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

D5.1: PHOENI2X framework - MVP

Abstract: This document reports on the work carried out within the context of PHOENI2X's Work Package 5 (WP5) and more specifically Task T5.2, which aims to deliver the PHOENI2X platform in two major iterations; the MVP version of the framework on M18 (presented in this deliverable) and the second version of the framework on M32 with the final improvements leading to the final one on M35 (being deliverable D5.3). The main goal of this document is to accompany the first (MVP) version of the integrated PHOENI2X framework, which includes the baseline components from WP2, and all available components from WP3 and WP4, released on M16). The outcome of this process will be fed back to Work Packages 3 and 4 for the next iteration of implementation and delivery of the PHOENI2X components towards the V1 PHOENI2X framework release and its evaluation.

Contractual Date of Delivery	31/12/2023
Actual Date of Delivery	31/12/2023
Deliverable Security Class	PU
Editor	Vassiliki Andronikou (NPS)
Contributors	UPAT, SANL, UPC, WSE, AEGIS, SEA, ATOS, UiO, PPC, NCSA
Quality Assurance	WSE, AEGIS

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: #7 WORLDSENSING SL (WSE)

Reviewer #2: #8 AEGIS IT RESEARCH GMBH (AEGIS)

Revisions

Version	Date	By	Overview
1	23/10/2023	Editor	ToC released
2	30/10/2023	AEGIS, PPC, WSE, NPS	Input to section 2 (2.1, 2.2)
3	03/11/2023	Editor	Input to sections 3.1, 3.3
4	05/11/2023	Editor	Input to sections 3.2, 3.4
5	10/11/2023	SANL, NPS, AEGIS, ATOS, UPC, PPC, WSE	Input to section 3.4
6	17/11/2023	AEGIS, PPC, WSE, NPS	Updated input to section 2 (2.1, 2.2)
7	21/11/2023	UPAT	Input to section 2.3
8	24/11/2023	SANL, NPS, AEGIS, ATOS, UPC, PPC, WSE	Updated input to section 3.4
9	01/12/2023	Editor	Input to sections 1 and 5
10	11/12/2023	Editor	Overall improvements
11	14/12/2023	SANL, NPS, AEGIS, ATOS, UPC, PPC, WSE	Input to section 4
12	18/12/2023	AEGIS, WSE	Review
13	29/12/2023	Editor	Final version for submission

Table of Contents

List of Tables 1
List of Figures 2
List of Abbreviations 3
1 Introduction 5
1.1 Purpose of the Document 5
1.2 Overall Methodology 5
1.3 Relationship with other PHOENI2X deliverables 5
2 The PHOENI2X Integrated Framework (1st release - MVP) 7
2.1 The PHOENI2X Framework 7
2.2 Components' Interactions 10
2.2.1 Use Case 1 – Energy 10
2.2.2 Use Case 2 – Transport 12
2.2.3 Use Case 3 – Healthcare 13
2.3 PHOENI2X Framework Deployment 18
3 The PHOENI2X Integrated Methodology and Plan 21
3.1 PHOENI2X Integration Plan 21
3.2 Integration Methodology 22
3.3 Integration Management 28
3.4 Components' Interfaces 29
4 PHOENI2X Framework Testing 35
5 Conclusion and Next Steps 47
Annex – Examples of components' interactions 48

List of Tables

Table 1 PHOENI2X VMs..... 18
Table 2 PHOENI2X Integration Plan..... 21
Table 3 PHOENI2X Framework Integration Matrix..... 24
Table 4 PHOENI2X and UC1 Integration Matrix..... 25
Table 5 PHOENI2X and UC2 Integration Matrix..... 26
Table 6 PHOENI2X and UC3 Integration Matrix..... 27
Table 7 PHOENI2X Components Integration Endpoints (incl. UCs)..... 30
Table 8 PHOENI2X framework interactions testing..... 36

List of Figures

Figure 1: Detailed PHOENI2X architecture 9
Figure 2: UML diagram of UC1 11
Figure 3: UML Diagram of UC2 13
Figure 4: UC3 pre-attack scenarios sequence diagrams 16
Figure 5: UC3 attack scenarios sequence diagrams 17
Figure 6: The UPAT resources for PHOENI2X 18
Figure 7: PHOENI2X Deployment diagram 20
Figure 8 Test for getting data between NDR-SmartMeter 48
Figure 9 Sample of data received by PMEM from CMT_CLoud 48
Figure 10 Sample of data received by PMEM from NDR 48

List of Abbreviations

AMI: Advanced Metering Infrastructure

ARP: Address Resolution Protocol

AI: Artificial Intelligence

CACAO: Collaborative Automated Course of Action Operations

CERCA: Cyber Risk Assessment Calculator

CI: Critical Infrastructures

CMT: Connectivity Management Tool

CR: Cyber Range

CTI: Cyber Threat Intelligence/Information

DLMS: Device Language Message Specification

DDoS: Distributed Denial of Service

DoA: Description of Action

DSM: Design Structure Matrix

DX.X: Deliverable X.X

EDR: Endpoint Detection and Response

EU: European Union

FVT: Forensics Visualization Toolkit

GUI: Graphical User Interface

IoT: Internet of Things

IR: Incident Response

IRP: Incident Response Platform

MISP: Malware Information Sharing Platform

MiTM: Man in The Middle

MVP: Minimum Viable Product

NDR: Network Detection and Response

OES: Operators of Essential Services

OpenVAS: Open Vulnerability Assessment Scanner

OT: Operational Technology

PMEM: Predictive Maintenance

RCR: Resilience Cyber Range

ROAR: Resilience Automation, Orchestration, and Response

SDN: Software-Defined Networking

SIEM: Security Information & Event Management System

SMIR: Smart Mandatory Incident Reporting

SOAR: Security Orchestration, Automation and Response

SPA: Security & Privacy Assurance

STIX: Structured Threat Information Expression

TAC: Threat Actor Context

TII: Threat Intelligence Integrator (TII)

TIP: Threat Intelligence Platform

UC: Use Case

UEBA: User and Entity Behavior Analytics

VM: Virtual Machine

WP: Work Package

1 Introduction

1.1 Purpose of the Document

The purpose of this document is to accompany the first release (MVP) of the PHOENI2X platform, being the outcome of Task 5.2 “Framework Integration & Testing”. This report initially presents the overall PHOENI2X architecture based on which the integration activities have been set up in section 2. This section also describes the interactions of each use case both in terms of their internal components as well as with the PHOENI2X framework, towards the realisation of their scenarios, as the latter have been presented in D2.1 “PHOENI2X Requirements & Architecture”.

In section 3, the integration plan is presented, which provides an overview as well as the individual steps of the integration process both for the MVP as well as the intermediate and the final release of the integrated PHOENI2X framework. For each step, the PHOENI2X components involved, the partners responsible as well as the timeline are documented. This section also describes the integration methodology through the specification of the integration matrices which include the directional interactions between the PHOENI2X components as well as the use cases components. Each interaction is linked with an identifier which is used across the document as reference. Furthermore, it presents the tools used for monitoring the integration activities in the project. Finally, the identified interactions are documented in terms of their data types and protocols, and implementation status.

In section 4, the testing process of the PHOENI2X framework is described, while for each interaction identified in section 3, the respective test, its result and current status are listed. Finally, section 5 concludes the deliverable’s work and briefly presents the next steps to be followed towards the next releases of the PHOENI2X framework with a clear view towards the final one.

1.2 Overall Methodology

Initially, the PHOENI2X architecture was analysed and the diagrams showing the interactions between the different components were developed. Each diagram aimed at encapsulating all the interactions necessary for the realisation of the three (3) use cases’ scenarios, as presented in D2.1 “PHOENI2X Requirements & Architecture”. Based on these diagrams, the interactions were documented in a series of tables which resulted in the integration matrices towards the MVP integrated PHOENI2X framework. These matrices were used throughout the integration process as a means of monitoring progress. Regular teleconferences were held with the PHOENI2X component owners and the UC partners in order to clarify open issues and / or further specify the interactions being implemented towards the development of the MVP integrated PHOENI2X framework (M18) and its interlinking with the use cases’ infrastructures. The interactions between the different components were tested to ensure proper operation of the integrated framework. As the integration process is driven by the PHOENI2X architecture, the latter will be amended, if necessary, to be consistent with the actual implementation of the components and their interactions. The MVP integrated framework will serve as a basis for its next releases through a continuous integration process. Hence, developments of the WP3 and WP4 will be incorporated throughout the project’s lifetime, leading to a second release on M24, a pre-final one on M32 and the final PHOENI2X integrated framework on PM35, encapsulating all refinements of the PHOENI2X components.

1.3 Relationship with other PHOENI2X deliverables

This deliverable is based on D2.1 “PHOENI2X Requirements & Architecture” (officially submitted on M10), which has provided the specifications and the architecture of the PHOENI2X framework and its three (3) use cases, as well as the implementation of the first version of the PHOENI2X components in WP3 and WP4, which are to be documented in D3.1 “AI-assisted Situational Awareness, Prediction & Response Enablers v1” and D4.1 “Coordinated Response & Preparedness Enablers v1” (M21).

Moreover, this deliverable is highly interlinked with the activities in Task 5.1 “Testbed Setup, Evaluation Methodology & Baseline Establishment” which involves the setting up of the testbeds for the PHOENI2X use cases as well as the specification of the PHOENI2X framework evaluation process and the definition of the criteria to be applied for the latter. As this report accompanies the released PHOENI2X framework, together they will feed the activities in Task 5.3 “Use Cases Demonstration & Validation”. The second and final iteration of the project’s developments will rely on the current deliverable (D5.1) and the resulting feedback from the validation process.

2 The PHOENI2X Integrated Framework (1st release - MVP)

2.1 The PHOENI2X Framework

This section provides an overview of the building blocks constituting the PHOENI2X integrated framework, as outlined in deliverable "D2.1 - PHOENI2X Requirements & Architecture" and specified in deliverable "D2.2 - Baseline Enablers".

1. **Baseline Prevention, Detection & Response toolset building blocks**
    a) OpenVAS vulnerability scanner (Dynamic testing)
    b) Vulnerability scanning component of SPA Suite (Static testing)
    c) Wazuh (Security Information and Event Management - SIEM)
    d) MISP (Cyber Threat Intelligence - CTI)
    e) Wazuh (Endpoint Detection and Response - EDR)
    f) pfSense, Zeek, CICflowMEter and Wireshark (Network Detection and Response - NDR)
    g) Conpot (Deception Tool)
    h) Forensics Visualization Toolkit - FVT (Digital Investigations and Forensics)
    i) Apache Kafka (Message Bus)
2. **AI-assisted Situational Awareness, Prediction & Response building blocks**
    a) SPHYNX Analytics Platform - AP (User & Entity Behaviour Analytics)
    b) Cyber Risk Assessment Calculator - CERCA (Risk Impact Assessment & Prioritisation)
    c) Predictive Maintenance Tool - PMEM (Risk Impact Assessment & Prioritisation)
    d) Threat Intelligence Integrator - TII (CTI Discovery, Analytics & Threat Hunting)
3. **Preparedness building blocks**
    a) SPHYNX Cyber Range - CR (Cyber Range)
    b) HATCH - Physical Serious Games (Social Engineering Awareness Education)
    c) PROTECT - Digital Serious Games (AI-Enhanced Security Education)
4. **Resilience Orchestration, Automation & Response (ROAR) building blocks**
    a) Incident Response Platform - IRP (Incident Response Manager)
    b) Security Playbook Extension for STIX2.1 (Resilience Playbooks)
    c) Security Playbook Object Template for MISP Threat Information Sharing Platform (Resilience Playbooks)
    d) CACAO Specification and JSON Schemas for Validation (Resilience Playbooks)
    e) TAC ontology (Resilience Playbooks)
5. **Security Assurance & Certification building blocks**
    a) SPHYNX Security & Privacy Assurance Platform - SPA (Security & Privacy Assurance)
6. **Alerting, Reporting & Information Exchange building blocks**
    a) Smart Mandatory Incident Reporting - SMIR (Incident Management & Reporting)
    b) CTI Mediator (Information Sharing)

In D2.1, the detailed, implementation-level architecture of PHOENI2X has been presented, as derived based on the high-level, conceptual architecture and the individual building blocks mentioned above. In D2.2, the baseline enablers (point 1 in the list above) have been further specified, based on the desk research performed by the responsible partners.

