HORIZON EUROPE PROGRAMME
HORIZON-CL3-2021-CS-01-01

A EUROPEAN CYBER RESILIENCE FRAMEWORK WITH ARTIFICIAL INTELLIGENCE -ASSISTED ORCHESTRATION & AUTOMATION FOR BUSINESS CONTINUITY, INCIDENT RESPONSE & INFORMATION EXCHANGE

D3.1: AI-assisted Situational Awareness, Prediction & Response Enablers

Abstract: This document reports the work conducted in the PHOENI2X Work Package 3 (WP3) and more specifically in Task 3.1 (User & Entity Behaviour Analytics), Task 3.2 (CTI Discovery, Analytics & Threat Hunting), Task 3.3 (Attack Prediction, Response Recommendation & Adaptation) and Task 3.4 (Risk Impact Assessment, Alert Triage & Response Prioritisation). The main goal of this document is to describe the design and development done for each one of the AI-assisted Situational Awareness, Prediction & Response enablers. The developed enablers have been integrated in the PHOENI2X framework in the context of Task 5.2 (Framework Integration & Testing) and their functionalities demonstrated in the energy, transport and health sectors via the PHOENI2X Use Cases.

Contractual Date of Delivery	31/03/2024
Actual Date of Delivery	31/03/2024
Deliverable Security Class	PU
Editor	Eva Rodríguez (UPC)
Contributors	UPAT, SANL, UPC, FGC, PPC, WSE, AEGIS, SEA, ATOS, APS, NODALPOINT, UiO
Quality Assurance	NODALPOINT, DSA

This project has received funding from the Horizon Europe Research and Innovation programme under Grant Agreement No 101070586

Document Revisions & Quality Assurance

Internal Reviewers

Reviewer #1: #13 NODALPOINT SYSTEMS (NODALPOINT)

Reviewer #2: #16 ARCHI PSIFIAKIS ASFALEIAS (DSA)

Revisions

Version	Date	By	Overview
1	06/02/2024	Editor	Initial draft, with ToC & assignments
2	29/02/2024	Editor	Initial content & structure update
3	01/03/2024	UPC	Input to section 2 and section 5
4	03/03/2024	SEA	Input to section 3.1
5	07/03/2024	SANL	Input to section 4.1
6	14/03/2024	SANL	Input to section 3.2
7	14/03/2024	ATOS	Input to section 4.2
8	15/03/2024	UiO	Input to section 4.3
9	15/03/2024	ATOS	Input to section 6
10	18/03/2024	UPC	Input to section 1 and section 7
11	21/03/2024	AEGIS	Input to section 4 and section 6
12	27/03/2024	DSA, NODALPOINT	Review
13	29/03/2024	Editor	Final version for submission

Contents

List of Tables

List of Figures

List of Abbreviations

ABIS: Abbreviated Impulsiveness Scale

AI: Artificial Intelligence

AMI: Advanced Metering Infrastructure

API: Application Programming Interface

AutoML: Automated Machine Learning

ATC-IB: Attitudes Towards Cybersecurity and cybercrime In Business

CBS: Conservative Behavior Scale

CERCA: Cyber Risk Assessment Calculator

CLI: Command Line Interface

CTI: Cyber Threat Intelligence

DDoS: Distributed Denial of Service

DLMS: Device Language Message Specification

DoS: Denial of Service

EOS: Exposure to Offence Scale

EPS: Electric Power System

EWS: Early Warning Systems

EXM: Entity eXtraction Module

GDPR: General Data Protection Regulation

IDS: Intrusion Detection Systems

IoR: Internet of Railways

IoT: Internet of Things

IPS: Intrusion Prevention Systems

HAIS-Q: Human Aspects of Information Security Questionnaire

HTML: HyperText Markup Language

HTTP: HyperText Transfer Protocol

IoCs: Collecting Indicators of Compromise

JSON: JavaScript Object Notation

KAB: Knowledge, Attitude, and Behaviour

LoRaWAN: Long Range Wide Area Network

LSTM: Long Short-Term Memory

MISP: Malware Information Sharing Platform

ML: Machine Learning

MQTT: Message Queuing Telemetry Transport

MVP: Minimum Viable Product

NLP: Natural Language Processing

OCS: Online Cognition Scale

OSINT: Open Source Intelligence

PMEM: Predictive Maintenance

PoC: Proof-of-Concept

PSCC: Power System Control Centers

RBS: Risky Behaviour Scale

RNNs: Recurrent neural networks

RPS: Risk Perception Scale

RScB: Risky cybersecurity Behaviours scale

SeBIS: Security Behaviour Intentions Scale

SDO: STIX Domain Object

SQL: Structured Query Language

SRO: STIX Relationship Object

STIX: Structured Threat Information Expression

TAC: Threat Actor Context

TII: Threat Intelligence Integrator

TIPs: Threat Intelligence Platforms

TPS: Traction Power Systems

UEBA: User & Entity Behaviour Analytics

UML: Unified Modelling Language

VM: Virtual Machine

1 Introduction

1.1 Purpose of the Document

The main purpose of this document is to provide the design and development performed towards the first version of the AI-assisted Situational Awareness, Prediction & Response enablers of the PHOENI2X framework, which will improve the AI-assisted situational awareness and prediction capabilities with risk impact assessment, facilitating prioritisation, recommendation and adaptation of system response. To this end, the following activities are conducted in the context of the WP3 tasks:

- Task 3.1 - User & Entity Behaviour Analytics: This task designs, develops, and delivers the AutoML-based UEBA and explores AI based security training.
- Task 3.2 - CTI Discovery, Analytics & Threat Hunting: This task designs, develops and delivers the CTI Discovery & Analytics, Threat Intelligence Integrator (TII) and Threat Actor Context (TAC) Ontology.
- Task 3.3 - Attack Prediction, Response Recommendation & Adaptation: This task designs, develops and delivers the Attack Prediction, Response Recommendation & Adaptation enabler that will provide functionalities for attack categorization, AI-based attack detection and anomaly prediction, and attack response.
- Task 3.4 - Risk Impact Assessment, Alert Triage & Response Prioritisation: This task designs, develops and delivers the Risk Impact Assessment, Alert Triage & Response Prioritisation enablers based on Cyber Risk Assessment Calculator (CERCA).

This document is structured to address the key subjects detailed above. Specifically, section 2 presents an overview of the AI-assisted Situational Awareness, Prediction & Response enablers, their role in the PHOENI2X framework, as well as their interactions with the other components. Section 3 presents the User & Entity Behaviour Analytics enablers, section 4 the CTI Discovery, Analytics & Threat Hunting enablers, section 5 the Attack Prediction, Response Recommendation & Adaptation enabler, section 6 the Risk Impact Assessment, Alert Triage & Response Prioritisation enabler. For each enabler an overview, its design, implementation details and a recap of its current status and next steps are provided.

1.2 Overall Methodology

This section presents an overview of the methodology followed in the preparation of this deliverable, the main objective of which is the design, development and delivery of the first release of the AI-assisted Situational Awareness, Prediction & Response Enablers.

Initially, the technologies or components offered by WP3 partners, being the foundation for the different AI-assisted Situational Awareness, Prediction & Response enablers, were detailed in D2.1 “PHOENI2X Requirements & Architecture”. For each one of the components the fundamental implementation-level details (hardware requirements, type and size of input and output data), interfacing, interactions, and functionalities were defined. The specification of the components laid the foundations for the definition of the detailed PHOENI2X architecture. Subsequently, the interactions between the enablers and the baseline prevention, detection and response toolset were defined, as well as the UML use case diagrams for the three use cases’ scenarios, in D5.1 “PHOENI2X framework - MVP”. This work served as the basis for the refinement of the key components which constitute the AI-assisted Situational Awareness, Prediction & Response enablers, their development and integration in the MVP PHOENI2X framework. Further developments of these components will be incorporated in the PHOENI2X framework throughout the duration of the project, achieving the second and final version of AI-assisted Situational Awareness, Prediction & Response enablers on M35.

1.3 Relationship with other PHOENI2X deliverables

This deliverable provides the design and development of the first version of the AI-assisted Situational Awareness, Prediction & Response enablers based on the requirements and architectural design of the PHOENI2X framework as presented in D2.1 “PHOENI2X Requirements & Architecture”. The AI-assisted Situational Awareness, Prediction & Response enablers will communicate with the baseline toolset defined in D2.2 “Baseline Enablers” and with the Coordinated Response & Preparedness enablers described in D4.1, as specified in the PHOENI2X architecture, in order to jointly achieve their main objective, i.e., enhance the AI-assisted situational awareness and prediction capabilities of the PHOENI2X framework.

The first version of the AI-assisted Situational Awareness, Prediction & Response enablers has been integrated into the first (MVP) version of the PHOENI2X framework across the project’s use cases as described in D5.1 “PHOENI2X framework - MVP”.

2 AI-assisted Situational Awareness, Prediction & Response

The primary output of this deliverable is the first version of the AI-assisted Situational Awareness, Prediction & Response enablers, as designed and developed in WP3 activities. These enablers are intended to enhance the AI-assisted situational awareness and prediction capabilities of the PHOENI2X framework. They integrate risk impact assessment functionalities, facilitating prioritisation, recommendation and response adaptation. Key features include:

- User & Entity Behaviour Analytics (UEBA), analysing and interpreting user and entity behaviour patterns to enhance threat detection and response strategies.
- CTI discovery, extraction, analytics and threat hunting, including analytics for CTI contextualisation and Natural Language Processing (NLP) with Sentiment Analysis for extracting security-pertinent indicators from human-readable text and attack prediction.
- AI-driven attack categorisation, prediction, and response recommendation and adaptation, defining the set of actions to be proactively taken to minimize the chances for a system to be attacked.
- Real-time, holistic, qualitative and quantitative risk impact assessment, combining results from all previously-mentioned capabilities and considering business impact to drive alert-triage and response prioritisation.

To this end, four enablers, depicted in Figure 1, have been designed and developed:

User & Entity Behaviour Analytics (UEBA): It enhances threat detection and response strategies by leveraging model-driven data analytics using Machine Learning (ML) and statistical analysis, as well as Automated Machine Learning (AutoML). Moreover, it identifies cyber-risk caused by insecure user behaviour, providing the appropriate cybersecurity training.

CTI discovery, extraction, analytics and threat hunting: The Threat Intelligence Integrator (TII) contextualizes and enhances the CTI information received through the following capabilities: (i) Collecting Indicators of Compromise (IoCs) from different Threat Intelligence Platforms (TIPs) that are used to collect several OSINT data provided from diverse feeds; (ii) Normalizing, deduplicating, and aggregating IoCs to identify relevant interconnected information, forming a composed IoC; and (iii) Collecting static and real-time information from the infrastructure, that together with the composed IoCs, allow the calculation of a score that will enrich the IoCs. The TII also extends information sharing capabilities by creating an encryption layer on top of MISP, allowing anonymization of some IoCs and providing access control to the shared information.

Attack Prediction, Response Recommendation & Adaptation: It minimizes the chances for a system to be attacked, using supervised and unsupervised learning techniques, integrating the following functionalities: (i) attack categorization to identify the scope and impact of potential attacks; (ii) attack detection and prediction through training ad-hoc behavioural models to estimated probability for the system to report an abnormal behaviour; and iii) attack response and adaptation defining the proactive actions or mitigation strategies according to the type of attack predicted or detected.

Risk Impact Assessment & Prioritisation: It analyses the risk based on the probability of an incident to happen in the monitored infrastructure and on an evaluation of the impact in case of that incident happening. The impact to the organization is calculated considering security and economic aspects, as well as the confidentiality, integrity, and availability. The business profile of the company, the results of vulnerability scans, and the real-time monitoring of the target are considered to execute quantitative and qualitative rules providing both, qualitative and quantitative risk scores. Risk scores are used to improve the proactive and reactive response to the risks, by including a countermeasures prioritization based on situational awareness and CTI data.

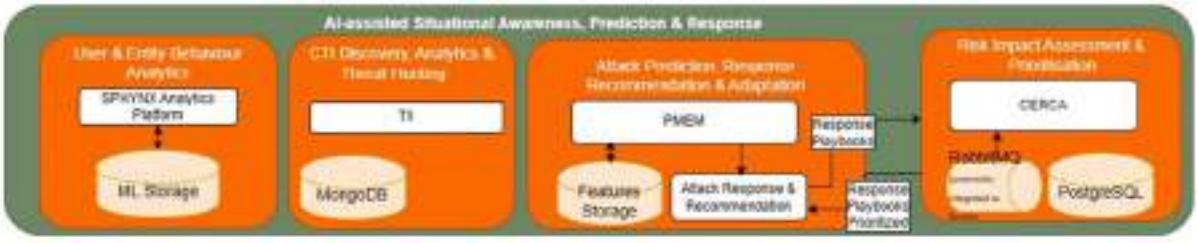

Figure 1. AI-assisted Situational Awareness, Prediction & Response Enablers

The AI-assisted Situational Awareness, Prediction & Response enablers are integrated within the PHOENI2X framework, as illustrated in Figure 2, facilitating the continuous monitoring, analysis, and understanding of the threat environment. These four enablers will allow the framework to conduct an accurate assessment, prediction, and proactive and reactive response to the most pertinent risks. To this end, the AI-assisted Situational Awareness, Prediction & Response enablers interact with the other modules of the PHOENI2X framework, as defined in the architectural design reported in D2.1 "PHOENI2X Requirements & Architecture" and illustrated in the UML sequence diagrams, defined for each one of the use cases, in D5.1 "PHOENI2X framework - MVP".

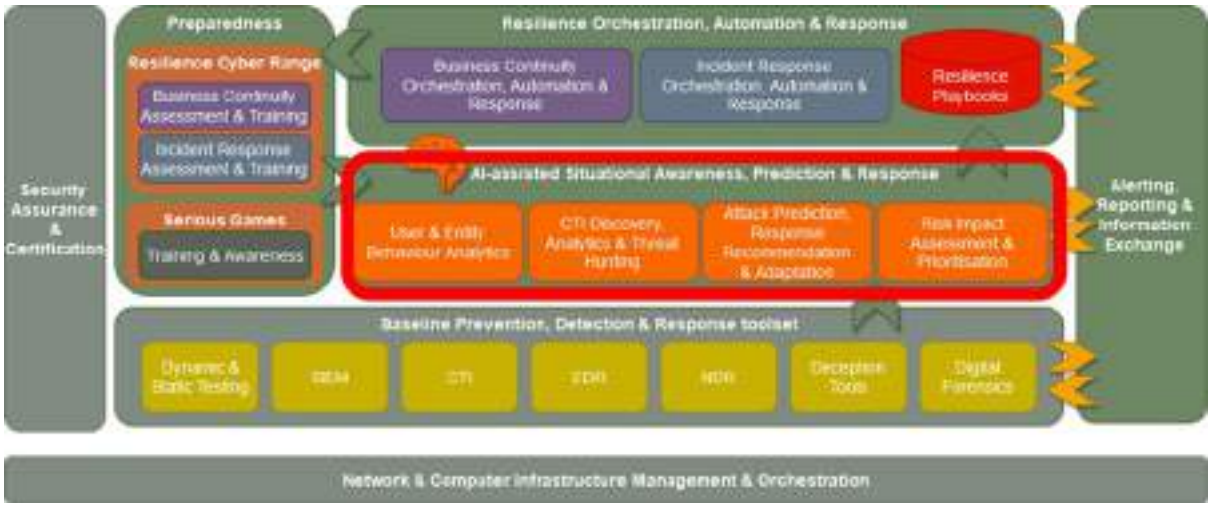

Figure 2. AI-assisted enablers in the PHOENI2X architecture.

3 User & Entity Behaviour Analytics Enabler

This section presents the work done in Task 3.1 “User & Entity Behaviour Analytics”. In this task, the AutoML-based UEBA enabler, which analyses the behaviour of users to detect insider threats, compromised accounts, and other malicious activities, has been developed. Moreover, the usage of AI tools to assist entities on security training based on User Security Behaviour Scales has been explored.

3.1 AutoML-based UEBA

3.1.1 Overview

User Entity Behavioural Analytics (UEBA) is a cybersecurity approach that focuses on detecting insider threats, compromised accounts, and other malicious activities by analysing the behaviour of users and entities within a network. It utilizes ML algorithms and statistical modelling to establish baselines of normal behaviour for individual users and entities, and then identifies anomalies or deviations from these baselines that may indicate potential security incidents.

UEBA solutions collect and analyse data from various sources within an organization's IT environment, including network traffic, log files, endpoint devices, and applications. By correlating this diverse set of data points, UEBA platforms can uncover patterns of behaviour that may be indicative of malicious intent, such as unauthorized access attempts, data exfiltration, or unusual file transfers.

Typical UEBA use cases include:

- Insider Threat Detection: UEBA can help identify insider threats by monitoring user behaviour patterns. For example, sudden access to sensitive data by an employee who has never accessed such data before may raise a red flag. Similarly, unusual login times or repeated failed login attempts from a trusted user's account could indicate unauthorized access or account compromise.
- Account Compromise Detection: UEBA systems can detect when user accounts have been compromised by analysing deviations from normal behaviour. For instance, if a user suddenly starts accessing resources from unusual locations or at odd hours, it may suggest that their account has been hijacked.
- Anomaly Detection: UEBA platforms excel at identifying anomalies that may signal security threats. For instance, if a printer suddenly starts sending large volumes of data to an external IP address, it could indicate a data exfiltration attempt or a compromised device on the network.
- Data Loss Prevention (DLP): UEBA can enhance DLP efforts by detecting unusual data access patterns. For example, if an employee with no prior history of accessing sensitive financial data suddenly starts downloading large amounts of such data, it could signal a potential data breach or insider threat.
- Privileged User Monitoring: UEBA solutions can help monitor the activities of privileged users, such as system administrators, to detect any misuse of their elevated access rights. For instance, if an administrator starts accessing systems or files outside of their usual scope, it may indicate an attempt to abuse their privileges for malicious purposes.

Overall, UEBA plays a crucial role in enhancing an organization's cybersecurity posture by providing real-time insights into user and entity behaviour, enabling rapid threat detection and response. By

leveraging advanced analytics and ML techniques, UEBA helps organizations stay ahead of evolving cyber threats and protect their sensitive assets effectively.

In PHOENI2X, the UEBA enabler will be based on the SPHYNX Analytics platform (herein referred to as "SphynxML"), and more details will be provided in the subsections that follow. Its placement in the high-level and detailed PHOENI2X architecture is presented in Figure 3 and Figure 4, respectively.

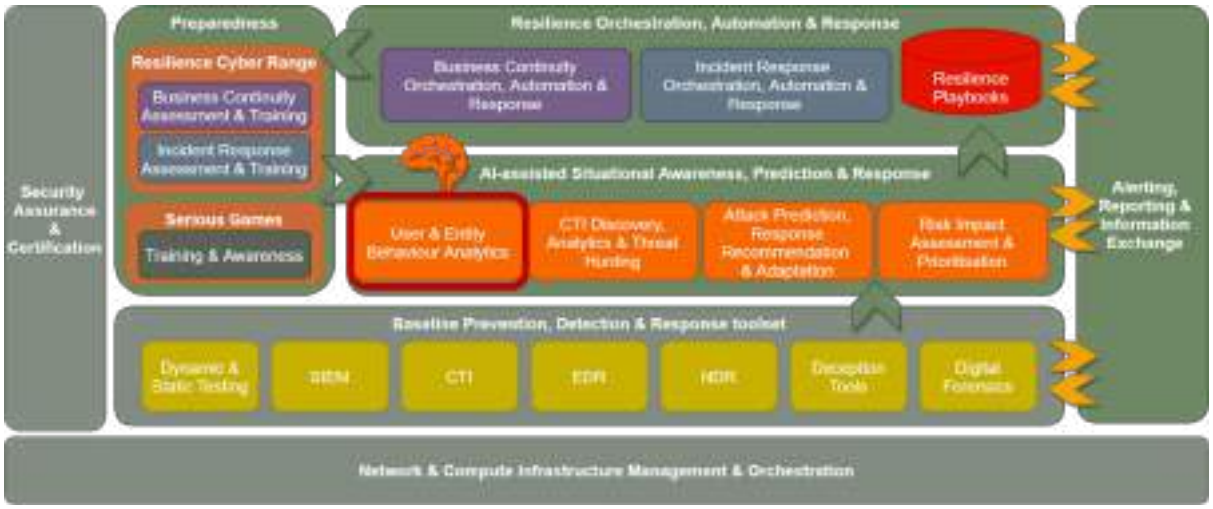

Figure 3. UEBA in the high-level PHOENI2X architecture.

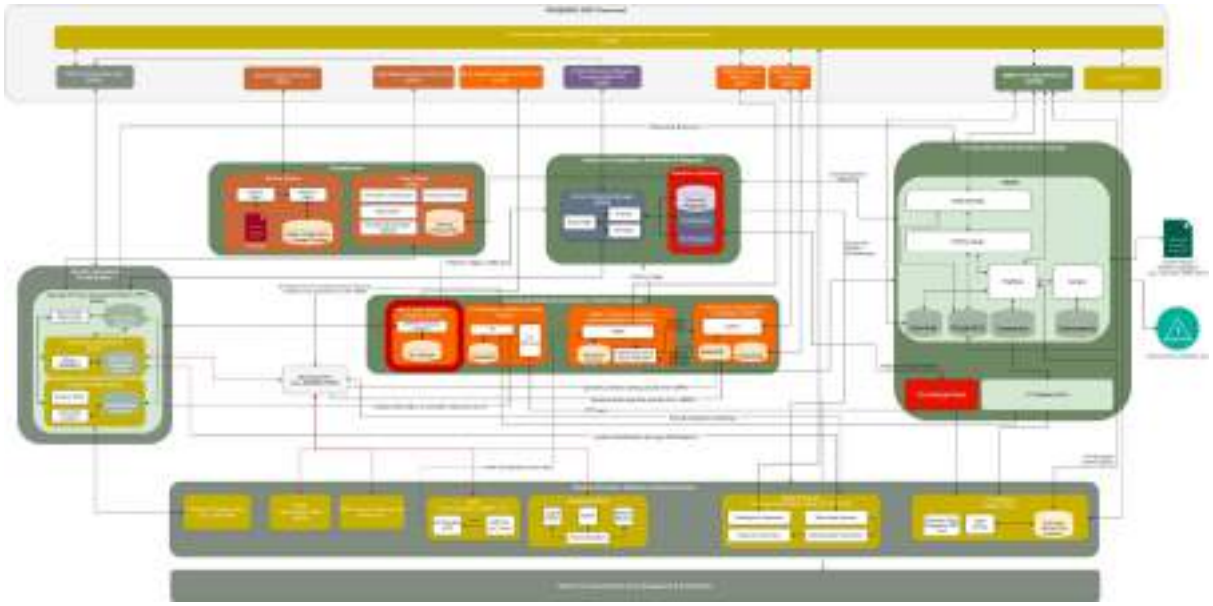

Figure 4. UEBA in the detailed PHOENI2X architecture.

3.1.2 Design Details

SphynxML UEBA leverages advanced ML techniques to proactively detect potential security threats, by drawing insights from historical user and entity data within the network environment. Through the formulation and execution of tailored User and Entity Behavioural Analytics use cases, SphynxML constructs customized ML models capable of discerning suspicious and potentially malicious behaviour exhibited by participating actors, whether they are users or network devices.

This approach not only identifies overtly malicious activities but also flags instances of anomalous behaviour that may warrant investigation, based on patterns identified in historical data. Unlike static

rule-based systems, SphynxML adapts dynamically by generating its own rules during the training process and applies decision-making algorithms to individual events or aggregated behavioural patterns, such as daily user action scores.

With that said, SphynxML ought to not take automated decisions, but rather raise alerts and notify the appropriate teams to review its findings. This information is also relayed to the security operator, to investigate, for example, if one of its existing Intrusion Detection Systems (IDS) static rules needs to be updated for improved threat detection accuracy. Continuous refinement of ML models is integral to UEBA's effectiveness. Employing a selected ML update policy and rigorous health checks, the models are continuously fine-tuned to provide precise and actionable insights.

3.1.3 Implementation Details

The UEBA enabler will be deployed in a containerized form. It fetches the network data stored in one of the project's databases, most likely an elastic database, and uses them for creating the desired ML models. SphynxML is using Automated ML for this procedure, so once the user has gathered the required data, multiple ML methods are tested, and predictive models are available for use. All SphynxML-related metadata are stored in its own Postgres database. Once a predictive model is deployed for predictions, new events are shared from EVEREST via a messaging queue and SphynxML communicate its responses via the same channel. Regarding the explanations of predictions, required by other tools, both a messaging queue, as well as RESTful API options are available means of communication. SphynxML comes as a part of SPA with its own user interface, simplifies all training, overview, and maintenance procedures.

An overview of the enabler's internal architecture is provided in Figure 5.

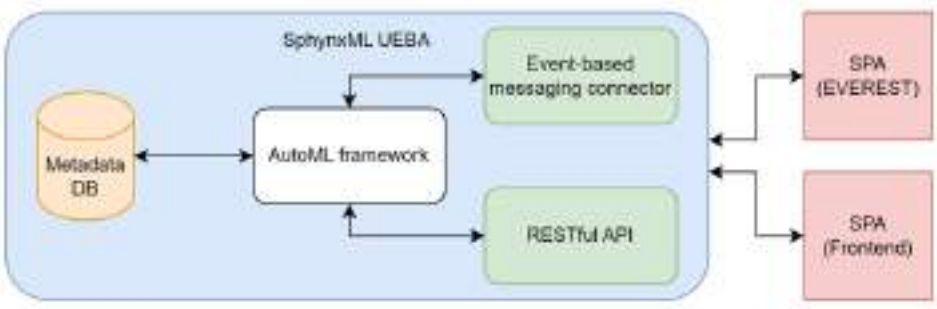

Figure 5. Internal architecture of the UEBA enabler.

3.1.4 Recap of Current Status & Next Steps

Efforts in the first phase of the project focused on implementing & testing the baseline functionality of the SPHYNX ML component to support the UEBA enabler and its applications envisioned within PHOENI2X. Further, the potential use cases of UEBA have been evaluated in the context of the intricacies of each of the PHOENI2X UC environments, followed by an analysis of the available datasets derived from said UC environments to support the training & eventual testing and demonstration of the ML components at the heart of UEBA.

As next steps, the PoC evaluation of the main identified scenarios will be carried out, based on available preliminary datasets coming from the UC environments. The early focus will be on UC3, i.e., the healthcare use case, where rich logs are available from the development environment of NPS. Based on these, a characteristic use case will be demonstrated, whereby UEBA will detect a malicious user who, using legitimate developer credentials and the access said developer has on the development environment, tries to compromise the development process (e.g., to install a backdoor through a vulnerable library, to delete branches of code). Such activities would not be detectable with

traditional security controls (e.g., signature-based EDR, NDR), thereby showcasing the potential of UEBA to detect and mitigate such complex attacks.

3.2 User Security Behaviour Scales

The understanding of users and their cyber risk perception is essential, due to the fact that any security approach is only as good as its weakest link. This often involves humans targeted with social engineering attacks to breach a security perimeter. Once the perimeter is overcome, companies and organizations face the impact of cyber-attacks, affecting both their reputation and finances. In order to strengthen the weakest link effectively and efficiently with targeted security awareness training a user behaviour analysis has to be conducted. This shall result in an understanding of what users think, know, or do about security issues. Furthermore, the security issues may relate to certain entities such as smart phones. The analysis shall discover this relationship as well and focus awareness-trainings to include these entities.

