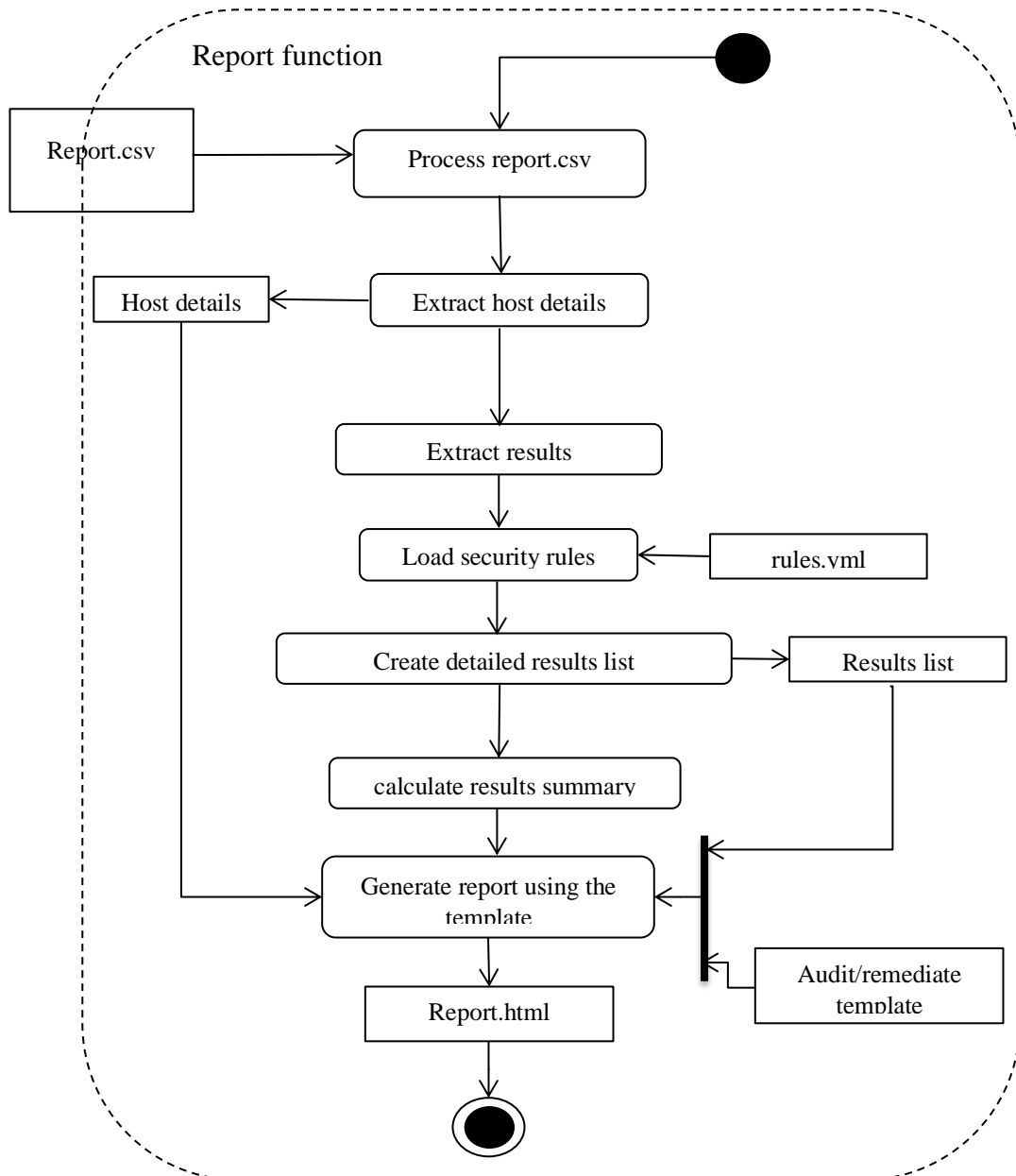


Figure 2.11: Report function process

Once report function is called report function will read received audit reports from audit or remediate. Then report function extracts rule ids from audit reports and calls rule processor to get more information about a rule. Rule processor loads rules.yml file and search rules by rule_id and obtain the information about rules stored in rules.yml file. The rules.yml file contains following information shown in table 2.5.

Table 2.5: List of Information in rules.yml

Label	Description
rule_id	Contains the security rule ID.
name	Contains the security rule name.
scored	States whether security rule has an impact or not.
severity	Criticality of the security rule, if the vulnerabilities were exploited.
version	Version of the security rule.
description	Brief description about the security rule explaining what vulnerabilities are mitigated by the security rule.
rationale	Brief description about the outcome, if vulnerabilities mentioned in description are exploited.
applicable_to	States which types of devices are affected by the security rule.

Once information about rules are returned to report function, the function will determine rule severity and calculate audit summary based on audit result and rule severity of each rule. Once all rules in audit report are processed, report function write rule results and information to html template files using ‘jinja2’ python library. There are two different templates for audit and remediate because systems are audited twice during remediation.