As is obvious, all core components are depicted and interconnected, to the best of the current knowledge and agreement between involved partners. A short text description can also be found on all key envisioned interfaces, aiming to provide a preliminary view of the main interactions expected. Interactions with external solutions (platforms, technologies, data sources, etc.) are also depicted in an abstracted manner (Figure 1). The consortium maintains a “live” version of this figure, updating it as the development of PHOENI2X matures, towards the first release of the framework. Each integrated release of PHOENI2X will come with the latest version of this architectural diagram.

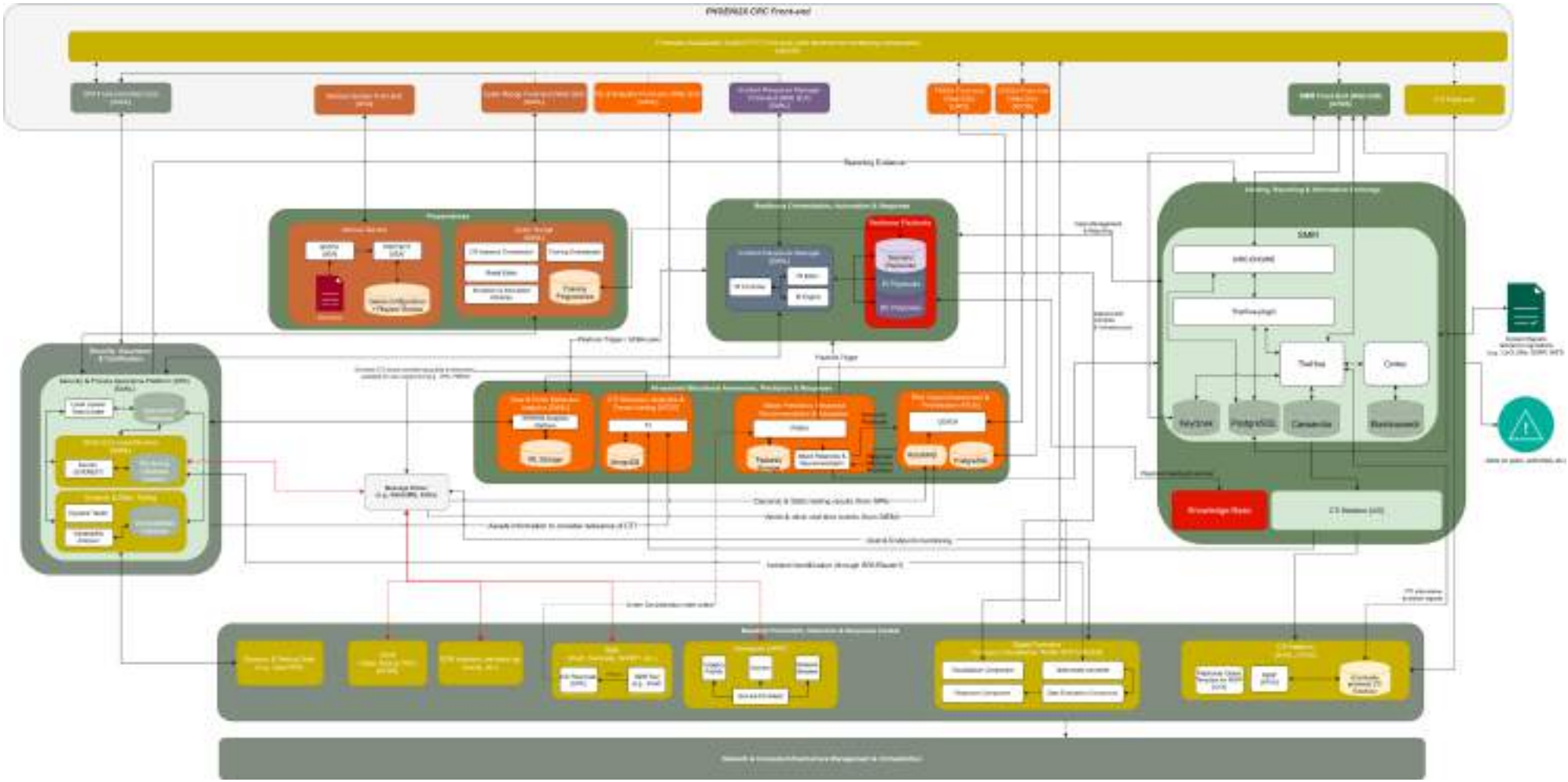

Figure 1: Detailed PHOENI2X architecture

2.2 Components' Interactions

2.2.1 Use Case 1 – Energy

Use Case 1 aims to demonstrate the PHOENI2X framework on an attack and response scenario that takes place on an energy infrastructure, replicating an Advanced Metering Infrastructure (AMI). More specifically, the Use Case scenario breaks down into the following sub-scenarios:

- Sub-scenario 1: Prevention and Preparedness.
- Sub-scenario 2: During the Attack Phase.
- Sub-scenario 3: Post Attack Phase.

Each of these sub-scenarios is thoroughly analysed in subsequent sections. Additionally, Figure 2 presents a UML diagram of the Use Case, encapsulating all the sub-scenarios for a comprehensive overview.

2.2.1.1 Sub-scenario 1: Prevention and Preparedness

During this stage, the monitoring and a periodic security assessment of an Advanced Metering Infrastructure (AMI) is performed. The infrastructure under monitoring consists of a DLMS-based smart meter and the AMI headend, which is a server that periodically collects measurements of power consumption from the smart meter. The network activity of this infrastructure (mainly, the DLMS messages) is monitored by an NDR tool, which passes the relevant data to PMEM for advanced analytics and the prediction of potential threats. At the same time, host-based logs and data are also collected by the EDR tool (Wazuh Agent), which are also passed to PMEM for the respective predictive analytics. Security events from both NDR and EDR are collected by the SIEM (Wazuh), which in turn provides evidence about the detected threats to SPA. FVT also collects the said evidence from the SIEM, along with measurements from the AMI headend and potential input (e.g., historical data) from the operator, for visualisation. Cyber-threat information, relevant to the AMI infrastructure, is periodically collected by CTI sources and aggregated by TII. SPA periodically triggers vulnerability assessment of the AMI system through the available baseline tools.

Finally, to augment the prevention and preparedness activities of this sub-scenario, serious games are employed by PROTECT, in order to train human operators to withstand social engineering attacks. First, a trainer configures the game settings to suit the Energy scenario. The module delivers AI-customized content to the operator, depending on the operator's profile and past interactions, including personalized content, tailored training modules and learning material. The operator in turn, by playing the game, generates data for PROTECT to analyse and make further adaptations to the training content. Feedback can also be provided to PROTECT, to improve performance and content quality. Finally, PROTECT generates two reports, one for the trainer to assess the operator's progress and one for the operator with the results of the specific operations and tasks.

2.2.1.2 Sub-scenario 2: During the Attack Phase

During the attack phase, the malicious actor performs a Man in The Middle (MiTM) and/or a Distributed Denial of Service (DDoS) attack. In the case of MiTM, the attacker performs ARP poisoning to intercept the communication between the smart meter and the AMI headend, and then alters the measurements transmitted from the smart meter to the AMI headend through the DLMS communication protocol. In the case of the DDoS attack, the attacker launches multiple botnets that flood the AMI headend, aiming to exhaust the available resources and prevent access to legitimate users (e.g., the AMI headend operators).

Based on the logs and traces generated by the attack, PMEM detects the anomaly and triggers a response playbook through the ROAR component. Alternatively, in case the attack is not detected by

PMEM, the operator is able to inspect relevant historical data through FVT, detect a possible anomaly, and trigger the respective playbook manually through ROAR. The playbook trigger is logged into FVT, while a new incident is created by ROAR and submitted to SMIR.

Based on the attack characteristics, ROAR may invoke one of the following response playbooks, both aiming to mitigate the attack:

- ROAR interacts with the REST API of the firewall (pfSense) to insert the appropriate firewall rules in order to prevent the attacker(s) from communicating with the victim;
- ROAR interacts with the SDN Controller to isolate the attacker, by inserting the appropriate OpenFlow rules to the intermediate SDN switch.

Finally, the ROAR component sends the playbook, along with a notification of the incident, to FVT for logging. The operator is also informed accordingly by ROAR about the isolation.

2.2.1.3 Sub-scenario 3: Post-Attack Phase

In the post attack phase, the human operator, first, consults the FVT tool in order to perform a root cause analysis of the incident. Next, the operator interacts with SMIR to generate an incident report. The report is inspected by the operator, who decides to finally submit it to the relevant national authority.