Overall, we aim to achieve the following goals as part of this project:

- Identify cyber-risk caused by insecure user behaviour
- Relate the user risk to devices (entities)
- Leverage AI to conduct, analyse and predict user behaviour to scale this work

3.2.1 Selection of a User Security Behaviour Scales

An effective approach could be to use field observation to assess the security attitude and behaviour of participants. However, this approach also has some major downsides; it is both expensive as well as time-consuming, and assessing the full aspects of security behaviours using this approach can be a challenging endeavour. This may be a reason why most studies rely on self-reported measures (e.g., surveys and interviews) for assessing the cybersecurity knowledge, attitude, and behaviour (KAB) of participants. Self-reported measures are less resource-constrained and can easily integrate different aspects of cybersecurity behaviours. A major problem with many studies exercising self-reported measures is that they develop their own questionnaires (or measurement), which are often non-standardized (does not follow a standard process to design questionnaires and analyse the data). Moreover, such works often examine only one or a few selected components of cybersecurity. Even worse, some studies delivering security tools or proposing a framework perform surveys with the sole intention to establish and prove their work is relevant and useful. In order to overcome these issues related to the use of self-reported measures, some selected studies, listed in Table 1, have produced standardized and well-validated scales and questionnaires intended to measure the cybersecurity awareness of participants. These scales and questionnaires have either followed standard procedures for their design or adapted scales already established in other fields of study. Therefore, we aim to use one of these well-established scales of user behaviour to ensure its empirical effectiveness.

However, self-reported questionnaires might be biased, as they might not only be influenced by the participants' mood, but participants also often get annoyed if they need to repeatedly answer the same questions or too many questions. In order to measure the risk awareness for cybersecurity reliably, it is necessary to measure the risk awareness repeatedly. This is particularly important if one wants to study the long-term effect of applied cybersecurity training measures. As it is not well-researched how repeatedly answering the security and privacy awareness questionnaires might change the results, the repetition might also have an effect on the measurement. On the other hand, it is not possible to just use different scales as the scales are hard to compare and this would not allow a conclusion on how a participant’s sentiment towards cybersecurity risks develops over time.

Still, if risk awareness is measured for a certain time frame, other events in the participants’ lives, such as reports in the media about data leaks or security incidents, might also influence their replies. Therefore, repeated replies of the same participant in different times will get a more reliable outcome.

Table 1 provides an overview of well researched scales to measure the risk knowledge, attitude and planned behaviour of users.

Table 1. Overview of Scales for Cyber risk Behaviour Analysis

Scale	Measurement	Development Processes
Human Aspects of Information Security Questionnaire (HAIS-Q) [1]	Measures security knowledge, attitude, and behaviour. Focuses on 7 security areas.	Used a hybrid methodology that incorporates the inductive and exploratory approaches as recommended by Karjalainen [2] to design the questionnaire. The questionnaire was empirically validated in three phases: - First phase validation used a survey in addition to think-aloud and verbal probing by an expert. - Second phase validation used a pilot survey with 113 valid responses - Third phase validation was performed using a survey with 500 valid responses
Security Behaviour Intentions Scale (SeBIS) [3]	Measures adherence to computer security advice (i.e., attitude and behaviour). Focuses on 4 security areas/dimensions.	Followed the four-step approach as outlined by Netemeyer et al. [4] to develop the scale: - Construct definition and content domain - Generate and judge the measurement items - Design and conduct studies to deploy and refine scale, and - Finalize the scale The scale was validated using a multi-round sequential survey - First-round validation used a survey with 479 valid responses - Second-round refining used a survey with 456 valid responses
SA-6 scale[5]	Measures security attitude. Focuses on 6 security items/questions (using a question for each item).	Utilized Netemeyer et al. [4] and other studies, as well as authors' own experience and that of colleagues for the scale development - Item generation - Survey development - Finalizing candidate items - Finalizing scale items The scale was validated with a U.S. Census- tailored Qualtrics panel. This is followed by a survey with a sample size of 209.
Rajivan et al.'s questionnaire [6]	Measures security skills and knowledge. Focuses on 4 security items/questions.	Used the four-step procedures from Netemeyer et al. [4] for the questionnaire development. - Identify and define the variable intended to be measured using the scale. - Develop the actual items for the scale - Perform exploratory factor analysis to reduce the scale and extract latent factors that summarize the relationship among original variables to build a prediction model. - Confirm the scale fits the intended model
Hadlington's scale [7]	Measures human factors in cybersecurity (attitude and behaviour).	Utilized two existing established scales and developed two remaining scales with the help of experts from related fields of study. - Abbreviated impulsiveness scale (ABIS) [8] - Online cognition scale (OCS) [9] - Risky cybersecurity behaviours scale (RScB)- this scale was based on the SeBIS and was created with input from digital forensic investigators and law enforcement

		- Attitudes towards cybersecurity and cybercrime in business (ATC-IB)- this scale was constructed using expertise from the Police, Digital Forensics, Criminal Psychology, and Cyberpsychology
Öğˇütçü et al.'s scale [10]	Measures security attitude, behaviour, and overall awareness.	Utilized the existing established scales: - Risky Behaviour Scale (RBS) - Conservative Behaviour Scale (CBS) - Exposure to Offence Scale (EOS) - Risk Perception Scale (RPS)
Smartphone Security Behaviour Scale (SSBS) [11]	Measures security behaviour. Focuses on 4 security dimensions/areas using 14 security items/questions.	In the first phase, the authors attempt to adapt the SeBIS for smartphone users. But this did not result in the best-fit items. In the second phase, the authors employed the procedures used by SeBIS for the development of a new set of items for smartphone users.

We decided to use SeBIS [3], because its measures focus on cyber security advice i.e., attitude and behaviour. Furthermore, the scale measures risk perception related to entities in the form of used devices, such as smart phones. Finally, it has a reasonable amount of questions for asking the participants. The 16 questions can be answered in a couple of minutes by all participants and is validated with over 400 participants.

In summary, SeBIS is a psychological assessment tool designed to measure individuals' intentions regarding engaging in computer security behaviours. Developed to provide a comprehensive understanding of the factors influencing a person's security practices, SeBIS encompasses four dimensions: Password Management, Secure Browsing, Personal Information Management, and Updating. These components collectively aim to evaluate the extent to which individuals are likely to adopt behaviours that protect them from cybersecurity threats. The scale uses a series of statements to which respondents indicate their level of agreement, thus providing insights into their security-related behavioural intentions.

SeBIS is particularly valuable for Phoeni2x when seeking to understand and improve user behaviour in the context of cyber security. By identifying specific areas where individuals may be more or less inclined to practice secure behaviours, interventions can be tailored to address these gaps. For example, if a person obtains a low score in the Password Management domain, targeted educational initiatives could be developed to enhance their understanding and practices in this area. Overall, the SeBIS offers a structured approach to diagnosing and improving the security posture of individuals by focusing on their behavioural intentions, which are critical to preventing cyber-attacks and protecting sensitive information.

SeBIS consists of 16 questions divided into four categories, each aimed at assessing different aspects of users' security behaviours. Below, a list of the questions categorized by their respective domains:

Password Management:

- I intend to use long passwords.
- I intend to use complex passwords.
- I intend to use different passwords for different accounts/sites.

Secure Browsing:

- I intend to verify the URL of websites I visit.
- I intend to check for the lock icon in the browser before entering sensitive information into a website.
- I intend to avoid clicking on links in email messages from unknown senders.

Personal Information Management:

- I intend to regularly check the privacy settings on websites/applications I use.
- I intend to limit the amount of personal information I put online.
- I intend to avoid disclosing personal information unless necessary.

Updating:

- I intend to regularly update my computer's operating system.
- I intend to regularly update the software applications I use.
- I intend to regularly update the apps on my mobile devices.

Additionally, there are four more questions that are used for validation purposes and to cover broader aspects of security behaviour:

- I intend to use antivirus software on my computer.
- I intend to back up important information from my computer.
- I intend to use security features on my mobile device (e.g., screen lock).
- I intend to follow my organization's rules regarding information security.

These questions are designed to be answered using a Likert scale (e.g., Strongly Agree to Strongly Disagree) to measure the respondent's intentions towards engaging in secure computer and internet use behaviours.

3.2.2 Integration in the PHOENI2X approach, leveraging AI

Multiple tools exist to conduct user surveys and process the results. Some already leverage ML. Leveraging Artificial Intelligence (AI) for conducting user surveys represents a significant evolution in how data is collected and analysed, offering a more dynamic, efficient, and personalized approach to gathering insights. AI technologies, including NLP and ML, can be used to design surveys that adapt in real-time to the respondent's answers, enabling a more conversational and engaging experience. This adaptability ensures that the questions are relevant to the user's previous responses, thereby increasing the accuracy and depth of the data collected. AI can also analyse open-ended responses at scale, identifying patterns, sentiments, and themes that might be overlooked in manual analysis. This capability not only speeds up the analysis process but also uncovers richer, more nuanced insights into user attitudes and behaviours.

Furthermore, AI-driven survey tools can enhance respondent engagement by utilizing predictive analytics to determine the optimal time and medium for reaching out to potential respondents, thereby increasing response rates. Personalization algorithms can tailor the content of the survey to match the interests and previous interactions of the respondent, making the survey feel more relevant and less intrusive. Additionally, AI can significantly reduce the bias in question framing and interpretation by ensuring consistency in how questions are presented and responses are interpreted. By automating the tedious aspects of survey administration and data analysis, researchers and businesses can focus on strategizing and implementing actionable insights derived from the surveys. Ultimately, the integration of AI into user surveys represents a powerful tool for capturing more accurate, meaningful, and actionable user feedback at scale.

To sum up, we will leverage AI in PHOENI2X to:

- Assist survey participants in replying to the questions as they are intended and clear up misconceptions.
- Automatically analyse the results as they come in and provide updates in short time intervals.
- Follow up on the risk awareness of devices such as computers and mobile devices.
- Suggest training content for a user including considering the user’s risk attitude towards entities as elicited by AI.

4 CTI Discovery, Analytics & Threat Hunting Enabler

This section presents the work done in Task 3.2 "CTI Discovery, Analytics & Threat Hunting", the main outputs of which are the CTI Discovery & Analytics; the Threat Intelligence Integrator (TII); and the Threat Actor Context (TAC) Ontology.

4.1 CTI Discovery & Analytics

4.1.1 Overview

The aim of the CTI Discovery & Analytics component within PHOENI2X is to provide a framework that can support human actors in the Cyber Threat Intelligence (CTI) loop; i.e., to augment the gathering, processing, analysis and dissemination of CTI, providing a more comprehensive view of the threat landscape, in addition to the CTI information received through more conventional channels (e.g., OSINT & commercial CTI feeds).

This can be achieved with the use of web crawling and scraping to gather unstructured threat information from the web, which includes numerous useful sources, such as dark web forums and blogs, social platforms (e.g., Twitter, Telegram) and other relevant sources (e.g., RSS feeds). ML and NLP are valuable tools in this regard, as they are utilized to process that data and information into a standardized CTI format, like Structured Threat Information eXpression (STIX) and "MISP" and provide further analytical capabilities. A dedicated TAXII server is also in place to disseminate these packets of information to clients with various CTI platforms, like OpenCTI or MISP, that is already in place as part of the baseline tools of PHOENI2X.

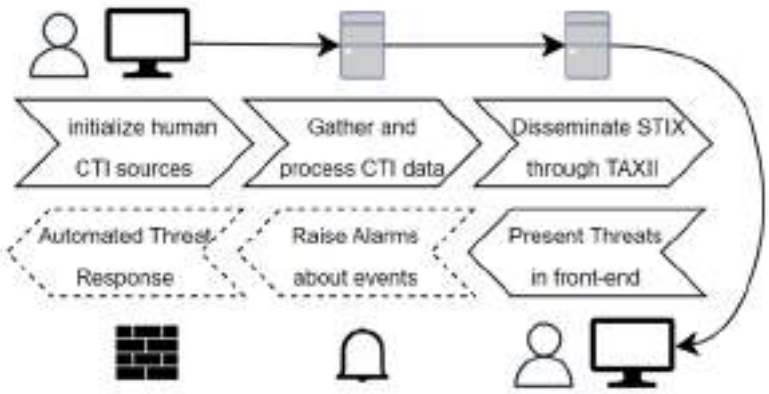

Figure 6. Workflow/flow of information from the CTI Discovery & Analytics component

Overall, the CTI Discovery & Analytics module functions as an autonomous ingestor for the unstructured, human generated content that has CTI significance, online, encompassing the following flow of activities (also visualised in Figure 6):

- The flow of information begins from the initialization tasks that human operators must apply to the framework, in order for it to work according to their needs.
- The framework autonomously scrapes online resources to find relevant information that might be useful in a CTI assessment.
- The scraped results are then filtered to exclude any irrelevant data, and the useful data are kept in the database of the framework.
- After their initial storage, the CTI-related data are processed and annotated, in order to be directly translated into valuable STIX 2.1 objects, that describe each post/article/text as a whole (keeping any precious metadata).

- These STIX 2.1 bundles are then sent to a TAXII server that supports the 2.1 protocol, and then are disseminated to any potential CTI clients, such as OpenCTI.
- These disseminated information, can be used later to inform Early Warning Systems (EWS) and Intrusion Detection and Prevention systems (IDS/IDP) about current and emerging threats "in the wild".

The positioning of the CTI Discovery & Analytics component within the high-level and detailed PHOENI2X architectures is shown in Figure 7 and Figure 8, respectively.

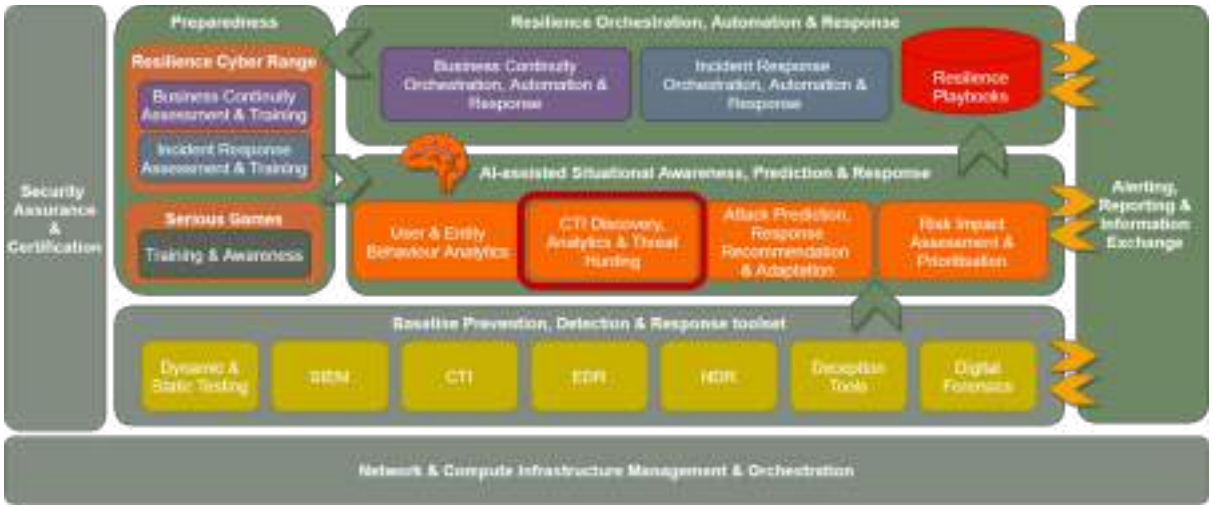

Figure 7. The CTI Discovery & Analytics component within the high-level PHOENI2X architecture.

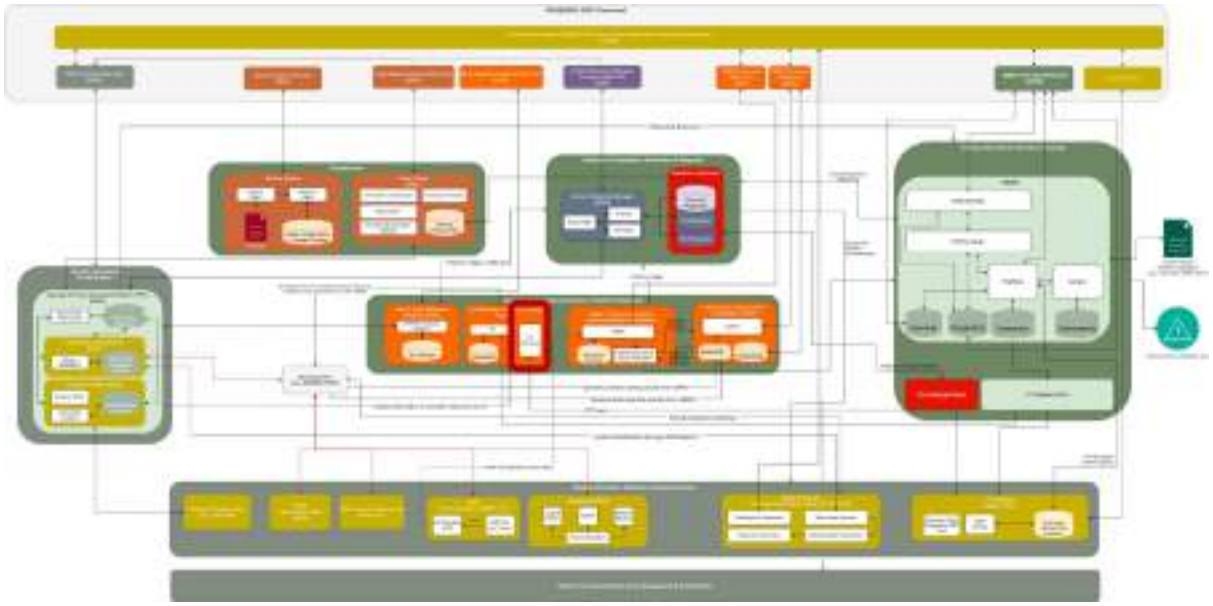

Figure 8. The CTI Discovery & Analytics component within the detailed PHOENI2X architecture.

4.1.2 Design Details

The main components of the Proof-of-Concept (PoC) design of the CTI Discovery & Analytics components are visualised in Figure 9.

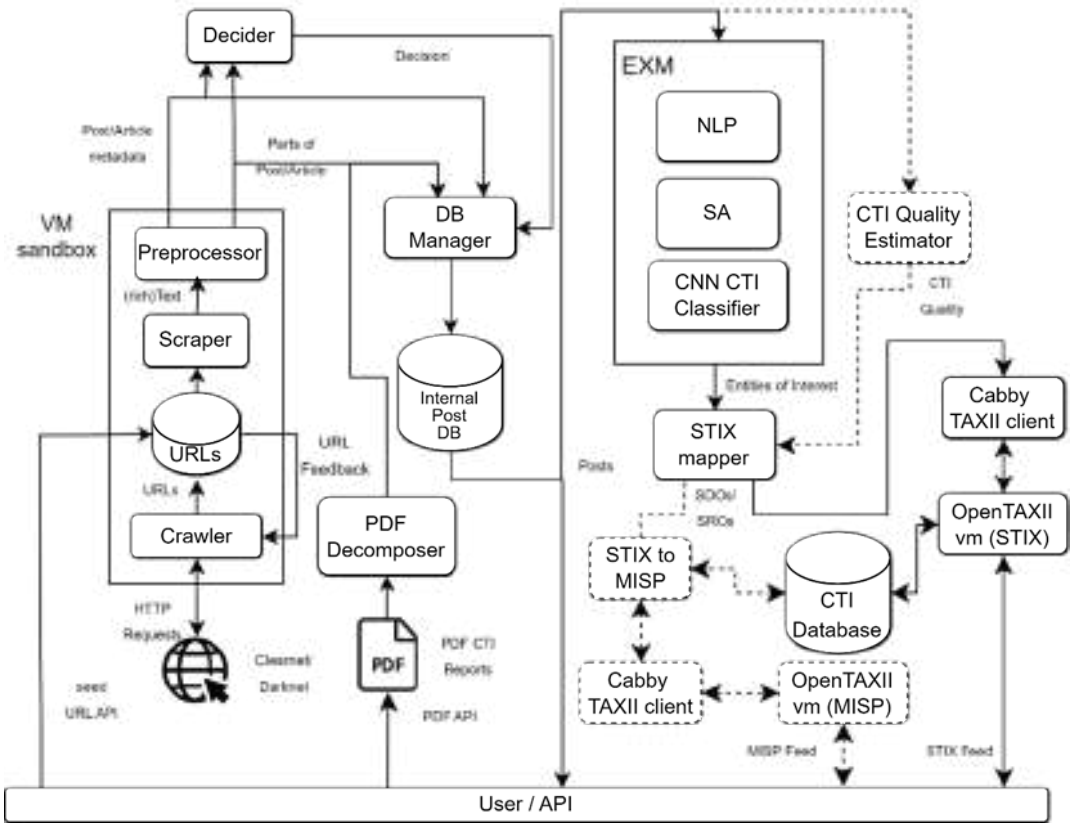

Figure 9. Internal architecture of the CTI Discovery & Analytics component's Proof-of-Concept design

In more detail, the following key components and associated functionalities are included:

- A **Crawler** component that navigates through websites of interest, actively seeking new links to follow according to specified regular expressions. It is considering its history of links to avoid double-accesses. It is also initialized by providing initial seed links to the URL database.
- The crawler then provides the **Scraper** with a list of URLs to scrape (extract the data).
- The above components must be in a **sandbox environment** – e.g., in a VM - to protect the tool from malicious activity against it.
- Because CTI reports can often be in a PDF format, a **PDF Decomposer** is proposed which will convert PDFs to HTML, to continue through the same pipeline as scraped data from websites.
- The rich text (HTML) will be dissected from the **Preprocessor** into its respective parts of interest.
- Right after the crawling/scraping phase, a **Decider** module must decide if the post it was given is security related and contains valuable CTI.
- The internal **DB Manager**, which takes input from the Decider module to either structure appropriately and write to the Internal Post database or discard an incoming piece of scraped data as non-security related.
- The **Internal Post database**, which enables storage and asynchronous use of the data scraped from the previous modules, e.g., for model training and evaluation.
- The most complex part of the design will be **the Entity eXtraction Module (EXM)**, which will be responsible for locating, making sense, and extracting the different STIX Domain Objects (SDOs) and STIX Relationship Objects (SROs) from the plaintext it is given, according to the STIX entities (like Identity, Indicator, Malware, Threat Actor, Tool, etc.). This module will include:
    - A **Natural Language Processor (NLP)** to extract entities directly form complex human expressions.

- A **CTI Quality Estimator,** which will judge the confidence level we can have on a certain part of intelligence. It is outside the scope of this project, but it would constitute a valuable extra.

- All that information must be used by the **STIX mapper** to create STIX objects and knowledge graphs, from the entities that EXM provided, according to specific rules.
- Another optional component will be the in-line **MISP to STIX** converter, that will enhance the tools connectivity, e.g., with the PHOENI2X project (that uses the MISP CTI representation). It is also an optional addition, but it will greatly benefit the system usability.
- To disseminate either STIX or MISP data, an **OpenTAXII instance** is needed. It is a server specifically designed by the creators of STIX to enable sharing of CTI, and it is payload-agnostic, meaning it can accommodate both CTI formats.
- This OpenTAXII instance, needs an external database to store all its data: the **CTI database**. This is a PostgreSQL database that holds all the CTI that the tool has collected and analysed from the dark web in this preliminary version (additional sources to be added as the development evolves).
- To interface the OpenTAXII server, a custom REST API controller is used, that crafts the requests to communicate between the framework and the OpenTAXII server.

All the above components need to be integrated and work together to bring the final capabilities of the CTI Discovery & Analytics envisioned for PHOENI2X. They require at least a single server with virtualization capabilities, and their architecture is designed in a way that they can be spread across many VMs and match the internal structure of an organization. Integration with existing databases and services is also possible. Furthermore, FVT is used as an additional entry point for CTI platform and also assists in the analysis and correlation of system events with threats in the CTI database. The FVT as a common entry point for PHOENI2X contains logs from all relevant components that generate events, allowing the end user to investigate system alerts and map them to known CTI threats.

4.1.3 Implementation Details

The Proof of Concept implementation of the above architecture is concentrated in a few modules that incorporate many of the above functions, as visualised in Figure 10.

A list of the functionalities covered by each module can be found below. Note that many of the modules are open-source existing projects, with some being a standard across the industry, like Elasticsearch[1].

- The **torproxy**[2] docker container, routes our web requests to the onion network, to be able to access and crawl dark web websites.
- The **ACHE crawler**[3] is responsible for all the crawling, being an open-source multifunctional crawler developed by the Visualization Imaging and Data Analysis Center of the New York University.
- The **internal post database** is responsible for holding all the information from crawled websites, with the additional responsibility of keeping all the annotated texts and STIX bundles.
- The **Kibana**[4] instance serves as a tool for developers and administrators to quickly visualise and interact with the internal data of the framework. It is a feature very useful for providing

[1] https://www.elastic.co/
[2] https://github.com/dperson/torproxy
[3] https://github.com/VIDA-NYU/ache
[4] https://www.elastic.co/kibana

statistics of the framework's operation, while also being helpful for the development of any new features, since it contains all information from crawls, even if it is never used.

- The **cti-dath server** holds the main logic of the framework. It is the controller that undertakes the administration of all the other modules. It is a dockerized service written in Java and is also responsible for all the human text filtering, processing and structuring. Also, it is responsible for the STIX and MISP engines implementation, which translate the structured data from the internal ctip-dath format to STIX and MISP respectively.
- Finally, the **OpenTAXII**[5] server acts as the connector of the framework to the organizations infrastructure, by incorporating its TAXII feed into CTI management systems such as OpenCTI.
- The **configuration file** is the interface from which the user can control the operation of the whole framework. It contains the initialization seeds for the crawler, the settings for the TOR proxy, the settings for the Elasticsearch/Kibana and the configuration for the OpenTAXII server.

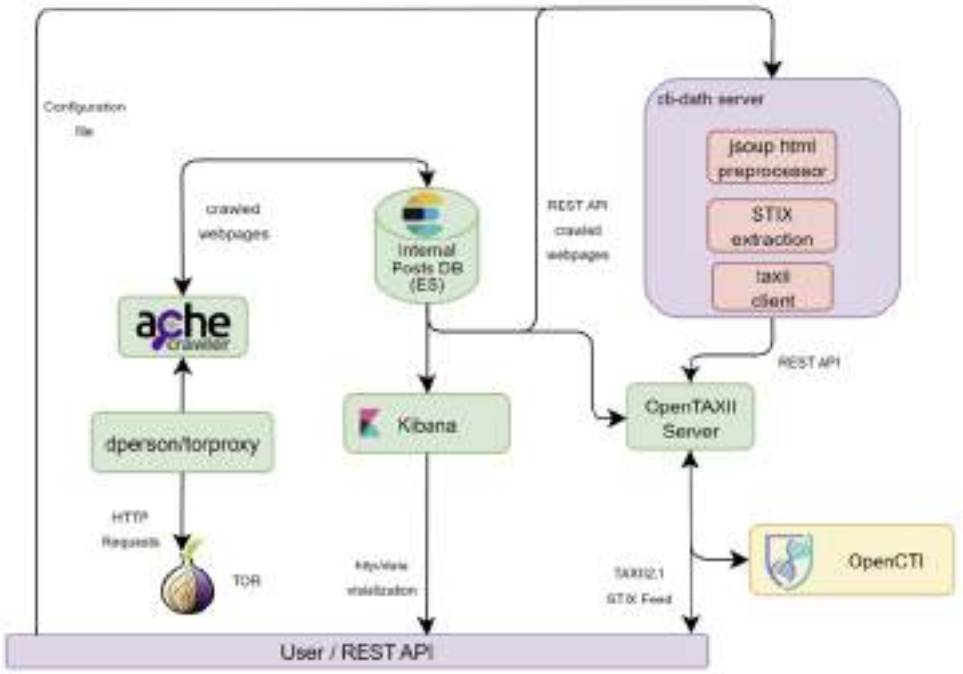

Figure 10. Internal architecture of the CTI Discovery & Analytics Proof-of-Concept implementation.

In practice, the system described above consists of two main parts: one is the cti-dath server, and the other is the set of services it needs to operate (ache, Elasticsearch, OpenTAXII, etc.). In a real scenario, the set of services are brought up first in a server, using the docker-compose of the project. The service stack of the CTI Discovery & Analytics component (internally referred to as *"cti-dath"*) is shown in Figure 11.

