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

Deliverable D4.1
Security Automation for 6G

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:	25/12/2024	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:	D4.1
Document Title:	Zero-touch Security Automation for 6G
Editor(s):	Pietro G. Giardina, Marco Ruta (NXW)
Authors:	Contributors and their organizations are listed below
Contractual Date of Delivery:	31/12/2024
Dissemination level:	PU[1]
Status:	Final
Version:	1.0
File Name:	ROBUST-6G D4.1_v1.0

Revision History

Revision	Date	Issued by	Description
0.1	04.12.2024	ROBUST-6G WP4	Partial draft for 1st review round
0.2	10.12.2024	ROBUST-6G WP4	1st round comments addressed
0.9	17.12.2024	ROBUST-6G WP4	2nd round of comments addressed – Ready for QA
1.0	24.12.2024	NXW	Final version ready for submission

Abstract

The main objective of WP4 is to bring security in complex multi-domain and multi-stakeholder 6G environments, by implementing a specialized ROBUST-6G Zero-Touch Security Platform. This deliverable lays the foundation for the security platform's design and development. The document identifies four key mechanisms which can be considered the 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.

Keywords

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

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

PU = Public

List of Contributors

Participant	Short Name	Contributors
Nextworks	NXW	Enrico Alberti, Giada Landi
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	Charilaos C. Zarakovitis, Chih Yang Pee, Wei Chuen Yau
EURECOM	EUR	Marios Kountouris, Ioannis Pitsiorlas
Linkopings Universitet	LIU	Nikolaos Pappas, Eunjeong Jeong

List of Reviewers

Participant	Short Name	Contributors
Universidad de Murcia	UMU	José María Jorquera Valero
University College Dublin	UCD	Bartlomiej Siniarski

Executive Summary

With the advent of 6G mobile networks, the services and functionalities offered to stakeholders are becoming more and more complex and stratified. This complexity affects the management plane as well and is further exacerbated by the integration of pervasive mechanisms for network and service automation and AI/ML algorithms by design. As a result, 6G systems are exposed to new sets of security threats never seen before.

The ambitious primary goal of ROBUST-6G WP4 is to design and implement a specialized Zero-Touch Security Platform capable of enforcing, monitoring, and maintaining the required level of security in 6G environments. The platform will be built throughout the project and the different phases of evolution reported in three deliverables of WP4. D4.1 is the first milestone in this process.

The work conducted to define the Zero-Touch Security Platform has identified four key mechanisms which can be considered the pillars of the security platform's architecture: i) security service and resource orchestration, ii) programmable pervasive monitoring, iii) threat/anomaly detection and prediction, and iv) closed-loop-based security automation. All of them are analysed in the document.

In the beginning, an initial comprehensive study of the State of the Art has been performed, with the main focus on the concept of security orchestration as reported in the literature and the OnSOAP architecture which offers a solution for modelling security systems and workflows following an ontology-driven approach. Concerning security automation, both ETSI ZSM definitions of closed-loop and NIST solutions for security automation have been investigated and a mapping between the two has been executed.

Then, an early-stage functional architecture of the Zero-Touch Security Platform has been designed as a set of Macro-Services covering the four pillars: the modules in charge of their implementation have also been identified and widely described throughout the document.

The Security Orchestrator implements the functionalities for security service orchestration, which includes the definition of security service and its lifecycle management. The modelling of the security services and systems and remediation plans are based on a custom extension of the ontology proposed by OnSOAP and will be integrated into the Security Orchestrator, whose architecture is provided as well.

The Security Resource Orchestrator implements the functionalities for security resource orchestration, by incorporating mechanisms for secure resource allocations such as the risk-averse resource management. It also interacts with different cloud platforms to provision security applications e.g., intrusion detectors.

The Programmable Monitoring Platform is devoted to data collection in different segments and layers of the 6G systems, which is crucial for anomaly detection and prediction. The main idea behind this component is to provide a completely programmable software that allows the selection of the data and data sources, capable of (near)real-time data exposure as well as time-series storage, useful for offline analysis and training of AI/ML algorithms.

The mechanism of security automation, i.e., the capability of the system to maintain the required level of security, is guaranteed by the implementation of dedicated AI-driven security closed-loops. Here AI/ML algorithms are exploited to perform anomaly detection and prediction, the intelligent part of the security automation mechanism. Algorithms and techniques for AI-driven incident mitigation have been investigated and the security closed-loop modelling itself has been also described. This mechanism ensures that the security platform remains adaptive and capable of responding dynamically to emerging threats without human intervention, when appropriate.

