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

D4.1: Coordinated Response & Preparedness Enablers v1

Abstract: This document reports on the first release of enablers developed under PHOENI2X WP4 ("Coordinated Response & Preparedness Enablers"), specifically covering the outputs of Task 4.1 ("Resilience Orchestration, Automation & Response"), Task 4.2 ("Resilience Playbooks Specification, Translation & Lifecycle Management"), Task 4.3 ("Cyber Range & Serious Games for Resilience Assessment & Training"), and Task 4.4 ("Task 4.4 - Alerting, Reporting & Information Exchange"). The main goal of this document is accompany the release of the enablers, providing some additional details on their design and development.

Contractual Date of Delivery	31/03/2024
Actual Date of Delivery	08/04/2024
Deliverable Security Class	PU
Editor	*Konstantinos Fysarakis (SANL)*
Contributors	UPAT, SANL, COSM, FGC, PPC, WSE, AEGIS, SEA, ATOS, APS, NPS, UiO, DSA
Quality Assurance	*UPC, PPC*

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

Document Revisions & Quality Assurance

Internal Reviewers

UPC (Eva Rodriguez**)**

PPC (Dimitrios Merkouris, Nikolaos Nikoloudakis**)**

Revisions

Version	Date	By	Overview
0.1	14/02/2024	Editor	Initial draft, with ToC & assignments
0.3	29/02/2024	Editor	Initial content & structure update
0.4	05/03/2024	Editor	Contributions' integration
0.5	18/03/2024	ATOS	Add section 5.1
0.6	19/03/2024	Editor	Introductory & Concluding remarks
0.7	21/03/2024	AEGIS	Input to section 3
0.9	25/03/2024	Editor	Creation of review-ready version

Contents

List of Tables

List of Figures

List of Abbreviations

AI: Artificial Intelligence

APT: Advanced Persistence Threat

ATM: Attack Categorization Modelling

BC: Business Continuity

CACAO: Collaborative Automated Course of Action Operations

CI: Critical Infrastructures

CNN: Convolutional Neural Network

CRC: Cyber Resilience Center

CTI: Cyber Threat Intelligence

DoA: Description of Action

DTL-EL: Deep Transfer Learning for Exploit Labelling

DX.X: Deliverable X.X

ENISA: European Union Agency for Cybersecurity

EU: European Union

FAIR: Factor analysis of Information Risk

FSM: Finite State Machine

GAN: Generative Adversarial Network

GUI: Graphical User Interface

IoC: Indicator of Compromise

IR: Incident Response

KPI: Key Performance Indicator

LSTM: Long Sort Term Memory

ML: Machine Learning

MS: Member State

MSA: Micro-Service Architecture

MSRA: Maritime Supply Chain Risk Assessment

MTTA: Mean Time To Acknowledge

MTTR: Mean Time To Remediate

MVP: Minimum Viable Product

NLP: Natural Language Processing

OES: Operators of Essential Services

PMEM: Predictive Maintenance

RCR: Resilience Cyber Range

ROAR: Resilience Automation, Orchestration, and Response

RP: Resilience Playbook

SIEM: Security Information & Event Management System

SOAR: Security Orchestration, Automation and Response

TIP: Threat Intelligence Platform

TL: Transfer Learning

TTP: Tactics, Technics and Processes

UEBA: User and Entity Behaviour Analytics

WP: Work Package

XDR: eXtended Detection and Response

1 Introduction

1.1 Purpose of the Document

This deliverable is the first output of WP4 ("Coordinated Response & Preparedness Enablers"), documenting the delivery of the first version of the relevant components of PHOENI2X. As such, it documents the delivery of enablers coming from the underlying tasks, including:

- **Task 4.1 - Resilience Orchestration, Automation & Response**, providing components enabling to the automation & orchestration of resilience (incident response & business continuity) – related workflows. These results are documented in Section 2.
- **Task 4.2 - Resilience Playbooks Specification, Translation & Lifecycle Management**, providing the machine-processable & executable playbooks themselves that encode the above workflows. These results are documented in Section 3.
- **Task 4.3 - Cyber Range & Serious Games for Resilience Assessment & Training**, providing the components supporting the preparedness of involved users (both concerning testing & training at the process, but also the overall training and awareness at the human level). These results are documented in Section 4.
- **Task 4.4 - Alerting, Reporting & Information Exchange**, providing the components and processes enabling the timely production & exchange of information, as needed both operationally (e.g., to ensure timely alerting & repones), but also from a regulatory perspective. These results are documented in Section 5.

Details on the above will be provided in the sections that follow, with each Task having a dedicated section, as noted. Finally, Section 6 provides the concluding remarks and pointers to next steps.

1.2 Overall Methodology & Relationship with other PHOENI2X Activities & Deliverables

Overall, the above WP4 components cover a big part of the overall PHOENI2X framework, as visualised in Figure 1.

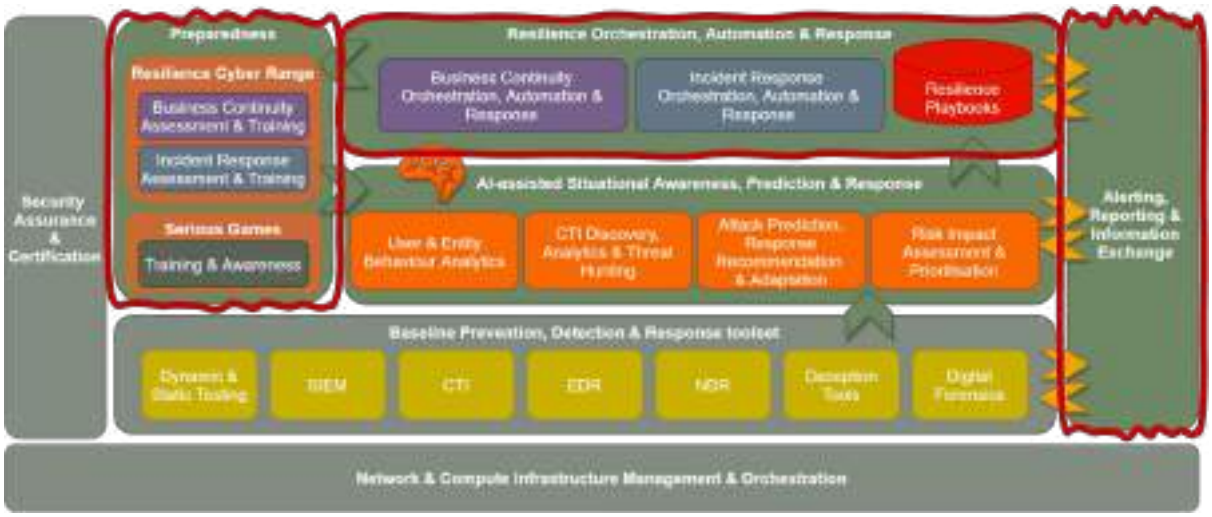


Figure 1. WP4 components within the PHOENI2X high level architecture.

In fact, WP4 outputs provide key enablers aligned with (and necessary to achieve) 3 of the project's overarching objectives, including:

- **Objective 2** (*"To design & develop Resilience Orchestration, Automation and Response mechanisms, encompassing proactive and reactive business continuity, recovery and incident handling tasks"*), covered by Tasks 4.1 & 4.2.
- **Objective 3** (*"To offer enhanced Preparedness through a Resilience Cyber Range and Serious Games"*), covered by Task 4.3.
- **Objective 4** (*"To provide Alerting, Reporting & Information Exchange mechanisms & processes enabling collaboration between private and public critical sector actors at the national and European level"*), covered by Task 4.4.

From a practical perspective, the component design & development activities described herein used the outputs of WP2 as foundational inputs, including the component-level specifications, functional & non-functional requirements and interfacing requirements stemming from the PHOENI2X architecture, as documented in D2.1 ("PHOENi²X Requirements & Architecture"). With the above at hand, the development of the individual WP4 components proceeded, with inputs from other activities & actors (e.g., use case owners), as needed.

The natural output of WP4 outputs (along with the outputs of WP3) is WP5, for the integration and eventual demonstration & validation of PHOENI2X. In this context, valuable input for the development of V1 of WP4 components was also provided by the early integration & demonstration efforts carried out within WP5 to derive the MVP version of PHOENI2X (as documented in D5.1 - "PHOENI2X framework - MVP").

In the same manner, iteratively, the components presented herein will be delivered to WP5, to ensure their integration into the release of V1 of PHOENI2X, which in turn will provide feedback for the 2nd and final release of the components, leading the 2nd and final release of PHOENI2X (to be documented in D5.2 - "PHOENI2X framework - Final".

2 Resilience Orchestration, Automation & Response

2.1 Overview

The Resilience Orchestration, Automation & Response (ROAR) component of PHOENI2X is built upon the SPHYNX Incident Response (IR) tool; a system that enables the manual or automated execution of Collaborative Automated Course of Action Operations (CACAO [1]]) security playbooks (for more details on the Playbooks we defer the reader to Section 3 that follows). The system offers a graphical drag-n-drop interface for creating and editing CACAO security playbooks, that can later be executed or exported as CACAO JSON files following the CACAO specification. In essence, this component acts as a security orchestration, automation, and response (SOAR) solution supporting the prevention, detection, investigation, and response to cybersecurity attacks, offering:

- User-friendly specification and testing of incident response CACAO playbooks via a graphical editor that requires no coding;
- Automated importing and execution of CACAO playbooks specified by 3rd parties;
- Continuous run-time monitoring and analysis of CACAO playbook execution;
- Orchestration of external tools as part of CACAO playbook executions;
- End user notifications and probing;
- Extensible integrations with other essential tools (e.g., ticketing & notification systems, security appliances)

The tool can be set up either as a standalone system or as a module (tool) integrated with the SPHYNX Security and Privacy Assurance Suite (SPA; detailed in Section 4 of deliverable D2.2 - “Baseline Enablers”).

The positioning of the ROAR tool in the PHOENI2X high-level and detailed architecture is shown in Figure 2 and Figure 3, respectively.

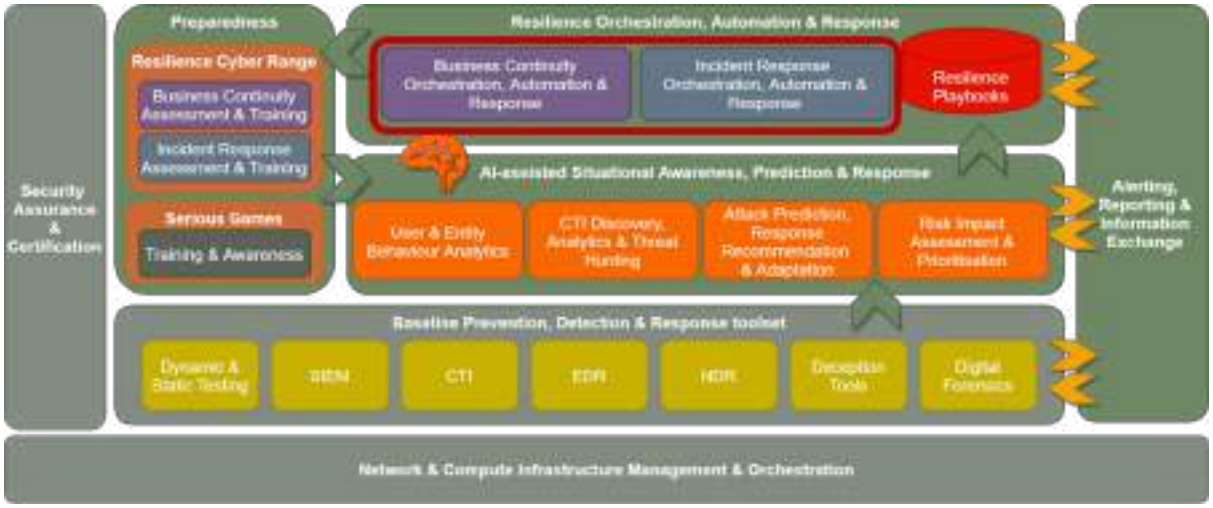


Figure 2. ROAR in the high-level PHOENI2X architecture.

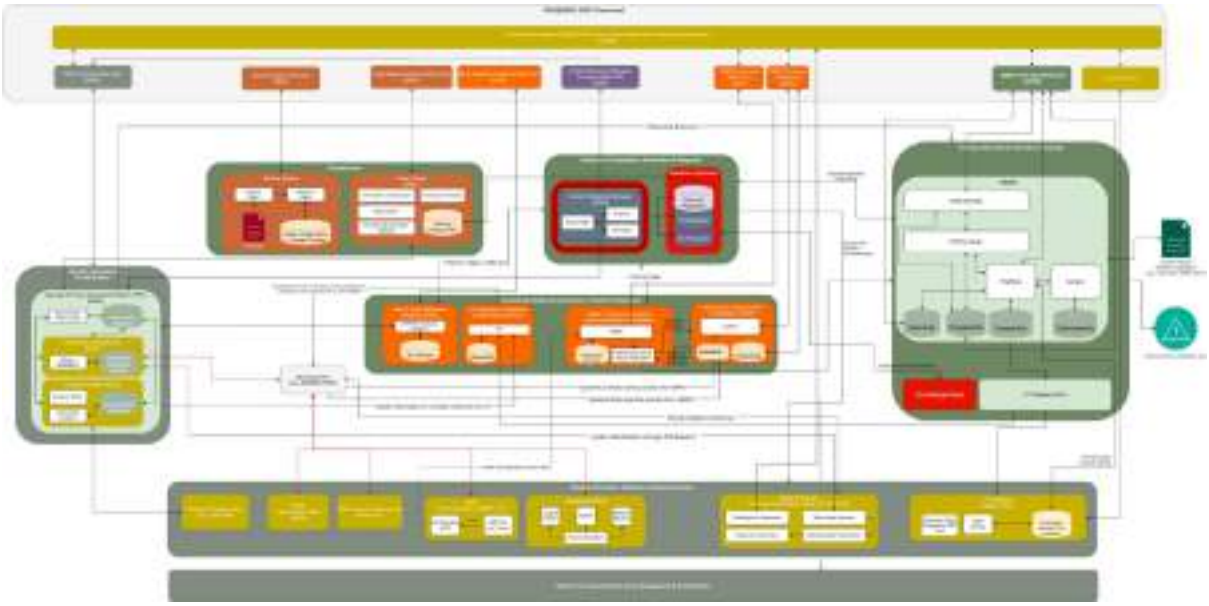


Figure 3. ROAR in the detailed PHOENI2X architecture.

The subsections that follow provide more details on the design & development of the ROAR tool in PHOENI2X.

2.2 Design Details

As mentioned, the ROAR tool offers an interactive dashboard that allows for the specification, editing & execution of playbooks. It also provides real-time views of the system's status and the execution of security playbooks along with high- and low- level logs, KPIs, user notification and other information.

Figure 4 shows how a playbook (in this case a preventative playbook related to the FuzzyPanda malware) can be defined within the ROAR GUI. The use of ROAR & CACAO-based playbooks ensures that a defined playbook can be shared between organisations and, when customised, to be executed as needed.

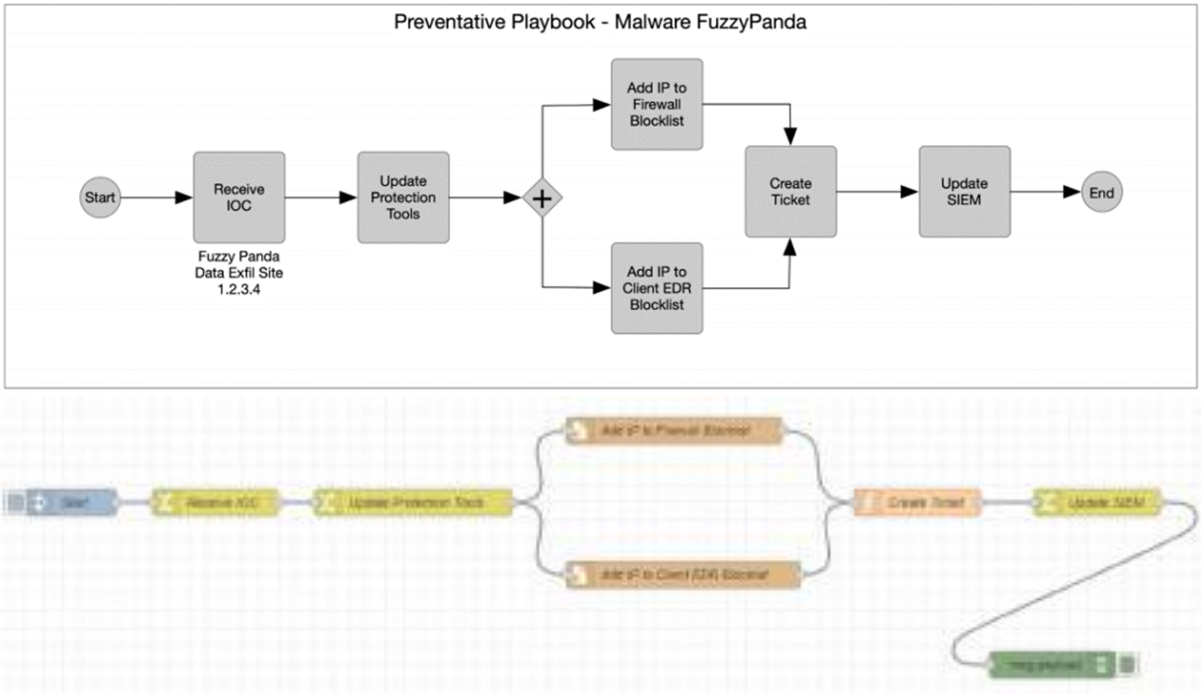


Figure 4. Preventative playbook concerning the FuzzyPanda malware workflow (top) & ROAR representation (bottom).

The ROAR-defined playbooks can be as complex as needed, interacting with security appliances, messaging & ticketing systems, or even integrating other playbooks within their workflows, as supported by the CACAO specification.

Overall, as the ROAR design & capabilities are fully tailored to supporting & implementing the CACAO specification, we defer the reader to Section 3below, where we provide more details on the Resilience Playbooks themselves (e.g., on the design of each of the playbook steps supported by ROAR).

2.3 Implementation Details

ROAR's internal architecture is composed of five main components as presented in Figure 5. The various components can be co-located on a single host or distributed across multiple hosts. Each component is isolated in its dedicated Docker container while native deployment is also available but not recommended. Also, the components should preferably be deployed on a secured and monitored infrastructure, outside of the target organization to ensure their uninterrupted and tamper-proof execution.

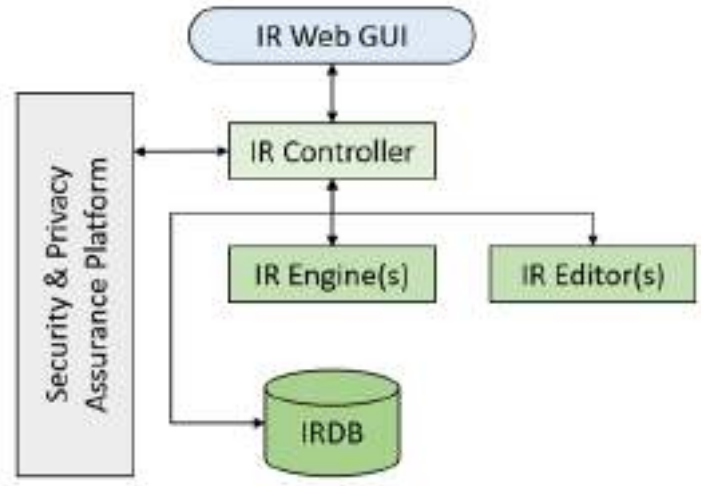


Figure 5. Internal architecture of ROAR.

The main components are as follows:

- **IR Web GUI:** The Web-based front-end through which the end users and administrators can interact with and control the ROAR. The GUI is implemented using the Angular[1] framework and provides dedicated views for browsing the available playbooks and allows users to enable or disable a playbook's availability for execution. Also, the GUI offers real-time views of enabled playbooks through which the platform's operators can monitor, start, stop, or restart a playbook's execution. The monitoring view includes an interactive, real-time graph presenting each playbook's step status as the playbook is executing, high- and low-level logs, the event that triggered the execution and various statistics, such as execution metrics. Also, administrators can manage the available playbook execution engines, create, or delete engine instances and assign playbooks to the available engines. Finally, the GUI provides a graphical drag-and-drop editor where playbooks can be edited or created by modifying their execution graph, exported as CACAO JSON files, or imported via valid CACAO JSON files.