[5] https://github.com/eclecticiq/OpenTAXII

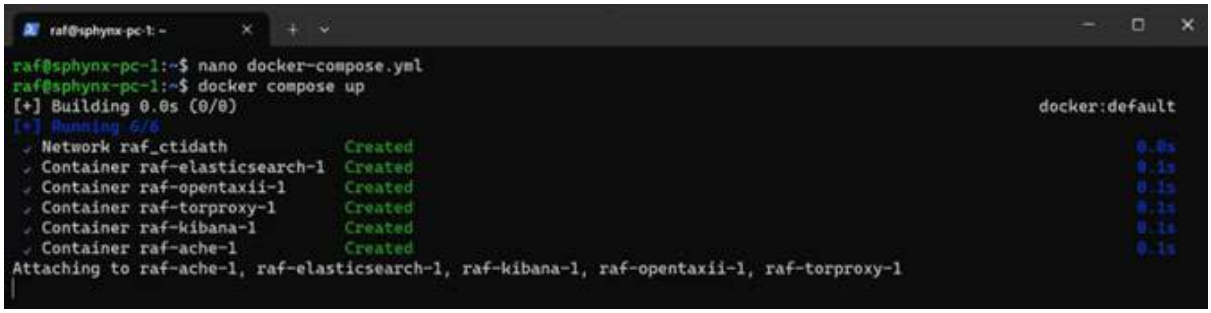

Figure 11. Service stack of the CTI Discovery & Analytics component.

After that, and provided that all containers mentioned above are operational, we can configure and start the *cti-dath* server anywhere with connectivity to the above services. Configuration is currently done via the configuration text file (see Figure 12), which contains options such as the scope of crawling, rate of data gathering, models used for data extraction (e.g., Named Entity Recognition models), structure of crawled websites, and more.

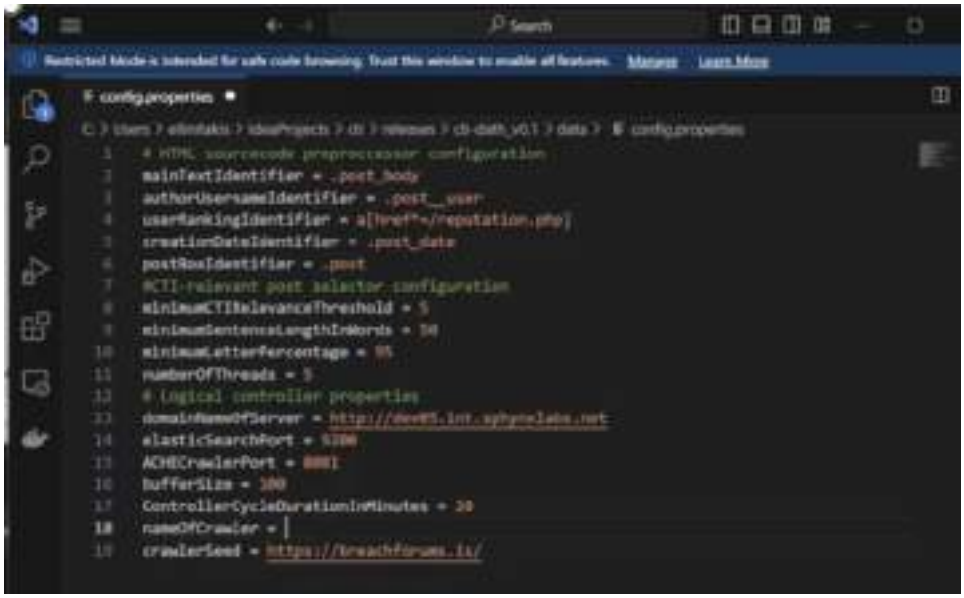

Figure 12. Configuration file of the CTI Discovery & Analytics component.

After the configuration is done, the service can be initiated from the command line and can be left to autonomously gather and process data. A screenshot of the Command Line Interface (CLI) of *cti-dath* after initialisation is shown in Figure 13.

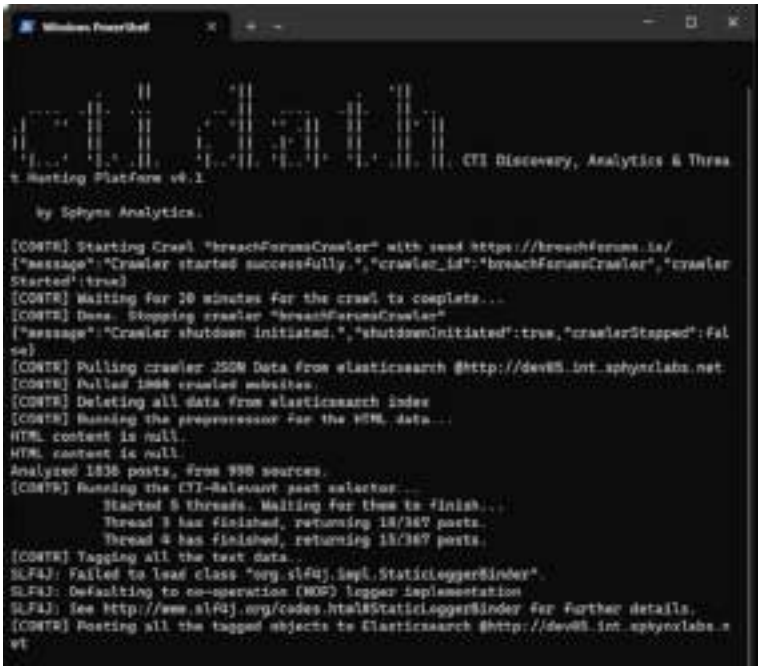

Figure 13. The CLI of the CTI Discovery & Analytics component after initialisation.

Overall, the operation of the component is split into cycles, with each cycle consisting of six separate phases; these are visualised in Figure 14.

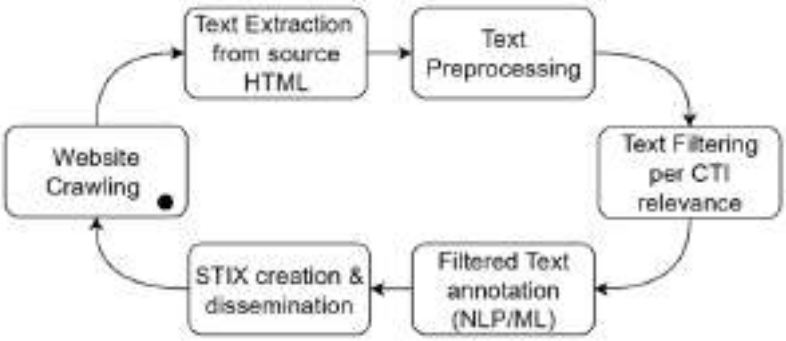

Figure 14. The stages of the autonomous workflow of the CTI Discovery & Analytics component.

Each of these phases are undertaken by a different component of *cti-dath*, and all are moving to the next autonomously. Depending on our internet connection speeds and of course the configuration of the cycle length that is written in the cti-dath config file, this cycle can last for some minutes to a few hours. This is also subject to other limitations, such as storage and computing power available.

During this process, the *cti-dath* will process data using local computing resources, communicate back and forth with all the servers (e.g., when pulling crawler data from Elasticsearch, or posting STIX to the OpenTAXII server), and write logs to the local storage.

While this is running, any software capable of ingesting TAXII feeds can be connected to the server and take advantage of the CTI produced by *cti-dath*. An example tested in our testbed is with an OpenCTI instance that is configured accordingly and can display all the data produced by *cti-dath* in real time. This is visualised in Figure 15.

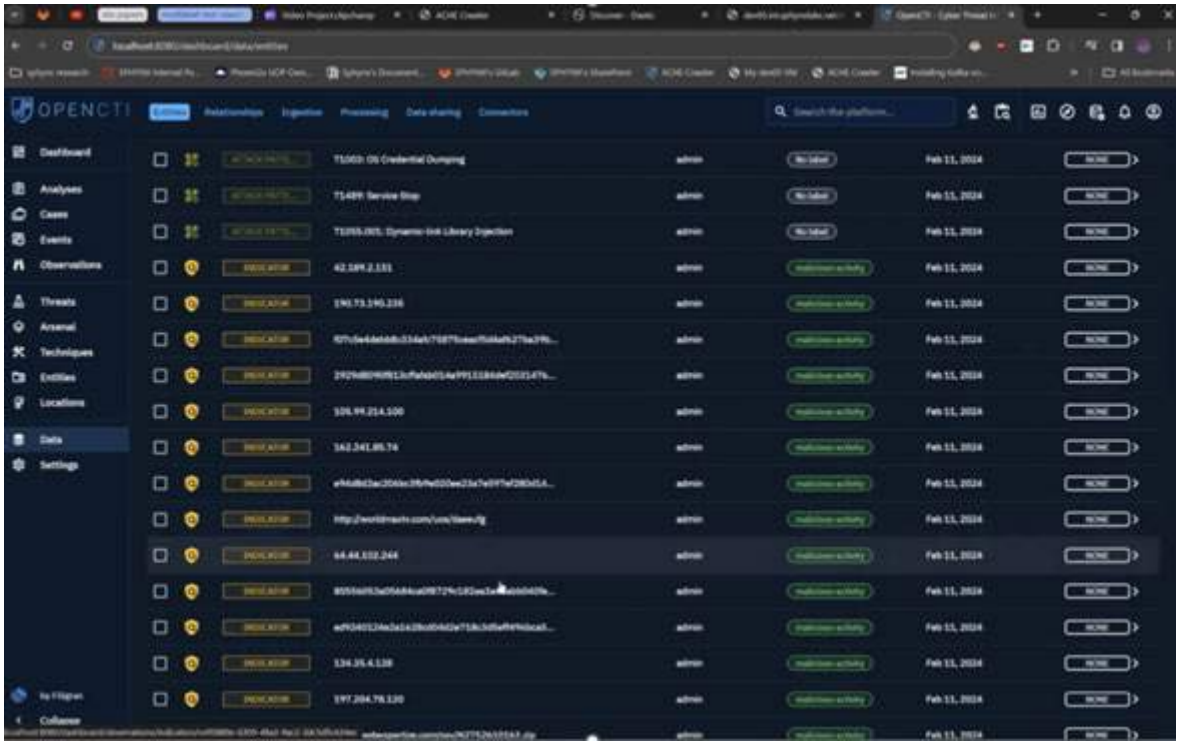

Figure 15. OpenCTI instance configured to ingest the TAXII feed originated form cti-dath.

This connectivity can be leveraged to serve multiple ingestors that span across different networks. It also offers the ability to access the disseminated data programmatically, e.g. with a TAXII client like Cabby for Python.

4.1.4 Recap of Current Status & Next Steps

The baseline functionality provided by the PoC design & associated implementation described above validated the feasibility and potential of the CTI Discovery & Analytics component to provide additional sources of CTI information in an automated manner, with insights aggregated from the Dark Web that would not be as readily available in external CTI feeds.

Future efforts will focus on the integration of additional information sources (e.g., PDF files, Twitter posts & RSS feeds), as well as on improvement on the core NLP capacities of the tool.

4.2 Threat Intelligence Integrator (TII)

4.2.1 Overview

TII is a tool used for CTI Discovery and Analytics. The main idea behind TII is to be able to reduce CTI that does not add value to our organization. The amount of CTI sources and the information we can extract from them is overwhelming; often the volume of information is much higher than the processing speed achieved by CTI analyst teams. That is why it is vital to be able to identify which information is valuable and which is disposable. The same occurs in Security Operation Centers (SOCs); they can receive millions of events every day, the analysis of which would be impossible without a SIEM (System Information and Event Management). However, through tools like SIEM, events are correlated to create alerts that may have varying degrees of importance. But what tools do we have to correlate CTI information with the information we would consider valuable? This is the purpose of TII.

4.2.2 Design Details

TII mainly works with MISP[6] (Malware Information Sharing Platform); which can be used as the CTI source. When we install MISP, it comes pre-configured with some default sources that can be activated, and MISP can also be configured to add sources in addition to those that come by default. As one can imagine, the amount of information can be enormous. TII acts as a filter; every time a new event is received in MISP, TII detects it and performs a series of computations to associate a score with the event.

Figure 16 abstractly describes the architecture of the component.

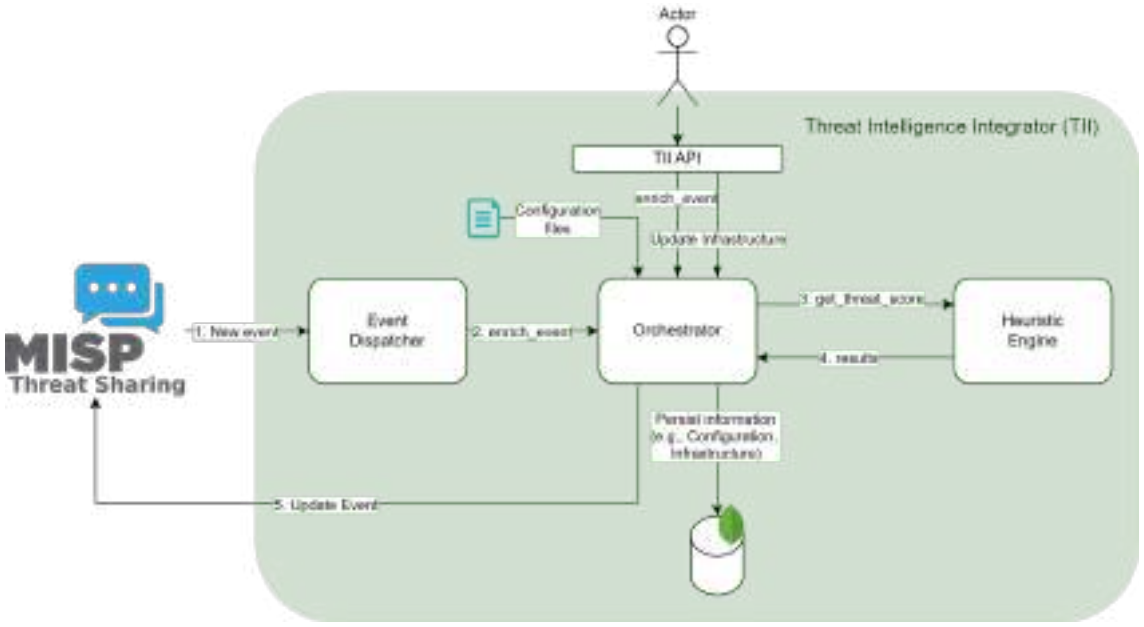

Figure 16. Overview of the architecture of the Threat Intelligence Integrator

The following steps describe the operation of TII before the improvements introduced during PHOENI2X:

1. The event dispatcher detects that a new event has arrived at MISP. New events can reach MISP in various ways:
    - Shared by another instance.
    - Manually included by an analyst.
    - Obtained from one of the default sources configured in MISP.
    - From a source external to those configured by default.
2. After performing some lightweight checks, the event dispatcher asks the orchestrator what information is needed to enrich the event. This endpoint can also be accessed through the API exposed by the orchestrator. This allows users or other components to request the enrichment of an event that does not come directly from MISP, but it needs to follow the format of MISP events.
3. The orchestrator, configured with the infrastructure and other information such as the instance of MISP to be used, gathers the necessary information by communicating with MISP and the database, and sends everything necessary to the HeuristicEngine.

[6] https://www.misp-project.org/

4. In the HeuristicEngine, the event is evaluated based on criteria that will establish the event's score. These criteria and how they are implemented will be described in more detail in section 4.3.1.2.

5. The orchestrator receives the result and adds the score to the event as a Tag. The event is updated in MISP to be visible to the rest of the components.

During the first reporting period of the PHOENI2X project, a series of improvements have been added that increase the simplicity of configuration, the validity of the score assigned to each event, and the flexibility, which allows easier integration with other components. These improvements are detailed in the following subsections. The implementation, including examples of how to configure and use these new functionalities, is described in section 4.3.1.

4.2.2.1 Infrastructure Information

Infrastructure is one of the most crucial parts for filtering valuable information. Without knowing what we have in our infrastructure, the task of identifying what is valuable and what is not, i.e., what affects us and what does not, is difficult. The fact that a vulnerability has a CVSS score of 10 does not make it more important than one with a CVSS score of 8. This is easy to see if we know that the first one refers to a specific version of Nginx that we are not using, but the second one refers to a version of Django that we are using. The same could happen with indicators of other types. If we receive CTI indicating that several APTs are attacking organizations within the healthcare sector, if our organization is dedicated to transportation and is not related with the healthcare sector in any way, the risk is reduced. In previous versions of TII, the infrastructure contained just enough information to involve it in the score calculation. Now, more details have been added to provide granularity, allowing it to be considered in more specific scenarios where they can be really useful. The following subsection highlights one of these scenarios.

4.2.2.2 SBOM and Vulnerability Objects

Software Bill of Materials (SBOM) is a file that lists all the software components used by a software product. This file can be very useful from a security perspective. If we have the SBOM of a software product and know where that software product will be deployed, we can add those components to our infrastructure. TII achieves this by implementing an endpoint that allows receiving an SBOM and associating it with an asset within the infrastructure. There are multiple SBOM formats; so far, TII is capable of parsing CycloneDX, but adding other formats in the future is considered. The schema followed by the infrastructure will be detailed in section 4.3.1.5. Once the infrastructure information is prepared to store SBOM information, it is used with vulnerability objects. These objects are clearly linked to the information contained in the SBOMs. During the calculation of the score for events that have a vulnerability object, relationships between the vulnerability object and the software components used within the various assets of the infrastructure are sought. This allows us to calculate a more meaningful score for vulnerability objects. This functionality is very useful in the following cases:

Updating the infrastructure when deploying new software:

1. Several events with vulnerability objects arrive at MISP. One of them has a Spring Boot vulnerability.

2. A new SBOM is posted to the endpoint indicating that new software is being deployed on an asset and that this software is using Spring Boot.

3. TII recalculates the score for all vulnerability objects present after the infrastructure information has been updated and marks vulnerability objects related to Spring Boot with a high score.

4. If the score exceeds the threshold, this MISP event is sent to CERCA through Kafka to be considered in the risk assessment. An API request is also made to ROAR, which will take the necessary actions to minimize or mitigate the risk.

Discovery of a new vulnerability:

1. The infrastructure is updated, but no vulnerability related to it is found.
2. A new vulnerability is discovered in a library (e.g., Spring Boot). TII checks the infrastructure and sees that this library is being used within an asset, which is considered for the score calculation.
3. If the score exceeds a set threshold, again, that MISP event is sent through Kafka, and an API request is made to the ROAR endpoint to trigger a playbook.

4.2.2.3 TII Input and Output

As seen in section 4.2.1.1, the main input of TII is MISP events. TII is synchronized with MISP so that every time a new event occurs, TII can enrich it. However, TII also has an API; among all its endpoints, we can highlight the following:

- *Write_new_sbom*: Allows adding SBOM information to an existing asset in the infrastructure.
- *Get_threat_score*: Allows calculating the score of an event. This endpoint is used internally to synchronize MISP and TII; however, it can also be accessed directly via API. More details on these endpoints are provided in section 4.3.1.1 (MISP – TII Synchronization).

The output of TII is one of the improvements introduced during this project. Previously, TII was synchronized with MISP, and when TII calculated the score of an object, it updated the event within the MISP instance. Now, in addition, TII can be configured to send that enriched event through other channels. These channels are Kafka and API endpoints. Kafka allows us to redirect the enriched event to other components that need it, such as CERCA to recalculate the risk when necessary. Using endpoints allows us to perform more advanced actions such as executing a playbook when an event's score exceeds a certain threshold. These improvements have opted for flexibility, so the output format, whether through Kafka or API endpoint, is fully configurable and no-code. JSON templates are used that are then dynamically filled.

This functionality not only added flexibility to the asset but also was the key point for integration with other components of PHOENI2X like CERCA and ROAR. The detailed steps to configure the output of TII are described in section 4.3.1.4.

4.2.3 Implementation Details

4.2.3.1 MISP – TII Synchronization

The basis of synchronization between MISP and TII is ZeroMQ[7] (ZMQ). ZMQ is a high-performance library for asynchronous message exchange. This library is particularly known for being lightweight, protocol-agnostic, scalable, and supporting programming languages such as C++, Python, Java, C, and more. MISP comes with a default plugin that allows us to activate a ZMQ queue for communication. We can configure ZMQ in "Administration/Server Settings & Maintenance/Plugins/ZeroMQ". To configure the plugin, we need to modify some configurations:

- Plugin.ZeroMQ_enable to true.
- Plugin.ZeroMQ_event_notifications_enable to true. This will make MISP send any new event that arrives into the instance to ZMQ.

[7] https://zeromq.org/

- Plugin.ZeroMQ_include_attachments to true. This configuration will make MISP include attachments if any.

We can configure additional parameters related to security by modifying the host (Plugin.ZeroMQ_host), port (Plugin.ZeroMQ_port), username (Plugin.ZeroMQ_username), and password (Plugin.ZeroMQ_password) of ZMQ. Figure 17 shows all the different parameters that can be configured for the ZMQ plugin in MISP.

Figure 17. ZMQ plugin configuration in MISP

In the architecture of TII, there is a component that reads from the ZMQ, which is responsible for detecting new events, performing basic processing on them, and sending them to the orchestrator. In this step, events can be sent to the orchestrator for multiple reasons related to different functionalities of the TII. These functionalities are: secure sharing, enrichment, and automated prevention. In the use case of PHOENI2X, we focus on the enrichment functionality, where a score is calculated for the event. This calculation is detailed in the following section. Communication between the ZMQ client and the orchestrator is done through REST API. This allows other users to also use TII through API calls. To request the enrichment of a series of events to the orchestrator, we only need to indicate the list of events and the instance of MISP, from which those events come and where the result will be saved, including the API Key so that the orchestrator can interact with the instance. An example payload to the orchestrator would be as follows:

```
POST https://tiiatos.phoeni2x.eu/orch/get_threat_score
Payload:
{
    "misp_event": {
         <misp event in JSON format>
    },
    "misp_info": {
        "url": "https://mispatos.phoeni2x.eu/",
        "api_key": "x32932wz32J32Yrr_REALAPIKEY_p32532Jo32lf"
    }
}
```

When the calculation is complete, it is time to synchronize those results with MISP. For this, purpose, the PyMISP[8] library is used. This library allows access to the MISP platform through the REST API. When the score is calculated, it is inserted into a tag, and that tag is added to the MISP event, as can be seen in Figure 18.

[8] https://github.com/MISP/PyMISP

Figure 18. Tag added by TII into a MISP event to indicate the level of threat score.

This tag follows a taxonomy that has been created for TII, which is detailed in section 4.2.3.3.

The case of events with vulnerability objects is somewhat different from the rest of the objects because an extra heuristic, relevance, is considered here. Therefore, additional tags are added to indicate if the vulnerability object of the event is related to any of the libraries present in any of the assets of the infrastructure, as presented in Figure 19.

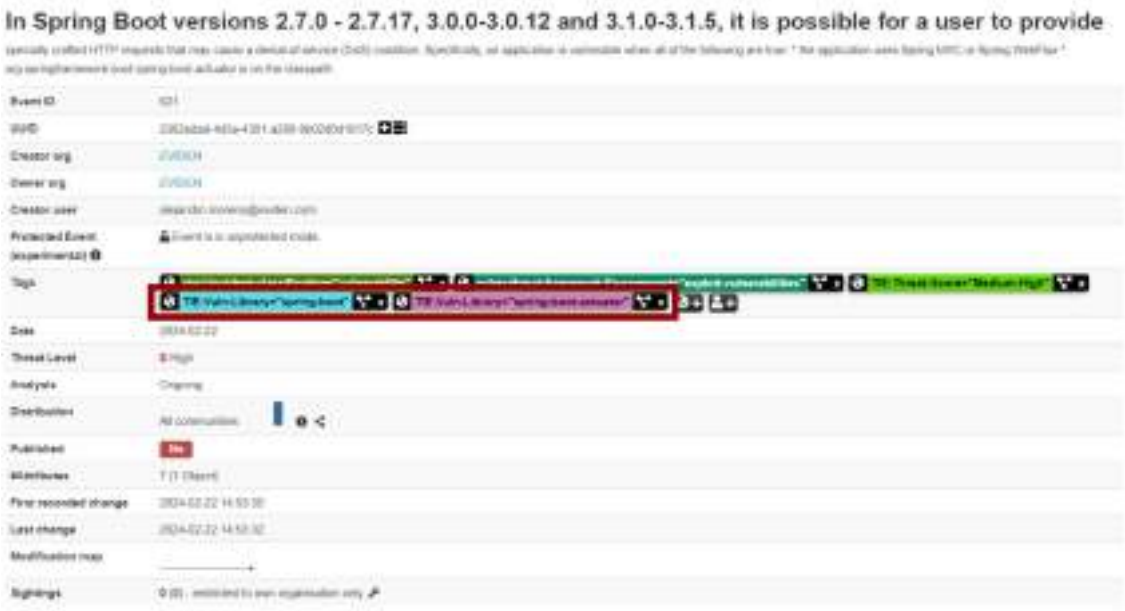

Figure 19. Tags added by TII when a vulnerability is related to libraries present in the infrastructure.

As can be seen, the tags indicate that the libraries spring-boot and spring-boot-actuator are being used within the infrastructure, which may be related to the vulnerability object presented in the previous figure.

4.2.3.2 Score Calculation Mechanism

Although we are working on TII's capabilities to calculate the score for all MISP objects, currently it only covers Vulnerability, Domain-Ip, BTC-Address, and File objects. Several heuristics are applied to

each object, scoring between 0 and 5, and then a weighted average of these heuristics is taken. The description of these heuristics is extracted from other projects such as CONCORDIA [12] 80and SUNRISE [13] where these heuristics have already been presented.

- **Timeliness**. All objects unless file object have a field that is time dependent: published, last-seen, time, respectively. A decay function will be used against those fields: first 48 hours returns a high score (near to 5), after that, it falls exponentially. If the attribute is not present, an OSINT (open-source intelligence) search that finds when it was last mentioned will give us this value. Each object makes use of specific platforms to perform this search, e.g., OpenCVE[9] (vulnerability), VirusTotal[10] (domain-ip and file), URLhaus[11] (domain-ip and file) and BitcoinAbuse[12] (btc-address).
- **Trending**. Google Trends[13] provides the trend of a search in a timespan. The most significant keywords received from the MISP event will be used to evaluate the trending value within a week. If the number of results is increasing from past days it will mean that the threat is urgent, meanwhile if the number of searches is decreasing is possible that the threat has been mitigated and it is less critical.
- **Completeness**. Evaluates the quantity of information inside the Event. If all Required Attributes are present, the score will be 4/5. Optional attributes contribute to the remaining point.
- **Relevance**. In the case of a vulnerability object, the attributes' fields containing keywords that refer to the vulnerability, e.g., summary or description, are used as search parameters. Subsequently, these keywords are used to search for the effect that vulnerability may have on the infrastructure. The value given to the targeted infrastructure is what is returned as score.

4.2.3.3 TII Taxonomy

MISP taxonomies are a standard way of classifying information shared by MISP. The difference from tags is that taxonomies provide common terminology across different instances of MISP, i.e., it is a global definition of how to classify certain information. Taxonomies facilitate multiple organizations using the same terminology to classify data, which is why a taxonomy has been created for TII. Figure 20 shows the TII taxonomy.

Figure 20. TII taxonomy seen from MISP GUI.

A series of labels are defined that may or may not have an associated numeric value. After calculating the event score, the orchestrator maps the original values from 0-5 to 0-100 and decides which label within the taxonomy to use for that event. Although numeric values are limited to those used within the taxonomy, TII's output can be regulated based on the numeric value (i.e., from 0 to 5) to have greater granularity in deciding what to alert on.