Security closed-loops can be indeed used for both zero-touch automation (automation with limited or even without human intervention) and/or support "Human Oversight", the concept of human-in-the-loop as per the EU AI Act. The closed-loop, in ROBUST-6G view, is based on the ETSI ZSM work consisting of four stages, Monitoring, Analysis, Decision, and Execution, completely configurable at runtime. Monitoring is covered by the Programmable Monitoring Platform while the Analysis, in charge of detecting or predicting anomalies, can be implemented either by a well-known anomaly detection software (detection) or by a custom AI/ML algorithm (detection/prediction), as mentioned.

The ROBUST-6G Zero-Touch Security Platform presented in this deliverable is still in its early stages. D4.1 mainly reports on studies, investigations of base concepts, and architecture design with the implementation of the various components set to begin in the second year of the project. The progress in continuous refinement of the platform will be reported in D4.2 and D4.3 with the release of the prototype at the conclusion of WP4.

Table of Contents

List of Tables

List of Figures

Acronyms and abbreviations

Term	Description
IT	Information Technology
AI/ML	Artificial Intelligence/Machine Learning
CL	Closed-Loop
DoS	Denial of Service
ETSI	European Telecommunications Standards Institute
ICT	Information Communication Technology
IDS	Intrusion Detection System
MANO	Management and Orchestration
ME	Managed Entity
NFV	Network function Virtualisation
NS	Network Slicing
NSF	Network Security Function

QoE	Quality of Experience
QoS	Quality of Service
RL	Reinforcement Learning
RM	Resource Management
SDN	Software Design Networking
SOAR	Security Orchestration, Automation, and Response
SOC	Security Operation Centre
NS	Network Slicing
AR	Augmented Reality
XR	Extended Reality
ZSM	Zero-touch Service Management
IR	Incident Response
VIM	Virtual Infrastructure Manager
PP	Preparation Plan
RRP	Response and Recovery Plan
Z-SO	Zero-touch Security Orchestrator
IRP	Incident Response Plan

1 Introduction

1.1 Scope and Objectives

Security and automation are two key aspects of modern mobile networks. The grooving complexity of the network, which started with the advent of 5G, in terms of functionalities offered and service supported, is reaching its maximum level in the last generation, the 6G. In 6G systems, the implementation of mechanisms for enabling automation is highly pervasive, as well as the usage of AI algorithms to support or even make decisions on behalf of human operators. The management is becoming more and more complex and stratified, covering all the network and cloud segments, up to the devices belonging to the users, offering the possibility to a variety of stakeholders to create and consume advanced network services.

The main objective of WP4 is to bring automation to security in complex multi-domain and multi-stakeholder 6G environments, by implementing specialized mechanisms capable of enforcing, monitoring, and maintaining a given level of security in a target critical environment within 6G systems. The set encompassing all these security mechanisms takes the name of ROBUST-6G Zero-Touch Security Platform and represents the final target of WP4, whose advancements will be progressively reported in the 3 deliverables of the work package (D4.1, D4.2, and D4.3).

This deliverable reports the work carried out during the first period between M03 and M12, and lays the foundation for the design and development of the ROBUST-6G Zero-Touch Security Platform. In particular, the document identifies four key mechanisms which can be considered the 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**.

1.2 Document Outline

The document is structured as follows. Section 2 investigates the State of the Art (SotA) concerning security orchestration, identifying the main functionalities and their application to secure a target system, the theoretical basis for modelling a security service and related workflows, and the SotA for security automation based on NIST and ETSI works. Section 3 is the core of the document and describes the early-stage work on the design of the four pillars of the security platform. At the beginning, an overview of the functional architecture of the Zero-Touch Security Platform is provided as an introduction then, the technical subsections describe the modules and techniques used to implement the above functionalities. In particular, Section 3 includes dedicated subsections for security orchestration, secure resource orchestration, monitoring and threat detection, incident prediction and mitigation, and closed-loops for security automation.