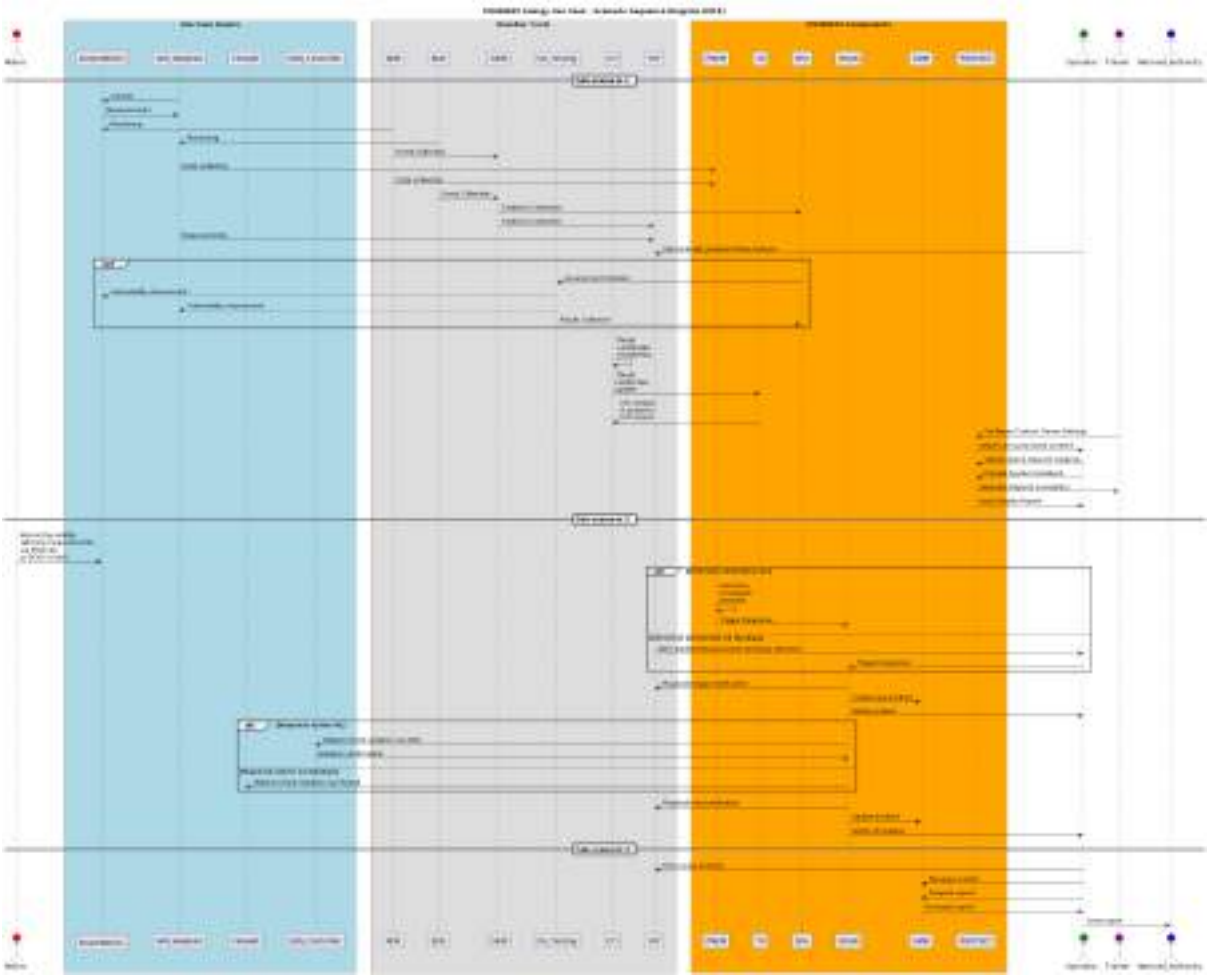

Figure 2: UML diagram of UC1

2.2.2 Use Case 2 – Transport

Use Case 2 aims to demonstrate the PHOENI2X framework on an attack and response scenario that takes place on a transport infrastructure on the OT system for safety monitoring of a railway section. More specifically, the Use Case scenario breaks down into the following sub-scenarios:

- Sub-scenario 1: Prevention and Preparedness.
- Sub-scenario 2: During the Attack Phase.
- Sub-scenario 3: Post Attack Phase.

The sub-scenarios are analysed in the following sections, while Figure 2Figure 3 depicts the UML diagram of the Use Case, which summarises all sub-scenarios.

2.2.2.1 Sub-scenario 1: Prevention and Preparedness

The objective of sub-scenario 1 is to define the baseline activity for the OT system, including edge devices and cloud services. Other actions to be performed at this sub-scenario are the initial cyber-hygiene analysis and the definition of the playbooks.

The baseline activity definition is based on the data collected from the IoT edge devices, and the metadata from the edge devices and the cloud services. Both data and meta-data streams are continuously being sent to PMEM to train the tool in the establishment of the normal activity pattern. The data stream is managed by the Connectivity Management Tool (CMT) as the cloud system that receives all data from IoT Edge devices, while the metadata stream is collected by the NDR component for the UC collecting metadata from radio connectivity and also resource and network usage of the CMT cloud system.

A general cybersecurity analysis is also part of the sub-scenario, offering an initial vulnerability assessment of the cloud IT & OT solution that collects all data from edge devices, and monitors the network status of them. The analysis is carried out following a model-driven specification of the IT & OT assets present in the environment that is within the protection scope.

Finally, the operator defines the playbooks to be used when the attack scenarios are triggered.

2.2.2.2 Sub-scenario 2: During the Attack Phase

The Attack phase for UC2 has several cyber incidents identified to demonstrate the response against such situations. Two incidents are related to the IoT edge device layer, including data tampering and Denial of Service to any of the components of the edge device layer. The other incident is related to possible attacks to the CMT cloud system, as a way to disrupt data or device availability to the OT system of the railway operator.

To manage the IoT edge device layer incidents, PHOENI2X uses the existing data and metadata streams from IoT edge devices and CMT cloud system as defined in the sub-scenario 1. PMEM identifies anomalies and notifies the IR tool (ROAR). ROAR, in turn, triggers the playbook defined for responding to such anomaly and also creates an incident case at SMIR and notifies the operator. Based on the information received, the operator needs to handle the situation by connecting to CMT and some IoT edge devices, to decide on the veracity and range of the threat detected. After handling the incident, the operator closes the incident case at SMIR, while SMIR notifies ROAR, ending the execution of the relevant process. Throughout this process, the operator has a view of the status at the FVT.

In the particular case of attacks against the CMT cloud system, several EDR agents are reporting the events to the main EDR server. The EDR module identifies events, anomalies and threats that should be managed. All situations are reported to the SIEM and the status is visualised at FVT.

2.2.2.3 Sub-scenario 3: Post-Attack Phase

In the post attack phase, the human operator, first, consults the FVT in order to perform a root cause analysis for the incident. Next, the operator interacts with SMIR to generate an incident report. The report is inspected by the operator, who decides to share it to the relevant organisations. The report to be shared is generated by CTI and distributed according to the defined rules.

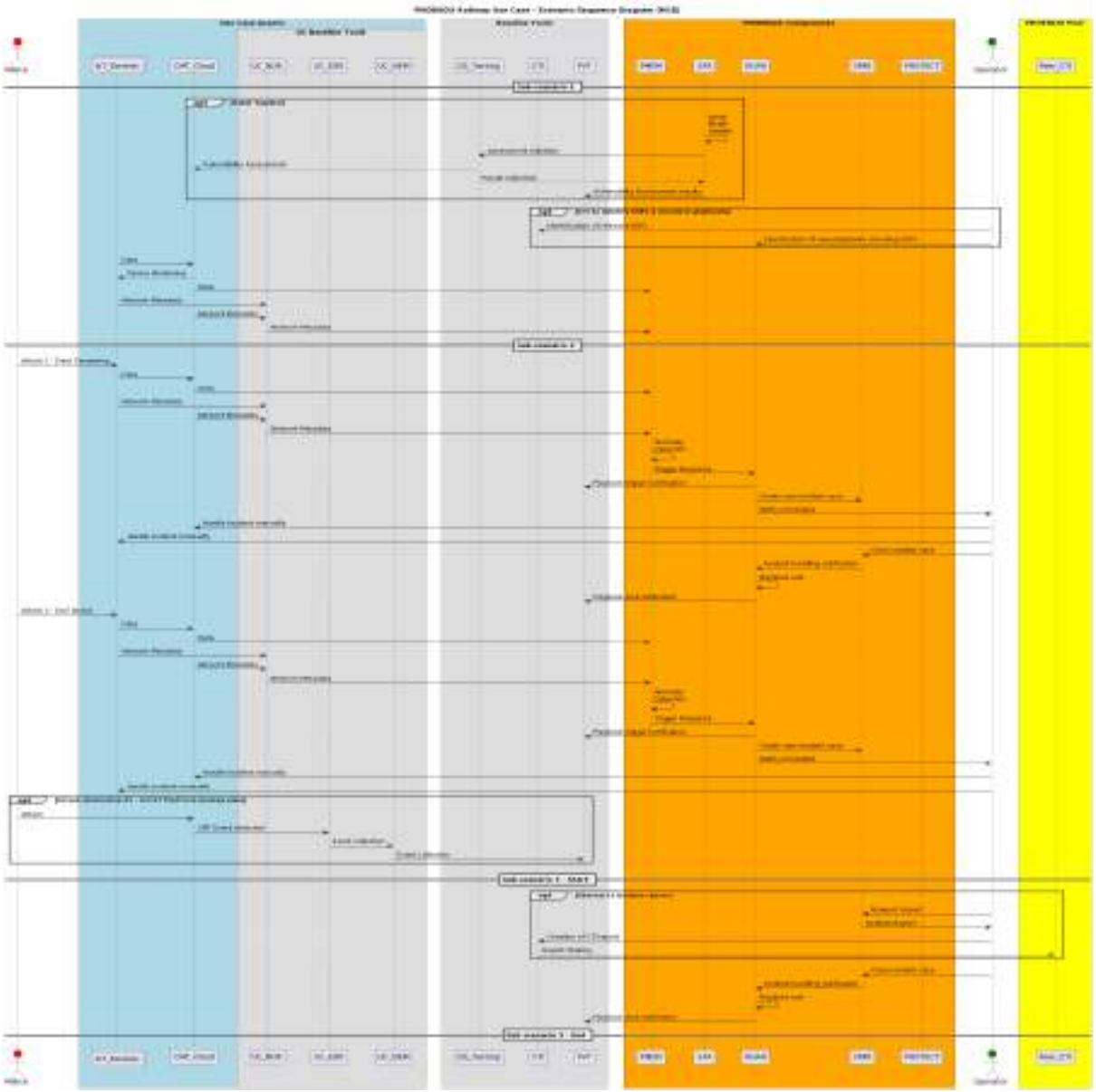

Figure 3: UML Diagram of UC2

2.2.3 Use Case 3 – Healthcare

Within the context of Use Case 3, the PHOENI2X integrated platform in tandem with the use case infrastructure will demonstrate the following scenarios:

- Baseline activity (Prevention and Preparedness)
- Use of a vulnerable library (Prevention and Preparedness)
- Unauthorized commit (During the Attack Phase)
- Removal of a code branch (During the Attack Phase)

A short description of the scenarios is detailed in the following sections and the respective UML sequence diagrams are presented in Figure 4

2.2.3.1 Baseline activity (Prevention and Preparedness phase)

In the daily workflow of healthcare portal development, a custom scheduler has been implemented in the testbed in order to simulate the activities of five developers who regularly commit code to the central server using remote access tokens. The respective network and firewall logs are collected from the various components of the use case testbed (Git, Jenkins, HAProxy, OPNSense, Keycloak, Wazuh) and are sent via the Elastic server log collector to the PHOENI2X integrated framework, and specifically to the FVT module for facilitating the analysis. In this baseline scenario, PHOENI2X employs AI-based algorithms and ultimately classifies the logged activities as non-malicious; the routine commits and remote interactions align with expected coding and behavioural practices, reflecting a secure development environment. This scenario also illustrates the level of robustness of the existing cybersecurity measures of the use case testbed, with using normal development activities to identify vulnerabilities and potential threats via a risk assessment report.

2.2.3.2 Use of a vulnerable library (Prevention and Preparedness phase)

In a deviation from the routine, a developer executes a commit and push operation which introduces a seemingly innocuous change that conceals a potential threat — the addition of a vulnerable library into the codebase. This modification is embedded within files like "pom.xml" or "package.json" and sets the stage for a nuanced cyber threat within the healthcare portal's development environment.