[9] https://www.opencve.io/
[10] https://www.virustotal.com/
[11] https://urlhaus.abuse.ch/
[12] https://www.bitcoinabuse.com/
[13] https://trends.google.com/trends/

4.2.3.4 Configuring the Output of TII

Once the score of an event has been calculated, the orchestrator evaluates whether it is necessary to alert on that event. This evaluation is done by comparing the score value with the value defined in the COMM_THRESHOLD environment variable; if the value is greater than this variable, the event will be sent using the configured mechanisms. This new functionality has been developed during the course of this project and also allows configuring various output methods:

- Apache Kafka. The event will be written to a specified topic as long as this functionality is active. The functionality can be activated by setting the ENABLE_KAFKA_OUTPUT environment variable to "True". Multiple brokers and topics can be configured via a configuration file. The following example is a configuration file for writing events to the "TII_out" and "UC3_cerca_in" topics on the first broker, and to the "TII_out" topic on the second broker.

```
{
    "192.168.211.22:9092": ["TII_out", "UC3_CERCA_in"],
    " kafka-test-kafka-1:9092": ["TII_out"]
}
```

The configuration file must be accessible from the container where the orchestrator is deployed, and the KAFKA_CONFIG environment variable pointing to that configuration file must exist. This type of communication is used to communicate high-risk MISP events to CERCA so that it can recalculate the risk based on those events.

- API Request. This type of output can be more complex and allows triggering API endpoints when the score threshold is exceeded. It is very useful for enabling flexible and quick automations. The configuration is similar to that of Kafka. First, we must activate this functionality by setting the TRIGGER_API environment variable to "True". Then, we will select the method we are going to use in the API request, for example, "POST" in the API_REQUEST_METHOD environment variable, and the URL in API_REQUEST_URL, for example, to trigger a playbook in ROAR we can use "http://192.168.211.16:1880/playbook_575caca0-0011-ca5e-0003-321de6565443".

There are occasions when we do not want to send the MISP event through Kafka or API endpoint, but a specific payload. To facilitate this, a no-code solution has been opted for, in which we configure the output format with a template. Later, TII will fill in that template with the available data. We can configure the Kafka output using the OUTPUT_PAYLOAD_TEMPLATE environment variable, which points to a file indicating the template. To modify the API request output, we use the API_REQUEST_PAYLOAD_TEMPLATE variable. In the output templates, we can configure all necessary fields and values, and we can also access variables within TII. Currently, we can access the "NOW" variable that TII replaces with a timestamp, and the MISP_EVENT variable, a dictionary that we can use directly or to select specific values such as the id. The following template is an example of output configuration to trigger a playbook in ROAR.

```
{
  "rq_type": "START",
  "caller": "TII",
  "caller_info": {
    "timestamp": "{{ NOW }}"
  },
  "in_data": {
    "misp_event_id": "{{ MISP_EVENT.Event.id }}",
    "misp_event": {{ MISP_EVENT | tojson }}
  }
}
```

4.2.3.5 Configuring the Infrastructure

Current infrastructures present a complexity that exceeds the limits of what is necessary for our asset. With heavy use of machine virtualization, network functions, and more, we have chosen to opt for our own infrastructure format, which is based on previous developments like the one presented in the project ELECTRON [14]. The TII infrastructure is used to represent assets but not how they are interconnected, as this is not necessary for TII. Within each asset, we have data such as the name, machine description, and assigned IPs. Then we have two main sections.

- Hardware. Contains a list of hardware components, with information such as component type (HDD, SDD, processor, or RAM), version, and manufacturer for each component.
- Software. Contains information about the Operating System (OS) running on the machine, different services and open ports, and a list of libraries extracted from an SBOM file.

The following example describes an asset within TII.

```
    "Assets": {
        "ba0436f3-4206-434c-baf0-422458c1ae27": {
            "name": "Web Server",
            "description": "This is a web server containing multiple open
and private services (Apache, OpenSSH, MariaDB)",
            "software": {
                "os": {
                    "distributor": "Ubuntu",
                    "name": "Ubuntu 20.04.6 LTS",
                    "release": "20.04",
                    "codename": "focal"
                },
                "services": [
                    {
                        "name": "Apache",
                        "version": "2.2",
                        "ports": [
                            "80",
                            "443"
                        ]
                    },
                    {
                        "name": "OpenSSH",
                        "version": "X.Y",
                        "ports": [
                            "22"
                        ]
                    },
                    {
                        "name": "MariaDB",
                        "version": "10",
                        "ports": [
                            "3306"
                        ]
                    }
                ],
                "libraries": {
                    "logback-classic": {
                        "purl": "pkg:maven/ch.qos.logback/logback-
classic@1.4.11?type=jar",
                        "group": "ch.qos.logback",
                        "name": "logback-classic",
```

```
                        "version": "1.4.11",
                        "description": "logback-classic module"
                    },
                    "spring-boot-starter-logging": {
                        "purl":
"pkg:maven/org.springframework.boot/spring-boot-starter-
logging@3.1.3?type=jar",
                        "group": "org.springframework.boot",
                        "name": "spring-boot-starter-logging",
                        "version": "3.1.3",
                        "description": "Starter for logging using
Logback. Default logging starter"
                    }
                }
            },
            "hardware": [
                {
                    "component": "hdd",
                    "manufacture": "WesternDigital",
                    "version": "1.1.0",
                    "details": "more details"
                },
                {
                    "component": "processor",
                    "manufacture": "AMD",
                    "version": "arm64",
                    "details": "more details"
                }
            ],
            "ips": [
                "172.12.25.2",
                "86.158.52.3"
            ]
        },
}
```

Every time changes are made to a component deployed on one of the assets, a new SBOM is created. The libraries from this new SBOM are introduced into the TII infrastructure through a POST request to the write_new_sbom endpoint of TII. In this request, the following is specified: the asset to which the SBOM is associated, the SBOM data, and the MISP instance data. The MISP instance data includes the URL and an API key, which are necessary because when TII receives a new SBOM and extracts the libraries into its infrastructure, it also checks the MISP database to recalculate the score of vulnerability objects. This allows us to detect potential risks in the event that a vulnerability is detected in a library that we were not using before but we are using now. The following excerpt is an example of how to update the TII infrastructure with an SBOM.

```
POST to https://tiiatos.phoeni2x.eu/orch/write_new_sbom
Payload:
{
    "sbom_data": {
        "bomFormat": "CycloneDX",
        "specVersion": "1.4",
        … rest of the SBOM data …
    },
    "asset_uuid": "ba0436f3-4206-434c-baf0-422458c1ae27",
    "misp_info": {
        "url": "https://mispatos.phoeni2x.eu/",
```

```
        "api_key": "xmp9fswzBBJplYrrSCb1seswV4AWpUf5n3JoVIlf"
    }
}
```

4.2.3.6 TII Deployment

The deployment of TII is very simple; the component is fully dockerized and can be deployed using docker compose.

Sudo docker compose --env-file .production.env up -d --build

The .production.env file contains the environment variables that configure TII, including those mentioned to configure the TII output. Other configurations include the URL and API Key of MISP, different configurations related to auxiliary services of TII, and configuration related to the different modules of TII. This allows us to explain different modules of TII in different locations or even deploy modules multiple times if scaling the system is necessary.

To check that the different modules have been deployed correctly, we can use docker logs. We will start by checking that the ZMQ client is waiting for new events from MISP. In Figure 21, we can see the keep-alives that MISP sends to the ZMQ queue to indicate that the connection is healthy (see Figure 21).

```
phoeni2x@PHOENI2X:~/tie-tinted$ sudo docker logs tie-zmq_client | head
{'status': "While you're dying I'll be still alive.", 'uptime': 2760592}
{'status': "And when you're dead I will be still alive.", 'uptime': 2760653}
{'status': 'And believe me I am still alive.', 'uptime': 2760664}
{'status': "I'm doing science and I'm still alive.", 'uptime': 2760674}
{'status': "I feel FANTASTIC and I'm still alive.", 'uptime': 2760684}
{'status': "While you're dying I'll be still alive.", 'uptime': 2760694}
{'status': "And when you're dead I will be still alive.", 'uptime': 2760704}
{'status': 'And believe me I am still alive.', 'uptime': 2760714}
{'status': "I'm doing science and I'm still alive.", 'uptime': 2760725}
{'status': "I feel FANTASTIC and I'm still alive.", 'uptime': 2760735}
```

Figure 21. Keep alives of ZMQ_client module. ZMQ_client correctly deployed.

Similarly, we can see that the orchestrator is ready to receive requests in Figure 22.

```
phoeni2x@PHOENI2X:~/tie-tinted$ sudo docker logs tii-orchestrator
Initializing Kafka  with the following configuration:
['kafka-test-kafka-1:9092': ['TII_out', 'UC3_CERCA_in']}

kafka-test-kafka-1:9092 ['TII_out', 'UC3_CERCA_in']
 * Serving Flask app 'orchestrator' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://0.0.0.0:51001/ (Press CTRL+C to quit)
```

Figure 22. TII orchestrator has been deployed.

Finally, we can confirm that the heuristic engine is ready to calculate the score of events in Figure 23.

```
phoeni2x@PHOENI2X:~/tie-tinted$ sudo docker logs tie-heuristicengine-api
 * Serving Flask app 'heuristicEngine'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5100
 * Running on http://172.19.0.6:5100
Press CTRL+C to quit
```

Figure 23. Logs of Heuristic Engine being deployed successfully.

4.2.4 Recap of Current Status & Next Steps

The efforts in this first phase of the project have been focused on integrating TII with the rest of the components and adapting it to various use cases, especially the energy and health use cases. The tool has been adapted so that its output can be shared using Kafka and custom API endpoints, enabling integration with other components like CERCA and ROAR. The infrastructure has also been improved and adapted for the project, including valuable information from SBOM files, which allows the seamless integration of TII with the health use case.

In future efforts, the mechanism used by TII to enrich CTI information will be improved so that it can be further adapted to each use case. Additionally, mechanisms will be developed to allow the automatic creation of playbooks based on CTI and other available information to TII.

4.3 Threat Actor Context Ontology

4.3.1 Overview

The proliferation of STIX has enabled cyber defenders to use one common language to represent and exchange CTI. STIX enables organizations to share CTI with one another in a consistent and machine-readable manner, allowing security communities to better understand what computer-based attacks are most likely to see and to anticipate and respond to those attacks faster and more effectively. STIX is designed to improve many different capabilities, such as collaborative threat analysis, automated threat exchange, automated detection and response, and more[14].

Vendors and practitioners have created commercial and open-source tools around this framework, notably in the form of Threat Intelligence Platforms (TIPs).

STIX 2 content is encoded in JSON, a simple-to-use and lightweight serialization format for representing data structures and constraints. Some limitations of JSON regarding representing complex domains typically involve that the data is represented in a flat structure with key-value pairs and is not natively strong in defining relationships between objects, even though STIX 2 has a workaround by representing relationships as objects that connect other contextual objects through their identifiers. Also, representing relationships in JSON may lead to redundancy, especially when multiple objects need to reference the same entity. Another limiting factor is that encoding information in JSON presupposes the interconnection of data and schema into singular objects, making the schema evolution challenging and potentially leading to data inconsistency and compatibility issues (breaking changes). Furthermore, querying and traversing information from a graph-resembling JSON structure can be less efficient and more verbose compared to graph-based data models, thwarting effective and efficient analysis. Also, knowing that some organizations might have more refined and use-case-oriented representations for the same concepts makes it challenging to harmonize and interconnect those and keep consistent and proper semantics in place. Lastly, understanding that CTI supports a plethora of operational roles, bringing together information from heterogeneous data sources to assist all these different functions is a challenge, traditionally targeted programmatically, minimizing the potential of using this information flexibly and to the best of our ability.

In that respect, formal ontologies have been proven to be a great alternative for representing complex domains and offer several benefits, including scalability and flexibility, context-aware queries for analytics, data federation between systems, reduced ambiguity and ensuring interoperability, and support inferencing and logical reasoning, enabling automated deduction and analysis of data. The latter allows for the discovery of implicit knowledge and the derivation of new insights from existing information. The integration of a formal ontology and data allows the creation of rich knowledge

[14] https://docs.oasis-open.org/cti/stix/v2.1/stix-v2.1.html

graphs, strengthening data discovery, exploration, analysis, and deriving insights in support of decision-making.

Capitalizing on the above observations, a standards technical committee, the Threat Actor Context Technical Committee (TAC TC) within the OASIS standards developing organization was established to create a formal ontology for the STIX 2.1 model and provide other semantic extensions to support advanced use cases and further requirements in the knowledge representation of the domain. It's a way to organize and represent knowledge in a manner that is both understandable to humans and processable by machines. The ontology defines terms about threat actors to provide semantic interoperability between the variety of systems contributing threat intelligence. This is a significant benefit to existing open sources such as MITRE's ATT&CK and MISP Galaxies by strengthening their abilities to corroborate and cross-reference with other repositories. This also enhances the usability of STIX by providing a bridge to other representations. The development of this work has been consistently supported by the PHOENI2X consortium with the aim of making the TAC ontology the de facto semantic representation for STIX 2.1 adopted and supported by the broader community. The ontology and basic documentation are available on GitHub[15].

Adopters can use Commercial-Off-The-Shelf (COTS) or open-source products for ontologies and triple graphs to spin up the TAC ontology and adapt the ontology based on their business needs. For example, an organization with a requirement to use CTI for vulnerability management can interconnect the TAC ontology with its in-house data sources and ontologies for assets and vulnerability management in a flexible manner. Thereafter, they can stay threat-informed by performing analytical jobs, derive insights, and use inferencing in support of automation.

4.3.2 Design details

The TAC ontology is encoded in Web Ontology Language (OWL), a formal language for knowledge representation designed to enable machines to understand and reason about the meaning of information. OWL is based on description logics and allows defining concepts, properties, and relationships within a domain, including class hierarchies, property restrictions, cardinality constraints, and logical axioms for automated reasoning and inference over the represented knowledge.

The TAC ontology has been developed following a modular approach to assist in the effective and efficient representation of the underlying complex domain and make it easier to track issues, allowing the broader community to contribute to the project and extend STIX 2.1 when applicable or integrate and harmonize with other ontological representations and data. This modular approach is reflected in Figure 24, resembling the way the standard is captured in its technical specification, where each STIX SDO comprises a unique .owl file. Overall, all files are called by a parent file that constructs the full ontology.

[15] https://github.com/oasis-open/tac-ontology

Figure 24. Modular folder and file hierarchy of the TAC ontology (STIX 2.1 representation).

In summary, SDOs and SCOs are defined as classes with their underlying data and object-type properties, whereas SROs are defined as object-type properties connecting different entities altogether comprising triplets. This is depicted in Figure 25 and Figure 26, respectively.

Figure 25. STIX 2 attack pattern domain object class definition in OWL.

```
<owl:Ontology rdf:about="http://docs.oasis-open.org/cti/ns/stix/relationship-types">
        <owl:imports rdf:resource="http://docs.oasis-open.org/cti/ns/stix"/>
</owl:Ontology>

<owl:ObjectProperty rdf:about="&stix;analysis-of">
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="&stix;attributed-to">
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="&stix;authored-by">
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="&stix;based-on">
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="&stix;beacons-to">
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="&stix;characterizes">
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="&stix;communicates-with">
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="&stix;compromises">
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="&stix;consists-of">
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="&stix;controls">
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="&stix;delivers">
</owl:ObjectProperty>
```

Figure 26. STIX 2 relationship objects defined as object type properties in OWL.

4.3.3 Implementation and use case of inferencing

To demonstrate the TAC ontology and, in particular, the subsumed STIX ontology, we use the Protégé[16] software. The ontological model (TBox), including the class hierarchy, the definition of a concept (attack pattern SDO), and its properties (subclasses following the set theory) are depicted in Figure 27.

[16] https://protege.stanford.edu/

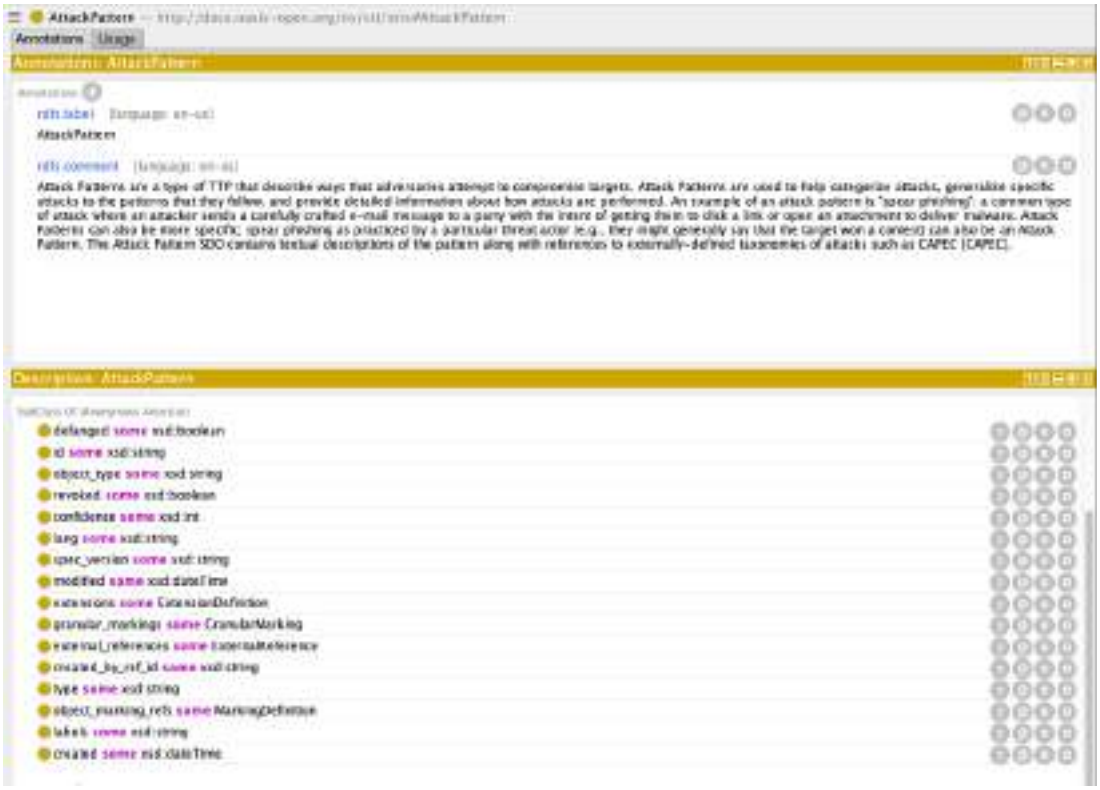

Figure 27. STIX 2.1 Attack Pattern class definition through property restriction using the Protégé editor.

The model has been reviewed by a number of Technical Committee members and has been considered of sufficient quality in the context of representing the STIX 2.1 specification accurately in Web Ontology Language (OWL).

Having the model in place, we use Stardog[17] (COTS software) to import a dataset and generate a knowledge graph. A knowledge graph comprising threat actor context is depicted in Figure 28.

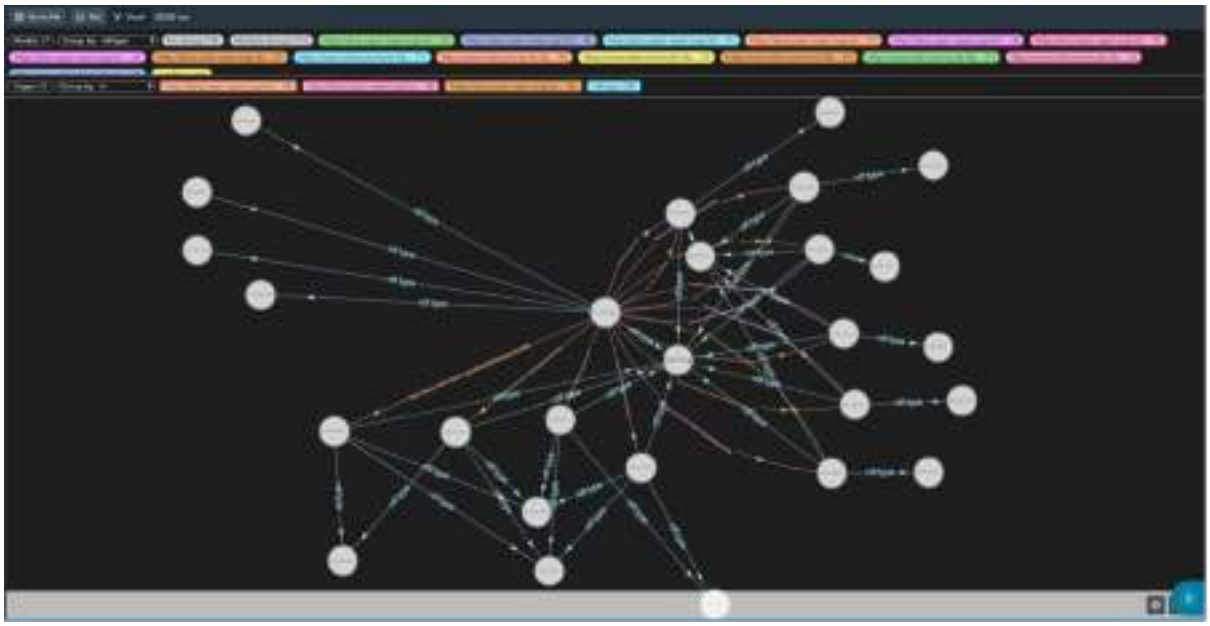

Figure 28. STIX 2.1 knowledge graph using the TAC ontology and Stardog.

[17] https://www.stardog.com/

Using a set of inferential statements (concept equivalence “C ≡ D”) based on Intel’s Threat Agent Library - TAL - [15], we can derive new knowledge regarding the type of adversary (e.g., anarchist) and its relations to STIX 2 axioms. This is depicted in Figure 29. In particular, TAL is a set of definitions and descriptions to represent significant threat agent categories. The TAL was developed to support risk management processes by simplifying the identification of threat agent archetypes that pose the most significant risk to specific assets. Based on the available information on each archetype class, an organization can get an insight into current adversarial activities and consequently take action to improve its security posture. The library (Figure 30) enumerates twenty-one archetypes (e.g., government spy, radical activist, untrained employee, disgruntled employee) and their associated defining attributes: access, outcome, limits, resources, skills, objective, visibility, and motivation. The defining attributes reflect the typical characteristics of each threat actor type. Regarding the TAC ontology, it subsumes an OWL representation of TAL, meaning an ontology. The inferential statements developed followed the defining characteristics of threat agents as they are depicted in Figure 30.

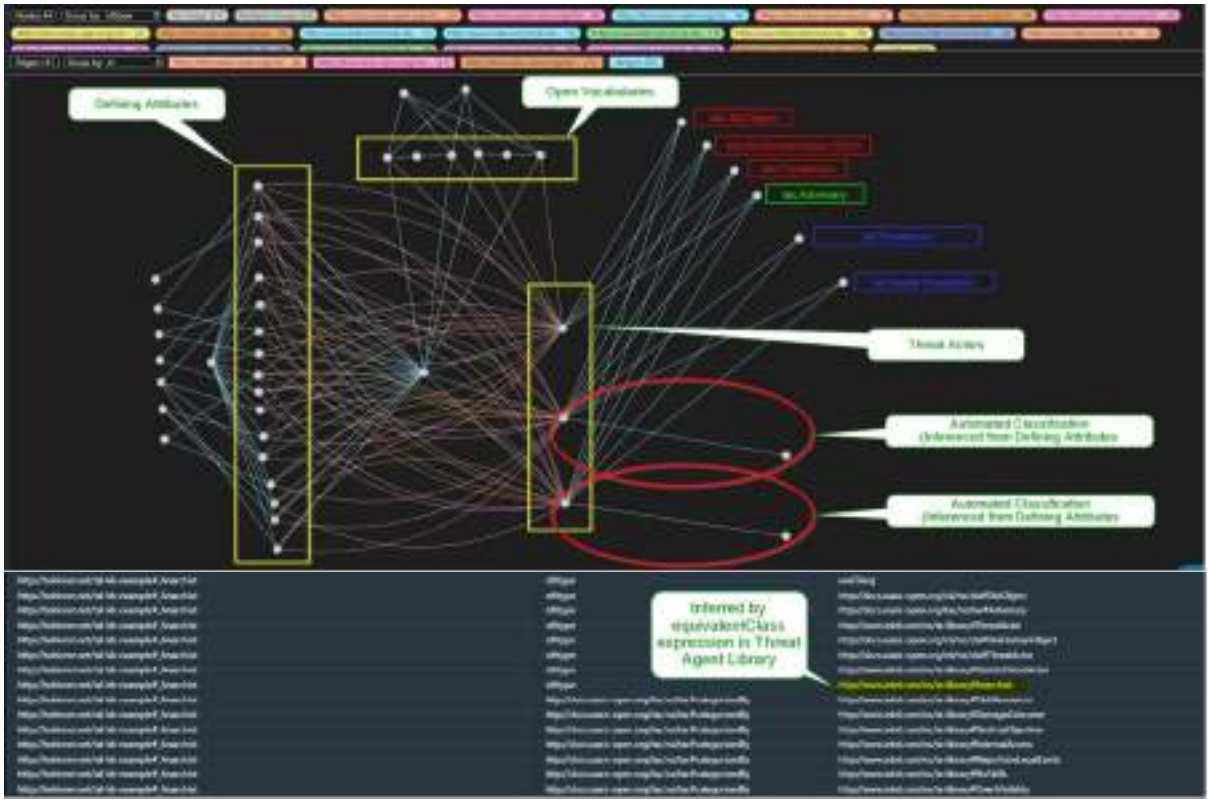

Figure 29. Example of derived information after running a reasoner and based on inferential statements.

Figure 30. Intel's Threat Agent Library.

4.3.4 Next steps

The consortium will continue the development of the TAC ontology focusing on specific use cases. The next developments will be documented in D3.2. In particular, we will encode the CACAO specification/standard that the consortium utilizes for automation capabilities in OWL providing a semantic representation and bridge to STIX 2.1. The use case will account for connecting CTI with playbooks that recommend standard operating procedures or intelligence generated playbooks for specific threats. This will be achieved using reasoning. For example, when a STIX sighting instantiates and is related to ransomware, the reasoner should be able to infer the recommendation of either a generic playbook with SOPs pertaining to ransomware or a tailored playbook for responding to a specific identified threat.

5 Attack Prediction, Response Recommendation & Adaptation Enabler

5.1 Overview

The Attack Prediction, Response Recommendation & Adaptation Enabler analyses the framework behaviour and based on data analytics minimizes the chances for the framework to be attacked proposing specific preventive actions and mitigation strategies. It consists of three sub-modules, as depicted in Figure 31, providing the following functionalities.

- *Attack categorization*: Identification of potential attacks and definition of its scope and impact. This process will be used to foresee the effects a potential attack may have on a system and thus the intensity of the proactive measures to be applied.
- *Attack detection and prediction*: This sub-module detect abnormal behaviours in the system, based on ML algorithms, which may end up in a red flag warning about a potential attack.
- *Attack response and adaptation*: This sub-module defines the list of actions to be enforced by the framework when an attack is detected or predicted. They are defined in two different ways. First, a set of proactive actions, identified according to the type of attack and the impact it is expected to have, to be enforced when attack is predicted. Second, the mitigation strategies to be enforced when an attack is detected.

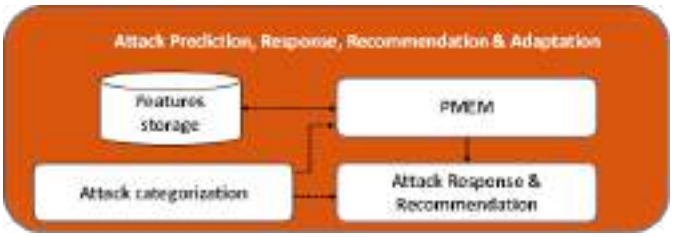

Figure 31. Attack Prediction, Response, Recommendation & Adaptation Enabler.

The placement of the Attack Prediction, Response Recommendation & Adaptation Enabler within the detailed PHOENI2X architecture is shown in Figure 32.