2 Security Orchestration and Automation in 6G

Orchestration is defined as the coordinated execution of multiple IT (Information Technology) tasks or processes that are applied across multiple systems, applications, and domains. It is a concept strictly related to automation: while automation is using software to perform tasks without human intervention, orchestration coordinates automated tasks across multiple domains into higher-order workflows so that individual tasks can work together to serve a specific function or process [RED23]. The coordination of multiple tasks is performed by an entity called Orchestrator, which can be realized in multiple manners, e.g., in the form of a framework, set of modules, by following a centralized deployment, distributed, etc. Modern mobile networks are built upon technologies such as Network Slicing (NS), Network Function Virtualization (NFV), and Software Defined Networking (SDN) that enable scalable, flexible, and programmable networking. However, the increasing number of services and network slices needing different resource requirements, and particularly new demanding applications based on emerging technologies such as Augmented Reality (AR) and Extended Reality (XR), increase the need for scalable, decentralized, automated, and efficient orchestration frameworks [VPC+22]. Previous works on 5G network orchestration that have been previously proposed in the literature suffer from multiple limitations/shortcomings: state-of-the-art solutions as ETSI NFV's MANO [NFV-MAN 001 V4.4] framework are deployed in centralized locations, representing a single point of failure; other solutions, focus their orchestration on a single layer of the architecture, resulting in a non-optimal allocation and utilization of all the available resources at different layers [LJN+23]. A group of the European Telecommunications Standards Institute (ETSI), the ETSI Zero-touch Service Management (ZSM) focuses its work on this topic and has already issued a reference architecture featuring distributed multi-layer management and orchestration [ETS002].

2.1 Security Orchestration: State of the Art

The automation of cyberattack prevention and response is accomplished by integrating different tools, defining task flows, encapsulating them at higher level logic, and developing incident response plans as a response to different events. A multi-vocal review of security orchestration defines the functionalities, components, and capabilities of a security orchestrator, extracted by the current state-of-the-art [SJ23]. First, it tries to answer the question: *"What challenges do a security orchestrator intend to solve?"* these challenges can be divided into two different categories:

- **Technical issues:** lack of interoperability among tools, lack of tools and technologies to automate response, and limitation of existing tools to provide complete and transparent services to the end user.
- **Socio-technical issues:** lack of skills and experience, incomplete frameworks and insufficient collaboration that result in a human-centric approach that relies too strongly on human experts.

A security orchestrator has a dual role: it must flatten technical barriers between tools, allowing seamless integration of them, and support experts to reduce their workload and reduce congestion and human dependence on the decision process. The functionalities of a Security Orchestrator, depicted in Figure 2-1, can be grouped into three categories:

- **Middleware/Hub**: unifies different tools, exposes an endpoint for human investigation, and shares insights between tools and experts.
- **Orchestration of security activities**: abstracts processes and tasks into high-level workflows, provides different deployments and configurations and determines the right incident response plan to be used as a response to an incident.
- **Automated response**: enables automating both repetitive and manual tasks and policy enforcement on different solutions.

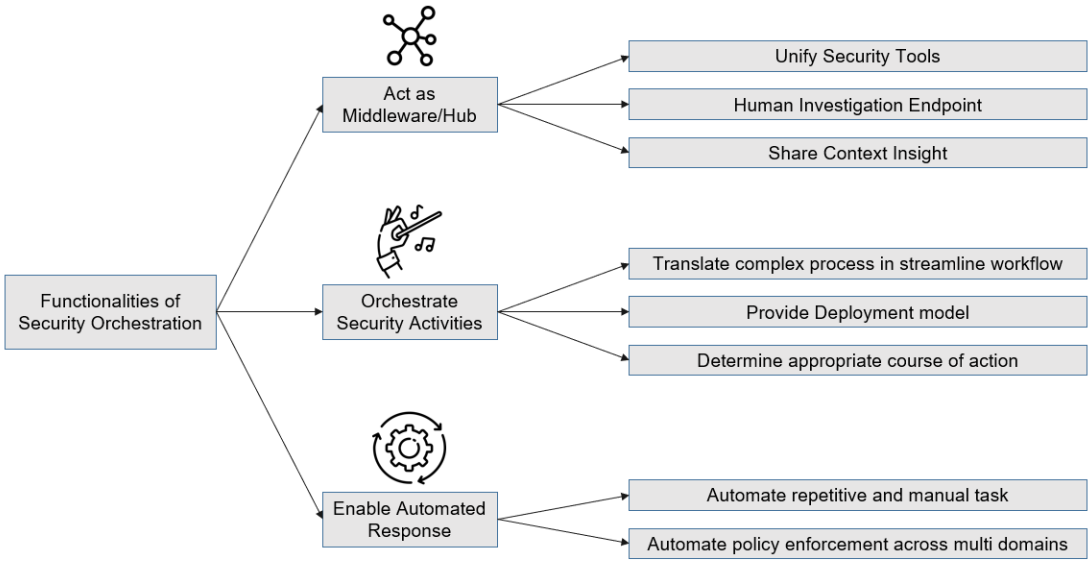

Figure 2-1 Security Orchestration functionalities [SJ23]