Upon the developer's push, the Git source code server promptly dispatches the modified file contents to the NPS_Control Plane module, which acts as a gatekeeper and directs the file to the SPA module of PHOENI2X for in-depth scrutiny. The Control Plane module initiates this process by sending a request to a dedicated endpoint in SPA, which encapsulates the file content as the body of the request. The SPA notifies the TII and CERCA modules about the new file contents.

- In the case of active detection, i.e., the library is *already known* to be vulnerable, the SPA component detects the vulnerability and notifies the ROAR component (Alternative #2 in the UML diagram).
- Alternatively, if the vulnerability of the library is *not yet known* at this point in time but is discovered at a later stage, the CTI module, which is subscribed to external events containing new vulnerability announcements, notifies the TII module about the new vulnerability *once it becomes known*. Herein, the TII module, matches the new vulnerability with the respective existing library (or libraries) used, and then notifies the ROAR component to send a response to the testbed (Alternative #1 in the UML diagram).

In both cases, the ROAR component activates the appropriate playbook to respond to the threat. As part of the playbook, the ROAR component swiftly communicates its findings back to the Control Plane module. This communication serves as a directive, indicating that the identified library should not be incorporated into the codebase due to its vulnerability. In response to the PHOENI2X verdict, the NPS_Control Plane notifies the source code server either to abort the ongoing commit operation (Alternative #2) or notifies the appropriate component in the testbed about the need to change/update the library (Alternative #1).

This rapid and automated intervention prevents the integration of the compromised library into the codebase, thereby safeguarding the integrity and security of the healthcare portal's development environment. This scenario exemplifies the proactive nature of the PHOENI2X cybersecurity measures in place and showcases the system's ability to detect and mitigate potential threats immediately after being publicly disclosed, and in real-time (i.e., commit-time) when already known.

2.2.3.3 *Unauthorized commit (*During the attack *phase)*

In this sub-scenario, a threat actor successfully compromises the remote access token for the source code server, thus breaching the controlled environment of the healthcare portal's development infrastructure. Leveraging this unauthorized access, the attacker, operating from an alternative IP address, attempts to execute a commit and push operation, seeking to introduce unauthorized code modifications into the source code. During this activity, logs are captured from the Git and Jenkins servers, and in addition to the already existing network and firewall data captured, are sent via the Elastic server log collector (ELK_LogCollector) to the PHOENI2X integrated framework, and specifically to the FVT module to facilitate the analysis. Furthermore, the non-whitelisted IP present in the logs triggers a notification to the SPA module, which recognizes the anomalous nature of the commit/push attempt and in turn notifies the ROAR module to activate the appropriate playbook for responding to this attack. Concurrently, SPA also informs the CERCA module, which generates a report about the incident, providing a detailed account of the unauthorized access and modification. The ROAR module responds back to the NPS_ControlPlane, signalling the security incident. The NPS_ControlPlane then instructs the Git server to abort the ongoing commit/push operation. This orchestrated response mechanism efficiently neutralizes the malicious attempt, thereby preserving the integrity and security of the codebase of the healthcare portal.

2.2.3.4 *Removal of code branch (During the attack phase)*

Similar to the previous sub-scenario, a threat actor successfully compromises the remote access token for the source code server. Exploiting this unauthorized access, the attacker, again utilizing an alternative, distinct IP address, initiates a strategic attempt to eliminate a specific code branch from the source code server, aiming to disrupt the codebase's continuity and integrity. Throughout this attempt, logs are captured from the Git and Jenkins servers, and in addition to the already existing network and firewall data captured, are sent via the Elastic server log collector (ELK_LogCollector) to the PHOENI2X integrated framework, and specifically to the FVT module to facilitate the analysis. In addition to the actions triggered by the detection of a non-whitelisted IP by SPA, the leveraging of contextual analysis capabilities through the UEBA module can be envisaged in a future release, so as to determine whether the code branch removal attempt is malicious. The result of the analysis is fed into the PHOENI2X ROAR module, which activates the appropriate response playbook and notifies the NPS_ControlPlane that the removal attempt should be aborted. Upon receiving this notification, the NPS_ControlPlane sends a command to the Git server to abort the removal attempt, thus preventing any compromise to the codebase's integrity, and thereby fortifying the overall integrity and security of the healthcare portal development environment.

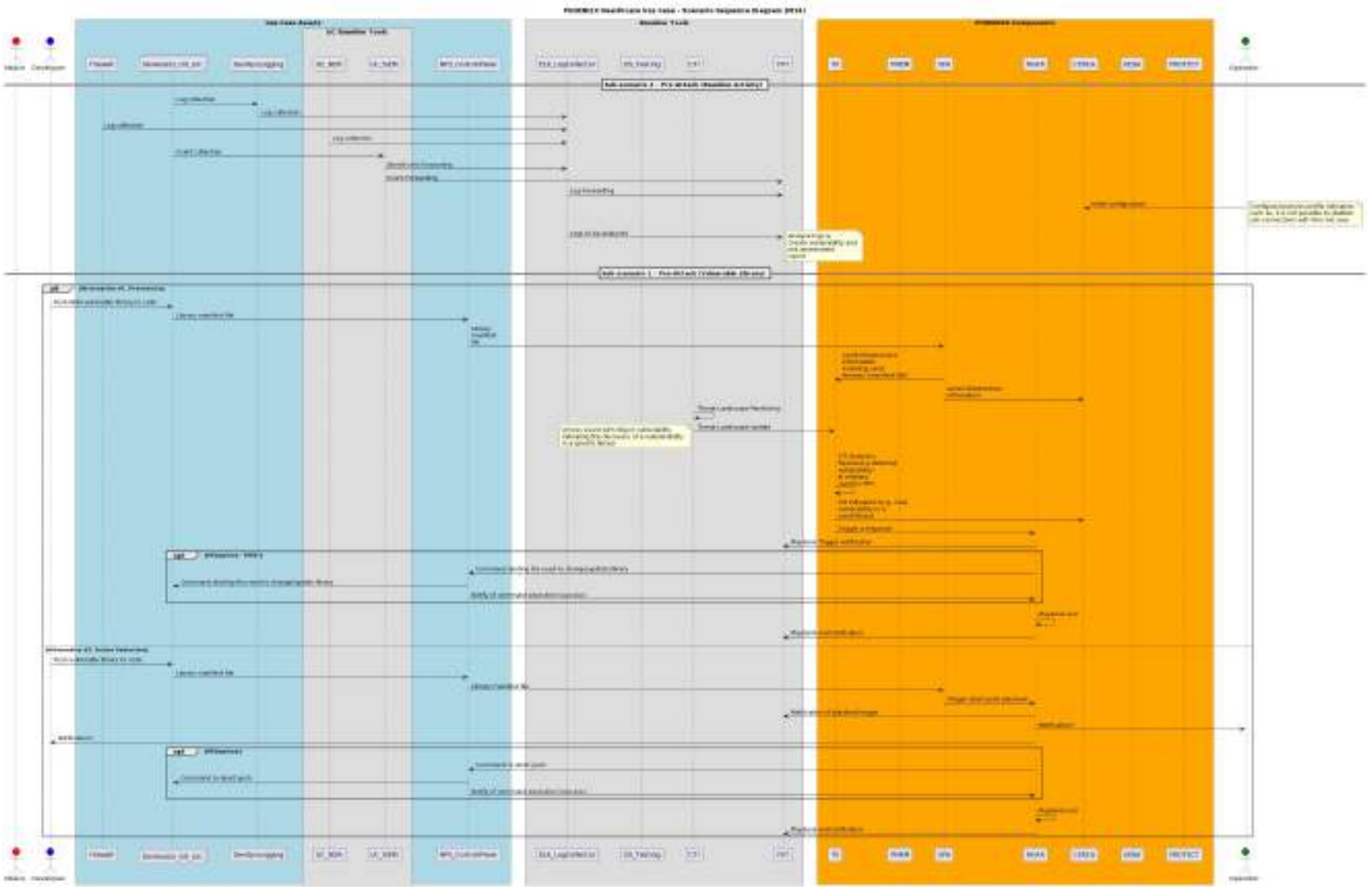

Figure 4: UC3 pre-attack scenarios sequence diagrams

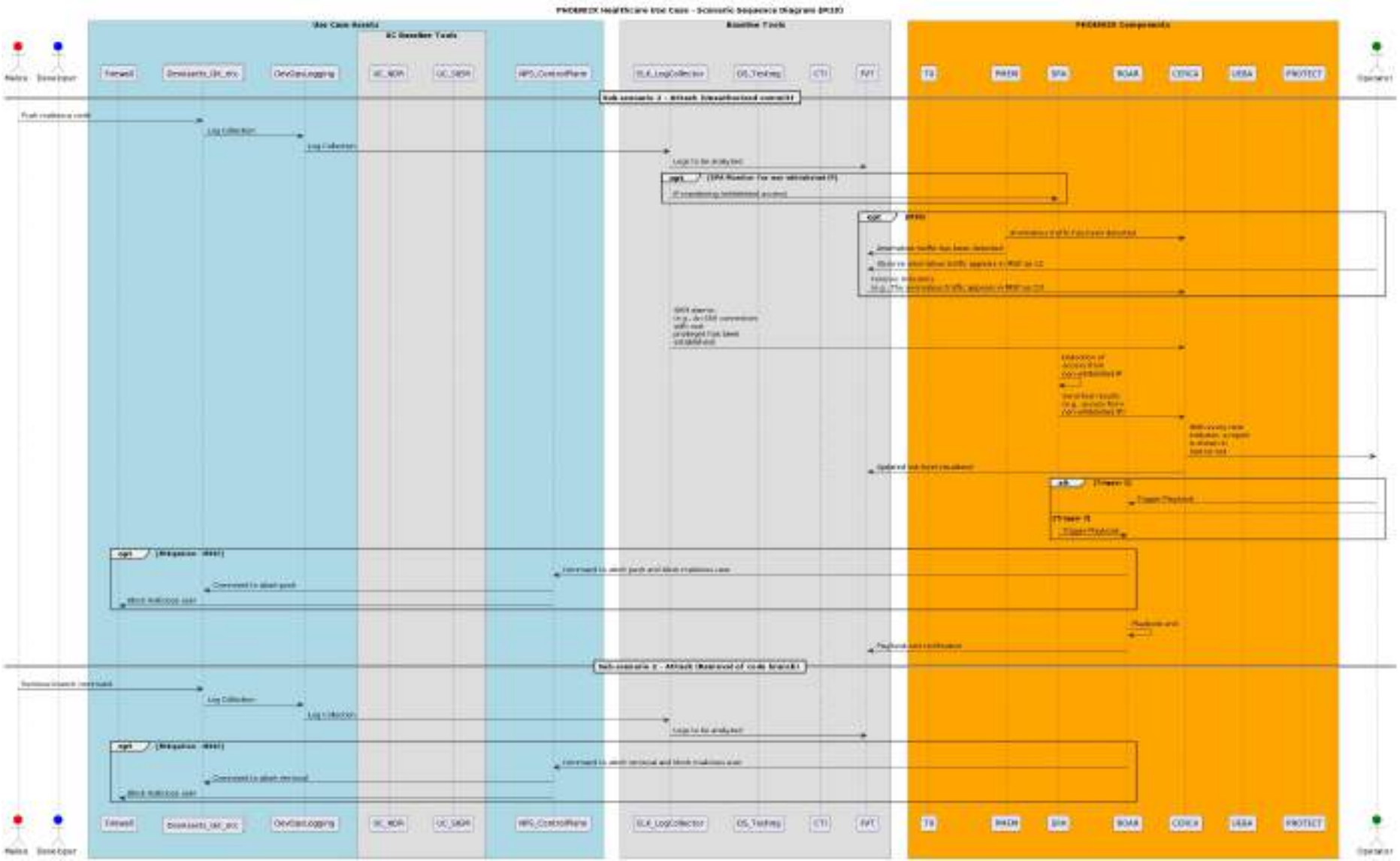

Figure 5: UC3 attack scenarios sequence diagrams

2.3 PHOENI2X Framework Deployment

This section presents the deployment of PHOENI2X framework inside the cloud infrastructure provided by UPAT. The overall resources provided by UPAT, are VCPU 200, RAM 256GB, DISK 1000GB and their current usage is depicted in the following Figure 6.