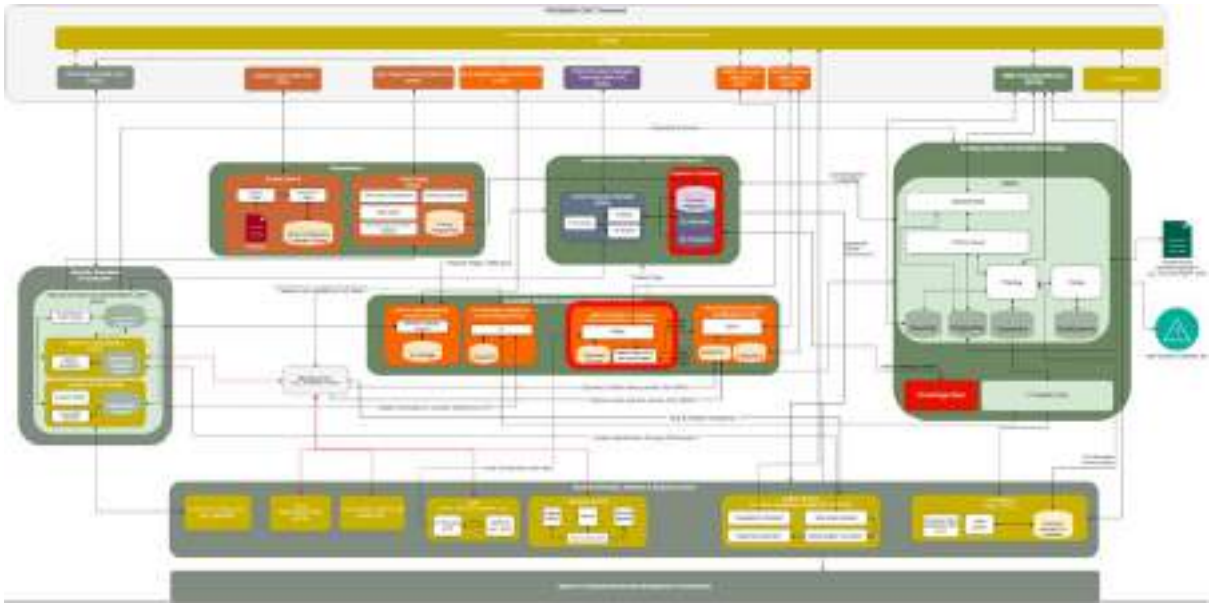

Figure 32. Attack Prediction, Response, Recommendation & Adaptation Enabler within the PHOENI2X architecture.

5.2 Design details

This section presents in detail the design done for the Attack Prediction, Response, Recommendation & Adaptation Enabler. First, the attack categorization done is presented. Then, the design and development done for the Predictive Maintenance (PMEM) tool and for the Attack Response and recommendation.

5.2.1 Attack categorization

This section presents the attack categorization done for the three PHOENI2X use cases. For each one of the use cases the critical assets in the infrastructure that can be targeted by a cyberattack, the potential attack(s), its representation in the MITRE ATT&CK framework[18],and their potential impact on the targeted critical asset have been identified. Table 2 summarizes the attack categorization done for the Energy Use Case, Table 3 for the Transport Use Case, and Table 4 for the Health Use Case.

Table 2. Attack categorization for the Energy Use Case.

Critical Asset	Potential attacks	Impact	MITRE ATT&CK
Electric power system (EPS)	False data injection	Loose operational control and function.	T1055 - Process Injection
Power System Control Centers (PSCC)	Desynchronization and timing based attack Data integrity attack	Data integrity. Data exfiltration.	T1029 - Scheduled Transfer M0810 - Out-of-Band Communications Channel
Substations	Spoofing, jamming multiple message	Incapacitation of the substation, communication networks, control devices, etc.	T0855 - Unauthorized Command Message
Data	Data leakage	Loss of Data. Business organization's operation might be at risk.	T1213.001 - Data from Information Repositories
Smart Meters	Remote attack	Privacy, unauthorized remote load control, and interoperability problem.	T1219 - Remote Access Software
Advanced metering infrastructure (AMI)	Energy theft, Leakage of customer information	Shortage of energy. Negative economic effects, Loss of confidence.	ID: G1004 LAPSUS$ T1078.004 Valid Accounts: Cloud Accounts
Communications (between smart meter and headend)	Man-in-the-middle	Alter the smart meter measurement	T1557 - Adversary-in-the-Middle
Communications	Data tampering	Alter power measurements in the exchanged packets.	T1565 - Data Manipulation
Communications	DoS	Instability of the EPS.	T1499.004 Endpoint DoS

Table 3. Attack categorization for the Transport Use Case.

Critical Asset	Potential attacks	Impact	MITRE ATT&CK
Operation Control Centre	Malware	Damaging critical rail infrastructure.	T1587.001 - Malware
Internet of Railways (IoR) network	DoS Attack	Disable the operation of railway network.	T1499.004 - Endpoint DoS: App. or System Exploitation
Train signalling system	Remote computers attack	Derailment and collision between trains.	T1219 - Remote Access Software
Electric traction substation	False Data Injection Reconnaissance	Traction Power Systems (TPS) make wrong decisions of power absorption/injection.	T1055 - Process Injection TA0043 - Reconnaissance
VANET Signal	Spoofing attack Jamming attack	Transmission delays, derailments.	T0856 - Spoof Reporting Message

[18] https://attack.mitre.org/

Ticketing system	Ransomware	Disruption for rail passengers. Financial loss.	T1486 - Data Encrypted for Impact
Operational Information	Theft of information	Tarnished reputation, Regulatory sanction (GDPR).	T0882 - Theft of Operational Information
Data (Employees)	Data leakage	Company business operations might be in risk.	T1213.001 - Data from Information Repositories: Confluence
Business Information	Sensitive business information leakage	Business reputation.	T1213.001 - Data from Information Repositories: Confluence
Cloud	Account hijacking Advanced persistent threats Data loss DoS Insecure API	Service disruption. Negative economic effects. Loss of confidence.	T1078.004 - Valid Accounts: Cloud Accounts T1005 - Data from Local System M0803 - Data Loss Prevention T0814 - Denial of Service T0871 - Execution through API
End-to-end communication path	Data tampering from the OT (IoT sensors)	Alter sensors' measurements.	T1565 - Data Manipulation

Table 4. Attack categorization for the Health Use Case.

Critical Asset	**Potential attacks**	**Impact**	**MITRE ATT&CK**
Services and Network	DoS, DDoS	Disrupt the services, overwhelm the networks, slow/shut down the network. Financial losses.	T1499.004 - Endpoint Denial of Service: Application or System Exploitation T1498.001 - Network Denial of Service: Direct Network Flood
Healthcare System web portal	DoS, DDoS	Turn down/ make unavailable the online healthcare website. Prevent the accessing of sensitive data.	T1499.004 - Endpoint Denial of Service: Application or System Exploitation T1498.001 - Network Denial of Service: Direct Network Flood
Computer's information (Patient's account Records, tamper the information in the computer)	Malicious software: Trojans. Spyware, Rootkits Network packet detection IP spoofing Password assaults Intrusion attacks	Data is hacked/altered. Lose the control over the user account, password. Tamper the information of the database. False statistics injected by the hackers for wrong communication signal.	S0001 - Trojan. Mebromi T1587.001 - Malware T0851 - Rootkit T0842 - Network Sniffing T1110.001 Brute Force: Password Guessing DS0004 - Malware Repository
Information (Patient's financial information)	Phishing Spam attack Injection attack Cryptographic Attack	Business and financial loss Steal personal information.	T1566 - Phishing T1055 - Process Injection T1600 - Weaken Encryption
Data Breaches (Patient's health and medical insurance data)	Formjacking Browser Extension Malware Man in the Middle or Eavesdropping Ransomware	Loss of data confidentiality. Loss of financial and personal data. Data integrity and authenticity	T1055 - Process Injection T1176 - Browser Extensions T1587.001 - Malware T1638 - Adversary-in-the-Middle T1486 - Data Encrypted for Impact
Health infrastructure supply chain	Compromise source code of software artefacts Use malicious libraries during software builds Replace original execution artefacts with malicious artefacts	Malicious code deployed and running on production servers. Disruption of delivery of critical software updates. Introduction of malicious hardware in medical operations.	T1195.001 - Supply Chain Compromise / Software Dependencies Development Tools T1195.002 - Supply Chain Compromise / SW Supply chain

	Render software delivery pipeline unavailable Manipulate hardware used in the medical supply chain		T1195.002 - Supply Chain Compromise / HW Supply chain

5.2.2 PMEM

PMEM is a tool designed to identify anomalies within a system, leveraging ML models. It consists of two different modules. The first module focuses on attack detection in network traffic. While, the second module forecasts measurements from IoT devices and identifies abnormal deviations from anticipated values. Each time an anomaly or deviation is detected, PMEM generates an alert to notify a potential malicious activity. PMEM outputs are stored into a local database accessible through either a dashboard or an API.

Figure 33 illustrates the architecture of PMEM. The left side represents the attack detection module, while on the right side, the forecasting module predicts IoT device measures by detecting abnormal deviations from expected values.

PMEM can receive two types of data as input (see Figure 33):

- Network traffic data (Unsupervised component input interface): Network traffic flows that are extracted from the monitored infrastructure. This traffic can be captured typically using Tcpdump or Wireshark. Afterwards, it is transmitted to PMEM via Apache Kafka or MQTT protocol.
- IoT measurements data (Forecasting component input interface): Time series data that includes the measurement value captured by an IoT device and the corresponding timestamp. This time series data can be received by PMEM via MQTT and Apache Kafka.

The PMEM's output, for both components, is stored into a database. PMEM stores both, the input data and the corresponding output (prediction made). These results can be accessed via an API or visualized through a dashboard.

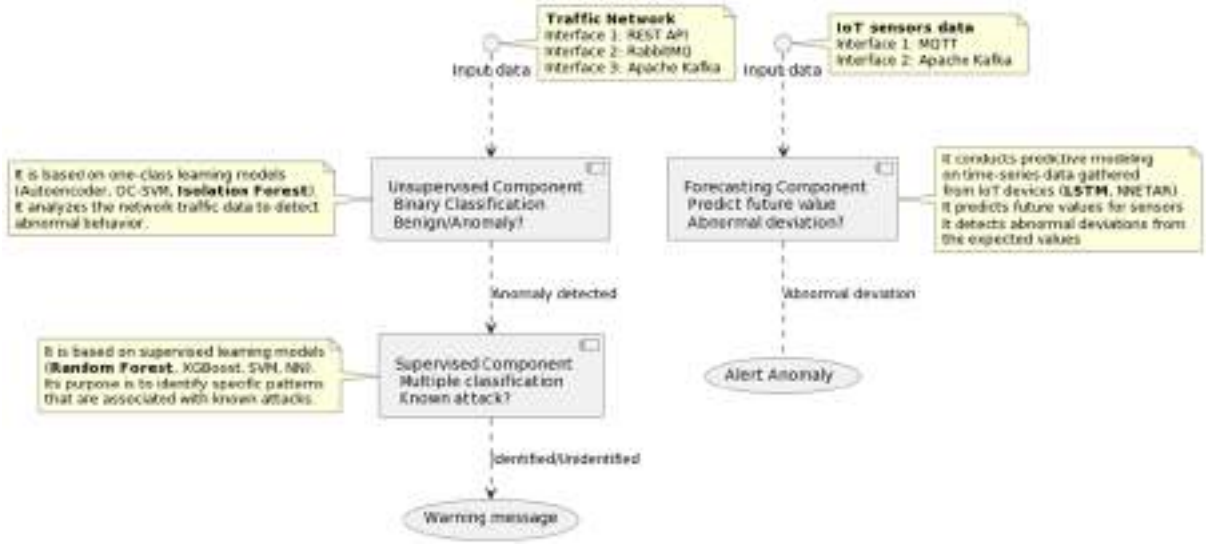

Figure 33. PMEM architecture design.

5.2.2.1 Attack detection module

The input of the Attack detection module is based on the network traffic of the monitored system, and it is responsible of detecting specific cyberattacks within the input data. This module contains two different components: the unsupervised and supervised component (see Figure 33). The input flow goes through the first component, which has as objective to carry out a binary classification of the inputs to the system, assigning to each one of the flows the category of "benign entry" or "anomaly." When an entry is detected as an anomaly, it is forwarded to the second component, where various classifications are made to detect the attack type, if it is a known attack. For zero-day attacks, the anomaly is classified as an "unknown attack". Figure 34 illustrates the process of detecting and classifying attacks in the PMEM attack detection module.

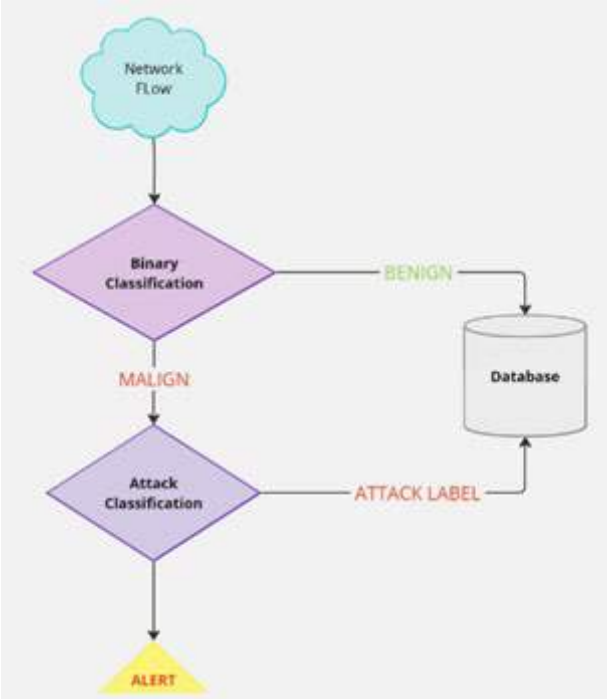

Figure 34. Attack detection and classification flow of the attack detection module of PMEM.

The first component of the Attack detection module is based on a semi-supervised one-class ML model. This model, based on the Isolation Forest ML algorithm, has been trained using a benign entries dataset (collected from the system that is monitored).

The second component uses an ensemble of different models based on supervised ML algorithms. Each model has been trained to detect one particular known attack. This ensemble has been implemented in such a way that models are applied in parallel. Then, the whole process runs significantly faster than a simple serial application of all of them.

5.2.2.2 Forecasting module

The forecasting component predicts future values to detect abnormal deviations from the expected values. This component is based on ML forecasting models trained with time series data. It receives input data from multiple IoT devices that capture measurements periodically. A time series model has been developed for each sensor and measurement to forecast future values based on historical data.

This component is implemented using Deep Learning models specialised for time series tasks. The model used is Long Short-Term Memory (LSTM), a model based on recurrent neural networks (RNNs) designed to analyse sequential data. An LSTM model has been trained for each sensor using the collected historical measurement data. Furthermore, the models undergo continuous refinement and monitoring to ensure optimal performance and reliability in forecasting future trends. To identify

deviations, a maximum and minimum threshold have been computed for each time series data. These thresholds serve as boundaries to flag any values that fall outside the expected range. When a predicted data point surpasses the maximum or minimum threshold, it triggers an alert, indicating a potential anomaly.

5.2.3 Attack response and recommendation

The attack response and recommendation module makes use of the detection and prediction achieved from PMEM to make reasoning about possibilities of different violations and anomalies that are being observed in the monitored assets. Based on the detected and predicted attacks, the module provided the list of recommended mitigation actions aligned with the MITRE ATT&CK framework. The module is responsible for mapping the detected violation and anomalies with the list of the potential attacks. Based on the nature of the attacks and their impact on the assets, it provides the list of the recommended actions. These lists of actions help the network administrator to mitigate the ongoing cascading effects of the attacks and will help in taking timely actions to avoid the cause of the malfunctioning. These recommendations will be used to ultimately enforce actions in the lower infrastructure. For the first version of the enabler, this module is focused on providing the list of recommended actions for the DoS and false data modifications attacks. It sees the endpoint where the anomalies are detected and based on this information provides a suitable suggestion. These lists of actions are predefined and are mapped using the MITRE ATT&CK framework.

5.3 Implementation Details

This section presents the implementation done for the Attack Prediction, Response Recommendation & Adaptation Enabler for the Energy and Transport use cases. It details how the PMEM and Attack Response and Recommendation sub-modules are implemented, as well as the data collection process that has been conducted for both use cases in the energy and transport test-beds developed for the PHOENI2X framework.

5.3.1 Tools

This section describes the main tools used for PMEM implementation, taking into account the PHOENI2X use cases.

- **Python**[19]: The programming language used is Python, which is a high-level, interpreted programming language. The main library used are the following:
  - **Pandas**[20]: An open-source data manipulation and analysis library. It provides data structures and functions for efficiently working with structured data such as tabular data, time-series data, etc.
  - **Scikit-learn**[21]: An open-source ML library. It provides tools for data mining and data analysis, including various algorithms for classification, regression, clustering, and more.
  - **TensorFlow**[22]: An open-source ML framework widely used for building, training, and deploying ML models, especially neural networks.
  - **FastAPI**[23]: It is a web framework for building APIs in Python. This tool is known for its ease of use, speed, and support for asynchronous programming.
  - **Psycopg2**[24]: It is a PostgreSQL Python adapter that enables Python scripts to interact with PostgreSQL databases by providing an interface for executing SQL commands and managing database connections.

[19] https://www.python.org/
[20] https://pandas.pydata.org/
[21] https://scikit-learn.org/
[22] https://www.tensorflow.org/
[23] https://fastapi.tiangolo.com/
[24] https://pypi.org/project/psycopg2/

  - **Paho-mqtt**[25]**:** A MQTT client library for Python which provides functions to publish and subscribe to MQTT (Message Queuing Telemetry Transport) messages.
  - **Elasticsearch**[26]: It is a Python client for Elasticsearch which provides functions to interact with Elasticsearch clusters from Python scripts. It enables the execution of various operations such as indexing, searching, updating, and deleting documents in Elasticsearch indexes.
  - **Requests**[27]: It is an HTTP library for Python which enables sending HTTP requests and processing responses.
- **PostgreSQL**[28]: An open-source relational database management system known for its reliability, robustness, and feature completeness. PostgreSQL supports various advanced features like JSON support, transactions, indexing, and more.
- **MQTT**[29] (Message Queuing Telemetry Transport): It is a publish-subscribe messaging protocol designed for constrained devices and low-bandwidth, high-latency, or unreliable networks. This protocol is used in IoT applications for efficient communication between devices.
- **Kafka**[30]: An open-source distributed event streaming platform developed by the Apache Software Foundation. Kafka is designed for high-throughput, fault-tolerant, and scalable event streaming. This tool is suitable for building real-time data pipelines and streaming applications.
- **Grafana**[31]: An open-source analytics and monitoring platform that allows to visualize and analyse metrics from multiple data sources. Grafana provides a rich set of features for creating dashboards, graphs, and alerts.
- **Docker**[32]: A platform for developing, shipping, and running applications in containers. Docker containers encapsulate applications and their dependencies, providing a consistent environment for development and deployment across different systems.

5.3.2 Enabler implementation for the Energy Use Case

The Energy use case monitors network traffic and data from smart meters within the energy critical infrastructure. The following technologies are used to capture data, process it, forecast evolution and detect potential anomalies:

- Gurux python scripts to interact with the DLMS protocol
- A kafka server to transmit information between relevant nodes
- PostreSQL and the local file system to store information
- A combination of different Python libraries (Tensorflow, pandas and others) to process them
- Grafana to show a local dashboard of the state.

5.3.2.1 Data collection

The collection of the relevant data is made through the DLMS protocol [16]. This communication is performed between a Python script executed in a VM and Ami Headend, which in turn accesses the specific smart meters' data. The received DLMS messages are then logged to a file and broadcasted to the respective topic in the Kafka server. On the forecasting and processing end of the communication, there is a script subscribed to the topic which receives the messages and analyses them when it receives a complete measure reading. Figure 35 illustrates the complete data collection process.

[25] https://pypi.org/project/paho-mqtt/
[26] https://www.elastic.co/es/elasticsearch
[27] https://pypi.org/project/requests/
[28] https://www.postgresql.org/
[29] https://mqtt.org/
[30] https://kafka.apache.org/
[31] https://grafana.com/
[32] https://www.docker.com/

Figure 35. Communication on the Energy testbed premises.

5.3.2.2 Modelling

The ML model used in the Energy use case is shown in Figure 36. Initially, the previous N measures are taken, where in our scenario, N is set to 10, which are then scaled. Subsequently, these values are passed through a LTSM and a dense neuron layer to obtain the next measure. Finally, we inverse the scaling to obtain the value in the same range as the real measurements.

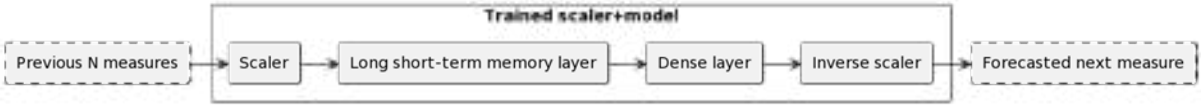

Figure 36. Predictive Model structure – Energy Use Case.

The models for each measure identifier are trained (or retrained) every two days, following the steps outlined in Figure 37. First, all the valid samples for the measure are obtained. Then, these measures are pre-processed, obtaining the scaler and the corresponding dataset for training. Afterwards, the existing model is obtained if available, or a new untrained model is generated. Finally, the model is trained or retrained and stored in the file system.

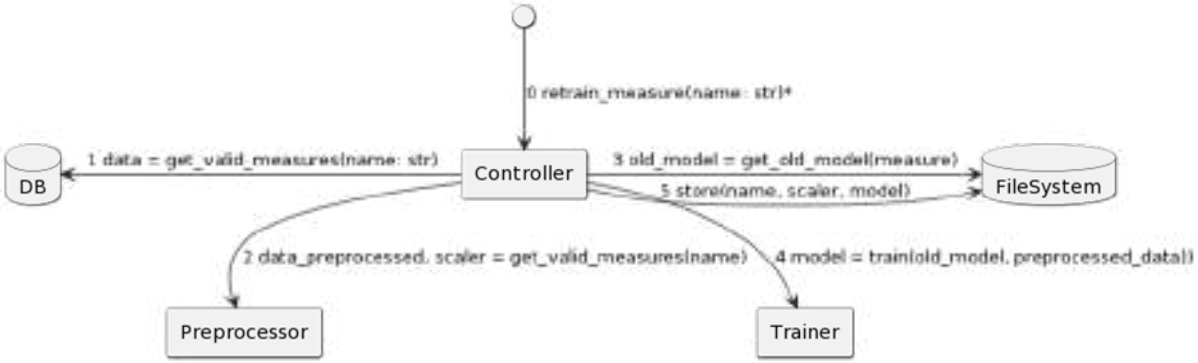

Figure 37. Predictive Model training flow – Energy Use Case.

5.3.2.3 Measure reception and forecasting

Every time a valid complete measure is received, the steps showcased in the Figure 38 are taken. If a forecasted value exists for the real value received, the differences are computed as prediction error, and the database is updated accordingly. When there are at least 300 forecasted values with their corresponding errors computed, anomaly detection is performed. Finally, the next N values given the current information are forecasted.

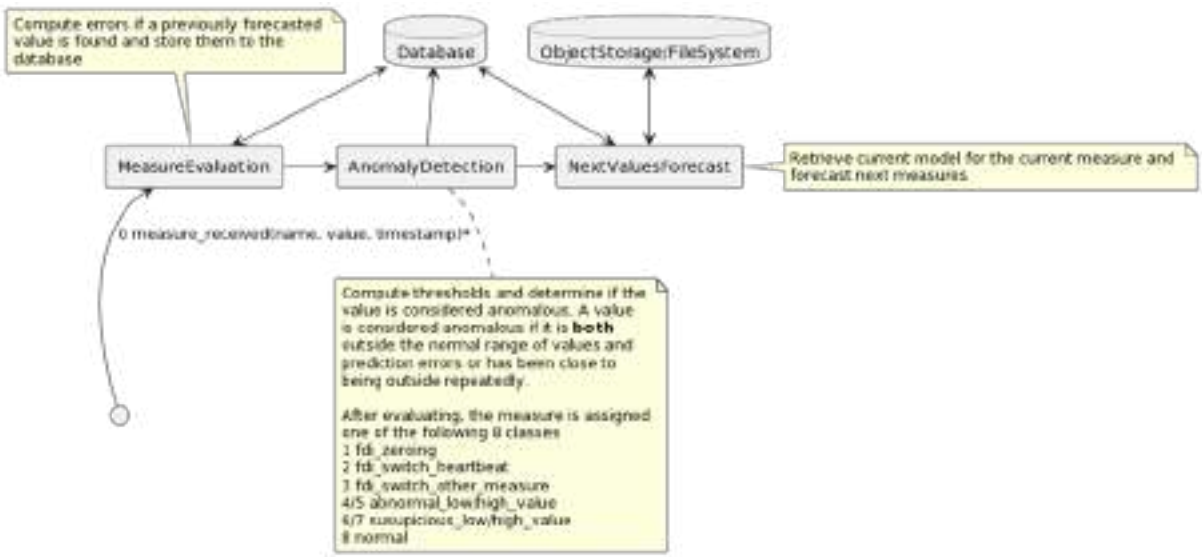

Figure 38. Measure reception flow – Energy Use Case.

5.3.2.4 Anomaly detection

Anomaly detection involves computing threshold barriers and subsequently checking if values fall outside them. In this case, the threshold barriers are derived from the interior/exterior barriers of a boxplot. If a value and its error are outside of the external barrier (either inferior or superior), the sample is considered anomalous. Additionally, if the value remains outside the interior barriers on the same side for more than three samples, it is considered too anomalous.

The rationale behind identifying if a sample is abnormal is as follows. If a sample lies beyond the typical value range but the ML model predicts it with low error, it indicates a known behaviour. On the contrary, if the model fails, but the value is typical in the distribution, then the model isn't adequately tuned to the smart meter's usual behaviour of the smart meter. However, if a value falls outside the normal range and the model fails to predict the movement, then it is an anomalous value and should be analysed. The reason to also trigger an analysis when a value remains within the same interior and exterior barriers is justified by the low probability of such an occurrence in a true random variable.

When further analysis of the measure is triggered, it is first compared to known special values that certain attacks might generate, such as 0 (indicative of a zeroing attack) and "0x0000280000FF" represented as an integer (associated with a heartbeat switch attack). If it isn’t a known value, the obtained measure is compared to the rest of the available measures barriers to check if it would fit another known smart meter time series. If a match is found, it suggests a high likelihood of a switch attack involving another measure.

5.3.2.5 Features Storage

The messages received from the Kafka server on the processing side are stored as files, as depicted in Figure 39. In case the main database is lost, the code is able to regenerate it using the stored files. During the regeneration process (“catching up”), the new messages are stored as log files. Once the program has processed all the files, including both old and newly generated ones, normal execution is resumed.

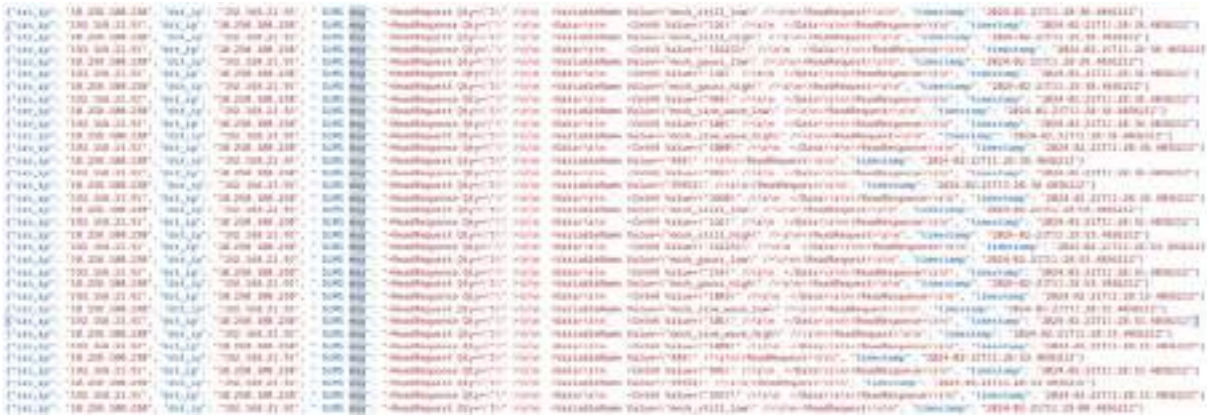

Figure 39. File system with log files.