These three macro-functionalities are carried out by the three core components of security orchestration depicted in Figure 2-2**Error! Reference source not found.**: i) the *unification unit*, responsible for unifying all the existing security tools activities, ii) the *orchestration unit*, responsible for automating the control of deployment and configuration of all the security tasks, and iii) the *automation unit* which performs all the automated tasks deployed and managed by the orchestration unit.

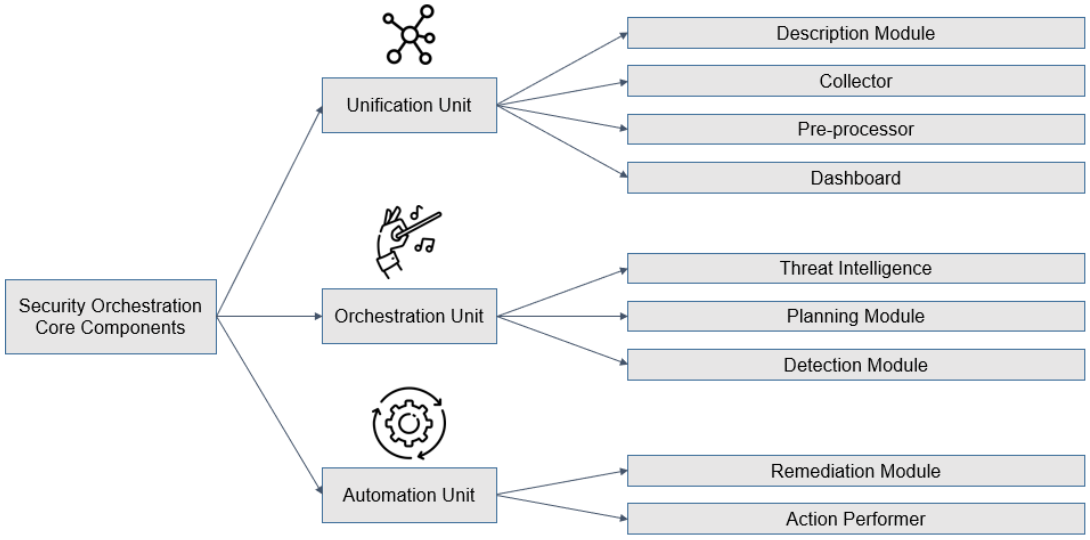

Figure 2-2 Security Orchestration Core Components

While the functionalities and core components of a security orchestrator are well-defined, there are several aspects of its design and implementation that can differentiate one orchestrator from another. The taxonomy presented in Figure 2-3 reports all the possible configurations of the execution environment, the automation strategy, the type of deployment, the mode of task, and the resource type that can be made during the design of a security orchestrator solution.

