Smart, Automated, and Reliable Security Service Platform for 6G

Deliverable D4.3
ROBUST-6G AI/ML Driven Zero-Touch Security Management Platform Consolidated Design

Co-funded by the European Union

6GSNS

ROBUST-6G project has received funding from the Smart Networks and Services Joint Undertaking (SNS JU) under the European Union's Horizon Europe research and innovation programme under Grant Agreement No 101139068.

Date of delivery:	30/09/2025	Version:	1.0
Project reference:	101139068	Call:	HORIZON-JU-SNS-2023
Start date of project:	01/01/2024	Duration:	30 months

Document properties:

Document Number:	D4.3
Document Title:	ROBUST-6G AI/ML Driven Zero-Touch Security Management Platform Consolidated Design
Editor(s):	Alberto Pérez García (UMU), José María Jorquera Valero (UMU)
Authors:	Contributors and their organisations are listed below
Contractual Date of Delivery:	30/09/2025
Dissemination level:	PU[1]
Status:	Final
Version:	1.0
File Name:	ROBUST-6G D4.3_v1.0

Revision History

Revision	Date	Issued by	Description
0.1	07-05-2025	ROBUST-6G WP4	Template for Deliverables 4.3
0.2	31-07-2025	ROBUST-6G WP4	First complete draft
0.3	22-08-2025	ROBUST-6G WP4	Second complete draft
0.4	12-09-2025	ROBUST-6G WP4	Polish phase before internal and external review
0.5	15-09-2025	ROBUST-6G WP4	Internal and external review
0.6	23-09-2025	ROBUST-6G WP4	Final complete draft after review
1.0	30-09-2025	ROBUST-6G WP4	Final version

Abstract

The principal goal of WP4 is to lay the foundations of ROBUST-6G Zero-Touch Security Management Platform by consolidating the last design of components involved in platform and declaring a first round of development activities as part of an initial prototype. The Zero-Touch Security Management Platform covers four main pillars such as i) security service and resource orchestration, ii) programmable pervasive monitoring, iii) threat/anomaly detection and prediction, and iv) closed-loop-based security automation.

Keywords

Zero-touch security automation, Security orchestration, Service resource orchestration, AI/ML for security, AI/ML-based incident prediction, Threat detection, Programmable monitoring, Security closed-loops

Disclaimer

[1] SEN = Sensitive, only members of the consortium (including the Commission Services). Limited under the conditions of the Grant Agreement

PU = Public

Funded by the European Union. The views and opinions expressed are however those of the author(s) only and do not necessarily reflect the views of ROBUST-6G Consortium nor those of the European Union or Horizon Europe SNS JU. Neither the European Union nor the granting authority can be held responsible for them.

List of Contributors

Participant	Short Name	Contributors
Nextworks	NXW	Pietro G. Giardina, Marco Ruta
THALES SIX GTS FRANCE SAS	THALES	Louis Cailliot
Universidad de Murcia	UMU	Alberto García Pérez, José María Jorquera Valero, Manuel Gil Pérez
AXON LOGIC IDIOTIKI KEFALAIOUXIKI ETAIREIA	AXON	Chih-Yang Pee
EURECOM	EUR	Marios Kountouris
Linkopings Universitet	LIU	Nikolaos Pappas, Eunjeong Jeong

List of Reviewers

Participant	Short Name	Contributors
Telefónica Innovación Digital	TID	Riccardo Nicolicchia
THALES SIX GTS FRANCE SAS	THALES	Louis Cailliot

Executive Summary

This document reports on the final design of ROBUST-6G Zero-Touch Security Platform, including the security service and resource orchestration, programmable pervasive monitoring, threat/anomaly detection and prediction, and closed-loop-based security automation. These are the pillars identified in D4.1 [R6G24-D22] for the Zero-Touch Security Platform, and from which this deliverable presents a comprehensive overview of components to cover the principal milestones established in D4.1. In particular, this new deliverable incorporates the latest changes in terms of design and the first round of development characteristics for the WP4 components: zero-touch security orchestrator, security resource orchestrator, risk-averse resource management, programmable monitoring platform, semantic-aware anomaly detection, rule-based anomaly detection, threat predictor, and threat detector.

In this sense, the early prototype implementations are aligned with the design guidelines reported in previous deliverables, as well as new updates reported in this document, by following a continuous development process. The results of continuous developments and functional tests have been performed by multiple software teams, taking into account the validation feedback collected from WP6, where these components will be integrated together to showcase real scenarios.

This document is organised to describe:

- The latest architecture design for the components associated with the WP4 pillars: security service and resource orchestration, programmable pervasive monitoring, threat/anomaly detection and prediction, and closed-loop-based security automation.
- The main functionalities and software components that have been implemented inside each one of the WP4 pillars.
- An early software prototype implementation details for each developed component, including the results of functional tests to validate its deployment and internal communication flows as part of the forthcoming ROBUST-6G Zero-Touch Security Platform.
- References to the major prototype components and the respective source code, software tools, libraries, APIs, and the following standards.
- Results of validation components by including functional tests for the various modules that a component can develop to demonstrate the proper functionality before being integrated with other components.

In conclusion, this deliverable serves as a critical resource for advancing the understanding of the ROBUST-6G Zero-Touch Security Platform, considering the considerations and discussions of the last eight months of the project. The insights and recommendations provided in this report are essential for concluding the design of WP4 and driving the current technical phases, including development (WP4) and integration (WP6), to deliver a unified prototype platform.

Table of Contents

List of Tables

List of Figures

Acronyms and abbreviations

Term	Description
ADMM	Alternating Direction Method of Multipliers
AI	Artificial Intelligence
AoCE	Age of Consecutive Error
AoI	Age of Information
APIs	Application Programming Interfaces
CM	Catalogue Manager
CPT	Cumulative Prospect Theory
GAN	Generative Adversarial Network
IR	Incident Response
LCM	Life Cycle Management
LR	Lagrangian Relaxation
MID	Machine ID
ML	Machine Learning
NBI	Northbound Interface
OM	Ontology Manager
PAV	Pool Adjacent Violators
PMP	Programmable Monitoring Platform
PSM	Projected Subgradient Method
QoE	Quality of Experience
QoS	Quality of Service
RAN	Radio Access Network
SBI	Southbound Interface
SCA	Successive Convex Approximation
SCM	Security Context Manager
S-CL Mgmt	Security Closed Loops Management
SLO	Security Level Objective
SOAR	Security Orchestration, Automation and Response
S-RO	Security Resource Orchestration
SSLA	Security Service Level Agreement
SQP	Sequential Quadratic Programming
VIM	Virtual Infrastrucutre Manager
ZTSP	Zero Touch Security Platform
ZTSO	Zero Touch Security Platform

Introduction

1.1 Scope and Objective

6G networks aim to introduce higher automation and security levels in the cutting-edge solutions being deployed under scenarios fully decentralised and heterogeneous. Nevertheless, automation and security can be understood as two complementary dimensions that depend on each other. In this vein, the use of Artificial Intelligence (AI) techniques may be considered as a technology to introduce smart decisions into conventional solutions, as well as reduce human interaction in the lifecycle of a solution, e.g., the deployment of a network service across multiple stakeholders. Likewise, AI may also serve as an attack vector for intruders who want to introduce chaos in terms of resource or service disruption, monetary costs, or even safety. At this point, society may ask itself if we want AI for security, which means the leverage of AI techniques to proactively predict threats, detect well-known attacks, or automate countermeasures based on learning, or if we want security for AI, which entails creating robust, trustworthy, and explainable AI solutions that deliver secure AI models. ROBUST-6G project aims at covering both approaches to secure the use of future 6G networks from all angles. In particular, this deliverable is going to put the spotlight on automation-driven solutions for securing 6G networks.

Therefore, the primary goal of WP4 is to build automation for secure 6G deployment across multiple stakeholders and administrative domains, minimising the human interaction in the lifecycle of network service provision, orchestration, and management, whilst also using AI techniques to predict and detect potential attacks. Thus, a set of security mechanisms is going to be described to ensure the enforcement, the monitoring, and the maintenance of security levels for the principal activities of 6G zero-touch network orchestration.

This deliverable reports the ROBUST-6G efforts conducted between M13 and M21, laying the foundations of an AI/ML-Driven Zero-Touch Security Management Platform. Concretely, the D4.3 (previously named as D4.2) presents the latest updates in terms of architecture designs of former components described in the first deliverable D4.1. Hence, D4.3 showcases a consolidated version of the component architecture designs. In addition, it also includes an initial prototype of multiple software components associated with the four pillars of the security platform: i) security service and resource orchestration, ii) programmable pervasive monitoring, iii) threat/anomaly detection and prediction, and iv) closed-loop-based security automation.

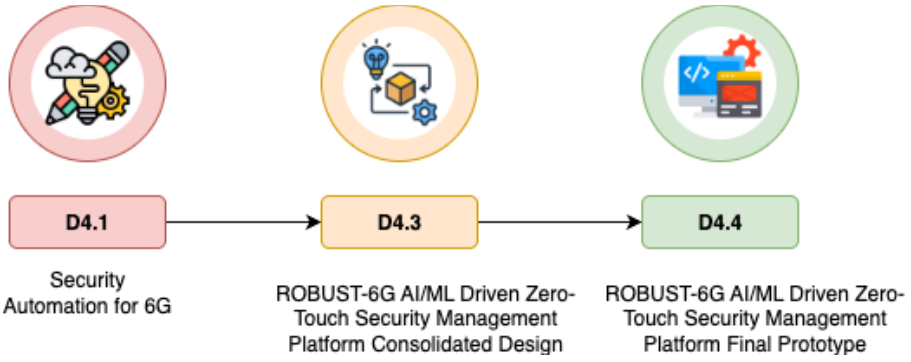


Figure 0-1: Progression of Work Package 4 deliverables

As Figure 0-1 illustrates, D4.3 also aims to pave the way for an early prototype implementation, which will serve as the baseline for D4.4, delivered close to the end of the project (M27), as well as consolidating security capabilities for 6G network service orchestration.

1.2 Document Outline

The document is structured as follows: Section 2 provides the utmost important information regarding the high-level architecture updates, the description of new functionalities of each software component, and an initial prototype implementation. Since this deliverable is the first one dealing with implementation in WP4, Section 2 also gathers technical characteristics. This comprises public Application Programming Interfaces (APIs) of components, communication paradigms used to interact with them, specific technologies or tools used to cover the component's functionality, GitHub repositories containing source code, and methods supported per component, along with their input and output parameters. Such information is provided by all

subsections: Security Service and Resource Orchestration (2.1), Continuous Monitoring and Threat Detection (2.2), and Incident Prediction and Mitigation (2.3) under Section 2. Finally, Section 3 presents some concluding recaps based on ongoing activities conducted over the last few months, highlights the most significant achievements of the principal components, and outlines the next steps for WP4 toward the final Driven Zero-Touch Security Management Platform Prototype.

ROBUST-6G Zero-Touch Security Platform

The ROBUST-6G Zero-Touch Security Platform (ZTSP) architecture presented in [R6GD41] has evolved further in terms of its components and interaction with external ROBUST-6G components. As depicted in Figure 0-1, a new functionality has been added to support the Security Service Orchestration: Generative AI for Security Orchestration, Automation and Response (GenAI4SOAR). GenAI4SOAR enables the dynamic generation of remediation workflows by leveraging GenAI contextualisation and generalisation capabilities and also offers a human interface (chatbot) for generating and validating remediation workflows.

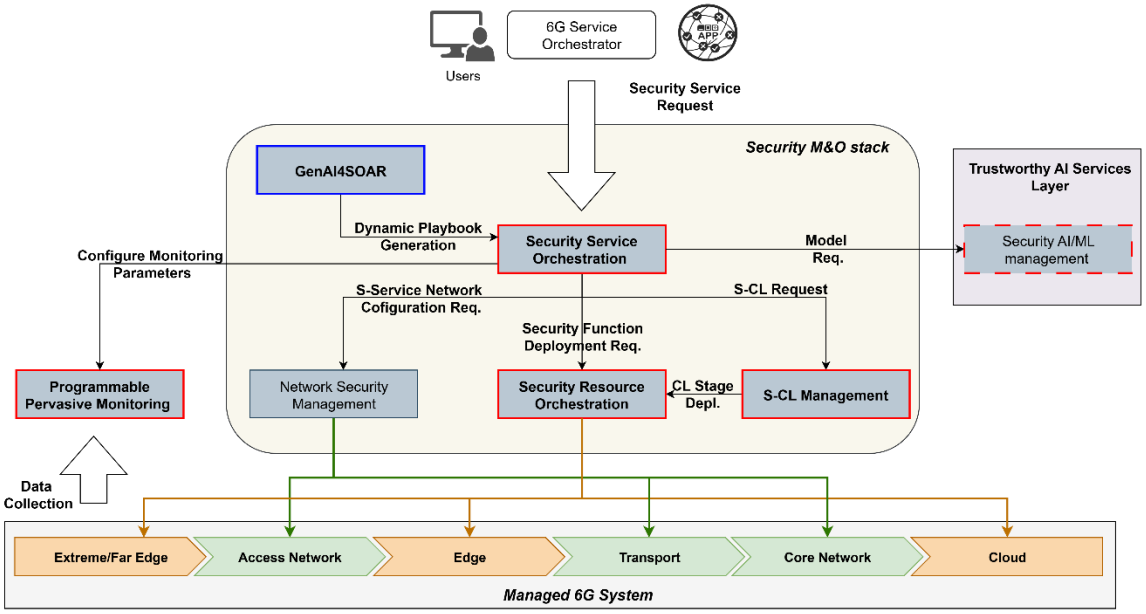


Figure 0-1: ROBUST-6G Zero-Touch Security Platform functional architecture: in red and blue functions discussed in this document; in blue functions integrated with respect to the latest functional architecture

The components described in this document correspond to the functionalities of the ZTSO already discussed in [R6GD41], and the following chapters present design updates and initial prototypes of these components, focusing on their implementation and integration aspects.

1.3 Security Service and Resource Orchestration

This section introduces the Security Service and Resource Orchestration layer of the ROBUST-6G Zero-Touch Security Platform. It describes how the Zero-Touch Security Orchestrator (ZTSO), Security Closed-Loop Management (S-CL Mgmt) and Security Resource Orchestrator (S-RO) work together to translate Security policies, in Security Service Level Agreements (SSLA) format, at the business level into actionable security functions establishing and maintaining an optimal security posture in a target infrastructure. As detailed in Section 1.3.1, the ZTSO operates at both the business and service levels: at the business level, it accepts security policies from verticals via the North-Bound Interface and interprets them by translating SSLAs into a set of Security Level Objectives (SLOs) metrics; at the service level, it composes infrastructure-specific security services, composed of security applications, configurations, and automations, that are deployed in the target infrastructure to enforce these policies. In order to implement these policies within the target infrastructure, the ZTSO coordinates two deployment/orchestration components: the S-RO and the S-CL Mgmt. As detailed in Section 1.3.3, the S-CL Mgmt is responsible for instantiating, orchestrating and coordinating Security Closed Loops (S-CLs). These are automation constructs that establish and maintain the security posture derived from a policy. As detailed in Section 1.3.4, the S-RO acts as the actuator across the cloud–edge continuum, deploying and operating the security applications and S-CL functions required by the

ZTSO's plans and by the S-CL management to establish and enforce security automations. The following sections provide a more detailed overview of the main functionalities, design updates and initial implementation of these components.

1.3.1 Zero-Touch Security Orchestrator

As described in Deliverable 4.1 [R6G24-D41], the Zero-Touch Security Orchestrator's (ZTSO) role within the ROBUST-6G Zero Touch Management Layer is to ensure that the security conditions and requirements agreed upon in Security Service Level Agreement (SSLA) between verticals and service providers are fulfilled through the orchestration and the management of security services. These services, deployed in vertical infrastructures, are composed of security applications, configurations, and closed-loop automations implemented through the ETSI ZSM concept of Closed Loop [ETS 002]. The deployed services are used to enforce and maintain a security posture in line with the NIST IRP R3 [800-61-R3] specification.

To align with the project's vision of leveraging AI to enhance and drive zero-touch security orchestration, the ZTSO is augmented with generative AI capabilities for automatic security playbook generation, and with semantic intelligence and reasoning to enforce security services. The ZTSO leverages symbolic AI and semantic reasoning to manage the intricate security orchestration ontology and knowledge graph, facilitating advanced knowledge representation and inferencing aligned with multi-domain scenarios. Complementing this, generative AI model enables the AI-augmented ZTSO to dynamically produce tailored playbooks for security remediations. This synergy empowers the ZTSO to continuously monitor, enforce, and adapt security conditions defined in the Security Service Level Agreements (SSLA) with minimal human intervention, thus realizing a truly zero-touch, intelligent security management paradigm.

Main functionalities

As reported in [R6G24-D41], the main functionalities of the ROBUST-6G ZTSO can be summarised as the three types of security orchestration it provides through the configuration and management of security services:

- **Proactive Security Orchestration**: it begins with an agreement in between the vertical and the service provider through a SSLA, and results in the establishment of a security posture and the deployment of a security service, composed of security applications, configurations, and security automations. These automations are configured with remediation workflows tailored for the vertical infrastructure, in order to satisfy the SSLA.
- **Reactive Security Orchestration**: defines the Incident Response (IR) Lifecycle part, starting from the detection of new or uncovered security threats that are not addressed by security services deployed during proactive orchestration. It encompasses the analysis and handling of such threats, and results in the deployment of new security services or the calibration/adaptation of existing ones, enabling the system to automatically respond to them in the future.
- **Predictive Security Orchestration**: defines the IR Lifecycle part, starting from the prediction of potential security threats that are not yet addressed by security services deployed during proactive orchestration. It encompasses the anticipation and analysis of such threats, and results in the deployment of new security services or the calibration/adaptation of existing ones, enabling the system to automatically prevent or mitigate them in the future.

Design updates

With respect to the functional architecture provided in [R6G24-D41], the following modification to ZTSO functional architecture, depicted in Figure 0-1 were made:

- The Semantic Translation functionality has been renamed Security Context Management to better reflect its scope within the ZTSO. While Semantic Translation focused on converting high-level ontology output into infrastructure-specific functions and environments, Security Context Management accepts plan descriptions and configurations from Policy Management and converts them into infrastructure-specific security services, which are composed of security applications and automations. This is made possible by retrieving information on available security functions and target environments from the infrastructure, and on automation and application templates from the S-RO and CL-Mgmt. In essence, Security Context Management provides lifecycle management for security services, that are deployed and managed to enforce policies.

- The link between the Security Context Management and the Ontology was removed, and a link in between the Security Context Management and the Catalogue Management module has been added. This choice was made to make the decoupling between the abstract and tool/infrastructure specific logic of the ZTSO more explicit by letting the Security Context Management deal only with the infrastructure, environments, and functions in the catalogue and the plan description generated by the Policy Management module.
- In the domain-specific orchestrators layer, Security Orchestration, Automation and Response (SOAR) frameworks, augmented with generative AI, have been added: they enable the dynamic generation and enforcement of remediation workflows to maintain security policies or respond to security anomalies detection during the system's lifecycle.
- A direct link from the southbound interface to the Alert Management module has been added to indicate how this module deals with domain-specific alerts that comes from components that are outside the ZTSO.
- A direct link has been added between the Catalogue Management module, and the Northbound Interface has been added to show how verticals can access the Catalogue to onboard their infrastructures (e.g. Smart Building), environments (e.g. IoT environment), and proprietary security functions (e.g. deployed Firewall).
- A direct link between the Ontology and the Northbound Interface has been added to indicate how the security orchestration ontology and the related knowledge graph can be accessed. The ontology can be accessed by an admin to update the entities/relationships or to upload new entities and relationships.

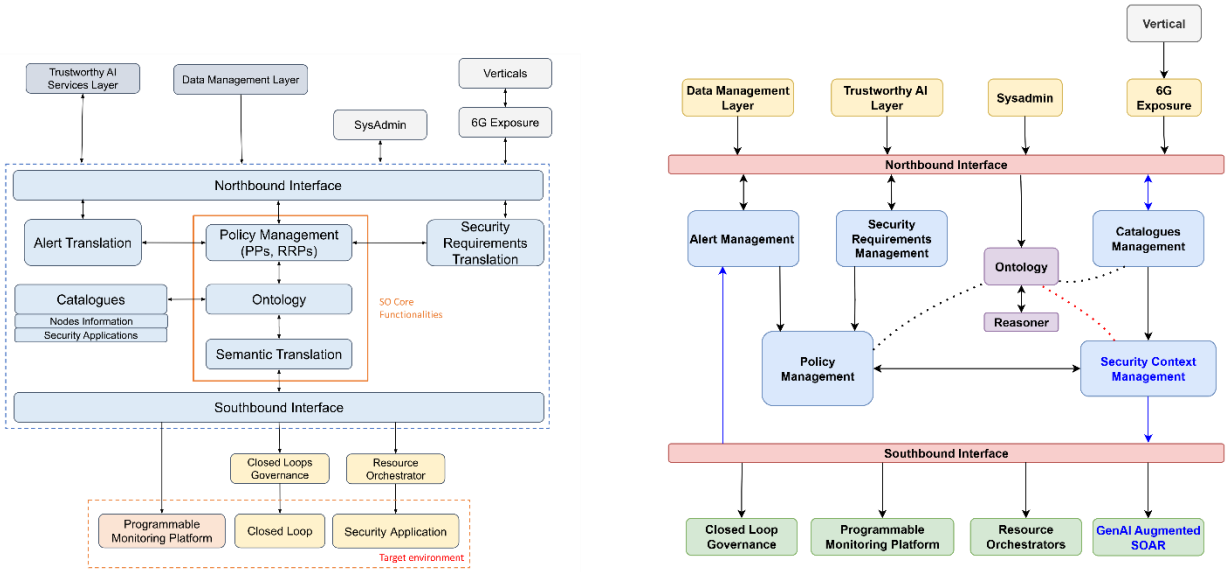


Figure 0-2 ZTSO: Functional Architecture (left: original architecture, right: updated architecture)

The updated functional architecture of Figure 0-2, with new components and links highlighted in blue and removed elements in red, serves as a foundation for the software component architecture that has been designed and is presented in Figure 0-3. This architecture defines all the software modules that are responsible for implementing the ZTSO's functionalities, along with their interconnections and the information they exchange both internally and externally on the NBI and SBI. An initial prototype implementation of some of these modules is provided in Section 0 While components such as the Policy Manager, SLLA Manager, Ontology Manager and Catalogue Manager already have defined interfaces, working prototypes and validated interaction with other components, others, as the GenAI Gateway, Alert Manager, and Security Context Manager, are still at an early stage of implementation although their design is mostly completed. For this reason, the different subsections of Section 0 include implementation details of each of the ZTSO modules.

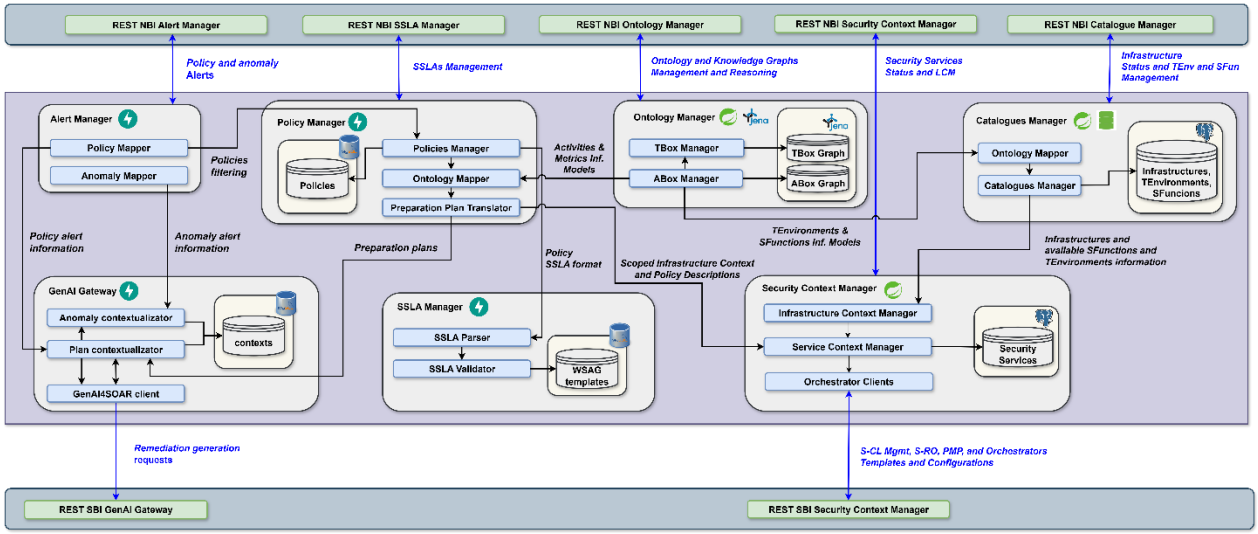


Figure 0-3: ZTSO Software Components Architecture

Initial Prototype Implementation

The initial implementation of the ZTSO started with the top layers, which govern the security orchestration logic, leading to the design, implementation, and integration of five microservices: Ontology Manager, Semantic Translator, Catalogue Manager, SSLA Manager, and Policy Manager. As previously discussed, these components already have defined interfaces, working prototypes, and validated interactions with other modules, while their finalized design and full implementation will be reported in Deliverable D4.4. Other modules, such as the GenAI Gateway, Alert Manager, and Security Context Manager, are still at an early stage of implementation, although their interfaces and interaction designs are discussed in this document.

The ZTSO software architecture follows a microservices-based approach, where each module corresponds to a service encapsulated in a Docker/Podman container, and the entire system is deployed and orchestrated in Kubernetes (K8s) using a Helm chart. In the following subsections, an in-depth description is provided for each implemented microservice, covering its technological stack, exposed APIs, communication paradigm, and other relevant details.

Policy Manager

The Policy Manager serves as the entry point for submitting and managing security policies (see Figure 0-4). These policies act as reference models for generating preparation plans that guide the decomposition and orchestration of security functions. On a technical perspective, the Policy Manager is a REST API developed in Golang, and packaged as a Linux Container. It can be deployed with container engines like Docker or Podman, or as a Kubernetes Pod using a Helm chart.

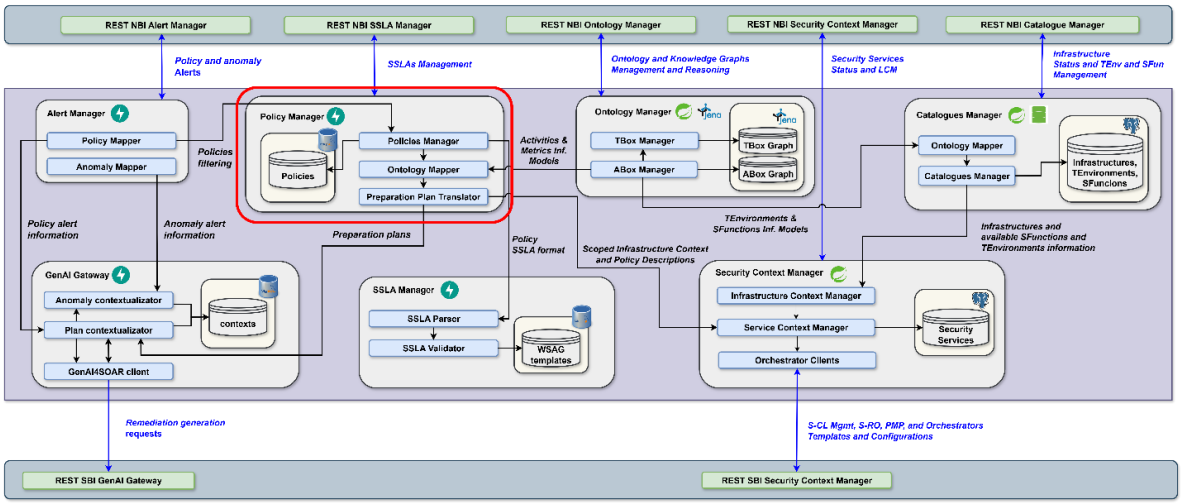


Figure 0-4: Policy Manager Role in the overall ZTSO architecture

Within the security orchestrator architecture, the policy manager is responsible for maintaining the knowledge base of the policy-driven security domain. It must interact with several core components to manage security policies and develop preparation plans for orchestration. The process of generating these preparation plans can be divided into six main steps as shown in Figure 0-5:

- Ingesting and exposing security policies via a northbound interface.
- Validating and parsing security policies using according to supported standards (e.g., SSLA format) to extract security requirements, such as Security Level Objectives (SLOs)
- Querying the *Ontology Manager* for available activities and their mappings to related security metrics.
- Creating the orchestration plan by mapping ontology activities, their supported metrics, and the security requirements for each metric
- Submitting the orchestration plan request to the *Security Context Manager* for further proactive preparation plan generation for the orchestration.
- Sending the preparation plan and security requirements to the GenAI Gateway to generate the appropriate remediation workflows for security violations

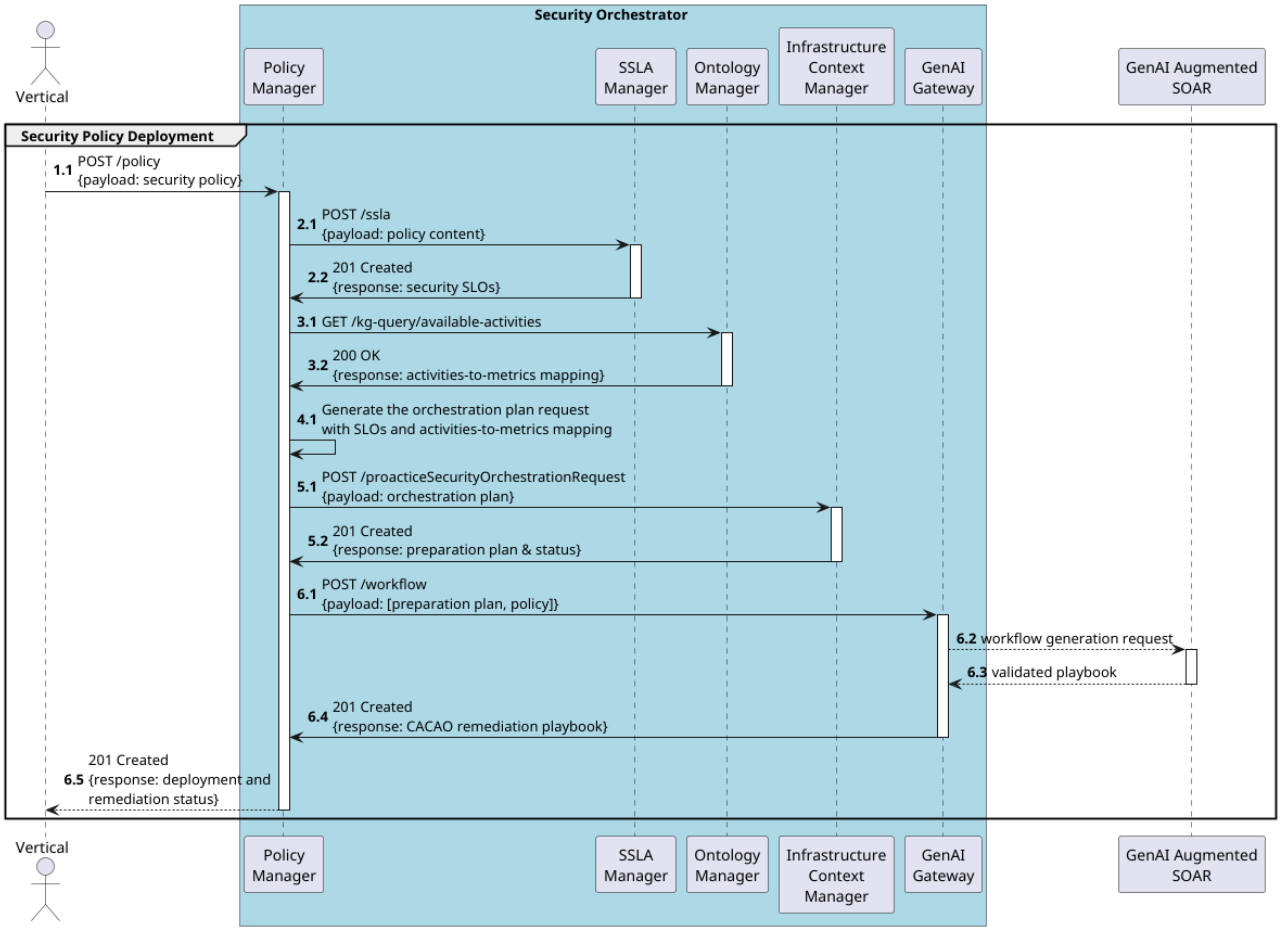


Figure 0-5: Sequence diagram of the policy manager's interactions with the other main security orchestrator's components

The OpenAPI specification of the policy manager API is described in Annex 0, and the following Table 0-1 showcases the API for uploading a new SSLA.