As mentioned in previous deliverables (e.g. D2.1 and D2.2), the PHOENI2X architecture follows a modular approach which allows for different types of deployments. Depending on their needs and requirements, the end user (e.g., Critical Infrastructure provider) may select to host the complete PHOENI2X solution in its premises, use the PHOENI2X platform as a service, or follow a hybrid solution. Since the scope of this deliverable is to present the MVP of the PHOENI2X framework, below we only describe the PHOENI2X deployment inside UPAT cloud. This is the type of deployment which allows our solution to be offered as a service (no core components installed at user premises). However, since our pilots have been selected to demonstrate all different types of deployments (local, service, hybrid), in D5.2 (Testbed Setup, Evaluation Methodology & Baseline Establishment) in which we will describe all testbeds in full detail, all the PHOENI2X platform deployment types will be presented.

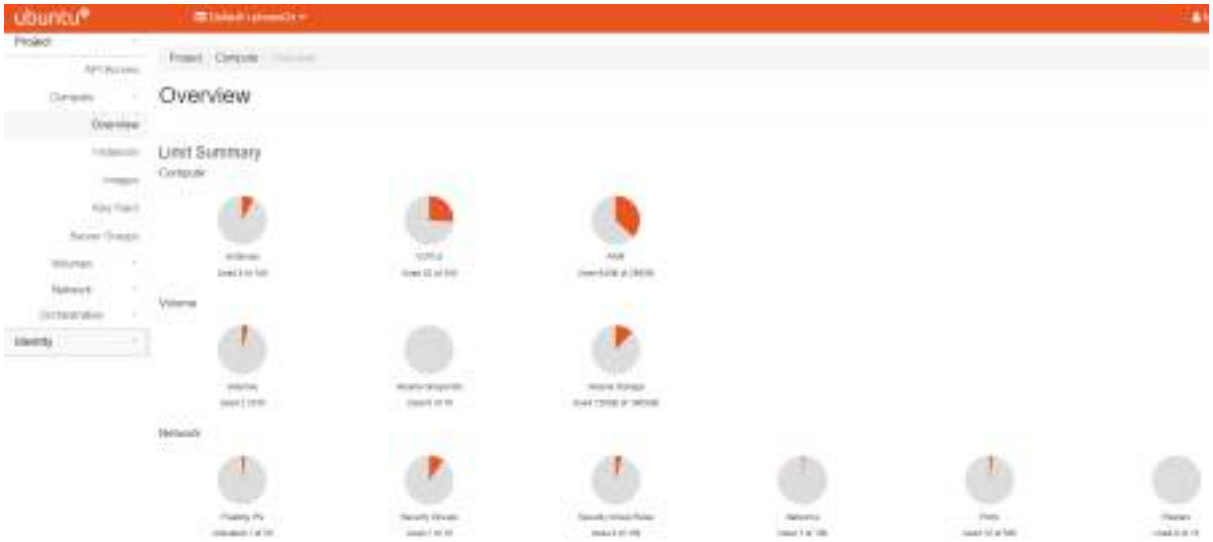

Figure 6: The UPAT resources for PHOENI2X

Below we describe all the PHOENI2X components that have been deployed and orchestrated inside the UPAT cloud. The following Virtual Machines (Instances) have been created (Table 1). Each one hosts the following services and/or PHOENI2X tools and components.

Table 1 PHOENI2X VMs

Instance	
Name:	Phoeni2x_Sphynx
Flavor:	VCPU 8, RAM 16GB, DISK 120GB
Owner	SANL
PHOENI2X Components:	• Dynamic/Static Testing • Security Assurance Platform • Resilience CR • UEBA • ROAR - Engine • ROAR - Playbooks

Instance	
Name:	aegisfvt
Flavor:	VCPU 2, RAM 12GB, DISK 40GB
Owner	AEGIS
PHOENI2X Components:	• Digital Forensics • Visualization

Instance	
Name:	atossiem
Flavor:	VCPU 16, RAM 24GB, DISK 80GB
Owner	ATOS
PHOENI2X Components:	• SIEM (Wazuh)

Instance	
Name:	atosmessagebus
Flavor:	VCPU 8, RAM 6GB, DISK 80GB
Owner	ATOS
PHOENI2X Components:	• Apache Kafka

Instance	
Name:	Upatrasgit
Flavor:	VCPU 2, RAM 12GB, DISK 40GB
Owner	UPAT
Purpose	ELK (Elasticsearch, Kibana, Beats, and Logstash)

Instance	
Name:	Phoeni2x_GW
Flavor:	VCPU 8, RAM 16GB, DISK 80GB
Owner	UPAT
Purpose	This VM acts as the gateway of the PHOENI2X cloud infrastructure. It hosts the VPN server as well as the NAT service.

Instance	
Name:	upatrasgit
Flavor:	VCPU 4, RAM 8GB, DISK 80GB
Owner	UPAT
Purpose	This VM acts hosts the project's Gitlab to act as the code repository and enable collaborative software development.

Instance	
Name:	Phoeni2x_atos_misp
Flavor:	VCPU 4, RAM 8GB, DISK 40GB
Owner	ATOS
Purpose	This VM acts hosts the MISP instance for Peer_CTI sharing demonstration

The deployment diagram is depicted in Figure 7.

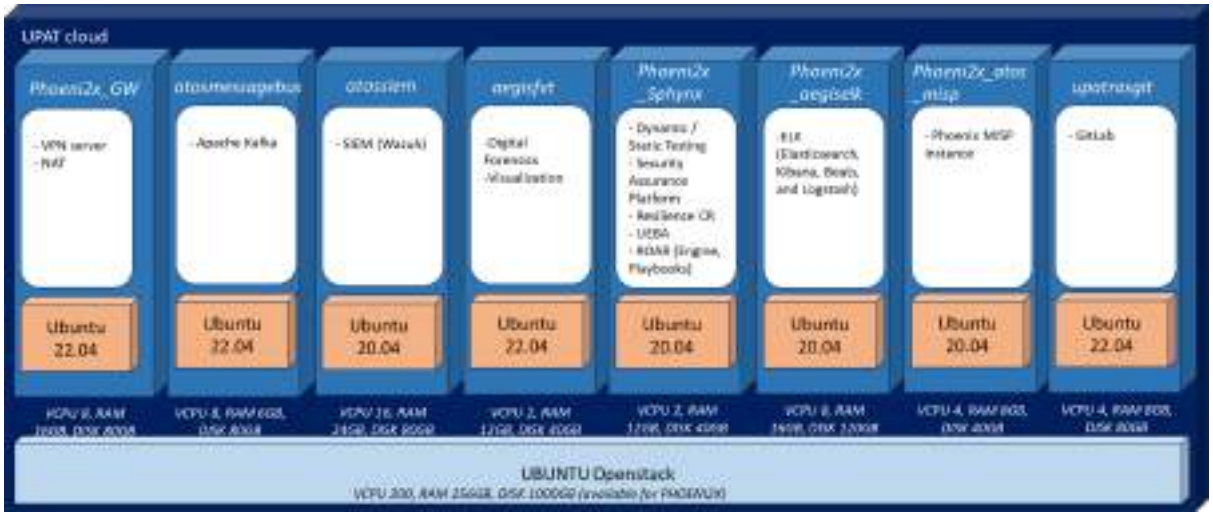

Figure 7: PHOENI2X Deployment diagram

3 The PHOENI2X Integrated Methodology and Plan

3.1 PHOENI2X Integration Plan

The PHOENI2X components are developed with different technologies and cover different technical requirements as well as configurations. The development and integration of the PHOENI2X framework and the pilots follow an iterative approach, which includes the following releases (Table 2):

(i) the first release on M18, constituting the MVP version of the framework and includes the WP2 baseline components as well as all available WP3 and WP4 components (released by M16),

(ii) the second release (V1) on M24, which includes the PHOENI2X components release on PM21 as part of D3.1 and D4.1,

(iii) the third release (V2) on M32, which will be the result of continuous integration activities on top of the V1 integrated framework based on the results from WP3 and WP4, and

(iv) the final refined integrated framework on M35.

The PHOENI2X architecture, being the result of T2.2, drives the integration of the framework. Any refinements of the architecture will be included in the implementations of WP3 and WP4, and, consequently, reflected in the PHOENI2X framework releases.

Table 2 PHOENI2X Integration Plan

Iteration	Integration Activities	Components	Responsible	Date
MVP PHOENI2X framework release	MVP Tools Release	All components of PHOENI2X framework	UPAT, SANL, ATOS, UPC, UiO,AEGIS,SEA	M16
	MPV integration activities	All components of PHOENI2X framework	NPS	M18
MVP (1st) Integration with UCs	Installation of Baseline Tools	PHOENI2X Baseline Tools	UPAT, ATOS, PPC, UPC,NPS	M15
	Integration with UC1 for MVP	**PHOENI2X Core Components:** DS_Testing, EDR, NDR, SIEM, CTI, FVT, SPA, ROAR, SMIR, TII, PMEM, PROTECT **UC1 Components:** SmartMeters, AMI_Headend, Firewall, SDN_Controller	PPC	M18
	Integration with UC2 for MVP	**PHOENI2X Core Components:** NDR, EDR, SIEM, DS_Testing, CTI, FVT, SPA, ROAR, SMIR, PMEM, Peer_CTI **UC2 Components:** IoT_devices, CMT_Cloud	PPC (as T5.3 leader), WSE, FGC	M18
	Integration with UC3 for MVP	**PHOENI2X Core Components:** NDR, SIEM, DS_Testing, ELK_LogCollector, CTI, FVT, SPA, ROAR, TII, CERCA **UC3 Components:** Firewall, DevAssets_Git_etc, DevOpsLogging, NPS_ControlPlane	PPC (as T5.3 leader), NPS	M18