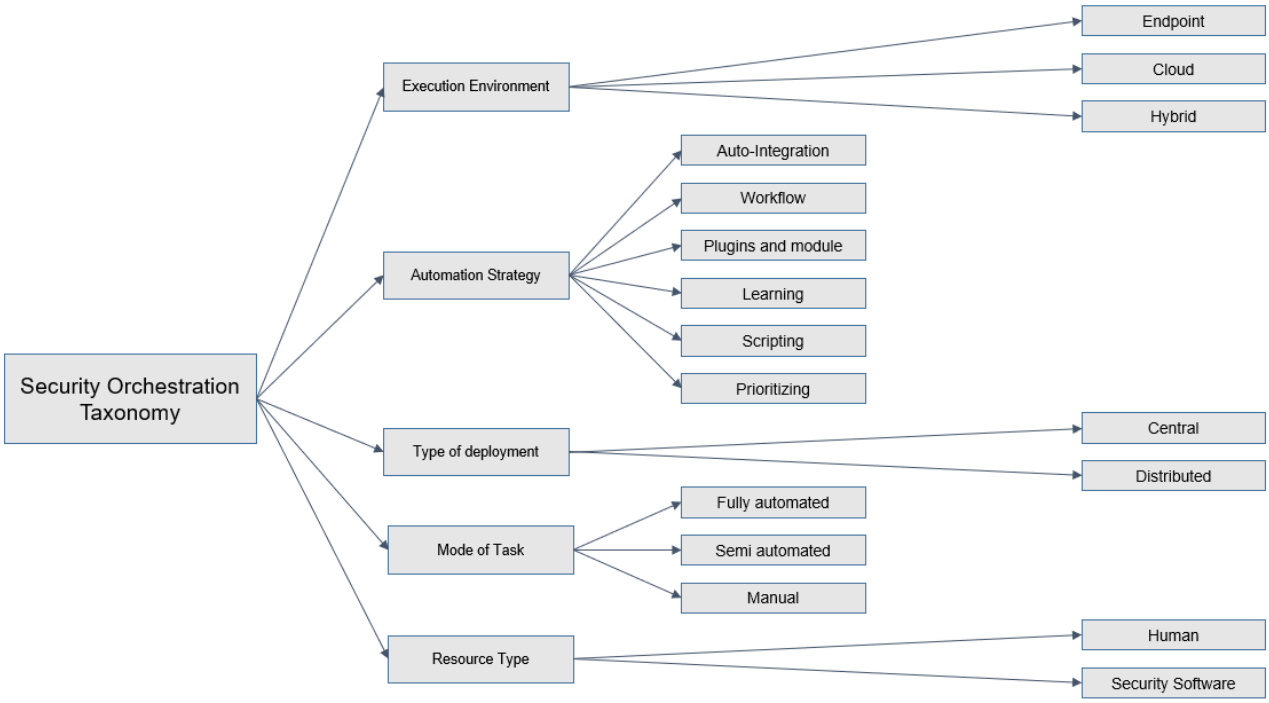

Figure 2-3: Taxonomy of Orchestration platform [SJ23]

Another important distinction, proposed in [MZZ+], is about the workflow of security orchestration, which can be classified into proactive, reactive, and predictive security orchestration.

- **Proactive security orchestration** aims to prevent potential security threats from occurring. This approach is triggered by intents that specify the requirement for a particular deployment, the orchestrator analyses these requirements and verifies if there is any conflict with the current system status. After doing so, the orchestrator deploys the most suitable security solution by analysing the available resources.
    - ***Example:*** a vertical asks for the deployment of an IDS module on its system and the security orchestrator, after analysing the status of the system, chooses and deploys the most suited IDS solution.
- **Reactive security orchestration** aims to quickly detect and respond to security threats. Expectations and requirements are defined for a particular service (as threshold, rules, etc.) and an agent is triggered by messages (i.e. that includes the type of threat and the criticality level) and sends a message to a module that analyses and creates policy-based countermeasures according to the threat.
    - ***Example:*** an IDS deployed on a system sends an alert to the security orchestrator for a detected DoS attack. The orchestrator, after analysing the alert and the current state of the system, applies a countermeasure.
- **Predictive security orchestration:** an extension of reactive orchestration that does not react to threat notifications but tries to predict them by actively analysing the state of a service and using predictive algorithms (i.e., AI/ML-based algorithms) to predict potential future threats.
    - ***Example:*** the orchestrator analyses the pattern alerts received in the past by a system IDS and, using a DL threat prediction model, predicts that a threat could arise in the future. Taking into account the current state of the system and the prediction, the security orchestrator applies pre-emptive actions on the system.

A security orchestrator can also provide both two types of orchestration, adopting a proactive strategy to harden the systems and reacting/predicting emerging threats using a reactive approach. All these kinds of orchestration require an "intelligence quota" to be carried out: while in a traditional approach, this was a strictly human-related task, carried out by an expert or based on indication given by the expert, the current research is investigating the integration of Artificial Intelligence (AI) as a mean to overcome this human-centric approach. The new generation of mobile networks, 6Gwill heavily rely on pervasive AI which, combined with deeply converged Information Communication Technology (ICT) systems that feature connectivity, and storage at the edge, will become a native trait [HUA23]. AI, along with all the other possible uses, will be used to both automate and orchestrate both operational and functional elements for 6G networks.