The templates contain placeholders for device details summary, results table, results summary, and additional details about rules. Templates are used because reports are dynamic based on devices and their system configurations. These templates contain html code with bootstrap scripts to generate dynamic tables, graphs, and modals. The bootstrap ‘datatables’ library allows generation of dynamic tables with sorting, searching, and pagination functionalities. A sample results table is shown in Figure 2.12. According to Table 2.5, rules contain variety of information that cannot be entered to results tables because it causes table to be unreadable. Therefore, modals are used to display that additional information to the user. A sample modal is shown in Figure 2.13.

Show 10 entries

Search:

ID	Name	Result	More details
T_101	Ensure mounting of cramfs filesystems is disabled	Pass	+
T_102	Ensure mounting of squashfs filesystems is disabled	Pass	+
T_501	Ensure cron daemon is enabled	Pass	+
T_502	Ensure permissions on /etc/crontab are configured	Pass	+
T_503	Ensure permissions on /etc/cron.hourly are configured	Pass	+
T_504	Ensure permissions on /etc/cron.daily are configured	Pass	+
T_505	Ensure permissions on /etc/cron.weekly are configured	Pass	+
T_506	Ensure permissions on /etc/cron.monthly are configured	Pass	+
T_507	Ensure permissions on /etc/cron.d are configured	Pass	+
T_508	Ensure at/cron is restricted to authorized users	Pass	+

Showing 1 to 10 of 10 entries

Previous

1

Next

Figure 2.12: Sample report results table

Ensure mounting of cramfs filesystems is disabled	
Attributes	Details
ID	T_101
Name	Ensure mounting of cramfs filesystems is disabled
Scored	1
Version	1.0.0
Severity	high
Description	The cramfs filesystem type is a compressed read-only Linux filesystem embedded in small footprint systems. A cramfs image can be used without having to first decompress the image.
Rationale	Removing support for unneeded filesystem types reduces the local attack surface of the server. If this filesystem type is not needed, disable it.
Applicable To	IOT,ROS
Audit Result	Pass

Figure 2.13: Additional details about rule T_101

2.2 Commercialization Aspects of the Product

Most manufacturers switch to industry 4.0 to increase their productions and security is not their primary concern. Only larger manufacturers have enough budget to hire employees to handle cyber security of their manufacturing systems. Even though, most manufacturers neglect security or transfer responsibilities of handling security to third vendors. Small and medium manufacturers may have to completely neglect security due to budget constraints.

The security configuration management tool is ideal for large, medium, and small manufacturers to secure their IoT systems. The tool automates security hardening and it does not require security professionals to use the tool. System administrators can effortlessly audit and remediate IoT systems using the tool. This allows manufacturers to reduce budget spent on security. Manufacturers can reduce production loss due to security procedures by using the tool rather than performing same security procedures manually.

We will gauge our target audience through Facebook, Twitter, Instagram campaigns, and official webpage.

2.3 Testing

A virtual environment is used for testing the security configuration management tool. VirtualBox is used to create virtual machines and virtual network for testing simulations. 6 virtual machines are created for testing, one virtual machine acts as ansible controller while other 5 virtual machines act as client devices. Ansible controller runs on Ubuntu Linux. 3 client devices run on Raspberry Pi OS and 2 client devices run ROS. The security configuration management tool is installed in ansible controller with all its dependencies. SSH connection is configured on client devices and ansible controller. Administrative accounts are created on client devices and the credentials are given to the security configuration management tool. The devices are connected to each other through a Host-Only network shown in Figure 2.14.

