ZEROSWARM

D5.4 - Penetration and hypothesis testing diagnostic plugins

ZERO-enabling Smart networked control framework for Agile cyber physical production systems of systems

Project funded by Horizon Europe, Grant Agreement #101057083

Topic	HORIZON-CL4-2021-TWIN-TRANSITION-01-08
Project Title	ZERO-enabling Smart networked control framework for Agile cyber physical production systems of systems
Project Number	101057083
Project Acronym	Zero-SWARM
Contractual Delivery Date	M16
Actual Delivery Date	M17
Contributing WP	WP5
Project Start Date	01/06/2022
Project Duration	30 Months
Dissemination Level	Public
Editor	CERTH
Contributors	NX-SE, S21Sec

Author List

Leading Author (Editor)			
Surname	***Initials***	***Beneficiary Name***	***Contact email***
Lazaridis	GL	CERTH	glazaridis@iti.gr
Co-authors (in alphabetic order)			
Surname	***Initials***	***Beneficiary Name***	***Contact email***
Hatzidiamantis	NH	CERTH	hatzidiamantis@iti.gr
Mpatziakas	*AM*	*CERTH*	ampatziakas@iti.gr
Contributors (in alphabetic order)			
Surname	***Initials***	***Beneficiary Name***	***Contact email***
Deshmukh	SF	NE-SE	shreya.deshmukh@se.com
Borne	RB	S21Sec	rborne@s21sec.com
Egaña	JE	S21Sec	Jegana@s21sec.com
Fritz	AF	NX-SE	artur.fritz@se.com
López	OL	S21Sec	olopez@s21sec.com

Reviewers List

List of reviewers (in alphabetic order)			
Surname	***Initials***	***Beneficiary Name***	***Contact email***
Castellvi	SC	IDSA	Silvia.Castellvi@internationaldataspaces.org
Drosou	AD	CERTH	drosou@iti.gr
Khodashenas	PK	HWE	pouria.khodashenas@huawei.com
Parker	SP	ACC	stephen.parker@accelleran.com

Document History

Document History			
Version	***Date***	***Author***	***Remarks***
0.1	17/01/2023	CERTH	Table of Content
0.2	21/02/2023	All	Updates and assignment of work
0.3	10/04/2023	All	Section 3
0.4	26/06/2023	CERTH and S21Sec	Section 5
0.5	21/08/2023	CERTH	Section 6
0.6	25/09/2023	All	Sections 1, 2, 4, 7, 8
0.7	11/10/2023	CERTH	Sent for internal review
0.8	17/10/2023	All	Addressing the review comments
1.0	18/10/2023	CERTH	Final submission

DISCLAIMER OF WARRANTIES

This document has been prepared by Zero-SWARM project partners as an account of work carried out within the framework of the contract no 101057083.

Neither the Project Coordinator, nor any signatory party of the Zero-SWARM Project Consortium Agreement, nor any person acting on behalf of any of them:

- makes any warranty or representation whatsoever, express or implied,
    - concerning the use of any information, apparatus, method, process, or similar item disclosed in this document, including merchantability and fitness for a particular purpose or
    - that such use does not infringe on or interfere with privately owned rights, including any party's intellectual property
- that this document is suitable to any particular user's circumstance; or
- assumes responsibility for any damages or other liability whatsoever (including any consequential damages, even if the Project Coordinator or any representative of a signatory party of the Zero-SWARM Project Consortium Agreement, has been advised of the possibility of such damages) resulting from your selection or use of this document or any information, apparatus, method, process, or similar item disclosed in this document.

Zero-SWARM has received funding from the European Union's Horizon Europe research and innovation programme under grant agreement No 101057083. The content of this deliverable does not reflect the official opinion of the European Union. Responsibility for the information and views expressed in the deliverable lies entirely with the author(s).

Table of Contents

List of Figures

List of Tables

List of Acronyms

Abbreviation	Definition
AAS	Asset Administration Shell
AI	Artificial Intelligence
BSIMM	Building Security In Maturity Model
CAT	Composite Automation Type
CPS	Cyber Physical System
CPSoS	Cyber Physical System of Systems
CPU	Central Processing Unit
CVSS	Common Vulnerability Scoring System
D	Deliverable
DoS	Denial of Service
FB	Function Block
GA	Grant Agreement
GA	Genetic Algorithm
HMI	Human Machine Interface
IDE	Integrated Development Environment
IDS	Intrusion Detection System
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronics Engineers
IIoT	Industrial Internet of Things
IoT	Internet of Things
IP	Internet Protocol
IPS	Intrusion Prevention System
ISECOM	Institute for Security and Open Methodologies
ISSAF	Information System Security Assessment Framework
IT	Information Technology
KPI	Key Performance Indicator
LFA	Link Flooding Attack
M	Month
ML	Machine Learning
MQTT	Message Queuing Telemetry Transport
NAT	Network Address Translation
OISSG	Open Information Systems Security Group

OPC-UA	Open Platform Communications United Architecture
OS	Operating System
OSSTMM	Open-Source Security Testing Methodology Manual
OT	Operational Technology
OWASP	Open Web Application Security Project
PLC	Programmable Logic Controller
PT	Penetration Testing
PTES	Penetration Testing Execution Standard
QoS	Quality of Service
RTU	Remote Terminal Unit
SA	Static Analysis
SCADA	Supervisory Control And Data Acquisition
SFB	Service Interface Function Block
SUT	System Under Test
T	Task
TLS	Transport Layer Security
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
VM	Virtual Machine
VPN	Virtual Private Network
WP	Work Package

Executive Summary

In the context of Zero-SWARM, deliverable D5.4 - Penetration and hypothesis testing diagnostic plugins v1, is the first technical cybersecurity document of a series of four deliverables. This deliverable focuses on the cybersecurity aspects of Cyber Physical Systems (CPS), introducing penetration testing as a means of securing such industrial systems. We first try to connect these cybersecurity activities with the general architectural overview of Zero-SWARM (D2.2), focusing only on the automation architecture presented in D5.1 and extending it to showcase clearly the contribution of this task T5.4. Besides the aforementioned connections to these deliverables, there is also a connection to D2.3, where the cybersecurity activities of penetration testing and hypothesis testing should follow specific guidelines. Nevertheless, it is worth mentioning that D5.4 is also correlated to D5.5 regarding the anomaly detection and countermeasure selection modules and how all these cybersecurity modules exchange information between them.

The main activities of penetration testing include the examination of communication protocols, such as OPC-UA, MQTT and Modbus and the development of a module capable of testing these protocols. Moreover, research has been conducted in order to try to automate the penetration testing processes and make them more user-friendly, allowing any IT personnel to conduct such tests. To validate the developed penetration testing module, two initial stage simulation testbeds were deployed, one regarding the OPC-UA communications while the other was using the Modbus TCP communication protocol to exchange data. Several tests were performed and the results together with the deployment are reported in this deliverable. Besides these two testbeds, an IEC61499 simulation testbed was also developed, giving the opportunity to researchers to assess the industrial communication protocols and processes in a more realistic way.

On the other hand, this task also includes the development and validation activities of the hypothesis testing module. In the context of D5.4 some initial work has been conducted and the development of this module has advanced and has been reported in this document. The final development and validation of this module, in the context of T5.4, will be reported in the next version of this deliverable, namely D5.9.

1 Introduction

Within the domain of Cyber Physical Systems of Systems (CPSoS), the relentless pursuit of robust cybersecurity is elevated as a paramount and overarching objective. This first version of D5.4 serves as a comprehensive blueprint, outlining the multifaceted purpose and strategic roadmap envisioned to bolster cybersecurity within the CPSoS ecosystem. At its core, this attempt is driven by the imperative to enhance cybersecurity awareness and decision-making capabilities. The primary mission at hand is the implementation of diagnostic plugins, namely the penetration testing and the hypothesis testing modules, which will function as the linchpin in achieving these goals. These modules are not only poised to become integral components of the ambitious Zero-SWARM project. Still, they are also set to play an important role in executing meticulous security risk assessments and advanced threat modelling for the entire deployed CPSoS infrastructure. As a cornerstone of this initiative, adherence to the globally recognized IEC-62443 security standard, specifically tailored to industrial automation and control systems, underscores the commitment to ensuring the highest levels of security, including elements like confidentiality.

1.1 Purpose of the document

The purpose of this deliverable is to outline the comprehensive objectives and strategies aimed at enhancing cybersecurity within the context of the CPSoS. The primary goal of this endeavour is the development and integration of diagnostic plugins that will significantly augment cybersecurity awareness and decision-making processes within the CPSoS framework. These diagnostic plugins will play a crucial role in the realization of the Zero-SWARM initiative, focusing on conducting thorough security risk assessments and robust threat modelling for the deployed CPSoS. One fundamental aspect of this document is the commitment to adhere to the IEC-62443 security standard, a recognized benchmark for security risk assessment within industrial automation and control systems. Under Task T5.4, the project will prioritize specific security properties, such as confidentiality, to ensure that the CPSoS remains resilient against potential threats and vulnerabilities.

Moreover, a critical component of this task will involve the development of a conformance testing tool. This tool is designed to rigorously validate the stability and communication aspects of a secure CPSoS architecture from a design perspective. Furthermore, the project will employ penetration testing techniques to assess and verify the security configuration of the CPSoS, aiming to identify any potential vulnerabilities that could pose a risk to its integrity, confidentiality, and availability. These penetration tests will go beyond existing third-party frameworks, extending their capabilities to include network discovery and vulnerability exploitation functionalities tailored to the unique characteristics of industrial systems, such as OPC-UA and Modbus for communication protocols, MQTT for data gathering and cloud services.

The outcomes of these penetration tests will serve as a basis for the determination of appropriate mitigation actions. These actions will be carefully crafted to ensure that they do not compromise the integrity, confidentiality, or availability of the critical industrial processes underpinning the CPSoS. Furthermore, this document underscores the importance of a proactive approach in addressing security threats and risks faced by the CPSoS. It highlights the significance of utilizing data obtained through OPC-UA and Asset Administration Shell (AAS) to implement a Hypothesis Testing tool. This tool will empower system operators to assess how the application of various countermeasures can impact the overall security and functionality of the CPSoS.

In essence, this document encapsulates the overarching objectives of this project, which are geared

towards fortifying the cybersecurity posture of the CPSoS, safeguarding its critical components and enabling resilient decision-making processes in the face of evolving security threats and challenges.

1.2 Structure of the document

The document is structured as follows:

- **Chapter 1** is an introduction to the whole document, describing its scope and purpose, its structure, and the delivery plan during the project's lifetime, as well as outlining the task's objectives;
- **Chapter 2** provides the connection of deliverable D5.4 to other deliverables of the project;
- **Chapter 3** gives some introductory information regarding the two modules that this task should deliver, namely the penetration testing and the hypothesis testing modules;
- **Chapter 4** presents a security analysis of the Cyber Physical System of Systems (CPSoS);
- **Chapter 5** outlines the penetration testing modules focusing on two different protocols (OPC-UA, Modbus) and presenting the early-stage simulation testbeds, with the help of which the penetration testing modules will be validated;
- **Chapter 6** presents the initial idea and methodology for the development of the hypothesis testing module;
- **Chapter 7** introduces the IEC61499 simulation platform that will be used in the second stage of the task to validate the modules in a more realistic environment, closer to an industrial production line;
- **Chapter 8** will conclude the work of this deliverable, provide some insight regarding the next steps of this task and will comment on the task's activities.

1.3 Deliverable plan along the Zero-SWARM project and objectives

Based on the project's Gantt chart, task T5.4 activities started on M07 of the project. The first deliverable of this task D5.4 - Penetration and hypothesis testing diagnostic plugins v1 was planned to be delivered on M16. Due to the interconnection of this cybersecurity related document to deliverables D2.2 and D2.3, whose revisions will be submitted on M17, a delay of about fifteen days was requested in order to synchronize the conducted work. Moreover, even though it will be mentioned in section 5, part of the work related to penetration testing was submitted and accepted for presentation at the IEEE Conference on Standards for Communications and Networking after a review process. Due to all the above-mentioned reasons and without derailing any other activities, this deliverable, D5.4 will be officially submitted with a small delay.

In this first phase of task T5.4, partners were able to develop two early-stage simulation testbeds, one simulating industrial OPC-UA communications and the other simulating Modbus TCP communications, which will be described in this document. Based on these testbeds, we were able to conduct our initial penetration tests in order to explore and get to know these protocols better. To this end, the penetration testing methodology for industrial scenarios proved to be different from the classic methodology someone could use for performing penetration testing towards a web server. Moreover, in the same context, an IEC61499 simulation platform was developed, which will allow us to explore in a more realistic way and close to a real-life industrial production line, our penetration testing modules. All tests related to this simulation platform will be reported in the next version of D5.4, namely D5.9,

along with any activities carried out in the second phase of T5.4, towards M24.

On the other hand, some initial work and development on the hypothesis testing module has already been carried out and will be reported in section 6. In the second phase of T5.4 and in the context of the deliverable D5.9, all advancements and tests related to hypothesis testing will also be reported.

All in all, based on the Grant Agreement (GA) [37], task T5.4 aims to fulfill the following objectives:

- To implement diagnostic plugins that will provide cybersecurity awareness and decision making in CPSoS
- To perform security risk assessment & threat modelling of the deployed CPSoS
- To validate the stability and communication of a secure CPSoS architecture from a design perspective
- To identify possible vulnerabilities of the main technologies & devices deployed in Zero-SWARM and trials
- To implement pen tests for verification of the secure configuration of the deployed CPSoS
- To allow the system operator to examine the effect of applying different countermeasures for the CPSoS vulnerabilities identified

To a certain point, the above-mentioned objectives are in the process of being validated in this first phase of the project, as described in this document, but will be totally fulfilled by the end of this task. As someone can understand from the objectives, the majority of the work has been concentrated on the penetration testing activities, but, of course, the hypothesis testing activities have not been underestimated.

2 Connection with the relative tasks and deliverables

Deliverable D5.4 - Penetration and hypothesis testing diagnostic plugins is the first technical cybersecurity document of a series of four deliverables, in the scope of tasks T5.4 and T5.5. Due to the nature of this task, the cybersecurity activities are horizontally placed within the project, covering the cybersecurity aspects of nearly the whole project. For this reason, D5.4 is directly connected with a series of other deliverables of the project. More specifically, there is a connection among D5.4 and D2.2, D2.3, D5.1 and D5.5. The following sections present briefly the interconnection between these deliverables.

2.1 Connection to Zero-SWARM architecture, T5.1 and use-cases

The figure below illustrates a zoomed in architecture of the overall CPS structure, following the three-layer architecture as presented in the official Zero-SWARM deployment view architecture in D2.2 - Eco designed architecture, specifications & benchmarking [39]. The architecture presented in this section was thoroughly analyzed in D5.1 - Distributed automation and information management [41], showcasing all three layers but also extending the architecture in order to represent the IEC 61499 automation platform (simulation platform), which will be a starting point for the evaluation of the tools developed in the context of task T5.4. More specifically, the architecture depicts the different communication protocols that will be used for data exchange across the three layers, namely MQTT, OPC-UA, Modbus and IEC61499. These protocols will be examined from a cybersecurity perspective to understand if they have any vulnerabilities that could jeopardize an industrial production line, if compromised. Furthermore, besides any network vulnerabilities, the simulation platform will be assessed regarding its applications, its physical security, human factors, etc., as described in section

3.1.4. In the scope of WP6, the penetration testing and hypothesis testing modules are planned to be validated in the Node scenarios, taking advantage of the already gained knowledge over the testing of the modules with the IEC61499 simulation platform.

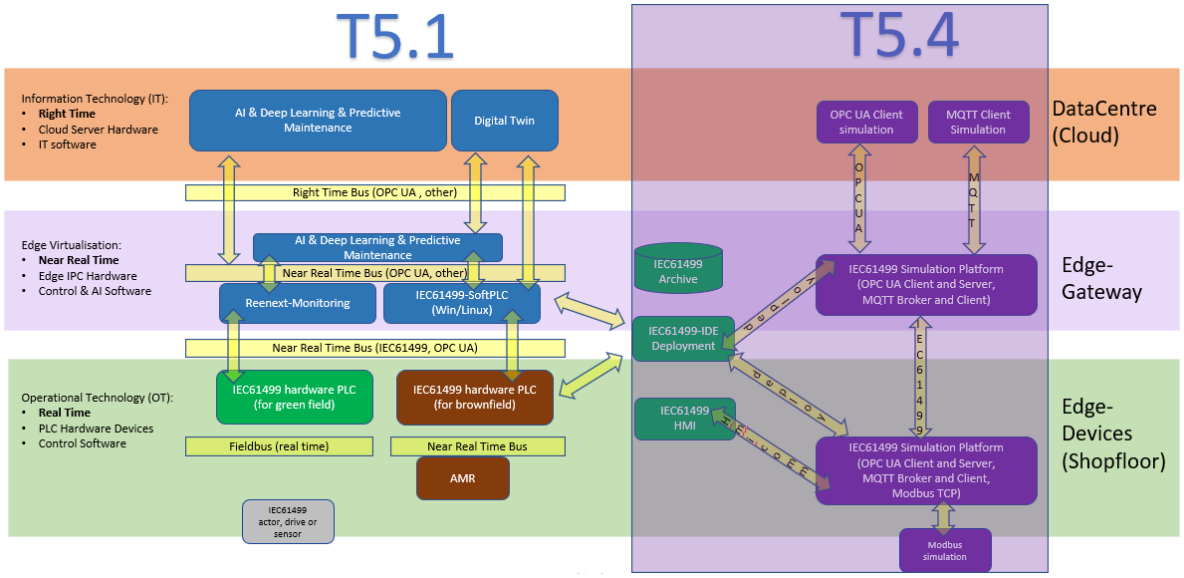

Figure 1: Zero-SWARM zoomed in CPS structure and connection to T5.4

2.2 Connection to anomaly detection and countermeasure selection modules

This section describes the interconnection between the two cybersecurity tasks of the Zero-SWARM project, namely T5.4 – Ad-Hoc penetration and hypothesis testing plugins and T5.5 - Anomaly detection and countermeasure selection modules. The following figure depicts a unified high-level workflow, showcasing how all four modules of these cybersecurity tasks (T5.4 – T5.5) interact with each other. More specifically, the penetration testing modules (1), having detected vulnerabilities and determined the attack surface of a specific CPS, will inform the anomaly detection module (2) about possible attack types and vulnerabilities. The next step, the anomaly detection module, is responsible for detecting known attacks or anomalous traffic. This information will be fed in the countermeasure selection module (3), informing it about ongoing attacks and anomalies. Then, the countermeasure selection module will decide the best strategy to handle attacks and anomalies, based on a predefined list of actions, regarding the software or hardware components attacked. The last step includes the update of the hypothesis testing module (4), with the output of the countermeasure selection module. In other words, the aforementioned module will inform the hypothesis testing module about the decided strategy and the predefined list of actions based on the component attacked. Last but not least, the hypothesis testing module will compare different mitigation strategies based on static KPIs and statistical difference, allowing the system operator to examine how the application of different countermeasures will affect the CPSoS. More information on the anomaly detection and countermeasure selection modules will be presented thoroughly in D5.5 - Anomaly detection and countermeasure selection tools.