In the context of security orchestration, AI algorithms can be used both to automate tasks and to streamline the orchestration of automated tasks by selecting the most suited incident response plan as a response to a threat. The analysis proposed in [KA21] shows how the current state-of-the-art SOAR makes extensive use of AI/ML algorithms in the automation of tasks and support to SoC analysts: Splunk [Splu24], FireEyes [Fire24] and other companies have implemented Machine Learning Toolkits that use AI/ML models for classification/clustering tasks as anomaly detection, user behaviour analysis, malware detection, and events classification; Other companies as Simplify (now part of Google Cloud Security) [Simp24] and IBM Resilient [Res24] make use of AI/ML as support to SoC (Security Operation Centre) analyst by categorizing, prioritizing, and assigning tasks to the analyst who have worked on previous similar cases. From the current state of the art can be noticed how the adoption of AI/ML algorithms in SOAR systems is focused on the automation of tasks rather than on the orchestration and the streamlining of operations. AI/ML algorithms are trained to discover hidden patterns in data without human involvement, however, their use to orchestrate and choose the best course of action is limited by the setting, which is a particularly complex and dynamic one. The rules to detect, prevent, remediate, recover, and respond to security incidents are still manually determined or rely heavily on the logic imposed by SoC analysts. Different kinds of models such as Reinforcement Learning (RL) could be suited to model software agents that can make autonomous observations and take sequential actions without or with limited prior knowledge, receiving rewards for their behaviour. Analysing the state-of-the-art, the following gap can be easily spotted: AI/ML algorithms are widely adopted in automation aspects of SOAR. Regarding the integration of AI/ML algorithms into orchestration processes in SOAR, the highly dynamic nature of the setting has slowed this process, and current solutions still rely partially or completely on human experts.

In improving the security of a system, it is always important to consider the balance between security and the overhead of the introduced solutions on the system performances. It is always important to address security issues while trying to optimize the effort by analysing constraints, available resources, and different security methods that vary in complexity. Orchestration has a central role in this: streamlining the execution of automated tasks must be done knowing the available resources and constraints. In a decentralized scenario, resource management has the double purpose of optimizing resource usage and avoiding conflicts between enforcements. An example of security resources orchestration, proposed by [MZZ+23], offers an intent-based policy-based approach to deal with heterogeneous devices: intents are used by users to define deployment constraints that can be hard (must be met) or soft (should be considered). These constraints range from latency to computational, storage, and security requirements. To implement and validate the orchestration, they use FLUIDOS [FLU22], a decentralized OS that unifies distributed nodes, resources, and capabilities under a homogenized view. FLUIDOS is used to have a global idea about the available network, storage, and security resources and to traduce intent security function deployments. In their vision, the intents can be both used to specify proactive security measures or reactive security measures as a response to security events. Another solution proposed by [GYC+22], uses SDN/NFV for programmable and dynamic deployment of network services as Network Security Functions (NSF) addressing issues related to scalability, responsiveness, and adversary resilience. Their architecture is composed of three modules: *Network monitoring agent*, responsible for collecting statistical data to monitor network states and selecting the best routing path for monitoring and detection of attacks; *Orchestration Agent*, which analyses the information provided by network monitoring and deploys the most suitable NSF directly on agents; *Response Agent*, which is capable of taking reactive actions to defend against network abnormalities. Their target environments are docker engines where the Orchestration Agent deploys containers that enforce SNORT [Sno24] rules. In this work, network resources are monitored using an SDN controller and security functions are deployed as VNF.

2.2 Security system and workflow modelling: the OnSOAP architecture

Ontologies, as tools for representing knowledge in a systematized form, hold a special place in modern information systems in allowing the creation of standardized data structures that can unify disparate pieces of information into a coherent representation [Pos23]. The lack of interoperability and interpretability of security tools, together with current approaches to security orchestration that are human-centred, make ontologies perfect fit for attempting to streamline, automate, and make the deployment and orchestration of security systems less human-dependent. In this context, the work proposed by [IBN19], is an Ontology-driven approach for a Security Orchestration Platform (OnSOAP) that tries to automate the process of integration of security

systems. In their work, they define an ontology that provides a formal definition of the core components of security orchestration using the following elements:

- Security systems $S = \{s_1, s_2, \ldots, s_N\}$
- Incident Response Plans $IRP = \{irp_1, irp_2, \ldots, irp_N\}$
- Activities $AC = \{ac_1, ac_2, \ldots, ac_N\}$
- Task $T = \{t_1, t_2, \ldots, t_N\}$
- Functional Capabilities $FC = \{fc_1, fc_2, \ldots, fc_N\}$