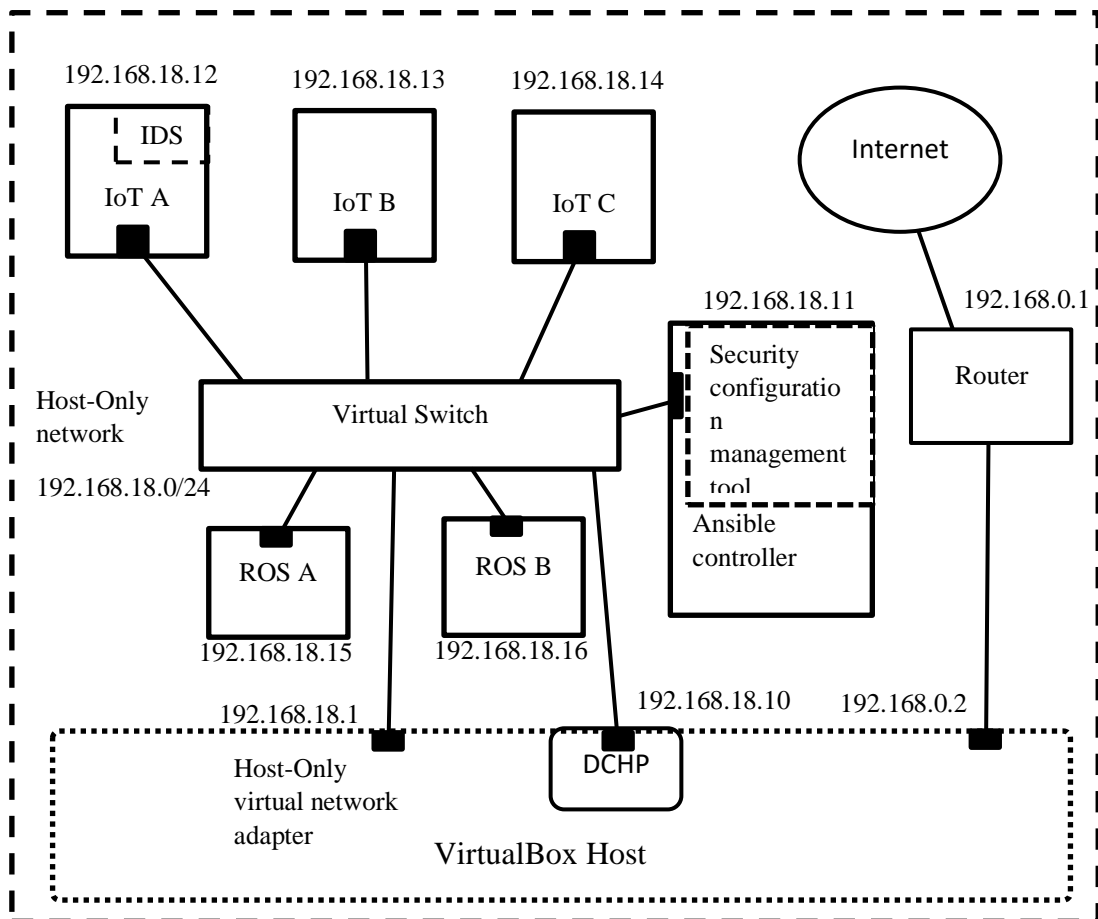


Figure 2.14: Network diagram of test scenarios

Two test scenarios are created to test the security configuration management tool using previously mentioned virtual machines. The propose of using these scenarios is to

identify the tools effectiveness and efficiency of using centralized system against decentralized system where manual command line configurations or shell scripts are used to secure devices.

In first scenario, 3 IoT systems and 2 ROS systems start with default configurations. In addition to default configurations active SSH service and administrative accounts are created on all these systems. To test whether security mitigations affects existing services IDS is installed on IoT A. The security configuration management tool is used to remediate devices and the time it takes to remediate IoT, ROS systems are recorded separately. The virtual machines are reverted. The same security mitigations applied manually using commands or shell scripts and the time it takes to remediate systems are recorded separately. The security rules or mitigations used during test is shown in Table 2.6.

In second scenario, 3 IoT systems and 2 ROS systems start with default configurations. In addition to default configurations active SSH service and administrative accounts are created on all these systems. To test whether that the tool can identify pre-remediated systems, IoT A and ROS A are remediated before the tests. To test whether security mitigations affects existing services IDS is installed on IoT A. IoT B is partially remediated to test whether the tool can identify already remediated security rules and remediate only necessary security rules that are not remediated before. The security configuration management tool is used to remediate devices and the time it takes to remediate IoT, ROS systems are recorded separately. The virtual machines are reverted to the test starting phase. The same security mitigations applied manually using commands or shell scripts and the time it takes to remediate systems are recorded separately. The summary of performed tests on two test scenarios are shown in Table 2.6. The security rules used in security profile for testing are listed in Appendix B.

Table 2.6: Summary of Performed Tests

Test ID	Test 1	Test 2	Test 3	Test 4
Used method	The security configuration management tool	Manual commands and shell scripts	The security configuration management tool	Manual commands and shell scripts
IoT A starting configuration	Default configuration	Default configuration	Secured	Secured
IoT B starting configuration	Default configuration	Default configuration	Partially secured	Partially secured
IoT C starting configuration	Default configuration	Default configuration	Default configuration	Default configuration
ROS A starting configuration	Default configuration	Default configuration	Secured	Secured
ROS B starting configuration	Default configuration	Default configuration	Default configuration	Default configuration

3. RESULTS & DISCUSSION

3.1 Results

The results from performed test scenarios are recorded in Table 3.1 and Table 3.2. Time in Test 2 and Test 4 includes time it took due to human error and filling check lists due extensive security rule list used for the tests. The ansible controller required at least 4 Giga bites of Random-Access Memory to execute the security configuration management tool without crashing.