Table 0-1: Policy Manager API

Operation name: ***Submit a new SSLA to orchestrate***		
Description	Returns the result of the SSLA submission in a preparation form	
endpoint	/api/v1/ssla	
Method	POST	
Input Parameters	**Type**	**Description**
ssla	Content-Type: multipart/form-data	Content of the SSLA xml file
Output Parameters	**Type**	**Description**
201 Created	Content-Type: application/json	The resulting preparation plan generated from the SSLA
Notes		
Operation performed by the Policy Manager to transform an SSLA into a preparation plan, working with the SSLA Manager, the Ontology Manager and the Semantic Translator to get to this result		

SSLA Manager

The SSLA Manager is, represented in Figure 0-6, a standalone microservice designed to manage, parse, and validate security policies formatted according to the Security Service Level Agreement (SSLA) SPECS standard [specs15]. On a technical perspective, the SSLA Manager is a REST API developed in Python and packaged as a Linux Container. It can be deployed with container engines like Docker or Podman, or as a Kubernetes Pod using a Helm chart. Security policies in SSLA format can be submitted to the manager via the REST API. These policies are then stored in an SQL database for subsequent client queries.

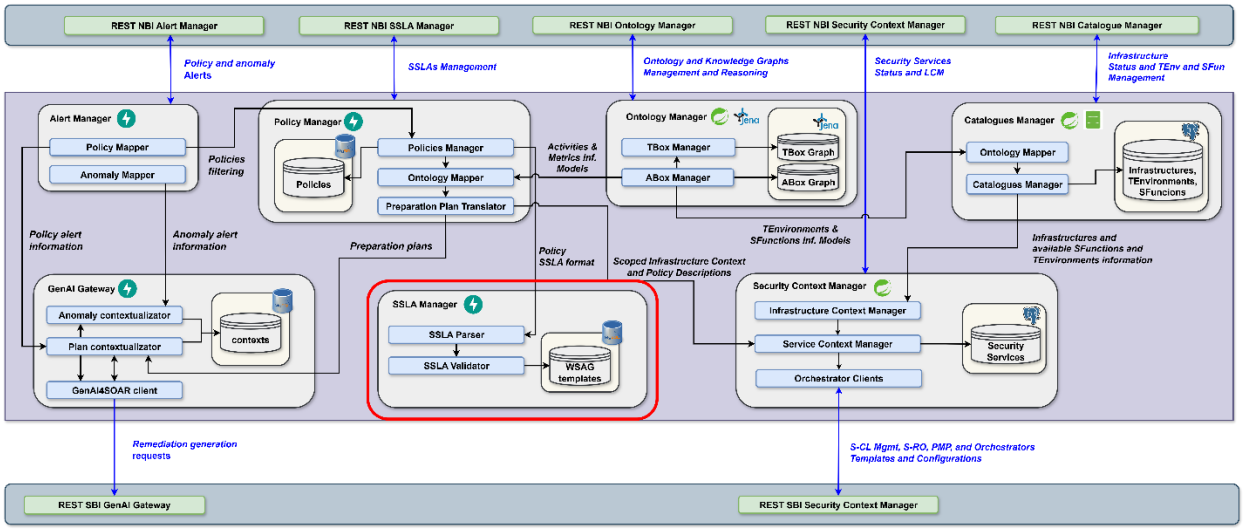


Figure 0-6: SSLA Manager Role in the overall ZTSO architecture

Each SSLA submitted to the manager is first validated against the Web Services Agreement specifications [Wsag11] using XML Schema Definition (XSD) files. Following validation, due to the complexity of the SSLA standard, the SSLA Manager operates as a parser to extract relevant information such as security metrics, security Service Level Objectives (SLOs), and security service properties associated with each agreement. The OpenAPI specification of the SSLA Manager is available in [ssla25a].

Ontology Manager

The Ontology Manager (OM) is the software component that manages the semantic layer of the ZTSO in terms of ontology, knowledge graphs, and reasoning (see Figure 0-7). The component was developed based on the division of description logic components into ABox and TBox, a management module for an ontology for security orchestration and a knowledge graph based on this ontology. The component is built using the Spring framework [SPR25] and its functionalities are based on the Apache Jena [JEN25] framework, the data managed by this component are stored in an external component, part of the same ZTSO Helm chart, that hosts a Jena Fuseki SPARQL Server [FUS25]. The OM exposes REST interfaces through several REST controllers and a Swagger interface.

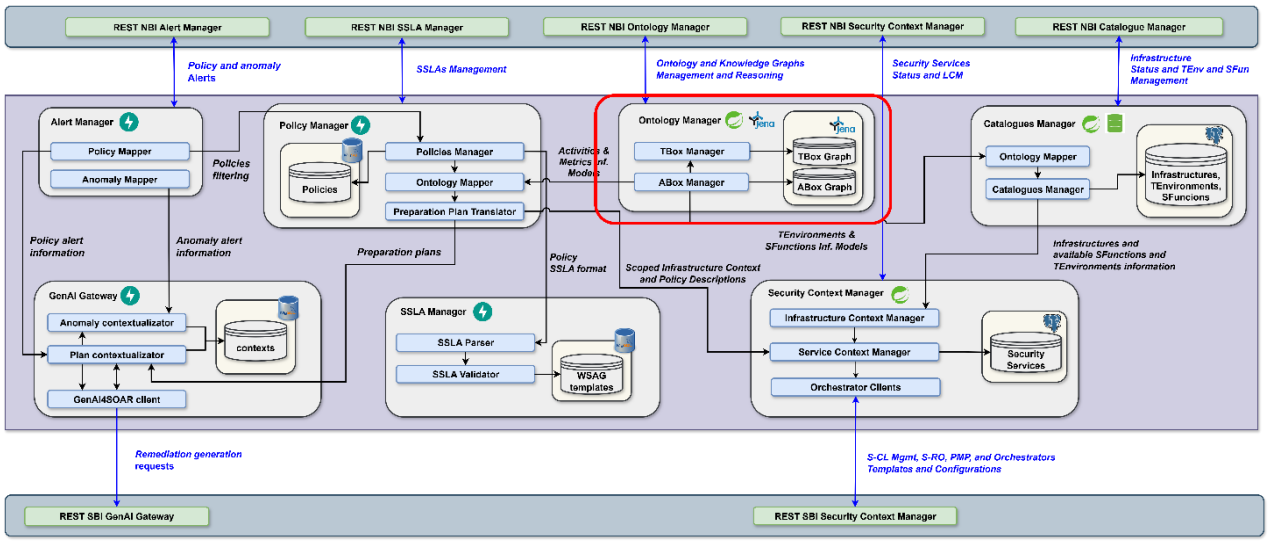


Figure 0-7: Ontology Manager Role in the overall ZTSO architecture

Beyond its technical implementation, the OM acts as the ZTSO's semantic engine. It provides an abstract representation of key concepts, such as security activities and functions, and target environments, which are connected by functional and non-functional capabilities. By decoupling these abstractions from specific security tools, platforms or frameworks, the OM provides a generalised, extensible foundation for describing the elements that comprise the ZTSO's security services. In this way, the OM plays a central role in supporting interoperability, reuse and reasoning across heterogeneous environments, ensuring the ZTSO can adapt and extend its orchestration capabilities flexibly. The security orchestration ontology has been defined using as a starting point the OnSOAP [IBN-19] Security Orchestration Ontology and extending it for a multi-domain scenario. The Classes, Relationships, and Rules of the ROBUST-6G Ontology for Security Orchestration are reported, respectively, in Table 0-2, Table 0-3, and Table 0-4. The OM GUI offers a visual representation of the ontology, which can be found in Annex 5.2.

Table 0-2: Security Orchestration Ontology - Entities

Name	Description
SecurityFunction	represents a generic security-related function, superclass of SecurityApplication and SecurityConfiguration.
SecurityApplication	specialized and deployable security application, subclass of SecurityFunction.
SecurityConfiguration	security configuration that can be applied to a security application, subclass of *SecurityFunction*.
TargetEnvironment	represents an environment where *SecurityFunctions* can be deployed.
PhysicalEnvironment	hardware-based environment like a physical server or network device, subclass of *TargetEnvironment*.
VirtualizedEnvironment	software-defined environment such as a virtual machine or container, subclass of *TargetEnvironment*.
Activity	task or operation within a security context.
Capability	general trait or feature supported by functions or environments.
FunctionalCapability	capability related to active functionality, subclass of *Capability*.
NonFunctionalCapability	capability referring to performance or quality aspects, subclass of *Capability*.
Metric	measurable property used to evaluate the behaviour or performance of *Activities* or *SecurityFunctions*.
Credentials	authentication or authorization data required to access a target environment.

Endpoint	network-accessible location for interacting with a target environment (e.g., IP address, URL).

Table 0-3: Security Orchestration Ontology - Relationships

Name	Description
suited_for_activity	links a *SecurityFunction* to an *Activity* it is suitable for.
suited_for_environment	links a *SecurityFunction* to a suitable *TargetEnvironment*.
has_non_functional_capability	links a *TargetEnvironment* to its *NonFunctionalCapabilities*.
has_functional_capability	links a *SecurityFunction* to the *FunctionalCapabilities* it provides.
requires_non_functional_capability	links a *SecurityFunction* to the *NonFunctionalCapabilities* it needs.
requires_functional_capability	links an *Activity* to the *NonFunctionalCapabilities* it needs.
can_be_measured_with	links an *Activity* to a *Metric* that can be used evaluate it.
has_access_endpoint	links a *TargetEnvironment* to its access *Endpoint*.

Table 0-4: Security Orchestration Ontology - Rules

Name	Rule
Rule_1	*[InferSuitedForActivity:* *(?app rdf:type ?type)* *(?type rdfs:subClassOf :SecurityFunction)* *(?app :has_functional_capability ?fc)* *(?act rdf:type :Activity)* *(?act :requires_functional_capability ?fc)* *->* *(?app :suited_for_activity ?act)* *]*
Rule_2	*[InferSuitedForEnvironment:* *(?app rdf:type :SecurityApplication)* *(?app :requires_non_functional_capability ?nfc)* *(?env :has_non_functional_capability ?nfc)* *->* *(?app :suited_for_environment ?env)* *]*

The complete list of APIs exposed by this component can be found in the OpenAPI descriptor available at [Nex25a]. A concise summary of the interfaces offered by the OM is provided in Table 0-5.

Table 0-5: Summary of APIS provided by the Ontology Manager

Interface	Consumers in the ZTSO	Description
TBox Management	ZTSO Admin	Provides administration functionalities for managing the ontology schema (TBox). They allow uploading, retrieving, and clearing the ontology definition in Turtle format. Through these operations, administrators can maintain and update the formal model of concepts, classes, and relationships that act as the foundation for validating and reasoning over the ABox knowledge graph.
ABox Management	ZTSO Admin	Enables the management of instance-level knowledge (ABox) within the security orchestration knowledge graph. They support the creation and deletion of entities and relationships, as well as the execution of reasoning rules to infer new knowledge. All operations are validated against the TBox ontology to ensure semantic consistency between the schema and its instances.

Data Models Management	Catalogue Manager, Policy Manager	Provide access to semantically enriched data models required by other ZTSO components. They expose structured representations of security functions, capabilities, metrics, activities, and target environments defined in the knowledge graph. These services are consumed by modules such as the Catalogue Manager and SSLA Manager to load, validate, and exploit the knowledge necessary for orchestration and data modelling.

Catalogues Manager

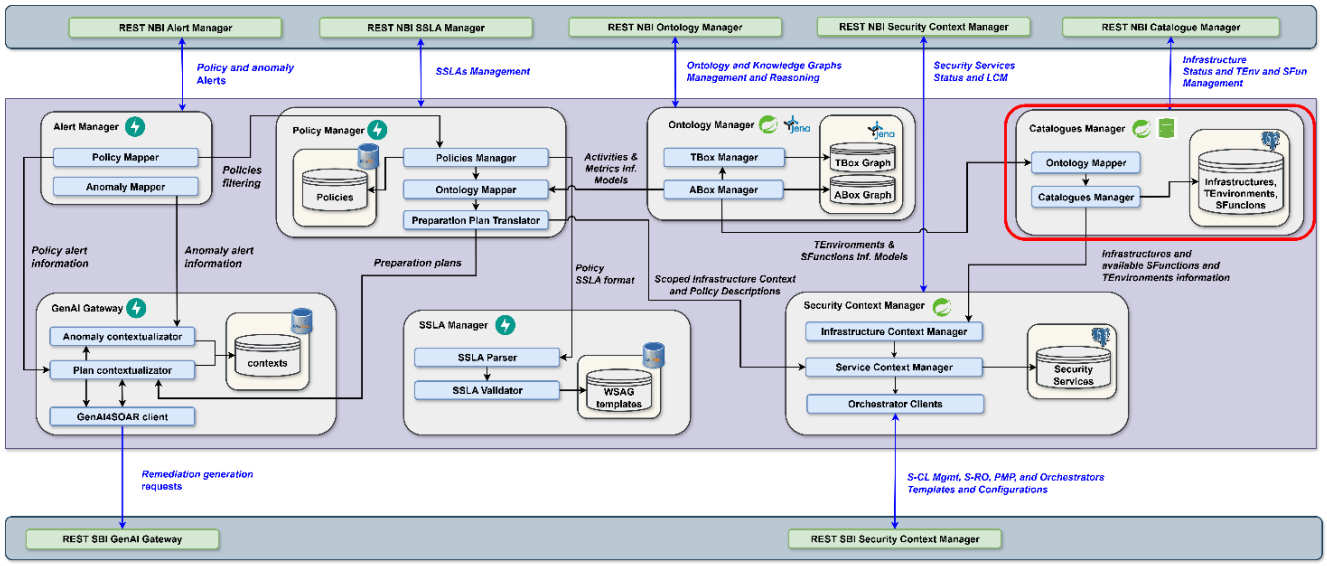


Figure 0-8: Catalogues Manager Role in the overall ZTSO architecture

The Catalogues Manager (CM) is the software component responsible for managing all the information about the infrastructures handled by the ZTSO (see Figure 0-8), in terms of available environments for the deployment of security functions, security functions that are already deployed, as well as those that are available for deployment. The data model of the environments and security functions is derived from the semantic model defined in the Ontology Manager, and the ontology is used as a support to link security functions to the security activities that can be performed by them. The component is built using the Spring framework and Spring JPA for data modelling. The data managed by this component is persisted in a PostgreSQL database deployed as part of the same ZTSO Helm chart. The Catalogues Manager exposes REST interfaces through a dedicated REST controller, enabling integration with the other ZTSO components for security orchestration operations. The CM's internal data model consists of three main entities: Infrastructures, Environments, and Security Functions and the particular attributes of these entities are retrieved at runtime from the ZTSO-TBOX using the dedicated Ontology Manager endpoints.

- *Infrastructures*: abstract concept representing a domain whose Security has to be managed by the ZTSO. It is composed of several Environments, each of them hosting multiple Security Functions.
  - examples: Smart Building, a Streaming Provider, etc.
- *Environments*: abstract concept representing something that can host a Security Function. An environment can be registered as part of different Infrastructures.
  - examples: Kubernetes Cluster, a Linux Server, SNORT, etc.
- *Security Function*: abstract concept that represent an actual security tool, service, or configuration that can be deployed in an Environment. A Security Function can be linked to different Environments.
  - examples: IDS, IP table rule, SNORT rule, etc.

The complete list of APIs exposed by this component can be found in the OpenAPI descriptor available at [Nex25b]. A concise summary of the interfaces offered by the CM is provided in Table 0-6.

Table 0-6: Summary of APIS provided by the Catalogue Manager

Interface	Consumers in the ZTSO	Description
Security Function Management	ZTSO Admin, Security Context Manager	Provides functionalities to onboard, query, and manage Security Functions. Admins can register and delete Security Functions, while the Security Context Manager retrieves their details or lists them to support security services composition.
Target Environments Management	ZTSO Admin, Security Context Manager	Provides functionalities to onboard, query, and manage Environments. Admins can register or delete Environments, while the Security Context Manager queries and lists them to support security services composition. It also supports linking and unlinking Security Functions to/from specific Environments, representing their deployment status.
Infrastructure Management	ZTSO Admin, Security Context Manager	Provides functionalities to onboard, query, and manage Infrastructures. Admins can register or delete Infrastructures, while the Security Context Manager queries to discover available environments and available/deployable security functions to compose security services. It also supports listing all Environments belonging to an Infrastructure, and linking/unlinking them as needed.

Security Context Manager

The Security Context Manager (SCM) is the software component responsible for managing the lifecycle of security services, which are composed of security applications and automations (see Figure 0-9). On the Southbound Interface (SBI), the SCM collects available orchestration templates from the S-CL Management and S-RO modules, which are then used to assemble security services in the form of applications and automations. On the Northbound Interface (NBI), the SCM exposes Service Lifecycle Management functionalities, enabling consumers to query the status of a particular service, stop it, or remove it. Although such operations could potentially affect SSLA agreements, they are necessary to support controlled management actions such as migration, planned shutdown, or maintenance without violating the SSLA.

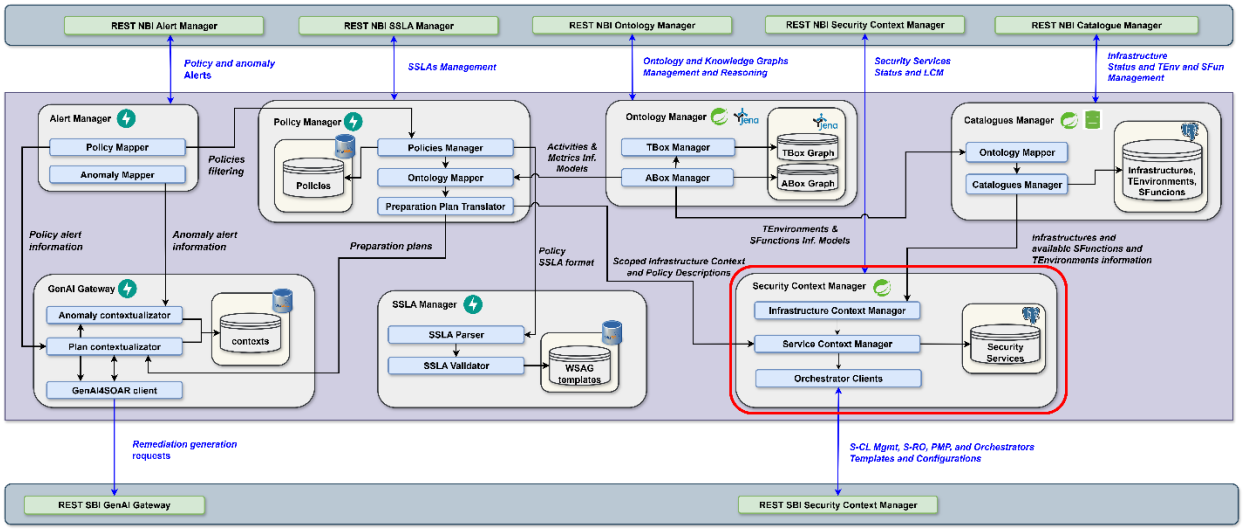


Figure 0-9: Security Context Manager Role in the overall ZTSO architecture

The SCM interacts with the Policy Manager through a two-step process. In the first step, it provides to the Policy Manager a scoped view of the infrastructure, including available security services, deployed and deployable security functions, target environments, and orchestration templates. This scoped context is then used by the Policy Manager, in conjunction with the GenAI Gateway and GenAI4SOAR, to build remediation plans tailored to the actual infrastructure status. In the second step, once the remediation plan has been

prepared, the Policy Manager submits the policy in the form of a preparation plan and remediation actions. The SCM uses this information, together with the scoped context and orchestration templates, to compose the concrete security service, deploying the required applications and automations.

From an implementation perspective, the SCM is still at an early design stage. A first prototype has been developed using the Spring framework, deployed as part of the ZTSO Helm chart, and relying on Spring StateMachine to maintain internal status during interactions with other components. The state is persisted in a PostgreSQL database, deployed within the same chart. At this stage, the only implemented API is the one toward the Catalogue Manager for retrieving the scoped context of an infrastructure. However, a concise summary of the interfaces offered by the SCM is provided in Table 0-7.

Table 0-7: Summary of APIs provided by the Security Context Manager

Interface	Consumers in the ZTSO	Description
Infrastrucure Context Retrieval	Policy Manager	Provides infrastructure context, including available services, deployed and deployable security functions, target environments, and active services. Consumed by the Policy Manager to prepare remediation plans.
Policy Ingestion & Service Assembly	Policy Manager	Accepts policies in the form of preparation plans and remediation actions. Uses them, together with orchestration templates and scoped context, to compose security services made of applications and automations.
Service Lifecycle Management	Policy Manager, ZTSO Admin	Exposes LCM functionalities to query the status of a security service, stop it, or remove it.

GenAI Gateway

As represented in

Figure 0-10, the GenAI Gateway is a component responsible for converting preparation plans, security requirements, and additional contextual information into coherent textual descriptions suitable for interaction with Generative AI (GenAI) assistants. Its primary role is to transform detailed, low-level data from the *Policy Manager* into high-level, natural language representations optimized for Large Language Models (LLMs).

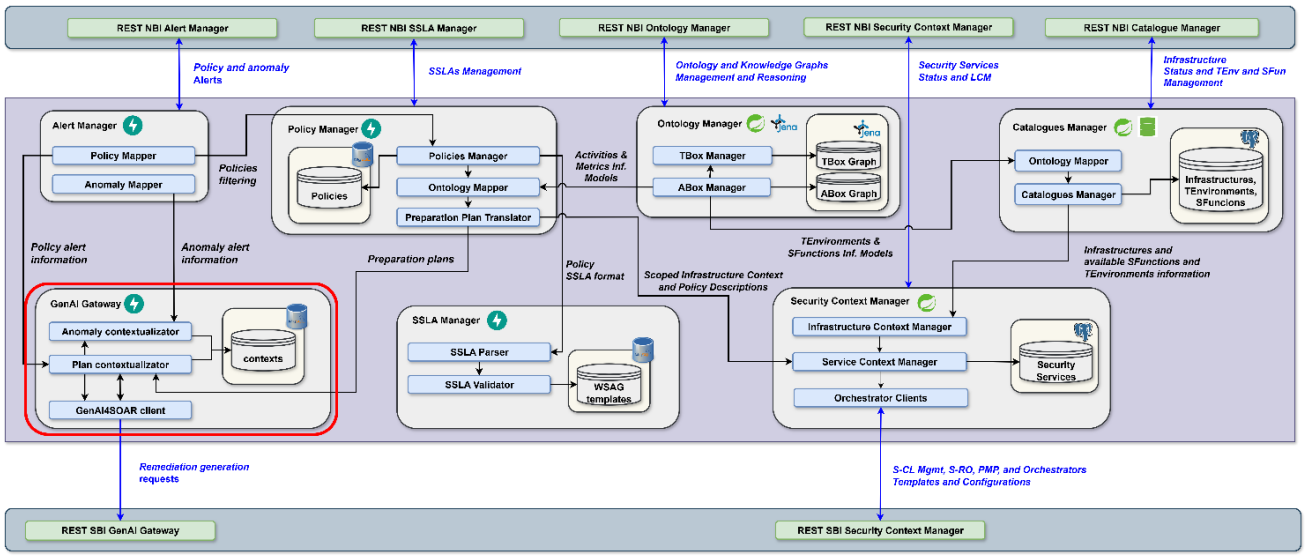


Figure 0-10: GenAI Gateway role in the overall ZTSO architecture

Initially, this translation process employs prompt injection techniques, where security policies, preparation plans, and contextual insights are interpreted and reformulated into clear, well-structured text that GenAI systems can effectively understand and utilize.

More advanced approaches aimed at improving the GenAI assistant's situational awareness will be explored in future development iterations.

With a generic, modular design, the GenAI Gateway is developed using plugins, to address a wide range of GenAI assistants, including those specializing in cybersecurity, threat analysis, and security contextualization. This component is still in an early development phase but will aim at a micro-service implementation in Golang, exposing a REST API (see Table 0-8), acting as a reverse proxy towards third-party AI assistants. It will be packaged as a Linux container and provided with a Helm chart for Kubernetes deployments.

Table 0-8: GenAI API

Operation name: new remediation prompt		
Description	*Submit new prompt elements to generate a remediation playbook*	
endpoint	/api/v1/playbook	
Method	POST	
Input Parameters	**Type**	**Description**
policy	Content-Type: multipart/form-data	Optional, content a security policy file (e.g. SSLA file)
plan	Content-Type: multipart/form-data	Optional, content of an orchestration plan (preparation, reactive, predictive)
catalog	Content-Type: multipart/form-data	Optional, list of available security functions or activities that can be part of the remediation
context	"Content-Type: text/plain"	Mandatory, contextual information to generate the remediation playbook
Output Parameters	**Type**	**Description**
201 Created	"Content-Type: application/json"	The resulting prompt that has been sent to the SOAR. The response payload contains the CACAO playbook.
Notes		

Alert Manager

The Alert Manager receives security alerts through the northbound interface (see Figure 0-11). These alerts are typically generated by SIEM with correlation rules based on security logs from monitoring systems, intrusion detection systems, or other security tools that identify potential security incidents. Upon receiving an alert, the Alert Manager queries a dedicated datastore that maintains mappings between alerts and corresponding remediation playbooks. This alert-playbook mapping is essential to quickly determine if there is a predefined remediation workflow available for the incoming alert.