[1] https://angular.io/

- **IR Controller:** The Java/Spring Boot-based[2] REST controller that orchestrates the GUI, the IR Engine(s), the IR Editor(s) and the IRDB. The controller provides separate APIs for managing: (i) the available IR Engines, (ii) the IR Editors used by each user, (iii) playbook modifications, (iv) logging capabilities, (v) playbook availability and execution status, (vi) metrics and KPIs. The IR Controller can be executed on a dedicated host, isolated from the rest of the platform's components. User privilege management is offered in combination with KrakenD[3] and Keycloack[4].
- **IR Engine:** A sand-boxed Node-RED[5] instance that hosts the CACAO security playbook engine module that serves as the platform's playbook execution engine. The instance's native Web GUI is disabled, and no playbook modifications are allowed for playbooks running in this instance. The engine module is developed using NodeJS, HTML, CSS, and Jolt Specs[6]. A ROAR instance can utilize more than one engine for redundancy, isolation, testing or scalability. Engine instances can be managed through the IR Web GUI or directly through the respective Engine REST API.
- **IR Editor:** A sand-boxed Node-RED instance that hosts the CACAO security playbook engine module that serves as the modification space for new or existing playbooks. Playbooks in this instance can only be modified and cannot be executed through this instance. IR Editor instances are unique per user. Each Editor instance is created on request and is discarded after use. The engine module found in these instances also contains the translation node responsible for transforming CACAO playbooks to Node-RED flows. Editor instances can be managed through the IR Web GUI or directly through the respective Editor REST API.
- **IRDB:** A PostgreSQL database that stores all the required information for playbook execution (e.g., metadata, logging from playbook executions) and storage. Essentially, this is the main database that stores all Resilience Playbooks (see Section 3 below).

Further, Figure 6 shows the ROAR architecture in the case of deployment in an actual system (e.g., the use case testbeds of PHOENI2X) and the connectivity with the external components needed to allow the execution of the orchestration processes encoded within the RPs.

[2] https://spring.io/
[3] https://www.krakend.io/
[4] https://www.keycloak.org/
[5] https://nodered.org/
[6] https://github.com/bazaarvoice/jolt

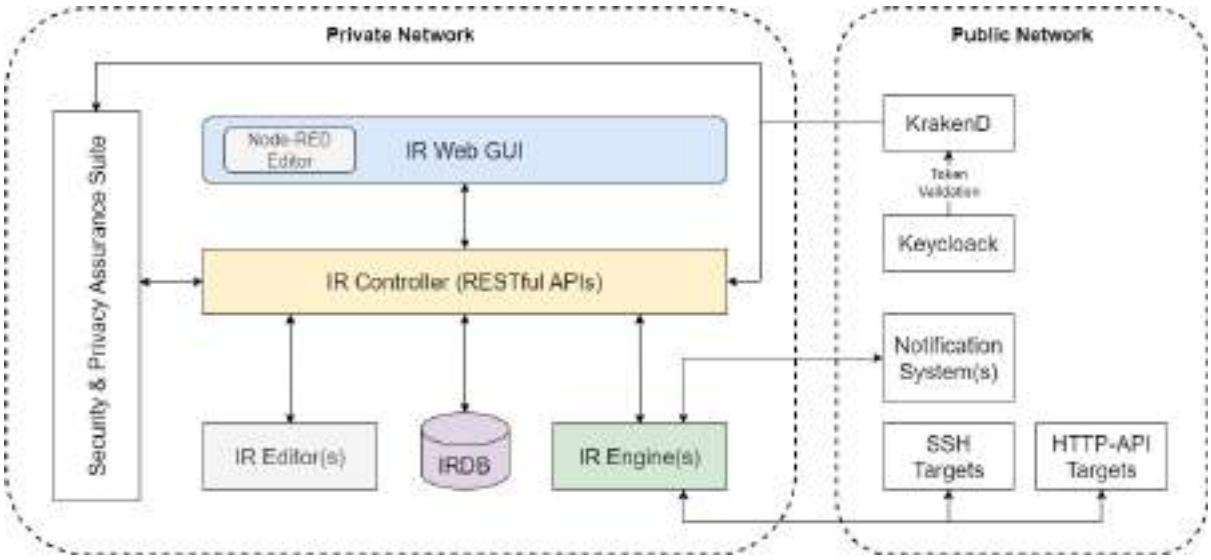


Figure 6. ROAR detailed (internal) architecture & connectivity with external components.

More screenshots and details on the use of ROAR are provided in the above-noted section dedicated to RPs (unavoidably also covering the overall handling of playbooks within ROAR), so they will not be repeated here for the sake of brevity.

Nevertheless, an overview of the process of importing & exporting RPs in the CACAO format on the ROAR tool is shown in Figure 7 and Figure 8, respectively.

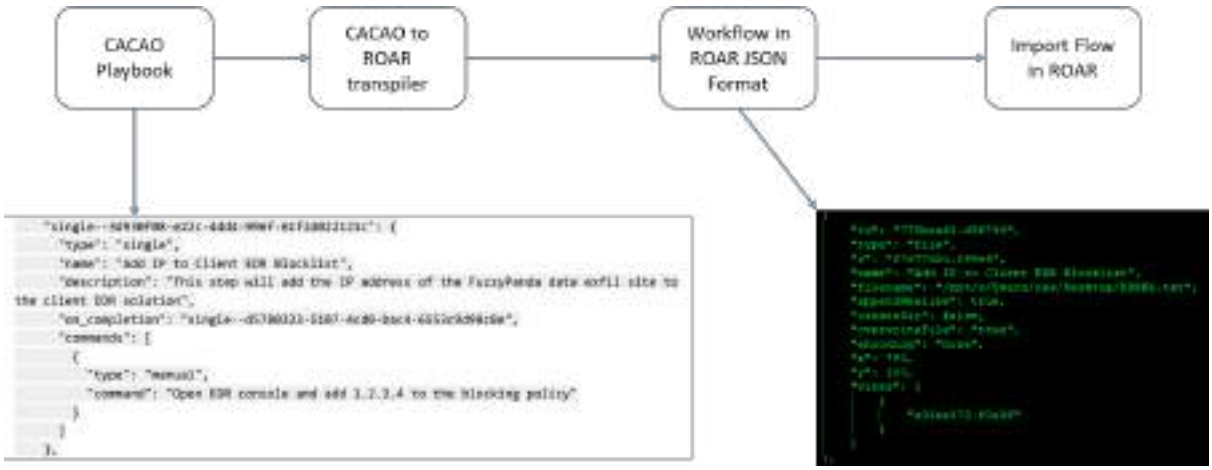


Figure 7. Process of importing CACAO playbooks into the ROAR tool (e.g., for execution and/or editing/customisation).

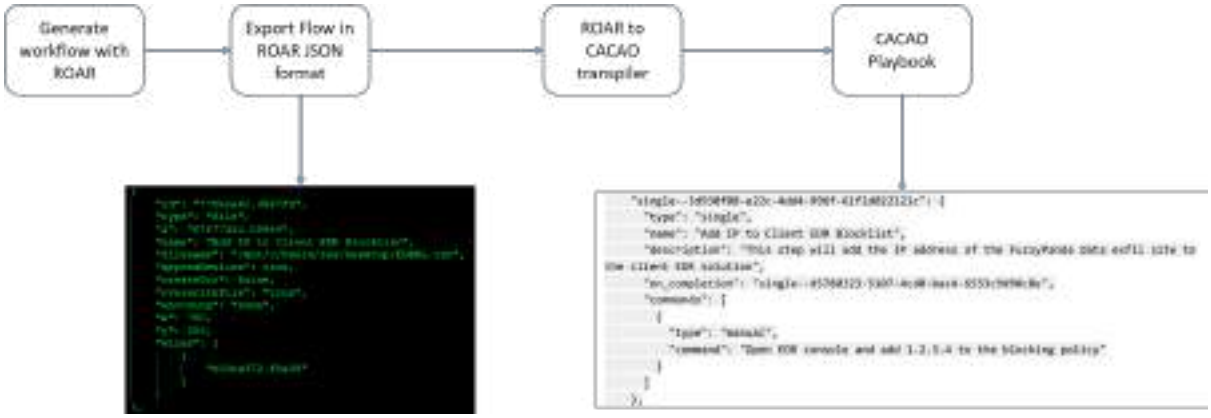


Figure 8. Process of exporting CACAO playbooks from the ROAR tool.

Further, additional interfacing and reporting capabilities were built into ROAR to facilitate the integration with the rest of the PHOENI2X components, not just for integrations that support the execution of Playbook actions, but also at the front-end level. An initial integration of ROAR with FVT is presented in Figure 9 bellow.

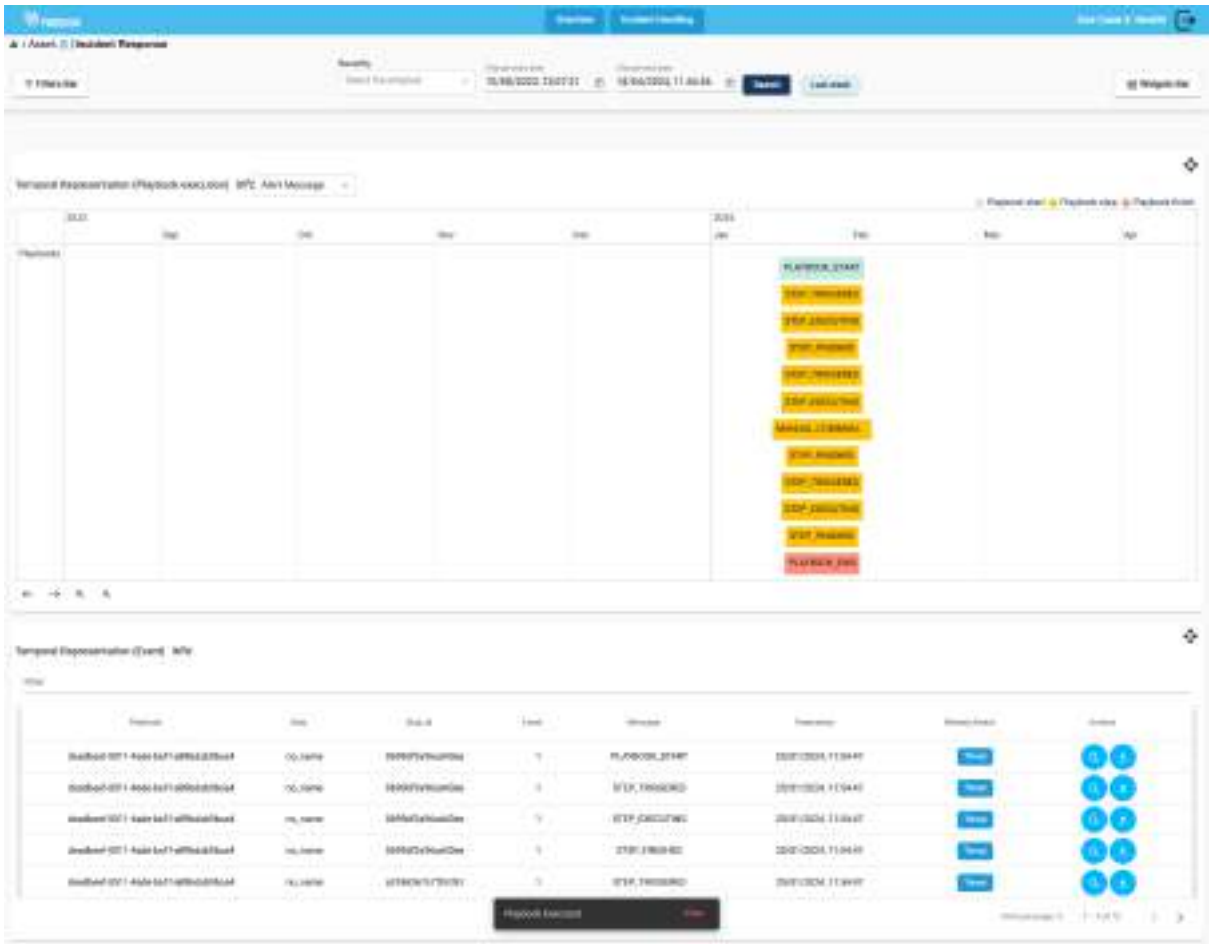


Figure 9. Integration of ROAR actions onto the FVT.

2.4 Recap of Current Status & Next Steps

The efforts in the first cycle of project focused on adapting & extending the underlying SPHYNX IR tool to support the full set of capabilities envisioned for the ROAR component of PHOENI2X and the relevant interaction & integrations (e.g., specification & Execution of baseline set of RPs, basic

integration with other components, such as SPA, CTI, SIEM & other baseline toolset components, the exact interactions & associated interactions mechanisms with other PHOENI2X components). The execution capabilities in particular focused on supporting the execution of the RPs (IR-focused RPs, mainly) for the first demonstrators of the project across all 3 use cases, as well as providing the foundations for the extended capabilities that will be pursued during the second cycle of the project.

Considering the latter, the focus will now shift to the support of additional types of RPs, but also mainly of the novel types of playbooks introduced in PHOENI2X, i.e., the Business Continuity playbooks. Further additional integrations with external systems will be pursued, to be showcased in the (even more complex) final demonstrators of PHOENI2X.

3 Resilience Playbooks Specification, Translation & Lifecycle Management

3.1 Overview

Fundamental to the PHOENI2X concept and the framework's ROAR capabilities are the Resilience Playbooks (RPs; developed in WP4, Task 4.2). RPs provide a structured, machine-processable encoding of a sequence of actions comprising the organisation's business continuity, recovery, and incident response operations. Each action represents a fundamental activity (e.g., add a rule to a firewall). Thus, through RPs, organisations will be able to specify, automate the execution (via the purpose-built execution and orchestration engine), monitor the progress, and assess the effectiveness of all their business continuity, recovery, and incident response-related processes.

Furthermore, RPs will be:

i. adaptable and contextualizable using inputs from the AI-assisted Situational Awareness, Prediction & Response capabilities of PHOENI2X and post-incident analyses;

ii. customisable to the intrinsic regulatory requirements, policies, and IT and OT environment of each OES;

iii. shareable across organisation boundaries at machine-speed, to augment the preparedness and coordinated response capabilities of all actors (see also 1.2.1.2.5);

iv. supporting what-if analyses, with the specification of orchestrations involving components and capabilities that are not (yet) present in the organisation, to drive decision making; and

v. translatable to support assessment and training in a realistic, simulated/emulated cyber range environment.

The placement of the playbooks and the associated databases in the overall PHOENI2X architecture is shown in Figure 10 & Figure 11. Some more details and the current implementation status are provided in the subsections that follow.

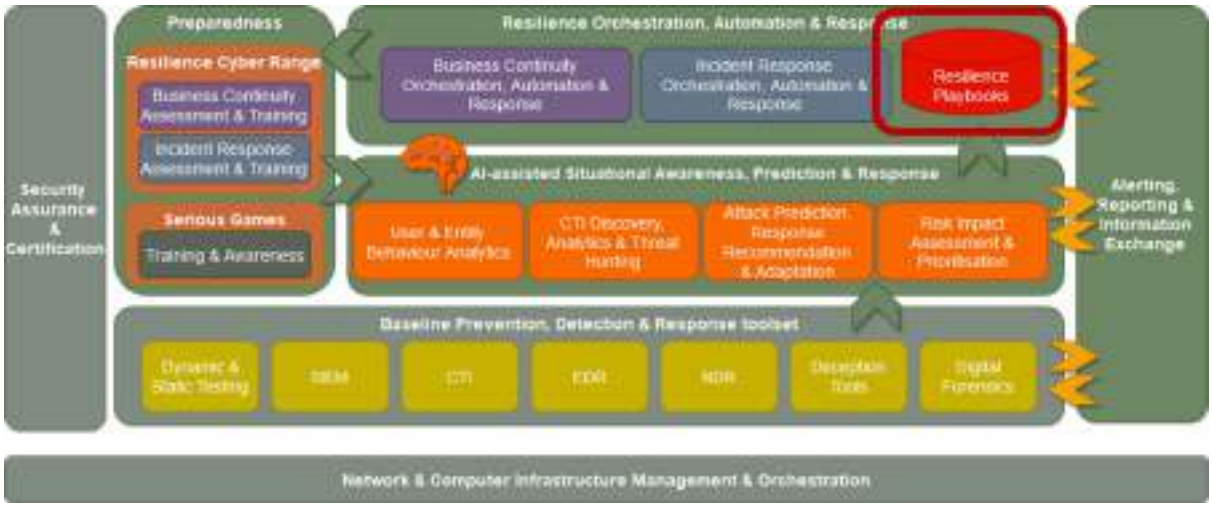


Figure 10. The RPs in the high-level PHOENI2X architecture.

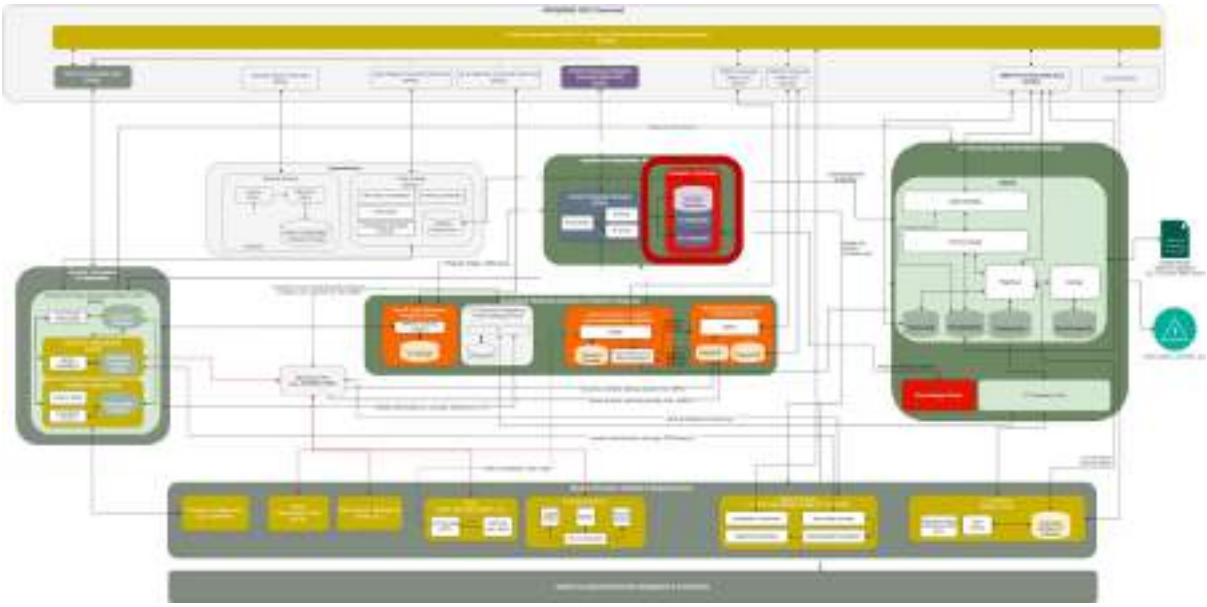


Figure 11. The RPs in the detailed PHOENI2X architecture.

3.2 Design Details

To achieve the above, RPs in PHOENI2X will adopt and extend the recently-released OASIS Collaborative Automated Course of Action Operations (CACAO) standard ([1]– v2.0 of the specification approved in Nov. 2023).

CACAO is a cybersecurity-specific schema and taxonomy for creating, documenting, and sharing playbooks in a structured and standardized format across organizational boundaries and technological solutions. Figure 12 illustrates the composition of a CACAO playbook. Briefly, a CACAO playbook comprises metadata, workflow steps with control logic, a set of commands to be performed, targets and agents that perform the commands, data markings that specify the playbook's handling and sharing requirements, and extensions that introduce functionality ad-hoc. For integrity and authenticity, CACAO playbooks can be digitally signed and countersigned. The signature design supports incorporating the signature in the playbook or releasing it separately as a detached signature.