Table 3.1: The Results from Performed Audits

Device	Test 1	Test 2	Test 3	Test 4
IoT A audit time	12 minutes	100 minutes	12 minutes	95 minutes
IoT B audit time	12 minutes	95 minutes	12 minutes	105 minutes
IoT C audit time	12 minutes	90 minutes	12 minutes	105 minutes
ROS A audit time	13 minutes	95 minutes	13 minutes	90 minutes
ROS B audit time	13 minutes	105 minutes	13 minutes	100 minutes
Total time	25 minutes	485 minutes	25 times	495 minutes

Table 3.2: The Results from Performed Remediations

Device	Test 1	Test 2	Test 3	Test 4
IoT A remediation time	23 minutes	190 minutes	23 minutes	0 minutes
IoT B remediation time	23 minutes	200 minutes	23 minutes	150 minutes
IoT C remediation time	23 minutes	185 minutes	23 minutes	195 minutes
ROS A remediation time	25 minutes	195 minutes	25 minutes	0 minutes
ROS B remediation time	25 minutes	205 minutes	25 minutes	200 minutes
Total time	48 minutes	885 minutes	48 minutes	495 minutes

Test 1 and Test 3 have shown that the security configuration management tool simultaneously audit/remediate all the devices in IoT host group and ROS host group. Moreover, Test 3 have shown that the security configuration management tool simultaneously audit/remediate one security rule at a time before moving on to the next security rule. IoT A remediation time in Test 4 is 0 because IoT A is already secured. However, in Test 3 IoT A remediation time depends on IoT C because ansible performs tasks sequentially. Therefore, a task in IoT A must wait until the same task is completed in IoT B and IoT C.

Test 2 and Test 4 shown that security auditing using manual commands takes at least 90 minutes per device. Therefore, total audit time for 5 devices using mentioned security profile add up to 450 minutes at least. Remediation using manual commands takes at least 185 minutes per device. Therefore, total remediation time for 5 insecure devices using mentioned security profile add up to 925 minutes at least.

Test 3 revealed that the security configuration management tool can identify insecure systems, partially secure systems, and secure systems to apply necessary required remediations on the systems. Although IoT A must wait until the same task is completed in IoT B and IoT C, the task did not make any changes to existing IoT A configurations. In Test 4 accidental misconfiguration or adding same configurations twice to some sensitive configurations files on IoT A resulted system and service failures. These configurations files include grub configuration files, PAM configuration files, SSH configuration file and other system configuration files.

Before configuring of mentioned sensitive configuration files all the selected CPS systems for remediation must halt their current activities to prevent any data loss that may happen during remediation. For an example, if SSH used to communicate between CPS devices misconfigurations in SSH configurations may cause SSH service failures which interrupts communication between CPS devices.

During manual remediations (Test 2 and Test 4), noticeable time were spent on correcting misconfigurations and human error due to extensive list of security rules used on several devices. Automated remediations using the security configuration management tool (Test 1 and Test 3) had shown that time was not wasted due to human

error or misconfigurations, if ansible tasks were properly tested prior to the remediations.

Since the security configuration management tool can analyze audit results and produce audit reports, users do not have pay much attention to outputs displayed by the tool during audits and remediations. However, during Test 2 and Test 4 outputs of audit commands must be manually analyzed and recorded to create reports. During Test 2 and Test 4 devices were audited individually to reduce human error when analyzing audit commands outputs and creating reports. This addition time to create report is considered when calculating execution time.

3.2 Research Findings

There are several security hardenings guides for Raspbian OS (Raspberry Pi OS) and ROS. However, there are lack of automated tools for security hardening on IoT devices. The security configuration management tool automates security auditing and remediation of IoT devices that runs on ROS and Raspbian OS. There are popular tools such as OpenSCAP and CIS-CAT Pro provides similar functionality as the security configuration management tool but none of these tools supports IoT based OS.

The centralized architecture of the overall system allows ansible controller to connect all devices at once if required. This allows the security configuration management tool to automate security auditing and remediation of the IoT devices from a single centralized server.

The security configuration management tool does not require special agent programs to be installed on client IoT devices. OpenSCAP and CIS-CAT Pro require special agent programs to be installed on client devices.

The security configuration management tool can audit or remediate multiple IoT devices at once. OpenSCAP and CIS-CAT Pro can audit or remediate a single device at a time. The simultaneous auditing and remediating of the security configuration management tool greatly reduces total system down time due to security hardening compared to manual security hardening. The automated tool greatly reduces human error during security hardening and auditing compared to manual security hardening and auditing.

Ansible normally used for configuring multiple systems remotely. However, in the security configuration management tool ansible is used for both auditing system configurations and configuring multiple systems remotely. Even though shell scripts and commands are more suitable for auditing configurations, ansible tasks provide reliable platform to audit multiple devices at once. Where shell scripts and commands can only audit single device at a time.

3.3 Discussion

The tool and manual security hardening are used on two separate systems that have N number of insecure IoT devices. Based on obtained tests results audit time, remediate time per device, and total system down time can be calculated using following equations.