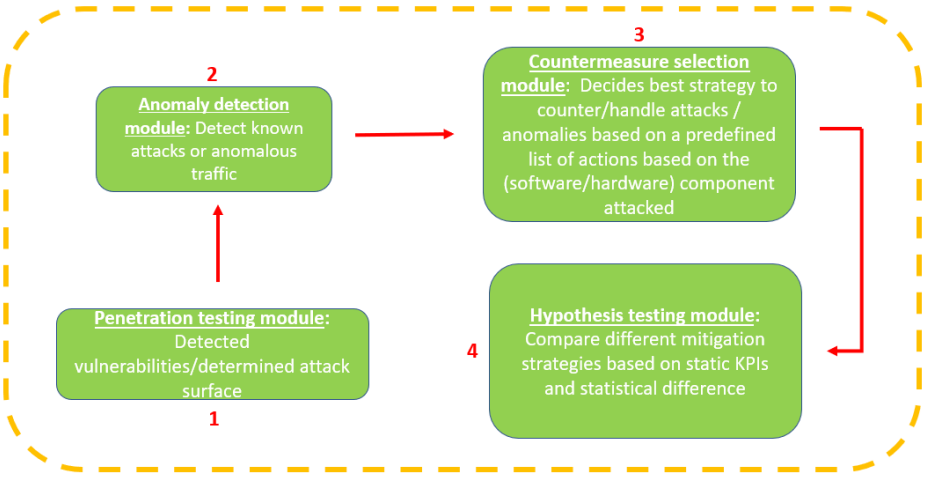

Figure 2: Unified high-level T5.4 - T5.5 interconnections

3 Introduction to penetration and hypothesis testing

3.1 Penetration testing

3.1.1 Penetration testing background

Penetration Testing (PT) has emerged as a fundamental practice within the realm of cybersecurity over the past decade. It entails the deliberate and controlled execution of an authentic attack on a digital asset, which may encompass computer systems, industrial systems, Internet of Things (IoT) devices, software applications or networks, with the primary objective of scrutinizing their security posture. The process of PT is meticulously structured into a series of sequential tasks, enabling a systematic and comprehensive evaluation of the target system's security. This often includes the proactive identification of vulnerabilities and the execution of a predetermined set of actions to assess the potential for compromise by employing exploits against these identified vulnerabilities. In practice, PT generally includes six process phases, which are depicted in Figure 3, but they can also vary from case to case. The pre-engagement interactions phase sets the foundation for a successful penetration test by ensuring that both the client and the testing team have a clear understanding of the scope, objectives, and rules governing the engagement. It helps establish trust and transparency between all parties involved, which is essential for a smooth and effective penetration testing process [27][28][29].

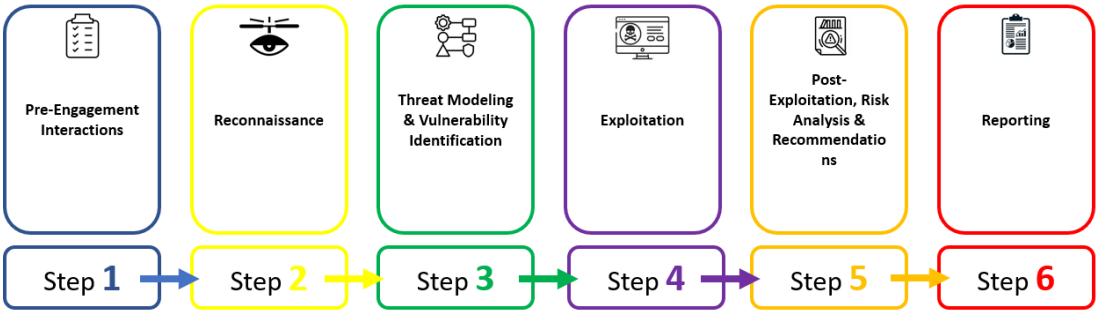

Figure 3: Penetration testing process phases

3.1.1.1 Step 1: Pre-Engagement Interaction

This first pre-phase called pre-engagement interactions in penetration testing is a crucial initial stage in the overall penetration testing process. It involves a series of activities and communications between the penetration testing team and the client or organization that has requested the

penetration tests. Below, some key aspects and considerations can be found, related to this pre-phase:

- **Client engagement**: The process typically begins with the client or organization expressing their intent to conduct a penetration test. This could be driven by regulatory requirements, security concerns or a proactive effort to assess and improve their security posture.
- **Scope definition**: The penetration testing team works closely with the client to define the scope of the engagement. This involves identifying the specific systems, networks, applications, or assets that will be tested. It is crucial to establish clear boundaries to avoid unintended consequences or disruptions during the testing.
- **Rules of engagement**: During this phase, the rules of engagement are established. These rules outline what is allowed and what is not allowed during the penetration test. For instance, the team may agree on whether they can attempt to exploit vulnerabilities, use social engineering techniques, or simulate insider threats.
- **Legal and Compliance Considerations**: Ensuring that the penetration test complies with all relevant laws and regulations is essential. Pre-Engagement Interactions may involve legal reviews and documentation to protect both the client and the testing team from legal repercussions.
- **Communication protocols**: Establishing clear lines of communication is vital. The client and the testing team need to agree on how and when they will exchange information, updates, and findings throughout the engagement.
- **Scheduling and timing**: The timing of the penetration test is determined during this phase. It is essential to coordinate the testing schedule with the client to minimize disruptions to their operations.
- **Resource and personnel allocation**: Identifying the resources and personnel needed for the engagement is part of pre-engagement planning. This includes specifying who from the client's side will be involved, as well as assembling the penetration testing team with the required skills and expertise.
- **Documentation**: Detailed documentation is essential. This includes creating a formal penetration testing agreement or contract that outlines all the terms and conditions, as well as objectives and expectations for the engagement.
- **Preparation**: Before the actual testing begins, both the client and the testing team need to ensure that all necessary preparations are in place. This may involve setting up test environments, acquiring any required tools or resources, and conducting any training or briefings for the personnel involved.

3.1.1.2 Step 2: Reconnaissance

The reconnaissance phase is one of the critical stages in the process of penetration testing. It involves gathering information about the target system, network or organization to comprehensively understand the environment before initiating any active attacks. This phase is often considered the first step in the penetration testing process and is primarily focused on passive information gathering. This step 2 phase provides penetration testers with a solid foundation of knowledge about the target environment, which they can use to plan and execute subsequent phases of testing. By identifying potential vulnerabilities and weaknesses early in the process, penetration testers can help organizations strengthen their security defenses and mitigate potential risks. Certain key aspects of the reconnaissance phase are listed below:

- **Passive information gathering**: Reconnaissance is primarily a passive phase, meaning that the penetration testing team collects information without directly interacting with the target system. This helps maintain stealth and avoid triggering alarms or security measures.
- **Objectives**: The primary objectives of the reconnaissance phase include identifying potential targets, understanding the target's infrastructure, discovering vulnerabilities, and gathering intelligence about the target organization's security posture.
- **Information sources**: Information is collected from various sources, including public records, online databases, social media, domain registration records, search engines, and public websites. The goal is to find publicly available information that could be useful for identifying potential weaknesses.
- **Types of reconnaissance**:
    - **Passive reconnaissance**: This involves collecting information that is publicly available or can be obtained without directly interacting with the target. This might include identifying IP addresses, domain names, email addresses, and employee names.
    - **Active reconnaissance**: While not as discreet as passive reconnaissance, active reconnaissance involves scanning and probing the target's network to discover open ports, services, and potentially vulnerable systems. It is a more intrusive phase and can sometimes be detected by network security tools.
- **Tools and techniques**: Penetration testers often use a variety of tools and techniques for reconnaissance, such as network scanning tools (e.g., Nmap), domain name lookup tools (e.g., WHOIS), search engines, social engineering tactics (e.g., phishing for information), and online forums or communities where information about the target organization may be discussed.
- **Information collected**: During reconnaissance, testers aim to gather information about the target's IP addresses, network architecture, domain names, subdomains, email addresses, employee names and roles, software and hardware used, and potentially any known vulnerabilities associated with the identified systems.
- **Documentation**: Thorough documentation of all collected information is crucial. This documentation will serve as the foundation for subsequent phases of penetration testing, helping testers make informed decisions about attack vectors and potential vulnerabilities.
- **Ethical considerations**: Reconnaissance activities must always be conducted ethically and within the bounds of any legal agreements or permissions obtained from the client. Engaging in unauthorized or malicious information gathering is not only unethical but also illegal.

3.1.1.3 Step 3: Threat Modelling & Vulnerability Identification

The threat modelling and vulnerability identification phase is a significant step in the overall process, as well. It involves systematic analysis and assessment of the target system, network or application to identify potential threats, vulnerabilities and weaknesses. This phase is essential for understanding the security landscape of the target and determining where attacks might be most effective. These are the main characteristics associated with this step:

- **Objective**: The primary objective of the Threat Modeling & Vulnerability Identification phase is to identify and document potential vulnerabilities and threats that attackers could exploit. This includes both known vulnerabilities and those that may not have been previously recognized.
- **Methodical approach**: Penetration testers use a systematic and structured approach to assess the target. This often involves reviewing architecture diagrams, network configurations,

application source code (if available), and other relevant documentation to understand how the target is designed and how it functions.

- **Threat modeling**: Threat modelling is a technique used to identify potential threats and attack vectors. It involves thinking like an attacker and considering various ways in which the target could be compromised. Threat modeling helps testers prioritize their efforts by focusing on the most critical assets and potential threats.
- **Asset identification**: In this phase, penetration testers identify and classify the assets within the target environment. Assets can include data, systems, applications, hardware, and any other components that are essential to the organization.
- **Vulnerability assessment**: Testers actively search for vulnerabilities within the target. This may involve scanning for open ports, services, and known vulnerabilities using tools like vulnerability scanners or manual testing techniques. Vulnerability assessment may also include code review for software applications.
- **Risk assessment**: Once vulnerabilities are identified, testers assess their potential impact and likelihood of exploitation. This helps prioritize which vulnerabilities should be addressed first, considering the potential consequences of a successful attack. Common scales used for risk assessment include CVSS (Common Vulnerability Scoring System) or similar frameworks.
- **Documentation**: Comprehensive documentation is an essential part of this phase. Testers create a detailed list of identified vulnerabilities, their descriptions, their locations and the potential risks associated with each asset. This documentation serves as a basis for the subsequent exploitation and post-exploitation phases.
- **Reporting**: After identifying and assessing vulnerabilities, penetration testers typically create a formal report for the client. This report includes a summary of findings, detailed descriptions of vulnerabilities, their potential impact on the organization, and recommendations for mitigation.

3.1.1.4 Step 4: Exploitation

The exploitation phase of penetration testing represents an important step within the assessment process. This phase is characterized by active attempts made by the penetration tester to exploit identified vulnerabilities within the target system, network or application to be checked. The primary objective of this step is to validate the existence and potential impact of the vulnerabilities identified in the previous step, simulating real-world attacks to determine whether a malicious user could successfully compromise the target. The most important characteristics of this step of penetration testing are listed below:

- **Systematic approach**: Penetration testers follow a methodical and well-planned approach during exploitation. They carefully strategize and execute attacks while adhering to predefined rules of engagement and the scope outlined with the client.
- **Active engagement**: Unlike the preceding reconnaissance and vulnerability identification phases, the Exploitation phase involves active engagement with the target. Testers actively seek out weaknesses and security flaws and attempt to leverage them to gain unauthorized access or control.
- **Tools and techniques**: A variety of tools, scripts, and techniques are employed by penetration testers during exploitation. These tools are often selected based on the specific vulnerabilities and attack vectors identified earlier in the assessment process.

- **Privilege escalation**: Initial exploitation efforts may grant testers limited access to the target. During this phase, they may employ privilege escalation techniques to gain higher levels of control over the system, increasing the potential impact of their actions.
- **Documentation**: Detailed documentation is an integral part of the exploitation phase. Testers meticulously record every step of the attack process, including the tools used, commands executed and outcomes achieved. This documentation serves as critical evidence and supports comprehensive reporting to the client.
- **Client communication**: Effective communication with the client may continue during this phase to provide progress updates and seek guidance in case of unexpected challenges. Maintaining transparency fosters trust between the testing team and the client.

3.1.1.5 Step 5: Post-Exploitation, Risk Analysis & Recommendations

The post-exploitation, risk analysis and recommendations stage of the penetration testing process follows the exploitation phase. It involves the systematic analysis of the impact of successful exploitation, assessment of risks and the formulation of actionable recommendations to improve security. The following listed items depict the key aspects of this stage:

- **Post-exploitation activities**: After successfully exploiting vulnerabilities in the target system, the penetration tester may engage in post-exploitation activities. These can include maintaining control over compromised systems, escalating privileges, exfiltrating sensitive data, or exploring the network further. Post-exploitation activities help simulate how an attacker might continue to exploit a compromised system.
- **Risk analysis**: Once post-exploitation activities are completed, the penetration tester conducts a thorough risk analysis. This analysis evaluates the potential consequences of the successful exploitation, considering factors such as data exposure, system compromise, business disruption, and regulatory compliance violations. The goal is to understand the true impact of the vulnerabilities and the potential risks to the organization.
- **Recommendations**: Based on the findings from the risk analysis, the penetration tester formulates a set of actionable recommendations. These recommendations are designed to help the client mitigate the identified vulnerabilities, improve security controls, and enhance the overall security posture. Recommendations may include software patching, reconfiguration of security settings, strengthening access controls, or employee training.
- **Prioritization**: Recommendations are typically prioritized based on the severity of the vulnerabilities and the potential impact on the organization. High-risk vulnerabilities that could lead to severe consequences are often addressed first. This prioritization allows the client to allocate resources efficiently to mitigate the most critical issues.
- **Documentation**: Detailed documentation of post-exploitation activities, risk analysis and recommendations, is crucial. The penetration tester provides a comprehensive report that includes a summary of findings, descriptions of vulnerabilities, their potential impact, and the recommended
- **Mitigation plan**: Following the consultation, the client develops a mitigation plan based on the recommendations provided by the penetration tester. This plan outlines the specific steps and timelines for addressing the identified vulnerabilities and improving security controls.
- **Reassessment**: In some cases, the client may request a follow-up penetration test to verify that the recommended security improvements have been effectively implemented and the vulnerabilities have been remediated.

3.1.1.6 Step 6: Reporting

The reporting stage is the final step of penetration testing, where the results and findings of the assessment are documented and communicated to the relevant stakeholders. This step is essential for ensuring that vulnerabilities and security weaknesses are properly addressed and that the organization can improve its security posture. Some of the main elements of the reporting phase in the context of penetration testing are listed below:

- **Report compilation**: Penetration testers compile all the data, evidence, and findings collected during the testing phase into a comprehensive report. This report typically includes detailed information about the vulnerabilities discovered, their severity, and how they can be exploited.
- **Executive Summary**: An executive summary is often included at the beginning of the report. It provides a high-level overview of the penetration test's objectives, methodology, key findings and recommendations. This section is designed for non-technical stakeholders, such as executives and managers.
- **Technical details**: The main body of the report contains detailed technical information about each vulnerability or security issue identified. This includes the affected systems, potential impact, and a step-by-step explanation of how the penetration tester was able to exploit the vulnerability.
- **Recommendations**: Penetration testers provide actionable recommendations for mitigating the identified vulnerabilities and improving security. These recommendations are often prioritized based on risk and potential impact to help organizations address the most critical issues first.
- **Remediation guidance**: In addition to recommendations, penetration testers may offer advice on how to remediate the vulnerabilities. This guidance may include technical steps, best practices, or suggested configurations to improve security.
- **Evidence & documentation**: The report should include evidence to support the findings, such as screenshots, logs and any other relevant documentation. This helps ensure the credibility of the report and provides a clear record of the vulnerabilities.
- **Reporting format**: The format of the report may vary depending on the organization's preferences and requirements. It can be presented in a written document, a presentation, or a combination of both. Some organizations also request an oral presentation of the findings.
- **Delivery & discussion**: The penetration testing team typically delivers the report to the client or relevant stakeholders. It's important to schedule a discussion or debriefing session to ensure that all findings and recommendations are well understood and can be acted upon effectively.
- **Follow-up**: After the report is delivered, the organization should follow up on the recommendations and begin the process of remediating the identified vulnerabilities. This may involve further testing to validate that the fixes were effective.

3.1.2 Penetration Testing Approaches

This section gives a high-level overview of the three penetration testing approaches, as depicted in the figure below, which are used to analyze and attack potential assets. This categorization tends to be theoretical considering that the actual circumstances are quite different and may demand a combination of the approaches listed below (Figure 4), depending on the complexity of the situation and the client's requirements. Furthermore, the type and scope of testing tend to be established and accepted before the launch of any test and could be revised or extended throughout the Zero-SWARM

project's lifetime.

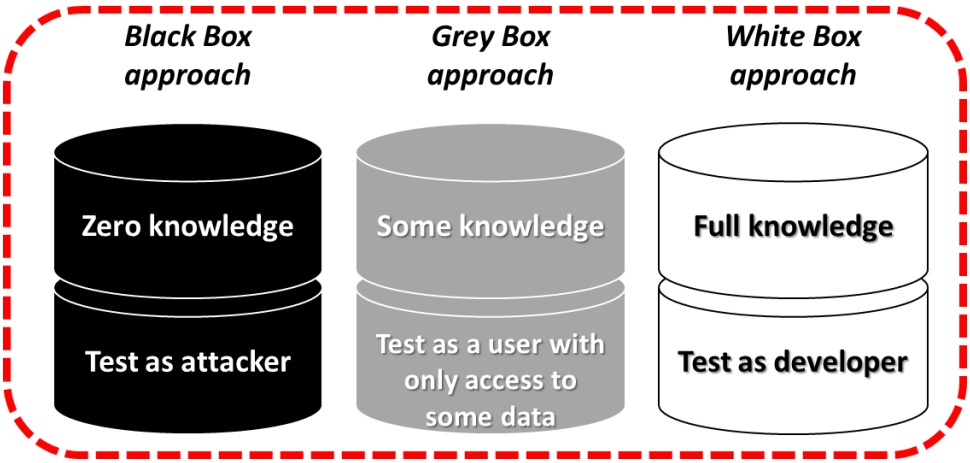

Figure 4: Differences between the three PT approaches

3.1.2.1 Black-box approach

Black box penetration testing is a cybersecurity testing approach where the penetration tester has no prior knowledge about the target system or network being assessed. This method simulates the perspective of an external attacker who has no insider information about the organization's infrastructure.

3.1.2.2 White-box approach

White-box penetration testing, also known as clear box testing or transparent box testing, is a cybersecurity assessment approach where the tester has full knowledge of the target system or network being evaluated. In contrast to black box testing, where the tester has no prior information, white box testing provides the tester with complete access to system documentation, source code, network diagrams and other relevant details. This approach is mostly used by developers who are responsible of assessing a system at the time it is being developed in order to provide a secure-by-design asset. The white-box approach is the most advanced among the three approaches stated in this section, namely the black-box, white-box and grey-box approaches.

3.1.2.3 Grey-box approach

The grey box penetration testing approach falls between black box and white box testing in terms of information disclosure. In grey box testing, the penetration tester has some limited knowledge about the target system or network, but not as much as in white box testing. This approach is designed to simulate the perspective of a semi-informed attacker who may have partial insider information.

3.1.3 Penetration Testing frameworks and methodologies

Penetration testing frameworks and methodologies offer a structured framework for the preparation, execution and documentation of cybersecurity vulnerability assessments. These resources encompass a range of activities aimed at establishing robust security methodologies. Below are some well-known penetration testing frameworks and standards, based on literature [52]:

- Open Worldwide Application Security Project (OWASP)
- Information System Security Assessment Framework (ISSAF)
- Open-Source Security Testing Methodology Manual (OSSTMM)
- Building Security In Maturity Model (BSIMM)
- Penetration Testing Execution Standard (PTES)
- Metasploit

3.1.3.1 OWASP

The Open Web Application Security Project, commonly known as OWASP, is a globally recognized non-profit organization dedicated to improving the security of web applications and software. Founded in 2001, OWASP has become a leading authority in the field of application security, offering a wealth of resources, best practices, and tools to assist organizations in identifying and mitigating security vulnerabilities in their web applications. Its primary mission is to make web applications and software more secure by raising awareness about security risks and providing guidance on best practices. The organization operates under the principles of openness, community collaboration and vendor neutrality. Moreover, OWASP's contributions have had a significant impact on the field of cybersecurity. Its resources and guidelines are widely used by organizations to secure their web applications and its annual OWASP Top Ten list serves as a reference for identifying and addressing the most critical security risks. Concluding, OWASP continues to empower organizations and individuals to build and maintain secure web applications in an ever-evolving threat landscape [30][31].

3.1.3.2 ISSAF

The Information System Security Assessment Framework (ISSAF) is a concise and thoroughly evaluated penetration testing guide, supported by the Open Information Systems Security Group (OISSG). Even though this methodology is not updated anymore, it is used, nowadays, for its comprehensive nature. More specifically, it separates information system security assessments into categories and specifies particular requirements for evaluating and validating each of these categories. The main objective is to deliver useful information into evaluations of security, which correspond to real-world events. ISSAF is a core tool for satisfying an organization's security evaluation requirements and may additionally be utilized as a reference for handling a variety of other information security concerns.