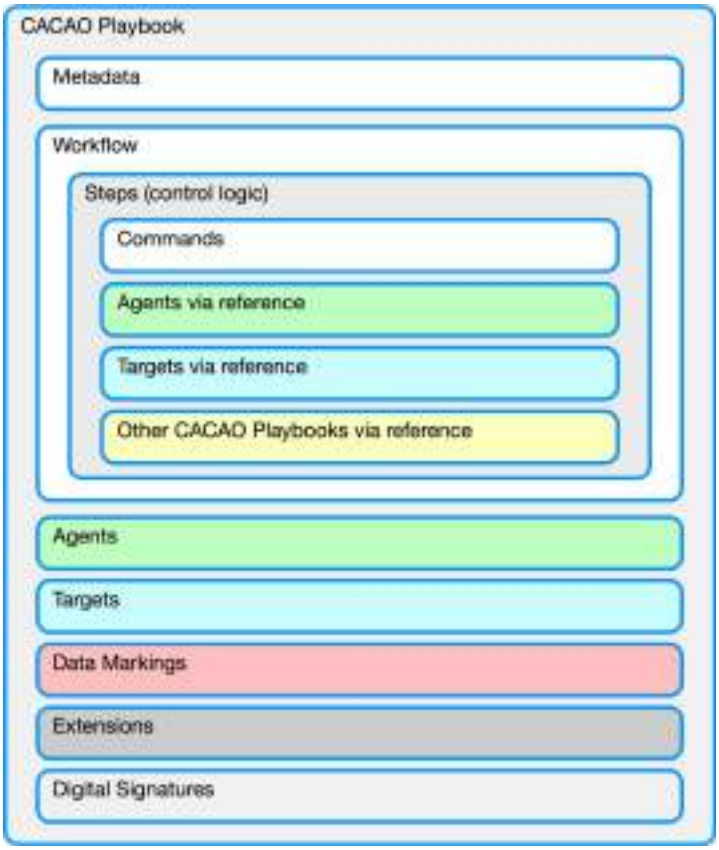


Figure 12. CACAO playbook structure (source: OASIS)

CACAO defines various playbook types to support different cybersecurity-related operational roles and functions, including notification (proposed and developed in the context of PHOENI2X), detection, investigation, prevention, mitigation, remediation, attack, and engagement. In PHOENI2X, specifically, to cover all business continuity, recovery, and incident response aspects and the corresponding PHOENI2X prevention, preparedness, response and recovery capabilities, the following types of playbooks will be supported and specified:

- **Business Continuity Playbooks**, focusing on business processes and maintaining required service levels:
    - **Proactive Business Continuity Plan (pBCP) Playbooks:** Encoding orchestrations of proactive actions (e.g., triggered by predictive maintenance or attack prediction) to ensure uninterrupted service offering in line with the organisation's requirements (e.g., SLAs).
    - **Reactive Business Continuity Plan (rBCP) Playbooks:** Encoding orchestration of reactive actions to detected incidents (e.g., on-going attack or system failure), aiming to ensure that business processes are not disrupted.
    - **Disaster Recovery Plan (DRP) Playbooks:** Focused on defining the orchestration steps required to recover from disruption of business processes (e.g., because rBCPs were not effective).
- **Incident Response Playbooks**, focusing on the technical side of cyber resilience:
    - **Prevention Playbooks:** Specify orchestration actions to prevent a known or expected security event, incident, or threat from occurring (e.g., implement best practices, counter suspicious activity).

- **Security Assessment Playbooks:** Encode orchestration steps required to execute security assessments on the protected infrastructure (e.g., penetration testing, vulnerability scanning, security controls test).
- **Detection Playbooks:** Specify orchestration actions to detect known or expected security activity (e.g., based on event received from early warning systems) or for threat hunting (e.g., via CTI indicators).
- **Mitigation Playbooks:** Focused on orchestration steps required to mitigate the effects of a security event or incident that has occurred, when remediation is not initially possible (e.g., isolate host from network, until root cause can be fixed).
- **Remediation Playbooks:** Encoding orchestration of steps to remediate, resolve, or fix the results of an incident, returning the system back to a good operating state.
- **Investigation Playbooks:** Focused on encoding post-incident analysis orchestration steps, to investigate how and why the incident took place and what its effects were, consequently informing future prevention, preparedness, response, and recovery actions for such incidents.

- **Scenario (What-if analysis) Playbooks**, focusing on the encoding of orchestrations involving capabilities and assets that are not yet part of the organisation, to be assessed via the Cyber Range features of the platform, providing insights about these scenarios and their (cost)effectiveness, to inform decision making.
  - **Business Continuity Scenario (BCS) Playbooks:** Encoding business continuity orchestrations that include hypothetical capabilities and assets of the organisation (e.g., a planned new business process with the assets needed to support it), to be executed and assessed on the framework's Cyber Range.
  - **Incident Response Scenario (IRS) Playbooks:** Encoding incident response orchestrations that include hypothetical capabilities and assets of the organisation (e.g., a new firewall), to be executed and assessed on the framework's Cyber Range.
- **Alerting, Reporting & Information Exchange Playbooks**, focused on the notification and reporting (e.g., alerting another PHOENI2XCRC about a security event, generation of report to be sent to National Authority) as well as the sharing of information (e.g., anything from IOCs, to RPs, and relevant AI models). For example, an Alerting, Reporting & Information Exchange Playbook can be triggered by another playbook to notify stakeholders within and across borders to inform about the attack and also append the needed RPs to promptly detect and mitigate it. These are "global" playbooks, in the sense that they will typically be triggered and/or run alongside other types of RPs, and their ingredients (e.g., an alerting step) will also be usable within other playbooks as well.

The above types of playbooks, in relation to the incident occurrence timeline, are shown in Figure 13.

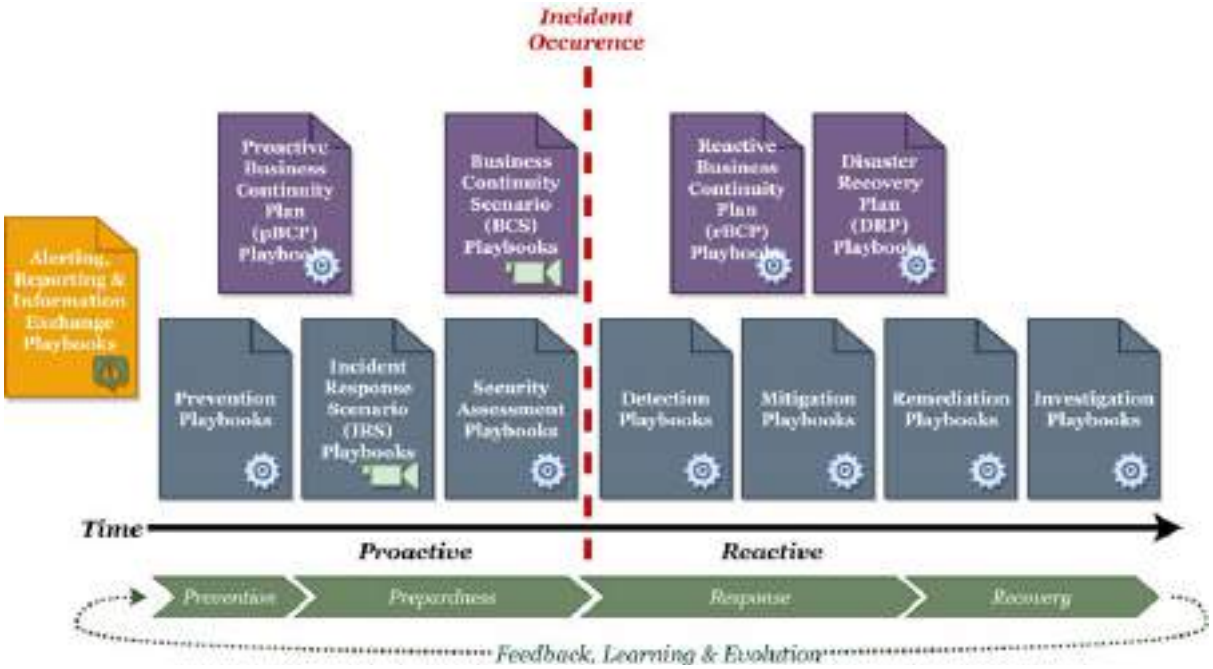


Figure 13. Different types of Resilience Playbooks envisioned in PHOENI2X.

It is worth mentioning that the consortium has developed and open sourced JSON validation schemas for CACAO Version 2.0. The validation schemas are hosted at the official GitHub[7] of the OASIS CACAO technical committee and are considered the de facto validation mechanism for all implementations.

Similarly, through our implementation we identified that the way we design and visualize playbooks inherently supports understandability and readability. However, the CACAO specification does not provide any standard approach in graphically representing playbooks. As a result sharing and importing CACAO playbooks across solutions leads to inconsistent visualization. On that regard, the consortium led the development of a new technical specification within the OASIS CACAO technical committee, namely, the CACAO Layout Extension. The Layout Extension allows to represent CACAO playbooks accurately and consistently across implementations. The technical specification has undergone through a public review[8] and is currently in the process of being approved as a committee specification. Design and implementation details about the CACAO layout extension can be found in the technical specification[9].

3.3 Implementation Details

The Resilience Playbooks in PHOENI2X are designed within the ROAR tool (detailed in Section 8 above), which is the main means of managing the RPs throughout their lifecycle (e.g., covering their creation, editing and execution).

3.3.1 Supported CACAO Playbook Steps

ROAR's current version enables the design and execution of CACAO security playbooks following the v1.1 CACAO specification. Accordingly, such playbooks can be composed of the following seven steps: (i) start, (ii) end, (iii) single, (iv) parallel, (v) if, (vi) switch, and (vii) while step. The playbook step which allows branching a playbook with other playbooks will be supported in the next version of ROAR and presented in D4.2

[7] https://github.com/oasis-open/cacao-json-schemas
[8] https://www.oasis-open.org/2024/01/25/invitation-to-comment-on-cacao-layout-extension-v1-0/
[9] https://docs.oasis-open.org/cacao/layout-extension/v1.0/csd01/layout-extension-v1.0-csd01.html

The start and end steps indicate the starting and ending point of each playbook and thus are always required. The single steps are responsible for executing the actions indicated at each execution point. These actions can be shell commands, execution of REST API calls or requests to be executed manually by a human operator. The parallel steps indicate the execution of a set of steps in parallel while the if and switch commands designate branching based on a condition. Finally, the while steps indicate looping based on a condition. The steps are linked together, starting from the start step and ending at the end step, thus composing the playbook's logic as a graph with each step being a node. ROAR also provides an additional step, dedicated to translating CACAO playbooks to IR Engine executable playbook flows in real-time, named translate. In the Engine, each step is available as a graph node, that can be dragged and dropped on the canvas in order to design or edit a playbook.

The list of available steps is presented in Figure 14.

Figure 14. CACAO playbook steps available as drag-and-drop enabled graph nodes within the ROAR GUI.

3.3.2 Playbook Design

Designing a playbook is performed in a drag-and-drop fashion, starting with placing a start and an end step on the canvas. Then, the playbook's logic is created by placing, single and conditional steps on the canvas and linking them together with edges connecting the various steps. The execution follows the edges from the start step to the end step and branches according to the specified conditional steps and the evaluation of each condition. An example playbook containing three single steps that will be executed in sequence is presented in Figure 15.

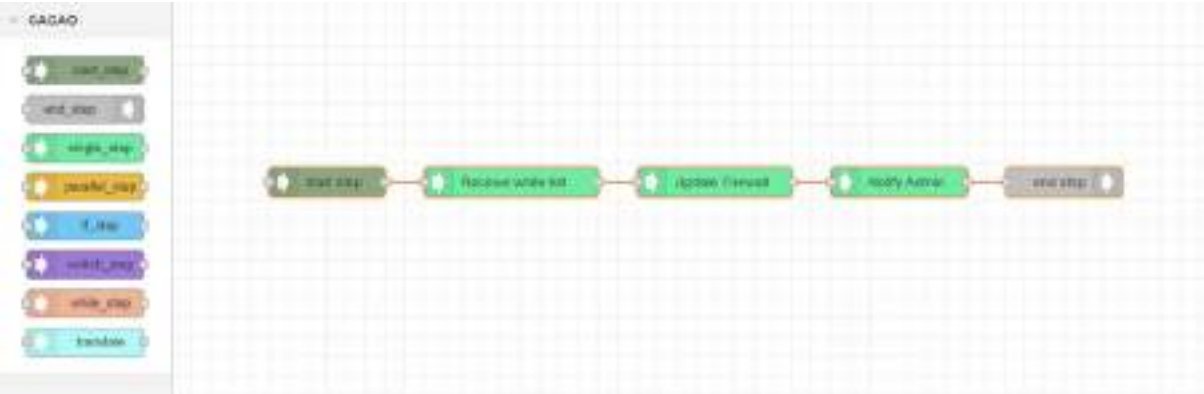


Figure 15. A simple playbook performing three operations.

The execution starts from the leftmost node (the start step) and follows the edge towards the first single step, named “Receive white list”. This step is responsible for performing the appropriate action(s) and command(s) that will retrieve the specified IP whitelist in this example. Steps can use variables to write and read data and these variables can be passed to the following steps while they are executed. Once this step is finished, the execution follows the edge towards the next single step named “Update Firewall”. This step is now responsible to update the firewall with the previously retrieved white list, utilizing the variables defined in the previous step. Then, the playbook executes the “Notify Admin” single step, responsible to notify the network administrator about the operations performed by the playbook. The execution terminates upon reaching the end step.

This simple playbook can be executed by the Engine or exported as a CACAO JSON file to be shared with other parties. While the playbook’s parameters like commands and agents and targets might require tuning based on the target system, the playbook’s process is preserved when sharing the playbook. For example, this playbook handles the logical operation of retrieving an IP white-list, updating the firewall with the white-list, and then notifying the administrator. However, the commands that have to be executed to perform these actions might need to be modified depending on the target system. For example, the playbook might need to retrieve the whitelist from a different location, update a different type of firewall system and notify a different administrator when it is utilized by another organisation. The following section is dedicated on step properties and playbook tuning.

3.3.3 Step Properties’ Definition

Each CACAO playbook step has a set of properties that dictate its operation. While some of these properties can be found across all steps (such as name, delay, timeout timer, etc.), each step has a set of unique properties depending on its type. These properties can be tuned using the step’s parameter form that appears by double clicking on a step placed on the canvas. In this section, we describe the parameters available for each step and how the steps can be linked together.

3.3.3.1 Common properties

Each playbook step has a *name*, and *description* that can be modified through their parameter forms. The name is also displayed on the node once modified. Also, every step has a *delay* and *timeout* counter that can be set using positive integer values. No value and 0 indicate that the counter is not used while the rest of the values indicate milliseconds. The delay counter specifies how many milliseconds the step should wait before executing once the execution reaches this point. The timeout counter indicates the time available for a step to execute. If the step’s execution is not finished within this time frame, the execution proceeds with invoking a step as a result of a failure (different sequence in the playbook process) and an error is generated.

Also, each step can be tuned to utilize separate execution paths for error handling. By default, every step provides an output point that is used to link it with the next step, except for the *end* step. On this output point, the steps provide messages about their completion status. When the *on_completion* flag is selected, success, warning and error messages will be forwarded by this single output point. If the "*on_success & on failure*" flag is selected, the step will provide two output points with the first being the successful completion channel and the second being the error handling channel. In this case, the successful execution will proceed following the first output point while in case an error is generated, the execution will proceed following the second output point.

3.3.3.2 Start Step properties

The *start* step's parameter form is used to specify the parameters affecting the step as well as the global parameters affecting the entire playbook. This list of parameters is presented in Figure 16, while the global parameters and their descriptions are provided in Table 1.

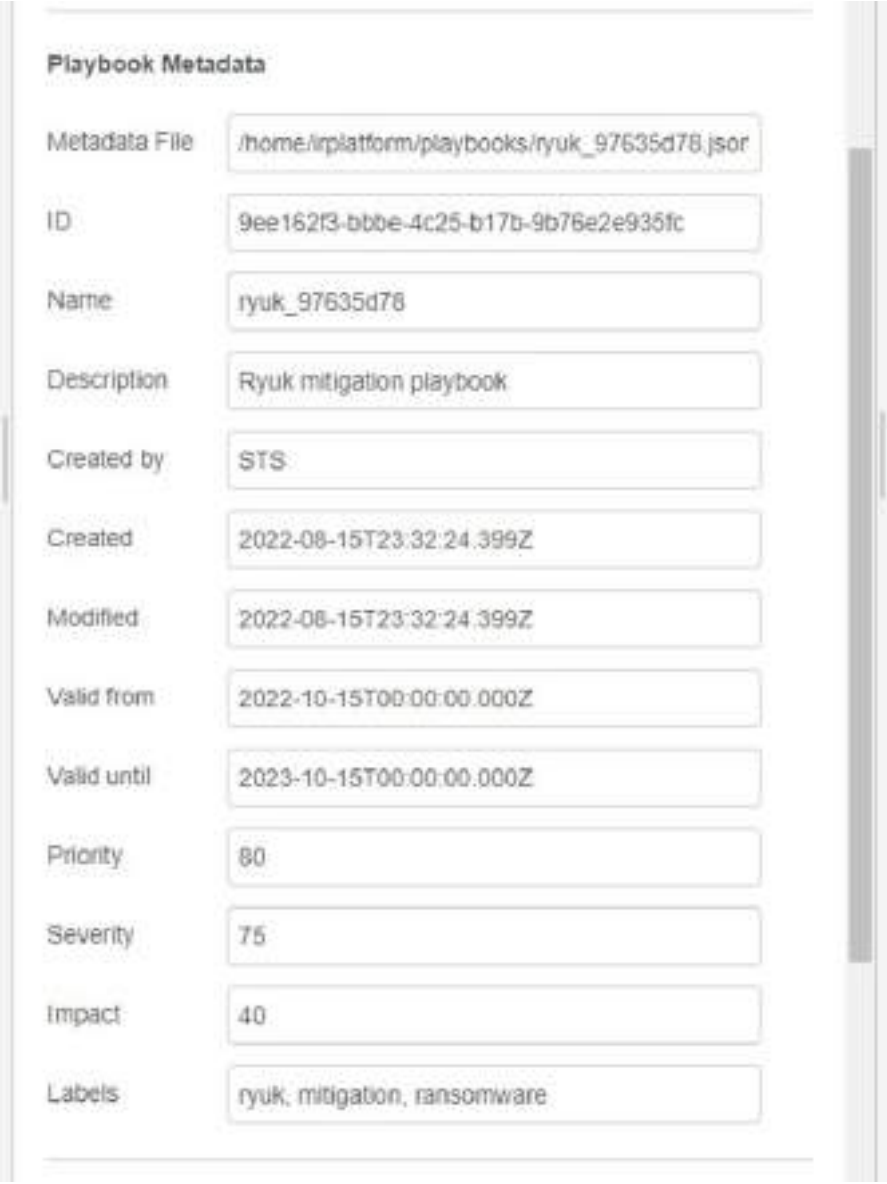


Figure 16. Global playbook metadata, modifiable via the start step's parameter form.

Table 1. Global properties of a playbook defined through the start step

Parameter	Description
Metadata file	Path to a JSON file providing the rest of the parameters
ID	The playbook UUID
Name	The playbook name
Description	The playbook free-text description
Created by	The creators free-text name
Created	Playbook creation date (yyyy-mm-ddThh:mm:ss[.s+]Z)
Modified	Last modification date (yyyy-mm-ddThh:mm:ss[.s+]Z)
Valid from	The date from which the playbook is valid (yyyy-mm-ddThh:mm:ss[.s+]Z)
Valid until	The date after which the playbook is invalid (yyyy-mm-ddThh:mm:ss[.s+]Z)
Priority	A positive integer that represents the priority of this playbook relative to other defined playbooks (1-100)
Severity	A positive integer that represents the seriousness of the conditions that this playbook addresses (1-100)
Impact	A positive integer that represents the impact the playbook has on the organization (1-100)
Labels	A set of free-text labels describing the playbook

As shown above, the addition of metadata relevant to the reason of the playbook execution is also considered per playbook type, to allow the mapping of playbooks executed to threats from the CTI platform.

Finally, the *start* step's parameter form provides a simple JSON editor where global playbook variables can be defined and later used by the playbooks steps and the option to designate a file path where alternatively these variables can be loaded from. An example of the *start* step's parameter form is presented in Figure 17.

Figure 17. The start step's global playbook variables editor.