If time loss due to human error and analysis time in manual auditing and remediating is neglected, down time of a device using both methods will be same. For a given security profile audit time per device is t_1 and remediate time per device is t_2 . Down time of a device is calculated below

$$\text{Down time of a device}(r) = t_1 + t_2$$

Depending on security profile,

$$5 \text{ min} \leq t_1 \leq 20 \text{ min}$$

$$15 \text{ min} \leq t_2 \leq 30 \text{ min}$$

Total system down time calculation is shown below, here r is down time of a device.

$$\text{Total system down time} = \sum_{r=1}^N r$$

Manual method total system down time is shown below.

$$\text{Total system down time} = \sum_{r=1}^N r = Nr = N(t_1 + t_2)$$

Since the security configuration management tool is capable of audit and remediating devices simultaneously total system down time is same as down time of a device (r) if all the devices are based on same OS.

$$\text{Total system down time} = r = t_1 + t_2$$

Due to extensive security rules in security profile and large number of devices (N), time spent on correcting human error, results analysis and results evaluation cannot be neglected. This has significant impact on total system down time when using manual security hardening methods. If total time spent on correcting human error, results analysis and results evaluation is μ , total system down time when manual security hardening methods are used is calculated as shown below

$$Total\ system\ down\ time = \left(\sum_{r=1}^N r \right) + \mu = Nr + \mu = N(t_1 + t_2) + \mu$$

Results analysis and evaluation is automated in the security configuration management tool because of this time spent on correcting human error, results analysis and results evaluation do not have any significant impact on total system down time when using the security configuration management tool.

Since the security configuration management tool is automated, its tasks must be properly tested prior to security hardening. Otherwise, the tool will generate false results and cause misconfigurations on the systems. However, this test is only required once. Compared to time spent on correcting human error, results analysis and results evaluation during manual security hardening this test is much less time consuming.

The performed tests and analysis of tests results proves that the security configuration management tool much more effective than manual security hardening. Total system down time due to security hardening is also greatly reduced when using the security configuration management tool and number of devices does not affect total system down time. On the other hand, total system down time increase with number of devices when manual security hardening is used.

Manual security hardening does not generate detailed reports. The system administrators or security personal in charge of handling security in IoT devices must put extra effort to analyze and evaluate audit results to create reports. However, the security configuration management tool generates comprehensive human readable reports without using any extra labor from employees.

Centralized architecture of ansible controller allows the security configuration management tool to connect all the IoT devices in the network. This becomes a disadvantage, if ansible controller is compromised to attackers all the IoT devices in the network will be compromised because the tool has administrative access to IoT devices. Therefore, ansible controller becomes single point of failure and it must be secured. The security configuration management tool is also not effective in a system that has de centralized architecture because the tool cannot access all the IoT devices

in the network at once. This will increase the total down time of the system because security hardening must be performed on devices separately.

With use of Ansible tasks combined with security rules, the security configuration management tool is scalable enough to accommodate further sections that involves services or application level security to audit or remediate services or applications that runs on IoT devices.

Existing security rules can be used on more than one security profile. Most security rules are applicable for both Raspbian OS and ROS. The tool reuses security rules several security profiles. Some of These rules are also applicable to other OS types as well. Therefore, these security rules can be reused when expanding the tool supportability to other OS types. Security profiles can be customized based on requirements allowing tool to be more flexible.

Since opensource technologies such as ansible and python are used to implement security configuration management tool, initial cost of implementing the tool is reduced.

The security configuration management tool is ideal for large, medium, and small manufacturers to secure their IoT systems. The tool automates security hardening and it does not require security professionals to use the tool. System administrators can effortlessly audit and remediate IoT systems using the tool. This allows manufacturers to reduce budget spent on security.

The tool can be further developed by increasing it supportability to other OS types and addition sections to ansible tasks to audit and remediate services and application such as web server applications and database server services.

4. CONCLUSION

Although industry 4.0 and textile 4.0 are still new concepts for most manufacturers and their priority being increased production. Most industry 4.0 systems lack security management because of this data and communications are insecure making systems vulnerable to attacks. Due to large number of CPS and IoT devices in an industry 4.0 system security management becomes a complex task that requires a lot of time and effort. The main objective of my research component to produce a centralized security management system that can perform security hardening on IoT devices. Security configuration management tool is created and installed it on a centralized server to accomplish the main objective. Security auditing and report generation based on audit result functionalities added in the tool to provide further understanding for the users or organization management. The report contains details about applied security measure on the devices and their purpose.

The security configuration management tool uses Ansible to deliver security configuration instructions to the IoT devices because of this the tool support IoT based OS such as Raspbian OS and ROS. Other popular industrial security hardening tools do not support these mentioned OS. Ansible allows the tool to simultaneously audit and remediate IoT systems. This increase the tools efficiency while reducing system down time due system hardening. With use of ansible task and security profiles the tool provides scalability for future development and reusability of existing security rules. The security profiles are also customizable based organizational requirements making the tool more flexible.

The summative conclusion is that, the security configuration management tool plays a major part in industry 4.0 systems security. Not only it provides centralized security management to audit and remediate IoT systems efficiently but also it is scalable, flexible and reusable allowing the tool to develop as the industry 4.0 develops.