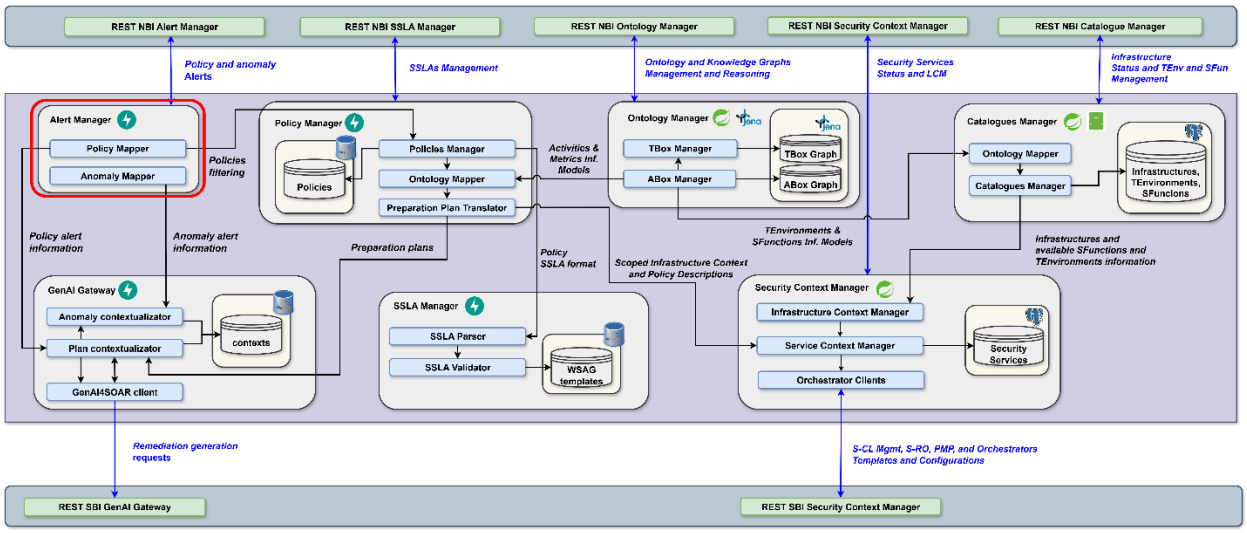


Figure 0-11: Alert Manager Role in the overall ZTSO architecture

If a corresponding playbook is already bound to the alert (i.e., the mapping exists), the Alert Manager proceeds with associating that playbook ID with the alert. If no playbook is found for the alert, the Alert Manager forwards the alert data to the GenAI Gateway. This component leverages generative AI models to dynamically create a new remediation workflow tailored to the alert specifics.

Once the playbook ID is determined (either from the datastore or by generation), the Alert Manager sends the alert along with the playbook ID to the SOAR platform. The SOAR system is responsible for enforcing then executing the remediation workflow defined by the playbook.

This component is still in an early development phase but will aim at a micro-service implementation in Golang, exposing a REST API (see Table 0-9) to receive alerts from analytics systems. It will be packaged as a Linux container and provided with a Helm chart for Kubernetes deployments.

Table 0-9: Alert Manager in ZTSO API

Operation name: raise a security alert to the ZTSO		
Description	*Submit a security alert to mitigate*	
endpoint	/api/v1/alert	
Method	POST	
Input Parameters	**Type**	**Description**
content	"Content-Type: application/json"	Content of the alert embedding details and information necessary for the remdiation
Output Parameters	**Type**	**Description**
201 Created	"Content-Type: application/json"	Acknowledgement of the alert
Notes		

Functional Tests

Several tests have been performed to verify the functionality of each of the ZTSO components and their communication with the internal modules and external components.

Table 0-10: ZTSO functional tests

Name	Description	Passed (Yes/No/Partially)
ZTSO Module setup	The module can be automatically installed and configured through HELM.	Yes

Ontology Management	The module correctly ingests and validates Ontology files in .ttl format through the Ontology Manager NBI.	Yes
Knowledge Graph Management	The module supports the instantiation of new entities and relationships in the Knowledge Graphs validating them to the existing ontology.	Yes
Reasoning	The module supports reasoning to populate the knowledge graph based on inference rules available in the ontology.	Yes
SF Catalogue Management	The module supports the creation of security functions to be deployed in a target environment. Each function is described by functional and non-functional capabilities as well as a list of metrics. Such information is validated against the ontology through the Ontology management interfaces.	Yes
TE Catalogue Management	The module supports the creation of environments to be deployed in an infrastructure. Each environment is described by non-functional capabilities which are validated against the ontology through the Ontology management interfaces. Additionally, this module allows to append inside an existing environment a set of security functions.	Yes
Inf Catalogue Management	The module supports the creation of infrastructures. Each infrastructure has an owner and a set of environments (and security functions as well). In particular, this module allows to append inside an existing infrastructure, a set of environments.	Yes
Information model retrieval	Policy Manager and Catalogue Manager can retrieve the Information Models from the Ontology Manager Knowledge Graph. Respectively, the Catalogue Manager can retrieve Security Applications and Target Environments while the Policy Manager can retrieve the Available activities and the Metrics that can be used to measure them.	Yes
Context retrieval	The Security Context Manager can retrieve a scoped context of the infrastructure from the Catalogue Manager which includes available Security Functions and Target Environments for a given Infrastructure.	Yes
SSLA Validation	The Security Policies, submitted through the northbound interface and via the Policy Manager, follow the SSLA format defined by the SPCES framework, and their content are validated.	Yes
SLO Decomposition	The Security Level Objectives (SLO) are properly extracted from a security policy and bound to metrics and security activities from the ontology manager.	Yes
Preparation Plan generation	An orchestration plan containing SLOs, metrics and security activities generates a preparation plan composed with security functions matching the security activities and fulfilling the SLOs of the preparation plan.	Yes

1.3.2 GenAI4SOAR

Main Functionalities

Streamlining Smart Security Remediations with Generative AI

To address the challenges of Security Orchestration, Automation and Response (SOAR) frameworks, detailed in the security orchestration state of the art in [R6G24-D22], we propose the use of generative AI models fine-tuned with standardized remediation playbook formats and processes to dynamically generate security remediation workflows. This approach enhances human-machine collaboration by providing a generative AI assistant called *GenAI4SOAR* that:

- Assists experts in efficiently reviewing and modifying generated playbooks with an assistant

- Generates playbooks that can support a wide range of technologies and plug them together to improve the interoperability and the time to generate such playbooks
- Generates playbooks that comply with *CACAOv2* [CACAO23] standard format. Moreover, the *OpenC2* [OpenC222] standard for technological interoperability is used in playbooks, alongside *CACAOv2,* to supports the wide range of security components and technologies to create a comprehensive playbook workflow

In the ETSI ZSM's closed loop perspective, this solution takes part in both the "decide" step for intelligence and the "act" step for orchestration and control. Indeed, the genAI assistant functionality helps as an intelligence to provide specific decisions and recommendations, prior to human validation, in the form of playbooks that can be enforced in SOAR frameworks, and executed in real-time, as workflows, for orchestration and control in case of security alerts.

CACAOv2 standard for remediation playbook format

Collaborative Automated Course of Action Operations (CACAO) standard from the OASIS, in its version 2, defines the schema and taxonomy to help organizations to detect, investigate, prevent, mitigate, and remediate threats by forming a cyber security playbook (see Figure 0-12). Its format is made to be shared in a structured and standardized way across organizational boundaries and technological solutions to improve their Cyber Threat Intelligence (CTI) common knowledge. For example, a CACAO playbook may reference other playbooks in such a manner that allows composition from smaller, more specific functional playbooks shared across different organizations.

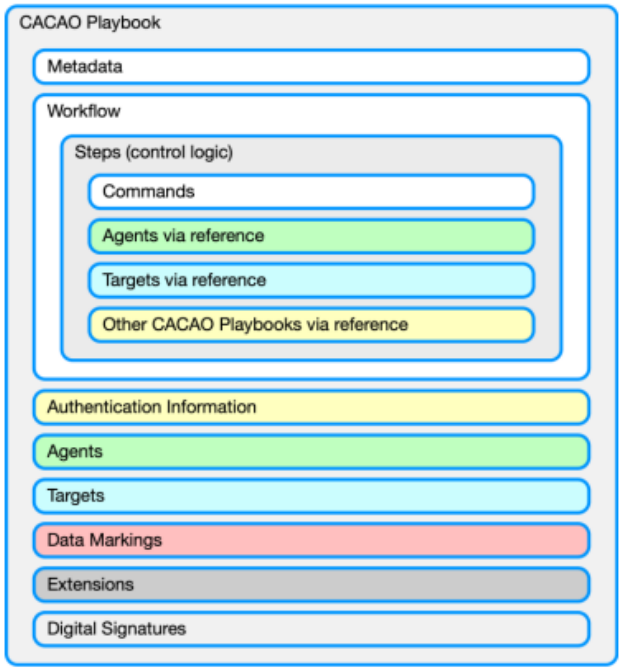


Figure 0-12: Overview of CACAO Playbook Structure and Classes of Objects

The *Workflow* class of CACAO playbooks for security orchestration contains a set of steps (security actions) to be performed based on a logical process and may be executed ad-hoc, periodically, or triggered by an automated or manual event or an observation. This class aligns well with our goal of establishing a common standard for workflow execution in SOAR frameworks. Leveraging generative AI, by training or fine-tuning it to create CACAO playbooks at scale, will significantly reduce the mean time to respond to security incidents, while maintaining compatibility with a wide variety of SOAR implementations for effective enforcement in real-world security orchestration environments.

However, the target types for CACAO workflow steps are specific to the technologies used to implement the workflow actions, such as *Elasticsearch [Oc2Elastic23]* or *Jupyter Notebook [Oc2Jupyter23]*. This aspect of the CACAO standardization is a weakness in CTI interoperability, as it may pressure organizations to adopt only the technologies supported by CACAO. Additionally, it poses a significant risk of rapid obsolescence for CACAO playbooks due to ongoing changes in the technology landscape.

Fortunately, the CACAO standard also supports the *OpenC2-HTTP* command [Oc2Http23] type, which aligns with another OASIS standard designed for technological interoperability. For the remainder of this study, we will exclusively use this command type (see example in Figure 0-13).

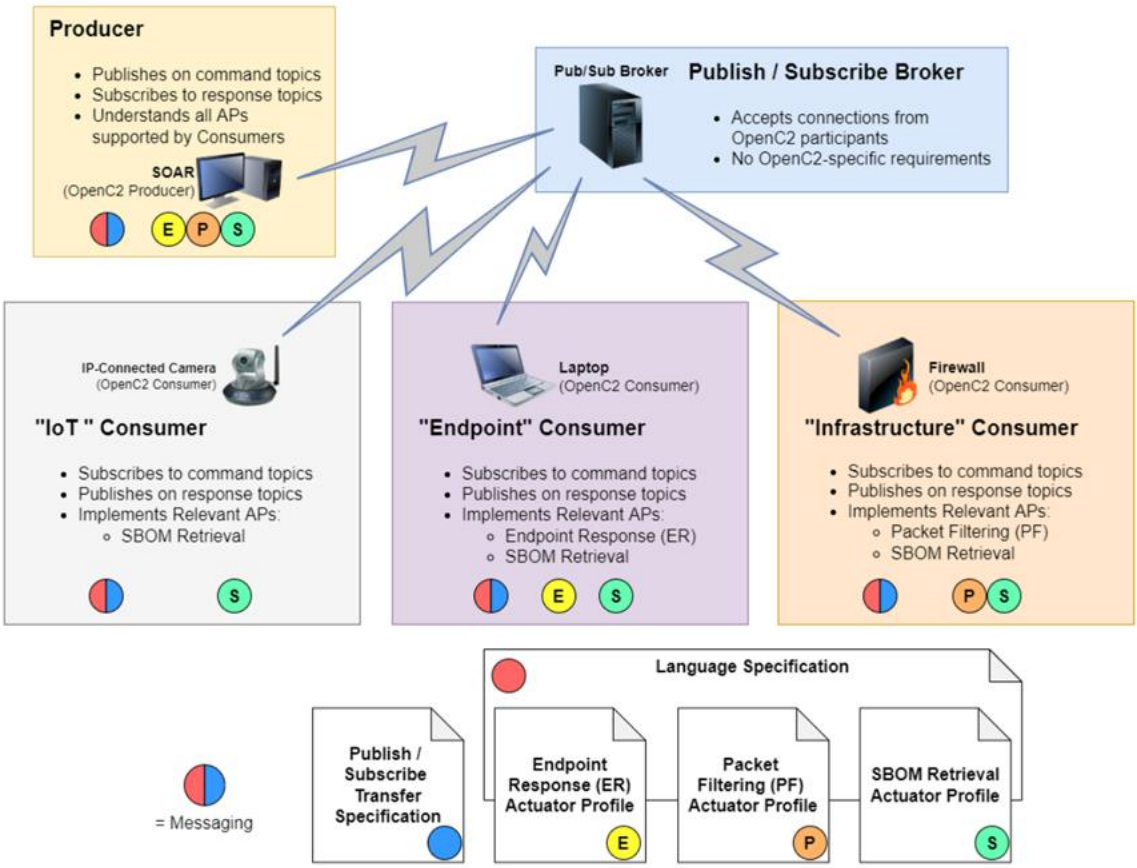
```
{
    "type": "openc2-http",
    "command": "POST /api1/newObjects/ HTTP/1.1",
    "content_b64": "<openc2_content_b64>",
    "headers": {
        "Content-Type": [
            "application/openc2+json;version=1.0"
        ]
    }
}
```

Figure 0-13: Example of an openc2-http command for CACAO encoded in JSON

OpenC2 standard for technological interoperability

Open Command and Control (OpenC2) is a concise and extensible language to enable machine-to-machine communications for cyber security components in a manner that is agnostic of the underlying technologies, transport mechanisms or other aspects of the implementation.

It consists in a publish-subscribe protocol with producers and consumers sending or receiving commands through a broker for messages transfer (see



).

Figure 0-14: Component view of the OpenC2 publish-subscribe protocol

OpenC2 allows the application producing the commands to discover the set of capabilities supported by the managed devices. These capabilities allow the managing application to adjust its behaviour to take advantage of the features exposed by the managed device. The capability definitions can be easily extended in a

noncentralized manner, allowing standard and non-standard capabilities to be defined with semantic and syntactic rigor.

From the perspective of our study and considering the CACAOv2 standard, which supports OpenC2-HTTP command types for remediation actions within playbook workflows, we can abstract these actions and send them to a remediation broker capable of distributing them across multiple execution environments. Within the ROBUST-6G architecture, this means we only need a single type of producer for remediation actions and one type of consumer for each resource orchestrator in the system. This approach not only makes remediation workflows abstract, interoperable, and flexible for ROBUST-6G but also allows other architectures and projects to reuse them, provided they adopt the same concept.

The example below showcases the payload of an openc2-http command, embedded in a CACAOv2 playbook and encoded in JSON (see Figure 0-15). It is meant to be sent to an OpenC2 consumer of a firewall to deny connections originating from a target IPv4 address and port, towards a targeted IPv4 address and port destination.

```
{
    "headers": {
        "request_id": "d1ac0489-ed51-4345-9175-f3078f30afe5",
        "created": 1545257700000,
        "from": "oc2producer.company.net",
        "to": [
            "oc2consumer.company.net"
        ]
    },
    "body": {
        "openc2": {
            "request": {
                "action": "deny",
                "target": {
                    "ipv4_connection": {
                        "protocol": "tcp",
                        "src_addr": "1.2.3.4",
                        "src_port": 10996,
                        "dst_addr": "198.2.3.4",
                        "dst_port": 80
                    }
                },
                "args": {
                    "start_time": 1534775460000,
                    "duration": 500,
                    "response_requested": "ack",
                    "slpf": {
                        "drop_process": "none"
                    }
                },
                "profile": "slpf"
            }
        }
    }
}
```


Figure 0-15: Example of an openc2-http command

Creating CACAO playbooks can be a complex and demanding task for humans. Cybersecurity teams must ensure compliance with the standards, follow the correct encoding syntax, design the workflow logic according to the specific incident to be remediated, sequence the actions appropriately, address the relevant technologies, and, in this context, compose the action content itself in accordance with the OpenC2 standard, which carries its own complexity. While feasible for experts, this process can be time-consuming and effort-costing, potentially increasing the mean time to respond for cybersecurity teams responsible for implementing remediations. Additionally, these teams must manage numerous workflows, including existing ones, while handling conflicts or contradictions that may arise between them.

In this challenging environment, generative AI offers a powerful solution. By leveraging a generative AI assistant properly trained and fine-tuned for cybersecurity contexts, as well as for the CACAOv2 and OpenC2 standards, we can support cybersecurity teams by generating remediation playbooks that are compliant with CACAOv2 and OpenC2 at creation time. These playbooks can then be reviewed and validated by human experts, significantly reducing the time needed for their development. Moreover, Retrieval Augmented Generation (RAG) techniques can be employed over a catalogue of security functions to inject coherent OpenC2-HTTP actions content into the CACAO playbooks, providing properly formatted JSON commands ready for direct transmission to the OpenC2 broker. Finally, RAG can assist the AI assistant in proposing new workflows that avoid conflicts with existing ones, alert human experts if an incident is already covered by existing workflows or even use the playbook composition feature of CACAOv2 by referencing existing playbooks withing the new one, through effective database searches.

The example below showcases a CACAOv2 playbook in Figure 0-16, to remediate to the CVE-2023-4914 [CVE20234914]. This workflow has been generated with GenAI4SOAR and Mistral Large Instruct 2411.

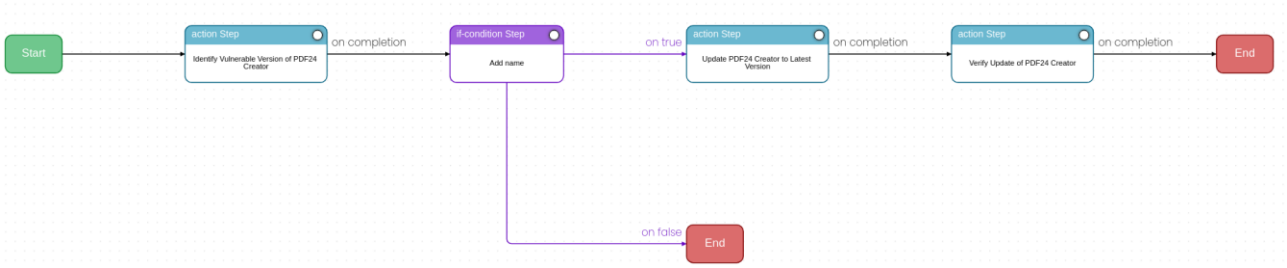


Figure 0-16: Example of a CACAOv2 playbook

In the context of security closed loops within the ROBUST-6G architecture for security orchestration, automatic playbook generation can be triggered in two scenarios:

- During the proactive orchestration phase, a security policy is submitted to the ZTSO via the northbound interface. This policy is translated into a preparation plan, which, along with additional high-level information and context, is sent to the *GenAI Gateway*. The output of this process is a CACAOv2 playbook designed to enforce and maintain the security policy components within the specified environment and infrastructure outlined in the preparation plan.
- During the reactive or predictive orchestration phases, an alert generated by an anomaly detection or prevision mechanisms is sent to the northbound interface of the security orchestrator. This alert, containing low-level data, is enriched with preparation plans and contextual security policies relevant to the infrastructure and environment involved. The enhanced alert is then forwarded to the *GenAI Gateway* component. The generation process produces a *CACAOv2* playbook aimed at remediating the anomaly or addressing the prevision that triggered the alert. This playbook is immediately ready for enforcement and can be executed to mitigate the current alert.

Our approach for dynamically generating the security remediation playbooks for proactive and reactive (or predictive) phases, and using generative AI within the Security Orchestration Layer, is managed by several components:

- The ZTSO component processes security policies or alerts, transforming them into high-level narratives and contextual information regarding the infrastructure and execution environment. In return, it receives CACAO playbooks designed to maintain security policies or respond to security incidents, which the Security Orchestrator can then forward to the S-CL Manager.
- The GenAI4SOAR component leverages a large language model (LLM) to create CACAOv2 playbooks for security remediations. These playbooks are generated based on narratives derived from security policies or alerts, details about the infrastructure and execution environment, and an automatic prompt engineering process to add CACAOv2 and OpenC2 specification contexts.
- The S-CL Mgmt component includes S-CL Function templates for the execution of S-CL Functions that can translate CACAOv2 playbooks into domain-specific workflows tailored for the SOAR solutions it supports.
- The SOAR component framework executes the final domain-specific workflows for security remediation and monitor security alerts to trigger these remediation actions.

The role of the ZTSO component in generating remediation workflows is fulfilled through a set of interconnected subcomponents:

- The Policy Manager subcomponent, detailed in Section 0, creates the preparation plan that outlines security functions, activities, the applicable security policy, and information about the infrastructure and execution environment.
- The Alert Manager subcomponent handles real-time anomaly alerts, a specific subset of security alerts that trigger immediate playbook generation and remediation for the detected anomaly. For the generation process only:
  - It extracts infrastructure and environment information from the alert and forwards this data along with anomaly details to the GenAI Gateway component.
  - As a results, it receives the identifier of the generated playbook and records it alongside the alert data in a datastore. This mapping facilitates triggering the correct remediation workflow if a similar anomaly alert arises in the future, thereby enhancing the ZTSO's continuous improvement capabilities for security anomalies.
- The GenAI Gateway subcomponent converts low-level technical data into high-level, contextualized narratives:
  - When provided with a new security policy request, along with the preparation plan and infrastructure/environment information, it stores these in a datastore keyed by the policy ID.
  - When provided with a new security alert originating from an anomaly detection, it uses environment and infrastructure identifiers to retrieve relevant data from the datastore and associates it with any existing policies.
  - In all cases, this component transforms the collected information into a high-level narrative intended for an LLM assistant to generate the necessary playbooks, covering both cases for policy posture maintenance and security mitigations.

The GenAI4SOAR component's role in generating remediation workflows is also accomplished through a combination of subcomponents:

- The CACAO Contextualizor subcomponent serves as an API Gateway that applies a prompt engineering logic to each incoming query. Specifically, it retrieves fine-tuned CACAOv2 and OpenC2 language specifications from a datastore and incorporates them as a prompt within the playbook generation requests it processes. It then forwards the original narrative, now wrapped with these CACAOv2 and OpenC2 prompts, to a large language model (LLM) to generate the playbook. The datastore managing the fine-tuned language specifications supports ongoing updates and maintenance of these specifications. This subcomponent also employs a Retrieval-Augmented Generation (RAG) approach using a datastore that holds existing CACAOv2 playbooks and containing OpenC2 commands. This setup allows the algorithm to either create new playbooks when none exist, retrieve existing playbooks, or combine multiple existing playbooks into a new one, leveraging CACAOv2's modularity. Each newly generated playbook is stored in the database to continuously enhance the knowledge base used for retrieval and generation.
- The LLM Model subcomponent consists of a vLLM instance deployed on premise, or an LLM remote service.

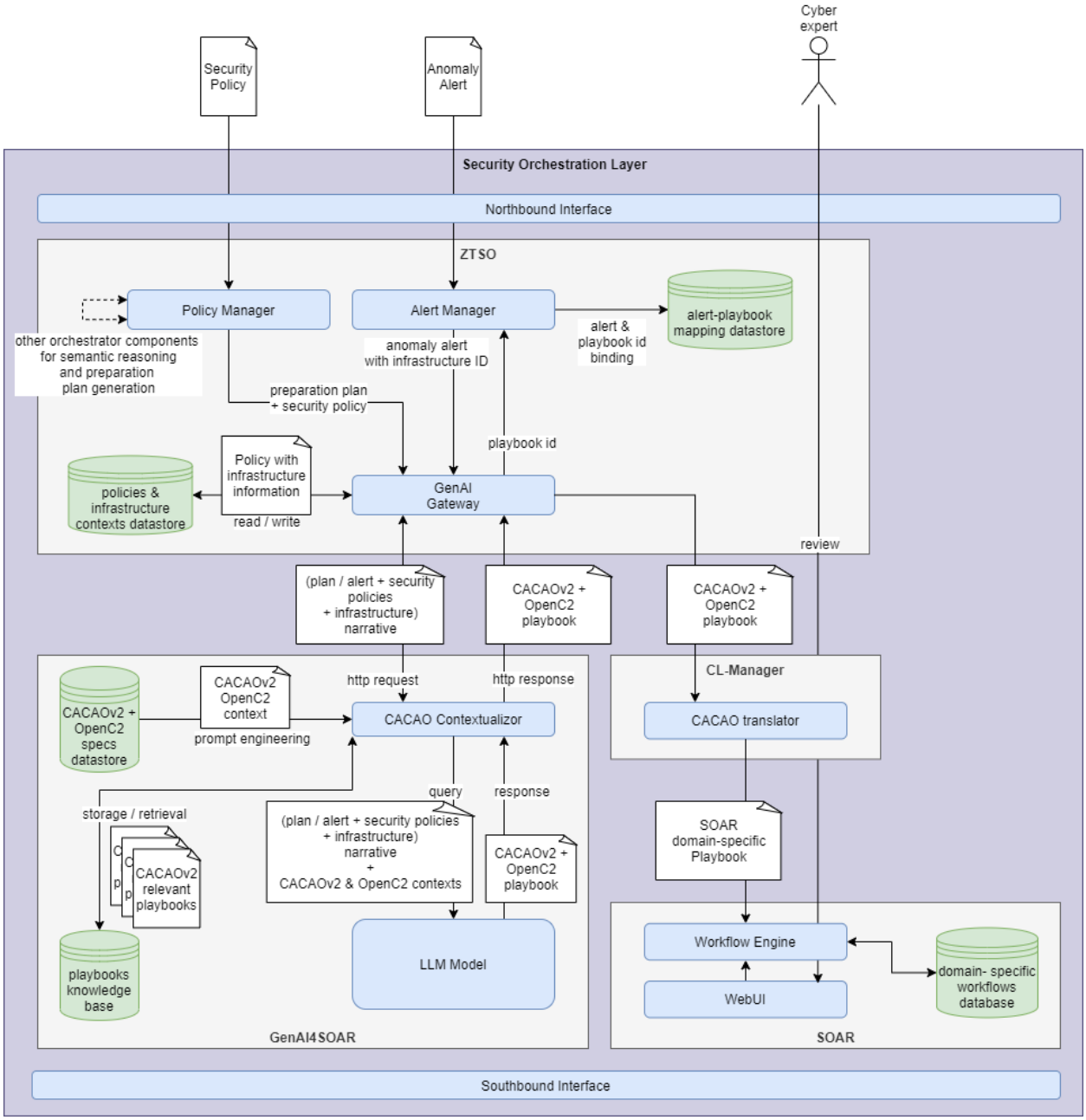


Figure 0-17: Component view of the process for dynamic generation of security remediation workflows using generative AI

In any of the three scenarios for proactive, reactive, or predictive orchestration, the outcome is CACAOv2 playbooks that require enforcement and execution through a SOAR framework. The benefit of utilizing the CACAOv2 standard lies in its ability to maintain compatibility across different SOAR implementations, ensuring agnosticism.

In the end, the CACAOv2 playbooks are sent to the S-CL Manager for translation and configuration according to the closed-loops characteristics being currently implemented. The workflows can be reviewed, validated or modified by cybersecurity experts using SOAR web interfaces.

Human in the loop for AI decisions over high-risk systems

SOAR frameworks serve as the primary interface between AI-driven decisions and human oversight. While human intervention is optional, it remains crucial to have it readily accessible throughout the lifecycle of the orchestrated systems.

The European Union, through the AI-ACT [aiact24], mandates "Human Oversight" when AI systems make decisions affecting high-risk environments. The security automation solution proposed in ROBUST-6G is designed to be general-purpose, applicable to a wide array of systems including those classified as high-risk.

Additionally, human involvement is supported by ETSI ZSM 009-1 [zsm21a], which emphasizes that although Closed-Loop Automation (CLA) aims to minimize direct human intervention, it is essential that autonomous systems still allow human operators to interact with and ultimately approve or reject actions taken by the AI. In light of this, the ROBUST-6G Zero-Touch Security Platform is planned to be a configurable, fully automated security solution where system administrators retain the option, under their responsibility, to disable zero-touch automation and engage themselves in the decision-making loop to approve or deny AI-based security actions.

Ultimately, ROBUST-6G's policy-driven approach to security states that the security intents for closed-loop automation originate from humans, such as through 6G systems security governance, and these intents are validated by humans as well. Although AI drives all intermediate steps in automating security closed loops, human validation ensures control and accountability. This approach aligns closely with the vision expressed in ETSI GR ZSM 011 v2.1.1 [zsm24a] on zero-touch network and service management and intent-driven autonomous networks.

Initial Prototype Implementation

We would like to draw the reader's attention to the fact that the system described in the previous section is still in the early stages of development. As a result, we are not yet able to provide complete details of its implementation.

More precisely, the CACAO contextualizor and CACAO translator components are currently at an early development stage. Their architectures, interfaces, and technical specifications are being defined to ensure seamless integration within the overall system.

In contrast, the SOAR functionality will leverage the existing open-source solutions, such as Shuffle, and the LLM will be met using an off-the-shelf solution, such as the Mistral Large Instruct model.

CACAO contextualizor

The CACAO contextualizor is the component responsible for prompt engineering for high-level narratives enrichment, embedding preparation plans, security policies and deployment infrastructure information, with CACAOv2 and OpenC2 languages specifications.

Using the narratives as inputs, it proceeds with the following computing steps:

- Accesses a dedicated datastore to retrieve fine-tuned specifications of the CACAOv2 and OpenC2 languages. These specifications are kept up-to-date within the datastore to ensure accurate and current language models are used.
- Applies prompt engineering logic by combining the retrieved CACAOv2 and OpenC2 language specifications with the original narrative. This process effectively creates a contextualized prompt that guides the large language model (LLM) on how to generate the playbook according to these domain-specific standards and commands.

As an output, it forwards this enhanced, prompt-wrapped narrative to the LLM model of the GenAI4SOAR component. The LLM then uses this input to generate a remediation playbook that complies with the CACAOv2 and OpenC2 standards, enabling subsequent automated workflow execution.

Table 0-11: CACAO contextualizor API

Operation name: new remediation prompt	
Description	*Submit new prompt elements to generate a remediation playbook*
endpoint	/api/v1/playbook
Method	POST

Input Parameters	Type	Description
prompt	"Content-Type: text/plain"	Text of a high-level narrative containing a preparation plan, a security policy or a security alert and the deployment infrastructure information
Output Parameters	**Type**	**Description**
201 Created	"Content-Type: application/json"	The CACAOv2 playbook embedding the OpenC2 commands to maintain the security policy or respond to the security alert in the infrastructure.
Notes		

CACAO Translator

The CACAO translator is a REST API designed to convert validated CACAOv2 playbooks into SOAR-compatible workflow definitions.