3.3.3.3 End Step Properties

The *end* indicates the playbook's (or branches) end and thus provides limited functionality. The only parameters that can be tuned are its *name* and *delay* counter.

3.3.3.4 Single Step Properties

The *single* step provides a "Commands" editor where a single or multiple commands can be specified using the "add" button. The commands will be executed top-down in the order they appear (sequentially). Each error generated when executing a command is forwarded through the output point(s) as specified by the "Outputs" flag described above. The available command types are (i) manual, (ii) bash, (ii) ssh and (iv) HTTP-API.

A *manual* command will notify a human operator with a free-text message which can be modified via the dedicated input box. The notification medium can be either e-mail or a Slack channel. The user can specify a single or multiple e-mail addresses, a Slack channel from the list of available channels or specify a Slack channel using its web-hook.

A bash command can be any arbitrary bash command and will be executed locally, on the machine hosting the engine. An ssh command can be any arbitrary bash command and will be executed remotely, on the host specified by the "SSH Target" section, as presented in Figure 18. In this section, the user can designate the remote host's IP address, port, username, password, and/or path to SSH key. Finally, the HTTP-API command can execute GET and POST requests.

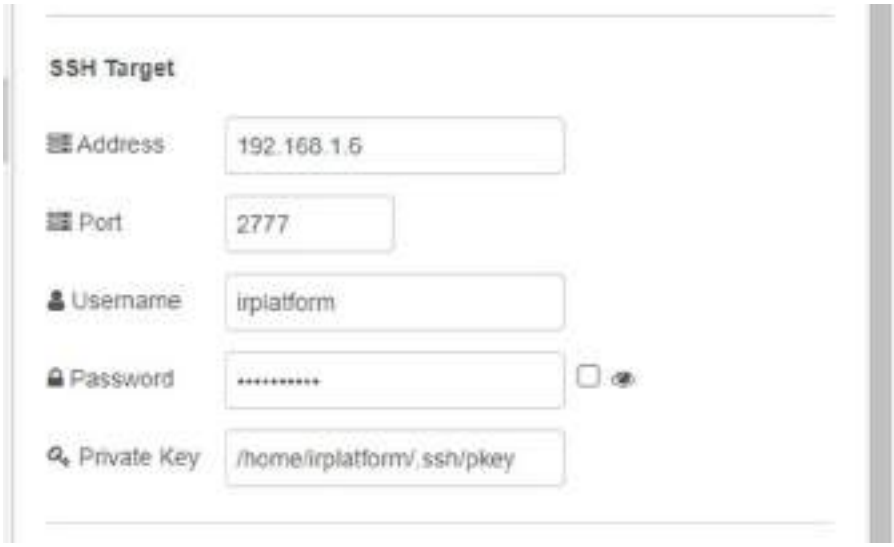


Figure 18. Remote target selection form.

It should be noted that the above reflect ROAR's Single Step capabilities (& associated properties) at the time of writing, but these are expected to be enriched throughout the duration of the project, in order to support additional integrations & more complex interactions.

3.3.3.5 Parallel Step Properties

The *parallel* step provides two groups of output points. The first point is used to connect the steps that will be executed in parallel. Once these steps finish, the execution returns to the parallel step and proceeds through the output point(s) depending on the value of the "Outputs" flag. An example of a *parallel* step in action is presented in Figure 19.

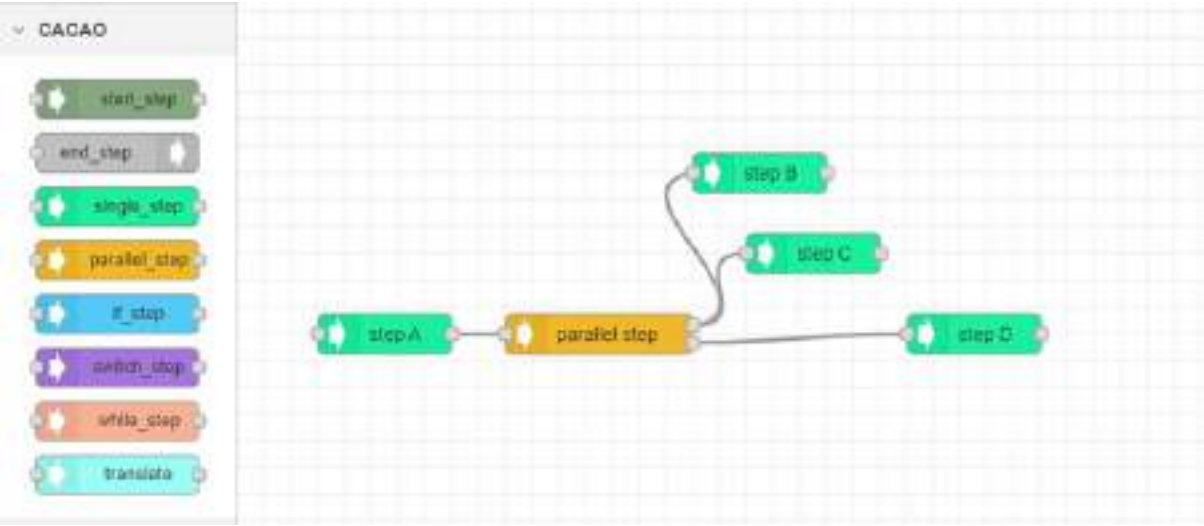


Figure 19. A parallel step executing two single steps.

Once the execution reaches the *parallel* step from "step A", it proceeds with steps "step B" and "step C" which will be executed in parallel. When both steps finish, the execution returns to the *parallel* step that hands over the execution to "step D". The only tuneable parameters are the *name, description, delay,* and *timeout.*

3.3.3.6 If Step Properties

The *if* step provides three groups of output points. The first output point is used to redirect the execution if the evaluated condition is *true* while the second point is used to redirect the execution if the evaluated condition is *false*. Once the *true* or *false* branches finish, the execution returns to the *if* step and is redirected through the last output point(s). An example of an *if* step in action is presented in Figure 20.

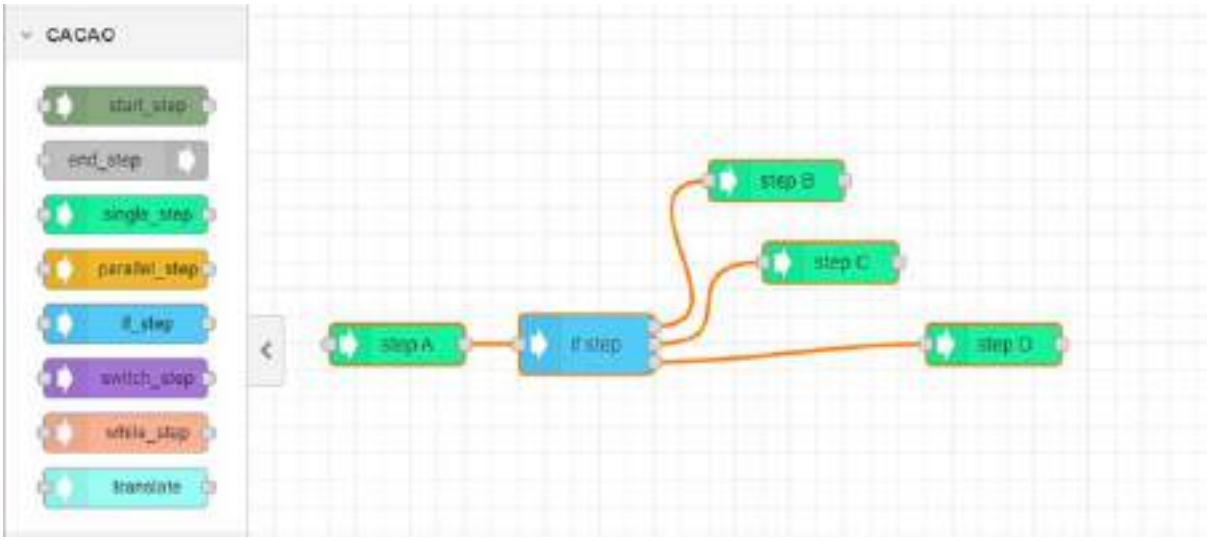


Figure 20. An if step linked with two different single steps to be executed based on the condition's evaluation.

Once the execution reaches the *if* step from "step A", the execution will be redirected to "step B" if the condition is *true* or to "step C" if the condition is *false*. When the selected branch finishes, the execution returns to the *if* step that hands over the execution to "step D". Except for the common parameters, the *if* parameter form allows the definition of the condition.

3.3.3.7 Switch Step Properties

The *switch* step provides two groups of output points. The first output point group is used to redirect the execution based on which case is evaluated as *true* and contains as many output points as the number of cases with the minimum being one. Once the selected branch of steps finishes, the execution returns to the *switch* step and is redirected through the last output point(s). An example of a switch step in action is presented in Figure 21.

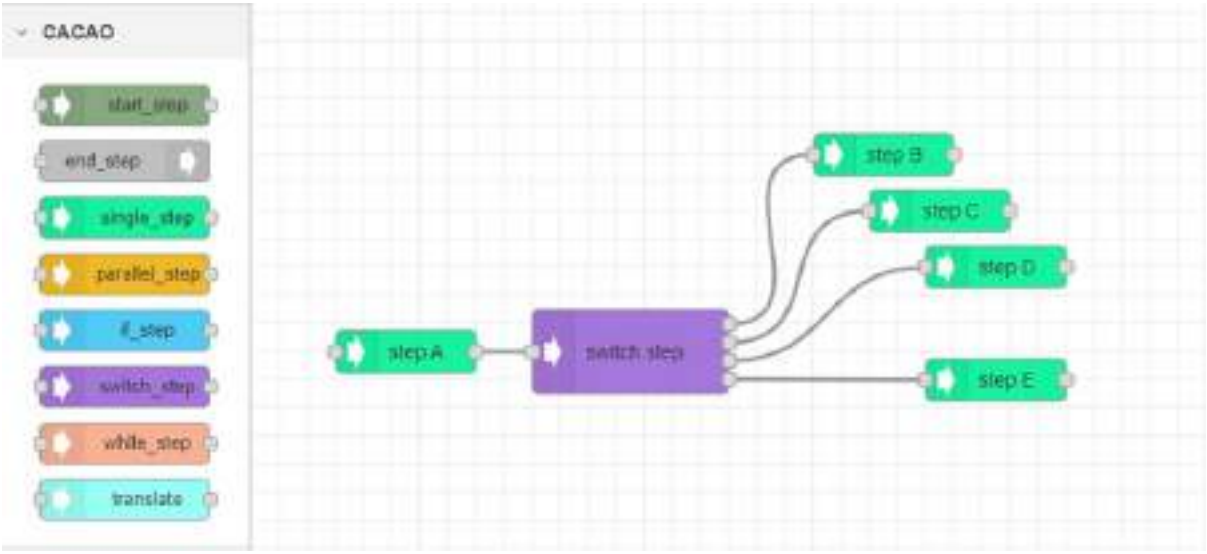


Figure 21. A switch step with three cases.

Once the execution reaches the *switch* step from “step A”, the execution will be redirected to “step B” if the condition is *true* for the first case, “step C” if the condition is *true* for the second case or to “step D” for the last case respectively. When the selected branch finishes, the execution returns to the *switch* step which hands over the execution to “step E”. Except for the common parameters, the *switch* parameter form allows the definition of the condition as well as the addition of switch cases. Each new case automatically generates an output point that can be used to redirect the execution.

3.3.3.8 While Step Properties

The *while* step provides three groups of output points. The first output point is used to redirect the execution if the evaluated condition is *true* while the second point is used to redirect the execution if the evaluated condition is *false*. Once the *true* or *false* branches finish, the execution returns to the *while* step and is redirected through the last output point(s). An example of a *while* step in action is presented in Figure 22.

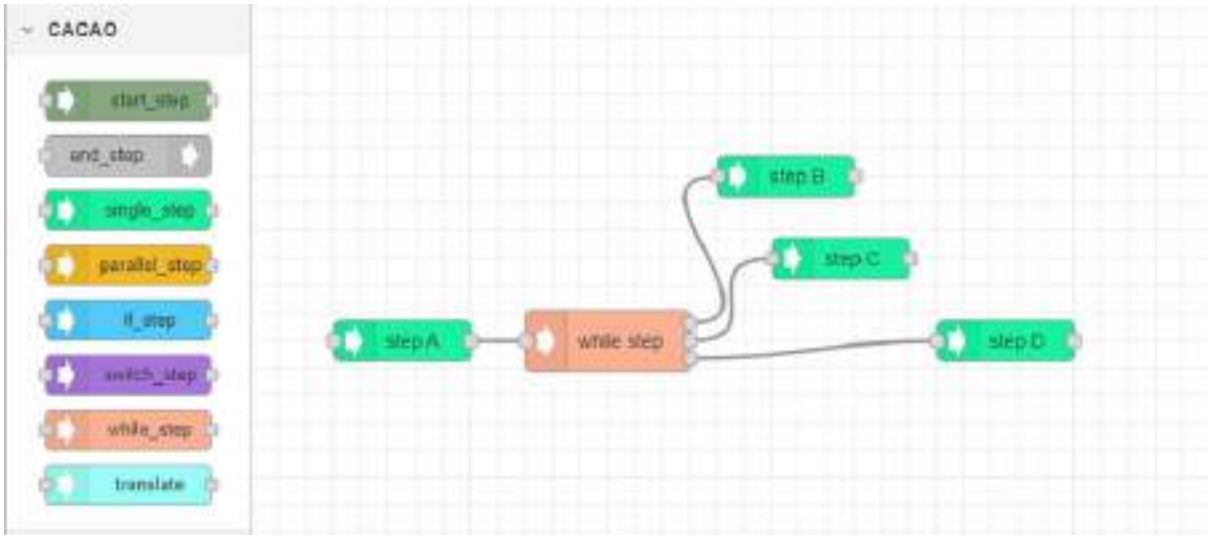


Figure 22. A while step connected with a step that executes every time the condition is true and a step that executes once the loop breaks.

Once the execution reaches the *while* step from "step A", the execution will be redirected to "step B" if the condition is *true* or to "step C" if the condition is *false*. When the selected branch finishes, the execution returns to the *while* step which hands over the execution to "step D". Except for the common parameters, *while* parameter form allows the definition of the condition.

3.4 Recap of Current Status & Next Steps

In terms of current status, for the first cycle of the project the emphasis has been on the delivery of Incident Response playbooks to support the use cases and the relevant demonstrators, with some preliminary work on the more novel application of playbooks for Business Continuity, while for the second cycle additional IR playbooks will be designed and tested, while also delivering the required BC-focused & scenario playbooks as well.

More specifically, the definition of the initial set of incident response playbooks, covering each of the core categories, i.e. (i) Prevention, (ii) Security Assessment; (iv) Detection; (v) Mitigation; (vi) Remediation; and (vii) Investigation, took place during the first cycle. Further, these were demonstrated & validated, as they were tailored to the demonstration scenarios of each of the project's use case demonstration scenarios (e.g., see Figure 23), as part of the assessment of the first integrated version of the PHOENI2X framework.

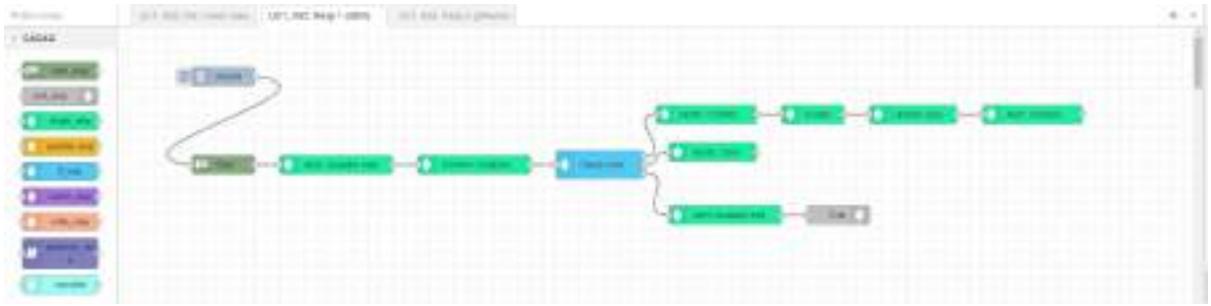


Figure 23. IR playbook used in Energy Use Case (UC1) to isolate attacker via SDN Controller.

Future work in terms of IR Playbooks includes the definition of the final set of incident response playbooks, covering all core IR playbook categories shown in Figure 13, as well as the demonstration & validation of IR-focused playbooks covering all final demonstrator scenarios, as part of the final demonstration & validation activities of the integrated PHOENI2X framework.

Considering the BC-focused playbooks, work in the first cycle of the project includes the identification of business continuity intricacies and requirements for each of the use case environments, through dedicated business continuity workshops. These activities supported the development of the initial high-level BC plan activation playbook (see Figure 24), based on established standards (namely following ISO 22301[10] for the overall process, 27035[11] for incident management & 22317[12] for the Business Impact Assessment), forming the baseline for the use case -specific playbooks.

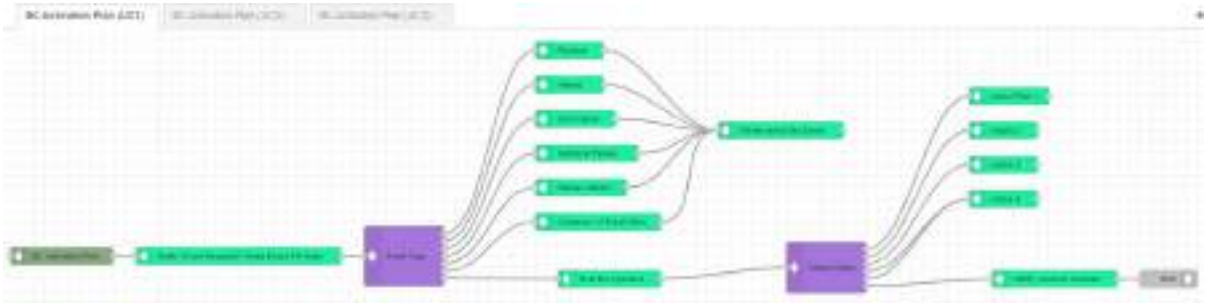


Figure 24. High-level BC process playbook

Future work in terms of BC Playbooks will focus on defining the final set of business continuity playbooks, covering the intricacies and requirements of each of the individual use case environments of PHOENI2X, and the four different types of playbooks supported (as shown in Figure 13). These will be demonstrated & validated (leveraging the Resilience Orchestration, Automation & Response - ROAR - enabler) in the context of the final integrated PHOENI2X framework deployments and associated use case demonstrators.

The final results of the above efforts will be documented in the follow-up to this deliverable, i.e. deliverable D4.2 - “Coordinated Response & Preparedness Enablers v2”.

[10] https://www.iso.org/standard/75106.html
[11] https://www.iso.org/standard/78973.html
[12] https://www.iso.org/standard/79000.html

4 Cyber Range & Serious Games for Resilience Assessment & Training

4.1 The Resilience Cyber Range

4.1.1 Overview

The Resilience Cyber Range (RCR; developed in WP4/Task 4.3) is one of the key Preparedness enablers in the PHOENI2X concept & overall vision, offering realistic, hands-on training in incident response, business continuity and other cybersecurity training & awareness –related scenarios.

In more detail, and as outlined already at the proposal stage, the RCR will support OES preparedness in two ways:

i. It will enable the assessment of defined RPs of all types in a realistic environment that emulates and simulates, as needed, the involved cyber systems and their interactions (e.g., to find gaps or inefficiencies, adapting them as needed);

ii. it will provide hands-on training to OES employees in the business continuity, recovery and incident response procedures encoded in the RPs (e.g., to ensure everyone is accustomed with a new BC approach).

Within (i), the execution of scenario playbooks involving hypothetical assets and tools will be supported, allowing for realistic what-if analyses to inform investment and decision-making.

In (ii) the appropriate trainee assessment mechanisms will need to be integrated, allowing to accurately assess their preparedness. In all cases, some sort of transition from the RPs into the specification that the RCR expects will be needed, including the exact emulated and simulated environment configuration.

The positioning of the RCR within the high-level and detailed PHOENI2X architectures (as presented in D2.1 - "PHOENI2X Requirements & Architecture") are shown in Figure 25 and Figure 26, respectively.