Figure 40 shows the table that stores the valid measurements. The database contains an index based on a B-Tree to quickly retrieve the relevant sorted data with low asymptotic complexity when given a measure name and a timestamp range. The Features storage database comprises the information detailed in Table 5.

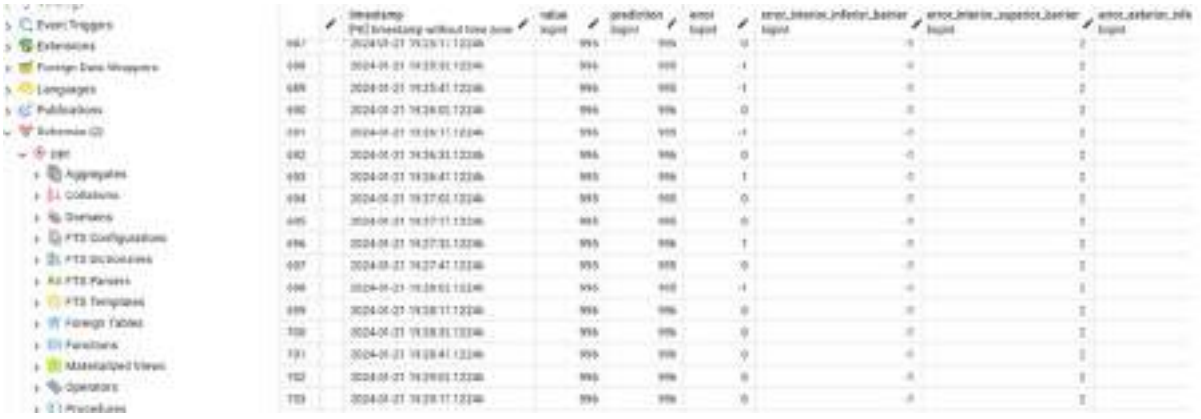

Figure 40. Sample of database contents accessed used pgAdmin.

Table 5. Features storage database for the Energy Use Case.

Feature	Description
Non null measure name	Identifies the specific smart meter that generated the sample
Non null timestamp	Identifies the moment when the sample was obtained
Nullable value	Stores the value of the received sample or NULL if the row only contains a forecast
Nullable prediction	Stores the value of the forecasted sample or NULL if the row only contains a received sample value because the ML model was not available when the prediction was attempted or there was not enough real data
Nullable error	Stores the difference between the real and forecasted value or NULL if we don't have real and predicted values
Nullable interior inferior barrier for the error	Stores the interior inferior barrier of the last 300 prediction errors or NULL if they aren't available
Nullable interior superior barrier for the error	Stores the interior superior barrier of the last 300 prediction errors or NULL if they aren't available
Nullable exterior inferior barrier for the error	Stores the exterior inferior barrier of the last 300 prediction errors or NULL if they aren't available
Nullable exterior superior barrier for the error	Stores the exterior superior barrier of the last 300 prediction errors or NULL if they aren't available
Nullable interior inferior barrier for the values	Stores the interior inferior barrier of the last 300 received values or NULL if they aren't available

Feature	Description
Nullable interior superior barrier for the values	Stores the interior superior barrier of the last 300 received values or NULL if they aren't available
Nullable exterior inferior barrier for the values	Stores the exterior inferior barrier of the last 300 received values or NULL if they aren't available
Nullable exterior superior barrier for the values	Stores the exterior superior barrier of the last 300 received values or NULL if they aren't available
Nullable abnormality indicator	Stores if the sample is considered abnormal or NULL if the barriers aren't available
Nullable evaluation name	Stores the evaluation text or NULL if the barriers aren't available. The current evaluation names are the following: - normal: for the values inside the barriers - abnormal_low_value: for the values below both the error and value barriers or consecutive suspicious values of the same kind without a known attack name. - abnormal_high_value: for the values above both the error and value barriers or consecutive suspicious values of the same kind without a known attack name. - suspicious_low_value: for the values between the interior and exterior inferior barriers. - suspicious_high_value: for the values between the interior and exterior superior barriers. - fdi_zeroing: for the anomalous values identified as zeroing. - fdi_switch_heartbeat: for the anomalous values identified as switch with heartbeat. - fdi_switch_other_measure: for the anomalous values identified as switch with other measure.

5.3.2.6 Dashboard

The dashboard (see Figure 41) provides a user-friendly interface for monitoring the current system status. Users can select the desired measure from a dropdown menu. The first graph displays the evolution of received sample values (continuous green line), predicted values (dotted yellow line), the range between interior barriers (green area), and the range between interior and exterior barriers (red-orange area). Below, a similar graph shows prediction errors. In the bottom-left section, a table lists anomalies along with their timestamps, values, predictions, errors, and evaluation names. Finally, in the bottom-right corner, forecasted values with their corresponding timestamps are displayed.

Figure 41. Attack Prediction, Response Recommendation & Adaptation Enabler Dashboard – Energy Use Case.

5.3.2.7 Evaluation

The forecasting component used in the energy use case was derived from the provided smart meter data. Specifically, the latest 1000 normal measurements of each smart meter were retrieved. Non-

normal measurements were intentionally introduced in the testing process and were automatically identified and manually revised.

For each measurement type, we conducted a 70/30 train-test split, the model was trained, validated, and Mean Absolute Error (MAE), Mean Squared Error (MSE), and Mean Percentage Error (MPE) were calculated. The MPE metric is particularly valuable for comparing models across different types of measurements, as some measurements exhibit large magnitudes. In the case of the poorest performing measurement (with an MPE of 15.91%), there is a notable sharp decrease in value at certain points, where the value drops to approximately 0.03% of its original value.

5.3.3 Enabler implementation for the Transport use case

This section describes the development carried out for AI-based anomaly detection in the Transport use case, based on the analysis of LoRaWAN network traffic metadata and the measurements obtained from the railway sensors.

5.3.3.1 Attack detection module

The primary objective is to design and implement an anomaly detection component capable of analysing the LoRaWAN network traffic metadata in the railway sensors' network and predict whether the packets are benign or anomalous. The anomaly detection is based on specific numeric LoRaWAN metadata fields. If a packet contains an unusual value (outliers) in any of these fields, then it is flagged as potential anomaly.

To implement this component, the process begins with the collection of network traffic data generated by the sensors. Then, this data is pre-processed and used to train an unsupervised ML algorithm to perform anomaly detection. All collected data samples from the sensors used to train the model are considered "benign". The collected traffic data and the output of the model are stored in a database. The model's results can be accessed through an API or a dashboard interface.

This component makes use of the following tools:

- Data collection: paho-mqtt (Python)
- Data pre-processing: pandas, numpy (Python)
- Model generation: scikit-learn framework (Python)
- Storage: PostgreSQL[33] database
- Dashboard: Grafana
- API to retrieve the results stored in the database: Fast API[34].

5.3.3.1.1 Data collection

The first task consists of collecting all the LoRaWAN network metadata in the Transport network into a central database. A Python script has been created to execute this task, as depicted in the diagram in Figure 42. The script uses the paho-mqtt library to subscribe to the provided MQTT broker (*eu1.cloud.thethings.industries*) to load the network metadata using the root topic to receive all the transmitted MQTT messages. Each time a MQTT message is received, the script extracts all the fields and values of the message in a tabular format and are stored into the PostgreSQL database. The sensors messages are stored in the following three different tables under the "ws_lorawan" schema:

- *uplink_messages*: this table stores the uplink messages, which corresponds to the messages that are sent by end devices to the Network Server.
- *downlink_messages*: this table stores the downlink messages, which are sent by the Network Server to only one end device.

[33] https://www.postgresql.org/
[34] https://fastapi.tiangolo.com/

- *mqtt_messages*: this table stores in raw format all the MQTT messages received from the MQTT broker.

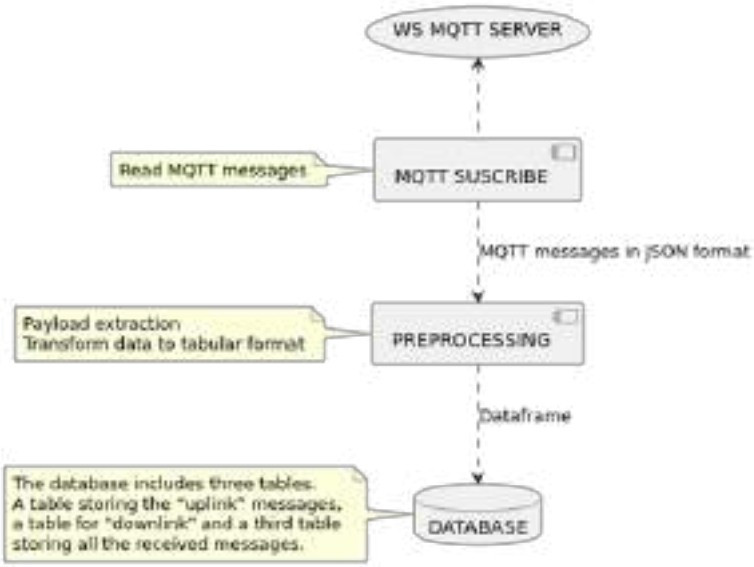

Figure 42. Data collection process for the Transport use case.

Thus, each time a new MQTT message is received, it is stored into the *mqtt_message* table in the database. Subsequently, it is checked if the message is an uplink or downlink and then stored in tabular format into the corresponding table.

5.3.3.1.2 Modelling

The chosen model for anomaly detection is Insolation Forest, which is an unsupervised model that uses binary trees to efficiently isolate anomalies within a set of data samples. It works by isolating observations in the dataset by randomly selecting a feature and then randomly selecting a split value between the maximum and minimum values of that selected feature. This process is repeated recursively to create a set of isolation trees. The anomalies are identified by the algorithm as instances that require only a few splits to be isolated from the rest of the data points, as opposed to normal instances that typically require more splits.

The anomaly detection model was trained using uplink messages collected from the sensors. Prior to training the model, we conducted feature selection (all the fields with unique or distinct values were removed from the dataset) and data standardization to the training data to enhance its ability to accurately detect anomalies.

Once the model is trained, it is exported to perform anomaly detection. Since we are continuously collecting data from the sensors, the model is continuously updated and being retrained using the accumulative new collected data.

5.3.3.1.3 Anomaly detection

The anomaly detection component (see Figure 43) analyses the input traffic messages transmitted by the sensors to detect if they are normal or anomalies. This component receives the input data via an MQTT client that is subscribed to the provided MQTT broker. Each time a new message is received, it is pre-processed by extracting the payload fields, selecting the features used by the model, and standardizing numerical features. Then, it is transmitted to the unsupervised component, which uses the Isolation Forest model trained with collected normal traffic network data. The model analyses the input data and checks if the input fields have any abnormal value.

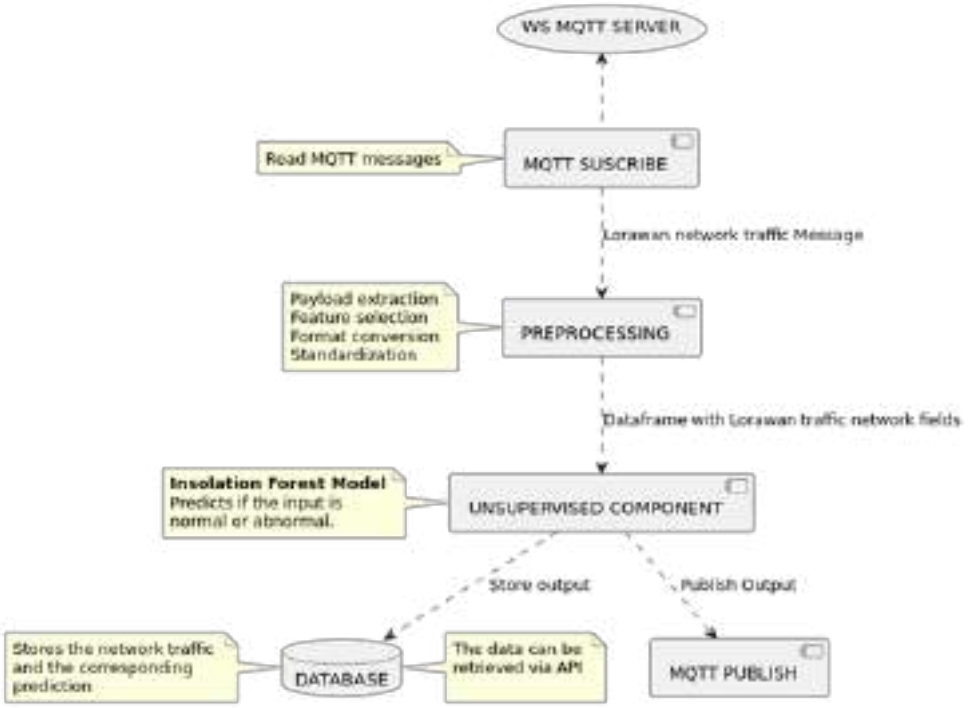

Figure 43. Anomaly detection process for network traffic data in the transport use case.

Once the model has performed the prediction, both the input message and the model output are stored in the *predicted_uplink_messages* table of the PostgreSQL database. This component analyses specifically the uplink messages, which are the messages transmitted by the sensors to the Network Server. The results stored into the database can be accessed through an API that reads the contents stored in the predictions table. Moreover, the results of the anomaly detection component are also published in real time through a local MQTT broker.

5.3.3.1.4 Features Storage

All the information stored in the database for this specific use case are named under the same schema, named “ws_lorawan”. This schema includes the following tables:

- *mqtt_messages*: This table (see Figure 44), stores all the packets sent by the sensors in raw format. It includes three fields, the timestamp when the packet is received, the MQTT topic and the message.
- *uplink_messages*: This table stores the uplink messages in tabular format.
- *downlink_messages*: This table stores the downlink messages in tabular format.
- *predicted_uplink_messages*: This table stores the output of the anomaly detection component for the uplink messages. It stores both, the information of the uplink message and the corresponding “Label” returned by the model.

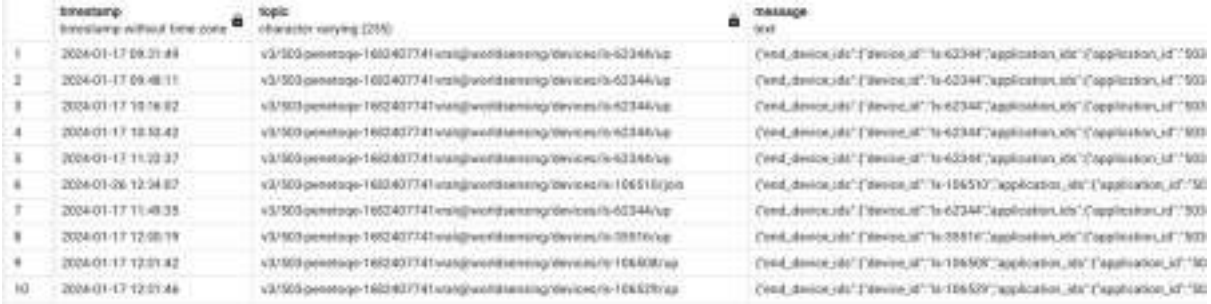

Figure 44. mqtt_messages table in PostgreSQL database.

5.3.3.1.5 Results

A Grafana dashboard was designed to visualize the results of the anomaly detection component, as illustrated in Figure 45 and Figure 46. The dashboard consists of two main tables: one showcasing data packet identified as normal by the unsupervised model (highlighted in green), and the other displaying packets identified as suspicious (highlighted in red). For each packet, the tables indicate the source device, the timestamp, and the main LoRaWAN network metadata configuration fields. The dashboard additionally includes graphical plots that summarize the information of specific packet attributes, such as RSSI, SNR, and F_count distributions of the collected packets.

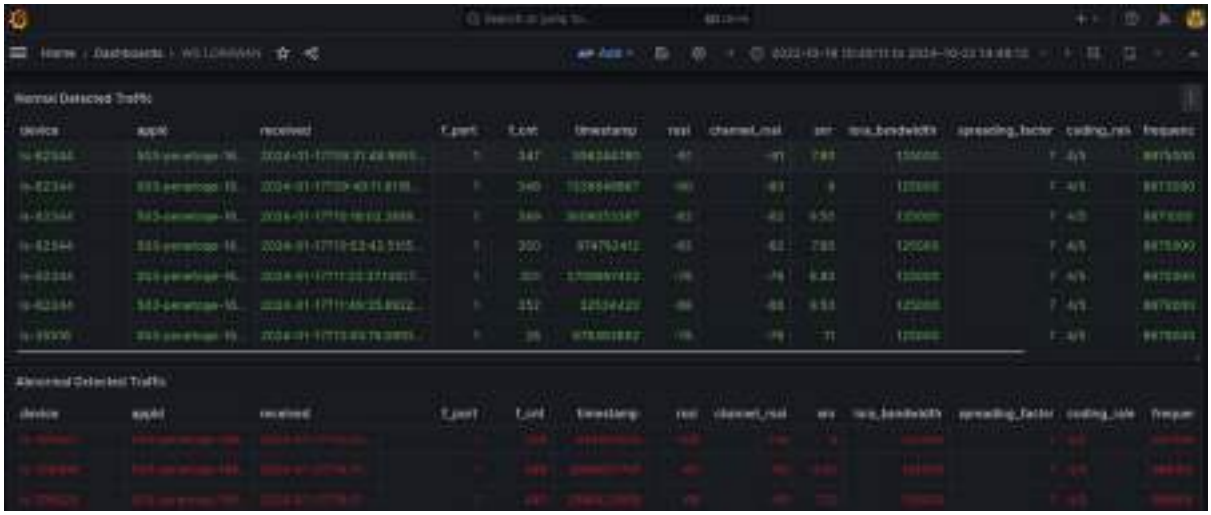

Figure 45. Grafana dashboard illustrating the anomaly detection output in tables.

Figure 46. Grafana dashboard illustrating plots about specific features of the anomaly detection output.

5.3.3.2 Forecasting module

5.3.3.2.1 Overview

The primary objective of this use case is to forecast future measurements obtained from sensors deployed in the railway testbed. Specifically, the focus lies on forecasting four variables: temperature, channel_1, channel_2 and kN. Railway sensors are categorized into two main groups: one group transmits data every 6 hours, while the other group transmits data every 30 minutes, as depicted in Figure 47.

LS-G6-DIG2 nodes acquire data every 6 hours. This data comes from a chain of inclinometers. Two nodes have 16 inclinometer sensors, while the other two 26. The sensors measure temperature values, as well as the sine of the angle of the A axis and the B axis. Data from these nodes include measurements from 51 sensors, covering variables such as temperature, channel_1, and channel_2. On the other hand, the LS62344 node of type LS-G6-VOLT-4 acquires data every 30m. This data is from two load cells (see Figure 47). The values obtained are the gross value of the production of a Wheatstone bridge, from which we then calculate its engineering units, in kN (kilo Newtons) and in Tn (Tons). Since the sensors of this node both are measuring the same parameter but in different units, we have chosen, for consistency, to process the data only in kilonewtons (kN).

Figure 47. Summary of the IoT sensors in the transport use case.

The implementation process of this component starts by collecting the data from the sensors and storing it into a storage system. Subsequently, the collected data is used to train a time series model for forecasting purposes, and this trained model is then deployed to perform prediction forecasting. The tools used for the implementation of this use case are:

- Python scripts that subscribe to the MQTT broker to read the messages transmitted by the sensors.
- Several Python libraries (Pandas, Numpy, Scikit-learn) to pre-process the received data.
- PostgreSQL to store the input (sensors data) and the output (sensors data and the component output) .
- TensorFlow for the model generation.
- Grafana for the design of a user interface to visualize the component output.

5.3.3.2.2 Data collection

The data of the sensors is transmitted through an MQTT broker. Therefore, a MQTT client is set up, which subscribes reads all the messages published by the sensors from the host *mqtt-external-cmt-phoeni2x.wocs3.com* and the following topic *e00467b7-4c06-4cd4-bd35-22ebd827e1ed/02754340-5ef2-11ee-a7f7-83c1d448950d/#*. The messages from the sensors are received in JSON format with the following fields: *messageId, mqttMessageId, readTimestamp, deviceId* and *payload.*

Each time a message is received, it is pre-processed and then stored into the database. For each message, we remove the MQTT fields and extract the "*readTimestamp*" and "*payload*" fields. The "*payload"* field includes one or multiple measurements (variables) values. The messages are filtered according to specific variables' names that appear in the payload, thus, the messages that do not include the variable names in Table 6 will be ignored. Figure 48 illustrates the complete process.

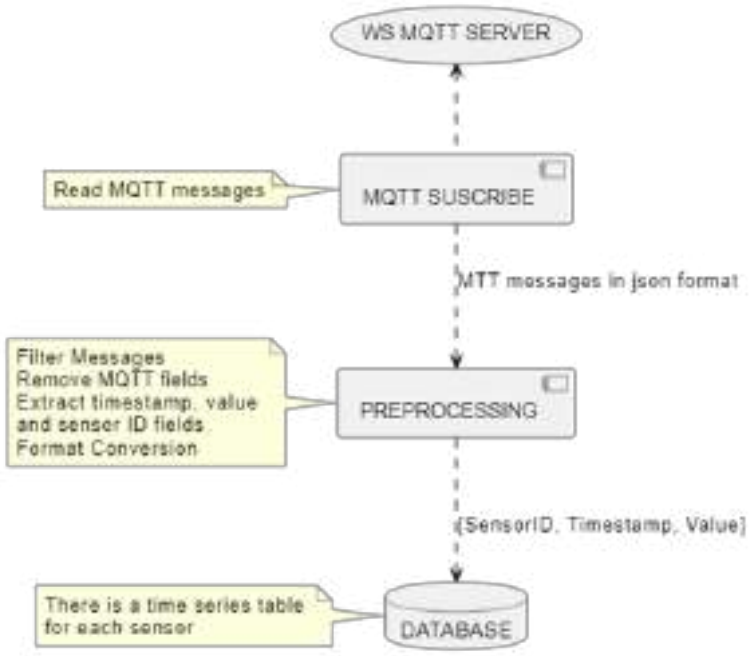

Figure 48. Data collection process for the forecasting WS use case.

In total, 53 sensors were registered, 51 of these sensors measure the variables "*channel_1*", "*channel_2"* and "*temperature"*, while the other 2 sensors measure the "*kN*". Therefore, there are in total 155 time series. For each time series, the data is collected and stored in its corresponding table in the database. For each time series, we store the timestamp and the measured value.

Table 6. Sensors' details in the transport use case.

Variable Name	Number of sensors	Nodes	Transmission time
channel_1	51	ls_106508hw ls_106510hw ls_106527hw ls_106529hw	Each 6 hours
channel_2	51	ls_106508hw ls_106510hw ls_106527hw ls_106529hw	Each 6 hours
temperature	51	ls_106508hw ls_106510hw ls_106527hw ls_106529hw	Each 6 hours
kN	2	ls_62344hw_gauge	Each 0.5 hours

Moreover, for backup reasons, all the messages received are stored in raw format into a JSON file.

5.3.3.2.3 Modelling

For each time series, a recurrent neural network (RNN) model has been trained, using the collected time series data to forecast future variable measurement values. Specifically, the model employed is a Long Short-Term Memory (LSTM), which is a type of recurrent neural network architecture designed to process and predict sequences of data. Unlike traditional RNNs, LSTM networks are capable of learning long-term dependencies in data by using a specialized memory cell and gating mechanisms.

The LSTM neural network has been defined using Keras[35], the model is composed by the input layer, the LSTM layer and a dense output layer:

- *Input Shape*: The model expects input data in the shape of (10, 1), where 10 is the number of time steps in each input sequence, and 1 is the number of features at each time step.
- *LSTM Layer*: The LSTM layer has 100 units (neurons) and uses the ReLU activation function. This layer processes the input sequences and learns to capture temporal dependencies within the data.
- *Dense Layer*: This layer consists of a single neuron responsible for predicting a single value.

The model was compiled using the Adam optimizer[36] and mean squared error (MSE) loss function. Table 7 summarizes the model configuration.

Table 7. Model configuration - transport use case.

Parameter	Value
Layers	Input layer LSTM layer Dense layer
Activation function	ReLU
Number LSTM unit	100
Number steps	10
Number features	1
Epochs	100
Optimizer	Adam
Loss	MSE

For each time series, a LSTM model was trained and exported to perform forecasting. The model is configured to be retrained each period of time in order to update it using the new collected data.

5.3.3.2.4 Measure reception and forecasting

The forecasting component uses the trained LSTM model to predict the next time series value using the N previous registered values. Figure 49 shows the process to perform predictions. The process starts at timestamp T performing the following actions:

1. The model predicts the next time series value for timestamp T + 1 and stores the output in the database.
2. The system receives and stores the corresponding real value measures by the sensor with its corresponding prediction made by the model for timestamp T.

Each time a message is received, it is pre-processed (filter the message, remove MQTT fields, extract timestamp and payload field) and then stored to the database.

[35] https://keras.io/
[36] https://keras.io/api/optimizers/adam/

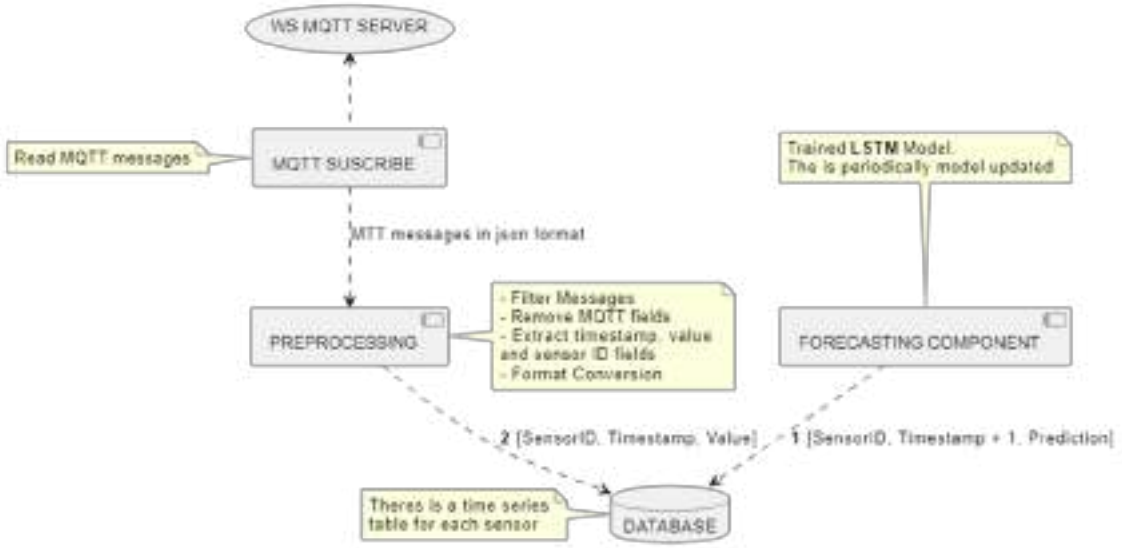

Figure 49. Forecasting and sensor measurement reception process of the WS use case.

The system follows this loop process by performing predictions and receiving the real measure value by the sensors. The models of each time series are updated periodically using the new real received data.

5.3.3.2.5 Features Storage

The features storage system of the forecasting component for the railway use case is a PostgreSQL database. All the tables of this use case are stored under the "ws_forecast" schema. This schema includes several tables, which can be classified intro the following categories:

- *Data collection tables:* these tables include two fields, the timestamp and the measurement value. Each table stores the historical measurements transmitted by a sensor for a specific variable. Thus, each table is named as "variableID_sensorID".
- *Output tables:* these tables store the (historic) results of the forecasting component (component output). These tables have three fields: timestamp, real measurement value, and the predicted value by the model. Similarly, each table stores the data for a specific variable and sensor. Each table has as name "predicted_variableID_sensorID".
- *Next predictions table*: this table stores only the forecasted values for the next timestamp T. It includes two fields: the time series ID ("variableID_sensorID") and the predicted value. This table stores the latest predictions made by the model for each time series. When the model performs a new prediction, it then updates the prediction field for the corresponding time series.