3.1.3.3 OSSTMM

OSSTMM stands for Open-Source Security Testing Methodology Manual. It is a peer-reviewed methodology for security testing, maintained by the Institute for Security and Open Methodologies (ISECOM). The OSSTMM is a comprehensive methodology that covers all aspects of security testing, from planning and scoping to reporting and remediation. It is divided into five channels, as listed below:

- **Physical security**: This channel covers the physical security of an organization, such as its buildings, perimeter, and assets.
- **Wireless communications**: This channel covers the security of wireless networks, such as Wi-Fi and Bluetooth.
- **Telecommunications**: This channel covers the security of telecommunications systems, such as phone networks and data networks.
- **Data networks**: This channel covers the security of data networks, such as LANs and WANs.
- **Human factors**: This channel covers the security of human factors, such as social engineering and phishing.

3.1.3.4 BSIMM

BSIMM stands for Building Security In Maturity Model and is a data-driven model that provides a baseline of observed activities for software security initiatives. The BSIMM is based on a survey of over 200 organizations and identifies 12 practices that are common to successful software security initiatives. The BSIMM can be used by organizations to assess their current software security posture and to identify areas for improvement. It can also be used to benchmark an organization's software security program against other organizations. These practices are organized into four domains:

- **Governance**: Practices that help organize, manage, and measure a software security initiative.
- **Intelligence**: Practices that help organizations gather and analyze security intelligence.
- **Secure Software Development Lifecycle (SSDL) touchpoints**: Practices that are implemented at different stages of the SDLC to improve security.
- **Deployment**: Practices that are implemented to secure software after it is deployed.

3.1.3.5 PTES

A group of information security specialists from many different industries developed and continue to update the Penetration Testing Execution Standard, also known simply as PTES. PTES defines a minimum standard for a penetration test, spanning from the initial client-tester negotiation to the contents of the report. PTES aims to raise the threshold for penetration testing quality by offering exceptional levels of guidance. Organizations have a better understanding of the services they are paying for because of to the standardization of penetration testing guidelines, which also provides penetration testers with precise instructions concerning what they are supposed to do during a penetration test.

3.1.3.6 Metasploit

Metasploit is a penetration testing framework designed to help security professionals find and exploit vulnerabilities in computer systems, networks and IoT devices. It is a powerful tool that can be used to simulate real-world attacks, helping organizations to identify and mitigate security risks. The Metasploit framework is modular, meaning that it is made up of a collection of individual tools that can be combined to create custom attacks. This makes it a very flexible tool that can be used to test a wide range of vulnerabilities. In 2003, Metasploit was officially launched as an open-source project but was acquired in 2009 by Rapid7, a company that is now responsible for its development and support. It is regularly updated with new modules and features, ensuring that it remains up-to-date with the latest security threats.

3.1.4 Overview of CPSoS penetration testing

Cyber-Physical Systems of Systems (CPSoS) represent complex interconnected systems that combine cyber elements (software, networks, data) with physical components (sensors, actuators, hardware). Ensuring the security of CPSoS is critical due to their real-world impact on industries like industrial automation, transportation and critical infrastructure. Penetration testing for CPSoS requires specialized approaches to address the unique challenges they pose. As a proposed overview of different penetration testing approaches for CPSoS based on [7][8][9][11] different phases of testing can be considered.

- **Network Penetration Testing**:
  - Focuses on the communication networks within the CPSoS
  - Identifies vulnerabilities in network protocols, routers, switches, firewalls, and other network components
  - Tests for unauthorized access, data leakage, and potential network-level attacks
- **Embedded System Penetration Testing**:
  - Targets the physical devices and components embedded in CPSoS
  - Aims to identify vulnerabilities in firmware, hardware, and communication interfaces of sensors, actuators, and controllers
  - Evaluates potential risks from compromised embedded systems
- **Application Penetration Testing**:
  - Concentrates on the software applications running in the CPSoS

  - Identifies vulnerabilities in web and mobile applications, including authentication flaws, input validation issues, and insecure APIs
  - Assesses the potential impact of application-level attacks on the CPSoS
- **Fuzzing Testing**:
  - Utilizes automated input generation to identify vulnerabilities in software components of the CPSoS.
  - Sends malformed or unexpected inputs to applications and systems to trigger crashes, errors, or unexpected behaviours.
  - Helps uncover memory-related vulnerabilities, input validation issues, and potential points of compromise [35].
- **Physical Security Penetration Testing**:
  - Assesses the physical aspects of the CPSoS, including access controls, surveillance systems, and physical barriers
  - Tests the effectiveness of physical security measures in preventing unauthorized access to critical components
- **Supply Chain Penetration Testing**:
  - Evaluates the security of third-party components and software integrated into the CPSoS
  - Aims to identify vulnerabilities and potential backdoors introduced through the supply chain
- **Human Factors Testing**:
  - Assesses the impact of human interactions on CPSoS security
  - Evaluates the effectiveness of security training, social engineering risks, and insider threats within the CPSoS environment
- **Resilience Testing**:
  - Focuses on the CPSoS ability to recover from cyberattacks
  - Tests how well the system can continue operating despite ongoing attacks or disruptions
- **SCADA System Penetration Testing**:
  - Specific to CPSoS used in industrial control systems (SCADA systems)
  - Assesses the security of industrial automation and control components, including PLCs, HMIs, and communication protocols
- **IoT (Internet of Things) Security Testing**:
  - Concentrates on IoT devices integrated into the CPSoS
  - Identifies vulnerabilities in IoT hardware, firmware, and communication protocols that might impact the CPSoS
- **OPC-UA Penetration Testing**:
  - Focuses on the security of OPC-UA communication protocols and implementations
  - Identifies vulnerabilities in OPC-UA servers and clients, ensuring secure data exchange and preventing unauthorized access

Each of these approaches plays a vital role in ensuring the security and resilience of Cyber-Physical Systems of Systems (CPSoS). The inclusion of fuzzing testing helps to address potential vulnerabilities in software components by simulating real-world attacks with unexpected inputs.

3.1.5 State-of-the-art for automated Penetration Testing

During the last decade, cyber-security firms and security systems developers have been heavily focusing on producing automated [38] and semi-automated PT frameworks and systems aiming to facilitate the work of network penetration testers and make the assessment of network security more accessible to non-experts. Multiple systems are available for public use varying from free and open source to more costly products. Popular products used in PT communities include Core- Impact, Nexpose, Nessus, Qualys, Tenable, Immunity Canvas and Metasploit. The main contribution offered by these systems compared with the traditional security and vulnerabilities scanners such as Nessus, is their functionalities (planning, scanning, and exploiting) along with simplicity and flexibility of use (automation of certain tasks, visualization, reporting). Yet, the offered automation (mainly related to the planning phase of the practice) remains limited to the planning of the practice, the organization of the tasks and the optimization/visualization and the automated reporting (phase 4). Nevertheless, the heart of the PT practice was often neglected or poorly exploited. In fact, determining the exploitable vulnerabilities and launching the relevant exploits, digging inside, and pivoting to create a new vector of attack is undoubtedly the most challenging part. The difficulty itself lies on the current PT systems which have radically changed and evolved and have become more complex, covering new attack vectors and shipping increasing numbers of exploits and information gathering modules. Thus, the problem of efficiency has emerged and controlling alike framework successfully along with maintaining efficiency, is indeed the most important challenge.

3.1.5.1 Fuzzing State of the Art

Fuzzing is an automatic testing technique consisting of generating various inputs to break the system. To do this, the system's output (System Under Test (SUT)) is monitored to detect any behaviour different from the usual one. This technique is very popular for finding 0 days since it detects errors never detected before without influencing the system or knowing its operation in depth [1].

The fuzzing inputs are specially designed to trigger unexpected behaviours in the SUT and allow the finding of errors such as bad memory violations, assertion violations, incorrect handling of nulls, locks, infinite loops, undefined behaviours or incorrect management of other resources. Compared to other vulnerability hunting strategies, such as code inspection or reverse engineering, fuzzing has the advantage that it can be done on a large scale and unattended since the fuzzing process is often automated. Today, there are a large number of fuzzers, and each of them has its characteristics. Therefore, there are different methods to classify fuzzers [10].

Type

Depending on the information that the fuzzer needs, it can be classified into the following three approaches, as already described in section 3.1.2:

- ***Black box fuzzer (B)***: The first fuzzers were of this type and did not need to know the system's internals or the source code. Although these fuzzers initially worked completely randomly, today, they use techniques such as grammar to generate new entries. In this type of fuzzers, it is difficult to determine the code coverage achieved.
- ***White box fuzzier (W)***: In this case, the fuzzier needs to access the source code or binary and know the internal operation; they can also use techniques such as instrumentation that involve making changes. Therefore, before running these fuzzers, it is necessary to perform a system

scan to detect their characteristics. Guided coverage and dynamic symbolic analysis are commonly used in these fuzzers.

- **Gray box fuzzer (G)**: These fuzzers need certain information to work correctly. Usually, this information is used to carry out the instrumentation. Nowadays, it is the most used technique since without having exhaustive information about the system that is being tested, it obtains better results [2].

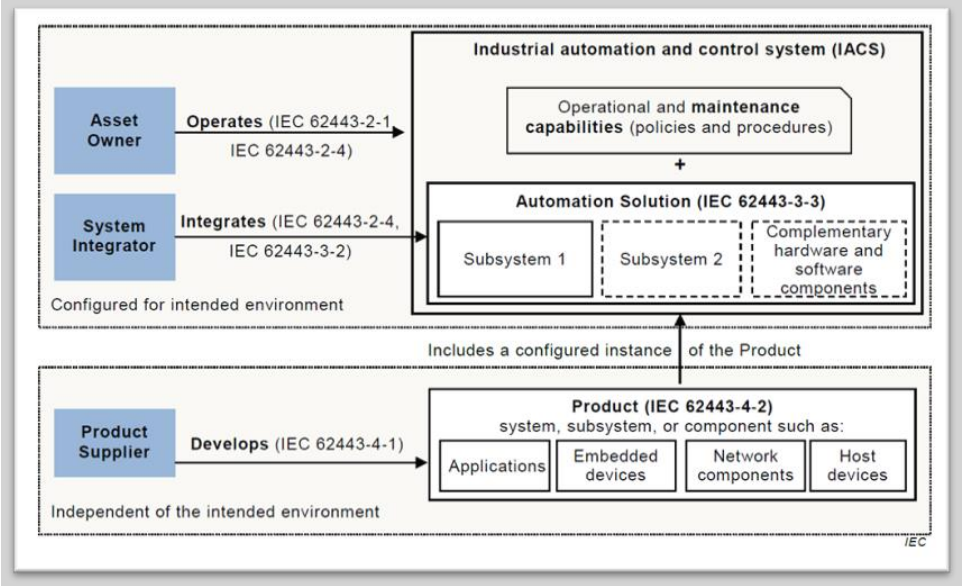

Figure 5: Example of the scope of the IEC 62443 standard documents

Generation of inputs

Fuzzers can also be classified depending on how they generate new inputs into two categories:

- ***Mutational (M)***: fuzzers are of this type when they generate new entries starting from those previously generated and performing different operations on them. This technique works very well when the inputs are complex, but they need to receive feedback from the system to work correctly.
- ***Generational (G)***: To generate the new entries, the SUT specifications are based, so it is necessary to know the syntax of the entries and the protocol used. It cannot be used with SUTs that are not precisely known but generate more accurate inputs than mutational ones.

Intelligence

Another way of classifying the fuzzers could be by taking into account whether the output of the system is fed back to generate the new inputs or not:

- ***Smart (S)***: These are the fuzzers that adapt the generation of inputs to the system's output; that is, they use the feedback technique to generate the new inputs.
- ***Dumb (D)***: These fuzzers do not use system output information; they execute faster than the Smart type but tend to be less effective in finding vulnerabilities.

Fuzzing technique

Fuzzers today use various techniques to improve the search for vulnerabilities by making improvements in different phases of fuzzing. Some of the most used techniques are Random Mutation

(MA), Grammar (GR), Dynamic Symbolic Execution (DS), Dynamic Spot Analysis (DM), Guided Coverage (CG), and Programming Algorithms (AP), static analysis (SA), Genetic Algorithms (GA) and Machine Learning (ML).

Target

Another characteristic to take into account when selecting a fuzzer is knowing the type of SUT it accepts as files (F), libraries (L), protocols (P), firmware (FW) and applications (A).

Table 1: Analysis of the characteristics of the fuzzers

Fuzzer	Characteristics				
	Type	Generation	Intelligence	Technique	Target
AFL	G	M	S	CG/GA	F/A
AFLfast	G	M	S	CG/AP	F
AFLgo	G	M	S	CG/AP	F
Boofuzz	B	G	S	CG	P
Frankestein	G	M	S	MA/CG	F
Honggfuzz	G	M	S	CG	F
IoTfuzzer	B	M	S	DS	FW
Learn&Fuzz	G	G	S	ML	F
Libfuzzer	G	M	S	CG	L
Peach	B	M/G	S	MA/GR	F/P
Smartfuzz	W	M	S	DS/CG	A

After analyzing the state of the art of fuzzing and the need to test Open 62541, the fuzzer that best fits the requirements is Boofuzz. Boofuzz is a Python-based fuzzer that can be downloaded from GitHub [6]. This fuzzer allows testing different protocols, which can be adapted to different scenarios.

3.2 Hypothesis testing

In the realm of statistics, hypothesis testing involves the utilization of a data sample to assess the credibility of a hypothesis related to the distribution of that particular sample. In the domain of IoT, a variety of algorithms rooted in hypothesis testing have found effective application in addressing domain-specific challenges.

Searching literature databases reveals the work performed in this specific area of research. More specifically, these techniques have been extensively employed, notably in the domain of attack detection. Authors in [42] employ a hypothesis testing algorithm to identify a Link Flooding Attack (LFA) within an IoT Network, whereas authors in [43] leverage from the hypothesis testing algorithm to enhance a distributed attack detection system. Besides of these aforementioned research papers, there are numerous other applications of Hypothesis testing-based algorithms. For instance, they were employed to conduct polling for the values of multiple Key Performance Indicators (KPIs) in a fog-based IoT sensor network [44], to streamline protocols for authenticating IoT devices [45], and to ensure the privacy of Smart Energy Meters [46]. Furthermore, decentralized methods have been proposed to optimize energy consumption in IoT sensors [47], safeguarding privacy [48], and maintaining robustness in scenarios involving noise and small-scale data sets [49].

As far as we know, no other hypothesis-based application in the existing literature assesses various

mitigation strategies within an industrial environment. To this end, section 6 will introduce an innovative approach to differentiate between distinct sets of mitigation measures. This method empowers the system operator to make adjustments to an existing set of mitigation actions and observe the resultant effects on the system. To elaborate, Key Performance Indicator (KPI) values generated from various mitigation sets are subjected to clustering via a machine learning algorithm. Subsequently, the divergence between these clusters is assessed utilizing a p-value, determined through a Monte Carlo-based technique known as Statistical Significance of Clustering (SigClust), employing Soft Thresholds [50].

4 CPSoS security analysis

To analyse and perform a cybersecurity assessment of different components, one of the first steps is to create a threat model. In D2.3, a general threat modelling process is described that also can be applied to the systems under the scope of Zero-SWARM. A detailed description of each phase is drafted in D2.3 [40].

Specifically, in this deliverable, a threat modelling of OPC-UA, Modbus and MQTT protocols, widely used in the industry and specifically in the Zero-SWARM project, has been carried out. In the following list, a description of each of the threat modelling phases for the specific implementation in Zero-SWARM has been carried out:

- **Form a team:** in this case, the requirements were obtained from the project use cases. Analysing the needs of the different use cases, it has been decided to analyse the aforementioned three protocols (OPC-UA, Modbus and MQTT). The team composed by a use case leader, participants and the cybersecurity expert partner will do a follow up of the different steps of the threat modelling in order to validate or update with new requirements.
- **Establish the scope**: The scope selected for each system will require identification on the industrial physical target, such as OPC-UA, Modbus, MQTT or others. The scope will also require the identification of the TCP IP port and other network perimetral conditions to have a holistic view of the protocol implementation in the project use cases.
- **Determine likely threats:** In our case, for the selected three protocols, we will need to determine threats. This will require a vulnerability assessment study and will be carried out using penetration testing modules that will be further describe in the section 5. At this moment in the project, this is the stage of threat modelling.
- **Rank each threat**: Once the vulnerability analysis of the implementation of the three protocols has been completed, a ranking of the detected vulnerabilities should be created, depending on their criticality. This ranking will allow us to prioritise the mitigation of critical vulnerabilities in the first part, leaving the less critical ones for the end.
- **Implement mitigations**: In this step, following the ranking done in the previous step, vulnerabilities will be mitigated. These can be mitigated through the creation of patches, for example, or if the criticality of the vulnerability is very low, the risk can be accepted and not mitigated.
- **Document results:** Once the entire threat modelling process has been carried out, it should be documented so that the same procedure can be followed if the threat reoccurs or to serve as an example for modelling new threats.

4.1 Application of standardized methodologies

IEC 62443 is a set of documents covering the cybersecurity needs for a complete solution in the field of industrial automation and control systems. Figure 6 represents the sphere of influence of each of the documents within a complete solution.

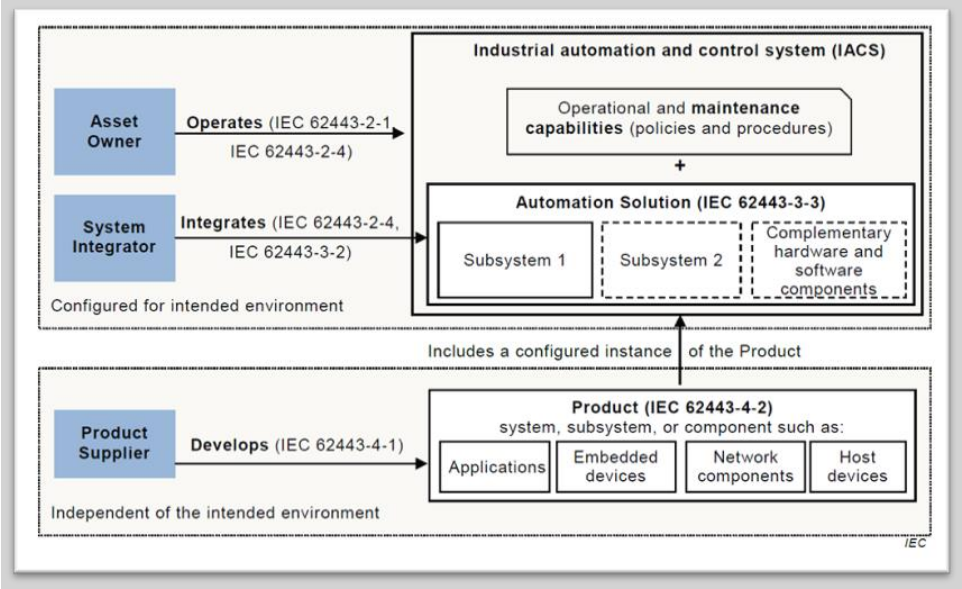

Figure 6: Example of the scope of the IEC 62443 standard documents

The figure shows three different roles: product supplier, system integrator and asset owner:

- A supplier of a product provides a component that must be integrated into a solution. IEC 62443-4-1 acts as a guide for suppliers to create industrial components that are cyber-safe. These components should adhere to specific technical cyber security obligations highlighted in IEC 62443-4-2. Implementing these requirements would ensure that products configured for integration into a complete industrial system comply with cyber safety obligations.
- The integration is carried out by the system integrator by what is indicated in the document IEC 62443-2-4 and IEC 62443-3-2, where it must be considered that there may be a multitude of products.
- Finally, the asset owner is responsible for the system's operation in accordance with what is indicated in the documents IEC 62443-2-1 and IEC 62443-2-4.