Iteration	Integration Activities	Components	Responsible	Date
V1 PHOENI2X framework release	V1 Tools Release	All components of PHOENI2X framework and Baseline Tools	UPAT, SANL, ATOS, UPC, UiO,AEGIS,SEA	M21
	V1 Integrated Framework Release	All components of PHOENI2X framework	NPS	M24
V2 PHOENI2X framework release	V2 Tools Release	All components of PHOENI2X framework	UPAT, SANL, ATOS, UPC, UiO,AEGIS,SEA	M32
	V2 integration activities	All components of PHOENI2X framework	NPS	M32
Final (V2', i.e., refined V2) PHOENI2X framework release	Final (V2') Tools Release	All components of PHOENI2X framework	UPAT, SANL, ATOS, UPC, UiO,AEGIS,SEA	M35
	Final (V2') integration activities	All components of PHOENI2X framework	NPS	M35
Final (V2') PHOENI2X integration with pilots	Integration with UC1 for Final (V2')	**PHOENI2X Core Components:** DS_Testing, EDR, NDR, SIEM, CTI, FVT, SPA, ROAR, SMIR, TII, PMEM, PROTECT, Deception Tools **UC1 Components:** SmartMeters, AMI_Headend, Firewall, SDN_Controller	PPC	M35
	Integration with UC2 for Final (V2')	**PHOENI2X Core Components:** NDR, EDR, SIEM, DS_Testing, CTI, FVT, SPA, ROAR, SMIR, PMEM, Peer_CTI **UC2 Components:** IoT_devices, CMT_Cloud	PPC (as T5.3 leader), WSE, FGC	M35
	Integration with UC3 for Final (V2')	**PHOENI2X Core Components:** NDR, SIEM, DS_Testing, ELK_LogCollector, CTI, FVT, SPA, ROAR, TII, CERCA, UEBA, PMEM **UC3 Components:** Firewall, DevAssets_Git_etc, DevOpsLogging,NPS_ControlPlane	PPC (as T5.3 leader), NPS	M35

3.2 Integration Methodology

The PHOENI2X integration process is facilitated through its proper documentation and monitoring. Towards this direction, we have developed integration matrices (one for the PHOENI2X framework and one for each use case implementation), which represent the interactions between the PHOENI2X

baseline and core components, on the basis of a Design Structure Matrix (DSM)[1]. In these matrices, each row and column represent a PHOENI2X component. A similar matrix has been constructed for each use case. It should be noted that a unique code has been used for each interaction which is in the form of [component X id].[component Y id]. *This code indicates a link from component X to component Y*. The id of each component is unique and is the same across all tables for consistency. Moreover, the ids of the components of the use cases are distinguished from the PHOENI2X components on the basis of letters and numbers, with the numbers indicating the use case that the component relates to. For example, the interaction between SDN_Controller (with component id D.1 indicating that it relates to the use case 1) and the ROAR (component id 9), with the first one being the source component and the second one being the target component, is depicted as D1.9.

[1] The Design Structure Matrix (DMS) home page, https://dsmweb.org/introduction-to-dsm/, Retrieved on 24/11/2023.

Table 3 PHOENI2X Framework Integration Matrix

	1. EDR	2. NDR	3. DS_Testing	4. SIEM	5. CTI	6. FVT	7. ELK_LogCollector	8. SPA	9. ROAR	10. SMIR	11. CERCA	12. TII	13. PMEM
1. EDR				1.4									
2. NDR				2.4			2.7						2.13
3. DS_Testing								3.8					
4. SIEM						4.6	4.7	4.8			4.11		
5. CTI												5.12	
6. FVT													
7. ELK_LogCollector						7.6		7.8					
8. SPA			8.3			8.6			8.9		8.11	8.12	
9. ROAR						9.6				9.10			
10. SMIR									10.9				
11. CERCA						11.6							
12. TII					12.5				12.9		12.11		
13. PMEM									13.9				

Table 4 PHOENI2X and UC1 Integration Matrix

	A1. SmartMeters	B1. AMI_Headend	C1. Firewall	D1. SDN_Controller	1. EDR	2. NDR	3. DS_Testing	4. SIEM	5. CTI	6. FVT	7. ELK_ LogCollector	8. SPA	9. ROAR	10. SMIR	11. CERCA	12. TII	13. PMEM
A1. SmartMeters		A1.B.1															
B1. AMI_Headend	B1.A1									B1.6							B1.13
C1. Firewall																	
D1. SDN_Controller													D1.9				
1. EDR		1.B1						1.4									
2. NDR	2.A1							2.4									2.13
3. DS_Testing	3.A1	3.B1										3.8					
4. SIEM										4.6		4.8					
5. CTI																5.12	
6. FVT																	
7. ELK_LogCollector																	
8. SPA							8.3										
9. ROAR			9.C1	9.D1						9.6				9.10			
10. SMIR																	
11. CERCA																	
12. TII									12.5								
13. PMEM													13.9				

Table 5 PHOENI2X and UC2 Integration Matrix

	A2. IoT_devices	B2. CMT_Cloud	1. EDR	2. NDR	3. DS_Testing	4. SIEM	5. CTI	6. FVT	7. ELK_ LogCollector	8. SPA	9. ROAR	10. SMIR	11. CERCA	12. TII	13. PMEM	14. Peer_CTI
A2. IoT_devices		A2.B2		A2.2												
B2. CMT_Cloud	B2.A2		B2.1	B2.2											B2.13	
1. EDR						1.4										
2. NDR															2.13	
3. DS_Testing		3.B2								3.8						
4. SIEM																
5. CTI																5.14
6. FVT																
7. ELK_LogCollector																
8. SPA					8.3			8.6								
9. ROAR								9.6				9.10				
10. SMIR											10.9					
11. CERCA																
12. TII																
13. PMEM											13.9					
14. Peer_CTI																

Table 6 PHOENI2X and UC3 Integration Matrix

	A3. Firewall	B3. DevAssets_ Git_etc	C3. DevOpsLogging	D3. NPS_ ControlPlane	1. EDR	2. NDR	3. DS_Testing	4. SIEM	5. CTI	6. FVT	7. ELK_ LogCollector	8. SPA	9. ROAR	10. SMIR	11. CERCA	12. TII	13. PMEM
A3. Firewall											A3.7						
B3. DevAssets_ Git_etc			B3.C3	B3.D3				B3.4									
C3. DevOpsLogging											C3.7						
D3. NPS_ControlPlane	D3.A3	D3.B3										D3.8	D3.9				
1. EDR																	
2. NDR											2.7						
3. DS_Testing																	
4. SIEM										4.6	4.7				4.11		
5. CTI																5.12	
6. FVT															6.11		
7. ELK_LogCollector										7.6		7.8					
8. SPA													8.9		8.11	8.12	
9. ROAR				9.D3						9.6							
10. SMIR																	
11. CERCA										11.6							
12. TII													12.9		12.11		
13. PMEM																	

3.3 Integration Management

This section outlines the approach for managing and monitoring the integration of the components, along with the implementation progress of endpoints from both a technical and project management angle. Managing the integration leverages process and tools designed to ensure seamless communication among the partner organizations, facilitates collaborative development, and records essential details about interface implementation.

3.3.1 Technical integration

From a technical standpoint, implementing a consistent and well-maintained API documentation platform is essential for transparent communication. To this respect, tools such as Swagger/OpenAPI are used to provide a standardized way to describe APIs comprehensively. This enables partner organizations to understand the functionalities, parameters, and responses of each endpoint. The WP leader has been overseeing the regular and timely updates to the documentation, so as to ensure alignment with the evolving project requirements. In addition to documenting the endpoints, a version control system tool (GitLab) is used as a collaboration hub, allowing partners to contribute code, review changes, and maintain a clear history of modifications. This ensures that all stakeholders have visibility into the latest developments and can track progress effectively.

3.3.2 Project management

From a project management perspective, during a regular weekly meeting with all partners the current state of integration is discussed and documented in a dedicated shared spreadsheet, which provides the quick go-to solution for offering a high-level overview of the status of activities. The regular meeting serves as a means to discuss challenges, share progress updates, and ensure that all stakeholders are on the same page. In addition to the regular meeting, ad-hoc meetings are organized by the work package leader with selected partners to discuss and facilitated specific implementation aspects (e.g., for particular use cases or module-to-module communication). The ad-hoc meetings are performed to assist the partners involved to flesh out interface details, prioritize actions, resolve issues and set imminent implementation timelines, while ensuring that they maintain an open line of communication with the work package leader. For the tracking of progress, issues are created in GitLab for each interface endpoint, as well as for any other item that needs attention (e.g., a technical challenge needing clarification or a bug). Issues can be created by and assigned to any partner, and serve as a central point for discussing and tracking progress on specific work items. To further assist project management of the integration, the flow of work is visualized and managed using Kanban-style ‘Issue boards’ in GitLab, and internal milestones are defined to organize and group related issues, e.g., as the completion of a Use Case scenario which consists of a series of individual message exchanges between components.

3.4 Components' Interfaces

Based on the presented interactions in section 2.2 and the integration matrices in section 3.2, the PHOENI2X components' interfaces have been developed, which are presented in the following table (Table 7). The information it encapsulates includes:

(i) ID, being the unique identifier as set in the section 3.2 integration matrices;

(ii) Component A and B, being the names of the integrated components (output from component A, input to component B);

(iii) Responsible, setting the partner(s) responsible for the implementation and documentation of the respective integration endpoint;

(iv) Data Type and Protocol, being the ones used for the information exchange between component A and B;

(v) Status, presenting the current implementation status for the specific integration endpoint.

It should be noted that in this table, rows in yellow depict integration between UC components (or baseline components that the UC owners are responsible for), green ones relate to integration between UC components (or baseline components that the UC owners are in charge of) and PHOENI2X components and blue ones show integration between components of the PHOENI2X framework.

Table 7 PHOENI2X Components Integration Endpoints (incl. UCs)

Integration Endpoints			Responsible For		Data Type & Protocol	Status
ID	Component A	Component B	CompA	CompB		
A1.B1	SmartMeters	AMI_Headend	PPC	PPC	Electricity measurements (DLMS/COSEM)	Completed
B1.A1	AMI_Headend	SmartMeters	PPC	PPC	DLMS/COSEM Commands	Completed
B1.6	AMI_Headend	FVT	PPC	AEGIS	Logs (JSON)	Completed
B1.13	AMI_Headend	PMEM	PPC	UPC	Logs (JSON)	Completed
D1.9	SDN_Controller	ROAR	PPC	SANL	SDN-Controller operations (REST API)	Completed
2.A1	NDR	SmartMeter	UPC	PPC	Network traffic collection via port mirroring, DLMS/COSEM messages	Completed
2.4	NDR	SIEM	UPC	ATOS	JSON over Apache Kafka server	Completed
2.13	NDR	PMEM	UPC	UPC	JSON over Apache Kafka server	Completed
1.B1	EDR	AMI_Headend	PPC	PPC	Host OS Logs	Completed
1.4	EDR	SIEM	PPC	ATOS	JSON over port 1514. Wazuh Agent -> Wazuh	Completed
3.A1	DS_Testing	SmartMeters	SANL	PPC	Vulnerability scanning (internal network access)	Completed
3.B1	DS_Testing	AMI_Headend	SANL	PPC	Vulnerability scanning (internal network access)	Completed
4.6	SIEM	FVT	ATOS	AEGIS	JSON over Apache Kafka binary over TCP	Completed