5.3.3.2.6 Results

A Grafana dashboard was designed to allow the illustration of the forecasting component output. The dashboard provides a time series plot that graphically shows the historic of the time series evolution, in green are the real measurements captured by the sensor and in yellow the prediction made by the LSTM model. This information is also provided as a table form, which includes for each timestamp the real value of the measurement and the predicted value, and also an additional column including the absolute error value (difference between the predicted value and the real value). Finally, the dashboard also shows the value forecasted by the model for the next timestamp.

The dashboard shows this information related to a specific time series. Figure 50 shows the dashboard for the variable "kN" of the sensor "ls_62344_gauge_4". It is worth noting that the dashboard provides a dropdown menu, which allows users to select the measure variable and sensor ID of interest.

Figure 50. Grafana dashboard which illustrates the output of the forecasting component for the transport use case.

5.3.3.2.7 Evaluation

The forecasting component was trained using the historical time series data collected from the sensors. The models were evaluated based on the differences between the predicted values made by the model and the real values collected from the sensors. Hence, the metric used for this evaluation is Mean Absolute Error (MAE), which calculates the average absolute difference between each predicted value and its corresponding true value. Table 8 displays the performance results for each variable measured by different sensors. Due to the extensive number of models evaluated, only a subset of the performance evaluations is presented in the table.

Table 8. Model evaluation – forecasting component.

Model	Time series		MAE
	Variable	Sensor	
Model 1	kN	62344hw_gauge_4	1.494
Model 2	kN	62344hw_gauge_2	30.498
Model 3	Temperature	ls_106508hw_2	2.313
Model 4	Temperature	ls_106508hw_5	0.167
Model 5	channel_2	ls_106527hw_14	9.03e-06
Model 6	channel_2	106529hw_8	0.0002
Model 7	channel_1	ls_106529hw_5	0.608
Model 8	channel_1	ls_106529hw_7	3.11e-05

5.4 Recap of Current Status & Next Steps

PMEM has been designed and implemented for both the Energy and Transport use cases. Initially, a study was performed to design the PMEM architecture and its corresponding components (supervised, non-supervised and forecasting components), involving the following tasks performed:

- **Data collection:** It has been implemented the data collection process for both use cases. The data is received in real-time, pre-processed and stored in a database system. The corresponding dataset is generated.
- **Modelling**: A study was carried out on different ML algorithms for the forecasting component. Several models were trained and evaluated for both use cases, and the models with better performance were selected. In addition to the forecasting component, the unsupervised component was also implemented for the transport use case following the same approach.

- **Storage system:** A storage structure system was designed to store both the input data that each component receives and the generated output.
- **Containerization**: Both components (forecasting & anomaly detection) were containerized using Docker containers to facilitate the importation and deployment of implemented modules.
- **Integration**: The connection of PMEM's output to the ROAR has been implemented, and the appropriate playbooks are triggered when an attack or anomaly is detected.
- **Component validation:** Several tests were performed to assess the performance of PMEM in detecting anomalies. This process is still in progress.

Taking advantage of PMEM's ability to forecast based on time series data, next steps will include the ability to apply this functionality to forecast real-time network traffic data. Using ML models trained with historical network traffic data and ML models for forecasting will allow us to anticipate possible threats or suspicious behaviour patterns before they materialise in actual attacks. Improve system performance by training the supervised component with labelled network traffic data from the Energy and Transport use cases. Moreover, it is planned to demonstrate PMEM in the Health use case in the second period of the project.

6 Risk Impact Assessment, Alert Triage & Response Prioritisation Enabler

6.1 Overview

6.1.1 The importance of risk assessing

Risk assessment plays a significant role in preserving the integrity, confidentiality, and availability of digital assets within the network infrastructures of many types of entities, including businesses, governments, and educational institutions. Methodical pinpointing of potential threats and vulnerabilities allows the assessment and prioritization of security challenges, making it easier to implement targeted and efficient mitigation strategies.

A key benefit of risk assessment in Cybersecurity is its proactive nature. Through regular assessments, organizations can anticipate and address potential weaknesses in their network infrastructures, thereby, reducing the risk of successful attacks. This empowers entities to stay a step ahead of evolving threats, enhancing their resilience against malicious actors.

Cyber-risk evaluation further encourages informed decision-making procedures by quantifying risks and their potential impacts, so that organizations may gain valuable insights into resource allocation, allowing them to direct their investments more efficiently. This approach ensures that the resources are dedicated to the most critical actions to be taken. Additionally, risk assessment may promote a culture of continuous improvement, enabling entities to adapt their security measures in response to emerging threats and changes in their operational environments.

Consequently, cybersecurity risk assessment should be considered as a fundamental component in establishing robust and persistent network infrastructures comprising digital assets in any business sector.

6.1.2 Traditional approaches

One recurrent issue in traditional approaches to risk assessment is the tendency to tackle the problem through a fixed set of rules or policies (i.e., adhoc tables or Excel sheets with predefined rules observed from past experiences). While such methods provide a structured framework for evaluating risks, they commonly oversimplify the evaluation or make it “static”. In many circumstances, these fixed approaches fail to capture the dynamic nature of cybersecurity threats, as well as the vast landscape of technology and business operations.

Consequently, organizations may overlook at emerging threats, failing to adapt their strategies accordingly. Moreover, relying on a predefined set of policies may contribute to creating a false perception of security, which may not be the case considering the inter- and intra- dependencies involved in modern, complex network infrastructures. To address this challenge, companies must adopt flexible and adaptive risk assessment methodologies that incorporate continuous monitoring, threat intelligence, and risk-based decision-making processes. This change enables organizations to better anticipate and respond to emerging threats, enhancing their resilience towards cyberattacks in an ever-changing security landscape.

6.1.3 Relevance to PHOENI2X Project

The approach considered is presented as a key component for critical infrastructures, such as energy, transport, health and other essential services providers. These infrastructures serve as backbone for the proper functioning of modern societies, providing indispensable resources and services for daily life and economic activities. Given their critical nature, these may become prime targets for malicious cyber actors seeking to disrupt operations, cause widespread damage or compromise general safety.

Thus, conducting regular, comprehensive assessments becomes imperative to identify and mitigate potential vulnerabilities that could lead to catastrophic consequences.

In this project, cyber risk assessment allows the involved organizations to anticipate and prepare for emerging threats, ensuring adaptability of security strategies to address evolving risks effectively. Continuous monitoring and reassessment of their risk strategic policies may enable these critical infrastructures to maintain operational continuity and minimize the likelihood of cyber incidents which could disrupt essential services and jeopardize the public safety.

6.2 Research & Design Details

6.2.1 How CERCA deals with the problem of risk assessment

The CERCA tool serves as a key feature in the infrastructure of the pilots, offering a holistic approach to risk assessment and management. Its major advantages over traditional risk workarounds include the following key-points:

1. **Flexibility and adaptability:** Unlike rule-based approaches, CERCA is flexible and adaptable to the dynamic nature of cybersecurity threats. It can incorporate new data sources, adjust risk models, and update assessment methodologies to reflect emerging risks in changing organization contexts. This adaptability ensures that risk assessments remain relevant and effective in evolving threat scenarios.
2. **Real-time analysis:** CERCA performs real-time analysis of cybersecurity events, allowing for timely detection and response to emerging threats. Traditional rule-based approaches often rely on periodic assessments, which may miss critical security incidents occurring between assessment intervals. By continuously monitoring and analyzing cybersecurity events, CERCA enables organizations to proactively identify and mitigate risks as they arise.
3. **Comprehensive risk assessment:** CERCA offers a universal approach to risk assessment by integrating data from multiple sources and employing both qualitative and quantitative assessment methodologies. This comprehensive approach provides a more accurate and subtle understanding of cybersecurity risks, allowing organizations to prioritize and address the most critical threats effectively.
4. **Decision support:** CERCA serves as a decision support tool by providing actionable insights and recommendations for risk mitigation. Traditional rule-driven approaches may lack the sophistication to offer targeted and context-specific mitigation measures. CERCA, on the other hand, leverages advanced algorithms and risk modeling techniques to generate customized mitigation strategies tailored to the organization's risk profile.
5. **Enhanced Cybersecurity awareness:** By generating management reports and visualizations of cybersecurity risks, CERCA enhances cybersecurity awareness among stakeholders, including C-level executives, security administrators, and IT personnel. This increased awareness fosters a culture of cybersecurity within the organization, promoting collaboration and alignment of security efforts across departments.

Specifically, CERCA addresses the aforementioned issues in fixed-rule scenarios by using dynamic indicator evaluation, yielding results of varied natures for interpretation (namely, qualitative and quantitative evaluations, and mitigation measures). The distinct means in the understanding of such interpretations are specifically:

1. **Dynamic assessment:** Traditional risk assessments rely on static criteria that change infrequently. This can lead to assessments that lag behind real-world changes in Cybersecurity threats and vulnerabilities. By introducing dynamic indicators with continuous monitoring of

real-time data, CERCA may adapt its results, making these evolve over time to better match up-to-date, relevant security postures.

2. **Qualitative and quantitative perspectives:** CERCA offers a comprehensive approximation to risk assessment by considering both qualitative and quantitative approximations. Qualitative assessment involves evaluating risks based on their characteristics, severity and potential impact, providing an understanding of the threat landscape. Quantitative assessment, on the other hand, involves assigning numerical values to risks, in this case monetary loss estimations, allowing for a precise risk quantification. By combining these two perspectives, CERCA enables organizations to gain a complete view of their security risks, facilitating more informed decision making.

Overall, the flexibility of the tool, its real-time analysis capabilities, comprehensive risk assessment methodologies, and decision support features make it a superior alternative to traditional risk assessment with static methodologies.

6.2.2 Implementation overview of CERCA

CERCA is roughly composed of three top-level modules:

1. **The web application:** This is a multi-tenant (supporting multiple users) user interface, compatible with any modern web browser, which allows to interact with the underlying data model and also to launch risk assessment actions. This includes the backend data engine, with a rich model covering all data categories, including the company and user data, models and indicators, reports and the assessed activities.

2. **The computation engine:** This piece drives the whole model evaluation process, as triggered from the web application or external events. Other tasks performed by this module include the initialization of indicators and the monitoring of external interfaces to receive notifications.

3. **A messaging broker:** This allows the intercommunication of the web application and the internal processing engine, transmitting notifications of actions initiated by the web application users (such as launching a risk assessment or a change in an indicator's value).

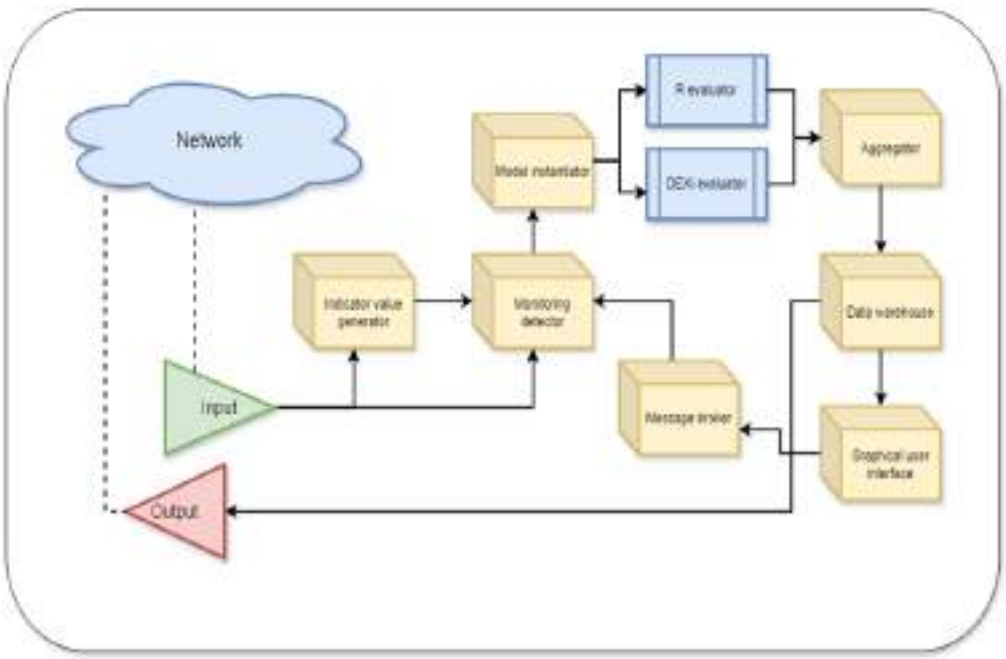

Figure 51. CERCA architecture.

Each of these modules is further divided into specialized components. Among these, the most remarkable being described in the sections below (see Figure 51).

6.2.2.1 Indicator value generator (IVG)

1. CERCA models are computed based on specific variables called indicators. These act as inputs to the models and can be of different types:

2. Questionnaire (or business) indicators: These map the answers to the questions about the details of the involved entities, which are expected to be provided by their administrators. The business questionnaire is part of the CERCA dashboard interface, and examples on these indicators may include:

    a. IN-1: Do you have application firewalls or proxies in your system?

    b. IN-33: Are there any restricted files or folders in the web server used by the web application(s)?

3. Real-time indicators: To account for information produced by external sources, real time indicators were introduced. These indicators can be set by the monitoring module, as triggered by external incident events (such as IDS), alarms (produced by SIEMs by correlating events from other sources), or vulnerability reports (obtained from the network infrastructure).

4. Consequence indicators: These store the asset monetary impact in the areas of Confidentiality, Integrity and Availability. Such information can be taken into consideration in the risk models, which are evaluated at target level (and aggregated at higher resolutions, such as report and overall levels).

The indicator value generator is responsible for initializing the indicators values at the startup of the application. It also updates the indicator values during runtime, according to updates being triggered by the event monitoring module (network alarms) or configuration changes from the graphical user dashboard (such as asset details or business profile). When selected risk models are affected by an update in an indicator value, this module also initiates the model reassessment, producing updated risk reports which better reflect the actual security landscape.

6.2.2.2 Monitoring detector

The monitoring module handles updates and changes corresponding to risk reports, which are triggered by any change in indicators values or by the selection of different risk models for evaluation. Any new situation with a potential impact on the currently existing reports is notified to this component, in order to update any existing risk assessment report.

This module, in turn, issues internal calls to the model instantiator module, which is responsible for the final evaluation of the programmatic models.

6.2.2.3 Model instantiator

This module is responsible for executing the model program files as DEXi XML files or R scripts, providing the required indicators as inputs and parsing the returned output. It does so by executing the R and Java (using a JAR library) processes in the system and capturing their output, after being called from the Monitoring detector module.

6.2.2.4 Aggregator module

The aggregator module is designed to combine individual risk assessments based on data processing activity per time, this is, organization, risk model, and risk type (i.e., confidentiality, integrity, availability). This delays the creation of a new risk report in the event of several network alarms or events received in a short period of time, and affects both the average and work case scenarios. Aggregated information is stored in the report tables in the data warehouse (see below), which will

be later presented in the tool's dashboard or optionally shared to external endpoints. Once the risk assessment data from the execution module is received, this component categorizes evaluated cyber and physical risk models into global scores for the entire infrastructure and data processing activities. Quantitative and qualitative risk scores are aggregated taking the maximum of the aggregated values, along with mitigation measures for each risk model. As mentioned, these reports are stored internally, displayed for visualization, and shared with other components as necessary.

6.2.2.5 Data warehouse component

Within CERCA, this module stands as the central data repository. This is where entity and model data are stored and accessed, serving as the backbone for the system's operations. The tool administrators may define new users, companies, their infrastructure (including related assets) through the web interface. Low-level access to the data model may be configured in the application initialization files, which define tables and relationships, including the model definitions, risks, mitigations, threats and other basic components. End-users, on the other hand, are allowed to configure the specific parameters which tailor the system to their preferences and requirements, ensuring a customized risk level assessment.

Completed risk assessment yield insights in the form of risk reports, which cover the evaluation at different target levels (per-asset, per-risk, per-model, and overall), focusing on the identified risks and recommended mitigation actions. Moreover, details from deployed sensors, as events reported by them, vulnerabilities found by scanners, and alarms triggered by the monitoring engine, are also considered to offer a comprehensive visibility into the security landscape. By serving as a centralized repository for this critical information, this module plays a pivotal role in meeting CERCA's objectives of empowering cybersecurity resilience and enabling effective risk management across various organizations.

6.2.3 The CORAS methodology

CORAS was developed as a methodology for conducting risk analysis in security environments, using a specifically designed modelling language, which extends the Unified Modelling Language (UML). Its primary objectives encompass describing the subject of assessment, facilitating interaction among diverse stakeholder groups, and documenting outcomes alongside underlying assumptions. This method delineates eight distinct steps, spanning the pivotal phases of a risk assessment endeavour, explicitly:

1. Preparation for analysis
2. Target presentation
3. Refining of target's description
4. Approval of the target
5. Risk identification
6. Risk estimation
7. Risk evaluation
8. Risk treatment

The CORAS methodology makes an intensive use of visualization diagrams, which cover assets, threats, risks, and treatments. It also acknowledges disparities between direct and indirect assets, incorporating a significance degree based on asset types. Initially, users define evaluation scales, which are subject to subsequent adjustments throughout the process. Identification and assessment of risks is strongly dependent on the initial recognition of threats, vulnerabilities, and incident scenarios. Finally, a fundamental aspect of CORAS [17] involves the creation of threat diagrams, by determining how various vulnerabilities interplay to compromise safety.

CORAS follows the standard ISO/IEC 27002:2022 [18] and it includes an open-source OS-independent utility, developed by its original authors and the open-source community. Some documentation is distributed with it as well.

6.2.4 Output

The output of CERCA is report-oriented, meaning that risk reports are considered in a data processing activity and user-aware scopes. Multiple users, entities and Data Processing Activities (DPAs) are supported, each with independent reports. DPAs allow to define application scopes for the different activities of an entity which are desired to be risk-evaluated.

The risk models are evaluated for each of the assets involved, producing a qualitative and a quantitative estimation for each of the specific risks defined by the model (namely, Confidentiality, Integrity and Availability impacts). The results of all assets involved in each risk are aggregated by taking the highest qualitative estimation and the sum of incurred quantitative assessments. Then, the aggregated values for each of the model risks are aggregated on a per-model basis in a similar way. Given that each risk report may include several models for assessment, the overall report values are computed yet again by applying a final aggregation. The aggregation levels are depicted in the formulae below:

QUALITATIVE ASSESSMENT LEVELS: Model-risk qualitative assessment: $q_{jk} = \max q_{ijk}$ for i=1…N Model qualitative assessment: $q_j = \max q_{jk}$ for k=1…S_j Overall report qualitative assessment: $q = \max q_j$ for j=1…P
QUANTITATIVE ASSESSMENT LEVELS: Model-risk qualitative assessment: $Q_{jk} = \text{sum } Q_{ijk}$ for i=1…N Model qualitative assessment: $Q_j = \text{sum } Q_{jk}$ for k=1…S_j Overall report qualitative assessment: $Q = \text{sum } Q_j$ for j=1…P
Assets: A_i for i = 1,…,N Models: M_j for j = 1,…,P Risks for M_j: R_{jk} for k = 1,…,S_j (typically S_j=3) Qualitative assessment for A_i in M_j and R_k: q_{ijk} Quantitative assessment for A_i in M_j and R_k: Q_{ijk}
Risk report Aggregated qualitative value Aggregated quantitative value Model no. 1 Aggregated qualitative value Aggregated quantitative value Risk to Confidentiality Asset no. 1 Qualitative value Quantitative value Asset no. 2 … Risk to Integrity …

```
            Risk to Confidentiality
                  …
      Model no. 2
      …
```

The diagrams above illustrate the aggregation by maximum qualitative assessments, and the sum of the incurred costs for quantitative estimates (i.e., economical losses). The illustrated output is accessible among other data through the user web interface, as a beautified, interactive report. Typically, reports are also obtained in a programmatic manner (i.e., Kafka, message brokers or HTTP APIs), for which the JSON encoding is most commonly used.

6.3 Implementation Details

6.3.1 Development Technologies

CERCA is an application composed of different independent application modules, each bearing its own internal technology upon which they are based. Roughly speaking, the applications making up CERCA can be divided into:

1. **Dashboard:** This is a Django application supporting multiple authenticated users, user roles, optionally authenticated API access and full integration of the programming model with the database model. The dashboard constitutes the graphical user interface for both administrators and end-users, and it can provide REST API access to internal data structures or processes (e.g., accessing risk reports or launching assessments).
2. **Data warehouse:** The data warehouse is the internal database of the application. This is implemented on a PostgreSQL server, but any SQL-compatible engine can be used as well. However, apart from PostgreSQL, this has only been tested on MySQL.
3. **Engine:** The main evaluation of the risk models takes place in this component, which is a Python-based application with third party library support for broker interaction and model evaluation. The main application runs in a multithreaded environment and it handles the indicators and generated report sharing with the database via communication with the dashboard. The models are evaluated using Java open-source libraries for the evaluation of DEXi and R languages in dedicated threads. The Engine is also responsible for monitoring and processing received network alarms and events originating from various sources (e.g., RabbitMQ, Kafka or HTTP REST endpoints).
4. **Broker:** Internal communication among components is handled by RabbitMQ, an open-source MQTT server which relays notifications from the Dashboard to the Engine as well as external MQTT messaging.
5. **NGINX:** This component provides SSL encryption for secure access to the dashboard GUI. While available, the component was replaced with a custom NGINX service in the current deployment of the project, as it was required to serve other applications.

As described, the Dashboard and Engine modules communicate via the RabbitMQ broker. Regarding external communications, CERCA implements input and output interfaces based on RabbitMQ (i.e., MQTT), Kafka, and web API REST. This later gateway offers a great flexibility, being able to handle both authenticated and unauthenticated requests, and it may provide access to the entire internal data model of the application, apart from its capacity to implement GUI commands as endpoint requests, as required by different components of the Project.

The deployment of the modules is carried out with Docker (see Figure 52), using a docker-compose type service catalogue, which allows setting the application configuration options through

environment variables, as well as the installation of specific packages (e.g., the statistical libraries required by the Engine, or Django frameworks for authentication or REST support). With different levels of customization, the components are evolved from baseline distributions of Python, Django, RabbitMQ, PostgreSQL, and Linux Alpine (some of which are based on the BusyBox suite, with minimal requirements).

Figure 52. Application launch with Docker.

The machine specifications for launching CERCA with Docker are: 4GB of RAM (8GB recommended), 20GB of storage (more is recommended, due to Docker's requirements for storing layers, caches, and volumes) and at least a processor with a minimum of 2 cores.

During the initialization of the Django web application, this service creates and populates the database model based on the information and relationships as described in a set of startup JSON files. These includes information about users, entities, data processing activities, assets, models, mitigations and more. Some of this information can also be accessed manually by an administrator through the web interface or the API, if required.

Once all the application services are up and running, the computation Engine initializes the indicators collection, by accessing the database under an access provided and authenticated by the Dashboard. This initialization process is rule-based, as defined in the indicator table, and it serves as a ground truth which will direct the demonstration of the tool. Typically, indicators used as flags for alarms or received events are set to 0 or “False” right after the application startup.

Part of the actions initiated by users of the web interface require the creation or update of new risk reports (for example, when modifying business questionnaire responses or the asset information, or when selecting different risk models to be considered in the report). Since the creation or update of these reports is carried out by the Engine (as the model executor module), RabbitMQ is used as the communication interface for such notifications between the Dashboard and the Engine, using different types of messages.

All accesses to the web API, the broker and the database server require authentication. Additionally, the Project assigns an nginx service for encrypting web connections between the browser and the web server, which provides standard SLL support. Although this service is external, CERCA also includes its own nginx service, which may be activated if necessary.

6.3.2 End-User workflow

The web Dashboard offers the end users the ability to access the most common functionalities provided by CERCA, as well as setting some of its most relevant adjustments.

Once the user is logged in (Figure 53), they may access the different tabs of the application on the top-level row:

Figure 53. User log in.

The Legal Entities Configuration and the Data Processing Activities Configuration tabs provide access to the company business profile (or questionnaire, Figure 54) and the asset list of the company (Figure 55). The information provided under these sections will be used to derive the required indicators for the models which will be later used.

Figure 54. Company business questionnaire.

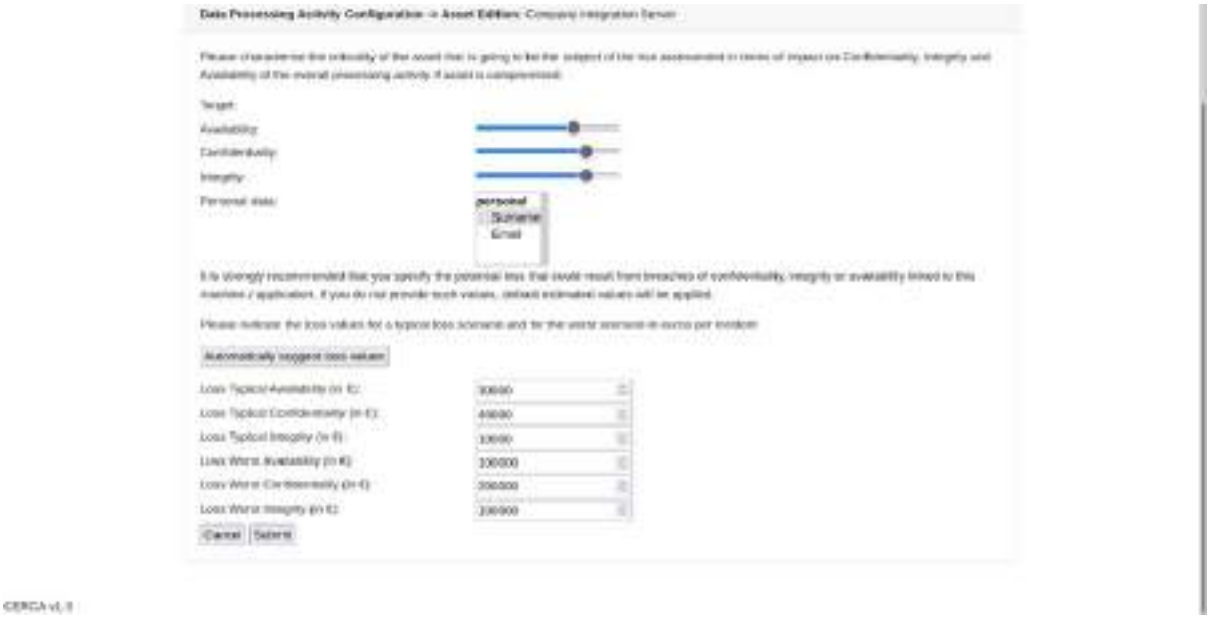

Figure 55. Asset configuration.

The risk report assessment can be launched manually by selecting the desired risk models (Figure 56 and Figure 57).

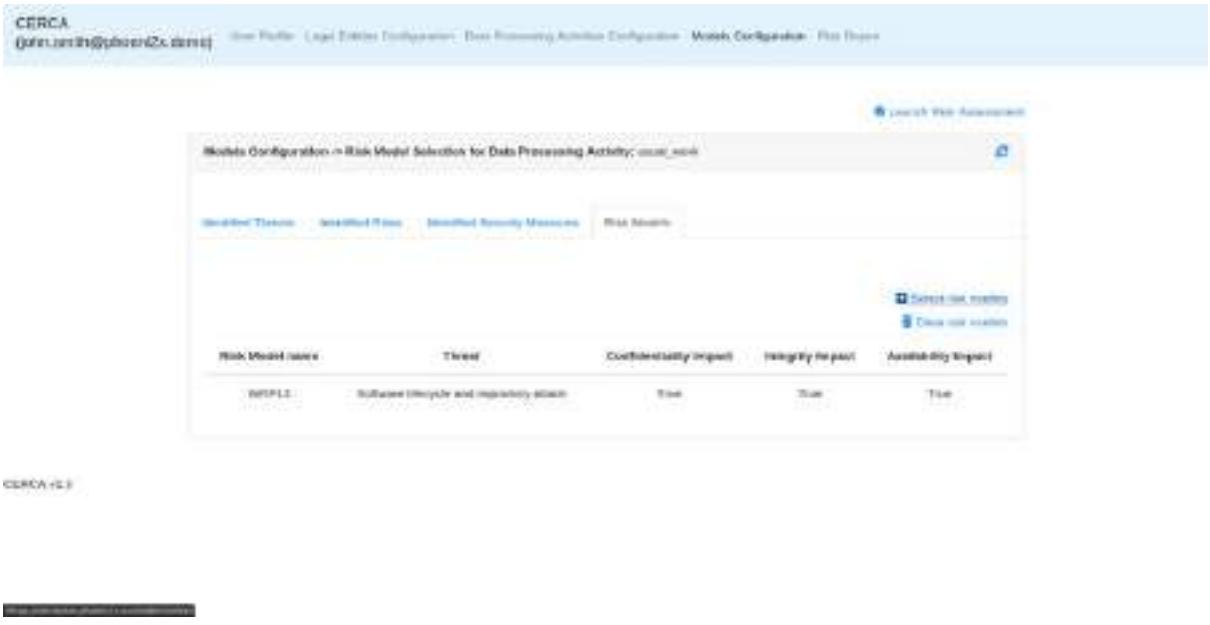

Figure 56. Select models dialog 1.