REFERENCE LIST

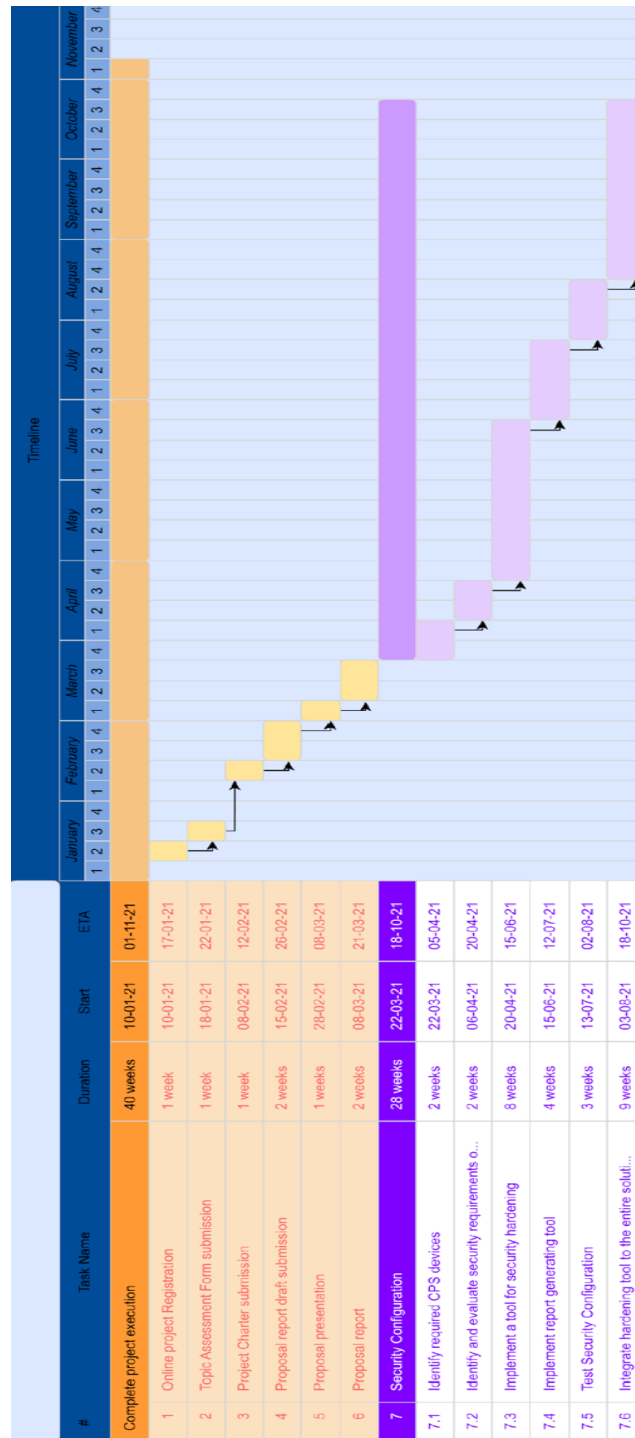
- [1] “Annual Report 2019 | Central Bank of Sri Lanka.”
<https://www.cbsl.gov.lk/en/publications/economic-and-financial-reports/annual-reports/annual-report-2019> (accessed Feb. 26, 2021).
- [2] S. Ramkalaon and A. S. M. Sayem, “Zero-Waste Pattern Cutting (ZWPC) to tackle over sixty billion square metres of fabric wastage during mass production of apparel,” *The Journal of The Textile Institute*, vol. 0, no. 0, pp. 1–11, Jun. 2020, doi: 10.1080/00405000.2020.1779636.
- [3] T. Lu, J. Lin, L. Zhao, Y. Li, and Y. Peng, “A Security Architecture in Cyber-Physical Systems: Security Theories, Analysis, Simulation and Application Fields,” *IJSIA*, vol. 9, no. 7, pp. 1–16, Jul. 2015, doi: 10.14257/ijsia.2015.9.7.01.
- [4] R. Langner, “Stuxnet: Dissecting a Cyberwarfare Weapon,” *IEEE Security Privacy*, vol. 9, no. 3, pp. 49–51, May 2011, doi: 10.1109/MSP.2011.67.
- [5] “What Is Stuxnet? | McAfee.” <https://www.mcafee.com/enterprise/en-us/security-awareness/ransomware/what-is-stuxnet.html> (accessed Feb. 27, 2021).
- [6] Y. Ashibani and Q. H. Mahmoud, “Cyber physical systems security: Analysis, challenges and solutions,” *Computers & Security*, vol. 68, pp. 81–97, Jul. 2017, doi: 10.1016/j.cose.2017.04.005.
- [7] C. Konstantinou, M. Maniatakos, F. Saqib, S. Hu, J. Plusquellic, and Y. Jin, “Cyber-physical systems: A security perspective,” in 2015 20th IEEE European Test Symposium (ETS), May 2015, pp. 1–8, doi: 10.1109/ETS.2015.7138763.
- [8] “OWASP Internet of Things Project - OWASP.”
https://wiki.owasp.org/index.php/OWASP_Internet_of_Things_Project#tab=IoT_To_p_10 (accessed Feb. 28, 2021).
- [9] M. A. A. Faruque, S. R. Chhetri, A. Canedo, and J. Wan, “Acoustic Side-Channel Attacks on Additive Manufacturing Systems,” in 2016 ACM/IEEE 7th International Conference on Cyber-Physical Systems (ICCPS), Apr. 2016, pp. 1–10, doi: 10.1109/ICCPS.2016.7479068.
- [10] S. Raza, ‘Lightweight Security Solutions for the Internet of Things’, PhD dissertation, Mälardalen University, Västerås, Sweden, Västerås, Sweden, 2013.