A Security System S_K has some functional capabilities FC_K and an Incident Response Plan IRP_K is composed of a set of security activities AC_K, a Task T_K is an action that a security orchestrator performs to automate the execution of activities of a given IRP. The main purpose of the security orchestrator can be summarized in the following sentence:

"*Given an incident* ***x*** *and the most suitable plan* IRP_x*, the OnSOAP has to find the security systems* $\boldsymbol{S_x}$ *who have the Functional Capabilities* FC_x *that allow to perform the set of tasks* T_x *needed to perform the actions* AC_x *of the* IRP_x*.***"**

To exploit this ontology, they propose an architecture, depicted in Figure 2-4, which is composed of three layers:

- **Application Layer:** provides fine-grained information about the security systems, hosts running security processes, and provides an editor for modifying the ontology.
- **Semantic Layer:** hosts the ontology and a semantic reasoner which is capable of exploiting the ontology to find the most suited solution to a given input problem. This layer exposes a query engine to the other layers as an interface to query the ontology.
- **Orchestration Layer:** is responsible for invoking appropriate tasks for the automation of the process of the integration of several security systems based on the activities of an IRP.

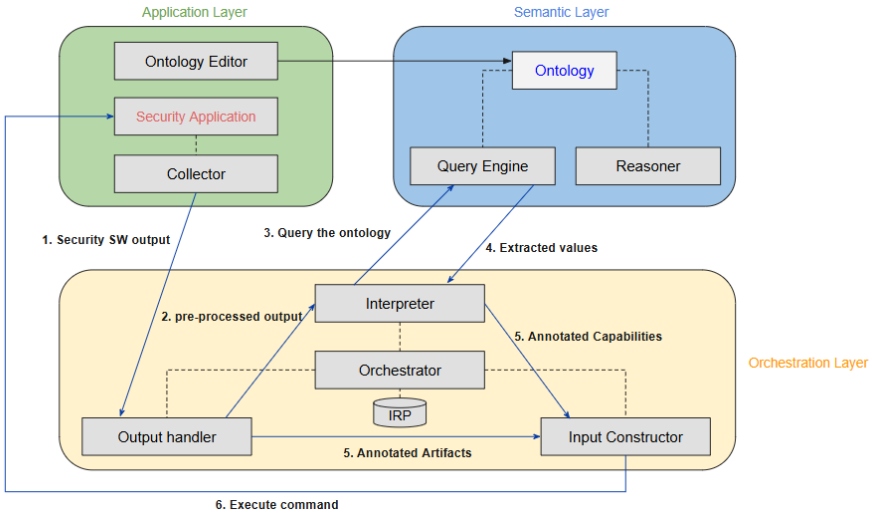

Figure 2-4: High-level architecture of OnSOAP

To better understand the workflow that starts from a notification (Security SW output) and leads to a remediation (Execute command), an example structured over the architecture represented in Figure 2-4 is reported in Figure 2-5: after receiving an alert, the output handler enriches the alert with relevant information and queries the ontology via the interpreter. The ontology reasoner, given the query, identifies the optimal IRP for the alert type and, based on the actions outlined in the plan, determines which tasks should be performed by the orchestrator and the needed functional capabilities. The orchestrator, given the needed functional capabilities, selects and orchestrates a set of security systems to fulfil the IRP specifications. The presented OnSOAP solution will be extended in Section 3.2.3 where an extended ontology will be used as the founding block of the proposed Zero-touch Security Orchestrator.