The translator will interact with the SOAR's API to import converted workflows and validate them to remain compliant with its specific workflow language, supporting error handling for format compatibility.

SOAR

In the context of ROBUST-6G, the SOAR serves the following key purposes:

- Enforce remediation workflows with a dedicated engine and execute them when they are triggered by an input. In our case, the input for the remediation workflow is always an alert.
- Facilitate human-machine interaction by incorporating humans into the decision process for high-risk situations, specifically by enabling workflow reviews before execution or by requiring validation of remediation actions that result from workflow execution.

The remediation workflows come embedded in CACAOv2 playbooks generated by the GenAI4SOAR component. To accommodate the unique characteristics of various SOAR implementations, each potentially employing its own workflow language, the CL-Manager converts the CACAOv2 playbooks into the specific language used by the target SOAR solution. After this translation, the workflows are loaded into the SOAR platform where they become available for human review and execution.

When an alert is received through the northbound interface, the Alert Manager processes it and checks the alert-playbook mapping datastore to determine if a playbook is already associated with this alert. If no playbook is linked, the Alert Manager forwards the alert data to the GenAI Gateway to generate a new remediation workflow. Regardless of this, the alert is then passed to the SOAR system along with the corresponding playbook ID for execution. The execution of the workflow produces one or more OpenC2 commands that represent the remediation actions. These commands are sent through an OpenC2 producer specifically designed for the SOAR solution responsible for carrying out the remediation.

In accordance with the OpenC2 standard protocol, the remediation actions are published to an OpenC2 broker (see Figure 0-18). Each Resource Orchestrator within the ROBUST-6G Security Orchestration Layer has a dedicated OpenC2 consumer subscribed to this broker. These consumers listen for actions applicable to their orchestration domain and are responsible for translating the received OpenC2 commands into the specific language of their orchestration environment to implement the necessary remediation.

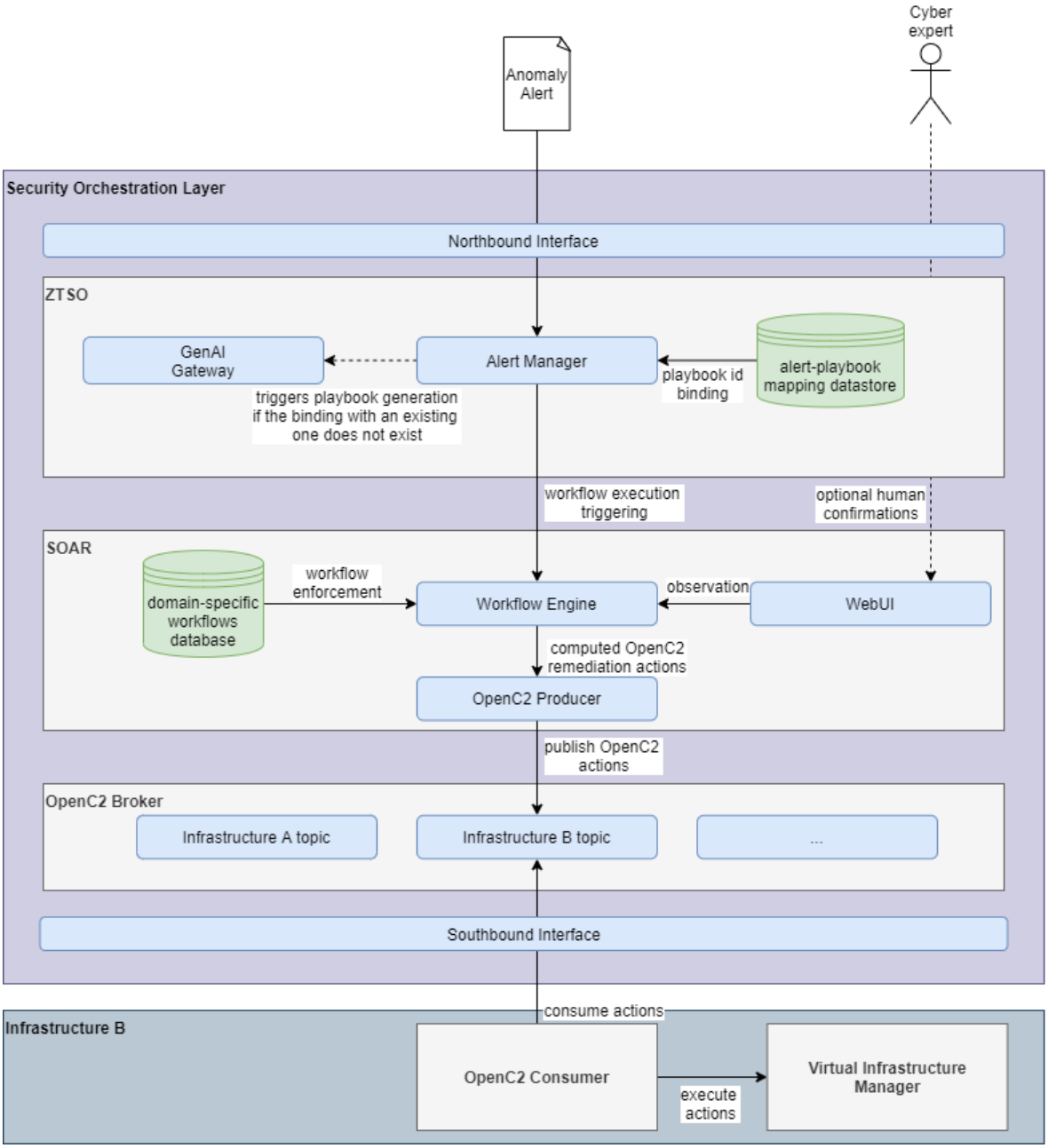


Figure 0-18: Component view of the process of workflow execution for dynamic security remediations

The execution of workflows involves less overhead and intermediate processing compared to the generation process. This streamlined approach helps to minimize the response time, measured from when an alert is triggered to when the Virtual Infrastructure Manager (VIM) successfully mitigates the security issue.

1.3.3 Security Closed Loop Management

Deliverable 4.1 [R6G24-D41] introduced Security Closed-Loop Management (S-CLMgmt) as the element of the Management and Orchestration (M&O) stack of the ROBUST-6G Zero-Touch Security Platform that is responsible for orchestrating and coordinating security automations. These automations, leveraging the ETSI ZSM framework, are implemented through the four classical stages of CL-Monitoring, CL-Analysis, CL-Decision, CL- Execution, supported by a transversal CL-Knowledge function.

Main Functionalities

The main functionalities of the S-CL Mgmt are to instantiate, orchestrate and coordinate Security Closed Loops (S-CLs), acting as a bridge between high-level security automation logic of the ZTSO and the execution of automated security actions. It receives inputs from the ZTSO, which defines the security automations resulting from its orchestration activities that needs to be instantiated preserve a security policy. Moreover, it interacts with the Security Resource Orchestration (S-RO) to provision the individual functions that compose a S-CL.

These functionalities are enabled by the concepts of *S-CL Descriptors* and *S-CL Function Descriptors*:

- ***S-CL Descriptors*** define the objectives, scope and operational policies of a S-CL. It specifies the main goals, the managed entities (i.e. Security Applications), the permitted operations over these entities, the S-CL Functions composing the S-CL and the runtime policies guiding its behavior (e.g. coordination with other loops or escalation rules).
- ***S-CL Function Descriptors*** provide detailed specifications of each loop's functional block (Monitoring, Analysis, Decision, Execution and Knowledge), including the software artifacts to be deployed and their required configuration to enable the automation.

Through this descriptor-based approach, the S-CLMgmt guarantees that closed-loops are instantiated in a consistent and interoperable way, aligned with the orchestration decisions of the ZTSO, and seamlessly integrated into the ROBUST-6G Zero-Touch Security Platform. In fact, the S-CL Descriptor provides the configuration template of the automation, specifying its objectives, scope, and functional composition, but it does not represent a fully implemented automation. To transform the automation template into an operational automation, inputs from the ZTSO are required: it is responsible for selecting the most suitable descriptor to satisfy an SSLA and for providing the concrete parameters needed to configure each of the required S-CL Functions. In this way, the ZTSO bridges the gap between the abstract loop definition in the catalogue and the security automation deployment.

Furthermore, the S-CLMgmt offers runtime governance and coordination capabilities for S-CLs, particularly when multiple loops operate concurrently or hierarchically across heterogeneous domains. In such cases, the S-CLMgmt component it's able to detect potential conflicts and arbitrates between alternative actions to ensure that security measures remain consistent and effective at the system level.

A complete example of S-CL and S-CL Functions descriptors can be found in Annex 3.3. From this example, the programmability and dynamicity of the S-CL approach to enabling security automations becomes evident. By configuring different parameters for the S-CL Functions, based on the security functions and applications available in the target infrastructure, the potential combinations of security automations are virtually infinite: in the *Monitoring* stage, the loop can be programmed to fetch data from the PMP, with its scope limited only by the monitoring capabilities exposed by the platform; in the *Analysis* stage, different algorithms can be selected to focus on specific aspects of the monitored data, constrained only by the variety of analytics functions available; in the *Decision* stage, the stage can be injected with domain and SSLA-tailored CACAO IR playbooks, each prescribing distinct remediation flows; in the *Execution* stage, enforcement actions can be triggered through available actuators, and, when necessary, this stage can also initiate the full orchestration cycle by invoking the ZTSO's North-Bound Interface and requesting the deployment of new security automation as well as the re-configuration of existing ones. The flexibility of this approach is enhanced by allowing execution stages to be tailored to specific technologies and SOAR platforms. For example, an execution stage could translate a CACAO playbook into OpenC2 commands (acting as the "producer" in the

OpenC2 architecture) or into platform-specific instructions, such as ThingsBoard [2] commands for the ThingsBoard IoT platform.

Design updates

The proposed S-CLMgmt Functional Architecture, depicted in Figure 0-19, illustrates the internal functionalities of the S-CLMgmt component together with the external modules it interacts with to enable S-CL based automations. The architecture is composed of the following functional blocks:

- **S-CL & S-CL Functions Descriptors Repository**: manages the onboarding, storage, and retrieval of S-CL and S-CL Functions descriptors. It enables the discoverability of S-CL templates from the ZTSO and provides the needed artifacts, together with their parameter's templates, to instantiate and configure a S-CL.
- **S-CL Life Cycle Management (LCM)**: provides the workflows for instantiating, updating, and terminating closed-loops. It maintains runtime information of active loops, supervise their execution, and coordinates with the ZTSO for S-CL management and with the S-RO for the deployment of S-CL Functions.
- **S-CL Coordination:** ensures consistency and coherence when multiple S-CL operate concurrently or hierarchically by detecting and resolving potential conflicts and managing coordination and dependencies between S-CLs.

At the orchestration level, the ZTSO connects to the S-CLMgmt via the North-Bound Interface to request the instantiation of closed-loops, to select the appropriate descriptors from the catalogue, and to provide the parameters required to configure each S-CL Function in line with the orchestration activities defined by the ZTSO. At the resource level, the S-RO consumes the outputs of the S-CLMgmt through the South-Bound Interface and is responsible for deploying and managing the functional components of each loop. Moreover, the active S-CL instances are connected to the S-CL LCM through the South-Bound interface, to provide analytics information about S-CL execution.

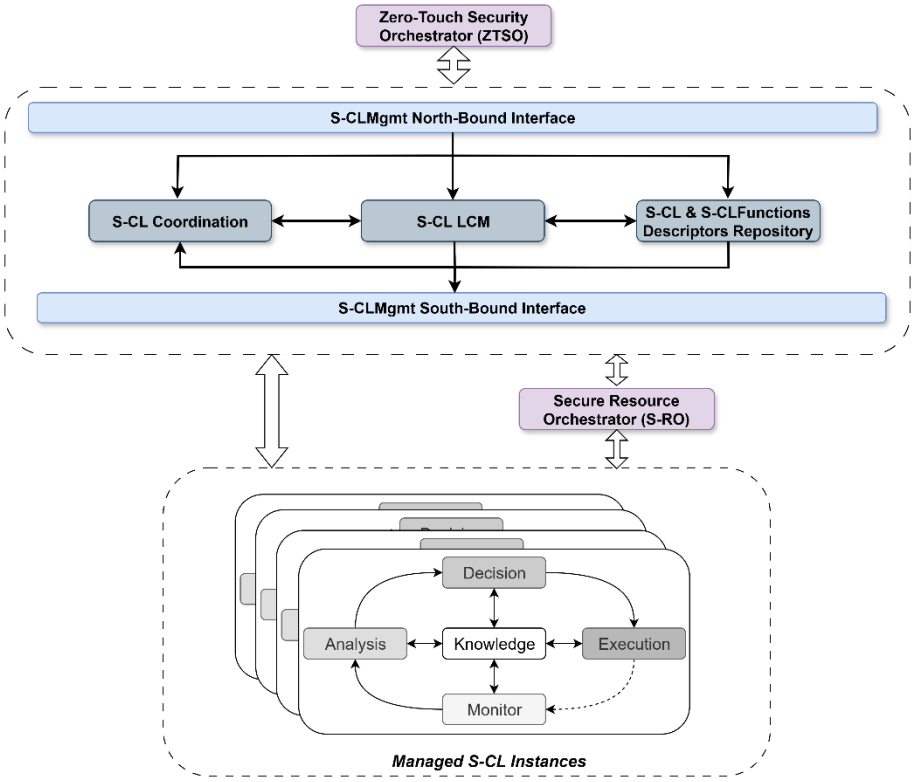


Figure 0-19: S-CLMgmt functional architecture

[2] https://thingsboard.io/

Initial Prototype Implementation

The software architecture of the S-CLMgmt is not directly reported in this deliverable, as it designs and early implementation follow the architecture of the Closed Loop LCM component developed in the HEXA-X-II project and described in [HXII-D63]. The complete list of APIs exposed by the S-CLMgmt module can be found in the OpenAPI descriptors available at [Nex25c] and [Nex25d]. In particular, the APIs of the CL Catalogue reported in [Nex25c] will be used as the baseline for the implementation of the S-CL and the S-CL Functions Descriptors Repository, while the APIs of the CL-LCM reported in [Nex25d] will serve as the baseline for the S-CL LCM implementation. An overall summary of the interfaces offered by the S-CLMgmt module is provided in Table 0-12.

Table 0-12: Summary of APIs provided by the Security Closed Loop Management

Interface	Consumers in the ZTSP	Description
Closed-Loop Lifecycle Management (LCM)	ZTSO	Provides lifecycle operations for closed-loops, including instantiation, update, and termination. Supports both synchronous and asynchronous calls.
Closed-Loop Instance Management	ZTSO	Enables the creation of new closed-loop instances, listing of active instances, and retrieval of detailed information for a specific instance.
Closed-Loop Function Management	ZTSO	Manages the internal functional blocks of a closed-loop (Monitoring, Analysis, Decision, Execution, Knowledge), with configuration and deployment details.
Error & Status Reporting	ZTSO	Provides standardised runtime and error reporting to ensure interoperability and proper error propagation across orchestrators and monitoring systems.

Functional Tests

Several tests have been performed to verify the functionality of the Security Closed Loop Management and its communication with the internal modules and external components.

Table 0-13: Security Closed Loop Management functional tests

Name	Description	Passed (Yes/No/Partially)
S-CL Mgmt setup	The module can be automatically installed and configured through HELM.	Yes
Closed-loop instances LCM	Create a Closed Loop instance, get the details of a specific instance, and list all existing CL instances to verify indexing and discoverability.	Yes
Closed-loop templates management	Upload a Closed Looptemplate, list all available templates, retrieve a specific template (by ID/name+version), and delete it.	Yes
CL function templates management	Onboard CL-functions templates (e.g., monitoring, analysis, execution functions), list them, retrieve details, and delete them.	Yes

1.3.4 Security Resource Orchestrator

As described in Deliverable 4.1 [R6G24-D41], the Security Resource Orchestrator's (S-RO) role within the ROBUST-6G Zero Touch Management Layer is the management and orchestration cloud-native application across the cloud edge-continuum to support security policy enforcement and security automation. In the overall ROBUST-6G Zero Touch Security Platform, the S-RO acts as an actuator for the ZTSO and the CL-Mgmt module by: (i) translating generic cloud application templates into infrastructure-specific deployment artifacts (ii) deploying and orchestrating cloud application that implement security services in accordance with specific security policies and (iii) by deploying and orchestrating cloud applications that realizes S-CL stages and enable security automation to preserve and enforce those policies.

Main Functionalities

As reported in Deliverable 4.1 [R6G24-D41], the S-RO, supports dynamic discovery, continuous monitoring, and orchestration of resources in multi-domain environments by providing the following functionalities:

- ***Resources Management***: the S-RO collects and manages resource information related to cloud platform in the cloud continuum. This information is consumed by the ZTSO or third-party algorithms to decide optimal resource allocation for specific applications or services.
- ***Cloud Applications LCM***: the S-RO performs lifecycle management of cloud applications that either implement security services in accordance with security policies or realize S-CL stages to enable security automation.
- ***Cloud Service Repository***: the S-RO maintains a cloud-native description of applications that can be remapped to specific cloud platforms for deployment.
- ***Platform Management***: the S-RO provides logic and interfaces to interact with different cloud platforms (e.g. Kubernetes, K3s, Openstack) through a set of plug-ins specific for the target environments.

As reported in the S-RO high-level architecture in [R6G24-D41], functionalities exposed by the S-RO are exposed by a unified Northbound Interface and consumed by two different entities: the ZTSO, which requests the deployment of cloud applications implementing security services in accordance with a security policy, and the CL-Mgmt, which requests the deployment of cloud applications realizing S-CL stages that enable zero-touch security automation to preserve established policies.

Initial Prototype Implementation

The software architecture of the S-RO is not directly reported in this deliverable, as it designs and early implementation follow the architecture of the multi-cluster extreme-edge resource orchestration component developed in the HEXA-X-II project and described in [HXII-D63]. The complete list of APIs exposed by the S-RO can be found in the OpenAPI descriptor available at [Nex25e], which reports the actual APIs of the component developed in HEXA-X-II and will serve as the starting point for the S-RO. A concise summary of the interfaces offered by the S-RO is provided in Table 0-14.

Table 0-14: Summary of APIs provided by the Security Resources Orchestrator

Interface	Consumers in the ZTSO	Description
Platform Management	S-RO Admin, ZTSO	Manages platform information and controls Security Applications. Includes listing, adding, and removing platform entries; retrieving platform details; checking Security Application status; and deploying, updating, or deleting applications.
Service Deployment Management	ZTSO, Security Closed-Loop Management	Provides lifecycle operations for Security Services, including deployment, update, and deletion.
Service Template Catalogue	S-RO Admin, ZTSO	Manages Service Templates, including listing, adding, removing, fetching, and downloading template artefacts.
Resource Inventory	ZTSO, Security Closed-Loop Management	Maintains a registry of infrastructure resources. Supports listing, adding, and deleting clusters, as well as listing and retrieving nodes globally or per cluster.

Functional Tests

Several tests have been performed to verify the functionality of the Security Resource Orchestrator and its communication with the internal modules and external components.

Table 0-15: Security Resource Orchestrator functional tests

Name	Description	Passed (Yes/No/Partially)

S-RO Module setup	The module can be automatically installed and configured through HELM.	Yes
Applications templates management	Upload an application template, list all templates, retrieve a specific template (by ID/name+version), download raw template, and then delete it.	Yes
Resource management	Add Kubernetes clusters to Resource, list/filter by type, inspect details (clusters/nodes where available), and truncate/clean the inventory.	Yes
Applications LCM	Deploy an Applicatio, check created DeploymentInfo, list all deployments, fetch by ID, update the Application, and finally delete it.	Yes

1.3.5 Risk-averse Resource Management Framework

In Deliverable D4.1 "Security Automation for 6G", the key elements of the proposed risk-aware resource control and management framework were introduced. This theoretical and algorithmic framework is designed as a robust and versatile component that could significantly contribute to the extension and evolution of the existing resource orchestrator architecture previously described. The rationale behind its development is to incorporate risk aversion strategies and perceptual or subjective evaluations of outcomes and decisions. This integration aims to enhance the system's ability to manage resources effectively in dynamic and uncertain environments, especially in decision-making under uncertainty in mission-critical scenarios, such as in secure 6G networks. This framework provides the analytical optimization framework and is designed to seamlessly integrate into the S-RO architecture.

Main functionalities

First and foremost, it should be emphasized that the proposed framework is theoretical and algorithmic in nature, aimed at optimizing resource allocation and scheduling based on risk-averse utility functions. As such, it is not intended for direct implementation within the platform but rather serves as a conceptual and analytical tool. While it may offer valuable insights, recommendations, or algorithmic principles for the development of components within the S-RO, it does not include a concrete or platform-specific implementation. It is a generic, standalone module with a strong theoretical foundation.

The framework is designed to manage the allocation of resources (e.g., bandwidth, power, etc.) across a set of users, even when their requirements are heterogeneous. It achieves this by optimizing each user's utility function, defined in terms of Cumulative Prospect Theory (CPT). This utility formulation captures key behavioural dimensions, including users' risk aversion (i.e., varying tolerance towards losses and gains), their sensitivity to deviations from a reference point, and their subjective perception of event probabilities. By integrating these factors, the framework models user behaviour more realistically and supports resource allocation decisions that align with diverse user preferences under uncertainty.

The input parameters to the framework are as follows:

- The number of users (N): representing the total set of users among whom resources are to be allocated.
- Each user's objectives and application-specific requirements: such as target service levels, Quality of Service (QoS), or Quality of Experience (QoE).
- Each user's risk attitude: characterized as risk-averse, risk-neutral, or risk-seeking, and captured through the parameters and shape of their respective utility functions.
- The reference point (x_0): which may represent a baseline or expected operating level (e.g., a minimum guaranteed service level under typical system conditions) and can vary depending on the application scenario.

These inputs are processed by the CPT-based optimization module, which formulates and solves an optimization problem using utility functions derived from CPT. The objective is to compute an optimal allocation of resources among users, ensuring that individual service level goals are met while accounting for their risk preferences and behavioural characteristics.

The output of the CPT optimization module consists of the recommended resource allocation per user. This output can be communicated to the S-RO via an API for potential integration into broader system-level decision-making.

Design updates

In addition to developing the theoretical framework that incorporates risk aversion and subjective perception of outcomes, we have introduced two significant advancements:

Generalization of CPT Utility Functions

A generalized utility function, which extends the classical forms used in CPT, such as the standard S-shaped utility functions, is proposed. This formulation addresses the structural limitations of existing models by offering a more flexible and comprehensive structure. It subsumes traditional utility functions as special cases while capturing a broader spectrum of behavioural patterns and functional characteristics. Importantly, the generalization retains the core principles of CPT, namely, reference dependence, loss aversion, and non-linear sensitivity to outcomes, while enabling enhanced adaptability in both theoretical analysis and empirical modelling.

The above solution addresses several key limitations of existing approaches:

- Current utility formulations are often defined only over unbounded or global domains, which restricts their applicability across the full range of real-world outcome spaces.
- Risk aversion behaviour is typically rigidly tied to the reference point (status quo), limiting the flexibility to model diverse user attitudes toward risk.
- Existing models generally fail to incorporate Neilson's distinctions between weak and strong loss aversion within a unified risk-sensitive framework.
- Finally, conventional formulations offer limited behavioural expressiveness, restricting the model's ability to represent a wide variety of user preferences, particularly in empirical or heterogeneous scenarios.

All technical details, including the mathematical formulation and theoretical justification of our proposed generalization, are provided in [VSK25].

Optimization algorithms

A major obstacle in applying CPT is the optimization of its utility function, which is inherently non-convex and non-smooth even in its baseline form. Most recent work tackles this difficulty in portfolio-selection settings. The principal prior approaches are: (i) Minorization–Maximization (MM): CPT utility is maximized via a convex–concave procedure that iteratively solves local convex approximations. Although practical, these methods are heuristic and offer no formal convergence guarantees; (ii) Alternating Direction Method of Multipliers (ADMM): Two ADMM-based variants have been explored—one paired with dynamic programming and another with the Pool Adjacent Violators (PAV) algorithm. While empirically successful, the PAV-based variant does not fully address the non-smooth, non-convex nature of CPT utilities.

An optimization framework tailored to resource-allocation problems with CPT-based users is presented below, demonstrating its potential on an example involving power-allocation scenarios in communication networks. The novelty lies in explicitly handling both non-smoothness and non-convexity throughout the optimization. Our method integrates three components:

- Successive Convex Approximation (SCA): Transforms the original non-concave maximization into a series of tractable concave sub-problems by iteratively refining a surrogate objective.
- Lagrangian Relaxation (LR): Introduces dual variables to manage resource-allocation constraints efficiently.
- Projected Subgradient Method (PSM): Mitigates non-smooth behaviour near reference points by projecting subgradient updates onto the feasible set.

In numerical experiments, this SCA + LR + PSM pipeline scales better with the number of users than Sequential Quadratic Programming (SQP) as implemented in MATLAB's Optimization Toolbox, while maintaining only a modest increase in execution time (see Figure 0-20). In summary, SCA handles non-concavity, LR addresses constraints, and PSM resolves non-smoothness at reference points, collectively enabling robust, scalable optimization for CPT-driven resource-allocation problems.

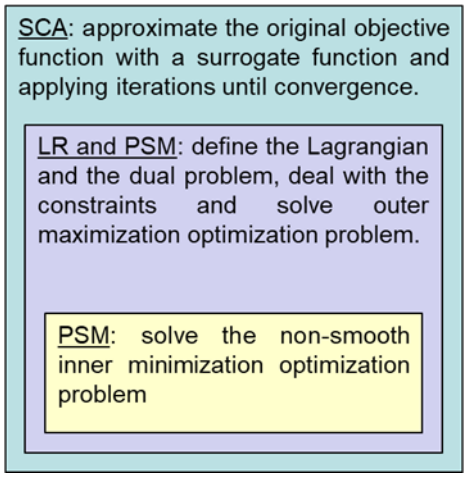


Figure 0-20: Conceptual representation of the proposed CPT optimization technique

Initial Prototype Implementation

A first application and testing of the proposed framework is presented below. Specifically, the power allocation problem in a system with three users experiencing Rayleigh fading is solved, where risk behaviour is modelled using an S-shaped CPT utility function. This simulation illustrates the convergence behaviour of our proposed optimization framework through a simple yet representative scenario. As shown in the contour plot in Figure 0-21, despite initializing near a local minimum, the algorithm quickly converges to a local maximum of the objective function.

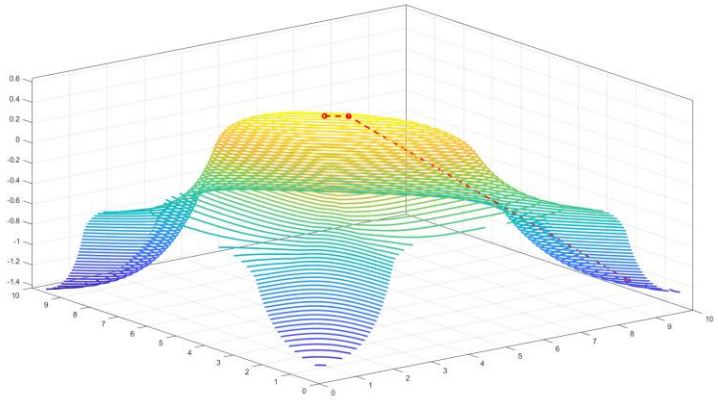


Figure 0-21: 3D contour plot illustrating the convergence process for a scenario with N = 3 users

Next, a system under three different scenarios of the user population is simulated: $N = 10, 30, 50$. Each system is simulated over 500 independent iterations. In each iteration, agents are assigned distinct CPT-based utility functions, and their channel gains are independently drawn from an exponential distribution with unit mean. The average SNR under equal power allocation is set to 7 dB. To evaluate the performance of our proposed optimization method, we use as a metric the percentage of cases where our solution achieves an equal or better outcome, within a tolerance of k %, compared to the solution obtained by MATLAB SQP Global Search toolbox. Specifically, a tolerance of $\pm k$ % implies that if the relative difference between our solution and MATLAB's falls within this range, the two are considered equivalent. The results are summarized in Table 0-16. We also report results based on the trimmed mean in Table 0-17.

Table 0-16: Comparison of objective function values: proposed method vs. SQP

N	Mean % better than SQP (0% tol.)	Mean % better than SQP (2% tol.)
10	38.9	49.8
30	72.8	73.6

50	96.2	96.4

Table 0-17: Trimmed mean of the percentage of quantitative better solution than SQP

N	Mean	1% Trimmed mean	2% Trimmed mean	5% Trimmed mean
10	-10.3613	-10.4444	-10.5075	-10.5354
30	40.6332	40.8660	41.2340	41.7988
50	81.3611	81.5843	81.6584	81.4217

As shown in the results tables, our proposed method outperforms MATLAB's toolbox as the number of users increases. However, this performance gain comes with a longer execution time. Specifically, our method is significantly slower for N=10 and N=30. Nevertheless, at N = 50, the gap narrows, with our approach being approximately four times slower. This trend suggests that as the user population increases, our method not only maintains its performance advantage over MATLAB but also achieves comparable execution times.