In the context of the project, two aspects are especially relevant: the supply chain, which is fundamental in two stages:

- **Development stage**. At this stage, the component manufacturer is in the middle of the supply chain and must control all the elements integrated internally/externally. To do this, the ISO/IEC 62443-4-1 standard in Practice 1 contemplates this type of situation in SM-9:
    - SM-9: Security requirements for externally provided components. This process should ensure that supply chain security is addressed with security best practices, including making updates, deployment guides, and the vendor's ability to respond when new vulnerabilities are discovered. Security in this process applies in turn to the provider's products that are used by the development team if they meet some of the following characteristics:
        - Degree to which the product conforms to the security context.
        - Degree of rigor applied to the implementation of the product.
        - Degree of verification and validation of the security of the component.

- Mechanisms for receiving and monitoring security incidents.
  - Vulnerabilities identification
- Presence of sufficient security documentation.
- Degree of support of the product by the provider.

Secondly, and associated with the role of component provider, the IEC 62443-4-1 standard in Practice 5 contemplates the identification of vulnerabilities, where it is specified that fuzzing is necessary:

- SVV-3: Vulnerability testing. There must be a process focused on identifying and detecting vulnerabilities in the product. This test must include:
  - Test with malformed inputs to detect vulnerabilities such as fuzzing.
  - Analysis of the attack surface to identify possible entry routes.
  - Exploration of known black box vulnerabilities.
  - For compiled software, software composition analysis on all executable binaries.
  - Dynamic tests of resource management at runtime.

4.2 Security by design (OPC-UA, MQTT)

Security by design is a fundamental consideration in implementing industrial communication systems. In addition, security in industrial communication is of vital importance today, especially in the context of industrial cybersecurity. Protecting critical systems is very important in a world where automation and connectivity are fundamental to the efficient operation of industry. In this paper, we will explore how to address security by design through three key points:

4.2.1 Mapping IEC 62443 – OPC-UA PART 2 SECURITY

One of the main pillars of security in industrial communication is effectively linking security principles between different standards. This is especially important in industrial cybersecurity, where interoperability and protection of critical systems are imperative.

IEC 62443 is an international standard focusing exclusively on cybersecurity in industrial automation systems. It provides a robust framework for assessing and mitigating security risks in critical industrial environments. On the other hand, OPC-UA is a widely adopted protocol for communication in industrial environments, known for its efficiency and scalability. A powerful synergy is achieved using IEC 62443 safety principles in OPC-UA.

This synergy enables consistent implementation of security measures from strong authentication to data encryption. These measures are essential to ensure that only authorized devices and users can access resources and that sensitive information is kept secure. The connection between IEC 62443 and OPC-UA establishes a solid foundation for implementing secure systems in the industry, helping protect critical systems and sensitive data.

4.2.2 OPC 11030 UA Modelling Best Practices

Security in OPC-UA is not limited to secure data transmission but extends to the efficient and secure representation of information in industrial systems. OPC 11030 UA Modelling Best Practices is a fundamental guide that provides detailed guidance on effectively modelling data in OPC-UA. Proper data modelling ensures efficiency and security in implementing OPC-UA-based systems [13][14].

This resource provides guidelines for defining data types, objects, and variables consistently and securely. By following these best practices, organizations can design systems with a data structure that is easy to maintain and simultaneously resistant to potential attacks or security breaches. This helps ensure data integrity and information confidentiality, crucial factors in industrial environments where accuracy and security are priorities [21][22].

4.2.3 Modern Protocol: OPC-UA vs MQTT

Choosing the right communication protocol is a strategic decision in any industrial application. In this context, it is essential to consider the differences and advantages between two widely used protocols: OPC-UA and MQTT.

OPC-UA was launched in 2008 to update the original OPC interoperability standard, designed for secure and reliable data exchange in industrial automation. OPC is based on a client/server architecture, where the OPC server becomes the hardware communication protocol, and any program that needs to connect to the hardware becomes the OPC client software.

However, it is crucial to consider particular challenges before implementing an OPC or OPC-UA-based architecture. The most commonly cited is the complexity of implementation, as the OPC-UA specification document is 1.240 pages long. In addition, full implementation of OPC-UA can be expensive, with high CPU resource requirements, development costs and ongoing support costs. OPC is also inflexible and needs help handling the various heterogeneous data structures and devices in today's industrial environment. In addition, it has limitations in handling multiple data consumers and requires to provide the proper data decoupling needed for a one-to-many approach.

MQTT, on the other hand, is a transport protocol that originated in 1999. MQTT is a lightweight network protocol based on the publish/subscribe (pub/sub) model that allows multiple data consumers and is designed for resource-constrained devices and networks with limited bandwidth, high latency or low reliability.

The MQTT specification is simple and easy to implement, with a specification document of approximately 80 pages, and can be extended with the Sparkplug specification, which adds another 60 pages. MQTT is lightweight and flexible, focusing on exception-only reporting and minimizing the data footprint. In addition, MQTT is cost-effective, is based on open standards, and provides TCP/IP layer-level security.

A significant advantage of MQTT is its growing industry adoption, with multiple vendors natively implementing MQTT-Sparkplug in both hardware and software. In addition, leading cloud platforms, IoT platforms, edge computing platforms, big data and third-party applications support MQTT.

Within the Eclipse Tahu project, the Sparkplug specification defines how to use MQTT in real-time industrial applications. Sparkplug establishes a standard namespace for MQTT topics, defines the payload structure and manages session state for industrial applications, meeting the requirements of real-time SCADA implementations. This facilitates the adoption of MQTT in applications that require data communication between Operational Technology (OT) and Information Technology (IT), providing contextual information that is essential for big data analytics, Machine Learning (ML) and Artificial Intelligence (AI).

Importantly, although OPC-UA and MQTT have different approaches to handling data, they can work

together harmoniously in specific scenarios. MQTT can be used to overcome some of the limitations and challenges associated with OPC-UA, enabling efficient and secure data communication in Industrial Internet of Things (IIoT) environments.

In summary, A Modern Protocol: OPC-UA vs. MQTT [7] is a valuable guide to help organizations decide which communication protocol to use based on their security and efficiency needs in specific industrial applications. Understanding the strengths and weaknesses of each protocol is essential to designing secure and robust systems.

5 Penetration testing modules

5.1 Zero-SWARM penetration testing methodology

The general background and principles of penetration testing were given in section 3.1, in order to concentrate mostly on technical aspects in this section. More specifically, as the Zero-SWARM project focuses on CPSoS environments, the penetration testing modules to be developed in task T5.4 will focus on these components. Specifically, in this first phase of the project, a fuzzer-based penetration test module will be developed for the OPC-UA protocol that is used in the project (implemented with Open62541) [5]. In parallel with the development of a fuzzer-based penetration testing module for the OPC-UA protocol and in order to examine other well-used industrial communication protocols, a first assessment of the Modbus communication protocol has been performed. A simulated industrial environment has been set up to evaluate the vulnerabilities of the Modbus protocol in an industrial production line. The methodologies used in both cases (OPC-UA, Modbus) will be thoroughly explained in the following sections.

5.1.1 OPC-UA

The Open Platform Communication Unified Architecture (OPC-UA) is an open communication standard that enables communication between industrial machines [3]. It is a protocol for transferring data in the form of objects instead of discrete data points. This increases the accessibility of plant data by allowing the information stored in a shared object to be reused. OPC-UA also has a service-oriented model, improving the platforms' security and interoperability [4].

The OPC-UA transport protocol, among other things, offers a reliable and secure communication infrastructure, as it manages lost messages, communication between nodes and failures. Security is built into OPC-UA and has been a design goal since its inception. On top of the transport layer, a secure channel using encryption and digital signatures protects messages from unauthorized alteration and eavesdropping. Furthermore, this layer authenticates and authorizes specific instances of OPC-UA applications using authentication procedures based on digital certificates. This allows administrators to implement granular access control across critical infrastructures. A session connects a client application and a server used to exchange information. OPC-UA servers must authenticate and authorise users intending to establish client-side sessions. The specification supports three mechanisms: username/password combinations, digital certificates, and WS-compliant user tokens [23][24][25][26].

Open62541

It implements OPC-UA, making it a machine-to-machine communication protocol for industrial automation and widely used in various industries such as manufacturing, energy, and transportation. This implementation is done in C, allowing developers to create OPC-UA servers and application clients.

This implementation is prepared to be portable, scalable and efficient [3].

The Open62541 project is open-source and encourages community contributions with a platform for collaboration and enhancements. The entire project is on Github under the Mozilla Public License so that developers can use and modify it [4].

The project aims to provide a flexible and robust framework for implementing OPC-UA functionality. It supports several features of the OPC-UA standard, including different transport protocols (such as TCP and HTTPS), security mechanisms (encryption, authentication, and authorization), and a wide range of data types.

Open62541 is an increasingly used project in industrial automation since it is compatible with various platforms and follows the OPC-UA standard.

OPC-UA Fuzzers

State of the art study of fuzzers details several fuzzers, and each has its characteristics, and the targets to which they are directed are also usually different. Therefore, when analyzing fuzzers for testing OPC-UA systems, it is first necessary to analyze the targets to which they are directed. Those directed to files will be unable to test them properly since only coding errors would be detected, not configuration errors. Therefore, different fuzzers have been analyzed to see which is most suitable for testing OPC-UA, specifically Open62541, due to this implementation will be used to test project implementation in Zero-SWARM.

5.1.2 Modbus TCP/IP communication protocol

Modbus, originally developed in the late 1970s, has cemented its place as one of the most prevalent communication protocols in industrial automation and control systems. Its straightforward design and ability to operate over a variety of physical media, including serial and Ethernet connections, have made it a preferred choice for connecting devices such as Programmable Logic Controllers (PLCs), sensors and Human-Machine Interfaces (HMIs). Modbus simplifies the process of acquiring data from sensors, controlling actuators and monitoring industrial processes, contributing significantly to the efficiency and productivity of industrial systems.

Modbus encompasses several variants, including Modbus RTU (Remote Terminal Unit), Modbus ASCII, and Modbus TCP/IP. Modbus RTU and ASCII primarily operate over serial connections, while Modbus TCP/IP uses Ethernet. RTU is known for its binary encoding, making it more efficient for serial communication, while ASCII employs ASCII characters for human readability. Modbus TCP/IP, on the other hand, is suitable for Ethernet-based networks, offering faster data transfer speeds and broader compatibility.

This communication protocol boasts several key features that make it highly desirable in industrial environments. It employs a master-slave architecture, where a master device initiates requests and slave devices respond with data. This simplicity facilitates easy integration and troubleshooting. Additionally, Modbus supports multiple data types, including discrete inputs, coils, input registers and holding registers, making it versatile for various data exchange requirements. Its ability to communicate over long distances and its resilience in noisy industrial environments further contribute to its popularity.

On the other hand, while Modbus offers numerous advantages, it is not without limitations. One significant drawback is its lack of built-in security features. Traditional Modbus implementations do

not include authentication or encryption, which can leave industrial systems vulnerable to unauthorized access and cyberattacks. As a result, security measures such as VPNs, firewalls and network segmentation are often necessary to protect Modbus-enabled systems from potential threats.

As industrial systems evolve towards the Industrial Internet of Things (IIoT), Modbus continues to play a primary role. With the emergence of gateways and converters, Modbus devices can seamlessly integrate into IIoT ecosystems, allowing for remote monitoring, predictive maintenance and real-time data analysis. Modbus remains a foundational protocol in the transition to more interconnected and data-driven industrial processes.

In summary, the Modbus communication protocol has established itself as a fundamental tool in industrial environments, enabling efficient and reliable data exchange among various devices. Its simplicity, versatility and widespread adoption continue to make it a cornerstone of industrial automation and control systems. However, as cybersecurity concerns grow, it is crucial to implement appropriate security measures when deploying Modbus in industrial networks to safeguard against potential threats.

5.2 Vulnerability analysis of CPS

5.2.1 OPC-UA communication protocol

OPC-UA (Open Platform Communications Unified Architecture) is a widely used communication protocol in industrial automation and control systems. OPC-UA penetration testing involves assessing the security of OPC-UA implementations to ensure that they are resilient against potential cyberattacks. An overview of the process:

- **Scope definition**: Define the scope of the penetration test, including the OPC-UA servers, clients, and related components to be tested. Determine the goals and objectives of the test.
- **Information gathering**: Gather information about the target OPC-UA systems, including network architecture, communication protocols, OPC-UA endpoints, and versions.
- **Vulnerability analysis**: Identify potential vulnerabilities in the OPC-UA implementation. This could involve manual analysis of configuration files, network traffic analysis, and the use of automated scanning tools.
- **Authentication & authorization testing**: Assess the effectiveness of authentication mechanisms and access controls within the OPC-UA infrastructure. Verify that only authorized users have the appropriate permissions to access resources.
- **Encryption and data integrity**: Evaluate the encryption and data integrity mechanisms used in OPC-UA communication. Ensure that sensitive information is protected during transmission.
- **Secure communication**: Test the implementation's adherence to secure communication practices, such as enforcing the use of secure OPC-UA endpoints and disabling insecure encryption algorithms.
- **Fuzzing & protocol testing**: Utilize fuzzing techniques and specialized OPC-UA protocol testing tools to send malformed or unexpected messages to the OPC-UA servers and clients. Monitor for crashes, unexpected behaviours, or vulnerabilities triggered by these inputs.
- **Boundary testing**: Test the system's behaviour when subjected to boundary conditions, such as sending extremely large or small data payloads, to identify potential buffer overflows or underflows.

- **Error handling & resilience testing**: Assess how the OPC-UA implementation handles errors, exceptions, and unexpected situations. Evaluate the system's resilience against Denial-of-Service (DoS) attacks.
- **Privilege escalation testing**: Attempt to escalate privileges within the OPC-UA environment, checking for potential vulnerabilities that could allow an attacker to gain unauthorized access.
- **Reporting**: Document all findings, including vulnerabilities, weaknesses, and recommendations for remediation. Provide a detailed report to the organization with actionable steps to improve the security of their OPC-UA implementation.

Fuzzing Penetration Testing in OPC-UA:

Fuzzing is a specialized penetration testing technique [9] that involves sending intentionally malformed or unexpected inputs to a target system to trigger vulnerabilities and uncover security weaknesses. Fuzzing is particularly useful for finding memory-related vulnerabilities like buffer overflows, format string vulnerabilities, and more. It can be automated and run continuously to identify new vulnerabilities that might emerge as the software evolves over time.

In the context of OPC-UA, fuzzing testing focuses on identifying vulnerabilities in the OPC-UA communication protocol and its implementations:

- **Fuzzing tool selection**: Choose or develop a fuzzing tool that is capable of generating invalid or unexpected OPC-UA messages and sending them to OPC-UA endpoints.
- **Input generation**: Configure the fuzzing tool to generate a variety of inputs, including malformed OPC-UA messages, to test different aspects of the protocol.
- **Message parsing and handling**: Send the generated inputs to the target OPC-UA server or client and observe its behavior. Monitor for crashes, exceptions, unexpected responses, or abnormal behaviors.
- **Code coverage analysis**: Analyze the code coverage achieved during fuzzing to understand which parts of the OPC-UA implementation are exercised by the generated inputs. This helps identify untested or potentially vulnerable code paths.
- **Error detection**: Identify how well the OPC-UA implementation detects and handles errors resulting from the malformed inputs. Look for cases where the system fails to gracefully handle unexpected data.
- **Vulnerability identification**: If crashes, unexpected behaviors, or other anomalies are detected, investigate these issues further to determine if they represent genuine vulnerabilities.
- **Reporting**: Document the results of the fuzzing testing, including the vulnerabilities found, their potential impact, and recommendations for remediation. Provide guidance on improving the robustness of the OPC-UA implementation against fuzzing attacks.

Fuzzing testing in OPC-UA aims to uncover vulnerabilities that might not be easily identifiable through traditional testing methods. It helps organizations ensure the reliability and security of their OPC-UA communication and minimize the risk of cyberattacks targeting industrial control systems.

5.2.2 Modbus TCP communication protocol

While Modbus offers significant advantages in terms of simplicity and efficiency, it is not immune to vulnerabilities that can potentially jeopardize the security and integrity of industrial networks. As

industries increasingly rely on Modbus for real-time monitoring and control, it becomes imperative to provide a vulnerability analysis to identify and address potential security threats. This analysis encompasses various aspects, including authentication and encryption challenges, the susceptibility of function codes to exploitation, network-based attacks, logging and monitoring limitations, firmware and software vulnerabilities, insider threats and the absence of redundancy and failover mechanisms. By thoroughly understanding these vulnerabilities and implementing robust security measures, organizations can fortify their Modbus-enabled systems against potential risks and ensure the continued reliability of critical industrial processes. The following list groups some of the most well-known vulnerabilities related to Modbus.

1. ***Authentication and encryption vulnerabilities***:

Traditional Modbus implementations often lack robust authentication and encryption mechanisms, leaving industrial networks vulnerable to unauthorized access and data interception. Attackers can exploit this weakness to gain unauthorized control over industrial devices or eavesdrop on sensitive data transmissions. To address this vulnerability, organizations should consider implementing secure communication channels using technologies like Virtual Private Networks (VPNs) or encryption protocols to protect Modbus traffic from unauthorized access and tampering.

2. ***Vulnerable Function Codes:***

Modbus function codes, which are used to execute specific actions like reading or writing data to registers, can introduce security risks if not adequately protected. Some function codes can be exploited by malicious actors to manipulate data or take control of industrial processes. It's crucial to implement strict access controls and role-based permissions to restrict the execution of critical Modbus function codes to authorized personnel only. Regularly reviewing and updating these access permissions is essential to maintaining a secure environment.

3. ***Network-based attacks:***

Modbus devices connected to Ethernet networks, especially Modbus TCP/IP, are susceptible to various network-based attacks. These attacks can include port scanning, Denial-of-Service (DoS) attacks, and intrusion attempts. To mitigate these risks, organizations should consider implementing network segmentation, firewalls, and intrusion detection systems to isolate and protect Modbus networks from external threats. This helps reduce the attack surface and enhances the security of critical industrial systems.

4. ***Logging and monitoring challenges:***

Many Modbus implementations lack comprehensive logging and monitoring capabilities, making it challenging to detect and respond to security incidents effectively. To address this vulnerability, organizations should invest in centralized logging and monitoring solutions that capture and analyze Modbus traffic for anomalies or suspicious activities. Real-time alerting can enable security teams to respond promptly to potential threats and incidents, enhancing overall security posture.

5. ***Firmware and software vulnerabilities:***

Modbus devices and associated software may contain vulnerabilities that can be exploited by attackers to compromise the integrity and availability of industrial systems. Regularly applying security patches [15] and firmware updates is crucial to mitigate these vulnerabilities. Organizations should also stay informed about vendor advisories and follow best practices when deploying and configuring Modbus devices to reduce the risk of exploitation.

6. ***Insider threats:***

Insider threats, including employees or contractors with access to Modbus systems, can pose significant risks if they misuse their privileges. To address this vulnerability, organizations should implement least privilege principles, conduct thorough background checks on personnel, and establish proper access controls and auditing to monitor and mitigate insider threats effectively.

7. Lack of redundancy and failover:

Many Modbus systems may lack redundancy and failover capabilities, making them susceptible to disruptions caused by hardware failures or network outages. Implementing redundancy and failover mechanisms can ensure continuous operation of critical Modbus systems in the event of failures, enhancing the reliability and resilience of industrial processes.

5.3 OPC-UA Penetration testing toolset

Figure 7 shows the general architecture diagram of the Metasploit and Fuzzer components that will be part of the module.

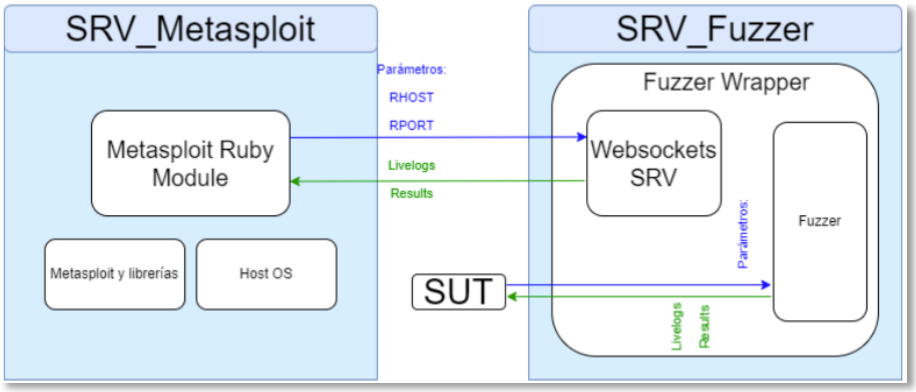

Figure 7: Metasploit and Fuzzing modules architecture

Two main machines can be identified in the system, SRV_Metasploit and SRV_Fuzzer, each playing a crucial role in the operation of the environment.