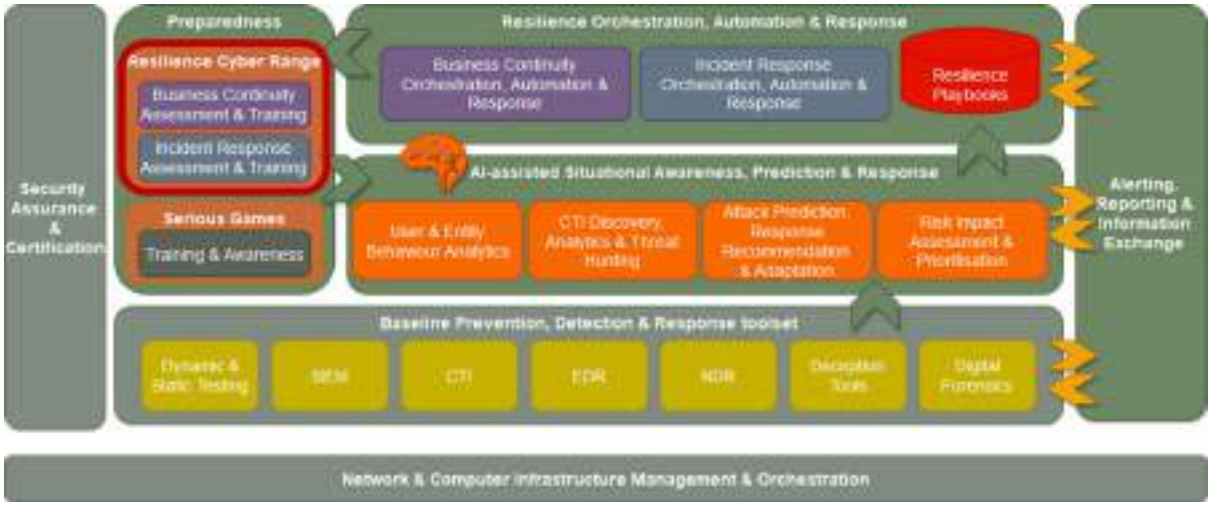


Figure 25. The RCR within the high level PHOENI2X architecture.

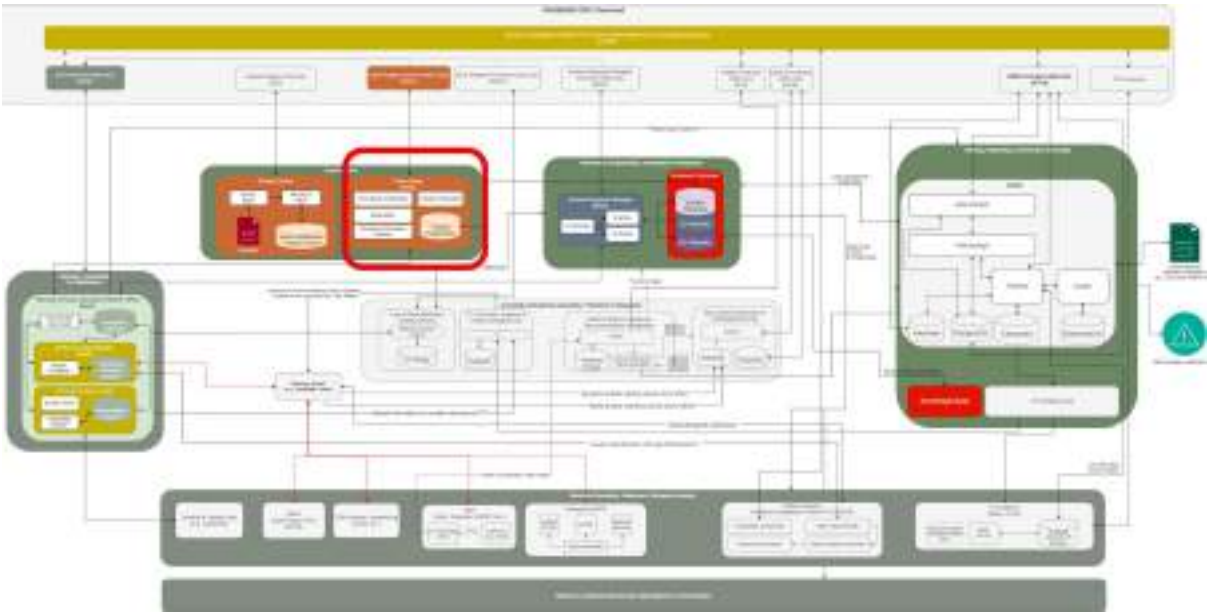


Figure 26. The RCR within the detailed PHOENI2X architecture.

The subsections that follow will provide more details on the current design & associated implementation of the RCR.

4.1.2 Design Details

The RCR will be built upon the SPHYNX Cyber Range platform. The latter is a part of an enhanced version of the SPHYNX Security & Privacy Assurance (SPA) suite, to offer cyber security training that covers a comprehensive spectrum of known and emerging security and privacy threats and is tailored to the particular security and privacy risks of different organisations. Key innovative features of our Cyber Range platform include [2]:

- Delivery of cyber range exercises for different assets (and combinations of assets) of an organisation, and particular types of security and privacy threats, vulnerabilities and risks identified for them;
- Support for asset emulation and simulation at different layers of the implementation stack;
- Model driven customisation of cyber range exercises.

Overall, the Cyber Range provides the necessary emulation & simulation capabilities to deliver training for both specialised (like cybersecurity engineers and programmers) and non-technical roles (such as accountants, secretaries, and managers) within an organization.

In more detail, key features include:

- Definition of training scenarios using SPHYNX's Cyber Range models, backed by the respective relational database schema.
- Automatic deployment of the training scenarios as specified by the models, including:
    - Deployment of assets (e.g., containers, virtual machines and networks).
    - Provision of those assets to accommodate the scenario's needs (e.g., add users, install software, place flags).
    - Deployment of agents that capture events generated by the trainees (e.g., opening of a file).

- Deployment of agents that facilitate cyber-attacks (e.g., run a network exploit, to generate network evidence for the trainee to investigate).

- Training Progression
  - Each step of the training is specified in the training model and is enabled by the FSM (i.e., Finite State Machine) Progression Engine.
  - Actions (i.e., events) of the trainees drive the training progression (e.g., answering a question wrong may trigger a training event).

- User evaluation based on
  - How trainees respond to questions.
  - The time they take to complete the training.
  - Their hint usage.
  - How they interact with the virtual lab.

The Cyber Range accounts for the following, distinct user roles, among which the platform's functionality is split, in the context of an organization using the platform:

- **Organization administrator:** Oversees all organization operations (training, training management and content creation) on the platform.
- **Content creator:** Can create and edit training content on the platform, such as training scenarios, programmes or access the training material library.
- **Training manager:** The training manager can assign training programmes to the trainees on the platform. A training manager can also edit such assignment, such as extending the assignment's due date.
- **Trainer:** A trainer can proctor a training scenario during its execution by a trainee. The trainer can interact with the trainee and the training scenario execution through a scenario management view, where functions such as messaging between trainer and trainee, or injection of hints, are provided.
- **Trainee:** The key target-user of the Sphynx CRP. A trainee can access and train through the programmes that have been assigned to them.

Concerning the latter two user roles, facilities for interacting with a trainer are provided through the training scenario execution view (such as the messaging functionality). The trainee can also view training material served to them during the training scenario's progress, as well as interact with training assets, such as virtual machines through a terminal interface provided in the training scenario view. A trainee can view statistics, in the form of radar graphs or metrics, about the training they have completed thus far on the platform. Training attributes, such as skills, that have been acquired during the user's training are quantified and displayed in a graphical interface.

4.1.3 Implementation Details

The CR platform consists of the following base components:

- **The progression engine**, which tracks the training steps and serves the training material and questions according to the user's training progress. It facilitates the transition between states of the training scenario's progression. During said states, state actions take place, such as questions or the display of training material.

- **The platform core**, consisting of a REST API interface and the platform's database. It provides the base REST APIs for creating, retrieving, updating and deleting programmes, scenarios and user data from the platform.
- **The communication interface**, through which all the platform's components communicate, in a bus-like interface. Data can be fetched and sent to the graphical user interface with a message format, facilitating for a more streamlined and immediate communication interface between the user and the platform's progression engine.

Some indicative screenshots of the key features of the CR are presented below. More specifically, the creation of a training scenario and a key phase in the scenario specification – the definition of the progression model – are shown in Figure 27 and Figure 28, respectively.

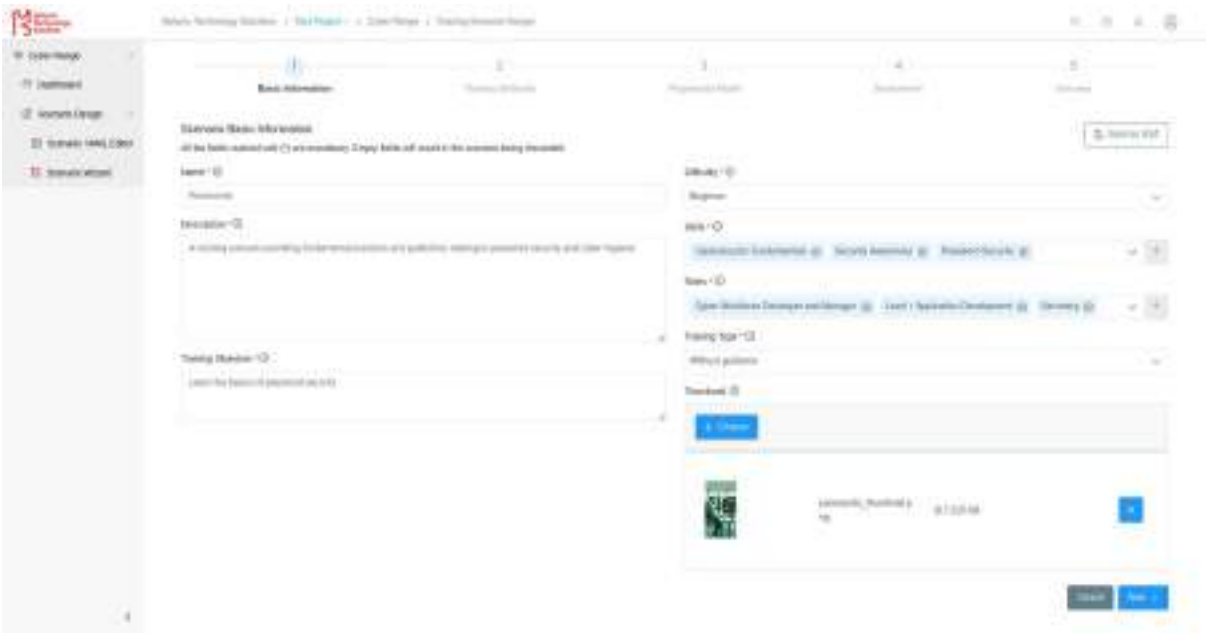


Figure 27. Beginning the specification of a new training programme.

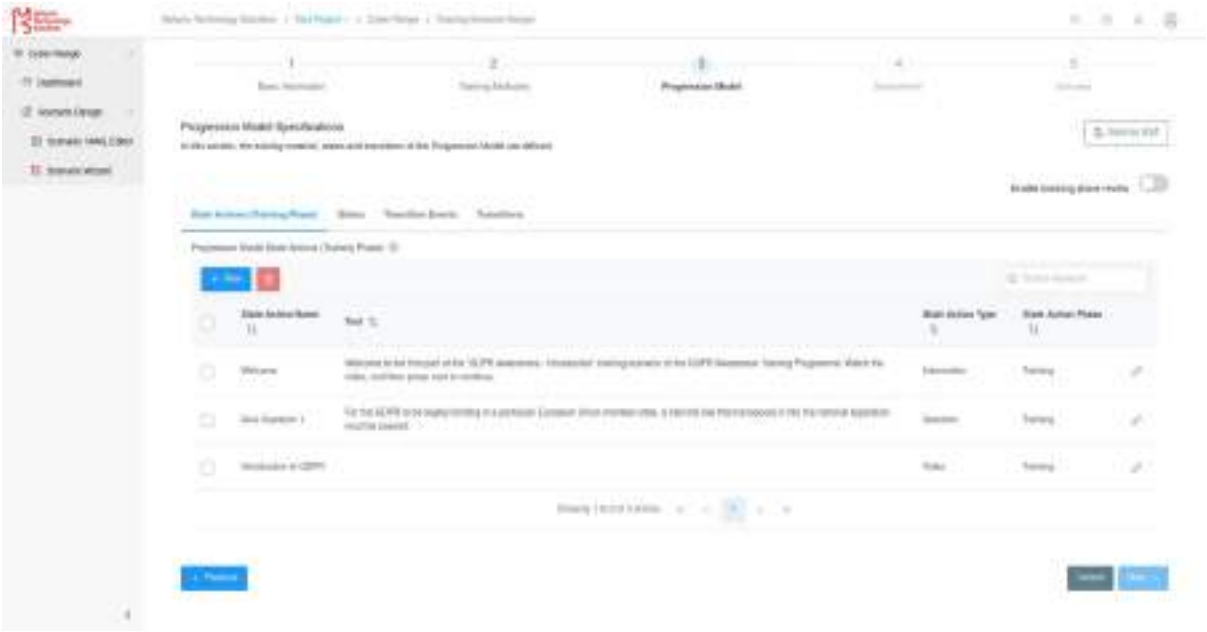


Figure 28. Defining the progression model of a new training programme.

Assuming several training scenarios are available, the interface to view, filter and select the training programmes are shown in Figure 29.

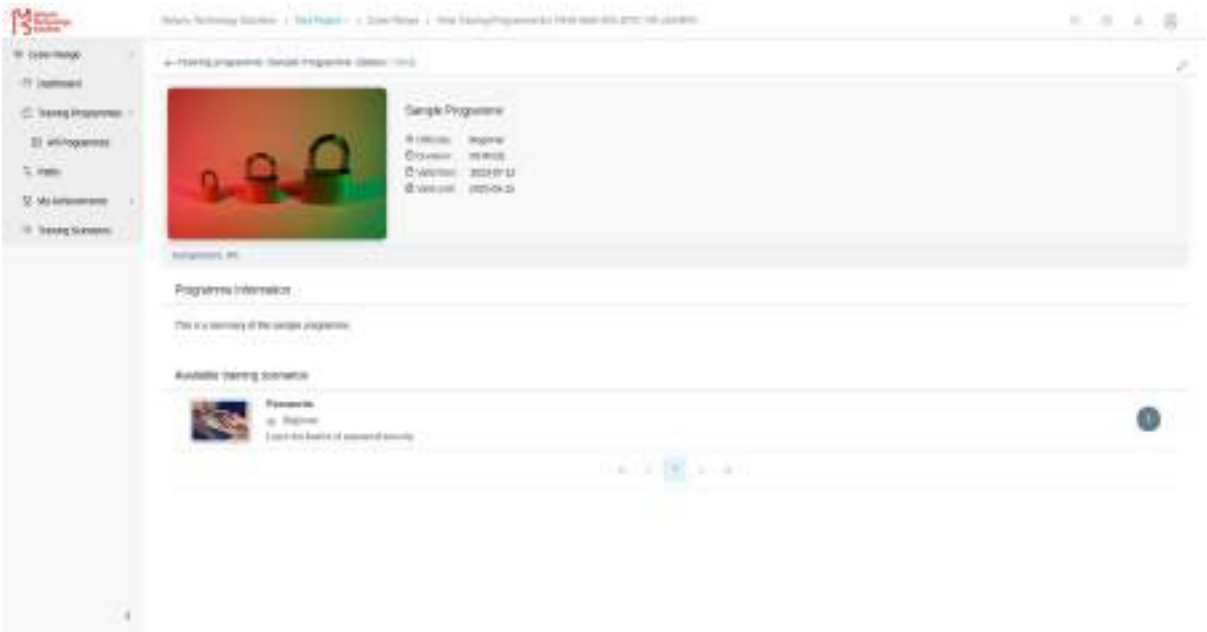


Figure 29. Listing & selection of available training programmes screen.

Further, screenshots showing the CR interfaces during the scenario execution are shown in Figure 30 and in Figure 31. The former provides the Trainee view (the interface the trainee uses to interact with the programme, enter her inputs, see her elapsed time, score, etc.). The latter then provides a Trainer view, whereby the training scenario's progression graph is provided, indicating the trainee's current state, as well as the overall flow of the training scenario.

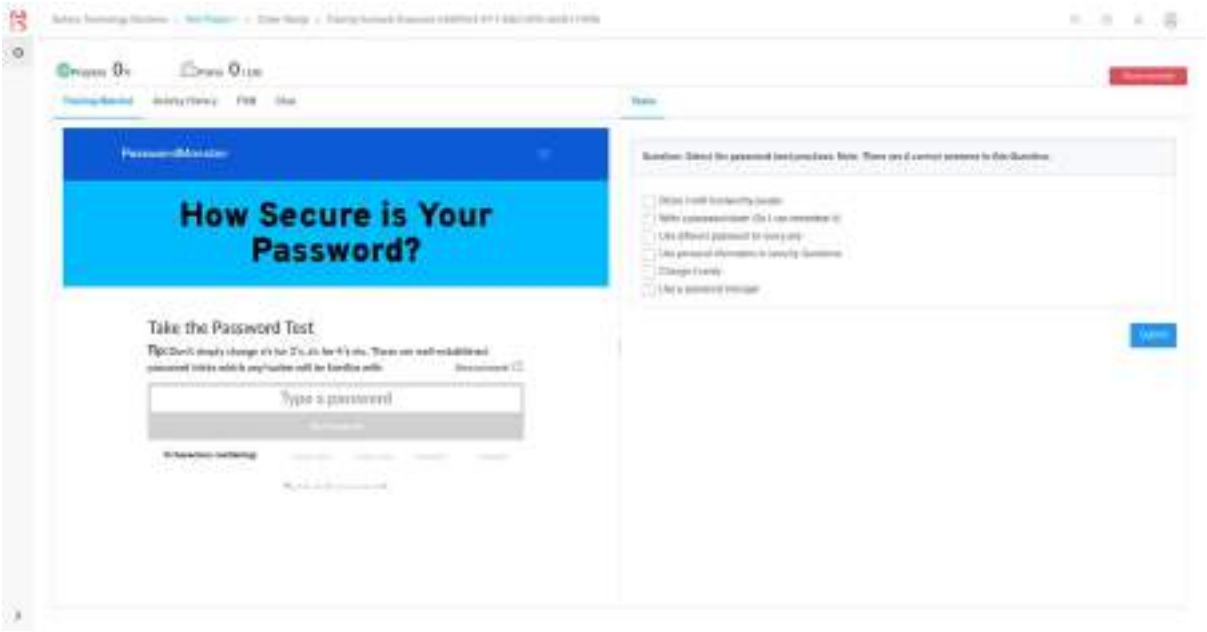


Figure 30. Execution of a training programme screen.

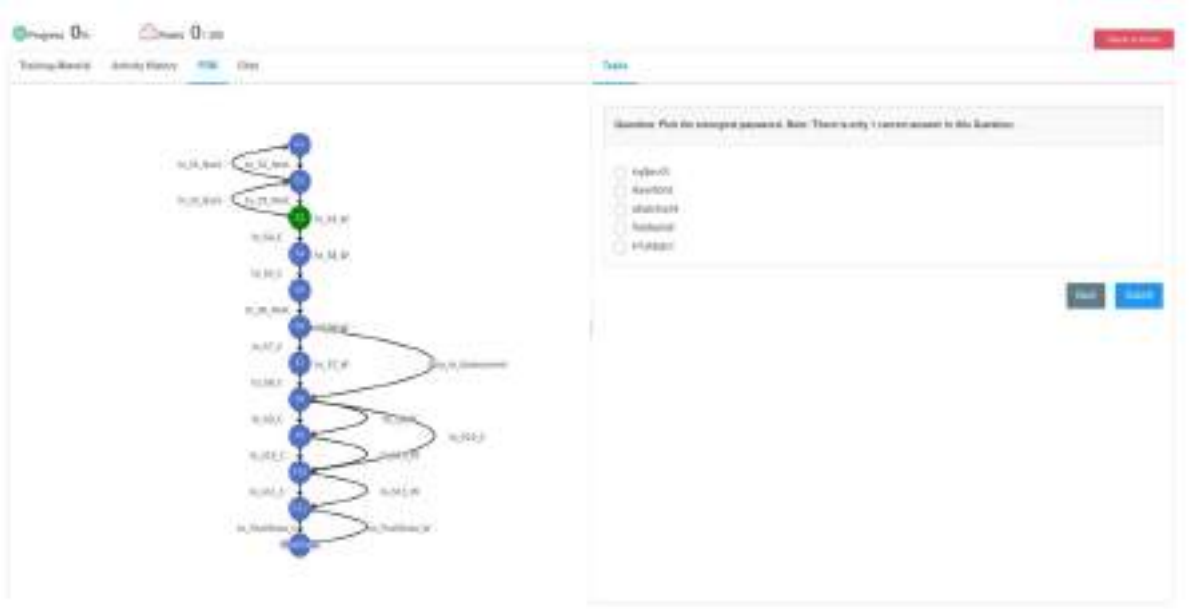


Figure 31. The FSM-based progression engine of the CR

Finally, Figure 32 shows the end screen that pops up to the trainee after the programme is complete, showing the final score and other relevant details.