Integration Endpoints			Responsible For		Data Type & Protocol	Status
ID	Component A	Component B	CompA	CompB		
4.8	SIEM	SPA	ATOS	SANL	Under discussion (Over Apache Kafka or at the Elastic DB level).	Completed
6.11	FVT	CERCA	AEGIS	ATOS	JSON over HTTP/REST API	Completed
5.12	CTI	TII	ATOS	ATOS	MISP Event (JSON) zeroMQ subscription	Completed
12.5	TII	CTI	ATOS	ATOS	MISP Event (JSON) via PyMISP	Completed
8.3	SPA	DS_Testing	SANL	SANL	Internal scripts & parser	Completed
9.C1	ROAR	Firewall	SANL	PPC	pfsense-api (REST API)	Completed
9.D1	ROAR	SDN_Controller	SANL	PPC	SDN-Controller operations (REST API)	Completed
9.6	ROAR	FVT	SANL	AEGIS	JSON over HTTP/REST API	Completed
9.10	ROAR	SMIR	SANL	ATOS	Over HTTP/REST APIs	Completed
13.9	PMEM	ROAR	UPC	SANL	Discussion in Progress	Completed
A2.B2	IoT_devices	CMT_Cloud	WSE	WSE	JSON over HTTP/REST API	Completed
A2.2	IoT_devices	NDR	WSE	WSE	JSON over HTTP/REST API	Completed
B2.A2	CMT_Cloud	IoT_devices	WSE	WSE	JSON over HTTP/REST API	Completed
B2.2	CMT_Cloud	NDR	WSE	WSE	Plain text over HTTP (Prometheus)	Completed

Integration Endpoints			Responsible For		Data Type & Protocol	Status
ID	Component A	Component B	CompA	CompB		
B2.1	CMT_Cloud	EDR	WSE	WSE	Plain text over HTTP/REST API (Wazuh)	Completed
B2.13	CMT_Cloud	PMEM	WSE	UPC	JSON/MQTT	Completed
2.13	NDR	PMEM	WSE	UPC	JSON/HTTP or MQTT	Completed
1.4	EDR	SIEM	WSE	ATOS	None (same tool used , Wazuh)	Completed
3.B2	DS_Testing	CMT_Cloud	SANL	WSE	Vulnerability assessment (local network access)	Completed
3.8	DS_Testing	SPA	SANL	SANL	Internal scripts & parser	Completed
4.6	SIEM	FVT	ATOS	AEGIS	JSON over Apache Kafka binary over TCP	Completed
5.14	CTI	Peer_CTI	ATOS	ATOS	MISP to MISP communication	Completed
8.3	SPA	DS_Testing	SANL	SANL	Internal scripts & parser	Completed
8.6	SPA	FVT	SANL	AEGIS	JSON over HTTP/REST API	Completed
9.6	ROAR	FVT	SANL	AEGIS	JSON over HTTP/REST API	Completed
9.10	ROAR	SMIR	SANL	ATOS	JSON over HTTP/REST API	Completed
10.9	SMIR	ROAR	ATOS	SANL	JSON over HTTP/REST API	Completed
13.9	PMEM	ROAR	UPC	SANL	Discussion in Progress	Completed
A3.7	Firewall	ELK_LogCollector	NPS	AEGIS	JSON over HTTP/REST API	Completed

Integration Endpoints			Responsible For		Data Type & Protocol	Status
ID	Component A	Component B	CompA	CompB		
B3.C3	DevAssetsGit_etc	DevOpsLogging	NPS	NPS	Kafka protocol	Completed
B3.D3	DevAssetsGit_etc	NPS_ControlPlane	NPS	NPS	JSON over HTTP/REST API	Completed
B3.4	DevAssetsGit_etc	SIEM	NPS	NPS	Wazuh agent	Completed
C3.7	DevOpsLogging	ELK_LogCollector	NPS	AEGIS	JSON over HTTP/REST API	Completed
D3.A3	NPS_ControlPlane	Firewall	NPS	NPS	JSON over HTTP/REST API	Completed
D3.B3	NPS_ControlPlane	DevAssetsGit_etc	NPS	NPS	Shell script over SSH	Completed
D3.8	NPS_ControlPlane	SPA	NPS	SANL	JSON over HTTP/REST API	Completed
D3.9	NPS_ControlPlane	ROAR	NPS	SANL	JSON over HTTP/REST API	Completed
2.7	NDR	ELK_LogCollector	NPS	AEGIS	JSON over HTTP/REST API	Completed
4.6	SIEM	FVT	NPS	AEGIS	JSON over HTTP/REST API	Completed
4.7	SIEM	ELK_LogCollector	NPS	AEGIS	JSON over HTTP/REST API	Completed
4.11	SIEM	CERCA	NPS	ATOS	JSON via Kafka	Completed
7.6	ELK_LogCollector	FVT	AEGIS	AEGIS	JSON over HTTP/REST API	Completed
7.8	ELK_LogCollector	SPA	AEGIS	SANL	JSON over HTTP/REST API	Completed
5.12	CTI	TII	ATOS	ATOS	MISP Event (JSON) zeroMQ subscription	Completed

Integration Endpoints			Responsible For		Data Type & Protocol	Status
ID	Component A	Component B	CompA	CompB		
12.9	TII	ROAR	ATOS	SANL	HTTP/REST APIs.	Completed
12.11	TII	CERCA	ATOS	ATOS	JSON via Kafka	Completed
8.9	SPA	ROAR	SANL	SANL	Internal Communication (broker-based)	Completed
8.11	SPA	CERCA	SANL	ATOS	JSON over HTTP/REST API or via Kafka	Completed
8.12	SPA	TII	SANL	ATOS	JSON over HTTP/REST API or via Kafka	Completed
9.D3	ROAR	NPS_ControlPlane	SANL	NPS	JSON over HTTP/REST API	Completed
9.6	ROAR	FVT	SANL	AEGIS	JSON over HTTP/REST API	Completed
11.6	CERCA	FVT	ATOS	AEGIS	JSON over Apache Kafka binary over TCP	Completed

4 PHOENI2X Framework Testing

In order to verify the interactions between the PHOENI2X components themselves, as well as between the UC components and the PHOENI2X components, a test case was developed for each interaction pair together with the expected results of the interaction. The following principles were taken into account:

- During the development and execution of the test case, the components being tested were considered as in isolation from the remaining system, allowing the focus to rest solely with the interaction.
- For the tests themselves, the interactions were performed under normal, expected conditions, i.e., without the handling of exceptional cases such as network outages.
- Where applicable, the format of the agreed upon API protocol was thoroughly inspected so as to ensure it met the defined specifications.
- Tests were performed iteratively, and always after implementing changes or updates to the components to ensure that ongoing development did not introduce new issues in the interactions.

The integration tests are summarized in the table below (Table 8), which outlines the actual tests performed for each interaction documented in section 3 as well as the result and the status of the testing process. It should be noted that tests regarding the interaction of components with the operator (OP) are also documented in this table.

Table 8 PHOENI2X framework interactions testing

Interactions			Test Description	Result	Status
ID	**Component A**	**Component B**			
A1.B1	SmartMeters	AMI_Headend	Check that SmartMeters are connected to the network correctly and can communicate with the AMI_Headend.	The SmartMeters are correctly connected to the network and can communicate with the AMI_Headend.	Completed
B1.A1	AMI_Headend	SmartMeters	Check that the AMI_Headend can request and receive readings from the SmartMeters through DLMS messages.	The AMI_Headend can request and receive readings from the SmartMeters.	Completed
B1.6	AMI_Headend	FVT	Check that AMI_Headend logs are transferred to FVT.	AMI_Headend logs are available at FVT.	Completed
B1.13	AMI_Headend	PMEM	Check that AMI_Headend logs are transferred to PMEM.	AMI_Headend logs are available at PMEM.	Completed
D1.9	SDN_Controller	ROAR	Check that the SDN Controller can notify the ROAR that the requested network configuration change has been implemented.	ROAR receives message and progresses playbook.	Completed
2.A1	NDR	SmartMeters	Check that the NDR makes specific requests to smart meters for specific readings	DLMS Response with the specific reading from smart meters.	Completed
2.4	NDR	SIEM	Check that the events from the NDR appear in the Wazuh Dashboard	NDR events are received by Wazuh.	Completed

Interactions			Test Description	Result	Status
ID	Component A	Component B			
1.B1	EDR	AMI_Headend	Confirm that the EDR is correctly installed and monitors the AMI_Headend host machine.	The EDR is correctly install and monitors the host machine where the AMI_Headend is installed.	Completed
1.4	EDR	SIEM	Check that the events from the Wazuh-agent are received in the Wazuh Server using the Wazuh Dashboard	Wazuh-agent events are received by Wazuh.	Completed
3.A1	DS_Testing	SmartMeters	Check that the dynamic testing also encompasses Smart Meters of the deployment (via network).	Smart Meter findings are included in dynamic testing results.	Completed (Not tested, but out of the box functionality of DS testing tool, as long as there is network access to the Smart Meters)
3.B1	DS_Testing	AMI_Headend	Check that the dynamic testing also encompasses the AMI Headend of the deployment (via network).	AMI Headend findings are included in dynamic testing results.	Completed (Not tested, but out of the box functionality of DS testing tool, as long as there is network access to the AMI Headend)
4.6	SIEM	FVT	Check that SIEM writes alerts in Kafka and FVT is able to subscribe to that topic	Wazuh alerts appear in FVT.	Completed

Interactions			Test Description	Result	Status
ID	Component A	Component B			
4.8	SIEM	SPA	Check that SIEM writes alerts in Kafka and SPA is able to subscribe to that topic	Wazuh alerts arrive to SPA.	Completed
6.OP	FVT	Operator	Check that FVT user interface is accessible to the operator behind Keycloak.	Accessible for PPC and UPAT deployments.	Completed
6.11	FVT	CERCA	Check that FVT can write into a Kafka topic and CERCA can subscribe to a Kafka topic in order to receive FVT messages	CERCA can receive indicators from FVT.	Completed
5.12	CTI	TII	Check that TII can receive updates via ZeroMQ when a new event is uploaded into MISP	New MISP events arrive to TII.	Completed
12.5	TII	CTI	Check that TII can access and modify events in MISp via PyMISP	MISP events are available for modification for TII.	Completed
8.3	SPA	DS_Testing	SPA triggers the DS_Testing tool.	DS-Testing tool starts upon trigger.	Completed
9.C1	ROAR	Firewall	ROAR interacts with Firewall to update rules.	Firewall rules updated per the ROAR configuration.	Completed
9.D1	ROAR	SDN_Controller	ROAR interacts with SDN controller to trigger network reconfiguration.	SDN controller updates network configuration based on ROAR configuration.	Completed