SRV_Metasploit hosts a number of components essential to our operation. On the one hand, Metasploit, a powerful cybersecurity platform, and on the other hand, all the libraries necessary for its proper functioning.

The SRV_Fuzzing machine will be in charge of fuzzing the OPC-UA system. Inside it, the Boofuzz fuzzer resides, which is being adapted to discover vulnerabilities and weaknesses in the OPC-UA system. However, this machine not only hosts the fuzzer, but also contains the program to run it efficiently and accurately. Furthermore, this machine acts as the central communication point with the Metasploit module hosted on the SRV_Metasploit machine, ensuring a smooth and coordinated interaction between the components.

5.3.1 Fuzzer

The main goal of fuzzing is to trigger unhandled exceptions, crashes, or unexpected behaviour that might indicate the presence of vulnerabilities [16][17][18][19][20][32][33][34]. The information below explains how fuzzing works:

- **Input Generation**: Fuzzing involves generating a wide range of input data, which can include malformed or unexpected data packets, files, or commands. These inputs are designed to stress the target application's input-handling mechanisms.

- **Input Mutation**: Fuzzing tools take the generated inputs and mutate them by modifying specific bytes, characters, or structures. This creates a diverse set of inputs to explore different code paths.
- **Execution and Monitoring**: The mutated inputs are fed into the target application, and the application's behavior is closely monitored during execution. The goal is to identify crashes, memory leaks, unexpected error messages, or any other signs of vulnerabilities.
- **Coverage Analysis**: Some advanced fuzzing tools can also analyze code coverage, identifying which parts of the code are executed with the given inputs. This helps testers understand the reach of their fuzzing efforts.
- **Corpus Management**: Fuzzing tools often maintain a corpus, which is a collection of inputs that caused interesting behavior or crashes. Testers can evolve this corpus over time to focus on specific code paths or potential vulnerabilities.
- **Triage and Reporting**: When a crash or unexpected behavior is discovered, the tester examines the issue to determine if it's a legitimate vulnerability. Valid vulnerabilities are documented and reported to the development team for remediation.

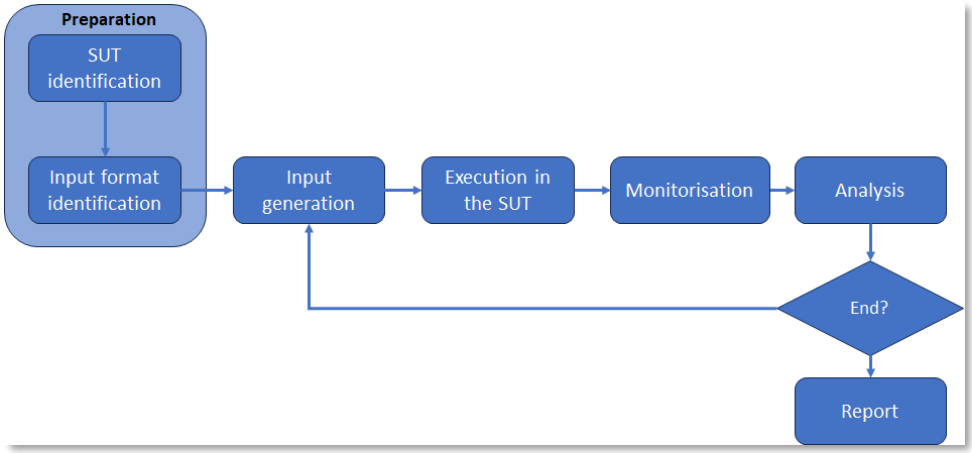

Figure 8: Phases of the fuzzing process

Figure 7 shows the fuzzing process designed in Zero-SWARM. First, a preparation phase is carried out where the system to be tested and the format of the inputs it executes are identified. This is a phase before the execution of the fuzzer. Once this is done, the generation of the input patterns begins; these are executed in the system, and their output is monitored and analyzed. This process will be repeated repeatedly using the system output as a source of information to generate the new entries.

5.4 Initial validation tests for the penetration test modules

5.4.1 OPC-UA simulation testbed

The complete architecture of the platform that will test the penetration test module has been designed. A modular and scalable platform has been designed, i.e. each module will perform a specific function, but will be interconnected to operate together. The system consists of the following six modules:

- **Telemetry-server**: This component will be in charge of capturing traffic during the course of the penetration test in order to have the evidence and analyse the data once the penetration test is finished.

- **OPC-UA server:** This component is responsible for sending data via the OPC-UA protocol to clients. The data to be sent are simulated in this first version of the validation environment.
- **Two OPC-UA clients**: OPC-UA clients consume the data sent by the server.
- **Penetration testing module**: this module runs the fuzzer that will detect OPC-UA vulnerabilities. The penetration testing will be launched using Metasploit

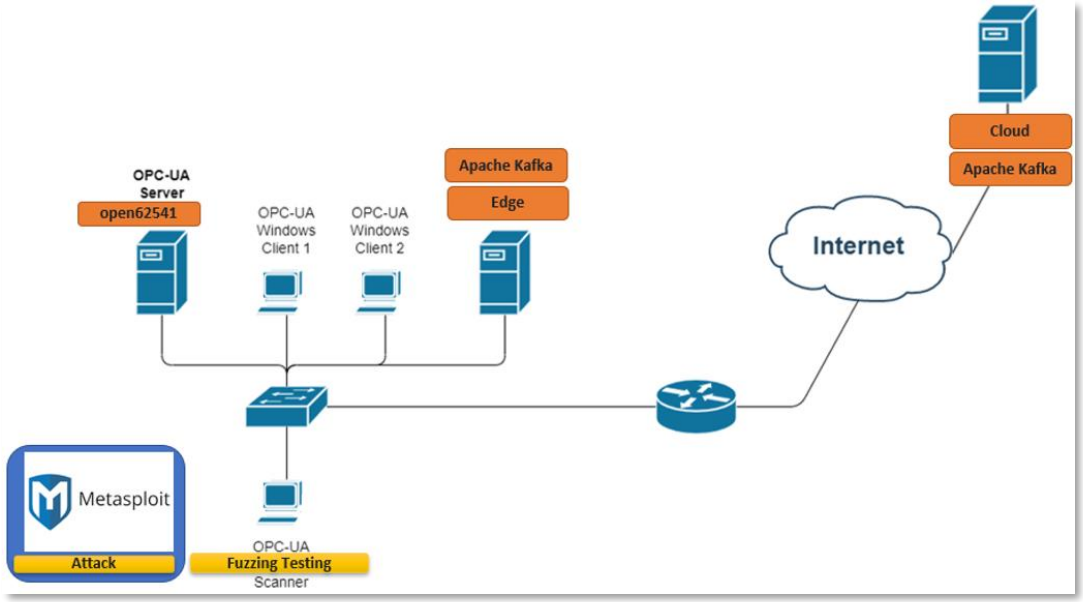

Figure 9: Preliminary penetration testing module validation environment

Validation test proves:

```
┌──(kali㉿kali)-[/opt/metasploit-framework/embedded/framework/modules]
└─$ pip3 list |grep opcua
opcua                         0.98.13
```

```
┌──(kali㉿kali)-[/opt/…/modules/auxiliary/scanner/opcua]
└─$ ls -las
total 32
 4 drwxr-xr-x  2 root root  4096 Sep  4 18:05 .
 4 drwxr-xr-x 89 root root  4096 Sep  4 18:05 ..
 4 -rwxr-xr-x  1 root root  2671 Sep  4 18:05 opcua_hello.py
 8 -rwxr-xr-x  1 root root  7426 Sep  4 18:05 opcua_login.py
12 -rwxr-xr-x  1 root root 11218 Sep  4 18:05 opcua_server_config.py
```

5.4.2 Modbus TCP simulation testbed

A major obstacle in the field of industrial cybersecurity is the high expenses associated with acquiring the necessary devices and software licenses for gaining hands-on experience in operating OT systems and procedures. Virtualization emerges as a valuable solution to surmount this hindrance, allowing individuals to familiarize themselves with the intricacies of ICS protocols and features at a comparatively modest initial expense.

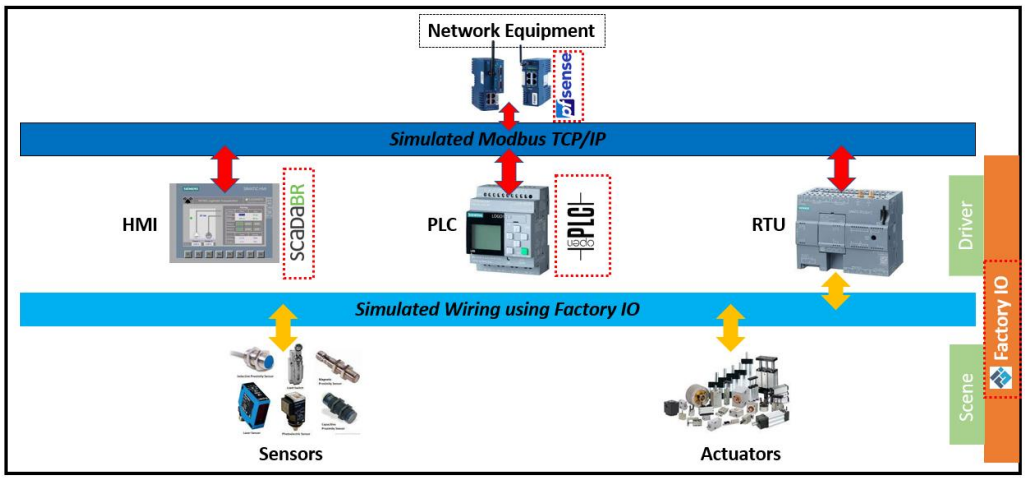

Figure 10: Physical architectural diagram of simulated environment

As illustrated in Figure 10, the simulated industrial environment comprises several components. All of these components were installed on a Windows 11 host machine with the following specifications: Intel Core i5-6600 CPU @ 3.30 GHz, 40GB RAM, and 2x 1TB SSD storage. These components were deployed either as Virtual Machines (VMs) or directly installed on the host.

- **pfSense**: Software for simulating the network equipment
- **SCADABR**: Software for simulating the HMI
- **OpenPLC**: Software for simulating the PLC
- **OpenPLC Editor**: Complementary software in order to program the PLC activities
- **Factory IO**: Software for simulating the RTU (driver) and the sensors & actuators (scene)
- **Kali Linux**: Operating system simulating the malicious actions against the Modbus TCP simulation environment

pfSense

pfSense is a widely used open-source firewall and routing software built on the FreeBSD operating system. Its origins trace back to 2004 when it emerged as a fork of the m0n0wall project. Over the years, pfSense has evolved into a versatile and powerful solution, making it a popular choice in the realm of network security and management. Widely known for its adaptability and scalability, pfSense caters to the diverse networking needs of businesses, educational institutions and individuals seeking to bolster the security and efficiency of their network infrastructure.

At its core, pfSense operates as a firewall and router, delivering critical services such as Network Address Translation (NAT), VPN support or traffic shaping. What sets pfSense apart is its user-friendly web-based interface, designed to give accessibility to users to take advantage of the majority of its capabilities. This feature empowers administrators to efficiently manage network traffic, allowing for the implementation of Quality of Service (QoS) policies to optimize bandwidth allocation and ensure that essential applications receive the requisite resources. Furthermore, pfSense's strength lies in its extensibility, boasting a robust package system that enables users to enhance its capabilities with additional functionalities. These can include intrusion detection and prevention systems (IDS/IPS), content filtering and proxy services. This flexibility and the wealth of features make pfSense a formidable choice for those seeking a versatile and powerful network security and routing solution.

SCADABR

SCADABR is a robust and highly adaptable open-source Supervisory Control and Data Acquisition (SCADA) system. It is a valuable tool for industries requiring real-time monitoring and control of their processes and equipment. Renowned for its web-based interface, proficiency in historical data management and extensive customization features, SCADABR has garnered popularity among organizations seeking cost-effective and versatile SCADA solutions. This SCADA system operates within a dedicated Ubuntu Server Virtual Machine (VM) and establishes a virtual interface connection through pfSense. This meticulously configured setup ensures both efficient and secure network connectivity, enabling comprehensive monitoring and control capabilities for industrial processes and equipment. In this current setup of the industrial testbed, SCADABR will not be used since no Human Machine Interface (HMI) was required.

OpenPLC

OpenPLC stands as a versatile and open-source Programmable Logic Controller (PLC) platform, extending the capability to design, simulate and deploy industrial automation solutions. Its versatility lies in its compatibility with diverse hardware platforms, programming languages and communication protocols. Combined with its simulation capabilities, OpenPLC emerges as an invaluable resource for both industrial automation engineers and enthusiasts alike, offering a cost-effective and highly customizable PLC solution. This platform operates seamlessly within an Ubuntu Server VM, establishing a connection through the same network interface created with pfSense. This strategic network configuration ensures efficient communication among pfSense, SCADABR and OpenPLC, fostering a cohesive ecosystem for industrial automation management and control.

OpenPLC Editor

The OpenPLC Editor is a software tool that functions as an Integrated Development Environment (IDE), specifically tailored for the creation and editing of control logic programs designed for the OpenPLC platform. This application is rich in features, offering a wide range of capabilities aimed at streamlining the development, testing and management of control logic programs. By simplifying the design and deployment of control systems, the OpenPLC Editor emerges as an invaluable resource for industrial automation engineers and developers who work with OpenPLC.

It's important to note that the OpenPLC Editor operates directly on the host machine, rather than within a VM. It serves as a complementary tool to the OpenPLC platform. The output files this editor generates are subsequently loaded onto OpenPLC to program and execute processes on the PLCs. Notably, OpenPLC supports multiple programming languages, including Ladder Logic, Structured Text, Instruction List, Function Block Diagram and Sequential Function Chart, all of which adhere to the IEC 61131-3 standard. This flexibility accommodates diverse programming needs within the realm of industrial automation.

Factory IO

Factory IO is a robust and interactive simulation software specifically designed for educational and training applications within the area of industrial automation and control systems. Its primary function is to offer a virtual environment that empowers users to conceptualize, experiment with and validate a multitude of industrial automation scenarios, all without the requirement for physical hardware. Factory IO is a commercial software, providing a variety of subscription plans to cater to different user needs. It is installed directly on the host machine and leverages the pfSense network interface, facilitating direct communication with the other interconnected components. This seamless

integration enhances its utility in creating comprehensive and immersive industrial automation simulations for educational and training purposes.

Kali Linux

Kali Linux stands out as a potent and widely adopted Linux distribution, meticulously designed to cater to the needs of cybersecurity professionals and enthusiasts. Renowned for its comprehensive toolkit, frequent updates and robust community backing, Kali Linux serves as an indispensable asset for individuals engaged in ethical hacking, penetration testing or cybersecurity evaluations. In the context of this task, the Kali Linux operating system is deployed within a distinct VM, enabling seamless communication with other VMs through the established network interface. This configuration ensures that Kali Linux remains a versatile and essential tool for conducting various cybersecurity tasks and assessments.

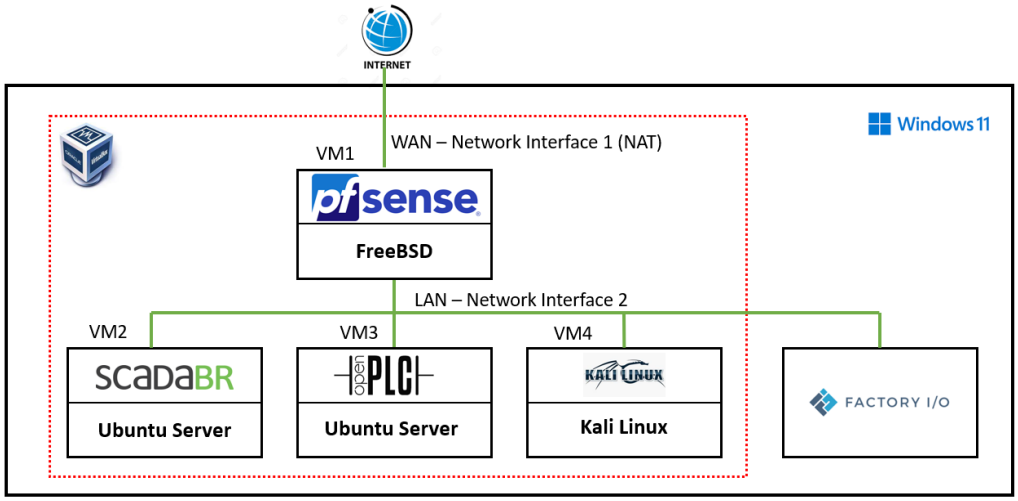

Figure 11: Simulated industrial environment showcasing the interconnections of all tools

Figure 11 illustrates the previously mentioned tools of which the industrial Modbus TCP testbed consists of. As seen, pfSense, SCADABR, OpenPLC and Kali Linux are independent VMs operating inside the same network, created by pfSense. Factory IO runs independently on the host machine, allowing the seamless Modbus TCP connection packet flow between OpenPLC and Factory IO. A screenshot of the scene designed in Factory IO can be found in the next figure, depicting two production lines that lead to two grippers, “baking” the product and then through a common industrial belt is lead the products to the end of the production line. This is only a very simple industrial scenario, showcasing the Modbus TCP communication between a PLC and a Remote Terminal Unit (RTU). This allows us to understand how Modbus TCP works and allows us to further examine the cybersecurity of industrial Modbus TCP communications.

5.4.2.1 Modbus TCP simulation testbed results

The main focus in this first phase of task T5.4, regarding Modbus penetration testing, includes mainly the first step of penetration testing, namely the reconnaissance phase – or information gathering. It is considered to be the most important one, as it provides us with important information that will allow us to map the production environment and its actions, in order to formulate and execute a precise attack. In most of the cases the attacker will probably make use of a hacking OS, such as Kali Linux or Parrot OS. Therefore, the choice of Kali Linux, not only represents an actual attacker, allowing us to familiarize with his/her toolset, but also includes the required software, that will enable us to monitor

and interpret communication and data exchanged between our targets.

Figure 12: Factory IO simulated scene

The initial step was to pinpoint ourselves in the current subnet/network. Though, the simulation testbed is given, and we are aware of all the components the systems consist of, it is always considered best practice to have an overview of the current network we are working in. This assists us not only by providing us the required targets, but also by allowing us to carefully examine the whole network and/or subnet for potential "threats" – tools that would comprise our attacking position. In order to do so, we use the terminal console of Kali Linux and insert the command "ifconfig" that allows us to find the IP address and check the subnet mask. What we notice, is that the subnet mask is 255.255.255.0, meaning that we are currently in a CLASS C network with CIDR /24 notation. This implies that in our network we could potentially have 254 assets, may that be computers, production machines, servers, etc.

```
(base) ┌──(h4x0r㉿HackBox)-[~]
└─$ nmap 192.168.88.0/24
Starting Nmap 7.93 ( https://nmap.org ) at 2023-09-30 13:06 EEST
Nmap scan report for 192.168.88.1
Host is up (0.00098s latency).
Not shown: 997 filtered tcp ports (no-response)
PORT    STATE SERVICE
53/tcp  open  domain
80/tcp  open  http
443/tcp open  https

Nmap scan report for 192.168.88.100
Host is up (0.00071s latency).
Not shown: 996 closed tcp ports (conn-refused)
PORT     STATE SERVICE
135/tcp  open  msrpc
139/tcp  open  netbios-ssn
445/tcp  open  microsoft-ds
3389/tcp open  ms-wbt-server

Nmap scan report for 192.168.88.201
Host is up (0.00070s latency).
Not shown: 997 closed tcp ports (conn-refused)
PORT      STATE SERVICE
22/tcp    open  ssh
8080/tcp  open  http-proxy
20000/tcp open  dnp

Nmap scan report for 192.168.88.207
Host is up (0.00061s latency).
Not shown: 998 closed tcp ports (conn-refused)
PORT     STATE SERVICE
22/tcp   open  ssh
3389/tcp open  ms-wbt-server

Nmap done: 256 IP addresses (4 hosts up) scanned in 8.58 seconds
```

Figure 13: Nmap execution to find IP addresses

By having this initial valuable information, the next step includes the execution of Nmap, a tool that is used to discover hosts and services on a computer network by sending packets and analyzing the responses. As already mentioned, we are somehow familiar with the topology of the simulated network, so one might think that performing this step is not necessary. However, this is far from true, since the nmap allows us to specifically distinguish the production components interacting within the network and will be easier for us, posing as attackers, to monitor the data exchange between them. The command that will give in our terminal is: nmap 192.168.88.0/24. The reply will return with 4 IP addresses – all parts of our simulated environment. More specifically, we have the following results, as presented in Figure 13:

Only with one command we were able to retrieve some basic information: in our network, there are currently 4 assets (SCADABR is not part of this simulation) and we need to understand which component is which.