Figure 32. Scoring screen after the completion

As the baseline capability of the CR is now in place, efforts will focus on developing the new capabilities that will be needed to support PHOENI2X and its requirements (e.g., support for playbook to training programme transition, business continuity training support, training programme generation tailored to the PHOENI2X use case environments). The final capabilities in this regard will be showcased in the second cycle of the project, giving enough time for the development of the additional capabilities but also of the actual training programmes to be executed.

4.1.4 Recap of Current Status & Next Steps

Current efforts have focused on creating a CR environment that is easy to maintain, robust and one that can support the execution & evaluation of the training programmes envisioned within PHOENI2X.

With the baseline functionality in place, future efforts will focus on the development of additional training programmes (e.g., for different incident response and business continuity scenarios, tailored to the project's use cases), as well as the more sophisticated features expected within the project, such as the connection between Playbooks and training material.

4.2 Serious Games

The concept of serious games within the PHOENI2X framework is a module that provides an interactive, context-specific learning. The initiative encompasses a series of games[13], each with a distinct focus: from the physical tabletop game HATCH, designed to foster understanding and mitigation strategies against social engineering attacks among domain experts, to digital games like PROTECT and AWARENESS QUIZ, which broaden the scope to include a wider audience. The concept is to apply that learning in dynamic and realistic scenarios, augmented by machine learning technologies to personalize and enhance the learning experience. This approach not only aims to increase cybersecurity awareness but also to refine the participants' ability to identify and defend against real-world security threats.

4.2.1 Overview

PHOENI2X project approach contains serious games that have the following goals:

- Context-specific Cybersecurity Awareness Training.
- Context-specific Social Engineering Threat elicitation and assessment.
- Simple, engaging and target training methods.
- Training content based on real world security incidents.
- Engaging graphical representation.

The gamification approach consists of one physical tabletop game that engages domain experts in an adventure to design social engineering attacks that work in their specific context. The proposed attacks are rated by the other players, who are also domain experts. The game is moderated by multiple cybersecurity experts to ensure a realistic output of threats. The tabletop game is played with a sample of stakeholders from the relevant context. By experience, playing with more players does not reveal a significant number of new threats after a certain threshold of players is reached. In addition, the game motivates the players to design realistic attacks to win the game. This triggers their learning motivation for cybersecurity.

After the threats have been elicited with the tabletop game HATCH, we use the elicited threats in a digital card game called PROTECT in which the target audience is given attack and defence cards. They have to decide for the correct defence of a social engineering attack derived from an elicited threat to win the game. Finally, the digital game AWARENESS QUIZ shows real attacks related to the elicited threats to convince the players that these are realistic threats and there is an urgency to take cybersecurity seriously.

The digital games have been augmented with ML technologies to:

- Adapt the difficulty of the game.
- Re-introduce attack/defence pairs that the player got wrong.
- Choose a set of questions in the quiz related to the threats player got wrong before.
- Support the player in replying to cybersecurity risk attitude test.
- Evaluate risk attitude test and recommend training content.
- Search for context-specific attacks in the web to create PROTECT card decks and quiz questions for AWARENESS QUIZ.

[13] https://www.social-engineering.academy/en/offers.html

4.2.2 Design Details

The design details of these serious games reveals a comprehensive structure tailored to the intricacies of social engineering threats. The HATCH game, for instance, provides an immersive experience that helps players recognize and counteract social engineering tactics in a controlled, game-based environment. This hands-on approach is supplemented by the PROTECT and AWARENESS QUIZ digital games, which extend the learning experience to include defence mechanisms against digital threats and to reinforce the importance of cybersecurity awareness through interactive quizzes. These games are meticulously crafted to ensure the educational value, the engagement and motivation for the players, leveraging real-world scenarios and threats to impart crucial cybersecurity knowledge and skills.

4.2.3 Tabletop Game: HATCH

In addition to IT infrastructure, employees are also a target for cybercriminals. Through social engineering attacks such as phishing emails, cybercriminals try to exploit the human characteristics of the participants in order to tempt them to perform harmful actions.

The serious game HATCH (Figure 33 & Figure 34) provides an entertaining, interactive group training that teaches employees to identify and successfully defend against social engineering attacks.

Figure 33. Main game board of game HATCH.

With the help of serious games employees can be engaged in security activities in an enjoyable and sustainable way, to increase the awareness of (and defensive behaviour against) social engineering threats. The trainings are conducted exclusively by cybersecurity experts.

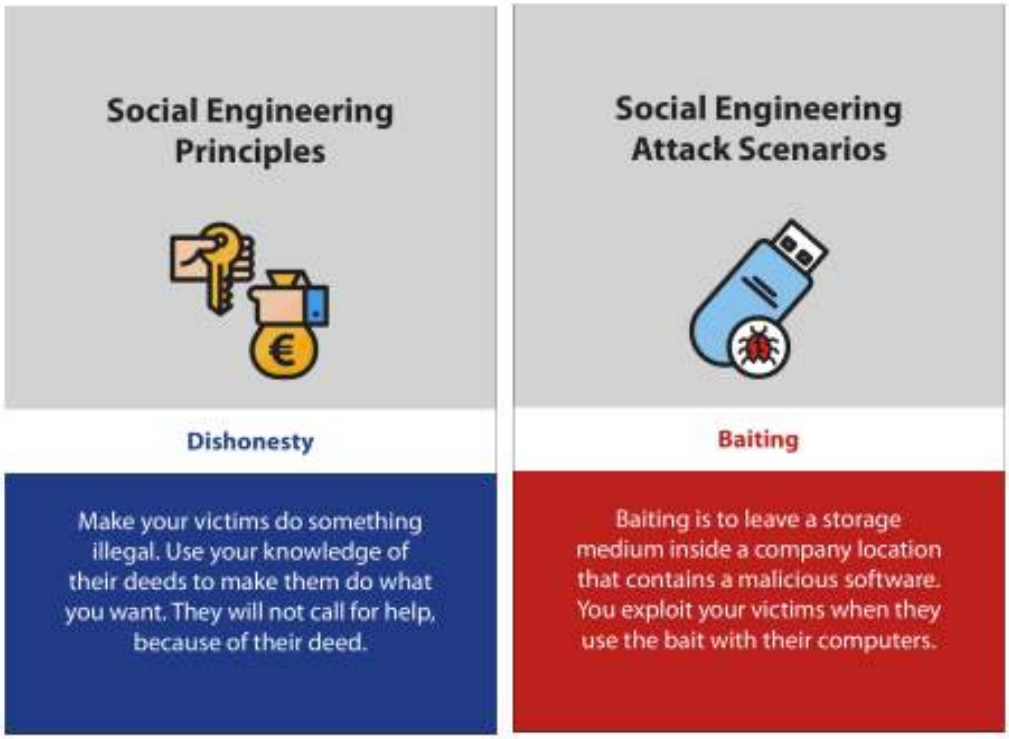


Figure 34. HATCH Sample Cards.

Some key features of HATCH are provided below.

Scenario-based training

- Simulation of actual attacks
- Interactive training of participants
- Complies with ISO 27001 Control A.7.2.2

Company-specific learning content

- Company-specific adaptation (e.g. industry sector) of the game scenario is possible

Threat analysis

- Use for threat analysis possible
- Identification and assessment of relevant social engineering threats

4.2.4 Online Game: PROTECT

In addition to IT infrastructure, employees are also targeted by cybercriminals. Through social engineering attacks such as phishing emails, cybercriminals try to exploit human characteristics of employees to tempt them into harmful actions.

The serious online game PROTECT (Figure 35 & Figure 36) enables a motivating and entertaining training that actively sensitizes employees against social engineering threats.

Figure 35. Main game board of game PROTECT

Training does not have to be boring. With the help of serious games employees can be engaged in security activities in an entertaining and sustainable way.

Figure 36. PROTECT Sample Cards

Key features of PROTECT include the following.

Relevant learning content

- Customization of contents to your security policies possible
- Complies with ISO 27001 Control A.7.2.2
- Multiple languages

Online provision

- Time and location independent training
- Scalable cloud solution
- Playable on PC and mobile devices

Performance measurement

- Data protection compliant evaluation
- High score leader board
- Certificates for participants

4.2.5 Online Game: AWARENESS QUIZ

The serious game CYBERSECURITY AWARENESS QUIZ (Figure 37) represents an online quiz that sensitizes the participants for social engineering threats and their possible effects.

Figure 37. Main game board of game AWARENESS QUIZ

The interactive and hands-on training with the online quiz CYBERSECURITY AWARENESS QUIZ imparts awareness with respect to the potential impact of social engineering attacks. The quiz content is based on real attacks, for which the sources are also provided. This creates a direct reference to the everyday work of employees and highlights the relevance of the content.

Key features of AWARENESS include the following.

Relevant learning content

- Based on real attacks
- Constant updating
- Customization of content to suit your organization.
- Complies with ISO 27001 Control A.7.2.2
- Multiple languages

Motivation

- Predefined quizzes with relevant topics
- Composition of individual quizzes
- Various multiplayer modes
- High score leader board

Online provision

- Time and location independent training
- Scalable cloud solution
- Playable on PC and mobile devices

SEA contributed two serious games for educating people of the dangers of social engineering attacks specific to the scenarios of PHOENI2X. Further, the tool receives exercise configuration data from the PHOENI2X platform to:

- Configurations for the run time of a game
- Data about attacks to be integrated in the games
- Difficulty level of a game
- Define further settings of a game

Technically, we summarize the digital games in a platform called Gamification Tool. For the Gamification Tool, two main games are currently documented by SEA (PROTECT and AWARENESS QUIZ) and are included in the project. The gameplay of the game PROTECT was further improved. This web-based game will be the first game that will be integrated to the platform. The initial steps towards its visualization are also documented in this period. Additionally, SEA documented the tabletop card game HATCH and improved the design as well as the content of the cards to align with the PHOENI2X use cases. Regarding HATCH, SEA also created specific persona cards and a game plan for the energy scenario. Training sessions with this scenario have been provided by SEA during the last project meeting (January 2024) in Athens.

Moreover, an initial analysis is performed on the tools' communications and data flow under the umbrella of the platform architecture activities. The results of the platform's architecture form the baseline for the activities and define the necessary communication mechanisms and interfaces for information exchange between the Training and Visualization Tools and the rest of the PHOENI2X components. Further analysis of the data format, security needs of communications (e.g. the use of Transport Layer Security, TLS) and type of communications such as synchronous or asynchronous has been performed for the needs of this deliverable.

Currently, the data communication to and from the ML engine are being designed.

4.2.6 Implementation Details

Figure 38 shows the Gamification Tool architecture high-level view. Two serious games are currently offered as part of the Gamification Tool component.

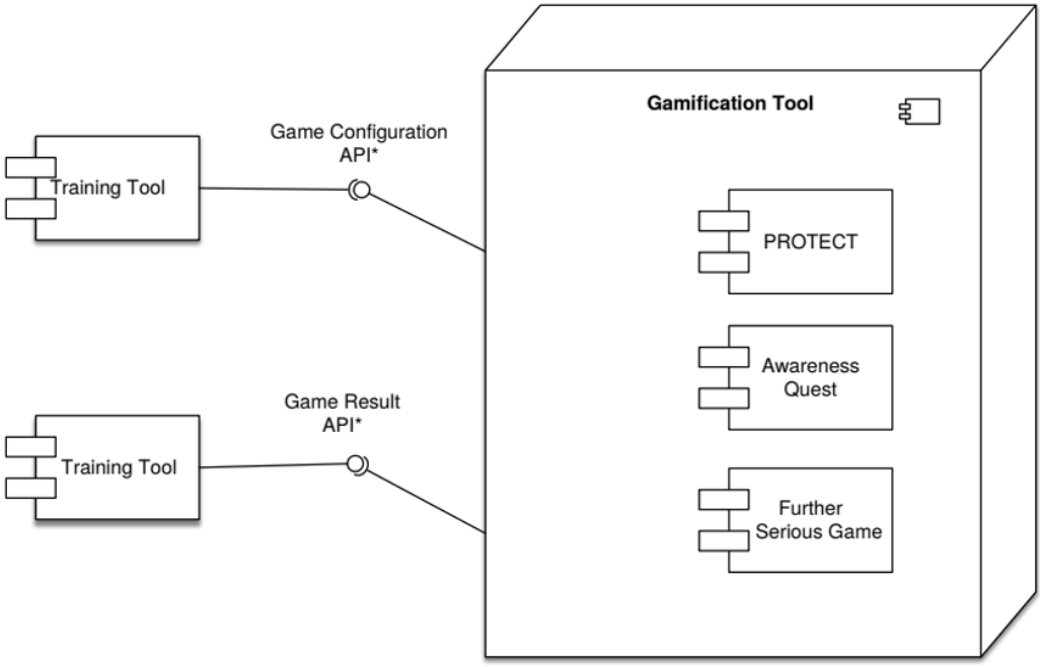


Figure 38. Gamification Tool Architecture and Message Flow

The serious games are started by a user via their own GUI. They are developed using Angular[14] 6 and use REST for all software interfaces. The players can play them on their browsers on PC or on a mobile device. The games support automatic scaling to the type of screen size they are played on. The games have each their own interface, because the data needed to configure them highly depends on the game itself and is almost not generalizable.

4.2.7 Gamification Tool Interfaces

Conceptually, there are two APIs provided/required: one for configuring games e.g. run time, difficulty, userid etc. by the Training Tool and one for transmitting the results and the userid to the Training Tool.

Table 2. Gamification Tool – Protect Interfaces

Interface/Operation Name	Input Data	Output Data	Description
GameSetting.Protect.playerID	Integer		Unique ID of player
GameSetting.Protect.playerName	String		Name of the player
GameSetting.Protect.gameTime	Integer		Time the game is running
GameConfiguration.Protect.difficulty	Integer		Difficulty level

[14] https://angular.io/

GameConfiguration.Protect.jokers	Integer		Number of jokers in the game
GameConfiguration.Protect.attacks	String		New attack and defence description
GameResult.Protect.score		Integer	Points scored in the game
GameResult.Protect.highscore		String	Entire high score list of all players
GameResult.Protect.playerID		Integer	Unique ID of player
GameResult.Protect.playerName		String	Name of the player

Table 3. Gamification Tool – Awareness Quiz Interfaces

Interface/Operation Name	Input Data	Output Data	Description
GameSetting.AwarenessQuiz.playerID	Integer		Unique ID of player
GameSetting.AwarenessQuiz.playerName	String		Name of the player
GameSetting.AwarenessQuiz.gameTime	Integer		Time the player can answer questions
GameConfiguration.AwarenessQuiz.challenge	Integer		Complexity level of the questions
GameConfiguration.AwarenessQuiz.Questions	String		Add new questions and answers to the game
GameResult.AwarenessQuiz.Correct		Integer	The number of correct answers a player provided
GameResult.AwarenessQuiz.playerID		Integer	Unique ID of player
GameResult.AwarenessQuiz.playerName		String	Name of the player

We specified the interfaces of the serious games PROTECT and Awareness Quiz in Table 2 and Table 3, respectively. We use REST services for the data exchange communicating in a JSON data format.

The games PROTECT and Awareness Quiz are programmed in Vue.js and are developed and communicated over REST web services using messages in the JSON format. All configurations and player data can be set and retrieved using REST and JSON. We selected common technologies that are used widespread to ensure compatibility with numerous end user devices and common standards for data exchange in web applications.

Note that these games do not store any user information after a game is finished. When a game is finished, the game reports userid and game result back to the training tool.

We have the following assumptions regarding the PHOENI2X platform:

- Identification of relevant staff for training and Access Management is provided by the PHOENI2X platform
- GDPR compliance for all user data e.g. consent of players to process their data is provided by the platform

5 Alerting, Reporting & Information Exchange

5.1 Smart Mandatory Incident Reporting Tool (SMIR)

5.1.1 Overview

Incident reporting is becoming more and more relevant and mandatory in any context involving critical infrastructures. Current disparity in procedures and lack of harmonization throughout the different regulatory frameworks (not only at a European level but also at a national and sectorial) that apply to the critical infrastructures, such as energy, transport, and health, makes it costly in time and resources to be compliant with all the mandatory incident reporting requirements.

Part of the Alerting, Reporting & Information Exchange capabilities of PHOENI2X are offered through the integration of the open-source Security Incident Response Platform TheHive[15] with the SMIR tool. SMIR, which is an evolution of the asset AIRE developed in the project CyberSec4Europe[16], emerges in the context of incident management and response process as a solution to support and guide the incident reporting teams through all the steps that need to be followed during the mandatory incident reporting process, from the collection of information until the reports are sent to the Supervisory Authorities, and to ease the automatic generation of the incident reports adapted to the different templates required for the applicable regulations. The underlying idea is to collect the information just once and produce all the necessary reports, with a unified incident reporting workflow that can be configured for the different regulations supported.

The positioning of the SMIR tool in the PHOENI2X high-level and detailed architecture is shown in Figure 39 and Figure 40, respectively.

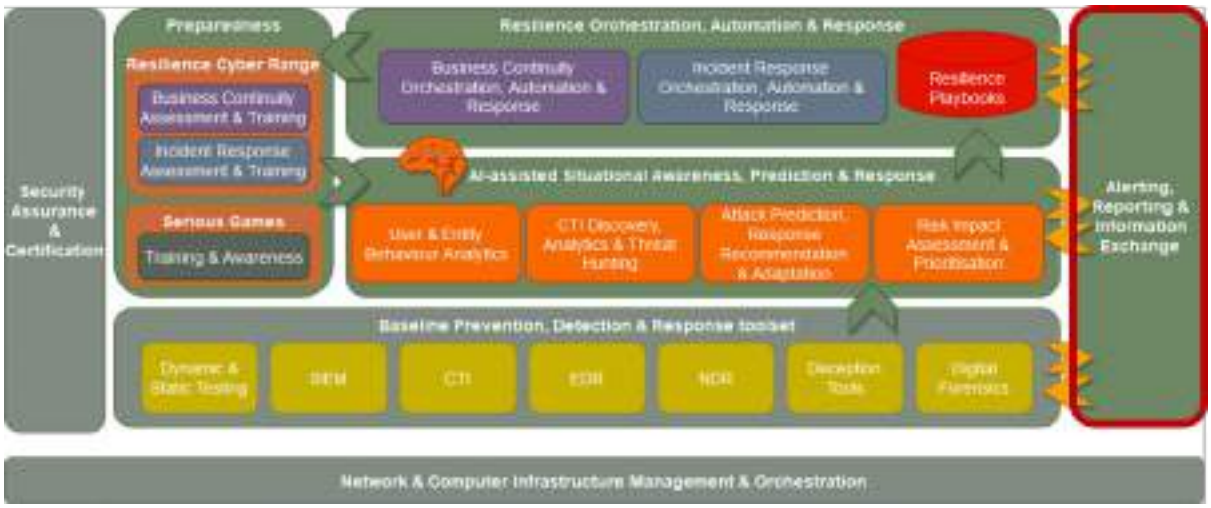


Figure 39: SMIR in the high-level PHOENI2X architecture.