- [11] E. K. Wang, Y. Ye, X. Xu, S. M. Yiu, L. C. K. Hui, and K. P. Chow, “Security Issues and Challenges for Cyber Physical System,” in 2010 IEEE/ACM Int’l Conference on Green Computing and Communications Int’l Conference on Cyber, Physical and Social Computing, Dec. 2010, pp. 733–738, doi: 10.1109/GreenCom-CPSCom.2010.36.
- [12] Q. Jing, A. V. Vasilakos, J. Wan, J. Lu, and D. Qiu, “Security of the Internet of Things: perspectives and challenges,” *Wireless Netw*, vol. 20, no. 8, pp. 2481–2501, Nov. 2014, doi: 10.1007/s11276-014-0761-7.
- [13] T. Lu, B. Xu, X. Guo, L. Zhao, and F. Xie, “A New Multilevel Framework for Cyber-Physical System Security,” p. 2.
- [14] W. Hummer, F. Rosenberg, F. Oliveira, and T. Eilam, “Testing Idempotence for Infrastructure as Code,” in *Middleware 2013*, vol. 8275, D. Eysers and K. Schwan, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 368–388.
- [15] P. Masek, M. Stusek, J. Krejci, K. Zeman, J. Pokorny, and M. Kudlacek, “Unleashing Full Potential of Ansible Framework: University Labs Administration,” in 2018 22nd Conference of Open Innovations Association (FRUCT), Jyvaskyla, May 2018, pp. 144–150, doi: 10.23919/FRUCT.2018.8468270.
- [16] P. Webteam, “Welcome to Puppet 7.4.1.” https://puppet.com/docs/puppet/7.4/puppet_index.html (accessed Mar. 03, 2021).
- [17] “Chef Documentation.” <https://docs.chef.io/> (accessed Mar. 03, 2021).
- [18] “Ansible Documentation — Ansible Documentation.” <https://docs.ansible.com/ansible/latest/index.html> (accessed Mar. 03, 2021).
- [19] “SaltStack Documentation.” <https://docs.saltproject.io/en/latest/> (accessed Mar. 03, 2021).
- [20] K. Zhou, Taigang Liu, and Lifeng Zhou, “Industry 4.0: Towards future industrial opportunities and challenges,” in 2015 12th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD), Zhangjiajie, China, Aug. 2015, pp. 2147–2152, doi: 10.1109/FSKD.2015.7382284.
- [21] H. Xu, W. Yu, D. Griffith, and N. Golmie, “A Survey on Industrial Internet of Things: A Cyber-Physical Systems Perspective,” *IEEE Access*, vol. 6, pp. 78238–78259, 2018, doi: 10.1109/ACCESS.2018.2884906.

- [22] S. R. Chhetri, N. Rashid, S. Faezi, and M. A. Al Faruque, "Security trends and advances in manufacturing systems in the era of industry 4.0," in 2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), Irvine, CA, Nov. 2017, pp. 1039–1046, doi: 10.1109/ICCAD.2017.8203896.
- [23] Kamble, S., Gunasekaran, A. and Gawankar, S., 2020. Sustainable Industry 4.0 Framework: A Systematic Literature Review Identifying The Current Trends And Future Perspectives.
- [24] Lins, Theo & Rabelo, Ricardo & Correia, Luiz & Sá Silva, Jorge. (2018). Industry 4.0 Retrofitting. 8-15. 10.1109/SBESC.2018.00011.
- [25] Moktadir, M., Ali, S., Kusi-Sarpong, S. and Shaikh, M., 2019. Assessing Challenges For Implementing Industry 4.0: Implications For Process Safety And Environmental Protection.