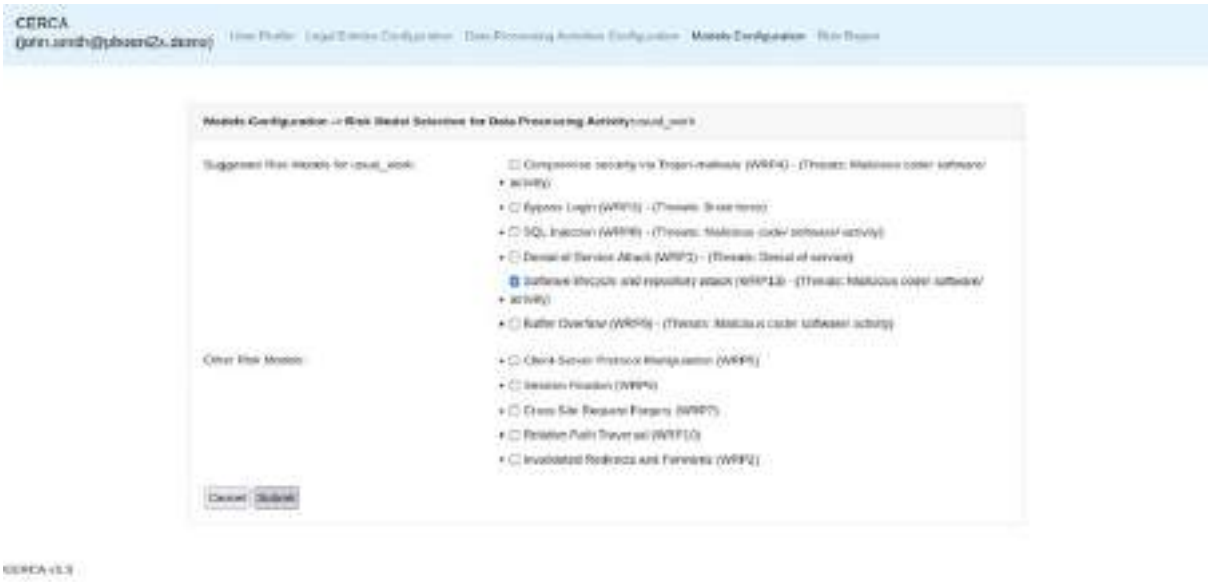

Figure 57. Select risk models dialog 2.

The computed risk report for the selected models can be accessed under the last menu entry “Risk Report”, which will load a page including both qualitative and quantitative assessments sorted as tabs (see Figure 58 and Figure 59).

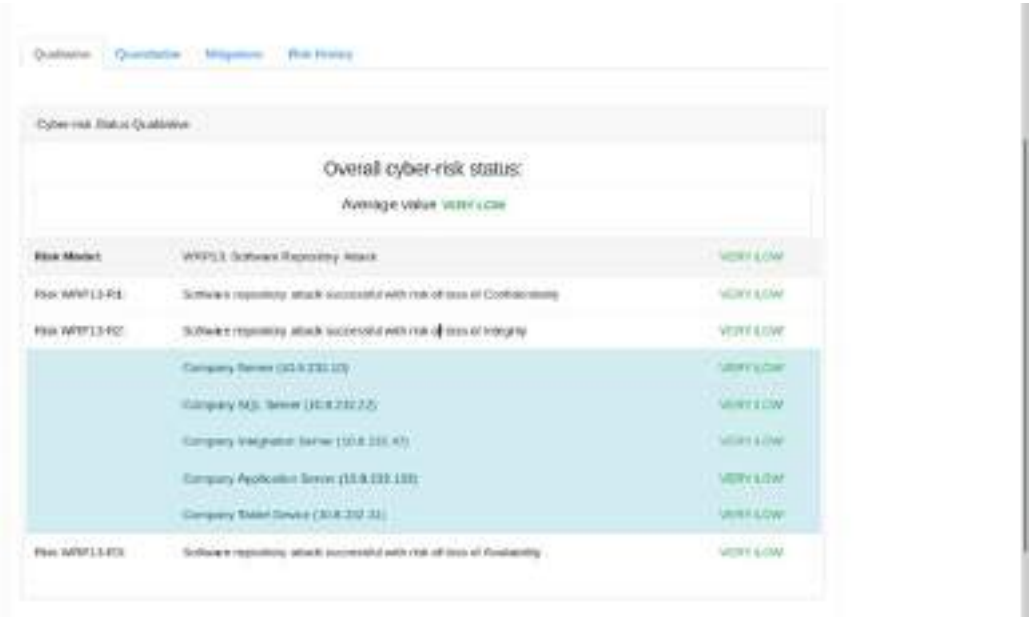

Figure 58. Qualitative risk report.

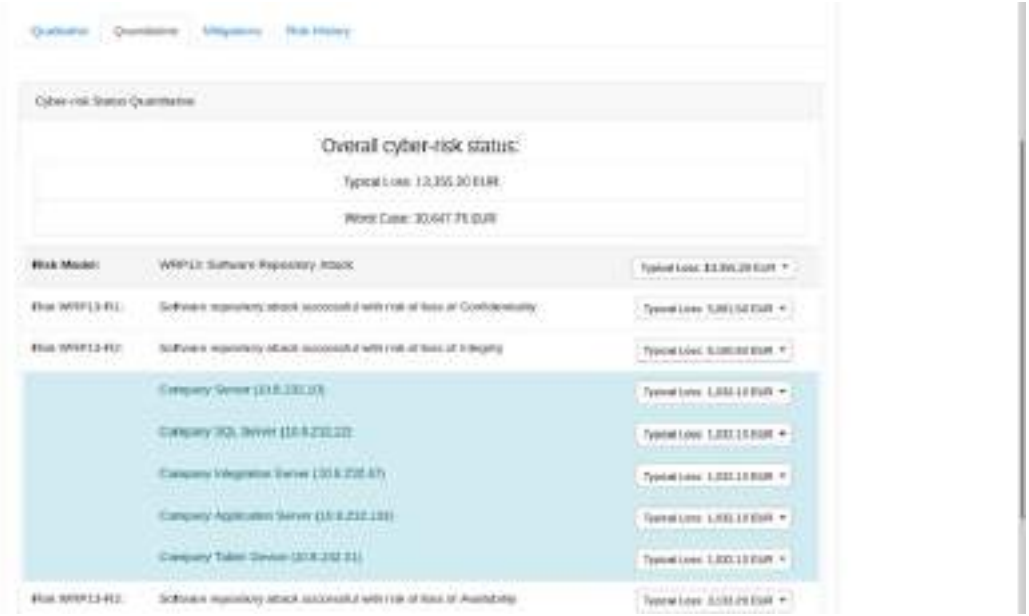

Figure 59. Quantitative risk report.

6.3.3 CERCA models

The models are programmed in the open-source languages R and DEXi, which are evaluated based on input indicators using open-source libraries and software.

For the Project's Use-Case 3, a specific model, named Software Lifecycle and Repository Attack (SLARA), has been created. This attack may target mainly source code repositories and version control systems, but also build pipelines or distribution channels. A malicious actor may attempt to inject malicious code, with vulnerable parts, or compromise the integrity of the code and its availability, affecting distribution channels as well.

The tactics required to evaluate a software repository attack stem from two major vulnerabilities, as described in the Common Weakness Enumeration system:

1. **CWE-267 (Privilege Defined with Unsafe Actions)**: This vulnerability describes situations where privilege granting may lead to undesired actions which pose a security risk. A basic example of this is using a privileged superuser account to perform regular user tasks on a software repository server, thus compromising the system and exposing it to potential breaches or threats.
2. **CWE-284 (Improper Access Control):** This weakness is related to situations in which legitimate access to a resource is not controlled or restricted appropriately. This case is given when appropriate access controls are not enforced, allowing a malicious actor to gain access to a sensitive asset. In the related use case scenario, we considered improper filtering of the repository users by the originating IP addresses, in particular.

Considering these two vulnerabilities, a selection of indicators was defined below (Table 9). Apart from these, in order to account for other possible vulnerabilities related to the use case scenario, we also considered the chance of new weaknesses affecting the project libraries being discovered in the future, for which an extra indicator was added. The complete list of indicators considered follows.

Table 9. Model indicators.

Indicator	Definition	Type
IN-A	Is it possible to establish SSH connections with the root user?	Business
IN-B	Was there a new vulnerability discovered in any library used?	Real-Time (Vulnerability)
IN-C	A connection was initiated from an IP address no present in the allowed list	Real-Time (Logs)
IN-D	A push to the codebase was performed by an IP not present in the allowed list	Real-Time (Logs)
IN-E	A code branch was deleted from an IP not present in the allowed list	Real-Time (Logs)
IN-F	An SSH connection with root privileges was established	Real-Time (Network)

These indicators map a variety of attack scenarios which will be the central part during the model evaluation phase, when their impacts on confidentiality, integrity and availability will be assessed.

The indicators IN-C to IN-E (access from unexpected addresses) account for improper access control conditions, while IN-A and IN-F (access from root account) model those behaviors associated with unsafe actions from largely privileged accounts (as a couple of examples). Finally, IN-B (new vulnerability found in library) covers any range of weaknesses imported from the required software libraries, which may not be available at the initial assessment. We believe that this selection of indicators constitutes a feature-rich environment to comply with the Use Case 3 requirements.

Regarding the nature (or type) of the indicators considered, IN-A was set up as a business (or questionnaire) variable, meaning the value of this indicator is defined by the answer provided by the administrator (within the web Dashboard) to the question about SSH connections being allowed from root accounts. This indeed serves as a prerequisite (or preconditioning) to the network alarm triggering indicator IN-F (SSH connection with root privileges established).

Figure 60. Question related to business indicator.

While IN-A is derived from the selection in the Dashboard questionnaire (Figure 60), and IN-C up to IN-F are flagged by explicit network alarms, the indicator IN-B is derived from cyber-threat intelligence reports issued by TII. This constitutes a groundbreaking achievement which will be revisited in detail later (see next section). Any change in these indicators will have an impact on the confidentiality, integrity or availability of the asset, and will therefore trigger a new report reassessment.

A CORAS model was defined for this use case comprising all the considered indicators, vulnerabilities, threat scenarios and unwanted outcomes, as illustrated below (Figure 61).

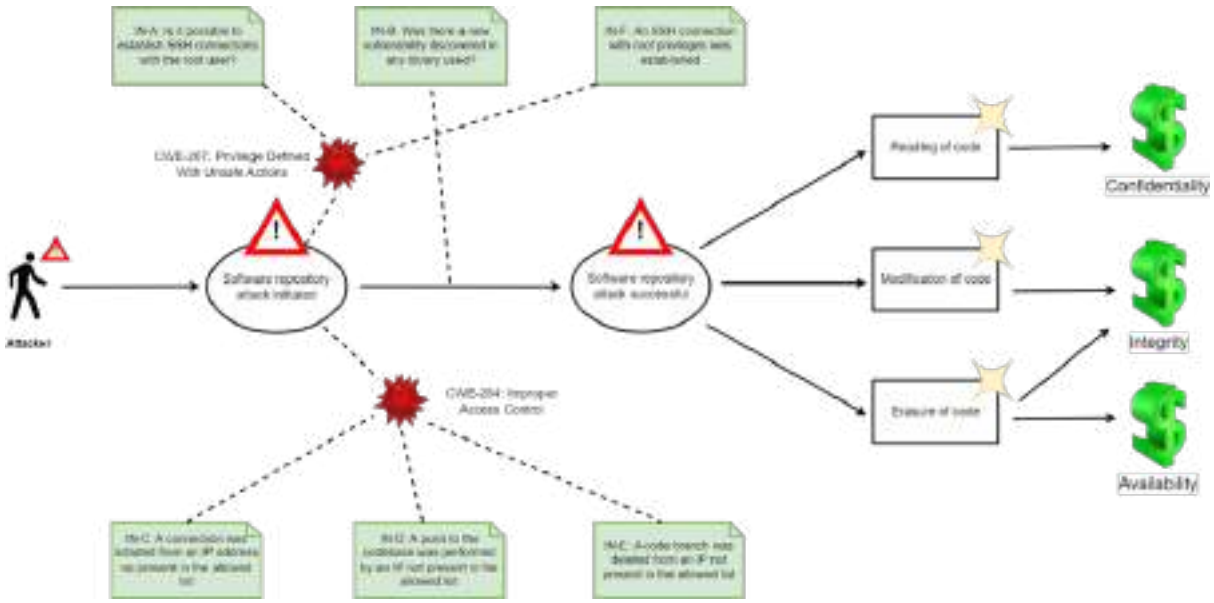

Figure 61. CORAS model diagram.

This diagram shows a threat scenario where the attacker may take different actions to reach the unwanted incident situations at the right end (e.g., reading, modifying, or erasing the repository code). The two vulnerabilities in the picture (marked with spiked red balls) represent weak points which may be exploited by the attacker to gain access to the underlying system.

Indicators (encoded in green-background paper notes) exhibit hypothesis on the belief network which will impact the estimated consequences of the threat scenario. Indicators are fed as inputs to the risk models (R and DEXi), which will in turn evaluate the impact of a successful end state under a given probability (i.e., carrying out the attack) for each of the associated risks (i.e., confidentially, integrity and availability).

The consequence relation can be thought of as the effect of the undesired incidents on each of the entity assets, where an asset is any physical or logical resource of the entity (typically, servers, devices, or information). The threat scenarios (white balloons) exhibit a likelihood which is dependent on the indicators' values, while the unwanted incidents (white rectangles) as computed as marginal probabilities of the previous scenarios leading to each of them.

The CORAS model was implemented using the R and DEXi technologies to produce quantitative and qualitative model assessments.

6.3.4 Integration with Third Parties

In CERCA, a new messaging module has been developed for the Project using Kafka, which allows for sending and receiving distributed notifications by subscribing to different topics on a Kafka broker. This facilitates integration with other components, relaying notifications almost instantly between applications. An example is FVT, the visualization platform of PHOENI2X, where the general results of CERCA will be displayed. The FVT will also redirect to the CERCA dashboard to visualize the results with a higher degree of detail, for example, to visualize the risk by asset and model. The messages are encoded in JSON format prior to dispatching or upon reception from the broker.

The integration has been performed at the Engine level, which previously was only able to process simple network alarms originating from MQTT or web interfaces. For this purpose, the Python package

Kafka was chosen, as it allows for publishing and consuming messages on broker in a straightforward manner.

Given that CERCA also produces information at risk reporting level, the use of Kafka ensures this information becomes readily available for consumption by third parties, providing analysts with risk information they can leverage (see Figure 62).

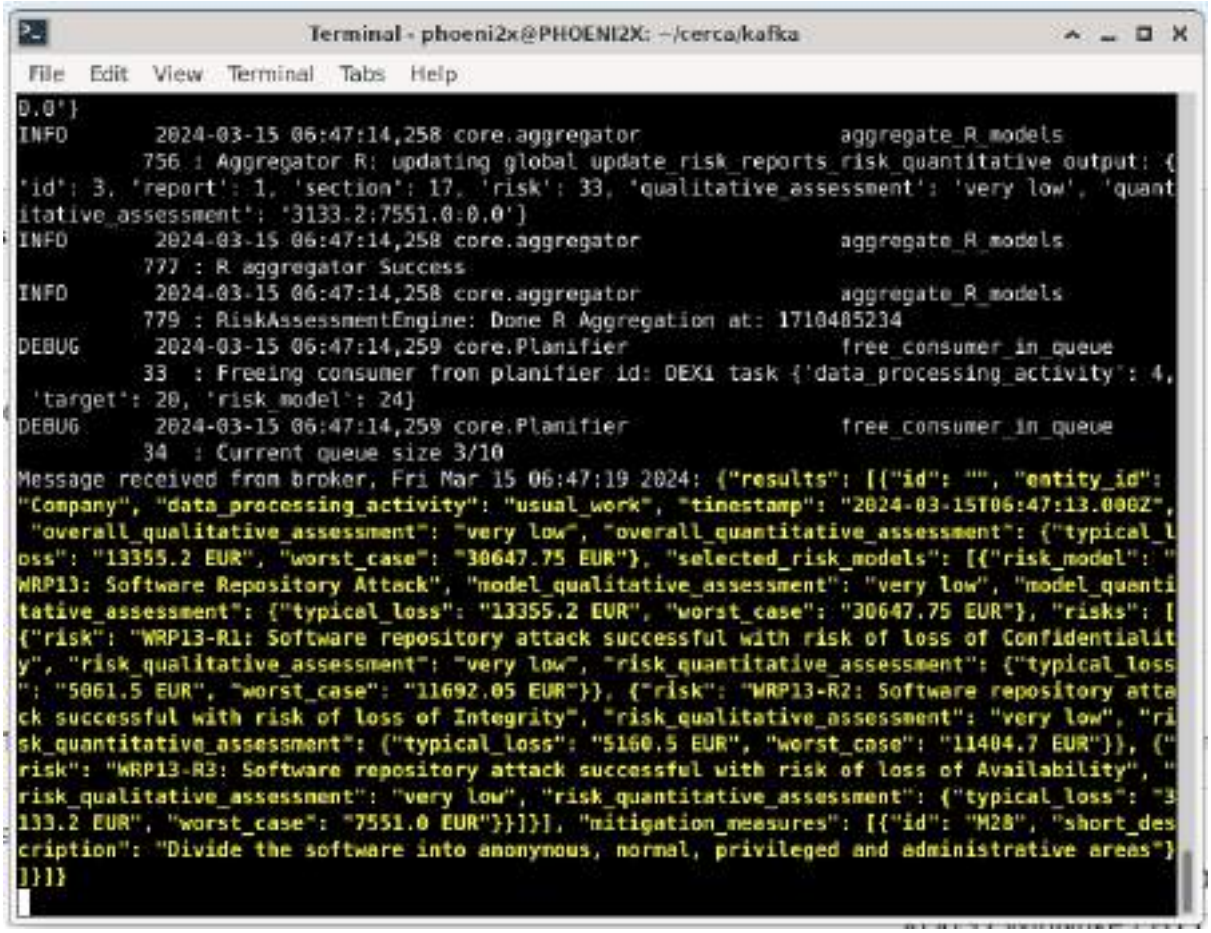

Figure 62. Example of CERCA risk report delivered to Kafka (in yellow).

The FVT will act as the common entry point from multiple components providing to the end user a holistic and intuitive overview of the status of the system along with sensitive notifications. The home dashboard is customized for each use case so that intuitive visualizations present the latest events. Additionally, an integrated alerting mechanism is monitoring the Apache Kafka topics for sensitive events that will be displayed as toast notifications in the interface, as can be shown in the Figure 63 bellow. Following the alerts or browsing the internal dashboards, the user can investigate and draw conclusions over system events with the use of more advanced visualizations, timelines and filtering capabilities.

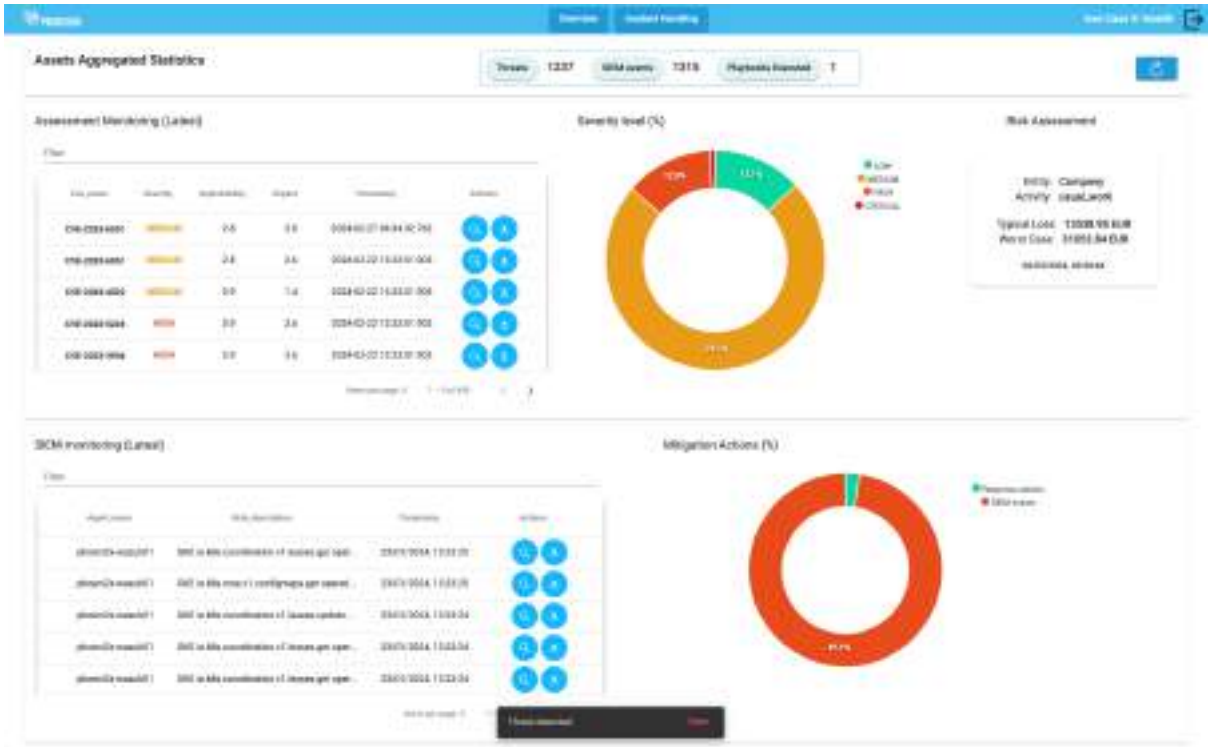

Figure 63. FVT dashboard.

Regarding the integration with TII, the digested information originating from different sources may be consumed by CERCA, which, by incorporating this CTI effectively, will enlighten a better understanding of the nature and severity of cyber threats, proactively identifying potential risks and responding to security incidents promptly. Another integration that Kafka has made possible is with Wazuh, the alarms generated by Wazuh are directed via Logstash and Kafka to CERCA, which will use those alarms as indicators to be considered in the risk assessment.

7 Conclusions & Next Steps

This deliverable has presented the overview, design and implementation details of the first version of the AI-assisted Situational Awareness, Prediction & Response enablers of the PHOENI2X framework. These include the User & Entity Behaviour Analytics enabler, the CTI Discovery, Analytics & Threat Hunting enablers (CTI Discovery & Analytics and TII), the Attack Prediction, Response Recommendation & Adaptation enabler and the Risk Impact Assessment, Alert Triage & Response Prioritisation enabler.

The AI-assisted Situational Awareness, Prediction & Response enablers have been integrated in the MVP PHOENI2X framework, in the context of WP5 activities, and have been applied to the three use cases of the project (energy, transport and health).

Further developments of the AI-assisted Situational Awareness, Prediction & Response enablers to complete the planned functionalities per enabler, jointly with the feedback obtained from the initial integration will leverage to the second and final version of the WP3 enablers on M35. The outcome of this work will be documented in D3.2 (AI-assisted Situational Awareness, Prediction & Response Enablers v2), which will provide the overview, design and implementation details of the second version of the AI-assisted Situational Awareness, Prediction & Response enablers.

References

[1] K. Parsons, A. McCormac, M. Butavicius, M. Pattinson und C. Jerram, "Determining employeeawareness using the Human Aspects of Information Security Questionnaire (HAIS-Q)," Computer & Security, Bd. 42, pp. 165-176, 2014.

[2] M. Karjalainen, "Improving employees' information systems (IS)security behaviour: toward a metatheory of is security training and a new framework for understanding employees' is security behaviour," University of Oulu, Oulu, Finland, 2011.

[3] S. Egelman und E. Peer, "Scaling the security wall: Developing a security behaviour intention scale (SeBIS)," in CHI, Seoul, Republic of Korea, 18-23 April 2015.

[4] R. G. Netemeyer, W. O. Bearden und S. Sharma, Scaling Procedures: Issues and Applications, SAGE Publications Inc., 2003.

[5] C. Faklaris, L. D. und J. I. Hong, "A self-report measure of end-user security attitudes (SA-6)," in USENIX Symposium on Usable Privacy and Security (SOUPS), Santa Clara, CA, USA, August 11 - 13, 2019.

[6] P. Rajivan, P. Moriano, T. Kelley und L. J. Camp, "Factors in an end user security expertise instrument," Information & Computer Security, Bd. 25, Nr. 2, pp. 190-205, 2017.

[7] L. Hadlington, "Human factors in cybersecurity; examining the link between Internet addiction, impulsivity, attitudes towards cybersecurity, and risky cybersecurity behaviours," Heliyon, Bd. 3, Nr. 7, Jlu 2017.

[8] C. G. Coutlee, C. S. Politzer, R. H. Hoyle und S. A. Huettel, "An Abbreviated Impulsiveness Scale (ABIS) Constructed through Confirmatory Factor Analysis of the BIS-11," Arch Sci Psychol, Bd. 2, Nr. 1, p. 1–12., April 2014.

[9] R. A. Davis, G. L. Flett und A. Besser, "Validation of a new scale for measuring problematic internet use: implications for pre-employment screening," Cyberpsychol Behav, Bd. 4, Nr. 5, pp. 331-345, August 2002.

[10] G. Öğ˘ütçü, Ö. M. Testik und O. Chouseinoglou, "Analysis of personal information security behavior and awareness," Computer & Security, Bd. 56, pp. 83-93, 2016.

[11] H.-Y. Huang, S. Demetriou, R. Banerjee, G. S. Tuncay, C. A. Gunter und M. Bashir, "Smartphone Security Behavioral Scale: A New Psychometric Measurement for Smartphone Security," in https://arxiv.org/abs/2007.01721, 2020.

[12] AEuropean Commission, "Cyber security cOmpeteNCe fOr Research anD InnovAtion," 1 January 2019. [Online]. Available: https://cordis.europa.eu/project/id/830927.

[13] European Commission, "Strategies and Technologies for United and Resilient Critical Infrastructures and Vital Services in Pandemic-Stricken Europe," 1 October 2022. [Online]. Available: https://cordis.europa.eu/project/id/101073821.

[14] European Commission, "rEsilient and seLf-healed EleCTRical pOwer Nanogrid," 1 October 2023. [Online]. Available: https://cordis.europa.eu/project/id/101021936.

[15] Casey, T. (2007). Threat agent library helps identify information security risks. Intel White Paper, 2.

[16] Ngcobo, T. J., & Ghayoor, F. (2022). An overview of DLMS/COSEM and g3-plc for smart metering applications. *International Journal on Smart Sensing and Intelligent Systems*, *15*(1).

[17] Burenok, D. S., & Voevodin, V. A. (2023). On the information security controls database developed in accordance with ISO/IEC 27002: 2022. International Journal of Open Information Technologies, 11(9), 118-124.

[18] ISO/IEC 27002:2022, Information security, cybersecurity and privacy protection. Information security controls.