Interactions			Test Description	Result	Status
ID	Component A	Component B			
9.6	ROAR	FVT	ROAR information is sent and/or retrieved to/by FVT.	ROAR information are visualised in FVT.	Completed
9.10	ROAR	SMIR	Check that TheHive inside SMIR can be accessed via API request and a new incident can be created.	TheHive tickets are created/updated, as needed, via ROAR.	Completed
9.OP	ROAR	Operator	Operator is able to access the ROAR GUI	Operator views/edits/updates/creates/deploys playbooks on ROAR.	Completed
10.OP	SMIR	Operator	Check that the operator can access the GUI of SMIR.	The operator can access the SMIR GUI and interact with it.	Completed
13.9	PMEM	ROAR	Check that notification about anomalies or mitigation is sent to ROAR	PMEM notifications trigger a playbook on ROAR.	Completed
A2.B2	IoT_devices	CMT_Cloud	Data from IoT devices is sent through radio connection, forwarded by gateways, network server and collected at CMT Cloud	Data is stored and visualised at CMT Cloud for the IoT devices deployed.	Completed
A2.2	IoT_devices	NDR	Metadata from IoT device connectivity is collected at the network server	Metadata is available at network server.	Completed

Interactions			Test Description	Result	Status
ID	Component A	Component B			
B2.A2	CMT_Cloud	IoT_devices	Re-configuration of operational parameters are sent from CMT Cloud to the IoT devices	IoT device changes operating parameters as requested at CMT Cloud.	Completed
B2.2	CMT_Cloud	NDR	Metadata from CMT Cloud several services is collected and stored at the NDR	Metadata from cloud services is available at NDR.	Completed
B2.1	CMT_Cloud	EDR	EDR is capable of identifying the events from the machines and services involved in the CMT Cloud provisioning	Event information available at EDR.	Completed
B2.13	CMT_Cloud	PMEM	Check that PMEM can subscribe to a topic and receive messages from CMT_Cloud via MQTT broker	PMEM received data from CMT_Cloud.	Completed
2.13	NDR	PMEM	Check that PMEM can subscribe to a topic and receive messages from NDR via MQTT broker	PMEM received data from NDR.	Completed
1.4	EDR	SIEM	Check that the events from the Wazuh-agent are received in the Wazuh Server using the Wazuh Dashboard	Wazuh-agent events arrive to Wazuh server.	Completed
3.B2	DS_Testing	CMT_Cloud	Dynamic Testing to cover the assessment of the CMT_Cloud	Results from CMT_Cloud assets are included in the dynamic testing results.	Completed

Interactions			Test Description	Result	Status
ID	Component A	Component B			
3.8	DS_Testing	SPA	Results of DS_Testing are relayed to SPA	SPA visualises the DS_Testing assessment results.	Complete
4.6	SIEM	FVT	Check that SIEM writes alerts in Kafka and FVT is able to subscribe to that topic	FVT can directly consume the Kafka topic over socket.IO.	Completed
5.14	CTI	Peer_CTI	Exchange between two CTI tool (MISP) instances.	CTI objects are successfully transferred from one MISP instance to the other.	Completed
8.3	SPA	DS_Testing	SPA triggers the DS_Testing tool.	DS-Testing tool starts upon trigger.	Completed
8.6	SPA	FVT	SPA information is sent and/or retrieved to/by FVT.	SPA information is visualised in FVT.	Completed
9.6	ROAR	FVT	ROAR information is sent and/or retrieved to/by FVT.	ROAR information is visualised in FVT.	Completed
9.10	ROAR	SMIR	Check that TheHive inside SMIR can be accessed via API request and a new incident can be created.	TheHive tickets are created/updated, as needed, via ROAR.	Completed
9.OP	ROAR	Operator	Operator is able to access the ROAR GUI	Operator views/edits/updates/creates/deploys playbooks on ROAR.	Completed

Interactions			Test Description	Result	Status
ID	Component A	Component B			
10.9	SMIR	ROAR	SMIR can access ROAR API in order to notify that an incident has been closed	SMIR can access ROAR API.	Completed
10.OP	SMIR	Operator	Check that the operator can access the GUI of SMIR.	The operator can access the SMIR GUI and interact with it.	Completed
13.9	PMEM	ROAR	Check that notification about anomalies or mitigation is sent to ROAR	PMEM notifications trigger a playbook on ROAR.	Completed
A3.7	Firewall	ELK_LogCollector	Check that firewall logs are transferred to via Kafka to the ELK_LogCollector	Firewall logs are visible in the Kibana dashboard of the ELK_LogCollector.	Completed
B3.C3	DevAssetsGit_etc	DevOpsLogging	Check that access logs are transferred to ELK_LogCollector	GIT https traffic logs are visible in the Kibana dashboard from HAProxy.	Completed
B3.D3	DevAssetsGit_etc	NPS_ControlPlane	Check that access logs are transferred to ELK_LogCollector	HTTP traffic logs are visible in the Kibana dashboard from GIT to PCP.	Completed
B3.4	DevAssetsGit_etc	SIEM	Check that the agent in Git VM is sending data to Wazuh	GIT/Wazuh are communicating and synchronized.	Completed
C3.7	DevOpsLogging	ELK_LogCollector	Check that Packetbeat and Filebeat agents transfer Git and Jenkins logs to the ELK_LogCollector via Kafka	Git and Jenkins logs are visible in the Kibana dashboard of the ELK_LogCollector.	Completed

Interactions			Test Description	Result	Status
ID	Component A	Component B			
D3.A3	NPS_ControlPlane	Firewall	Check that REST message from NPS_ControlPlane containing command to block IP is received by Firewall	Firewall blocks the IP denoted in the REST message.	Completed
D3.B3	NPS_ControlPlane	DevAssetsGit_etc	Check that NPS_ControlPlane is communicating with Git on push	Git server should execute the command git push and the traffic will be visible on Kibana	Completed
D3.8	NPS_ControlPlane	SPA	Check that REST/JSON message from NPS_ControlPlane containing the library manifest is received by the SPA component	SPA component should acknowledge the request and process the library manifest	Completed
D3.9	NPS_ControlPlane	ROAR	Check that REST/JSON message from ROAR component containing notification, e.g., about vulnerable library is received by the NPS_ControlPlane	NPS_ControlPlane should acknowledge the request and act on the notification.	Completed
2.7	NDR	ELK_LogCollector	Check NDR can access the ELK_LogCollector to store data	NDR logs are visible in the Kibana dashboard of the ELK_LogCollector	Completed
4.6	SIEM	FVT	Check that SIEM writes alerts in Kafka and FVT is able to subscribe to that topic	FVT can directly consume the Kafka topic over socket.IO	Completed
4.7	SIEM	ELK_LogCollector	Check that SIEM writes alerts in Kafka and ELK_LogCollector is able to	ELK_LogCollector receives and creates indices for Kafka	Completed

Interactions			Test Description	Result	Status
ID	Component A	Component B			
			subscribe to that topic and store the alerts for persistence.		
4.11	SIEM	CERCA	SIEM alerts are written into a Kafka topic SIEM_out, messages are forwarded to CERCA_in topic so they are accessible as indicators	CERCA can access Wazuh alerts	Completed
7.6	ELK_LogCollector	FVT	Check that FVT connectors can receive and transmit data to the ELK_LogCollector.	FVT can access ELK_LogCollector and visualize data stored in the Elasticsearch server.	Completed
7.8	ELK_LogCollector	SPA	ELK_LogCollector information should be relayed to SPA monitoring subsystem	ELK_LogCollector data are successfully integrated with SPA monitoring.	Completed
5.12	CTI	TII	Check that TII can receive updates via ZeroMQ when a new event is uploaded into MISP	New MISP events arrive to TII.	Completed
12.9	TII	ROAR	Check that TII has access to ROAR and can trigger playbooks via API.	TII can notify to ROAR the execution of a playbook when the score of a MISP event surpasses a specific threshold.	Completed
12.11	TII	CERCA	Check that TII writes events into Kafka topic and CERCA can access them	CERCA have MISP events available as CTI indicators	Completed (Need to integrate the data into CERCA but messages are

Interactions			Test Description	Result	Status
ID	Component A	Component B			
					received into CERCA_in topic)
8.9	SPA	ROAR	SPA should trigger ROAR as needed (e.g., via Monitor)	ROAR is triggered by SPA	Completed
8.11	SPA	CERCA	Infrastructure information is sent to CERCA via Kafka/API or CERCA can access that information via API	CERCA has access to infrastructure information.	Completed
8.12	SPA	TII	Infrastructure information and SBOM are sent to TII via Kafka/API, or TII can access that information via API	TII has access to infrastructure information and SBOM.	Completed
9.D3	ROAR	NPS_ControlPlane	ROAR should interact with NPS_ControlPlane to trigger mitigation.	ROAR-issued mitigation strategy is enforced within the NPS environment.	Completed
9.6	ROAR	FVT	ROAR information is sent and/or retrieved to/by FVT.	ROAR information are visualised in FVT.	Completed
9.OP	ROAR	Operator	Operator is able to access the ROAR GUI	Operator views/edits/updates/creates/deploys playbooks on ROAR	Completed
11.OP	CERCA	Operator	Make sure the CERCA GUI works so the operator can access CERCA reports	The operator can visualize detailed information about the reports in CERCA.	Completed

Interactions			Test Description	Result	Status
ID	Component A	Component B			
11.6	CERCA	FVT	Make sure CERCA write reports into a Kafka topic and that information is available by FVT via topic subscription	CERCA reports are available for FVT.	Completed

5 Conclusion and Next Steps

This document is the report accompanying the first release (MVP) of the PHOENI2X framework. It has presented an overview of the PHOENI2X components integrated in this first project iteration, as well as the three (3) use cases, in terms of their interactions both of their internal components and with the PHOENI2X framework. Moreover, the integration methodology, tools and plan have been described. Based on the latter, the detailed list of interactions has been documented followed by their status of implementation and the data types and protocols applied. The test cases have also been presented for each pair of interacting components based on which the framework has been tested.

The MVP integrated framework will serve as a basis for the next releases of the integrated PHOENI2X framework through a continuous integration process. Within this context, the components developed in WP3 and WP4 will be incorporated into the PHOENI2X integrated framework throughout the project's lifetime, feeding the activities towards the next (V1) release on M24, a pre-final one (V2) including also feedback from the validation activities (T5.3) and the final one (V2') which is to encapsulate all refinements of the PHOENI2X components. It should be noted that any requirements for modification and / or refinement of the PHOENI2X architecture will be captured and promptly and properly communicated, ensuring that each integrated PHOENI2X framework release is accompanied by a consistent and up-to-date architectural diagram.

Annex – Examples of components' interactions

Figure 8 shows that the communication between the NDR-SmartMeter is successful and the reading from the smart meter is received successfully. The requests for specific reading are being generated from the NDR using the DLMS protocols. These requests are being sent to the smart meter after a predefined interval. In the response of these requests the smart meter sends the specific meter readings requested by the NDR. These messages are further dissected and will be available on the Kafka server for the intended subscriber.

Figure 8 Test for getting data between NDR-SmartMeter

Figure 9 shows a sample of data received by PMEM from CMT_Cloud. The PMEM component subscribes to an MQTT topic provided by CMT_Cloud using an MQTT client. Each time the sensors send data, a MQTT message is received in JSON format. From these messages the payload and readTimestamp fields are extracted and are stored into a database.

Figure 9 Sample of data received by PMEM from CMT_CLoud

Figure 10 shows a sample of data received by PMEM from NDR. The PMEM component subscribes to an MQTT topic provided by NDR using MQTT client. Whenever the sensors transmit data, an MQTT message is received in JSON format. From these messages all the fields are extracted and stored in a database.

Figure 10 Sample of data received by PMEM from NDR