Several tests have been conducted to verify the functionality, correctness, and performance improvements introduced by the proposed framework and its optimization module. However, since the risk-averse resource management module is primarily theoretical and designed as a standalone component, it is not intended to undergo extensive functional testing and direct prototype deployment at this stage. As such, the implementation does not involve a defined infrastructure or communication architecture. Instead, only localized tests have been performed to validate its integration and communication with the platform's internal modules and external components. Comprehensive end-to-end testing is outside the current scope, given the conceptual nature of the framework. Evaluation code used in support of the theoretical results can be made available upon request.

1.4 Continuous Monitoring and Threat Detection

Continuous monitoring is composed of the Programmable Monitoring Platform, which is in charge of collecting information from data sources to expose it to external modules. Thanks to this data, a threat detection can be performed in two variants. The first one is related to semantic-aware anomaly detection, used for reducing redundant transmissions through estimation policies and detecting anomalies in large-scale RAN performance data. The second one is a rule-based threat detection module that analyses the network data to determine anomalous behaviour and generates alerts if anything suspicious is detected.

1.4.1 Programmable Monitoring Platform

In this section, the Programmable Monitoring Platform (PMP) will be analysed to explain its scope, functionalities, and the synergy with internal and external components that compound ROBUST-6G. PMP advances to satisfy the modules designed in the previous deliverable. For this reason, the modules evolve from a high-level to a medium-level design, implementing tools that bring real capabilities to the performance they realise. The definition of "tool" used in the section refers to third-party software. This generic term is used due to the number of software programs employed and the diverse functionalities they provide to the PMP.

Main functionalities

The PMP is in charge of collecting information from sources of the three layers (network, infrastructure and service) and exposes it to be used by other components. It has a module-based design to be adaptable and dynamic.

Two of the most important modules have been developed: the Data Collection Module and the Communication Bus. The first one is related to the information collection, due to the necessity of extracting data from sources to feed the external components. The second one is responsible for the communication between the collection module, other internal modules, and the external consumers through APIs.

The modules related to the dynamic configuration and the generation of the alerts are also advancing. The configuration module is in control of configuring the monitoring tools requested by the ZTSO. On the other hand, the alerting module uses the data collected by the PMP to report anomalous behaviour in the monitored devices and in the network segment.

The ZTSO interprets the SSLAs internally to generate tool-agnostic sigma rules [Sig25]. As a result, it can make requests attaching the necessary information to the PMP to configure the tools of the Data Collection module (Snort, Falco, Telegraf, etc.) that need to be deployed. This helps to have a more homogeneous environment, since the Configuration Manager can translate the sigma rules into actual configurations for each type of tool requested by the ZTSO.

In order to request the deployment, two options may be contemplated. In the first case, the ZTSO owns the logic, so the ZTSO should know which tools the PMP can deploy (e.g., Snort) and will configure the deployment on the PMP NBI (e.g., deploy SNORT with this specific rule for this tool). In the second case, the PMP owns the logic, so the ZTSO does not know the monitoring tools that the PMP can deploy, but it can ask for "activities" with sigma rules (e.g., the ZTSO will ask the PMP to monitor traffic for detecting DoS, and the PMP translates sigma rules into SNORT deployment with a particular rule). It is worth mentioning that in both cases, the PMP assumes orchestration responsibilities because it is responsible for deploying the monitoring tools of its Data Collection Module on specific resources. Consequently, the PMP may need permissions and context information to manage the deployment. The final case to be selected will be declared in the next deliverables, so we are analysing all the available options.

The ZTSO, as shown in Figure 0-22, interacts through an API with the PMP to configure and deploy Data Collection tools in the data sources. The communication via Near Real-time Data Retrieval API, Data Exporter API and Historical Data Retrieval API from the Long-term Data Storage allows the information consumption by external modules to perform their functionalities.

Design updates

Figure 0-22 shows the architecture of the PMP with the selected tools per module and the APIs to interact with it. Main modules of the design present the tools explained in the last deliverable and provide the features to configure modules, collect and aggregate data, expose the information in several formats and generate alerts in case of anomalous behaviour. Although the tools appear in each module in Figure 0-22, some of them may need to be deployed in the data sources or in their environment. Due to possible future development issues, some tools may be replaced or supplemented by new ones. This disclaimer applies to possible bugs, incompatibilities between versions or new forms of development in tools that have not yet been implemented.

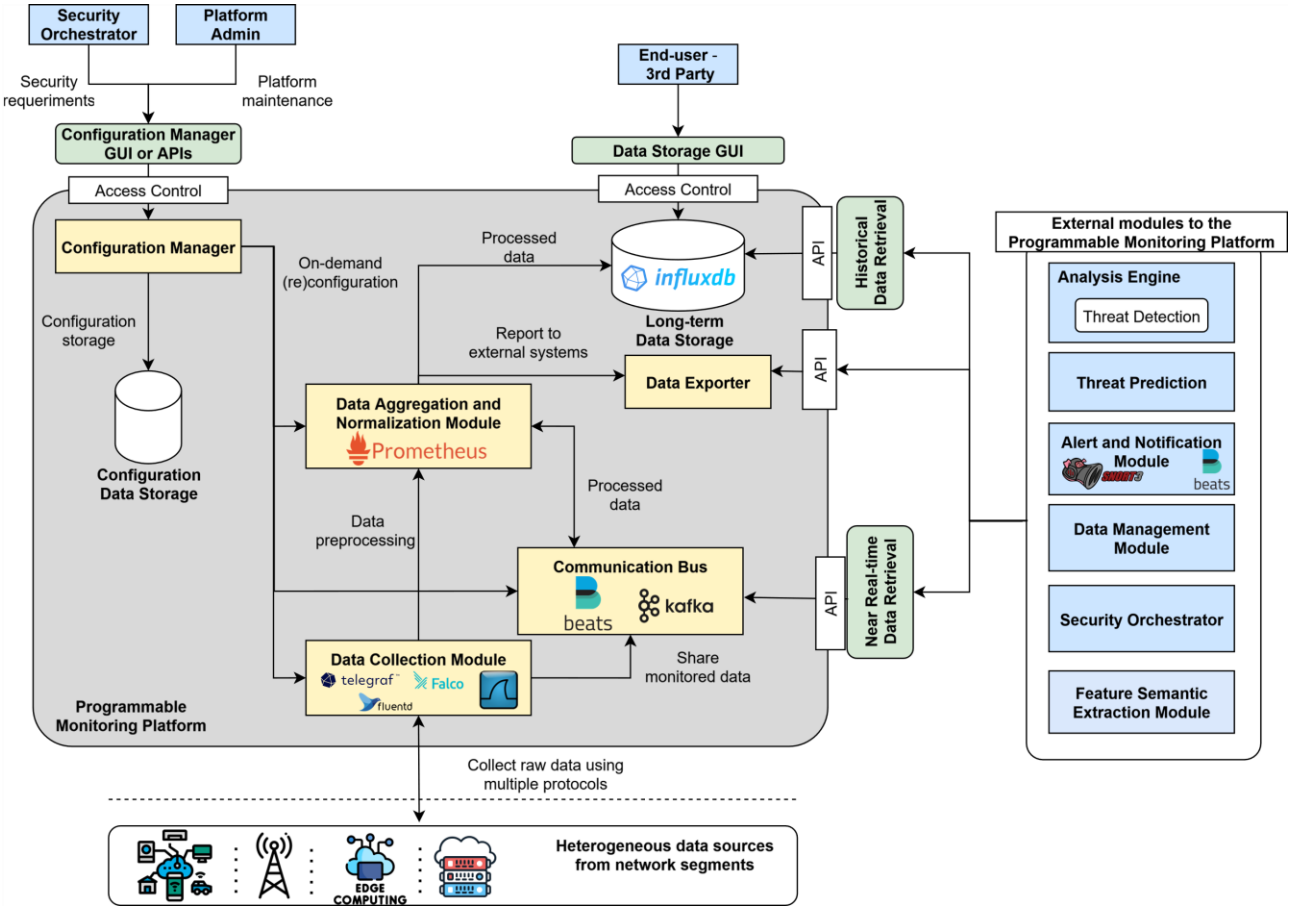


Figure 0-22: Programmable Monitoring Platform Architecture

Initial Prototype Implementation

External communication is one of the most important functionalities of the PMP, which implies a good management in terms of data request and responses. As a result, some APIs are designed to determine these petitions and responses using endpoints through the REST protocol. Swagger 3.0.1 has been used to model these APIs, generating a public web page [PMP25a] where all the information can be reviewed. PMP uses a Model-View-Controller (MVC) as a pattern.

The first interaction of the initial process is related to the ZTSO request with the Configuration Manager API. When a petition is received, the API transmits the information to the Configuration Manager backend. Endpoints provide options for configuring and updating a tool, and an endpoint for obtaining configuration options for each tool.

Table 0-18: Configuration Manager API

Operation name: *deployNetworkTool*		
Description	This endpoint allows the ZTSO to request the deployment of a network tool.	
Input Parameters	**Type**	**Description**
podIP	String.	IP address of the pod where the tool will be deployed.
interface	String.	Network interface on which the tool will sniff traffic.
toolName	String.	Name of the network tool to deploy.
sigma_rules	Array of string.	Sigma rules generated from the SSLA by the ZTSO.
Output Parameters	**Type**	**Description**
200	String.	Ok.
Notes		
Operation performed by the ZTSO to deploy a network tool into a Kubernetes POD or a device.		

Operation name: *deployInfrastructureTool*		
Description	This endpoint allows the ZTSO to request the deployment of an infrastructure tool.	
Input Parameters	**Type**	**Description**
podIP	String.	IP address of the pod where the tool will be deployed.
toolName	String.	Name of the infrastructure tool to deploy.
sigma_rules	Array of string.	Sigma rules generated from the SSLA by the ZTSO.
Output Parameters	**Type**	**Description**
200	String.	Ok.
Notes		
Operation performed by the ZTSO to deploy an infrastructure tool into a Kubernetes POD or a device.		

Operation name: *deployServiceTool*		
Description	This endpoint allows the ZTSO to request the deployment of a service tool.	
Input Parameters	**Type**	**Description**
podIP	String.	IP address of the pod where the tool will be deployed.
toolName	String.	Name of the service tool to deploy.
sigma_rules	Array of string.	Sigma rules generated from the SSLA by the ZTSO.
Output Parameters	**Type**	**Description**
200	String.	Ok.
Notes		
Operation performed by the ZTSO to deploy a service tool into a Kubernetes POD or a device.		

Operation name: *deploySecurityTool*		
Description	This endpoint allows the ZTSO to request the deployment of a security tool.	
Input Parameters	**Type**	**Description**
podIP	String.	IP address of the pod where the tool will be deployed.
toolName	String.	Name of the security tool to deploy.
sigma_rules	Array of string.	Sigma rules generated from the SSLA by the ZTSO.
Output Parameters	**Type**	**Description**
200	String.	Ok.
Notes		
Operation performed by the ZTSO to deploy a security tool into a Kubernetes POD or a device.		

Operation name: *getConfigurationOptions*		
Description	This endpoint allows the ZTSO to know what options it has for configuring a tool.	
Input Parameters	**Type**	**Description**
toolName	String.	Name of the tool for which the orchestrator wants to know the options.
Output Parameters	**Type**	**Description**
200	Array of string.	List of options. Can be one or more options.
Notes		
Operation performed by the ZTSO to obtain the options of one of the tools offered by PMP.		

Operation name: *updateConfiguration*	
Description	This endpoint allows the ZTSO to update the current deployed configuration.

Input Parameters	Type	Description
podIP	String.	IP address of the pod where the tool is already deployed.
toolName	String.	Name of the service tool to deployed.
tool_type	String.	Type of the service tool deployed.
sigma_rules	Array of string.	Sigma rules generated from the SSLA by the ZTSO.
Output Parameters	**Type**	**Description**
200	String.	Response message confirming the success of the request.
Notes		
Operation performed by the ZTSO to update the current configuration of a tool already deployed.		

A special API is generated to be used on the use case 2 scenario 1. It is employed by the ZTSO to interact with a Thingsboard manager via PMP to collect information between timestamps.

Table 0-19: Configuration Manager-Thingsboard API

Operation name: *collectDataFromThingsboard*		
Description	This endpoint allows the ZTSO to request the PMP to trigger data collection from Thingsboard.	
Input Parameters	**Type**	**Description**
thingsboardsIP	String	IP address of the Thingsboard instance from which data will be collected
thingsboardPort	Integer	Port number of the Thingsboard instance
entityId	String	A string value representing the entity id. For example, '784f394c-42b6-435a-983c-b7beff2784f9'.
entityType	String	A string value representing the entity type
timestampInit	String, with date-time format.	Timestamp of start of the information generated by Thingsboard
timestampEnd	String, with date-time format.	Timestamp of end of the information generated by Thingsboard
Output Parameters	**Type**	**Description**
200	String	Ok.
Notes		

On the other hand, information is also exposed thanks to different APIs. Each API provides the information in various formats and levels of processing. Therefore, they will be listed from the lowest to the highest level of processing. To reduce the potential expansion of tables due to the number of endpoints per API, Table 0-20 shows a summary of each API and the rest of the information can be found on the swagger website [PMP25a].

Table 0-20: Summary of PMP export APIs

Interface	Consumers	Description
Near Real-time Data Retrieval	External components.	Offer the possibility of retrieving a small piece of data stored in the current Communication Bus to be consumed by external modules including logs, health metrics, and network traces.
Data Exporter	External components.	Gather the processed information and enable the export of data and report to external systems or for offline analysis.
Historical Data Retrieval	External components.	Allows to retrieving large amounts of data or historical data using the startTime and endTime parameters. In

		addition, it can receive requests to extract information of an userID.

The Data Collection Module is represented by four containerised-based tools to cover all the three layers mentioned previously. These tools act by gathering network traces from the network layer, health metrics from the infrastructure layer and logs events from the service layer. In addition, the security is taking into consideration by one of the tools used on the service layer.

The module has another functionality to identify each monitored source, having in account that a source can be a Cloud continuum device. This feature is called "Machine ID" or its acronym "MID" [Mid25]. It uses the most static parameters of the physical machine and the operating system (MacOS, Windows and Linux) to determine a hash to identify a device univocally. Thanks to having each device associated with its MID, the internal and external consumer modules can recognise the health metrics, logs and events of a certain machine. Furthermore, the PMP can provide better historical data classified by the MID.

Once the MID of the machine on which the PMP is acting is established, the tools will be explained using the segmentation layers declared before.

First layer is related to the network. PMP implements a modified image for Tshark [Tsh25] as a network sniffer to collects traces. The image is based on Alpine's latest official version [Alp25]. The dependencies are covered using Tshark, Libcap [Lib16] and Python3 [Pyt25]. Tshark and Libcap are used to gather the network traces and Python3 to launch the image entry point to be started when the container is created. Additionally, permissions are set to use Tshark without being "root" by changing the Linux capabilities [Lin25], since Alpine is based on Linux distribution. An entry point script in Python is created to search the interfaces actives on the host machine (not the container) and to create the Tshark command with this information. The output is generated in a file with JSON [Jso25] format. This file is rotated when reaches the size limit variable, being 30MB by default, but can be modified. A docker volume is created to manage the output JSON file. Due to the volume, the Communication Bus Module can transmit the network traces to other components/modules.

Second layer is associated with the infrastructure and the health metrics extracted from devices by Telegraf [Tel25a]. The official Telegraf image used is the 1.33.2 version [Tel25b]. In this case the image is not altered, since the container created contains all the necessary features to work, except for the configuration file. A docker volume is created to allow the container to access the configuration file.

Telegraf can handle several types of plugins in its configuration file, thus the data extracted is directly correlated with these plugins. A list of characteristics is selected to provide a minimum of data to determine the health status on the devices. The metrics correspond to total, available and used RAM; disk usage in terms of file system avoiding temporary files; CPU usage considering percentages, kernel and user space values and interruptions, among others; temperature, humidity, voltage and power to name a few from sensors; and internet speed, location, latency and jitter from the network interfaces of the devices. All these metrics are collected in thirty seconds intervals, except for internet data, which is sixty seconds due to network packets problems. These intervals are configurable, but due to some tests, several plugins need more time than others to process, generate and send the information to Telegraf. In the case of network packet plugin, the exception is related to the internal timeout of the plugin which caused a close connection and Telegraf cannot receive the request. Since the plugin's internal timeout cannot be modified because there is no access to that parameter in the configuration, the information exposure interval has been modified. This parameter is controlled by Telegraf, not by the plugin.

Output is controlled by Telegraf thanks to a Kafka [Kaf25a] plugin [Kaf25b] to publish the information directly in the topic created for this purpose on the Communication Bus.

The third layer is for services. In this regard, two tools are being used to gather event information such as logs or security incidents. Fluentd [Flu25a] is responsible for collecting the device's event logs and exposing its internal data. The tool is based on the containerised image 1.16.8-debian-1.0 version [Flu25b]. The main reason for using this version is because recent versions of the official image have crash errors when loading the initial configuration and makes calls to internal functions. The image is modified to install some Debian dependencies required and the Fluentd plugins.

Moreover, a script to calculate internal data and an entry point for tool execution and handle large log event files are created. The script needs to "warm up" the CPU measurement by avoiding the first values, as it generates errors. To solve this problem, the internal metrics of Fluentd cannot start at the same time than the event logs, having a delayed initialization during the first few seconds of tool launch. On the other hand, the entry point aforementioned creates the command and set the size limit of the log event file to 20MB by default to not exceed and saturate the device disk. Fluentd's configuration can be multisource, providing filters, parsers and even adding and listing elements in the process. Output brings several options such as write in a file, in a terminal or exposing the information using an endpoint with a socket or a plugin associated with a specific tool, among others. According to the configuration, the output option used is to write data sources like syslog and systemd log events in a JSON format to a file to be shared via a volume to transmit the information through the Communication Bus module.

Other tool used in the service layer is Falco [Fal25a]. PMP implements it as a mechanism to manage the security of the devices by deploying a modified image based on the official version named falcosecurity/falco:0.40.0 [Fal25b]. These configurations consist of two files and use the entry point associated to launch the process. One of the files is in charge of the configuration and the other of determining the triggers, since Falco is a rule-based agent.

Because of using the 0.40 version of Falco, the tool can be run in eBPF mode [Ebp23]. Main benefits are the support of syscalls, new types of buffers allowing and improving the use of multiple buffers per CPU and a least privileged mode to trigger events.

Falco's output is written in a file in JSON format, offering the possibility of adding new fields to each event triggered by the rules configured. The information extracted in the file previously named is handled by the Communication Bus to be exposed.

Once the Data Collection Module is generating the information from the data sources, the Communication Bus implements two tools to exchange it. Filebeat [Fil25] performs file reading from different sources at the same time providing filters and adding new fields if necessary. Despite being contained on the Communication Bus, Filebeat needs to be deployed in the same device than the Data Collection Module tools to read the files written by them. On the other hand, a Kafka bus is built to publish and expose the information via subscription. Filebeat works from the devices to transmit the data to Kafka bus using topics created dynamically.

Filebeat employs an official image with version elastic/filebeat:8.16.2 [Fil24] to generate the container and only demands access to the files written by the other tools and a configuration file. One volume is used for each tool, being able to have more than one file per volume. The configuration needs an entrance per input which each input is a file or more, because it manages filters with regular expressions and can specify different file paths. Finally, Filebeat use a plugin to expose the output through Kafka using the topics associated with the name field utilised in the entrances.

Kafka is based on the official image version apache/kafka:3.9.0 [Kaf24]. The configuration deploys three listeners as sockets for the internal and external brokers, and the controller. The internal broker is for the publish part linked to the tool's outputs, the controller oversees managing the connections and communications of the broker, and the external broker is used on the subscription part exposing the information via topic. Also allows the feature of hot-swap and create topics.

The Configuration Manager module and the Alert and Notification module is also advanced in terms of development but not finished yet. The Configuration Manager module works with petitions requested to its own API, processing the request, translating the sigma rules and then interacting with the Data Collection Module to select the tool and the possible options of the configuration.

The code is published in official GitHub repository of the UMU team [PMP25b]. It has public access for testing and can be downloaded and used by everyone.

Functional Tests

Several tests have been performed to verify the functionality of each tool used and its communication with the internal modules and external components.

As mentioned before, the tools used in the Data Collection Module and Communication Bus are containerised, so tests have been performed on these docker containers. For this purpose, the internal docker network has been used for communication, but the system is prepared to operate even outside of this network. Also, a docker compose file has been utilised to manage the container as microservices with their own configuration, permissions, environment parameters and volumes required.

The tests have been performed on a virtual machine with Ubuntu 22.04, 4 cores, 16GB of RAM and 50GB of storage. The main objective was to test the configuration, deployment and functionality of the modules. Therefore, an extensive default configuration has been generated for each tool in which plugins (if any) were used. After the configuration, the Machine ID has been generated univocally to identify the device on which the tools are deployed. In case of the network layer with Tshark, the currently active interfaces are searched for network trace extraction.

Since all containers are launched at the same time, a staggered deployment is required so that the Communication Bus tools are deployed first and the rest wait until they do not give a signal confirming their successful deployment. In the case of a real environment or a use case, this would not be necessary because the Communication Bus would already be deployed and managed previously.

Once all the containers have been correctly deployed as shown in Figure 0-23 the tools are working on all layers. Their status is "up" and the connections as ports and IPs are assigned. Because of this test, the configured IP is a "not specific direction", "wildcard" or also called 0.0.0.0, but in a real environment it has a public or private IP. This "wildcard" is used by the containers to deploy their plugins, brokers and additionally to consent to subscription of exposed information through a tool called Kafkacat [Kaf22]. This tool is used to verify the transmission of information is correct, although it is not implemented in the PMP. It can be taken as an example of what an external component would see when making a request to the Near Real-time Data Retrieval API that uses information directly from the Communication Bus.

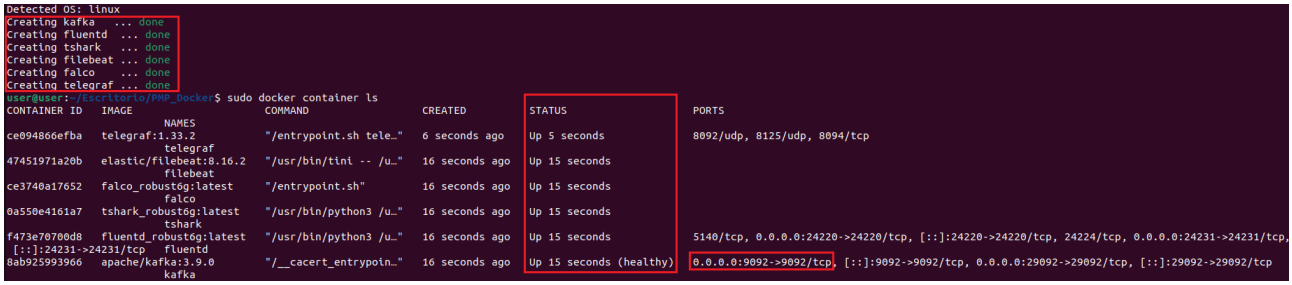


Figure 0-23: Deployment and configuration of the Collection Module and Communication Bus

Figure 0-24 shows various outputs in JSON format associated with the network, infrastructure and service layers. On the top left there is network traces extracted from the device's network, on the top right there are device health metrics related to memory and disk. On the bottom left are some system event logs from the operative system and on the right are the security events captured at the time. Furthermore, the "machine_id" is also exposed in order to identify the information associated with the device.

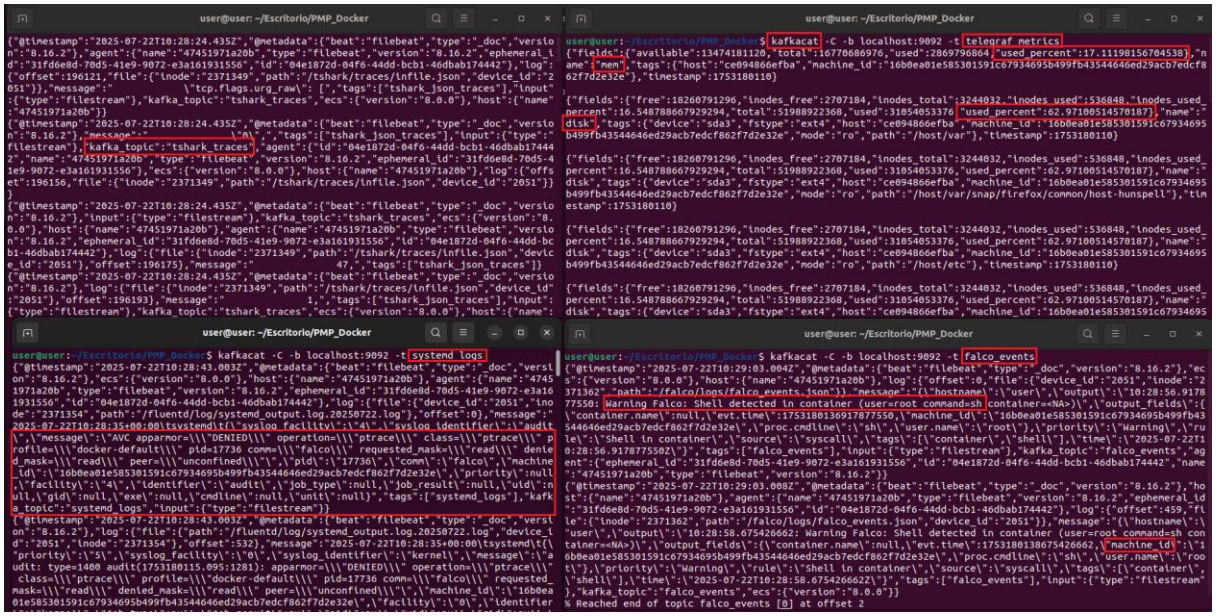


Figure 0-24: Data collected and exposed by the PMP

Following the above order shown in Figure 0-24 the first to be explained is the network layer. Traces are generated by Tshark in JSON format and reading automatically by Filebeat to encapsulate it in "beats" to be sent to the Kafka broker. The topic used is called "tshark_traces" and the message field register a line of a network packet. Second layer is related to infrastructure and employs Telegraf and several plugins to extract health metrics. As shown in Figure 0-24, Kafkacat read the topic allocated on Kafka broker. Memory and disk present their parameters such as used percent, total quantity or used in bytes. Also include the "machine_id" in each "beat". Finally, the third layer associated with the event logs is split into two functionalities. First one is on the bottom left of the Figure 0-24 providing system logs from systemd thanks to Fluentd. Message field has the reference of an event log in syslog produced by the Falco container. Second one is on the bottom right and it is referred to Falco generating security events logs. In this case, the event is forced as Figure 0-25 shows. A shell is opened into the container as an attacker did to exploit one or more vulnerabilities. In the exact moment that a security risk is detected by Falco, it checks its rule file to determine the output of the event and how to proceed. In this example, the tool writes a warning with some additional parameters as the user or the command, among others.

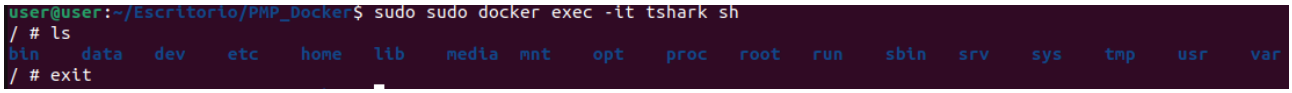


Figure 0-25: Creation of a shell into a container to force a Falco event

As seen in Figure 0-23 and Figure 0-24, the modules for data collection and transmission are fully functional thanks to the current configurations, which will be managed by the Configurator Manager on demand by the ZTSO. Finally, the main objectives related to configuration, deployment and functionality of the modules has been proven.

1.4.2 Semantic-aware Anomaly Detection

This module addresses anomaly detection through the use of metrics that incorporate the contextual and temporal relevance of information. It integrates two distinct lines of work: first, a remote estimation approach that leverages semantics-aware indicators to quantify the persistence and impact of estimation errors. Second, a data-driven anomaly detection framework employs a Generative Adversarial Network (GAN) to identify anomalous patterns within high-volume Radio Access Network (RAN) performance data. These components are designed to enhance the reliability of alert generation by prioritizing significant anomalies and reducing superfluous transmissions. The overall objective is to facilitate automated, efficient monitoring and response within the platform, particularly in contexts requiring timely and resource-aware system feedback.

Main functionalities

This module significantly enhances AI/ML-driven zero-touch security by integrating semantics-aware metrics and contextual knowledge into its anomaly detection capabilities. Functionally, it applies novel metrics like Age of Consecutive Error (AoCE) and Age of Information (AoI) during both the training and evaluation phases of AI/ML models. This approach ensures models are optimized to recognize anomalies based on their real-world impact and significance, leading to increased robustness. In operation, these metrics enable the precise and contextually relevant identification of anomalies while reducing redundant message transmissions by prioritizing critical alerts based on their semantic importance, ultimately solving more advanced optimization problems than traditional methods.

The key capabilities and internal processes are as follows:

- Temporal Pattern Recognition with Unsupervised Learning (RANGAN): RANGAN uses a GAN-based anomaly detector combined with a transformer architecture to identify complex temporal patterns and deviations in multivariate RAN performance data, all without needing labelled training data.
- Semantic-Driven Error Quantification (Insec-SPI): Insec-SPI quantifies the impact and timeliness of estimation errors. These metrics provide a critical assessment of error severity and persistence, beyond just their presence.
- Context-Aware Policy Optimization: Building on Insec-SPI's insights, this module develops efficient transmission and estimation policies. It selectively triggers updates only when semantic thresholds are breached, ensuring resource-constrained anomaly detection prioritizes high-impact events.
- Unification of Estimation and Detection: This module integrates estimation quality with anomaly detection, enabling the system to infer anomalies from the contextual degradation of estimation performance, even when data is noisy or incomplete.