We already know by "ifconfig" that we previously executed that .207 is our own IP. From ports 53 (dns), 80 (http) & (https), we can understand that the only device that requires dns is actually our pfSense router. As one may notice, by process of elimination, we are able to focus in the two machines-of-interest. However, none of them seems to be a production device using Modbus protocol. By default, Nmap only scans the most popular ports of each protocol, so in the case of Modbus TCP, port 502 TCP was only scanned after the correct configuration of the Nmap command, which is the following:

```
$ sudo nmap -sT -sV -O -A -vv -p- [IP]
```

The above command is executed for both IPs 192.168.88.100 & 192.168.88.201. The following figures (Figure 14 and Figure 15) illustrate the results:

```
# Nmap 7.93 scan initiated Thu Sep  7 13:35:59 2023 as: nmap -sT -sV -O -A -vv -p- -oN /home/h4x0r/Documents/FactoryIO_Nmap_report.txt 192.168.88.100
Nmap scan report for 192.168.88.100
Host is up, received arp-response (0.0014s latency).
Scanned at 2023-09-07 13:36:01 EEST for 216s
Not shown: 65519 closed tcp ports (conn-refused)
PORT     STATE    SERVICE       REASON      VERSION
135/tcp  open     msrpc         syn-ack     Microsoft Windows RPC
139/tcp  open     netbios-ssn   syn-ack     Microsoft Windows netbios-ssn
445/tcp  open     microsoft-ds? syn-ack
502/tcp  open     mbap?         syn-ack
3389/tcp open     ms-wbt-server syn-ack     Microsoft Terminal Services
```

Figure 14: Nmap execution for 192.168.88.100 to find open ports

```
# Nmap 7.93 scan initiated Thu Sep  7 13:37:37 2023 as: nmap -sT -sV -O -A -vv -p- -oN /home/h4x0r/Documents/OpenPLC_Nmap_report.txt 192.168.88.201
Nmap scan report for 192.168.88.201
Host is up, received arp-response (0.0022s latency).
Scanned at 2023-09-07 13:37:39 EEST for 183s
Not shown: 65531 closed tcp ports (conn-refused)
PORT      STATE SERVICE REASON  VERSION
22/tcp    open  ssh     syn-ack OpenSSH 8.9p1 Ubuntu 3ubuntu0.3 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   256 c896f282bfe3a130b212d9caf483c575 (ECDSA)
| ecdsa-sha2-nistp256 AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBDczw4IvqR+3uf9R7+tDm+dFUoNUDemrvF2wJuCK+GyAkrk37MgBtfA/SxMRreX3IRLmCy5QVa2Z56srOxhJrxc=
|   256 b1a96395783e750e4799f678f2bdff7e (ED25519)
|_ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIJaREwwsj9WPbsq9Wktrroj4+H6TKi2psEZDZVGAsyVI
502/tcp   open  modbus  syn-ack Modbus TCP
8080/tcp  open  http    syn-ack Werkzeug httpd 1.0.1 (Python 2.7.18)
| http-title: Site doesn't have a title (text/html; charset=utf-8).
|_Requested resource was http://192.168.88.201:8080/login
| http-methods:
|_  Supported Methods: HEAD OPTIONS GET
|_http-server-header: Werkzeug/1.0.1 Python/2.7.18
20000/tcp open  dnp?    syn-ack
MAC Address: 08:00:27:C4:5D:F1 (Oracle VirtualBox virtual NIC)
Device type: general purpose
Running: Linux 4.X|5.X
OS CPE: cpe:/o:linux:linux_kernel:4 cpe:/o:linux:linux_kernel:5
OS details: Linux 4.15 - 5.6
```

Figure 15: Nmap execution for 192.168.88.201 to find open ports

Now, the recognition is much easier allowing us to know in fact that Factory IO can be only deployed and executed in Windows environment, thus asset with IP 192.168.88.100 is the machine-of-interest. Indeed, we can see that apart from port 502 used in Modbus communication, a variety of Windows services leave no shadow of doubt. So, the asset with IP 192.168.88.201 could only be the OpenPLC

component. It is running on Linux OS using port 502, so now we can be sure about the identity of each machine.

Once the IP addresses of each VM were identified and confirmed, since the followed approach is the white box, Wireshark was executed in order to capture some exchanged packets between the OpenPLC and the Factory IO.

We execute Wireshark from our Kali Linux VM and we start capturing the traffic of interface eth0, which is in promiscuous mode and therefore able to monitor all network traffic. As we are only interested in Modbus communication packets, we use the filter "tcp.port==502". This allows us to follow all data exchanged between those two components.

The communication begins with the standard 3-way handshake, as all TCP connections require to do so. It is imperative to understand which machine initiates the communication, in order to be able to distinguish the Modbus server from the client. Since the one who has the production lines' instructions and transmits them is usually the OpenPLC, we expect to see that it is the one sending the first SYN to FactoryIO. FactoryIO replies with the standard SYN/ACK and as the last part of the handshake requires, OpenPLC responds with an ACK and communication channel is ready for use.

We can notice from the early beginning that there is a loop in the communication, which is described and well-documented in "Introduction to Modbus TCP/IP". In our scenario, Modbus uses two standard Modbus functions: The Read Discrete Inputs and the Write Multiple Coils. Each of the above cycles consists of the query, the response and the ACK part. Having this information, will allow us to monitor the communication easier and more effective.

For the purposes of this deliverable, we examine carefully a "Read Inputs" and a "Write Coils" request. We can easily identify from the Wireshark packet capturing the following information:

1. Source and Destination Port. As already explained, since the OpenPLC initiates the communication, it uses a random port, while the FactoryIO who is listening, awaiting instructions uses port 502.
2. Both for request and response the same Transaction ID is used.
3. For the first time, we are obtaining the RTU's ID number. In our simulation, the ID is 1, but in an actual production environment, that number may vary or even have multiple IDs of multiple RTUs.
4. From both request and response, we can see that the function code is 000 0010, which equals to decimal "2". This function code in Modbus actually means "Read Discrete Input".
5. Last, but not least, the request requires 16 bits (8 and 8 bits) starting at 0 (as a Reference number). In our environment, we configured OpenPLC and FactoryIO to use 2 production lines, each of 8 total inputs. So, the first number (query digits 1 & 2), refers to the inputs of the first production line, while the second one (query digits 3 & 4), refers to the inputs of the second production line.

So, in this example, we see numbers 00_{16} 44_{16} and we know that the first number refers to the 1st production line. As a first step, we convert the number to binary and 00_{16} <==> 00000000_2

We know that the first bit starts at position 0 (reference number), so we have the following bits in the following positions:

Position	0	1	2	3	4	5	6	7

Bits	0	0	0	0	0	0	0	0

We apply the same technique for the second production line. First converting the 44_{16} to binary, gives us the following bits: 010001002

Position	0	1	2	3	4	5	6	7
Bits	0	1	0	0	0	1	0	0

The response of the above is able to tell us what the status of each input is (1=on, 0=off), however we cannot retrieve the information of which input corresponds to which sensor. This is an issue that we will return to it after a while.

At this point we need the assistance of OpenPLC. From the captured values and by using the registers, we can understand which sensor corresponds to each component in our production lines. For example, if we have a payload of 0111011001000000_2, we can map it as follows:

Register	%IX100.0	%IX100.1	%IX100.2	%IX100.3	%IX100.4	%IX100.5	%IX100.6	%QX100.7	%IX101.0	%QX101.1
Bit	0	1	1	1	0	1	1	0	0	1

Now, we can see the exact state of all our sensors. It is worth mentioning, that the last 6 bits are not used, since in our schema, both production lines use only 10 sensors. Yet, we still don't know, which value corresponds to each component. We can use the assistance of OpenPLC, that reveals to us the mapping of the Registers with the Point Names:

I_Z01BC01ArrivedSens	BOOL	%IX100.0	No
I_Z01BC02EntrySens	BOOL	%IX100.1	No
I_Z01MC01_Open	BOOL	%IX100.2	No
I_Z01MC01_Bussy	BOOL	%IX100.3	No
I_Z01MC01_Error	BOOL	%IX100.4	No
I_Z01MC02_Open	BOOL	%IX100.5	No
I_Z01MC02_Bussy	BOOL	%IX100.6	No
I_Z01MC02_Error	BOOL	%IX100.7	No
I_Z01BC03ArrivedSens	BOOL	%IX101.0	No
I_Z01BC05EntrySens	BOOL	%IX101.1	No

Figure 16: Mapping of the registers with the point names

Now, the idea starts getting clearer. We are able to tell what the sensors are and in what production

line they belong. From the descriptive Point Names, we can always assume what are the performed actions. However, we need to understand which sensor belongs to each production line. In order to achieve this task, we need to correspond the point names with the actual components by using the FactoryIO and matching the values. By the end of this task, we will know with certainty how the sensors are correlated to the simulated environment. Following the next reconnaissance phase, we analyze the second request, namely the "WRITE COILS" function.

By analyzing the Modbus packets correlated to the "WRITE COILS" function, the below information could be retrieved:

1. Source and Destination Port. Same applies as in the previous example: the OpenPLC initiates the communication by using a random port and the FactoryIO, who is listening, awaits instructions in port 502.
2. Both for request and response the same Transaction ID is used.
3. In this field, we are expecting to see the RTU's ID number, which is 1, but -as already mentioned- in an actual production environment, that number may vary or even have multiple IDs of multiple RTUs.
4. From both the request and response, we can see that the function code is 000 1111, which equals to decimal "15". This function code in Modbus actually means "Write Multiple Coils".
5. Last, but not least, the "WRITE COILS" function requires 14 bits (7 and 7 bits) starting at 0 (as a Reference number). We configured OpenPLC and FactoryIO in our environment to use two production lines, each of 7 total outputs. So, the first number refers to the outputs of the first production line, while the second one refers to the outputs of the second production line.

In this case, we need to examine different frames of the captured packets in order to better understand the use of the actuators. This will allow us to see the changes as they occur and map them, in order to find out which actuator corresponds to which component in FactoryIO. So, in this example, let's see the output for both production lines:

Register	%QX100.0	%QX100.1	%QX100.2	%QX100.3	%QX100.4	%QX100.5	%QX100.6	%QX100.7	%QX101.0	%QX101.1	%QX101.2	%QX101.3	%QX101.4	%QX101.5	%QX101.6
Bit	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0

Now, we know the exact state of all our actuators. But we still do not know, which value corresponds to each component. We can use the assistance of OpenPLC, that will allow us to map the registers to the point names:

Q_Z01BC01	BOOL	%QX100.0	No
Q_Z01BC02	BOOL	%QX100.1	No
Q_Z01BC03	BOOL	%QX100.2	No
Q_Z01BC04	BOOL	%QX100.3	No
Q_Z01BC05	BOOL	%QX100.4	No
Q_Z01BC06	BOOL	%QX100.5	No
Q_Z01BC07	BOOL	%QX100.6	No
Q_Z01MC01Start	BOOL	%QX100.7	No

Q_Z01MC01Stop	BOOL	%QX101.0	No
Q_Z01MC01Reset	BOOL	%QX101.1	No
Q_Z01MC01_ProducingLids	BOOL	%QX101.2	No
Q_Z01MC02Start	BOOL	%QX101.3	No
Q_Z01MC02Stop	BOOL	%QX101.4	No
Q_Z01MC02Reset	BOOL	%QX101.5	No
Q_Z01MC02ProducingLids	BOOL	%QX101.6	No

Figure 17: Mapping of the registers with the point names

As in the previous step, we are now in position to distinguish the actuators and map them with the production line. From the descriptive point names, we can always assume which are the performed actions. However, we need to understand which actuator belongs to each production line. In order to perform this task, we need to correspond the point names with the actual components by using the FactoryIO and matching the values.

At this point, it is worth mentioning that in all Modbus communications (TCP or RTU), there are no security mechanisms implemented by design, meaning that someone could have full access to information leading to reading and writing all data. Consequently, the packet sniffing with Wireshark revealed several information that could be used to compromise the Modbus TCP communication. Based on the information gathered using the Wireshark tool and taking advantage of the known information on the sensors and actuators in Factory IO, also including the mapping of the components used in the scene, we are able to recognize the packet sequences they receive. By listing the exact bits they receive, the researcher has the ability to intervene and execute a Man-in-The-Middle (MiTM) attack and inject different bits than those sent regularly. This action was conducted using the Metasploit framework of Kali. More specifically, it provides a Modbus tool that allows the user, first of all to read the registers and then perform the writing of a different byte. This leads to the malfunction of the production line, causing several damage and financial loss. Besides Metasploit, another Modbus penetration testing tool was examined, called Smod, but the attacks conducted through this tool did not affect the operations of the testbed at all.

All in all, the above-mentioned work regarding the Modbus communication protocol was submitted as a research paper and was accepted after review for presentation at the IEEE Conference on Standards for Communications and Networking (CSCN 2023) with the title of the paper being “Securing Modbus TCP Communications in I4.0: A Penetration Testing Approach Using OpenPLC and Factory IO”.

5.5 Zero-SWARM exploitation and vulnerability report

Figure 18 illustrates the use of the opcua_server_config module, which is used to obtain security related information for OPC UA server instances, already being accessed. The module will report available endpoints, request information about other servers this server knows about and iterate all nodes looking for writable nodes. As depicted below, the steps to use the opcua_server_config module

include:

1. Start msfconsole
2. use auxiliary/scanner/opcua/opcua_server_config
3. set rhosts <IP>
4. set rport <port>
5. run

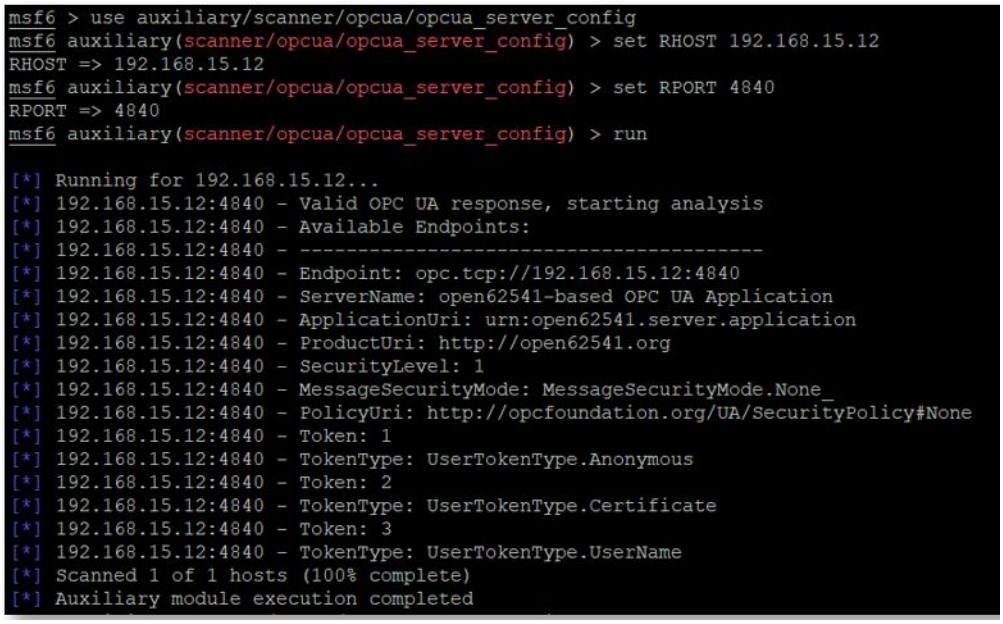

Figure 18: Metasploit OPC-UA scanning process in progress

Figure 19 shows the available OPC-UA modules from the Metasploit platform that can be used to examine an OPC-UA communication.

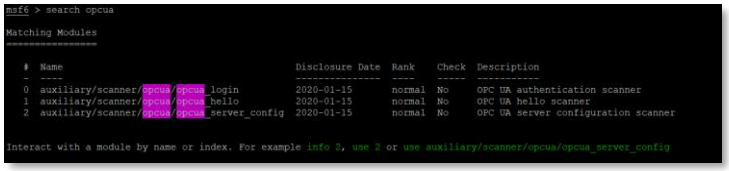

Figure 19: Available Metasploit OPC-UA related modules

For the purpose of this simulation environment, the UaExpert, a fully featured OPC-UA client was set up and used. The basic framework of UaExpert includes general functionality like certificate handling, discovering UA Servers, connecting with UA Servers, browsing the information model, displaying attributes and references of particular UA Nodes. As depicted in Figure 19, the Project pane shows the connected UA Servers and the open document plugins. The Address Space pane shows the UA Servers information model. Depending on the Node selected in the Browser the Attribute and Reference Windows show the attribute of the selected Node and its references within the meshed network of the server's address space.