APPENDICES

Appendix A: Gantt Chart



Appendix B: Security Rules List in Tested Security Profile

Rule ID	Rule Name
T_101	Secure system from mounting cramfs filesystems
T_102	Secure system from mounting freevxfs filesystems
T_103	Secure system from mounting jffs2 filesystems
T_104	Secure system from mounting hfs filesystems
T_105	Secure system from mounting hfsplus filesystems
T_106	Secure system from mounting udf filesystems
T_107	Secure system from mounting vfat filesystems
T_108	Secure /dev/shm with nodev option
T_109	Secure /dev/shm with nosuid option
T_110	Secure /dev/shm with noexec option
T_111	Secure all world-writable directories with sticky bits
T_112	Disable automounting
T_113	Install sudo
T_114	Secure sudo to run only from a psuedo-pty
T_115	Set a dedicated sudo log file
T_116	Install Advanced Intrusion Detection Environment (AIDE)
T_117	Set AIDE to run periodically
T_118	Secure permissions of bootloader
T_119	Enable address space layout randomization (ASLR)
T_120	Disable prelinking
T_121	Restrict core dumps
T_122	Install apparmor
T_123	Ensure apparmor enabled at boot time
T_124	Set all apparmor profiles to enforce state
T_125	Set message of the day banner properly
T_126	Set local login banner properly
T_127	Set network login banner properly
T_128	Secure permissions of message of the day banner

T_129	Secure permissions of local login banner
T_130	Secure permissions of network login banner
T_201	Uninstall xinetd
T_202	Uninstall openbsd-inetd
T_203	Install NTP
T_204	Enable timesync
T_205	Configure NTP
T_207	Disable Avahi server
T_208	Disable CUPS
T_209	Disable DHCP Server
T_210	Disable LDAP server
T_211	Disable NFS and RPC
T_212	Disable DNS Server
T_213	Disable FTP Server
T_214	Disable HTTP server
T_215	Disable email services
T_216	Disable HTTP Proxy Server
T_217	Disable Samba
T_218	Disable SNMP Server
T_219	Disable rsync service
T_220	Disable NIS Server
T_221	Uninstall NIS Client
T_222	Uninstall rsh client
T_223	Uninstall talk client
T_224	Uninstall telnet client
T_225	Uninstall LDAP client
T_302	Deny source routed packets
T_303	Deny ICMP redirects
T_304	Deny secure ICMP redirects
T_305	Log suspicious packets

T_306	Ignore broadcast ICMP requests
T_307	Ignore bogus ICMP responses
T_308	Enable Reverse Path Filtering
T_309	Enable TCP SYN Cookies
T_310	Deny IPv6 router advertisements
T_311	Disable Datagram Congestion Control Protocol (DCCP)
T_312	Disable Stream Control Transmission Protocol (SCTP)
T_313	Disable Reliable Datagram Sockets (RDS)
T_314	Disable Transparent Inter-Process Communication (TIPC)
T_315	Install UFW
T_316	Enable UFW service
T_401	Install auditd
T_402	Enable auditd service
T_403	Enable auditing for processes that start prior to auditd
T_404	Ensure audit_backlog_limit is sufficient
T_405	Configure audit log storage size
T_406	Configure audit rules
T_407	Install rsyslog
T_408	Enable rsyslog
T_409	Configure rsyslog default file permissions
T_410	Configure journald to send logs to rsyslog
T_411	Configure journald to compress large log files
T_412	Configure journald to write logfiles to persistent disk
T_501	Ensure cron daemon is enabled
T_502	Ensure permissions on /etc/crontab are configured
T_503	Ensure permissions on /etc/cron.hourly are configured
T_504	Ensure permissions on /etc/cron.daily are configured
T_505	Ensure permissions on /etc/cron.weekly are configured
T_506	Ensure permissions on /etc/cron.monthly are configured
T_507	Ensure permissions on /etc/cron.d are configured

T_508	Ensure at/cron is restricted to authorized users
T_509	Set minimum length of a password
T_510	Set password complexity
T_511	Use at least one digit in a password
T_512	Use at least one uppercase character in a password
T_513	Use at least one lowercase character in a password
T_514	Use at least one special character in a password
T_515	Configure number of attempts allowed before sending back a failure
T_516	Ensure password hashing algorithm is SHA-512
T_530	Configure permissions on /etc/ssh/sshd_config
T_531	Configure SSH Protocol
T_532	Configure SSH LogLevel
T_533	Disable SSH X11 forwarding
T_534	Configure SSH MaxAuthTries
T_535	Enable SSH IgnoreRhosts
T_536	Disable SSH HostbasedAuthentication
T_537	Disable SSH root login
T_538	Disable SSH PermitEmptyPasswords
T_539	Disable SSH PermitUserEnvironment
T_540	Configure SSH Idle Timeout Interval
T_541	Configure SSH Client Alive Count Max
T_542	Configure SSH LoginGraceTime
T_543	Configure SSH warning banner
T_544	Enable SSH PAM
T_545	Disable SSH AllowTcpForwarding
T_546	Configure SSH MaxStartups
T_547	Configure SSH MaxSessions
T_601	Configure permissions on /etc/passwd
T_602	Configure permissions on /etc/gshadow-
T_603	Configure permissions on /etc/shadow

T_604	Configure permissions on /etc/group
T_605	Configure permissions on /etc/password-
T_606	Configure permissions on /etc/shadow-
T_607	Configure permissions on /etc/group-
T_608	Configure permissions on /etc/gshadow