The module interrelates with the alert management system and zero-touch orchestration, by providing an intelligent solution for remote estimation and subsequent anomaly signalling. When raw network signals can be symbolized as states in a Markovian decision process, our component's core functionality, derived from Insec-SPI, triggers a transmission of an anomaly alert only when its calculated proposing semantics error (which quantifies the lasting impact of consecutive errors such as AoCE) exceeds a context-dependent threshold. This threshold itself dynamically adapts based on the AoI and the instantaneous estimation error, ensuring that only truly significant and timely anomalies are escalated. This approach directly feeds into automated decision-making processes, enabling the alert management system to prioritize and route critical events, and allowing the zero-touch orchestration layer to initiate proactive or reactive mitigation measures, ultimately minimizing human intervention and optimizing resource utilization across the platform.

Design updates

Building on prior work, a semantics-aware method has been developed to quantify the significance of estimation errors using semantics-aware metric. This enhancement allows the system to not only detect errors but also assess their temporal persistence and cumulative impact, leading to better decision-making with limited communication resources. From this, an optimal transmission policy has been derived, structured as a mixture of two switching policies. Each policy triggers a transmission only when the semantic errors such as AoCE or AoI exceed a predefined threshold, enabling more efficient and adaptive scheduling for remote estimation tasks like alert generation.

In parallel, a new framework has been proposed to improve the scalability and complexity of anomaly detection in RANs. This framework integrates a GAN with a transformer architecture to capture long-range temporal dependencies in high-dimensional KPI data. Designed for unsupervised operation, it provides robust anomaly detection without needing labelled data and continuously adapts to dynamic network conditions. These combined advancements extend the module's capabilities from semantics-aware estimation to comprehensive end-to-end anomaly detection and response, which can contribute to the security and management stack.

This module is derived from theoretical research and is not intended for direct prototype deployment at this stage. As such, the implementation does not involve a defined infrastructure or communication architecture. Evaluation code used in support of the theoretical results can be made available upon request.

Table 0-21: RANGAN APIs

Operation name: *RANGAN*		
Description	an anomaly detection framework that integrates a Generative Adversarial Network (GAN) with a transformer architecture	
Input Parameters	**Type**	**Description**
FH Traffic	Float	FH uplink/downlink usage in Gbps
Thread scheduling	Float	On and off CPU runtimes of threads
PTP logs	float	PTP frequency, RMS, delay, and max offset
Window size	Integer	Sliding window to partition the time series into overlapping fixed-length segments
Output Parameters	**Type**	**Description**
Generated samples	Array	Output samples of the generator model (size: window size x the number of features)
Authenticity indicator	Binary integer	Output samples of the discriminator model that distinguish whether the input is real or synthetic
Notes		

Insec-SPI

Structure-aware algorithm comprises two key components:

1. a structured policy iteration (SPI) module for computing a switching λn-optimal policy at each iteration n, and
2. an intersection search module for updating the Lagrangian multiplier.

Table 0-22: Insec-SPI APIs

Operation name: *SPI for computing λ-optimal policies.*		
Description	Computing a switching λn-optimal policy at each iteration n	
Input Parameters	**Type**	**Description**
States	Finite set / array	The set of all possible system states defined by AoI and AoCE values.
Initial switching policies	Array	Baseline policies used to initialize the iteration process.
Reference state	Integer	A fixed state used as an anchor to normalize policy evaluation.
Truncation sizes	Integers	Limits on AoI and AoCE dimensions to keep the state space finite.
Output Parameters	**Type**	**Description**
Switching policies	Array	Updated λn-optimal switching policies derived from structured policy iteration.
Notes		

Operation name: *Insec-SPI for finding optimal policies.*		
Description	An intersection search module for updating the Lagrangian multiplier.	
Input Parameters	**Type**	**Description**
λ_o^-, λ_o^+	Float	Initialized search interval

Output Parameters	Type	Description
λ*	Float	Optimal Lagrangian multiplier obtained from the intersection search.
\pi^*	Array	associated with the multiplier λ*.
Notes		

Functional Tests

Several tests have been performed to verify the functionality of the Semantic Aware Anomaly Detection component and its communication with the internal modules and external components.

Table 0-23: RANGAN's functional tests

Name	Description	Passed (Yes/No/Partially)
Sliding window test	The framework maintains high detection performance when using larger sliding window sizes.	Yes
Baseline comparison	The framework outperforms standard methods (e.g., IF, LOF, Autoencoder) on the same dataset and evaluation metrics (e.g., F1-score)	Yes
Anomaly score discrimination	The framework produces well-separated anomaly scores between normal and abnormal segments.	Yes

Table 0-24: Insec-SPI's functional tests

Name	Description	Passed (Yes/No/Partially)
Policy complexity test	The intersection search method reduces computation time compared to bisection.	Yes
Truncation robustness	The optimal policy remains accurate under AoI/AoCE truncation, with low KL distance.	Yes
Estimator performance	The MAP estimator provides higher estimation accuracy than the ZOH estimator.	Yes
Transmission frequency threshold	The policy achieves optimal estimation performance with low transmission frequency ($F_{max} \leq 0.3$), avoiding unnecessary updates.	Yes

Rule-based Threat Detection

In this Section threat detection is covered by a component external to the PMP but related to it. The Alert and Notification Module is characterised by analysing network traffic previously extracted by the PMP and generating rule-based alerts to inform the closed loop about it.

Main functionalities

The Alert and Notification Module's functionality is based on analysing the network traces coming from the data sources monitored by the PMP and generate alerts. These are then communicated to the closed loop via the PMP's Communication Bus. The alerts will be used by components such as the ZTSO, offering the possibility of taking measures to modify the configuration of other ROBUST-6G components, as well as establishing new rules in this module. In this way, the closed loop will always be protected and updated in case of anomalous behaviour in the network.

Despite of being an external component, the configuration of the Alert and Notification Module is done by the PMP, in the same way as the other internal components of the PMP. The ZTSO sends sigma rules generated from the SSLA to the PMP, internally the rules are translated into a configuration for the tools that compose the Alert and Notification Module. In case of the ZTSO need to take a measure related to this module, the process will be restarted to generate new sigma rules and configuration.

Furthermore, the deployment has two options. In the first one, the module is deployed by the ZTSO as another node of the closed loop with the configuration that the PMP send previously. In the second option the PMP has the responsibility of deploying the module, but as when deploying the monitoring tools, it would also need special permission and context of the environment to perform this functionality.

Design updates

The Alert and Notification Module has been updated internally. Its two main functionalities have been separated to manage more efficiently the work performed by each of them.

As shown in Figure 0-26, the module receives network data to analyses it and generate alerts that are shared with the internal Notification Module. This latter module contacts the Communication Bus located into the PMP in order to send the relevant alerts and expose them to the entire closed loop.

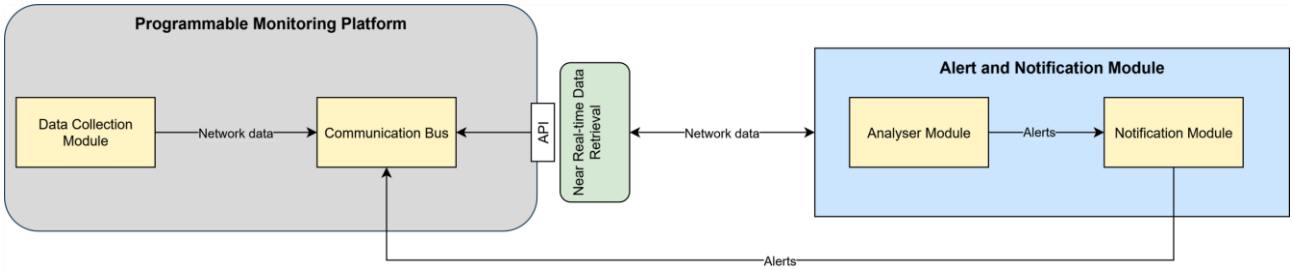


Figure 0-26: Alert and Notification Module high-level architecture

Initial Prototype Implementation

The Alert and Notification Module will be containerised. At the moment, it works as a local implementation but is designed to export its functionality to a container structure in which the tools will be deployed separately as microservices.

As shown in Figure 0-27, the Alert and Notification Module consumes network data in JSON format through the Near Real-time Data Retrieval API. This data is received by the module and translated using JSON2PCAP [Jso24] so enable the Analyser Module to interpret the network traces with Snort3 [Sno25]. The main reason for using a translator/parser is because Snort3 can only read file in PCAP format. Snort3 scans the traces like an IDS and generates alerts based on the rules set in its configuration. The alerts are written to a JSON file shared with the Notification Module that utilises Filebeat (explained in the PMP Section) to automatically send the alerts to the PMP's Communication Bus.

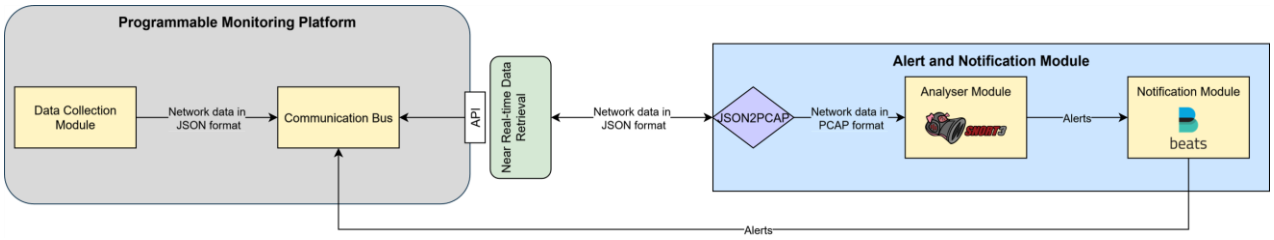


Figure 0-27: Alert and Notification Module low-level architecture

The Snort3 used version is 3.7.1. Configuration of Snort3 is complex due to several dependencies and internal components. The most important libraries are Libdaq [Lib25a] and Libpcre2 [Lib25b] which must be installed and configured before Snort3 in order to make use of the analysing features. An automatic installation file has been created which includes all the necessary dependencies, as well as intermediate error checks. On the other hand, a LUA file is generated that enables and configures many of the modules relied upon with Snort3. In addition, the rules file allows updates and changes to be made in response to new threats.

Functional Tests

A manual test has been performed on this component because it is not yet automated. Despite this, the functionality and the steps to follow are the same, meaning that after the test it can be verified that the component would work.

A virtual machine with Ubuntu 22.04 has been used to carry the test out. Main objectives are to check the configuration of the tools, as well as the workflow of the information that is being simulated due to the manual test. Tools selected are Tshark, JSON2PCAP and Snort3. The Notification Module would use the same container with Filebeat as the PMP in its Communication Bus to transmit the information, but in this case, it will not use to reduce the scope of the test.

Figure 0-28 start with the simulation of the PMP collecting network information from Tshark and saving the traces in a file. These traces have the same format that can be found exposed in the Communication Bus in a regular process. The file "infile.json" can be consume by the Alert and Notification Module, but JSON2PCAP needs to translate the JSON format to a PCAP due to restrictions of Snort3 mentioned in the previous Section.

```
user@user:~/Escritorio/test$ tshark -i enp0s3 -i lo -T json -c 100 -x --no-duplicate-keys > infile.json
Capturing on 'enp0s3' and 'Loopback: lo'
 ** (tshark:4934) 10:50:04.450302 [Main MESSAGE] -- Capture started.
 ** (tshark:4934) 10:50:04.451397 [Main MESSAGE] -- File: "/tmp/wireshark_2_interfacesCPI692.pcapng"
100
1 packet dropped from enp0s3

user@user:~/Escritorio/test$ python3 /home/user/Escritorio/parserjsonpcap/json2pcap/json2pcap.py -i /home/user/Escritorio/test/infil
e.json -o /home/user/Escritorio/test/traces.pcapng -v
```

Figure 0-28: Data network collection and translation into PCAP format

The data is parsed and saved as "traces.pcapng" to be interpreted by Snort3 in Figure 0-29, where a segmentation by colours indicates the different parts of the configuration. Red is for the tool call, orange to load the LUA configuration for the internal modules and the selected fields to be written into the output, blue to import the activation rules, and finally yellow for the storage of the alert file.

```
user@user:~/Escritorio/test$ snort -c /home/user/snort3/lua/snort.lua -R /home/user/Escritorio/test/alert_rules.rules -r /home/user/
Escritorio/test/traces.pcapng -A alert_json --lua "alert_json = {file = true, fields = 'msg timestamp pkt_num proto pkt_gen pkt_len
dir src_ap dst_ap rule action'}" -l /home/user/Escritorio/test/
--------------------------------------------------
o")~   Snort++ 3.7.1.0
--------------------------------------------------
Loading /home/user/snort3/lua/snort.lua:
Loading snort_defaults.lua:
Finished snort_defaults.lua:
        ssh
        host_cache
        pop
        so_proxy
        stream_tcp
        mms
        smtp
        gtp_inspect
        packets
        dce_http_proxy
        alert_json
```

Figure 0-29: Launch of Snort3 analysis

Once the analysis is completed, the alert file is generated as shown Figure 0-31 and Snort3 also present a summary of the packets as appear in Figure 0-30 where the tool detects 11 packets, with 9 flows composed of 100 network traces and 34 alerts as in Figure 0-31.

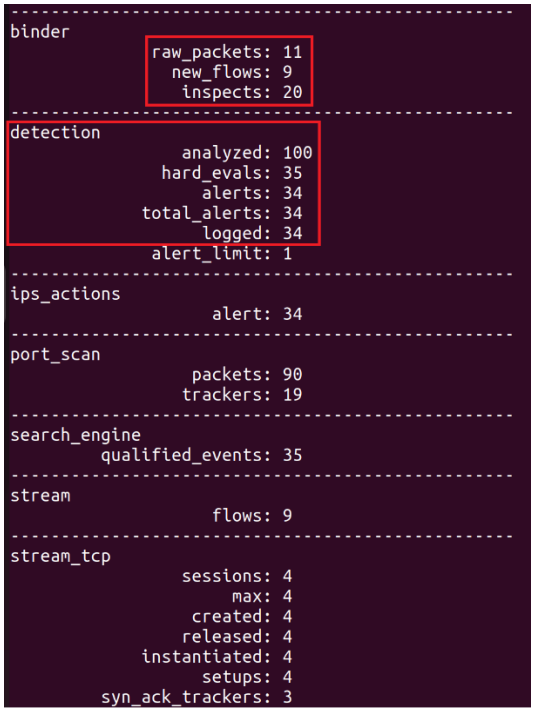


Figure 0-30: Snort3 analysis summary

This test uses a very basic configuration file to detect all traffic performing HTTPs over the 443 port as shown in Figure 0-31. Additionally, data such as the timestamp, the packet where the alert has been detected, the protocol over which it works, the source and destination IP and even the action that can be taken among others are also provided. All these options are configurable and one of the great potentials of Snort3 in terms of information granularity.

This information is what will be transmitted by the Filebeat container acting as the Notification Module. This part is omitted as it works in the same way as in the example given in the PMP section.

30 { "msg" : "Alert: Traffic on port 443 detected", "timestamp" : "07/28-10:50:23.398040", "pkt_num" : 90, "proto" : "TCP", "pkt_gen" : "raw", "pkt_len" : 40, "dir" : "S2C", "src_ap" : "34.160.144.191:443", "dst_ap" : "10.0.2.15:40522", "rule" : "1:1000001:1", "action" : "allow" }
31 { "msg" : "Alert: Traffic on port 443 detected", "timestamp" : "07/28-10:50:23.402170", "pkt_num" : 92, "proto" : "TCP", "pkt_gen" : "raw", "pkt_len" : 40, "dir" : "S2C", "src_ap" : "34.160.144.191:443", "dst_ap" : "10.0.2.15:40506", "rule" : "1:1000001:1", "action" : "allow" }
32 { "msg" : "Alert: Traffic on port 443 detected", "timestamp" : "07/28-10:50:23.406471", "pkt_num" : 93, "proto" : "TCP", "pkt_gen" : "raw", "pkt_len" : 78, "dir" : "S2C", "src_ap" : "34.160.144.191:443", "dst_ap" : "10.0.2.15:40522", "rule" : "1:1000001:1", "action" : "allow" }
33 { "msg" : "Alert: Traffic on port 443 detected", "timestamp" : "07/28-10:50:23.406495", "pkt_num" : 94, "proto" : "TCP", "pkt_gen" : "raw", "pkt_len" : 40, "dir" : "C2S", "src_ap" : "10.0.2.15:40522", "dst_ap" : "34.160.144.191:443", "rule" : "1:1000001:1", "action" : "allow" }
34 { "msg" : "Alert: Traffic on port 443 detected", "timestamp" : "07/28-10:50:23.421165", "pkt_num" : 98, "proto" : "TCP", "pkt_gen" : "raw", "pkt_len" : 86, "dir" : "S2C", "src_ap" : "34.160.144.191:443", "dst_ap" : "10.0.2.15:40506", "rule" : "1:1000001:1", "action" : "allow" }

Figure 0-31: Alerts generated by Snort3

The automation process is still in development. JSON2PCAP and Snort3 will be container-based to be deployed and will utilise volumes to transmit their information between internal modules. These tools in combination with Filebeat will constantly work to read the traffic, analyse it and report back in case a rule is triggered.

1.5 Incident Prediction and Continuous Mitigation

In this subsection, we present the development of the threat mitigation and prediction models and their interaction with the security orchestration process. As illustrated in Figure 2-32, telemetry and network traffic data captured by the Programmable Monitoring Platform (PMT) are first stored in the data fabric. The data are then retrieved and analysed by the modules to generate real-time reports for the security orchestrator. The core functionalities, deployed models, and their performance are detailed in the following subsections.

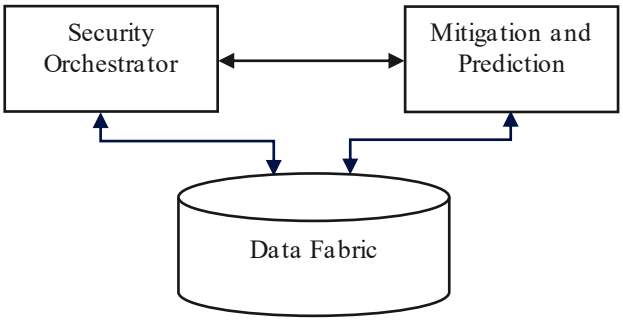


Figure 0-32: Architecture of Threat Prediction and Mitigation

1.5.1 Threat Mitigation Module

This module is designed to provide threat mitigation based on network and telemetry traffic conditions, leveraging AI-based techniques to enable proactive mitigation even without a prior threat detection mechanism.

Main functionalities

The key threats addressed in this project include but not limited to Denial of Service (DoS), Distributed Denial of Service (DDoS), infiltration, web attacks, and crypto mining. The proposed mitigation actions of the threats are as shown in Table 0-25. This is AI-driven threat mitigation based on network and telemetry traffic conditions. This method leverages network flow statistics generated by CICFlowMeter [AppCic] and IoT device telemetry collected via PMT. By analysing observed traffic patterns, the mitigation module recommends appropriate actions M1–M16, as defined in Table 0-25 and provide required parameters specified in Table 0-26 to the security orchestrator to counter potential system threats. Unlike traditional approaches, this module operates without prior threat detection knowledge, relying solely on real-time network traffic analysis. As a result, it can proactively identify and mitigate previously unknown threats, enhancing system resilience.

Table 0-25: Mitigation Strategies for Different Attack Types

Code	Mitigation of Attack	DoS / DDoS	Bot	Brute Force	Infiltration	Web	Cryptomining
M1	Isolating infected system	1	1			1	1
M2	Close all unused ports via firewall		1	1	1	1	
M3	Segmentation / isolation of critical system (to separate VLAN)	1	1				
M4	Block attacker	1		1	1		
M5	Blacklist and whitelist attacker	1*		1*	1*		1*
M6	Enhance system security credential			1	1	1	

M7	Rate limiting and throttling to limit the incoming connection attempts	1		1	1	1	
M8	Traffic filtering and scrubbing	1*	1				
M9	Perform patching/update		1			1	1
M10	Perform system reinstallation (or IoT device fresh and secure SW image)		1				1
M11	IoT platform reset / factory reset						1
M12	New IoT platform deployment						1
M13	Account lockout for repeated failures			1			
M14	Implement CAPTCHA		1	1	1		
M15	Notify network operator/provider	1*	1*			1*	1*
M16	Notify vendor	1*	1*			1*	1*
	1 – Mitigation on all Severity Level						
	1* – Mitigation on High Severity Level						

Table 0-26: Parameters Required for Mitigation Actions

Code	Mitigation of Attack	Parameters for mitigation action
M1	Isolating infected system	IP of infected machine
M2	Close all unused ports via firewall	IP of targeted machine
M3	Segmentation / isolation of critical system (to separate VLAN)	IP of targeted machine
M4	Block attacker	IP and / or MAC addresses of attacker
M5	Blacklist and whitelist attacker	IP and / or MAC addresses of attacker
M6	Enhance system security credential	IP of targeted machine
M7	Rate limiting and throttling to limit the incoming connection attempts	IP of targeted machine
M8	Traffic filtering and scrubbing	IPs of attacker and targeted machines
M9	Perform patching/update	IP of infected machine
M10	Perform system reinstallation (or IoT device fresh and secure SW image)	IP of infected machine
M11	IoT platform reset / factory reset	IP of infected machine
M12	New IoT platform deployment	IP of infected machine
M13	Account lockout for repeated failures	IP of targeted machine
M14	Implement CAPTCHA	IP of targeted machine

M15	Notify network operator/provider	IPs of attacker and targeted machines
M16	Notify vendor	IPs of attacker and targeted machines

Design updates

The architecture of the Threat Mitigation system is illustrated in Figure 0-33. In this setup, the security orchestrator initiates a request to activate the mitigation service. Upon receiving the request, the mitigation module retrieves real-time or near-real-time data from the data fabric and generates recommended mitigation actions for the security orchestrator.

The decision-making process for mitigation is detailed in Figure 0-33. Data retrieved from the data fabric undergoes preprocessing, feature selection, and normalization. The refined data is then input into the mitigation model to determine the appropriate actions. Among the evaluated models—Binary Relevance (BR) [Bou04], Classifier Chain (CC) [Rea11], and Long Short-Term Memory (LSTM) [Hoc97]—the best-performing one is selected and deployed to classify and mitigate anomalies in the network and IoT devices.

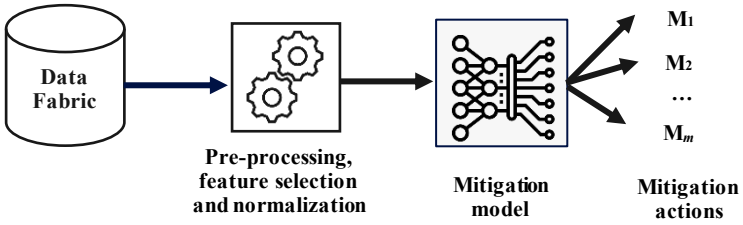


Figure 0-33: Threat Mitigation Process Flow

Initial Prototype Implementation

The mitigation module is developed in Python using machine learning frameworks including PyTorch [Pas19]], TensorFlow [Aba16], and Scikit-learn [Ped11]. Table 0-27 to Table 0-30 illustrate the implemented functionalities designed to address various mitigation scenarios. The module's API will be built using FastAPI [Ram18], exposing a RESTful interface for communication with the security orchestrator. Real-time flow data will be ingested through Kafka using a publish-subscribe mechanism. The entire system will be containerized with Docker to ensure scalable and portable deployment.

Table 0-27: Data Preparation on Network Flow Statistics for Mitigation Module

Operation name: Network Data Preparation for Mitigation		
Description	Perform preprocessing on network data which including data cleaning, data imputation, data normalization and feature selection	
Input Parameters	**Type**	**Description**
nfsdata	Structured tabular data	Network Flow Statistics extracted by CICFlowmeter from the network traffic
Output Parameters	**Type**	**Description**
clnfsdata	Structured tabular data	Cleaned, normalized and selected network flow statistics data ready for model training / testing
Notes		

Table 0-28: Data Preparation on Telemetry Data for Mitigation Module

Operation name: Telemetry Data Preparation for Mitigation

Description	Perform preprocessing on IoT data which including data cleaning, data imputation, data normalization	
Input Parameters	**Type**	**Description**
teldata	Structured tabular data	Telemetry data from IoT devices
Output Parameters	**Type**	**Description**
clteldata	Structured tabular data	Cleaned telemetry data ready for model training/ testing
Notes		

Table 0-29: Threat Mitigation using Network Data

Operation name: Network Mitigation		
Description	An AI-driven mitigation model that dynamically determines appropriate actions in response to anomalies detected in network traffic.	
Input Parameters	**Type**	**Description**
clnfsdata	Structured tabular data	Cleaned data obtain from Network Data Preparation For Mitigation module
Output Parameters	**Type**	**Description**
mitv	Boolean vector	Mitigation actions represented in the form of binary vector. A value of 0 indicates no action is required, while a value of 1 signifies that the corresponding mitigation action should be executed.
Mitparav	Structured tabular data	Parameters required for mitigation actions specified in Table 0-26.
Notes		

Table 0-30: Threat Mitigation using Telemetry Data

Operation name: IoT Mitigation		
Description	An AI-driven mitigation model that dynamically determines appropriate actions in response to anomalies detected in telemetry traffic.	
Input Parameters	**Type**	**Description**
clteldata	Structured tabular data	Cleaned data obtain from Telemetry Data Preparation for Mitigation module
Output Parameters	**Type**	**Description**
mitv	Boolean vector	Mitigation actions represented in the form of binary vector, mitigation actions defined in Table 0-25, A value of 0 indicates no action is required, while a value of 1 signifies that the corresponding mitigation action should be executed.
Mitparav	Structured tabular data	Parameters required for mitigation actions specified in Table 0-26.
Notes		

Network Mitigation Model Building

The Network Mitigation module is primarily developed using the CSE-CIC-IDS2018 [Ima18] and Cryptoming [Man24] datasets. During data cleaning, null values—especially common in the *Flow Byts/s* and *Flow Pkts/s* columns—were imputed using the median values calculated within each attack type. Columns containing only zeros, which contributed minimal information, were removed. For essential features with scattered zeros, attack-type-specific medians were used for imputation. Illogical negative values in features such as *Flow Duration* and *Flow IAT Min* were also corrected. Table 0-31 present the distribution of attack types following the data cleaning process.

Table 0-31: Distribution of Cleaned CSE-CIC-IDS2018 Dataset

Attack_Type	Count	Percentage (%)
Benign	13484708	82.98
DdoS	1263933	7.78
DoS	654300	4.03
Brute Force	380949	2.34
Bot	286191	1.76
Infiltration	161934	1.00
Cryptomining	16789	0.10
Web Attack	928	0.01

Mixed-type columns with numeric data stored as strings were converted to numeric with error handling. Categorical variables *Label* and *Attack_Type* were label-encoded for machine learning compatibility.

After cleaning, feature names were standardized, skewed features were log-transformed, and all numerical features were scaled to [0,1] using MinMaxScaler.

Two feature selection methods—Boruta [Kur10] and Correlation + Mutual Information (CorrMI) [Gon24] — were applied. Boruta, a Random Forest-based wrapper method, identified the top 20 statistically relevant features. CorrMI removed highly correlated features (correlation > 0.85), then selected the top 20 based on mutual information scores.

These methods yielded two alternative feature sets used for model training and comparison. Table 0-32 lists the features selected by each method.

Table 0-32: Features Selected using (i) Boruta and (ii) Correlation and Mutual Information

Boruta	Correlation + Mutual Information
Dst Port	Idle Min
Pkt Len Min	Active Mean
Pkt Len Max	Active Std
Pkt Len Mean	Bwd Pkt Len Max
Pkt Len Std	Fwd Pkt Len Std
Pkt Len Var	Bwd Pkt Len Mean
FIN Flag Cnt	Fwd Pkt Len Max
SYN Flag Cnt	Fwd Pkt Len Mean
RST Flag Cnt	Bwd IAT Std
PSH Flag Cnt	Pkt Len Var
ACK Flag Cnt	Bwd Pkt Len Std
URG Flag Cnt	Bwd IAT Tot
CWE Flag Count	TotLen Fwd Pkts
ECE Flag Cnt	Bwd IAT Mean
Down/Up Ratio	Fwd Pkt Len Min
Pkt Size Avg	Init Fwd Win Byts
Fwd Seg Size Avg	Flow IAT Std

Bwd Seg Size Avg	Flow Byts/s
Idle Std	Bwd Pkts/s
Idle Mean	Dst Port

Using mitigation strategy shown in Table 0-25, attack types are mapped to one or more mitigation strategies. The distribution of threat mitigations for CSE-CIC-IDS2018 dataset is presented in

Table 0-33.

Table 0-33: Distribution of Threat Mitigations

Mitigation Code	No Action	Action
M1	9,819,963	1,554,849
M2	11,060,474	314,338
M3	9,718,361	1,656,451
M4	9,652,031	1,722,781
M5	9,640,279	1,734,533
M6	10,994,794	380,018
M7	10,031,399	1,343,413
M8	9,831,065	1,543,747
M9	11,048,722	326,090
M10	11,162,726	212,086
M11	11,363,060	11,752
M12	11,363,060	11,752
M13	11,108,148	266,664
M14	10,907,814	466,998
M15	9,819,963	1,554,849
M16	9,819,313	1,555,499

The dataset was stratified by attack type and divided into training, validation, and testing sets in a 70:15:15 ratio. The training and validation sets were used for model development, while the testing set was reserved for evaluating module performance in the Functional Testing subsection.