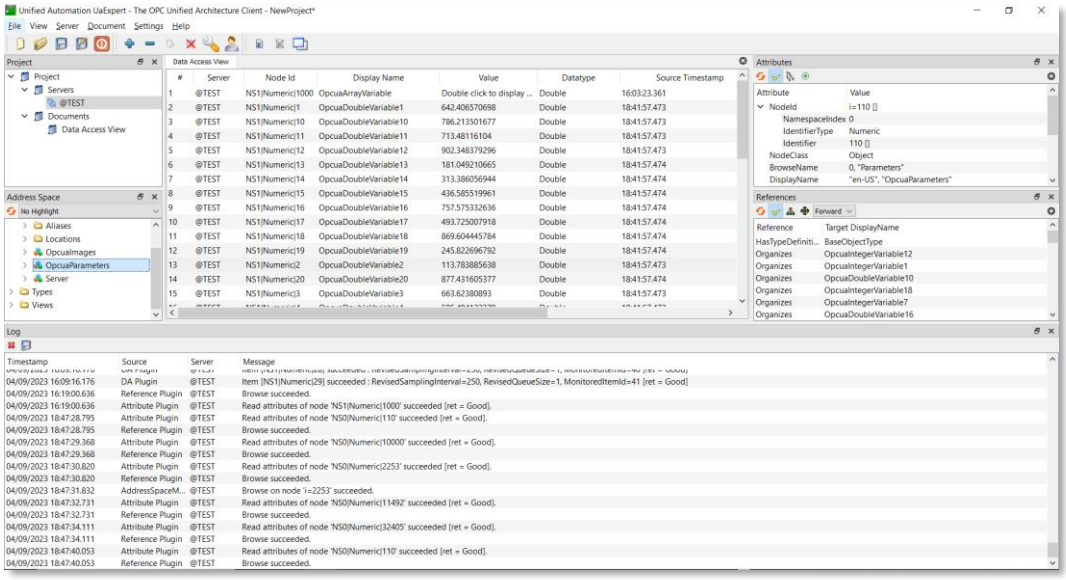

Figure 20: Configuration and User Interface of the OPC-UA client

6 Hypothesis testing plugin

6.1 Methodology used for the hypothesis testing realization

A brief presentation of the SigClust method with Soft Thresholds, as shown in [45] follows: Let X=[x_1,x_2,...,x_n], x∈R^d, be a dataset of n observations each containing the values d different KPIs. The method starts from the null hypothesis that the data of X come to form a single multivariate Gaussian distribution N(μ,Σ). A test level α, e.g., α=0.95 is pre-specified to finally test the Hypothesis. Let C_1 and C_2 be two disjoint sets resulting from the application of a clustering of the data points contained in X i.e., C_1∪C_2={1,2,...,n}. Then, an indicator of the strength of the clusters can be attained by the Cluster Index (CI), that is used as the test statistic of the method:

$$CI = \frac{\sum_{k=1}^{2}\sum_{i\in C_k}||x_i - \bar{x}^{(k)}||^2}{\sum_{i=1}^{n}||x_i - \bar{x}^{(k)}||^2}$$

where $\bar{x}^k$ is the mean of cluster $k \in [1,2]$ while $\bar{x}$ is the overall mean. Estimate values $(\widehat{\lambda_1},\ldots,\widehat{\lambda_d})$ for the eigenvalues of Σ must be computed. Let $(\widetilde{\lambda_1},\ldots,\widetilde{\lambda_d})$ be the eigenvalues of the sample covariance matrix. The covariance matrix Σ can be written as

$$\Sigma = \Sigma_0 + \sigma_N^2 I$$

for a low-rank positive semi-definite matrix Σ_0. Let W_0 be a positive semi-definite matrix W_0 with rank(Σ_0)=rank(W_0). Then the precision matrix C can be defined, where:

$$C \equiv \Sigma^{-1} \equiv (\Sigma_0 + \sigma_N^2 I)^{-1} = \frac{1}{\sigma_N^2} I - W_0 \ (1)$$

To estimate Σ, the negative log-likelihood is minimized to using C and the sample covariance

$$\tilde{\Sigma} = \frac{(X-\bar{X})(X-\bar{X})^T}{n},$$

to yield

$$argmin_c[-log|C| + trace(C\tilde{\Sigma})],$$

subject to (1) and $C, W_0 \succcurlyeq 0$.Let $M \geq 0$, be a tuning parameter. An additional constrained is set to control the signal versus the noise of the data:

$$trace(W_0) \leq M$$

Finally,

$$\widehat{\lambda_j} = \begin{cases} \tilde{\lambda}_k - \tau, \text{if } \tilde{\lambda}_k > \tau + \sigma_N^2 \\ \sigma_N^2, \text{if } \tilde{\lambda}_k \leq \tau + \sigma_N^2 \end{cases}$$

where τ is obtained by solving

$$\sum_{k=1}^{d} \left(\frac{1}{\sigma_N^2} - \frac{1}{(\tilde{\lambda}_k - \tau)_+} \right)_+$$

Using the computed eigenvalues, the theoretic optimal CI value is obtained by

$$TCI = 1 - \frac{2}{\pi} \frac{max((\widehat{\lambda_1}, \ldots, \widehat{\lambda_d}))}{\sum_{i=1}^{d} \lambda_i}$$

The rest of the process has four steps:

1. Data from the null distribution is simulated: $(x_1, \ldots, x_d)$ are independent with $x_j \sim N(0, max(\hat{\lambda}_j, \hat{\sigma}_N^2))$.

2. This data is clustered using the k-means algorithm with k = 2; the corresponding CI value is calculated.

3. By repeating steps 1 and 2 a large number of times, an empirical distribution of the values of CI is obtained.Using the CI values obtained by the simulation, calculate a p −value for the CI value of X.

4. A conclusion can be derived based on test level α.

By using the p-value from the final step the following hypothesis is answered: H_o The clusters come from the same distribution $(p \leq 0.05)$ or H_1 The clusters come from a different distribution $(p \geq 0.05)$.

To apply this method to distinguish the difference between different mitigation action sets the following procedure followed: Let $X = [x_1, x_2, \ldots, x_n]$, $x \in \mathbb{R}^d$, be a data set of n historical observations each containing the values d different KPIs with x_n being the latest observation. Let x_{n+1} be the KPI values occurring from modifying x_n. Some clustering algorithm is applied to all data, resulting in a partition of clusters $C = \{C_1, C_2, \ldots C_j\}, j \leq n$ for all data points. Let C_A,C_B be the clusters, the data-points x_n and x_{n+1} belong to. If $C_A = C_B$ the case is trivial, and no testing is required since the points belong to the same cluster. Else, let $x_{cluster} \subset X$ be the subset of all points that belong either in C_A or C_B. Then, the Statistical Significance of Clustering methodology described above is applied as described, answering the question: "Are C_A or C_B different in terms of their underlying distribution and is this difference statistically significant?".

The figure below (Figure 21) illustrates the overview of the hypothesis testing module, which will be developed in the context of this task T5.4. The evaluation of the performance of this tool will be performed in the second phase of this task, taking advantage of the simulation platform developed and described in section 7.

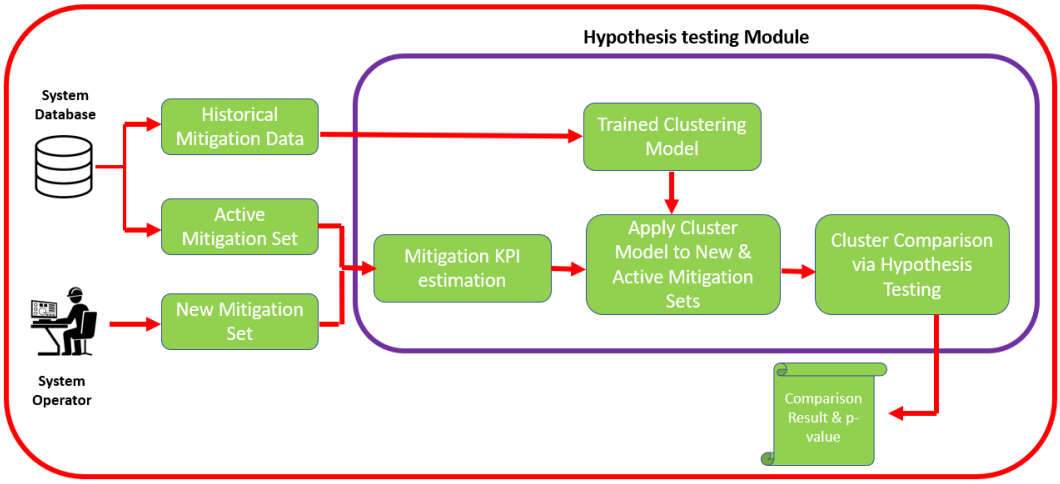

Figure 21: High level overview of the Hypothesis testing module

7 Simulation platform for evaluation

In essence, the penetration test in question is a meticulously orchestrated evaluation of the inter-device communication within the IEC 61499 environment, employing a carefully designed test application and SoftPLC emulations. By continuously monitoring KPIs and scrutinizing the behaviour of these critical devices, this test aims to provide invaluable insights into the reliability and security of industrial automation systems, ultimately contributing to enhanced system resilience and cybersecurity.

7.1 Test application design with IEC61499 cross communication over UDP

In the context of the penetration test conducted in T5.4, a specialized test application is meticulously crafted within an IEC 61499 environment. This test application is ingeniously designed and consists of two crucial Function Blocks, which are classified as CATs (Composite Automation Types). Within the test environment, these CATs are distinctly labelled as PLC1 and PLC2. The fundamental function of this test application is to evaluate and scrutinize the inter-device communication between two IEC 61499 devices, a task of paramount importance in the realm of industrial automation and control systems. The objective is to ensure the robustness and security of data exchange between these critical components.

As illustrated in the Figure 22 provided, PLC1 takes on the role of the initiator in this orchestrated interaction. At precise one-second intervals, PLC1 diligently transmits a signal, encapsulating critical data, to its counterpart, PLC2. The ingenuity of this setup lies in PLC2's responsive behaviour. It promptly reciprocates by reflecting the received signal back to PLC1. This bi-directional communication is pivotal in assessing not only the reliability but also the integrity of the data exchange process.

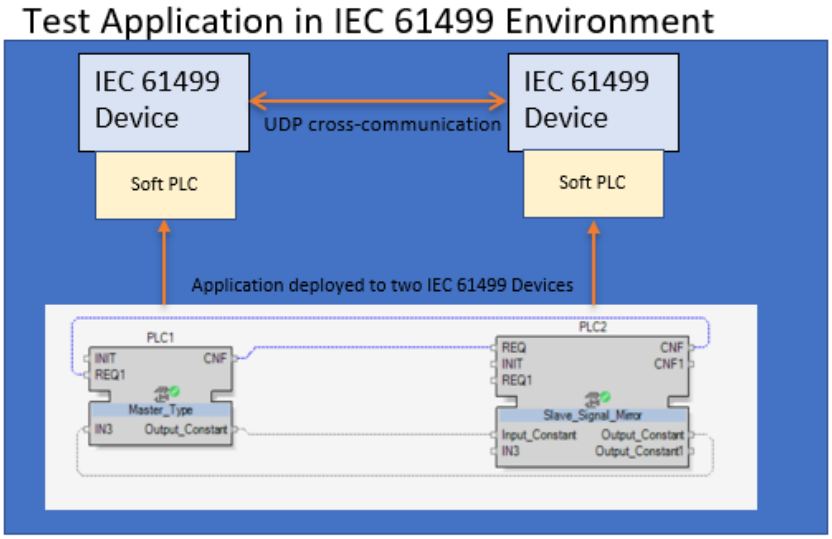

Figure 22: Test application in IEC61499 Environment - UDP cross communication

To facilitate this distributed communication of an IEC61499 platform and emulate real-world scenarios, the test harness leverages a IEC61499 UDP (User Datagram Protocol) communication node. Within this context, the SoftPLC, an emulation of a real-world Programmable Logic Controller (PLC) operating within the IEC 61499 environment, takes centre stage. This software-based PLC replica serves as the backbone of the testing infrastructure, enabling the seamless exchange of data between the two CATs (PLC1 and PLC2). These two SoftPLCs, acting as master and slave devices, engage in synchronized communication over the UDP protocol.

As the test application is prepared for deployment, it is meticulously mapped to two instances of the SoftPLC. This pairing simulates the actual deployment of PLCs in an industrial setting, allowing for a comprehensive evaluation of their performance and security in a controlled environment. At the heart of this assessment lies the monitoring and measurement of Key Performance Indicators (KPIs). These metrics are diligently tracked throughout the testing process to assess and quantify the behaviour of the PLCs under scrutiny. The KPIs encompass various aspects of the communication process, including latency, throughput, CPU and memory consumption and error rates, shown in Figure 23:

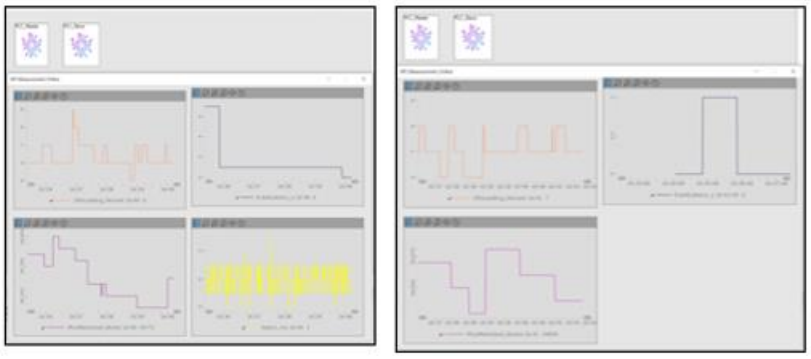

Figure 23: Test results for KPI measurement on the IEC61499 environment

Enriching analytical capabilities, the test application provides an intuitive Human-Machine Interface (HMI) that acts as the window into the inner workings of the IEC61499 automation platform and communication system. This real-time HMI visualization empowers engineers, and operators to gain immediate access to KPIs and test results. It is a tool of utmost significance in our pursuit of

understanding, optimizing, and safeguarding the performance and security of industrial automation systems.

7.2 Test application with MQTT communication between two IEC61499 platforms

This test application part within the MQTT communication node is an intricate dance of data orchestrated through the MQTT protocol. It not only showcases the elegance of MQTT's publish/subscribe methodology but also underscores our commitment to creating efficient and dependable communication systems for industrial automation and beyond.

Within the MQTT communication node, our test application operates seamlessly within the MQTT protocol framework. MQTT, known for its lightweight and efficient messaging paradigm, employs a publish/subscribe method to facilitate communication between two MQTT clients. Each of these clients boasts the capability to publish data under a designated Topic onto an MQTT broker and, conversely, subscribe to data under specific Topics through the same MQTT broker. It's a versatile system that offers flexibility in routing and managing data flows.

The MQTT broker, a pivotal component in this communication architecture, can be deployed either within the confines of the IEC 61499 environment or externally, depending on the specific testing requirements and architectural considerations. This flexibility ensures that our testing environment aligns precisely with the real-world scenarios our application is designed to address.

In the IEC 61499 engineering environment, we've ingeniously integrated Service Interface Function Blocks (SFBs) to streamline the process of creating applications that facilitate communication between two MQTT clients. This sophisticated orchestration enables a seamless flow of data between these clients.

Now, let's delve into the roles played by our CATs within this MQTT communication ecosystem. In the master-PLC, we've incorporated a Signal Sender Function Block (FB) equipped with MQTT client capabilities. This FB serves as the initiator, responsible for transmitting a signal at precise one-second intervals. This signal is encapsulated within the MQTT_PUBLISH SFB and is delivered to the MQTT broker under a distinct Topic name. On the flip side, our slave-PLC houses a Signal Receiver Function Block (FB) also equipped with MQTT client functionality. This FB assumes the role of the receiver within this dynamic communication dance. It subscribes to data using the very same Topic name used for publication via the MQTT_SUBSCRIBE SFB. Once the data is received from the MQTT broker, it is dutifully relayed back to the master-PLC CAT, completing the bi-directional data exchange.

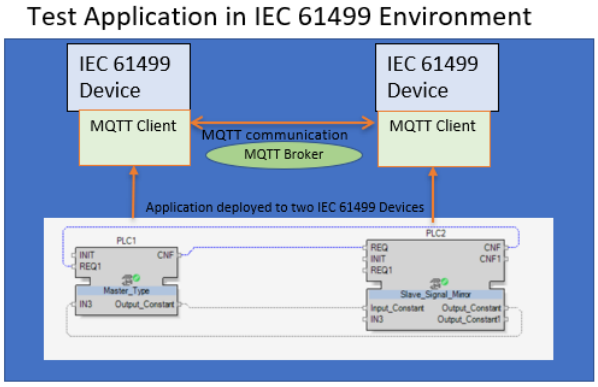

Figure 24: Test application in IEC 61499 Environment - MQTT communication

This synchronized communication dance, occurring at the cadence of one signal per second, provides us with a unique opportunity to meticulously observe and evaluate the performance and efficiency of our MQTT-based communication system. We will closely monitor the same metrics, as mentioned above, to ensure that the test application operates robustly and reliably.

7.3 Test application with OPC-UA communication between two IEC61499 platforms

Within the context of Zero-SWARM, we're harnessing the power of OPC-UA as our interface of choice for the integration of and communication with IT software modules. This versatile protocol plays a pivotal role in facilitating communication in two crucial directions within our industrial ecosystem.

Firstly, OPC-UA serves as the linchpin between the Production line and the IEC61499 control application. In this capacity, it enables seamless and robust communication, primarily for monitoring purposes. The Production line communicates with the Edge Gateway through OPC-UA, providing real-time insights into the operational status and performance metrics of the production processes. This flow of information is essential for monitoring and optimizing the efficiency and productivity of the production line.

Secondly, OPC-UA comes into play once again, this time serving as the communication conduit between the Edge Gateway nodes and an AI software node. This interaction is central to the integration of AI and automation, where data from the industrial processes is shared with AI algorithms for advanced analysis and decision-making. The bidirectional flow of data between the IEC61499 node, powered by the IEC61499 automation platform, and the AI software node ensures that intelligent insights are gained and acted upon in real-time.

Now, let's zoom in on the specific test application scenario you've described. Here, the master PLC is the focal point, equipped with an OPC-UA client. It assumes the role of the data initiator, sending values at precise one-second intervals. These values encapsulate critical data related to the production processes, which are essential for monitoring and decision-making, shown in Figure 25

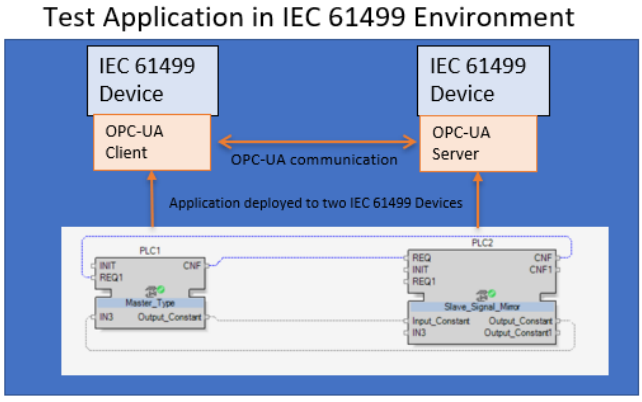

Figure 25: Test application in IEC 61499 Environment – OPC-UA communication

On the other side of the equation, the slave PLC plays the role of the OPC-UA server. It receives the incoming signals from the master PLC and mirrors them back to their source—the master PLC. This bi-directional communication loop is instrumental in not only evaluating the efficiency and reliability of the OPC-UA communication but also in assessing the responsiveness and robustness of the overall system.

Just as with other communication methods in our testing suite, we meticulously measure the same set of KPIs. These KPIs include parameters such as responsiveness of the OPC-UA communication. By scrutinizing these metrics, we gain invaluable insights into the performance of the OPC-UA-based communication system, which is central to the smooth operation of Zero-SWARM.

In summary, our utilization of OPC-UA as a communication interface in Zero-SWARM underscores its versatility and reliability in connecting production lines, Edge Gateway nodes, IEC61499 automation platforms, and AI software nodes. This orchestrated symphony of data exchange ensures that industrial processes run smoothly, are closely monitored, and can benefit from intelligent insights and decision-making capabilities.

7.4 Test application with Modbus TCP (Master/Slave) communication between two IEC61499 platforms

Our aim is to introduce Modbus as an additional load into our existing test application, effectively simulating a fieldbus connection. This strategic addition enables us to probe the system further, scrutinizing potential side effects while meticulously measuring the Key Performance Indicators (KPIs) of the IEC61499 automation platform.

Modbus, a versatile application layer messaging protocol, assumes a pivotal role in this endeavour. It facilitates client and server communications between devices, seamlessly bridging the gap on different types of buses or networks. Modbus's robust architecture offers a plethora of services, each elegantly specified by function codes, making it an ideal candidate for our communication needs.

In our sophisticated EcoStruxure Automation Expert environment, the configuration of Modbus is orchestrated with precision through a hierarchical structure consisting of three interconnected layers. Each layer, bound by a parent-child relationship, plays a distinctive role in ensuring seamless communication and parameterization:

Bus Layer: At the foundational level, the Bus layer defines the communication type between the client and server and outlines the blueprint for how communication will be orchestrated. It sets the stage for a well-coordinated dance of data exchange. Within EcoStruxure Automation Expert, we deploy both master and slave configurations for our test application. Here, one entity serves as an external simulator, intrinsically penetrating the system with predefined values. This dynamic interplay is central to our assessment of system behaviour under load.

Modbus Device Layer: Building upon the Bus layer, the Modbus Device layer takes centre stage by defining the specific communication settings between itself and the counterpart Bus (e.g., IP address, station ID, etc.). It acts as the bridge between the Bus and the actual data exchange. In our EcoStruxure Automation Expert setup, various types of Modbus devices are at our disposal, and for our test application, TCP is leveraged, ensuring robust and reliable communication.

Modbus Registers Layer: At the topmost layer of this hierarchy, the Modbus Registers layer comes into play. It meticulously defines all variables that are to be communicated, specifying where and how they are transmitted and clarifying their data types. This layer essentially encapsulates the core of the data exchange process. In EcoStruxure Automation Expert, we are presented with a range of Modbus register types, each with its own unique characteristics. Synchronizing the compatibility of the Modbus-simulator used within the system is of paramount importance at this layer to ensure seamless communication.