[15] https://thehive-project.org/
[16] https://cybersec4europe.eu/

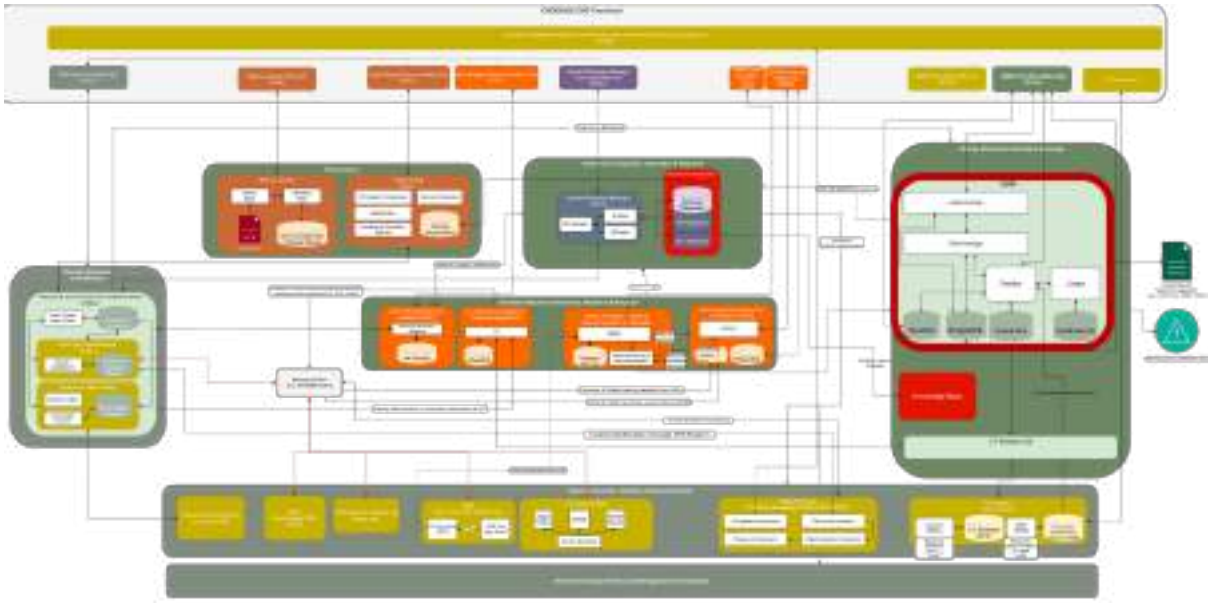


Figure 40. SMIR in the detailed PHOENI2X architecture.

The subsections that follow provide more details on the design & development of the SMIR tool in PHOENI2X.

5.1.2 Design Details

SMIR has a flexible and modular design composed by different services integrated with the open-source Security Incident Response Platform TheHive, as described in (Liliana Pasquale, 2022) for AIRE asset. The SMIR architecture is provided in Figure 41.

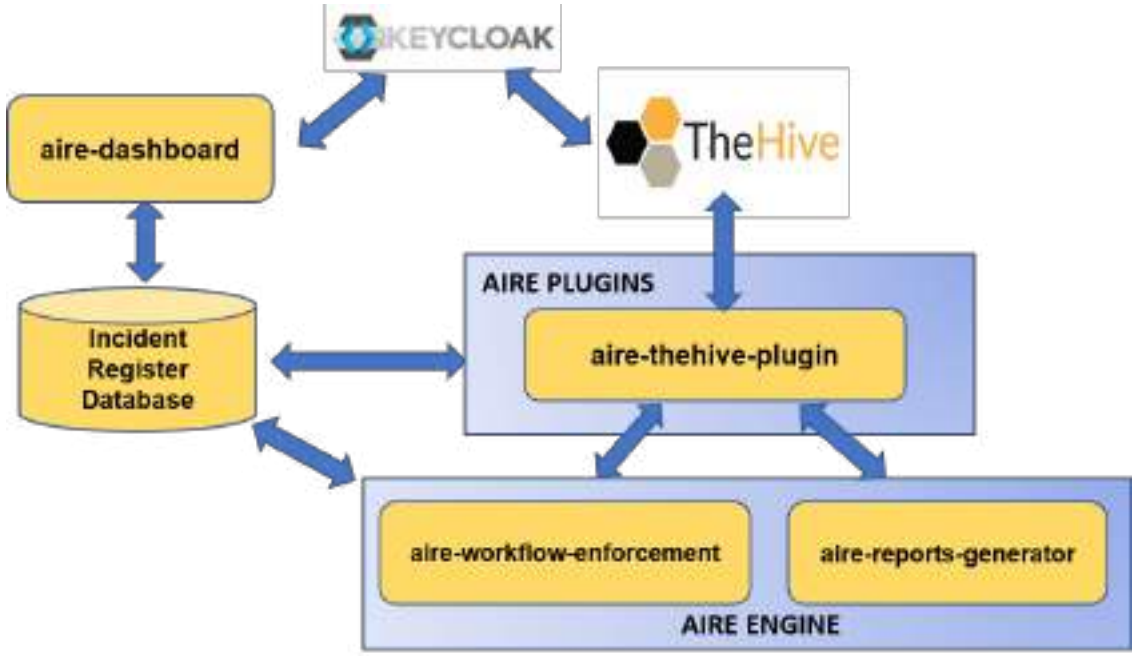


Figure 41: SMIR Architecture (extracted from Figure 36 of [Liliana Pasquale, 2022]).

It is composed of five modules:

- **Aire-workflow-enforcement**: This is a core module responsible for the enforcement of the incident reporting workflow that should be followed to be compliant with the different reporting procedures defined by the regulatory frameworks supported by the tool.
- **Aire-reports-generator**: This is the module in charge of the generation of the required incident report files following different formats and templates. It retrieves the information about the incident stored in the database and adapts it to the different report templates.
- **Aire-thehive-plugin**: This is the module that interacts with the open-source tool TheHive so the previous modules in the SMIR engine core can be agnostic of the incident management tool used.
- **Aire-dashboard**: It provides the graphical interface to configure all the information related to the different regulatory frameworks and it complements the dashboard provided by TheHive for the collection of additional information required for incident reporting.
- **Incident Register Database**: This database stores all the information required for the mandatory incident reporting process and where the information about the incidents is registered.

The mandatory incident reporting workflow, enforced through the use of SMIR is defined through two different Business Process Model and Notation (BPMN), one for the main incident reporting process (see Figure 42) and one for the escalation procedure to be followed when the reporting deadline timing comes (see Figure 43). Each stage in the BPMN process represents a task that is automatically created in the incident registered in TheHive (this is, in the case that shows the information about the incident).

The main incident reporting workflow is divided in four pools, each of them assigned to different user roles. The workflow is started by the **Incident Reporting Team** user and the first task assigned to this user is for data collection (*Data Collection*). Once this task is completed, there are two tasks that will be assigned to the **Incident Classification Team** in the second pool of the BPMN. First, to enrich the information about the incident (*Data Enrichment*) and then to perform the incident classification as significant or not significant based on the available information (*Incident Classification*). This classification must be approved by the user with the role **Controller** through a *Managerial Judgement* task in the third pool of the BPMN. Depending on the controller's decision, the workflow will go back to the data enrichment phase, the incident reporting workflow will be closed, or it will progress to the last pool in the BPMN assigned to the **Incident Reporting Team**. A task for data conversion (*Data Conversion*) will be assigned to the user with this role to generate the reports. This task is supported by the SMIR reports generator module that will produce the report templates associated with the regulatory frameworks enabled with the information required. Once the reports are ready, the workflow moves in the BPMN again to the Controller pool with a task "*Green-light for Reporting*" for approval. Depending on the feedback provided by this controller user through the form included in the SMIR dashboard, the incident will be closed (*Close Incident*), or it will move to the last step (assigned to the Incident Reporting Team role) for sending the report to the Supervisory Authorities (*Reporting & Release*). Depending on the regulation, after the generation of an initial report, it can be necessary to produce updated reports and, in this case, the workflow will go back in the BPMN to the Incident Classification Team pool for data enrichment. When an incident is closed, a notification is sent to the PHOENI2X ROAR component.

The timers included in the BPMN (they are shown in Figure 42 as annotations "can trigger Timer") represent the deadlines established by the different regulatory frameworks and they can be configured through the SMIR dashboard. This allows to adapt a common workflow to different procedures, such as the need of sending a unique report or first, an interim and a final report, and after different time periods (e.g. before 72 hours or before 2 hours) counting from different points in the workflow (e.g. once the incident has been detected, once the incident has been classified as

significant or once the incident has been reported). These timers trigger the escalation procedure defined by the BPMN shown in Figure 43, where the status of the report is verified and, in case it has not been released yet, the Incident Reporting Team users are notified.

More details about the SMIR architecture and the underlying BPMN workflows can be found in Lilian Pascqualle (2022).

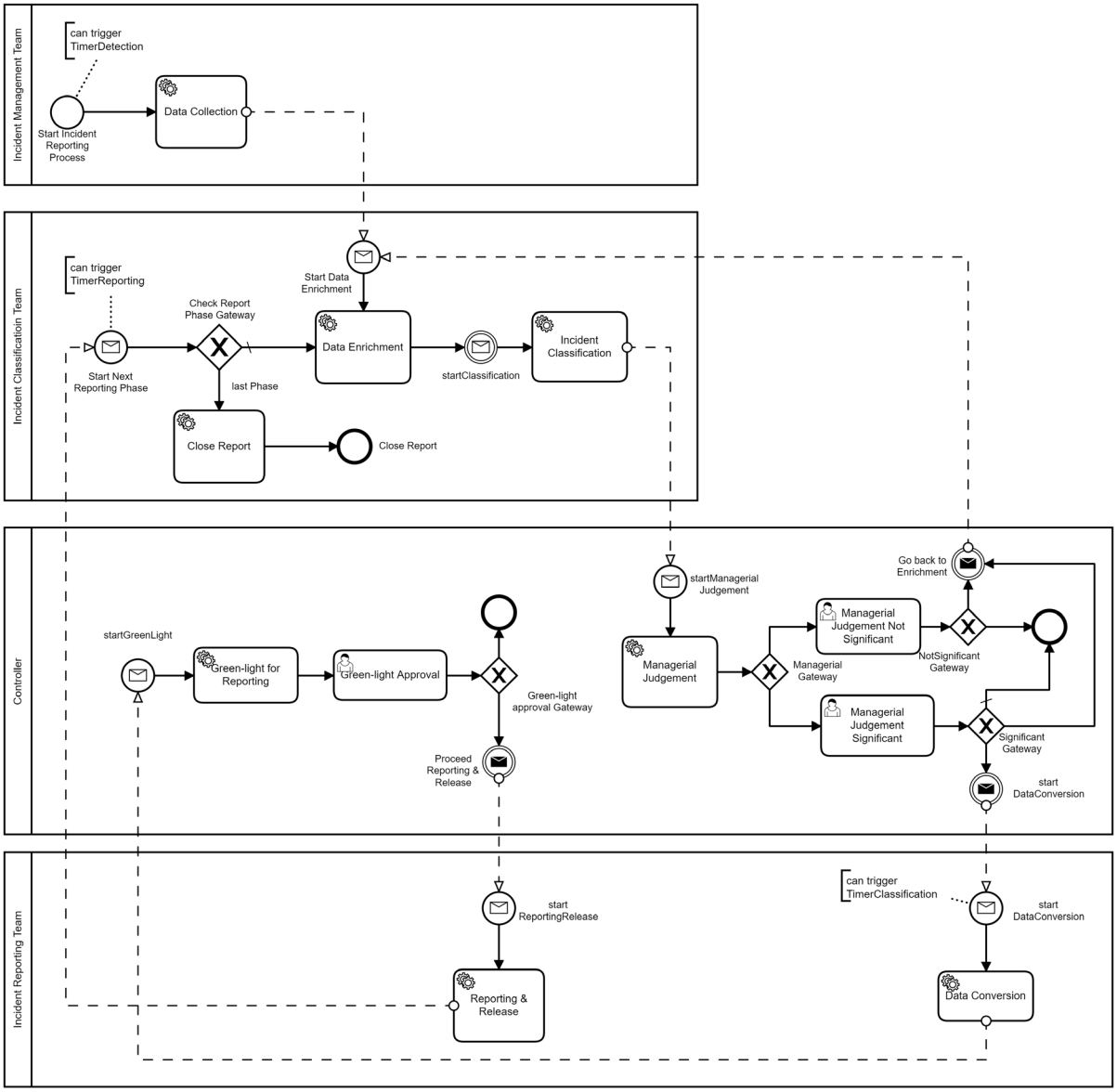


Figure 42: SMIR Incident Reporting BPMN (extracted from Figure 36 of [Liliana Pasquale, 2022]).

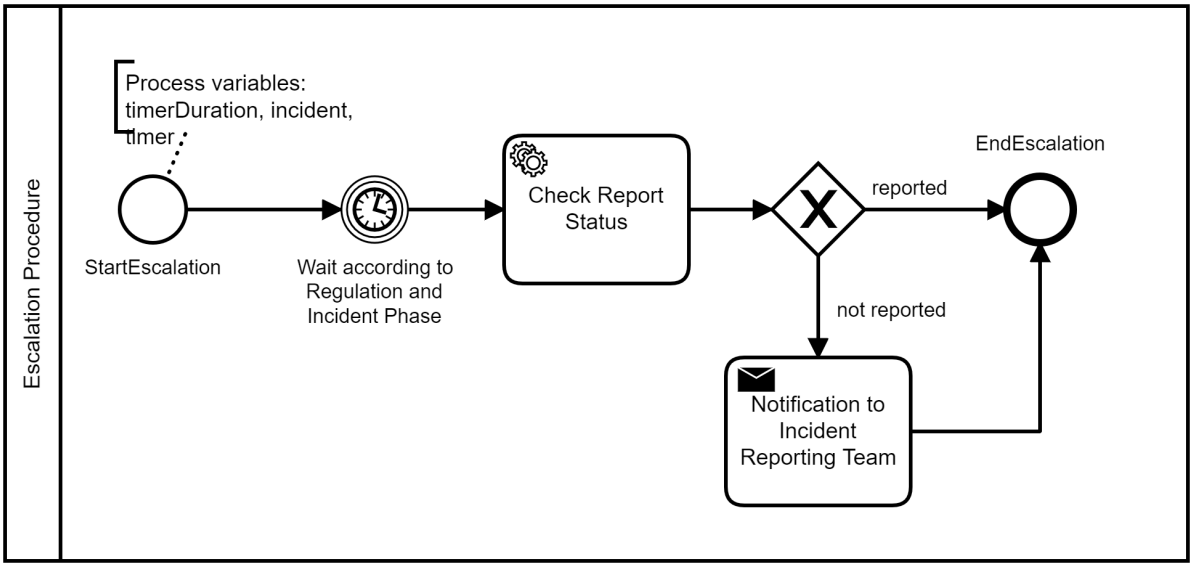


Figure 43: SMIR Incident reporting escalation procedure BPMN (extracted from Figure 36 of [Liliana Pasquale, 2022]).

5.1.3 Implementation Details

Different technologies have been used in the implementation of SMIR. The SMIR engine modules (responsible for workflow enforcement and reports generator) and the plugin to interact with TheHive have been developed as Springboot microservices[17]. The SMIR dashboard has been implemented in Django[18] and PostgreSQL[19] is used for the database. The invocation of the services provided by the SMIR engine from the open-source incident management and response tool TheHive are done through different Cortex[20] responders implemented in Python.

The reports generator module makes use of the Apache POI library[21] for Microsoft Documents to work with Excel and Word files and the Apache PDFBox library[22] for PDF documents. Both libraries are published under the Apache License v2.0. Additionally to the report template file (in Word, PDF or Excel format) provided by each regulation, it is necessary to create an Excel file thats specifies the mapping between the information required in each template and where this piece of data is available in the incident reporting data model. An example used for the Hellenic CSIRT template is shown in Figure 44. For each regulation supported by SMIR, both files (the report template and the Excel mapping file) can be uploaded using the SMIR dashboard.

Figure 44: SMIR Excel mapping file extract example.

[17] https://spring.io/microservices
[18] https://www.djangoproject.com/
[19] https://www.postgresql.org/
[20] https://docs.thehive-project.org/cortex/
[21] https://poi.apache.org/
[22] https://pdfbox.apache.org/

The workflow enforcement module uses Camunda[23] to automate the execution of the tasks included in the mandatory incident reporting process and the escalation procedure defined through BPMN (Business Process Model and Notation) files (see Figure 42 and Figure 43). It also integrates a BPMN parse listener associated to different stages of the BPMN to adapt the general incident reporting workflow to the specific requirements about timing and when to trigger the escalation procedure in case the report has not been sent to the Supervisory Authorities in the time established by a specific regulation.

Details about endpoints provided by each of SMIR modules are available in (Liliana Pasquale, 2022). The improvements developed in SMIR in the context of PHOENI2X are summarized below:

- The ROAR component can automatically create incidents via TheHive API and these incidents are automatically registered in SMIR (see example in Figure 45). The incidents can be enriched with CTI information from MISP. The incidents are created with a predefined template to collect the information required by SMIR for incident reporting and as soon as it is received, thanks to the webhooks generated by TheHive that are received by SMIR TheHive plugin listener, it is automatically triggered the incident reporting workflow. As a result, the first task in the incident reporting workflow (called "Data Collection") is created in the incident and assigned to the incident management team user (see Figure 46).

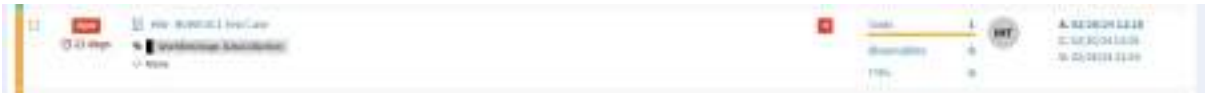


Figure 45: Incident registered in TheHive from ROAR

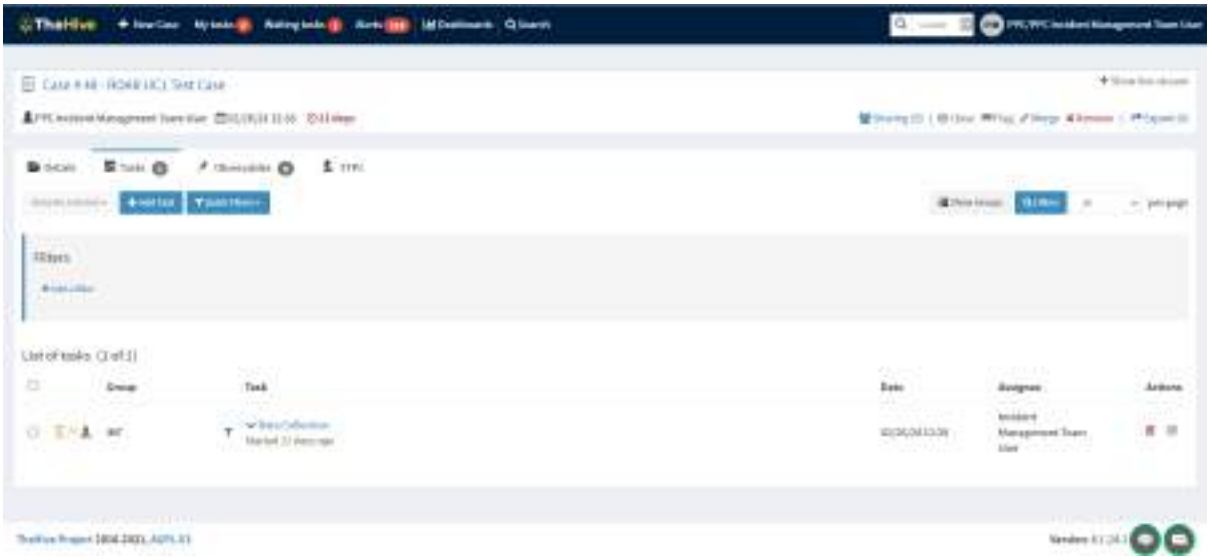


Figure 46: DataCollection task (incident reporting workflow) triggered by SMIR

- Since ROAR can register new incidents in SMIR, it also needs to receive a confirmation from SMIR when an incident is closed (e.g., to progress a playbook). As such, SMIR has been extended to support notification functionality to external endpoints.

- The graphical interface provided by SMIR allows the users to configure SMIR and interact with the information about the incident that has been extended for the needs of the PHOENI2X project with the possibility to configure notifications to ROAR (see Figure 47). These notifications can be attached to different organizations (because the endpoint can change depending on the use case

[23] https://camunda.com/

for a same deployment of SMIR) and to different stages in the incident reporting workflow (see example in Figure 48).