To support the recommendation of multiple mitigation strategies simultaneously, this study employed multi-label classification techniques, including Classifier Chain (CC), Binary Relevance (BR), and Long Short-Term Memory (LSTM) models. Multi-label classification is well-suited for cybersecurity, where a single anomaly may require several mitigation actions based on its severity, type, and location within the network.

IoTMitigation Model Building

The IoTMitigation module is primarily developed using the telemetry dataset from ToN-IoT's Train-Test dataset. The dataset consists of 7 IoT sensors. They are Fridge, Garage Door, GPS Tracker, Motion Light, Modbus, Thermostat and Weather sensors. The distribution of attack types by frequency and percentage are shown in the Table 0-34 and Table 0-35.

Table 0-34: Attack Types Distributions (by Frequency) of IoT Devices in ToN-IoT Train-Test Dataset

Attack_Type	Fridge	Garage Door	GPS Tracker	IoT Modbus	Motion Light	Thermostat	Weather
Backdoor	5000	5000	5000	5000	5000	5000	5000
DdoS	5000	5000	5000	0	5000	0	5000
Injection	5000	5000	5000	5000	5000	5000	5000
Normal	15000	15000	15000	15000	15000	15000	15000
Password	5000	5000	5000	5000	5000	5000	5000
Ransomware	2902	2902	2833	0	2264	2264	2865
Scanning	0	529	550	529	1775	61	529
XSS	2042	1156	577	577	449	449	866

Table 0-35: Attack Type Distributions (by Percentage) of IoT Devices in ToN-IoT Train-Test Dataset

Attack_Type	Fridge	Garage Door	GPS Tracker	IoT Modbus	Motion Light	Thermostat	Weather
Backdoor	12.52	12.63	12.83	16.07	12.66	15.26	12.74
DdoS	12.52	12.63	12.83	0.00	12.66	0.00	12.74
Injection	12.52	12.63	12.83	16.07	12.66	15.26	12.74
Normal	37.55	37.89	38.50	48.22	37.99	45.77	38.21
Password	12.52	12.63	12.83	16.07	12.66	15.26	12.74
Ransomware	7.27	7.33	7.27	0.00	5.73	6.91	7.30
Scanning	0.00	1.34	1.41	1.70	4.50	0.19	1.35
XSS	5.11	2.92	1.48	1.85	1.14	1.37	2.21

The features and their descriptions of the IoT devices are as shown in Table 0-36. The dataset was stratified according to attack type and partitioned into training, validation, and testing sets in a 60:20:20 ratio. Model development relied on the training and validation sets, while the testing set was reserved exclusively for assessing module performance in the Functional Testing Subsection.

To accommodate scenarios where multiple mitigation strategies may be required simultaneously, this study employed multi-label classification approaches, including Classifier Chain (CC), Binary Relevance (BR), and Long Short-Term Memory (LSTM) models. Such techniques are well-suited to cybersecurity contexts, as a single anomaly can necessitate several responses depending on its severity, category, and position within the network.

Table 0-36: Features of IoT Sources in ToN-IoT Dataset

IoT Source	Feature	Description
All	Ts	Timestamp of sensor reading data
	Date	Date of logging sensor's telemetry data
	Time	Time of logging sensor's telemetry data
	Label	Identify normal and attack records, where "0" indicates normal "1" indicates attacks
	Type	Record of normal or attack sub-classes
Fridge	Fridge_temperate	Temperature measurement of a fridge sensor
	Temp_condition	Temperature conditions of a fridge sensor, where temperature is high or low based on a predefined threshold value
Garage Door	Door_state	State of a door sensor where the door is closed or open
	Sphone_signal	State of receiving the door signal on a phone where the signal is true or false
GPS Tracker	Latitude	Latitude value of GPS tracker sensor
	Longitude	longitude value of GPS tracker sensor
Motion Light	Motion_status	Status of a motion sensor is either (on or off) where 1 indicates "on", and 0 indicates "off"
	Light_status	Status of a light sensor is either on or off
Modbus	FC1_Read_Input_Register	Modbus function code that is responsible for reading an input register
	FC2_Read_Discrete_Value	Modbus function code that oversees reading a discrete value
	FC3_Read_Holding_Register	Modbus function code that is responsible for reading a holding register
	FC4_Read_Coil	Modbus function code that is responsible for reading a coil
Thermostat	Current_temperature	Current Temperature reading of a thermostat sensor
	Thermostat_status	Status of a thermostat sensor is either on or off
Weather	Temperate	Temperature measurements of a weather sensor
	Pressure	Pressure readings of a weather sensor
	Humidity	Humidity readings of a weather sensor

Functional Tests of Mitigation Modules

The two main mitigation modules, Network Mitigation and IoT Mitigation, are respectively evaluated using the CSE-CIC-IDS2018 dataset and the telemetry dataset from ToN-IoT. We first evaluate the Network Mitigation module.

Functional Tests on Network Mitigation Module

The functional test results of the Network Mitigation module are presented in Table 0-37. As the module is still under development, the deployment container is not yet available. Consequently, the module cannot be automatically installed or configured at this stage. This issue is expected to be resolved upon completion of the development.

Table 0-37: Network Mitigation functional test

Name	Description	Passed (Yes/No/Partially)
Countermeasures test	Mitigation actions subject to network traffic.	Yes

The Network Mitigation module is tested using the testing dataset from CSE-CIC-IDS2018. The best-performing models are summarized in Table 0-38. For anomaly mitigation, a binary relevance multi-label classification technique—leveraging features selected based on correlation and mutual information—and Random Forest classifier outperformed the others. This technique has thus been selected as the primary mitigation model for the Network Mitigation module.

Table 0-38: Best Performing Models of Network Mitigation Module

Method	Model	Feature Selection Method	F1-Score	Accuracy	Precision	Recall
Anomaly Mitigation (Binary Relevance)	**Random Forest**	**Correlation+MI**	**0.9970**	**0.9997**	**1.0000**	**0.9941**
	LightGBM	Boruta	0.9969	0.9997	0.9999	0.9941
	XGBoost	Boruta	0.9969	**0.9997**	**1.0000**	0.994
Amonaly Mitigation (Classifier Chain)	**LightGBM**	**Boruta**	**0.9970**	**0.9997**	0.9999	**0.9941**
	XGBooost	Boruta	0.9969	**0.9997**	0.9999	0.9940
	Random Forest	Correlation+MI	0.9969	**0.9997**	**1.0000**	0.9939
Anomaly Mitigation (LSTM)	LSTM	Correlation+MI	0.9911	0.9994	0.9961	0.9864

Functional Tests on IoT Mitigation Module

Table 0-39 summarizes the functional testing outcomes for the IoT Mitigation module. Since development is still in progress, a deployment container has not yet been released. As a result, automated installation and configuration are currently unavailable. This limitation will be addressed once the module reaches full development.

Table 0-39: IoT Mitigation Functional Test

Name	Description	Passed (Yes/No/Partially)
IoT countermeasure test	Mitigation actions subject to network traffic.	Yes

The IoT Mitigation module was evaluated using the testing subset of the ToN-IoT [MouTon] Train-Test dataset, and the top-performing models are reported in Table 0-40. The single-layer LSTM model, trained with multi-offset sliding windows of length 12 and a stride of 3, delivered the best performance. This model

achieved accuracy, precision, recall, and F1 scores of 97.65%, 93.78%, 98.85%, and 95.62%, respectively, and was therefore selected as the primary mitigation model for the IoT Mitigation module.

Table 0-40: Anomaly Mitigation on IoT Devices using LSTM

Label	Accuracy	Precision	Recall	F1
M1	0.9420	0.8865	0.9741	0.9234
M2	0.9976	0.9976	0.9976	0.9976
M3	0.9128	0.7685	0.9595	0.8309
M4	0.9935	0.9936	0.9897	0.9916
M5	0.9929	0.9930	0.9887	0.9908
M6	0.9915	0.9842	0.9986	0.9912
M7	0.9986	0.9999	0.9973	0.9986
M8	0.9402	0.7503	0.9978	0.8334
M9	0.9939	0.9765	0.9975	0.9863
M10	0.9991	0.9916	0.9976	0.9944
M11	0.9988	0.9877	0.9976	0.9921
M14	0.9979	0.9995	0.9923	0.9959
M15	0.9557	0.9000	0.9745	0.9294
M16	0.9566	0.9005	0.9765	0.9309
Overall	**0.9765**	**0.9378**	**0.9885**	**0.9562**

1.5.2 Threat Predictor

This module is designed to predict threats based on network and telemetry traffic conditions, leveraging AI-based techniques to enable the security orchestrator to take preventive action.

Main functionalities

Subject to the constraint of the datasets, the key threats to be predicted in this project are as shown in Table 0-41. The approach leverages network flow statistics generated by CICFlowMeter and IoT device telemetry collected via the Programmable Monitoring Platform (PMT) described in Subsection 1.4.1. By analysing observed traffic patterns, the predictive module identifies potential threats and alerts the security orchestrator, enabling timely countermeasures against possible system compromises.

Table 0-41: Attack Types by Dataset

Attack_Type	CSE-CIC-IDS2018	ToN-IOT
Benign	√	√
DDoS	√	√
DoS	√	√
Brute Force	√	
Bot	√	
Infiltration	√	
Web Attack	√	
Backdoor		√
Injection		√
Ransomware		√
Scanning		√
XSS		√

Design updates

The architecture of the Threat Prediction System is illustrated in

Figure 0-34. Data is retrieved from the data fabric and processed through preprocessing, feature selection, and normalization. The refined dataset is then fed into the prediction model to predict potential attacks within the next five minutes. Two models—Binary Relevance and Long Short-Term Memory (LSTM)—are evaluated, and the best-performing model is selected for deployment to classify and predict anomalies in the network and IoT devices.

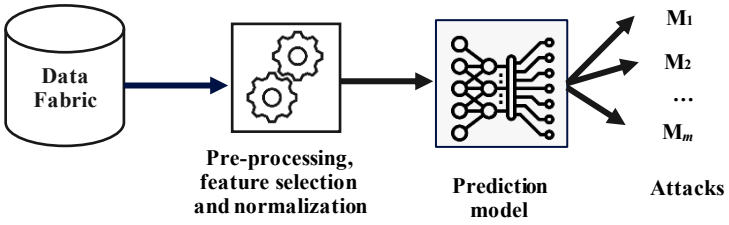


Figure 0-34: Threat Prediction Process Flow

Network Prediction Model Building

The Network Prediction module is primarily developed using the CSE-CIC-IDS2018 dataset. The raw data first undergoes a cleaning process as described in Subsection 0. Next, new columns are added to record the number of attacks for each attack type occurring within the subsequent 5-minute window.

The data is then consolidated into 1-second intervals, where each row represents one second of network activity. During this step, numerical attributes are aggregated using either mean or median values, while categorical features such as protocols and flag counts are assigned using the mode.

Since the CSE-CIC-IDS2018 dataset is structured such that each instance contains only benign traffic or a single attack type (with some attacks spanning across two days), attacks of the same type are merged and then split sequentially into training, validation, and testing sets with a ratio of 70:15:15. These sets are concatenated to ensure that temporal dependencies are preserved, as illustrated in Figure 0-35 to Figure 0-37. The distribution of training, validation, and testing data for each attack type is summarized in Table 0-41.

For feature selection, a total of 38 features were retained by combining the results from TSFresh, Mutual Information, and XGBoost-based methods. Each method captures different aspects of feature relevance, thereby producing a more comprehensive and robust feature set, as presented in Table 0-43 Table 0-43: Features Selected for Prediction on Consolidated CSE-CIC-IDS2018.

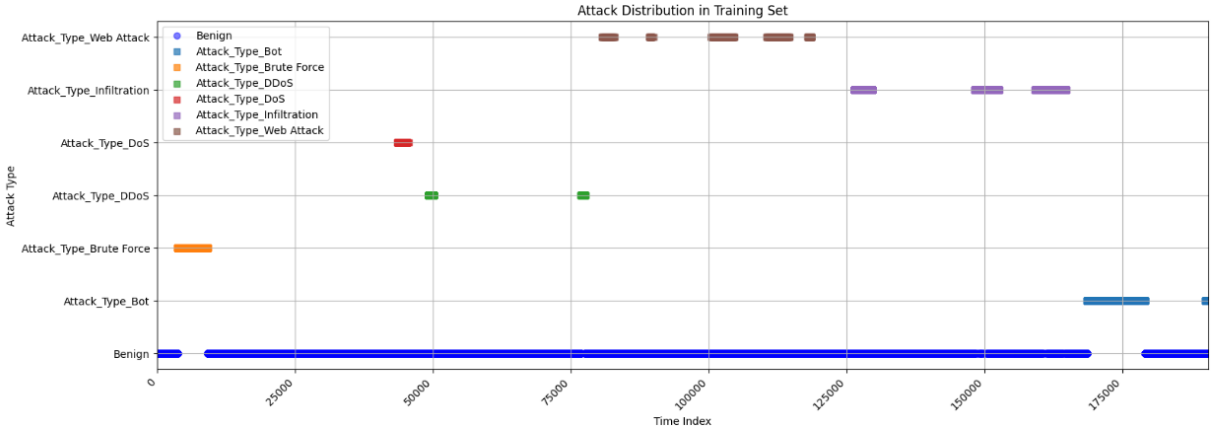


Figure 0-35: Final training set of CSE-CIC-IDS2018 formed by combining training subsets from all six attack types

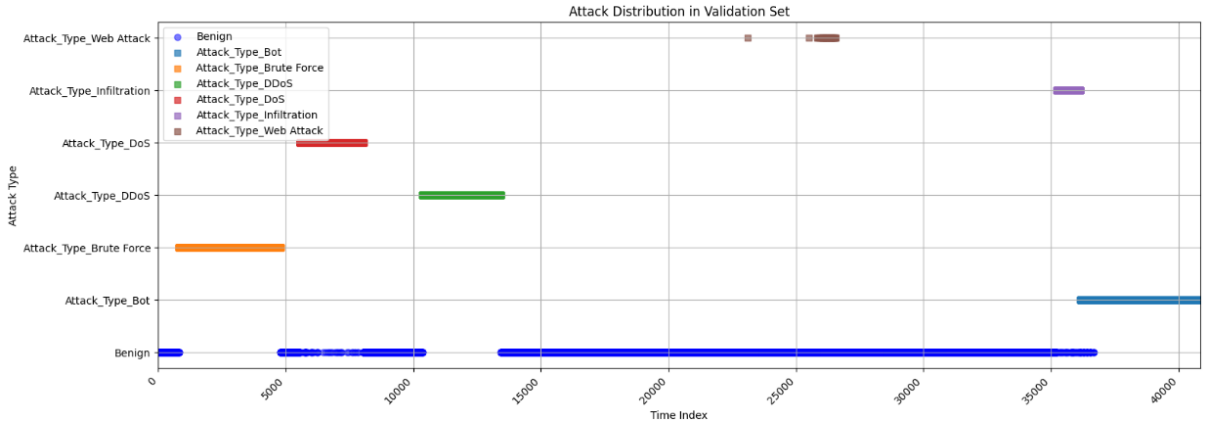


Figure 0-36: Final validation set of CSE-CIC-IDS2018 created by combining testing subsets across all six attack types

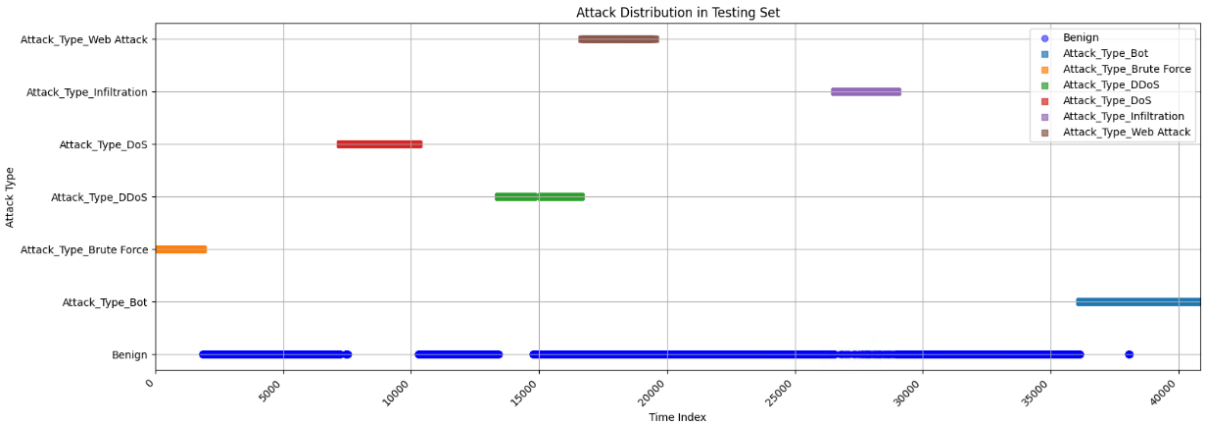


Figure 0-37: Final testing dataset of CSE-CIC-IDS2018 created by combining testing subsets across all six attack types

Table 0-42: Sample count and percentage distribution for each attack type in consolidated CSE-CIC-IDS2018

	Training Set		Validation Set		Testing Set	
Attack Type	**Count**	**Percentage**	**Count**	**Percentage**	**Count**	**Percentage**
Benign	159,593	83.71%	25,763	63.06%	26,611	65.13%
Bot	10,881	5.71%	4,661	11.41%	4,684	11.46%
Brute Force	5,442	2.85%	3,986	9.76%	1,840	4.50%
DDoS	1,155	0.61%	3,074	7.53%	2,327	5.69%
DoS	647	0.34%	2,449	5.99%	2,953	7.23%

Infiltration	12,511	6.56%	873	2.14%	2,254	5.52%
Web Attack	423	0.22%	46	0.11%	192	0.47%
Total	190,652	100%	40,852	100%	40,861	100%

Table 0-43: Features Selected for Prediction on Consolidated CSE-CIC-IDS2018

Dst Port	Protocol	Fwd Pkt Len Max	Flow IAT Max	Fwd IAT Max
Idle Max	Bwd Pkt Len Max	Bwd IAT Max	Flow IAT Mean	Pkt Len Std
Bwd Pkts/s	Fwd Seg Size Min	Fwd Pkt Len Std	Pkt Len Max	Pkt Len Var
Fwd Pkts/s	Fwd Act Data Pkts	Subflow Bwd Byts	Tot Fwd Pkts	Flow IAT Std
Active Max	Bwd Pkt Len Std	Subflow Fwd Pkt	PSH Flag Cnt	Bwd IAT Std
Active Min	Down/Up Ratio	Fwd IAT Min	ACK Flag Cnt	Fwd IAT Std
Active Mean	Bwd IAT Mean	Bwd Pkt Len Mean	TotLen Fwd Pkts	Fwd Seg Size Avg
Idle Min	Bwd Seg Size Avg	Fwd Pkt Len Mean	Attack_Type	

IoT Prediction Model Building

The processed dataset of ToN-IoT was used for prediction. The dataset was first cleaned and consolidated by aggregating records into one-second intervals, where each row represents a second of network activity. Categorical variables were reduced using the mode, with a random selection applied when no unique mode existed, while numeric variables were averaged. Attack type columns were aggregated by maximum to ensure any attack occurrence within the interval was preserved. Additionally, the normal traffic indicator was forced to zero whenever an attack flag appeared, making the labels mutually exclusive. Finally, categorical fields were converted into binary encodings to prepare the data for modeling.

To prevent temporal leakage, the data was split sequentially on a per-day basis. Each slice was divided into 80% training and 20% testing, with the training portion further split into 80/20 to yield final proportions of 64% training, 16% validation, and 20% testing. Rare attack categories, which had substantially fewer samples, were processed separately to ensure that each subset contained both benign and malicious examples. After splitting, the subsets were temporarily recombined to compute a future attack occurrence count using a 300-row forward-looking window, before restoring the original split boundaries.

Table 0-44 illustrate the consolidated processed telemetry dataset of the ToN-IoT. The distribution of the dataset is highly imbalanced, with normal traffic dominating across all devices and persisting as the majority class in every split. In contrast, categories such as ransomware, scanning, and XSS appear infrequently, leading to far fewer samples than other attack types. Detailed tables summarize per-device sample counts across the training, validation, and testing sets, while accompanying visualizations illustrate how different attacks are distributed across the timeline.

Table 0-44: Consolidated Processed Telemetry Dataset in ToN-IoT for Prediction

IoT Device	Attack Type	Count		
		Training Set	Validation Set	Testing Set
Fridge	Normal	64,753	16,202	21,574
	Backdoor	24,135	5,121	6,283
	DDos	2,433	1,405	1,438
	Injection	2,645	662	827

	Password	6,747	1,793	1,358
	Ransomware	1,755	439	549
	Scanning	338	85	106
	XSS	328	83	103
	Total	**103,134**	**25,790**	**32,238**
Garage Door	Normal	64,753	16,202	21,574
	Backdoor	24,135	5,121	6,283
	DDos	2,433	1,405	1,438
	Injection	2,645	662	827
	Password	6,747	1,793	1,358
	Ransomware	1,755	439	549
	Scanning	338	85	106
	XSS	328	83	103
	Total	**103,134**	**25,790**	**32,238**
GPS Tracker	Normal	81,591	22,177	29,098
	Backdoor	20,194	4,378	5,431
	DDos	6,272	1,843	1,853
	Injection	2,285	572	715
	Password	13,852	2,087	1,724
	Ransomware	1,521	381	476
	Scanning	352	88	110
	XSS	275	69	87
	Total	**126,342**	**31,595**	**39,494**
Modbus	Normal	89,677	27,570	33,173
	Backdoor	26,025	5,044	6,257
	Injection	2,918	730	912
	Password	15,419	171	1,553
	Scanning	338	85	106
	XSS	275	69	87
	Total	**134,652**	**33,669**	**42,088**
Motion Light	Normal	92,007	23,926	31,141
	Backdoor	16,377	3,435	4,255
	DDos	4,430	1,501	1,558
	Injection	2,572	643	804
	Password	9,944	1,880	1,461
	Ransomware	1,195	299	374
	Scanning	1,126	282	352
	XSS	269	68	85

	Total	**127,920**	**32,034**	**40,030**
Thermostat	Normal	93,272	24,235	30,420
	Backdoor	24,142	5,121	6,276
	Injection	3,089	773	966
	Password	4,403	1,102	1,378
	Ransomware	1,195	299	374
	Scanning	38	10	13
	XSS	269	68	85
	Total	**126,408**	**31,608**	**39,512**
Weather	Normal	82,038	22,015	28,369
	Backdoor	16,558	3,467	4,298
	DDos	2,732	1,566	1,659
	Injection	3,169	793	991
	Password	14,102	1,814	1,754
	Ransomware	1,240	310	388
	Scanning	338	85	106
	XSS	332	84	104
	Total	**120,509**	**30,134**	**37,669**

Initial Prototype Implementation

The prediction module is developed in Python using machine learning frameworks including PyTorch, TensorFlow, and Scikit-learn. Table 0-45 to Table 0-48 illustrate the implemented functionalities designed to prediction module for network and IoT devices. The module's API will be built using FastAPI, exposing a RESTful interface for communication with the security orchestrator. Real-time flow data will be ingested through Kafka using a publish-subscribe mechanism. The entire system will be containerized with Docker to ensure scalable and portable deployment.

Table 0-45: Data Preparation on Network Flow Statistics for Predictive Module

Operation name: NetworkDataPreparationForPrediction		
Description	Perform preprocessing on network data which including data cleaning, data imputation, data normalization and feature selection	
Input Parameters	**Type**	**Description**
nfsdatav	Array of structured tabular data	Network Flow Statistics extracted by CICFlowmeter from the network traffic
Output Parameters	**Type**	**Description**
clnfsdatav	Array of structured tabular data	Cleaned, normalized and selected network flow statistics data ready for model training / testing
Notes		

Table 0-46: Data Preparation on Telemetry Data for Predictive Module

Operation name: TelemetryDataPreparationForPredictive		
Description	Perform preprocessing on IoT data which including data cleaning, data imputation, data normalization	
Input Parameters	**Type**	**Description**
teldatav	Array of structured tabular data	Telemetry data from IoT devices
Output Parameters	**Type**	**Description**
clteldatav	Array of structured tabular data	Cleaned telemetry data ready for model training/ testing
Notes		

Table 0-47: Threat Prediction using Network Data

Operation name: NetworkPrediction		
Description	An AI-driven predictive model that predict the potential anomalies in next 5 minutes in network environment.	
Input Parameters	**Type**	**Description**
clnfsdatav	Array of structured tabular data	Cleaned data obtain from NetworkDataPreparationForPrediction module
Output Parameters	**Type**	**Description**
predv	Boolean vector	Predicted threats are represented as a binary vector, as defined in Column 3 of Table 0-41. A value of 0 indicates that no anomaly is expected, whereas a value of 1 signifies that the corresponding threat is predicted to occur within the next 5-minute interval.
Notes		

Table 0-48: Threat Prediction using Telemetry Data

Operation name: IoTMitigation		
Description	An AI-driven predictive model that predict potential anomalies in next 5 minutes interval on IoT Devices.	
Input Parameters	**Type**	**Description**
clteldatav	Array of structured tabular data	Cleaned data obtain from TelemetryDataPreparationForPrediction module
Output Parameters	**Type**	**Description**
prev	Boolean vector	Predicted threats are represented as a binary vector, as defined in Column 3 of Table 0-41. A value of 0 indicates that no anomaly is expected, whereas a value of 1 signifies that the corresponding threat is predicted to occur within the next 5-minute interval.
Notes		

Functional Tests of Prediction Module

The two main prediction modules, Network Prediction and IoT Prediction, are respectively evaluated using the CSE-CIC-IDS2018 dataset and the telemetry dataset from ToN-IoT. We first evaluate the Network Mitigation module.

Functional Tests on NetworPrediction Module

The functional test results of the Network Prediction module are presented in Table 0-49. As the module is still under development, the deployment container is not yet available. Consequently, the module cannot be automatically installed or configured at this stage. This issue is expected to be resolved upon completion of the development.

Table 0-49: Network Prediction Functional Test

Name	Description	Passed (Yes/No/Partially)
Key verification	Mitigation actions subject to network traffic.	Yes

The Network Prediction module is evaluated using the testing dataset from CSE-CIC-IDS2018. The top-performing models are summarized in Table 0-50. Among them, Binary Relevance with XGBoost achieved the best results, with F1-Score of 91.80%, Accuracy of 80.50%, Precision of 91.85%, and Recall of 91.75%. This is followed by the 3-Layer Stacked Bidirectional LSTM with a 300-window classifier, which obtained an F1-Score of 89.95%, Accuracy of 85.44%, Precision of 90.42%, and Recall of 89.49%. These two techniques have therefore been selected as the primary prediction models for the Network Prediction module.

Table 0-50: NetworkPrediction using 38 selected features and attack type

Model		F1-Score	Accuracy	Precision	Recall
Binary Relevance	Random Forest	0.8853	0.7949	0.9146	0.8577
	Light GBM	0.8811	0.7189	0.8867	0.8756
	Logistic Regression	0.8863	0.7517	0.9155	0.8589
	XGBoost	**0.9180**	**0.8050**	**0.9185**	**0.9175**
	Hist Gradient Boosting	0.8868	0.7309	0.8873	0.8863
	Gradient Boosting	0.8638	0.7551	0.9197	0.8143
3-Layer Stacked Bidirectional LSTM	Window = 5	0.7930	0.7098	0.7671	0.8207
	Window = 10	0.8520	0.8020	0.8730	0.8320
	Window = 30	0.8780	0.8325	0.8998	0.8572
	Window = 300	**0.8995**	**0.8544**	**0.9042**	**0.8949**

Functional Tests on IoT Prediction Module

The functional test results of the IoT Prediction module are presented in Table 0-51. As the module is still under development, the deployment container is not yet available. Consequently, the module cannot be automatically installed or configured at this stage. This issue is expected to be resolved upon completion of the development.

Table 0-51: IoT Prediction Functional Test

Name	Description	Passed (Yes/No/Partially)
Key verification	Mitigation actions subject to network traffic.	Yes

The IoT Prediction module was evaluated using the processed *ToN-IoT* dataset, with the top-performing models summarized in Table 0-52. Rows in bold indicate the best-performing models. Binary Relevance with XGBoost, Hist Gradient Boosting, LightGBM, Random Forest, and XGBoost classifiers achieved the highest performance on the Fridge, Weather, Modbus, Motion Light, and Garage Door devices, respectively.

Meanwhile, the Hybrid 3-Layer Sequential BiLSTM+CNN performed best on the Thermostat and GPS Tracker devices. These techniques were therefore selected as the primary prediction models for the IoTPrediction module.