As we embark on this journey of integrating Modbus into our test application, we are not only adding an additional layer of complexity but also a wealth of opportunities for evaluating system performance and robustness. By meticulously measuring KPIs and probing for side effects, we are poised to gain valuable insights into the behaviour of the IEC61499 automation platform under various communication scenarios, bolstering its reliability and resilience in industrial automation settings.

7.5 Combined load with external clients interacting with the IEC61499 platform

Expanding our scope of penetration testing within the realm of the IEC61499 automation platform, we are poised to embark on another scenario of critical assessment. In this scenario, we introduce external MQTT and OPC-UA clients into the equation, further intensifying the testing environment. This strategic move aims to scrutinize and evaluate the behaviour of the IEC61499 platform under the simultaneous influence of multiple loads, including the existing cross-application load and the Modbus simulation load.

This multifaceted test scenario is designed to assess the IEC61499 platform's resilience, performance, and security when subjected to a comprehensive array of loads, both internal and external. The integration of MQTT and OPC-UA clients into the testing environment brings a new layer of complexity, mirroring real-world scenarios where diverse communication protocols and sources converge within an industrial setting.

To execute this scenario effectively, we rely on the capabilities of our penetration testing tools, as illustrated in Figure 27.

These tools are instrumental in simulating the behaviour of external MQTT and OPC-UA clients, thereby generating additional load on the IEC61499 platform. The external load is meticulously controlled and manipulated to emulate various communication patterns, message volumes, and frequencies, akin to what might be encountered in a production environment.

The orchestration of this comprehensive test scenario demands a fine balance between all the loads involved.

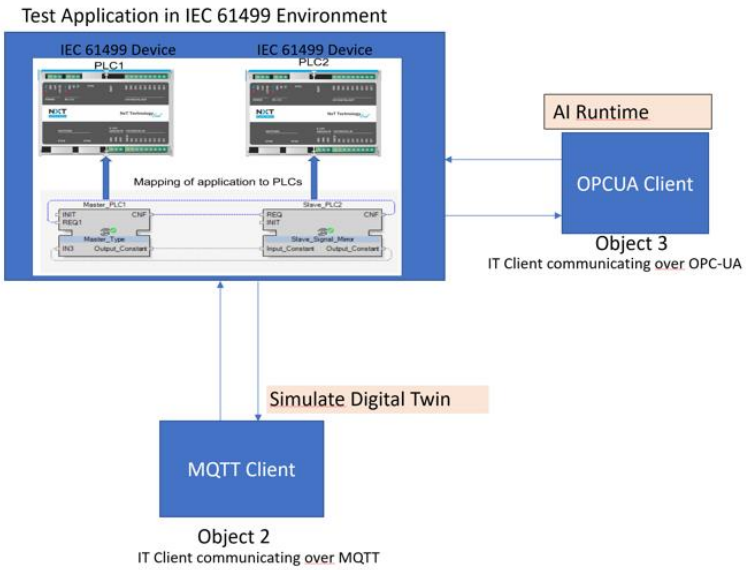

Figure 26: Test application in IEC 61499 Environment – combined load

7.6 Cybersecure and non-cybersecure use-cases in the IEC61499 automation platform

The schematic diagram in Figure 27 below provides a visual representation of the security measures implemented within our system. It uses color-coded lines to convey the status of security for various connections, offering a quick and intuitive overview of our security infrastructure.

Green Line Connections: These connections are depicted in green, signifying that all components linked by these lines benefit from robust TLS (Transport Layer Security) encryption. TLS is a well-established cryptographic protocol that ensures secure and private communication between devices. The presence of green lines assures us that data flowing through these pathways is protected from unauthorized access and tampering.

Red Line Connections: Conversely, the red lines indicate connections between devices that are not yet secured. This serves as a visual cue to draw attention to potential vulnerabilities in these communication channels. It is a reminder that further security measures may be needed to safeguard these connections adequately.

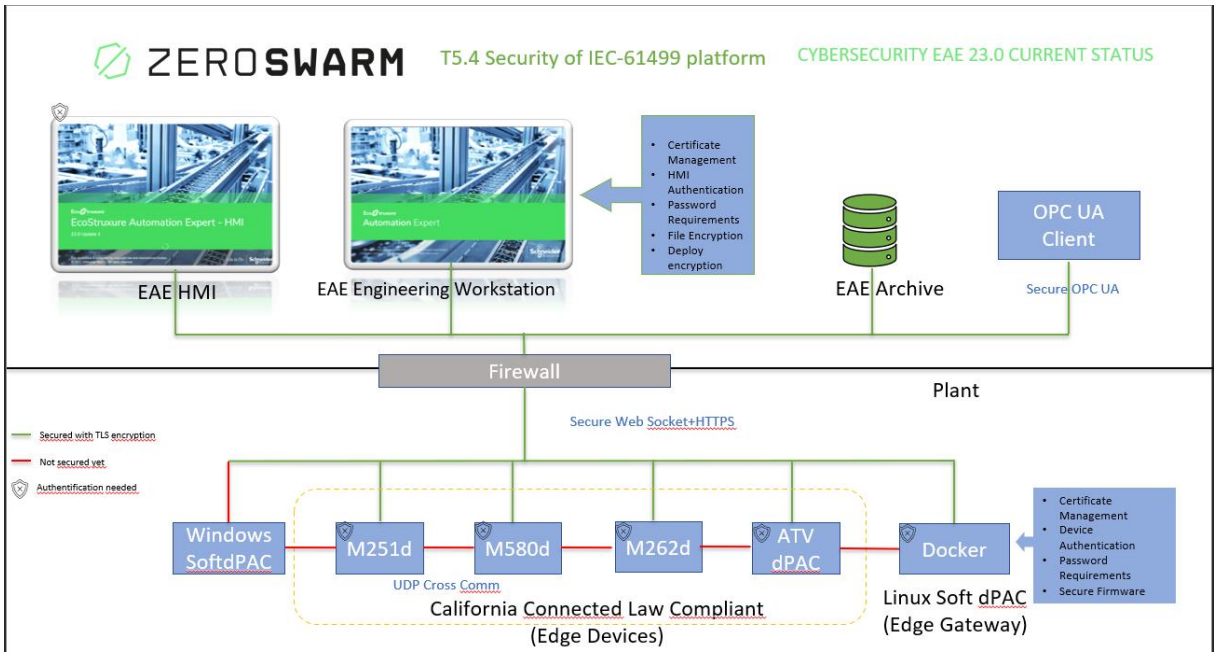

Figure 27: Cybersecure and non-cybersecure use-cases in the IEC61499 automation platform

Additionally, to bolster our security posture during penetration tests, we address specific scenarios beyond the core IEC61499 cross-communication and external client communication discussed in previous sections. These scenarios include:

Engineering Workstation Security:

Certificate Management: Our engineering workstation is fortified with robust certificate management capabilities. This ensures that digital certificates, a fundamental component of secure communication, are handled effectively. Certificates play a pivotal role in verifying the authenticity of devices and users, enhancing the overall security of our solution.

Password Requirement: A stringent password requirement is in place to secure access to the

engineering workstation. When creating a solution within the IEC61499 environment, users are mandated to establish an account with a unique USER ID and password. This initial authentication step is a crucial layer of defense against unauthorized access.

Deployment Security: During the deployment of our IEC61499 test application onto IEC61499 devices, a robust access permission mechanism is enforced. Devices are explicitly asked for access permission before the deployment process can proceed. Furthermore, the communication during deployment is encrypted, ensuring that the application reaches its destination securely and without interference.

HMI Authentication and Security:

Credential-Based Authentication: Our Human-Machine Interface (HMI) incorporates a robust authentication mechanism. When the HMI is initiated, it prompts users to enter the credentials established at the outset of the solution creation process. Without successful authentication, access to the HMI is denied. This stringent authentication layer ensures that only authorized personnel can interact with the HMI, enhancing the security of our system.

Encrypted Visualization: Within the HMI, the values and data changes of our test application are meticulously visualized. Importantly, this communication is encrypted, ensuring that data remains confidential and protected from eavesdropping or tampering.

Archiving Security:

Secure Data Archiving: When archiving values into a historical database, robust encryption measures are employed. This means that the data inputs and outputs of the function blocks within our test application can be configured for secure archiving. This ensures that historical data remains confidential and tamper-proof, safeguarding the integrity of our records.

Incorporating these security measures across various aspects of the IEC61499 system architecture underscores our commitment to protecting data, ensuring authorized access, and maintaining the confidentiality and integrity of our industrial automation solution. These security enhancements, alongside our rigorous penetration testing efforts, are pivotal in fortifying our system against potential vulnerabilities and threats.

8 Conclusion

In the scope of D5.4, an initial approach was examined in the domain of penetration testing, trying to identify potential threats in two industrial communication protocols, namely OPC-UA and Modbus TCP. Two separate simulation testbeds were deployed in order to validate the proposed approaches. Significant findings came up, revealing the need of adopting cybersecurity mechanisms inside industrial environments. The Modbus communication results were also submitted to an IEEE conference [51] and accepted for presentation, proving in this stage of the project that the cybersecurity activities in the scope of this task are novel. The intention of this task is to also try to automate penetration testing, in order to make some processes more straightforward with the adoption of AI algorithms. Furthermore, the development of an IEC61499 simulation platform will allow partners to further research on the security of industrial communication protocols, such as Modbus, OPC-UA, MQTT and IEC61499 communications in an environment, closer to a real industrial production line. Finally, with regards to the hypothesis testing module, some initial architectural designs have been defined, allowing the development of the module. All findings will be reported

thoroughly in the next version of this deliverable, D5.9, the successor of D5.4. On the other hand, we are in contact with Node leaders and Trial leaders in order to assess the possibility of working closely with the project trials and have the ability to validate our modules in the trials as well, in the context of WP6.

References

[1] V. Manes, H. Han, C. Han, S. Cha, M. Egele, E. Schwartz y W. Maverick, «Fuzzing: Art, Science, and Engineering» IEEE Transactions on Software Engineering, pp. 2312-2331, 2021.

[2] Y. Yu, Z. Chen, S. Gan y X. Wang, «SGPFuzzer: A State-Driven Smart Graybox Protocol Fuzzer for Network Protocol Implementations» IEEE Access, vol. 8, pp. 198668-198678, 2020.

[3] O. Foundation, «Unified Architecture - OPC Foundation» 2008. [En línea]. Available: https://opcfoundation.org/about/opc-technologies/opc-ua/. [Accessed: 17 08 2023].

[4] J. Furbush, «QUÉ ES LA OPC UA Y POR QUÉ ES FUNDAMENTAL PARA LA AUTOMATIZACIÓN INDUSTRIAL,» 22 10 2019. [En línea]. Available: https://www.cognex.com/es-mx/blogs/machine-vision/why-opc-ua-is-essential-for-factory-automation. [Accessed: 25 07 2023].

[5] R. A. T. U. D. a. F. Fraunhofer IOSB, «Open62541» 2023. [En línea]. Available: https://www.open62541.org/. [Accessed: 17 08 2023].

[6] O. project, «Open62541» 10 05 2023. [En línea]. Available: https://github.com/open62541/open62541. [Accessed: 17 08 2023].

[7] 2020, «A Modern Protocol: OPC UA vs MQTT», https://embeddedcomputing.com/technology/security/iec-iso-other-standards/a-modern-protocol-opc-ua-vs-mqtt

[8] Emma McMahon, Mark Patton, Sagar Samtani, Hsinchun Chen, «Benchmarking Vulnerability Assessment Tools for Enhanced Cyber-Physical System (CPS) Resiliency».

[9] Yuqi Chen, Bohan Xuan, Christopher M. Poskitt, Jun Sun, Fan Zhang, «Active Fuzzing for Testing and Securing Cyber-Physical Systems», https://cposkitt.github.io/files/publications/active_fuzzing_issta20.pdf

[10] Lucas McDonald, Muhammad Ijaz Ul Haq, Ashley Barkwort, «Survey of Software Fuzzing Techniques»,

[11] Baijie Yang Shaoyun Ge Hong Liu Junkai Li, «Resilience assessment methodologies and enhancement strategies of multi-energy cyber-physical systems of the distribution network».

[12] CPNI Centre for the Protection Of National Infrastructure, «CYBER SECURITY ASSESSMENTS OF INDUSTRIAL CONTROL SYSTEMS », https://www.ccn-cert.cni.es/publico/InfraestructurasCriticaspublico/CPNI-Guia-SCI.pdf

[13] OPC foundation, Security Working Group, «Practical Security Recommendations for building OPC UA Applications», https://opcfoundation.org/wp-content/uploads/2017/11/OPC-UA-Security-Advise-EN.pdf

[14] Alessandro Erba, Anne Müller, Nils Ole Tippenhauer, «Security Analysis of Vendor Implementations of the OPC UA Protocol for Industrial Control Systems», https://arxiv.org/pdf/2104.06051.pdf

[15] MIGUEL BIGUEUR, «VULNERABILITY & PATCH MANAGEMENT PROCESS», https://miguelbigueur.com/2016/12/24/vulnerability-patch-management-process/

[16] «What Is Fuzz Testing?», https://www.code-intelligence.com/what-is-fuzz-testing

[17] Sanaz Sheikhi, Edward Kim, Parasara Sridhar Duggirala, Stanley Bak, «Coverage-Guided Fuzz Testing for Cyber-Physical Systems», https://stanleybak.com/papers/sheikhi2022iccps.pdf

[18] Dimitrios Serpanos, Konstantinos Katsigiannis, «Fuzzing: Cyberphysical System Testing for Security and Dependability»

[19] Maialen Eceiza, Jose Luis Flores, and Mikel Iturbe, «Fuzzing the Internet of Things: A Review on theTechniques and Challenges for EfficientVulnerability Discovery in Embedded Systems».

[20] George Klees, Andrew Ruef, Benji Cooper, Shiyi Wei, Michael Hicks, «Evaluating Fuzz Testing», https://cseweb.ucsd.edu/~dstefan/cse227-spring20/papers/klees:evaluating.pdf

[21] Federal Ministry for Economic Affairs and Energy (BMWi) , «Secure implementation of OPC UA for operators, integrators and manufacturers», https://www.de.digital/DIGITAL/Redaktion/EN/Publikation/secure-implementation-of-opc-ua.pdf?__blob=publicationFile&v=1

[22] VAISHNAVI VARADARAJAN, «Security Analysis of OPC UA in automation systems for IIoT.», https://kth.diva-portal.org/smash/get/diva2:1653807/FULLTEXT01.pdf

[23] Dong-Hyuk Shin,1, Ga-Yeong Kim, and Ieck-Chae Euom2, «Vulnerabilities of the Open Platform Communication Unified Architecture Protocol in Industrial Internet of Things Operation», https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9460827/

[24] Federal Office for Information Security, «OPC UA Security Analysis», https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/Studies/OPCUA/OPCUA_2022_EN.pdf?__blob=publicationFile&v=4

[25] Jouni Aro, Heikki Tahvanainen, «OPC UA Enables Secure Data Transfer and System Integrations in Private and Public Networks», https://www.automaatioseura.fi/site/assets/files/1550/f2068.pdf

[26] «IoT Security Lab: What is OPC-UA and how does it manage security», https://www.cisco.com/c/en/us/td/docs/solutions/Verticals/IoT_Security_Lab/OPC-UA_WP.html
[27] Penetration Testing and Network Defense. Cisco Press, «Tiller, B. S.».
[28] Engebretson, P, «The Basics of Hacking and Penetration Testing: Ethical Hacking and Penetration Testing Made Easy».
[29] Weidman, G, «Penetration Testing: A Hands-On Introduction to Hacking».
[30] Matteo Meucci and Andrew Muller, «OWASP Testing Guide v4», https://owasp.org/www-project-web-security-testing-guide/assets/archive/OWASP_Testing_Guide_v4.pdf
[31] Dafydd Stuttard and Marcus Pinto, « The Web Application Hacker's Handbook: Discovering and Exploiting Security Flaws».
[32] Barton P. Miller, David Koski, Cjin Pheow Lee, Vivekananda Maganty, «Fuzz Revisited: A Re-Examination of the Reliability of UNIX Utilities and Services».
[33] Patrice Godefroid, Michael Y. Levin, David Molnar, «Automated Whitebox Fuzz Testing», https://www.ndss-symposium.org/wp-content/uploads/2017/09/Automated-Whitebox-Fuzz-Testing-paper-Patrice-Godefroid.pdf
[34] Jared D. DeMott, Richard J. Enbody, William F. Punch, «Revolutionizing the field of grey-box attack surface testing with evolutionary fuzzing. Black Hat Briefings», https://www.blackhat.com/presentations/bh-usa-07/DeMott_Enbody_and_Punch/Whitepaper/bh-usa-07-demott_enbody_and_punch-WP.pdf
[35] Barton P. Miller, Louis Fredriksen, Bryan So, «An empirical study of the reliability of UNIX utilities»
[36] Raviraj Mahajan, «Augmenting American Fuzzy Lop to Increase the Speed of Bug Detection»
[37] Zero-SWARM – Grant Agreement (GA), 2022, GA No: 101057083
[38] AI-enabled IoT penetration testing: state-of-the-art and research challenges, Greco Claudia, Fortino Giancarlo, Crispo Bruno, Choo Kim-Kwang Raymond, 2022, https://doi.org/10.1080/17517575.2022.2130014
[39] Zero-SWARM - D2.2 - Eco-designed architecture, specifications & benchmarking, 2023, Grant Agreement: 101057083
[40] Zero-SWARM - D2.3 - Cybersecurity implementation templates and methodological approach, 2023, Grant Agreement: 101057083
[41] Zero-SWARM – D5.1 - Distributed automation & information management, 2023, Grant Agreement: 101057083
[42] A. Allakany, G. Yadav, K. Paul, and K. Okamura, "Detection and mitigation of lfa attack in sdn-iot network," in Web, Artificial Intelligence and Network Applications, L. Barolli, F. Amato, F. Moscato, T. Enokido, and M. Takizawa, Eds. Cham: Springer International Publishing, 2020, pp. 1087–1096.
[43] F. Li, R. Xie, Z. Wang, L. Guo, J. Ye, P. Ma, and W. Song, "Online distributed iot security monitoring with multidimensional streaming big data," IEEE Internet of Things Journal, vol. 7, no. 5, pp. 4387–4394, 2020
[44] R. Kassab, O. Simeone, and P. Popovski, "Fog-based detection for random-access IoT networks with per-measurement preambles," 2020.
[45] M. Walshe, G. Epiphaniou, H. Al-Khateeb, M. Hammoudeh, V. Katos, and A. Dehghantanha, "Non-interactive zero knowledge proofs for the authentication of iot devices in reduced connectivity environments," Ad Hoc Networks, vol. 95, p. 101988, 2019.
[46] A. Ukil, S. Bandyopadhyay, and A. Pal, "Iot-privacy: To be private or not to be private," in 2014 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), 2014, pp. 123–124.
[47] A. Tarighati, J. Gross, and J. Jalden, "Decentralized hypothesis testing ´ in energy harvesting wireless sensor networks," IEEE Transactions on Signal Processing, vol. 65, no. 18, pp. 4862–4873, 2017.
[48] M. Sun and W. P. Tay, "On the relationship between inference and data privacy in decentralized iot networks," IEEE Transactions on Information Forensics and Security, vol. 15, pp. 852–866, 2020.
[49] M. R. Leonard, M. Stiefel, M. Fauß, and A. M. Zoubir, "Robust sequential testing of multiple hypotheses in distributed sensor networks," in 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2018, pp. 4394–4398.
[50] Huang H, Liu Y, Yuan M, Marron JS. Statistical Significance of Clustering using Soft Thresholding. J Comput Graph Stat. 2015;24(4):975-993. doi:10.1080/10618600.2014.948179
[51] IEEE Conference on Standards for Communications and Networking (IEEE CSCN 2023), 2023, Munich, Germany, https://cscn2023.ieee-cscn.org/
[52] Selection of penetration testing methodologies: A comparison and evaluation, 2015, AUSTRALIAN INFORMATION SECURITY MANAGEMENT CONFERENCE, https://ro.ecu.edu.au/cgi/viewcontent.cgi?article=1181&context=ism