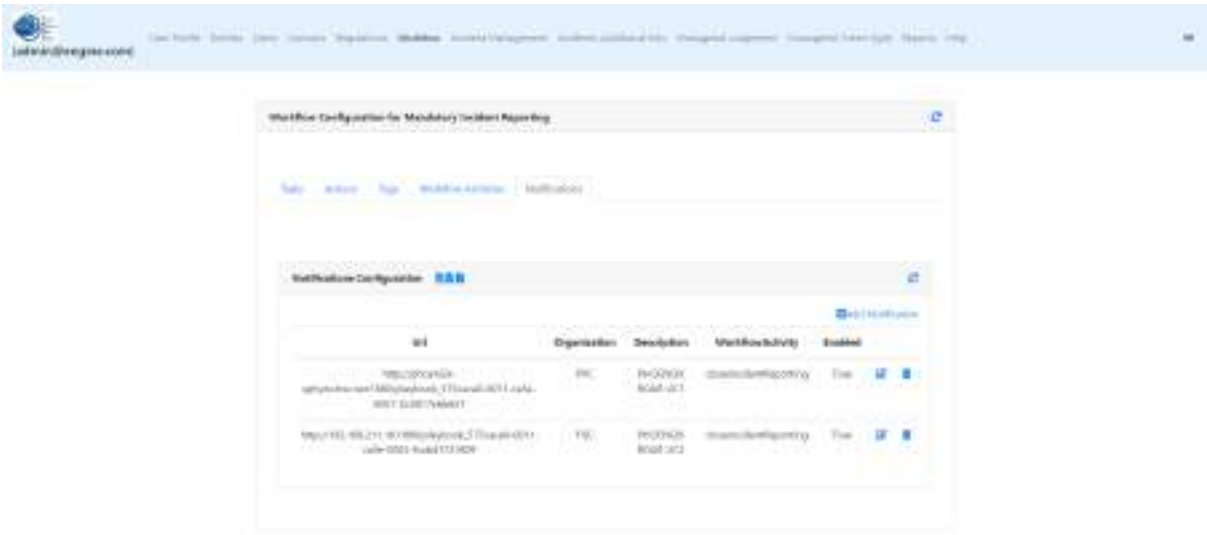
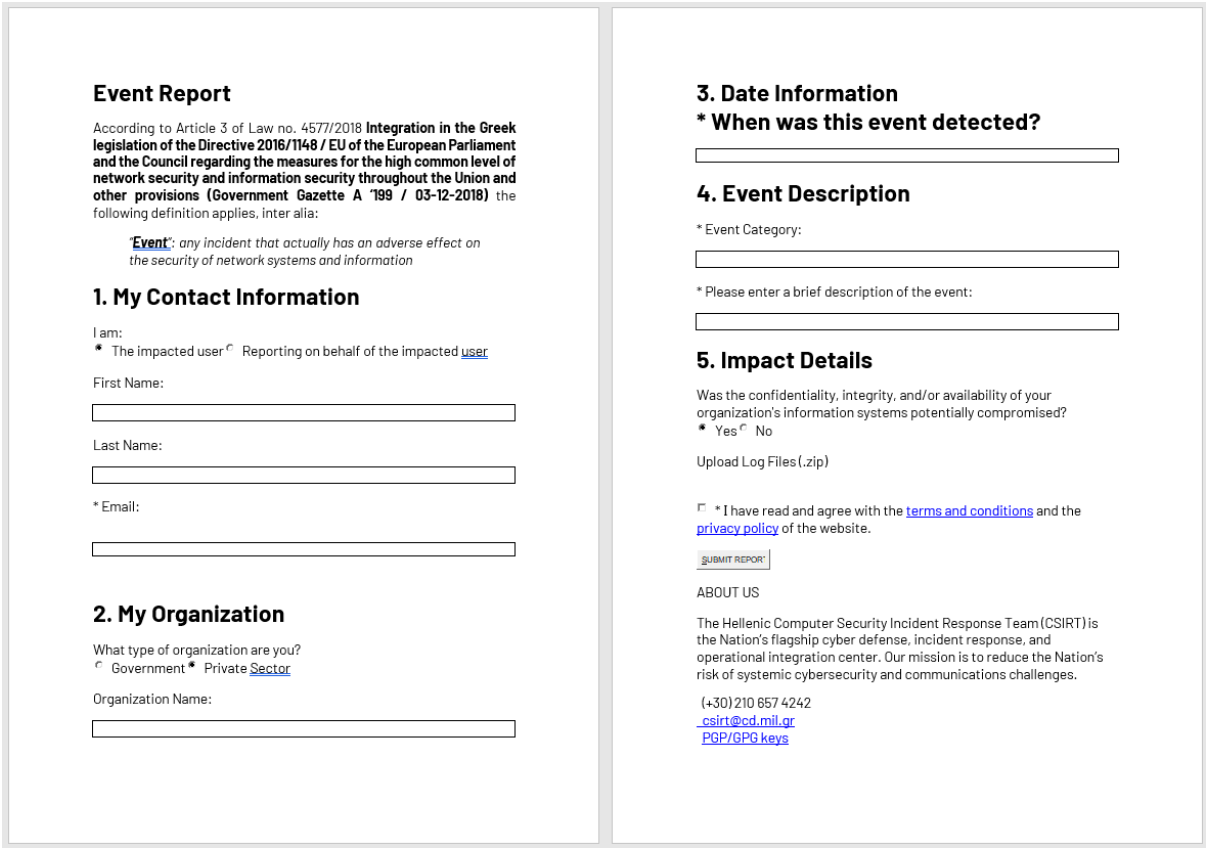


Figure 47: SMIR Dashboard – ROAR Notifications

Figure 48: SMIR Notification Configuration

- NIS2 directive and GDPR have been considered in PHOENI2X to cover the use case needs in order to report to their respective national authorities. For NIS2, the Word template offerd by the Hellenic CSIRT (see Figure 49) has been integrated into SMIR. For GDPR, an Excel template has been created based on the information required by FGC Cybersecurity procedures and regulations (see Figure 50). The templates can be configured via the GUI (see Figure 51).



Event Report

According to Article 3 of Law no. 4577/2018 **Integration in the Greek legislation of the Directive 2016/1148 / EU of the European Parliament and the Council regarding the measures for the high common level of network security and information security throughout the Union and other provisions (Government Gazette A '199 / 03-12-2018)** the following definition applies, inter alia:

*"**Event**": any incident that actually has an adverse effect on the security of network systems and information*

1. My Contact Information

I am:
* The impacted user ○ Reporting on behalf of the impacted user

First Name:

Last Name:

* Email:

2. My Organization

What type of organization are you?
○ Government * Private Sector

Organization Name:

3. Date Information
* When was this event detected?

4. Event Description

* Event Category:

* Please enter a brief description of the event:

5. Impact Details

Was the confidentiality, integrity, and/or availability of your organization's information systems potentially compromised?
* Yes ○ No

Upload Log Files (.zip)

* I have read and agree with the terms and conditions and the privacy policy of the website.

SUBMIT REPORT

ABOUT US

The Hellenic Computer Security Incident Response Team (CSIRT) is the Nation's flagship cyber defense, incident response, and operational integration center. Our mission is to reduce the Nation's risk of systemic cybersecurity and communications challenges.

(+30) 210 657 4242
csirt@cd.mil.gr
PGP/GPG keys

Figure 49: Hellenic CSIRT Word report template.

Qué Notificar	Descripción	Información a notificar
Asunto.	Frase que describa de forma general el incidente. Este campo será heredado por todas las notificaciones asociadas al incidente.	
OSE / PSD.	Denominación del operador de servicios esenciales o proveedores de servicios digitales que notifica.	
Fecha y hora del incidente.	Energía, transporte, financiero, etc.	
Fecha y hora de detección del incidente.	Indicar con la máxima precisión posible cuando ha ocurrido el incidente.	
Descripción.	Describir con detalle lo que ha pasado	
Recursos tecnológicos afectados.	Indicar la información técnica sobre el nombre y tipo de activos afectados por el incidente, incluyendo direcciones IP, sistemas operativos, aplicaciones, versiones...	
Origen del incidente.	Indicar la causa del incidente si se conoce. Apertura de un fichero sospechoso, conexión de un dispositivo USB, acceso a una página web maliciosa, etc	
Taxonomía (clasificación).	Posible clasificación y tipo de incidente en función de la taxonomía descrita.	
Nivel de peligrosidad.	Especificar el nivel de peligrosidad asignado a la amenaza	
Nivel de impacto.	Especificar el nivel de impacto asignado al incidente.	
Impacto transfronterizo.	Indicar si el incidente tiene impacto transfronterizo en algún Estado miembro de la Unión Europea.	
Plan de acción y contramedidas.	Actuaciones realizadas hasta el momento en relación al incidente. Indicar el Plan de acción seguido, junto con las contramedidas implementadas.	
Afectación.	Indicar si el afectado es una empresa o un particular, y las afectaciones según el nivel de impacto asignado.	
Medios necesarios para la resolución (J-P).	Capacidad utilizada en la resolución del incidente en Jornadas – Persona	

Figure 50: FGC Excel report template.

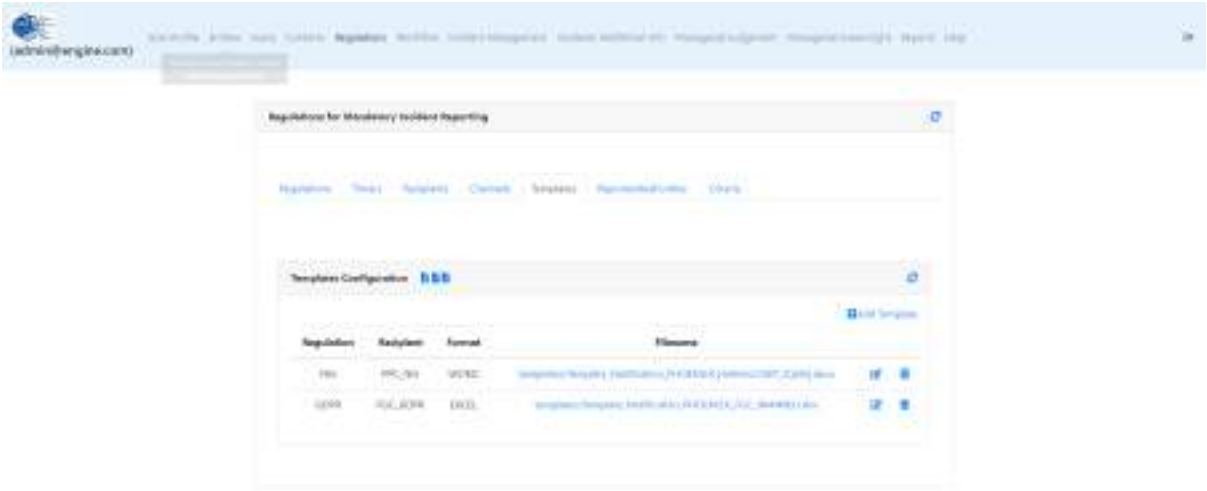


Figure 51: PHOENI2X Report templates configuration in SMIR

- Since in PHOENI2X there are different use cases, each of them is run by a different organization, it has been necessary to include multi-tenant support in SMIR. Now, each organization can configure its own API Key in SMIR to interact with TheHive. In this way, incidents are managed in a separated way for each organization and a user will only have access to incidents that have been assigned to their organization.

5.1.4 Recap of Current Status & Next Steps

The efforts have been focused on adapting the SMIR tool to the needs of PHOENI2X. These needs include adapting the communication between SMIR and the rest of the components, such as the ROAR-SMIR interactions to create new incidents and notify case closures within the platform. Additionally, the tool has been adapted for the use case 1 and use case 2, energy and transport pilots. This has involved including reports within SMIR that meet the pilots' needs. Furthermore, work has also been done to provide separation for different pilots within the same tool, so that only one instance of SMIR is needed to isolate cases and information based on the organization. Future efforts will focus on the following points:

- Adapting the tool to the health use case.
- Automating processes such as the sharing the generated report with relevant entities that have been previously configured in the platform.
- Including technical information in the reports that can come from different components within PHOENI2X. Within this technical information, we can emphasize security playbooks.

5.2 Notification Playbooks

In the context of Alerting, Reporting & Information Exchange, PHOENI2X introduced a new type of playbook in the CACAO standard, the notification playbook, including a set of machine-readable metadata that provide proper semantics and enable more verbose searching, indexing, and filtering. In particular, the definition of a notification playbook is as follows.

"*A playbook that is primarily focused on the orchestration steps required to notify and disseminate information and other playbooks about a security event, incident, or threat. For example, a notification playbook can be used to notify multiple entities about a new attack or campaign and disseminate information or playbooks to deal with it as quickly as possible.*" [3]

In addition, the actions and in particular the commands within a playbook can be semantically enriched with metadata (playbook activities) that define more detailed descriptions for the various activities that the playbook performs. In the case of a notification playbook the following playbook activities can be populated:

- **Compose content:** This activity composes the notification text that will be distributed. This activity must be used with notification playbooks.
- **Deliver content:** This activity delivers notification content to the intended audience. This activity should be used with notification playbooks.
- **Identify audience:** This activity identifies the audience of a notification. This activity should be used with notification playbooks.
- **Identify channel:** This activity identifies the method by which notification content will be sent. This activity should be used with notification playbooks.

The playbook activities can be used for "labelling" commands and inherently the action steps that encapsulate the commands and can be leveraged by a search engine like in the example provided right below.

To find all playbooks that are **investigative**, and include **scan systems** activities:

```
Select * playbooks where playbook_types contains "investigation" &&
            playbook_activities contains "scan-system"
```

PHOENI2X will utilize the introduced notification playbook to encode in a reusable manner the operating procedures required for various types of notification, including informing specific teams about incidents, forwarding relevant CTI, and supporting incident reporting requirements.

5.3 Playbook Exchange via MISP (Information Exchange)

5.3.1 Overview

The PHOENI2X consortium is one of the first proposers and adopters of a new CTI paradigm, which propels the integration of structured courses of action (e.g., security playbooks) with CTI. By utilizing automation and the exchange and use of intelligence and know-how to respond, defenders can establish anticipatory threat reduction capabilities and respond to these threats in cyber-relevant time. In that context, we have created extensions for MISP and STIX 2 to share cybersecurity playbooks, including CACAO, and couple them with CTI while also providing the ability for knowledge management such as semantic indexing, filtering, and searching through a rich set of metadata.

5.3.1 Design and Implementation Details

To extend the schema of MISP and STIX 2 to support security playbooks, their connection with CTI, and their exchange, we designed a harmonized template with a set of rich metadata and transposed it to object templates according to how each language defines its concepts. Our extensions are agnostic of the playbook format or serialization and can support CACAO or any other open or proprietary format, including playbooks in human natural language or plain graphical representations. The MISP object template can be found on the official GitHub repo[24] of the MISP project. The STIX 2.1 extension and documentation are available on a GitHub repository[25] maintained by University of Oslo. Both extensions wrap the properties and metadata appearing in Table 4.

[24] https://github.com/MISP/misp-objects/blob/main/objects/security-playbook/definition.json
[25] https://github.com/cyentific-rni/stix2.1-coa-playbook-extension

Table 4. MISP extension property specification

Property Name	Description
Playbook id	A value that (uniquely) identifies the playbook. If the playbook itself embeds an identifier then the playbook_id SHOULD use the same identifier (value) for correlation purposes.
Description	An explanation, details, and more context about what this playbook does and tries to accomplish.
Created	The time at which the playbook was created. This may be different than the time at which the "parent" Course of Action object instance was created.
Modified	The time at which the playbook was last modified.
Revoked	A boolean that identifies if the playbook extension (instance) is no longer valid.
Playbook creation time	The time at which the playbook was originally created.
Playbook modification time	The time at which the playbook was last modified.
Playbook valid from	The time from which the playbook is considered valid and the steps that it contains can be executed.
Playbook valid until	The time from which the playbook should no longer be considered a valid playbook to be executed.
Playbook creator	The identifier of the entity that created the playbook.
Labels	A set of labels for the playbook (e.g., adversary persona names, associated groups, or malware family/variant/name that this playbook is related to).
Organization type	The type of organization that the playbook is intended for.
Playbook standard	The standard/format/notation the playbook conforms to (e.g., CACAO, BPMN).
Playbook abstraction	The playbook's level of abstraction (with regards to consumption). Open Vocabulary: [template, executable]
Playbook type	A list of playbook types that specifies the operational roles this playbook addresses. The open vocabulary is based on the available options in the CACAO standard and NIST SP 800-61 rev2. Open Vocabulary: [notification, detection, investigation, prevention, mitigation, remediation, analysis, containment, eradication, recovery, attack]
Playbook impact	From 0 to 100, an integer representing the impact the playbook has on the organization. A value of 0 means specifically undefined. Impact values range from 1, the lowest impact, to a value of 100, the highest. For example, a purely investigative playbook that is non-invasive could have a low impact value of 1. In contrast, a playbook that performs changes such as adding rules into a firewall should have a higher impact value.
Playbook severity	From 0 to 100, an integer representing the seriousness of the conditions that this playbook addresses. A value of 0 means specifically undefined. Severity values range from 1, the lowest severity, to a value of 100, the highest.
Playbook priority	From 0 to 100, an integer representing the priority of this playbook relative to other defined playbooks. A value of 0

	means specifically undefined. Priority values range from 1, the highest priority, to a value of 100, the lowest. This property can support addressing different use cases and requirements of a producing or consuming entity.
Playbook base64	The entire playbook encoded in base64. Security playbook producers and consumers use this property to share and retrieve entire playbooks.

5.3.2 Recap of Current Status & Next Steps

PHOENI2X has designed and implemented extensions to standardized CTI schemas, such as MISP and STIX 2.1, to support security playbooks in threat information exchange. This approach enables organizations to exchange defensive tradecraft in addition to CTI, allowing organizations to respond faster and more effectively to threats. The next steps involve updating the schemas or, as stated above, the harmonized (common) metadata template that drives the development of the extensions based on the feedback received from end users. In addition, regarding STIX 2.1, the consortium will perform all the necessary actions required by the OASIS CTI technical committee to prepare and include this extension in the next version of the STIX standard.

6 Conclusions & Next Steps

This deliverable provided the overview, design & implementation details of the first version of Coordinated Response & Preparedness enablers, as delivered in M21 of the project. These included all planned components developed under the different WP4 tasks, including the Resilience Orchestration & Response editor & engine, the Resilience Playbooks themselves, the Resilience Cyber Range, different types of Serious Games, the Smart Mandatory Incident Reporting tool, the CACAO standard, a specific playbook type for notifications, the layout extension for graphically representing CACAO playbooks, and MISP and STIX 2.1 objects that enhance existing CTI approaches with the ability to exchange defensive tradecraft (i.e., security playbooks).

In terms of next steps, the output components will be leveraged in the context of WP5 activities to update the MVP version of PHOENI2X, as needed, in order to derive the first release (V1) of PHOENI2X, then carrying out the relevant demonstration and validation activities.

Feedback from the above will be leveraged to further improve the components which, in addition to already planned features per component, will drive the second (and final) release of said components. These efforts towards the design, development & delivery of V2 of the WP4 components will be documented in D4.2 (“Coordinated Response & Preparedness Enablers v2”), which will update this deliverable as needed.

References

[1] OASIS, Collaborative Automated Course of Action Operations (CACAO) Security Playbooks Version 2.0, Nov. 2023. https://docs.oasis-open.org/cacao/security-playbooks/v2.0/cs01/security-playbooks-v2.0-cs01.html

[2] Somarakis, I., Smyrlis, M., Fysarakis, K., Spanoudakis, G. (2020). Model-Driven Cyber Range Training: A Cyber Security Assurance Perspective. In: Fournaris, A., et al. Computer Security. IOSEC MSTEC FINSEC 2019 2019 2019. Lecture Notes in Computer Science, vol 11981. Springer, Cham. https://doi.org/10.1007/978-3-030-42051-2_12

[3] Smyrlis, M., Fysarakis, K., Spanoudakis, G., Hatzivasilis, G. (2020). Cyber Range Training Programme Specification Through Cyber Threat and Training Preparation Models. In: Hatzivasilis, G., Ioannidis, S. (eds) Model-driven Simulation and Training Environments for Cybersecurity. MSTEC 2020. Lecture Notes in Computer Science(), vol 12512. Springer, Cham. https://doi.org/10.1007/978-3-030-62433-0_2