Table 0-52: Prediction Results for IoT Devices

Device	Model	Classifier	F1	Accuracy	Precision	Recall
Fridge	Binary Relevance	Random Forest	0.9252	0.9738	0.9354	0.9154
		Light GBM	0.9255	0.9743	0.9509	0.9014
		Logistic Regression	0.9016	0.9667	0.9445	0.8624
		XGBoost	**0.9287**	**0.9755**	**0.9565**	**0.9025**
		Hist Gradient Boosting	0.9219	0.9732	0.9498	0.8956
		Gradient Boosting	0.9052	0.9683	0.9615	0.8552
	LSTM	3-Layer Bidirectional LSTM	0.9025	0.9113	0.9661	0.8467
		Hybrid 3-Layer Sequential BiLSTM+CNN	0.9154	0.8636	0.9357	0.896
Weather	Binary Relevance	Random Forest	0.898	0.9703	0.9427	0.8573
		Light GBM	0.914	0.9753	0.9711	0.8633
		Logistic Regression	0.8851	0.9656	0.901	0.8697
		XGBoost	0.9099	0.9741	0.9695	0.8573
		Hist Gradient Boosting	**0.9141**	**0.9754**	**0.9761**	**0.8596**
		Gradient Boosting	0.9069	0.9732	0.9644	0.8559
	LSTM	3-Layer Bidirectional LSTM	0.9112	0.9323	0.97	0.8591
		Hybrid 3-Layer Sequential BiLSTM+CNN	0.8815	0.9052	0.9068	0.8576
Modbus	Binary Relevance	Random Forest	0.9077	0.966	0.963	0.8584
		Light GBM	**0.9267**	**0.9732**	**0.9896**	**0.8713**
		Logistic Regression	0.9051	0.9647	0.9513	0.8631
		XGBoost	0.9211	0.9712	0.9884	0.8624
		Hist Gradient Boosting	0.9226	0.9713	0.9734	0.8767
		Gradient Boosting	0.9186	0.9704	0.9883	0.858
	LSTM	3-Layer Bidirectional LSTM	0.9229	0.9575	0.9986	0.8578
		Hybrid 3-Layer Sequential BiLSTM+CNN	0.9231	0.9451	0.9918	0.8633
Motion Light	Binary Relevance	**Random Forest**	**0.9314**	**0.9795**	**0.9482**	**0.9152**
		Light GBM	0.925	0.9781	0.9671	0.8864

		Logistic Regression	0.8991	0.9704	0.9324	0.8682
		XGBoost	0.9262	0.9788	0.9871	0.8723
		Hist Gradient Boosting	0.9197	0.9766	0.9631	0.88
	LSTM	3-Layer Bidirectional LSTM	0.9165	0.9279	0.98	0.8607
		Hybrid 3-Layer Sequential BiLSTM+CNN	0.9258	0.9391	0.9677	0.8874
Thermostat	Binary Relevance	Random Forest	0.9205	0.975	0.9497	0.893
		Light GBM	0.9439	0.9823	0.9665	0.9224
		Logistic Regression	0.9014	0.9686	0.9164	0.8868
		XGBoost	0.9359	0.9803	0.9892	0.8879
		Hist Gradient Boosting	0.9314	0.9785	0.9644	0.9007
	LSTM	3-Layer Bidirectional LSTM	0.929	0.9763	0.9888	0.8761
		Hybrid 3-Layer Sequential BiLSTM+CNN	**0.9489**	**0.975**	**0.9774**	**0.9221**
GPS Tracker	Binary Relevance	Random Forest	0.8911	0.9685	0.9613	0.8305
		Light GBM	0.8972	0.9683	0.9029	0.8916
		Logistic Regression	0.8481	0.9518	0.8311	0.8658
		XGBoost	0.8935	0.9676	0.912	0.8757
		Hist Gradient Boosting	0.9042	0.9703	0.9079	0.9005
	LSTM	3-Layer Bidirectional LSTM	0.9102	0.9284	0.9872	0.8443
		Hybrid 3-Layer Sequential BiLSTM+CNN	**0.9137**	**0.9269**	**0.9752**	**0.8595**
Garage Door	Binary Relevance	Random Forest	0.8847	0.9664	0.9717	0.812
		Light GBM	0.9016	0.9708	0.9698	0.8424
		Logistic Regression	0.8833	0.9644	0.9201	0.8493
		XGBoost	**0.9137**	**0.9739**	**0.9606**	**0.8712**
		Hist Gradient Boosting	0.9027	0.9707	0.9549	0.8558
	LSTM	3-Layer Bidirectional LSTM	0.8962	0.8741	0.9714	0.8317
		Hybrid 3-Layer Sequential BiLSTM+CNN	0.9111	0.8595	0.9454	0.8792

3. Conclusions

The document aims to report the latest information regarding the zero-touch security management platform in Month 21 (M21). Due to the fact that the ROBUST-6G project merged D4.2, which focused on design and early prototypes, and D4.3, which was mainly focused on the final design, this deliverable covered several aspects related to the design and development phases for security orchestration, pervasive monitoring, and threat detection, as well as incident prediction.

First, the document discussed the principal updates with respect to the Zero-Touch Security Platform that is the nexus of WP4 by joining all key pillars: i) security service and resource orchestration, ii) continuous monitoring and threat detection, and iii) incident prediction and continuous mitigation. In this sense, new components and relationships have appeared to cover the flow, scenarios, and use cases in WP6. For i), the zero-touch security orchestrator, security closed-loop management, security resource orchestrator, and risk-averse management framework cover all components envisioned under this pillar. Beyond the new links between components and subcomponents, there were some major updates in terms of design.

For example, the Semantic Translation functionality has been renamed to Security Context Management. Therefore, these aspects required an update version of the ZTSO functional architecture. Besides, ZTSO also delivered a software component architecture with all the technologies for the northbound and southbound layers. Moreover, a generative AI approach for the generation of automatic security remediations alongside SOAR frameworks has been added to the ZTSO architecture, to improve the Act phase of the security closed loops architecture. The security closed-loop management also delivered updates to the prototype by reporting S-CL and S-CL Function descriptors, as well as OpenAPI interfaces. Additionally, both ZTSO and S-CL demonstrated their proper behaviour via functional tests. Regarding the security resource orchestrator, the main updates were related to the release of a software architecture and the definition of functional tests, ensuring the stability of the high-level architecture presented in D4.1. In contrast, the latest component of i), the risk-averse management framework, introduced modifications for the former high-level architecture since two new advancements. The generalization of CPT utility functions offers more flexibility and the use of novel optimization algorithms.

Secondly, the continuous monitoring and threat detection pillar gathered significant updates, mostly oriented towards software architectures and functional testing. In this case, the pillar was split into three different components: a) programmable monitoring platform (PMP), b) semantic-aware anomaly detection, and c) rule-based threat detection. When it comes to the PMP, a slight update of the former high-level architecture was conducted, as some components have been moved outside to focus on monitoring activities rather than detection. This was the reason for a new functionality, rule-based detection. ROBUST-6G handles two types of threat detections by using AI models (semantic-aware anomaly detection) and rules (rule-based threat detection). The former remained the high-level architecture presented in D4.1, whilst the latter described how the Alert and Notification Module interacted with the PMP to inform when misbehaviours appeared in network traffic. In terms of functional tests, the PMP and both types of threat detection solutions clustered several functional tests to guarantee that components worked properly before testing interactions in WP6 with other modules. Note that some software components in a more mature phase also presented Open API specifications or initial interfaces, like the PMP or the semantic-aware anomaly detection.

Thirdly, the incident prediction and continuous mitigation pillar was mainly divided into two components: the threat mitigation module and the threat predictor. In this case, both components maintained the previous high-level architecture, and the main novelties were associated with the interfaces to consume and request information from the AI models.

In summary, D4.3 compiles the final iteration for the design phase of components associated with the Zero-Touch Security Platform and delivers a first round of development and software architectures for building an early prototype. Such a prototype is expected to continue evolving over the next six months, nearing the end of the project and the submission of the final prototype deliverable, D4.4, in M27.

4. References

[R6G24-D22] Deliverable D4.1: Security Automation for 6G. Horizon Europe Project, Grant Agreement No. 101139068.

[800-61-R3] National Institute of Standards and Technology. Computer Security Incident Handling Guide (SP 800-61 Rev. 3). Gaithersburg, MD: U.S. Department of Commerce, May 2024

[ETS002] ETSI GS ZSM 002, "Zero-touch network and Service Management (ZSM); Reference Architecture", 2019.

[IBN19] Islam, Chadni & Ali Babar, Muhammad & Nepal, Surya. (2019). An Ontology-Driven Approach to Automating the Process of Integrating Security Software Systems. 54-63. 10.1109/ICSSP.2019.00017.

[Sig25] Sigma rules, 2025, Online access on September 23rd, 2025: https://sigmahq.io/docs/guide/about .

[HXII-D63] Deliverable D6.3: Initial Design of 6G Smart Network Management Framework. Horizon Europe HEXA-X-II Project, Grant Agreement No. 101095759.

[PMP25a] PMP swagger APIs, 2025, Online access on September 23rd, 2025: https://cyberdatalab.github.io/ROBUST-6G_PMP/ .

[Mid25] GitHub of Machine ID repository, 2025, Online access on September 23rd, 2025: https://github.com/doroved/mid .

[Tsh25] Tshark official website, 2025, Online access on September 23rd, 2025: https://tshark.dev/ .

[Alp25] Alpine official image in DockerHub repository, 2025, Online access on September 23rd, 2025: https://hub.docker.com/_/alpine .

[Lib16] GitHub of Libcap repository, 2016, Online access on September 23rd, 2025: https://github.com/mhiramat/libcap/tree/master .

[Pyt25] Python3 official website, 2025, https://www.python.org/ .

[Lin25] Linux man page about capabilities, 2025, Online access on September 23rd, 2025: https://linux.die.net/man/7/capabilities .

[Jso25] Json format official website, 2025, Online access on September 23rd, 2025: https://www.json.org/json-en.html .

[Tel25a] Telegraf official website, 2025, Online access on September 23rd, 2025: https://www.influxdata.com/time-series-platform/telegraf/ .

[Tel25b] Telegraf official image in DockerHub repository, 2025, Online access on September 23rd, 2025: https://hub.docker.com/layers/library/telegraf/1.33.2/images/sha256-0e20b846d9899ae9b86688ab34e6d3e836319fe78f596145f84f3a1afd84493a .

[Flu25a] Fluentd official website, 2025, Online access on September 23rd, 2025: https://www.fluentd.org/ .

[Flu25b] Fluentd official image in DockerHub, 2025, Online access on September 23rd, 2025: https://hub.docker.com/layers/library/fluentd/v1.16.8-debian-1.0/images/sha256-2e8c002c57ca2e5b99734e99a6d035d828049b1659e7b39415a6caf5364cd181 .

[Fal25a] Falco official website, 2025, Online access on September 23rd, 2025: https://falco.org/ .

[Ebp23] Modern eBPF probe is ready to shine in Falco, 2023, Online access on September 23rd, 2025: https://falco.org/blog/falco-modern-bpf-0-35-0/ .

[Fal25b] Falco official image in DockerHub, 2025, Online access on September 23rd, 2025: https://hub.docker.com/layers/falcosecurity/falco/0.40.0/images/sha256-287cc7dbbb2705d4c5ba02158b8a72dc9cccf49a94031da0ba8f53f25fe30295 .

[Fil25] Filbeat official website, 2025, Online access on September 23rd, 2025: https://www.elastic.co/beats/filebeat .

[Fil24] Filbeat official image in DockerHub, 2025, https://hub.docker.com/layers/elastic/filebeat/8.16.2/images/sha256-d108d5e90c6d2b727dff4487ffb318940a0568ce938270101079eaca563545d0 .

[Kaf25a] Apache Kafka official website, 2025, Online access on September 23rd, 2025: https://kafka.apache.org/ .

[Kaf25b] Telegraf's Kafka output plugin, 2025, Online access on September 23rd, 2025: https://github.com/influxdata/telegraf/blob/release-1.35/plugins/outputs/kafka/README.md .

[Kaf24] Kafka official image in DockerHub, 2024, Online access on September 23rd, 2025:https://hub.docker.com/layers/apache/kafka/3.9.0/images/sha256-5954ac731e7d22c29459b38903078f11ffa067d4ced93fc8cb20d838dec0ab76 .

[PMP25b] PMP official GitHub repository, 2025, Online access on September 23rd, 2025: https://github.com/CyberDataLab/ROBUST-6G_PMP/ .

[Kaf22] Kafkacat official GitHub repository, 2022, Online access on September 23rd, 2025: https://github.com/edenhill/kcat .

[Jso24] JSON2PCAP official GitHub repository, 2024, Online access on September 23rd, 2025: https://github.com/H21lab/json2pcap .

[Sno25] Snort3 official website, 2025, Online access on September 23rd, 2025: https://www.snort.org/ .

[Lib25a] Libdaq official GitHub repository, 2025, Online access on September 23rd, 2025: https://github.com/snort3/libdaq.git .

[Lib25b] Libpcre2 official GitHub repository, 2025, Online access on September 23rd, 2025: https://github.com/PCRE2Project/pcre2 .

[Wsag11] Web Services Agreement Specification (WS-Agreement), 2011, https://ogf.org/documents/GFD.192.pdf

[ssla25a] SSLA Manager openAPI specifications, ssla-manager-rest/openapi.yaml at main · ThalesGroup/ssla-manager-rest GitHub, Online access on September 23rd, 2025: ssla-manager-rest/openapi.yaml at main · ThalesGroup/ssla-manager-rest GitHub

[specs15] Valentina Casola, Alessandra De Benedictis, Massimiliano Rak, Villano Umberto, "SLA-Based Secure Cloud Application Development: The SPECS Framework", https://www.researchgate.net/figure/The-SPECS-application-use-cases_fig4_300415215

[VK25] S. Vaidanis and M. Kountouris, "Generalizing Utility and Loss Aversion in Cumulative Prospect Theory: A Theoretical Perspective", submitted to Journal of Risk and Uncertainty, June 2025

[Nex25a] Nextworks (Italy). (2025). *Zero-Touch-Security-Orchestrator Ontology Manager API*. Zenodo. Online access on September 23rd, 2025: https://doi.org/10.5281/zenodo.16925313

[Nex25b] Nextworks (Italy). (2025). *Zero-Touch-Security-Orchestrator Catalogues Manager API*. Zenodo. Online access on September 23rd, 2025: https://doi.org/10.5281/zenodo.16925327

[Nex25c] Nextworks (Italy). (2024). Closed Loop Governance - Catalogue API. Zenodo. Online access on September 23rd, 2025: https://doi.org/10.5281/zenodo.14193484

[Nex25d] Nextworks (Italy). (2024). Closed Loop Governance LCM API. Zenodo. Online access on September 23rd, 2025: https://doi.org/10.5281/zenodo.14193635

[Nex25e] Nextworks (Italy). (2024). Multi-cluster extreme-edge resource orchestration API. Zenodo. Online access on September 23rd, 2025: https://doi.org/10.5281/zenodo.14193659

[aiact24] Article 14: Human Oversight | EU Artificial Intelligence Act, 2024, https://artificialintelligenceact.eu/article/14/

[zsm21a] GS ZSM 009-1 - V1.1.1 - Zero-touch network and Service Management (ZSM); Closed-Loop Automation; Part 1: Enablers, 2021, https://www.etsi.org/deliver/etsi_gs/ZSM/001_099/00901/01.01.01_60/gs_ZSM00901v010101p.pdf

[zsm24a] GR ZSM 011 - V2.1.1 - Zero-touch network and Service Management (ZSM); Intent-driven autonomous networks; Generic aspects, 2024, https://www.etsi.org/deliver/etsi_gr/ZSM/001_099/011/02.01.01_60/gr_ZSM011v020101p.pdf

[36.888] 3GPP TR 36.888, "Study on provision of low-cost Machine-Type Communications (MTC) User Equipments (UEs) based on LTE (Release 12)", June 2013.

[Aba16] Abadi, Martín, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, et al. 2016. "TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems." *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*: 265–283. https://www.tensorflow.org/

[AppCic]Applications: CICFlowMeter / CIC-AB. Available online: https://www.unb.ca/cic/research/applications.html (accessed on 7 December 2024)[Bou04] Boutell, M. R., Luo, J., Shen, X., & Brown, C. M. (2004). Learning multi-label scene classification. Pattern recognition, 37(9), 1757-1771.

[FU98] G. D. Forney and G. Ungerboeck, "Modulation and coding for linear Gaussian channels", IEEE Transactions on Information Theory, vol. 44, no. 6, pp. 2384-2415, October 1998.

[Gon24]Gong, Huanhuan, Yanying Li, Jiaoni Zhang, Baoshuang Zhang, and Xialin Wang. 2024. "A New Filter Feature Selection Algorithm for Classification Task by Ensembling Pearson Correlation Coefficient and Mutual Information." *Engineering Applications of Artificial Intelligence* 131 (May): Article 107865. https://doi.org/10.1016/j.engappai.2024.107865[Hoc97] Hochreiter, S. & Schmidhuber, J. (1997). Long short-term memory. Neural Comput, 9(8), 1735-1780.

[Ima18] Iman Sharafaldin, Arash Habibi Lashkari, and Ali A. Ghorbani, "Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization", 4th International Conference on Information Systems Security and Privacy (ICISSP), Portugal, January 2018

[Kur10] Kursa, M. B., & Rudnicki, W. R. (2010). Feature Selection with the Boruta Package. Journal of Statistical Software, 36(11), 1–13. https://doi.org/10.18637/jss.v036.i11

[Man24] Mannella, L., Canavese, D., & Regano, L. (2024, April). Detecting Cryptomining Traffic in IoT Networks. In Proceedings of the 9th International Conference on Smart and Sustainable Technologies–SpliTech 2024. Institute of Electrical and Electronics Engineers (IEEE).

[Maz75] J. E. Mazo, "Faster-than-Nyquist signaling", Bell System Technical Journal, vol. 54, no. 8, pp. 1451-1462, October 1975.

[MouTon] Moustafa, N. ToN_IoT Datasets. Available online: https://research.unsw.edu.au/projects/toniot-datasets (accessed on 2 October 2024).

[Pas19]Paszke, Adam, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. "PyTorch: An Imperative Style, High-Performance Deep Learning Library." *Advances in Neural Information Processing Systems* 32: 8026–8037. https://arxiv.org/abs/1912.01703

[Ped11]Pedregosa, Fabian, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, et al. 2011. "Scikit-learn: Machine Learning in Python." *Journal of Machine Learning Research* 12 (85): 2825–2830. http://jmlr.org/papers/v12/pedregosa11a.html

[Rea11] Read, J., Pfahringer, B., Holmes, G., & Frank, E. (2011). Classifier chains for multi-label classification. Machine learning, 85, 333-359.

[Ram18] Ramírez, Sebastián. 2018. *FastAPI*. FastAPI Project. Online access on September 23rd, 2025: https://fastapi.tiangolo.com/

[TAZ+13] A. Tzanakaki, M. P. Anastasopoulos, G. S. Zervas, B. R. Rofoee, R. Nejabati and D. Simeonidou, "Virtualization of heterogeneous wireless-optical network and IT infrastructures in support of cloud and mobile cloud services". IEEE Communications Magazine, vol. 51, no. 8, pp. 155-161, August 2013.

[VSK25] S. Vaidanis, P. A. Stavrou, and M. Kountouris, "Optimization for semantic-aware resource allocation under CPT-based utilities, in Proc. IEEE Workshop on Signal Processing and Artificial Intelligence for Wireless Communications (SPAWC'25), 7-10 July 2025, Surrey, United Kingdom.

[CACAO23] CACAO Security Playbooks v2.0 - OASIS Open,Online access on September 23rd, 2025: https://www.oasis-open.org/standard/cacao-security-playbooks-v2-0/

[OpenC222] OASIS Open Command and Control (OpenC2) TC | OASIS, Online access on September 23rd, 2025: https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=openc2

[Oc2Http23] CACAO Security Playbooks Version 2.0 - OpenC2 HTTP Command, Online access on September 23rd, 2025: https://docs.oasis-open.org/cacao/security-playbooks/v2.0/cs01/security-playbooks-v2.0-cs01.html#_Toc152256498

[Oc2Elastic23] CACAO Security Playbooks Version 2.0 - OpenC2 Elastic Command, Online access on September 23rd, 2025: https://docs.oasis-open.org/cacao/security-playbooks/v2.0/cs01/security-playbooks-v2.0-cs01.html#_Toc152256494

[Oc2Jupyter23] CACAO Security Playbooks Version 2.0 - OpenC2 Elastic Command, Online access on September 23rd, 2025: https://docs.oasis-open.org/cacao/security-playbooks/v2.0/cs01/security-playbooks-v2.0-cs01.html#_Toc152256496

[CVE20234914] NVD - CVE-2023-4914, Online access on September 23rd , 2025: https://nvd.nist.gov/vuln/detail/CVE-2023-4914

[SPR25] Spring Framework. Online access on September 23rd, 2025: https://spring.io/projects/spring-framework

[JEN25] Jena Apache. Online access on September 23rd, 2025: https://jena.apache.org/

[FUS25] Apache Jena Fuseki. Online access on September 23rd, 2025: https://jena.apache.org/documentation/fuseki2/

Annex

3.1 OpenAPI specifications

Policy Manager

```yaml
openapi: 3.0.3
info:
 title: Robust Policy Manager
 description: Rest API to manage security policy for ROBUST-6G
 version: 0.1.0
servers:
 - url: 'http://localhost:8080'
paths:
 /api/v1/ssla:
   post:
     summary: "Submit new SSLA"
     requestBody:
       required: true
       content:
         multipart/form-data:
           schema:
             properties:
               ssla:
                 type: string
                 format: binary
     responses:
       201:
```

```
        description: SSLA submitted successfully
```

3.2 Ontology and Knowledge graph of OM

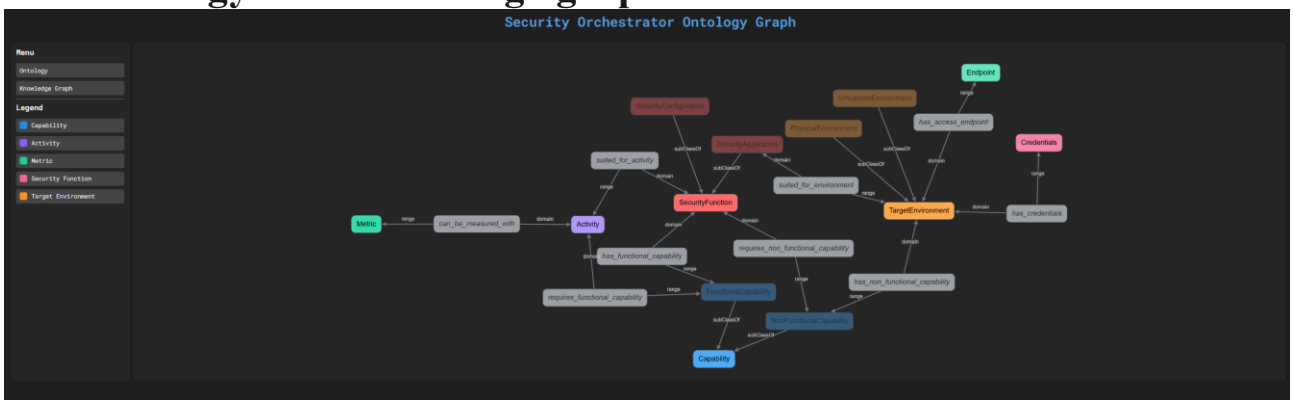


Figure 0-1: Ontology representation

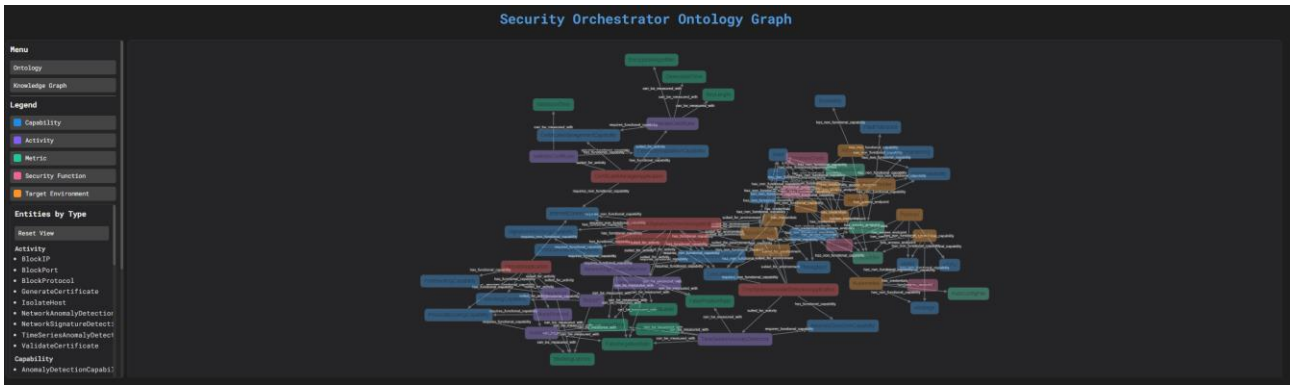


Figure 0-2: Knowledge Graph representation

3.3 S-CL and S-CLFs Descriptors Examples


```
{
  "name": "proactive-nw-attack-monitoring",
  "description": "Closed-loop for proactive detection and mitigation of network attacks using PMP telemetry, AI-based IDS analysis,
CACAO decision playbooks, and orchestrated execution.",
  "version": "1.0",
  "vendor": "NXW",
  "priority": "HIGH",
  "timingCategory": "NEAR_REAL_TIME",
  "reactionCategory": "PROACTIVE",
  "targetLayer": ["SERVICE", "NETWORK"],
  "cognitiveCl": true,
  "clCoordinationSupport": true,
  "goals": [
    {
      "description": "Detect and mitigate network attacks with minimal response time.",
      "statement": "Ensure continuous security posture by promptly identifying attacks and enforcing mitigations.",
      "goalType": "SECURITY_ASSURANCE",
      "goalConditions": [
        {
          "description": "AI-driven monitoring and detection of DoS, DDoS, backdoor and related network threats.",
          "conditionType": "attack_detection",
          "inputMetrics": ["net.flow.rate", "net.anomaly.score", "ids.alerts.count"],
          "targetConditions": [
            "attack_detected => mitigation_applied",
            "false_positive_rate <= threshold"
          ],
          "goalActions": [
            {
              "description": "Enforce the remediation defined in the CACAO playbook (e.g., block IP, rate-limit, isolate service).",
              "actionType": "re-configuration",
              "actionExecutorType": "service orchestrator",
              "actionTargets": ["SERVICE", "NETWORK"]
            }
          ]
        }
      ]
    }
  ],
  "components": [
    {
      "functionDescriptorId": "<filled-by-catalog>",
      "functionDescriptorType": "MONITORING",
      "functionDescriptorName": "cl-monitoring-nw",
      "sharable": false
    },
    {
      "functionDescriptorId": "<filled-by-catalog>",
      "functionDescriptorType": "ANALYSIS",
      "functionDescriptorName": "cl-analysis-ids",
      "sharable": false
    },
    {
      "functionDescriptorId": "<filled-by-catalog>",
      "functionDescriptorType": "DECISION",
      "functionDescriptorName": "cl-decision-cacao",
      "sharable": false
    },
    {
      "functionDescriptorId": "<filled-by-catalog>",
      "functionDescriptorType": "EXECUTION",
      "functionDescriptorName": "cl-execution-orchestrator",
      "sharable": false
    }
  ]
}
```

Monitoring Function (PMP network telemetry subscription)

```
{
  "functionDescriptorType": "MONITORING",
  "description": "Subscribe to PMP and forward network telemetry/packet traces to the CL bus.",
  "name": "cl-monitoring-nw",
  "version": "1.0",
  "orchestrated": true,
  "sharable": false,
  "orchestratorTemplate": {
    "name": "cl-function",
    "version": "e3a34070-74db-46c5-be28-abc6076d03c1"
  },
  "inputParameters": [
    "pmpEndpoint",
    "dataSourceId",
    "filters",
    "outTopic"
  ],
  "requirements": {
    "cpu": 0.128,
    "memory": 264241152,
    "storage": 0
  },
  "replicas": 1
}
```

Analysis Function (AI/ML IDS over network traces)

```
{
  "functionDescriptorType": "ANALYSIS",
  "description": "AI-based IDS consuming network traces and detecting attacks (DoS, DDoS, Backdoor, etc.).",
  "name": "cl-analysis-ids",
  "version": "1.0",
  "orchestrated": true,
  "sharable": false,
  "orchestratorTemplate": {
    "name": "cl-function",
    "version": "e3a34070-74db-46c5-be28-abc6076d03c1"
  },
  "inputParameters": [
    "inTopic",
    "outTopic",
    "modelId",
    "attackClasses"  // e.g., ["DoS", "DDoS", "Backdoor"]
  ],
  "requirements": {
    "cpu": 0.256,
    "memory": 536870912,
    "storage": 0
  },
  "replicas": 1
}
```

Decision Function (CACAO Workflow selection)

```
{
  "functionDescriptorType": "DECISION",
  "description": "Decision logic using CACAO playbooks to map detected attacks to mitigation plans.",
  "name": "cl-decision-cacao",
  "version": "1.0",
  "orchestrated": true,
  "sharable": false,
  "orchestratorTemplate": {
    "name": "cl-function",
    "version": "e3a34070-74db-46c5-be28-abc6076d03c1"
  },
  "inputParameters": [
    "inTopic",
    "outTopic",
    "cacaoPlaybook",        // CACAO Playbook as Object
  ],
  "requirements": {
    "cpu": 0.128,
    "memory": 264241152,
    "storage": 0
  },
  "replicas": 1
}
```

Execution Function (Remediation application)

```
{
  "functionDescriptorType": "EXECUTION",
  "description": "Execute the selected CACAO remediation plan via orchestrator and/or network actuators.",
  "name": "cl-execution-orchestrator",
  "version": "1.0",
  "orchestrated": true,
  "sharable": false,
  "orchestratorTemplate": {
    "name": "cl-function",
    "version": "e3a34070-74db-46c5-be28-abc6076d03c1"
  },
  "inputParameters": [
    "inTopic",
    "ztsoEndpoint",         // ZTSO NBI to request service/network reconfig
    "actuatorEndpoints",    // e.g., firewall API, SDN controller, WAF, etc.
    "rollbackPolicyId"      // how to rollback if mitigation fails
  ],
  "requirements": {
    "cpu": 0.128,
    "memory": 264241152,
    "storage": 0
  },
  "replicas": 1
}
```