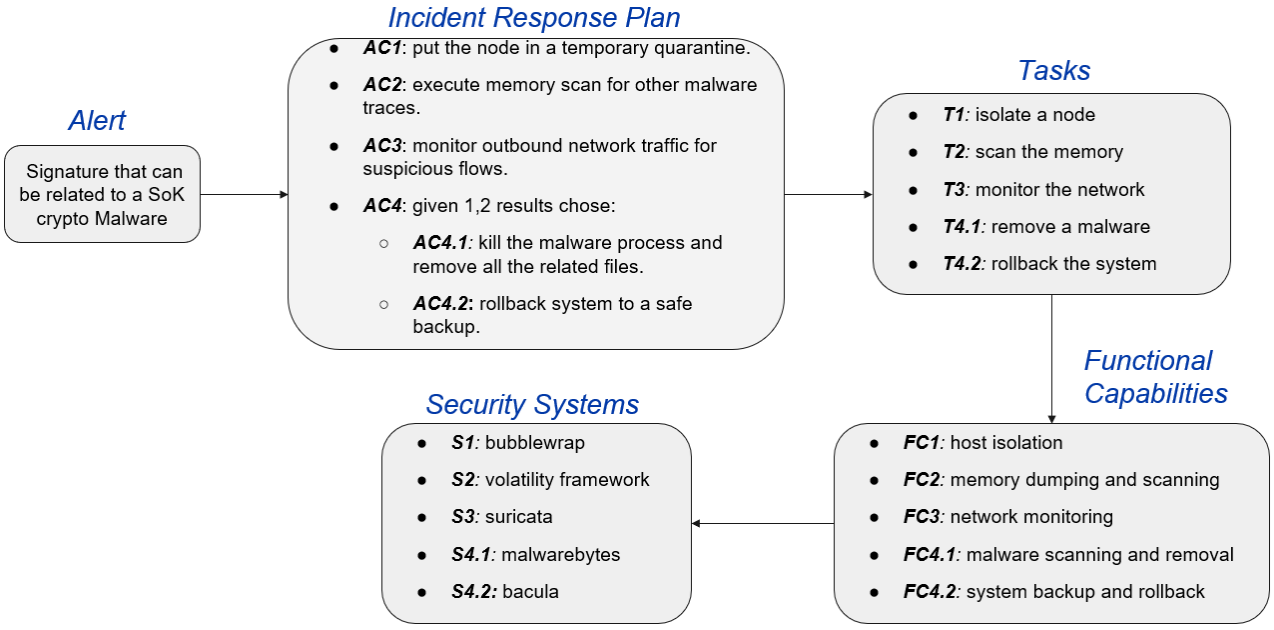

Figure 2-5: OnSOAP workflow example

2.3 Security Automation

The automation of a given process usually relies on the powerful concept of closed-loop (CL) or closed-control- loop, widely used by the industry in general. A CL is usually built of 4 stages which can assume different names for the same functions in different contests, mainly depending on which entity (e.g., SDO, Industry Association, etc.) provided the CL formalization. In ROBUST-6G, the CL follows the definition provided by ETSI ZSM (Zero-touch network & Service Management) ISG (Industry Specification Group) in ETSI ZSM 009-1 [ZSM009-1-2023]. ETSI ZSM CL defines four stages namely:

- **Monitoring**, in charge of collecting data from the target system or Managed Entity (ME, i.e., the part of the system whose control is automatized by the CL);
- **Analysis**, in charge to analyse the data collected from the ME and derive insights on its status;
- **Decision**, which decides the actions to be taken on the ME, given the output of the Analysis stage;
- **Execution**, in charge of enforcing the Decision's output on the ME, i.e., any corrective action needed to maintain the status of the ME, which represents the Goal (the set objectives) of the CL. In Figure 2-6 depicts the CL as defined by ETSI;

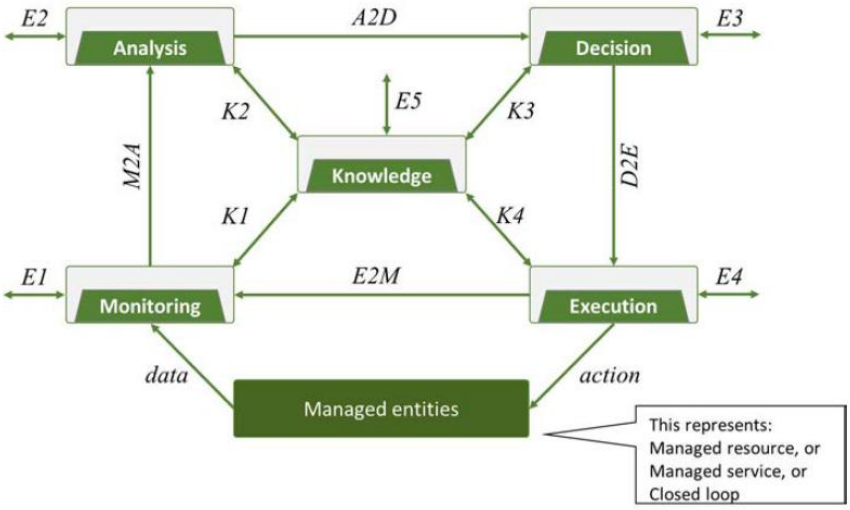

Figure 2-6: Functional view of a Closed Loop and its stages within the ZSM framework [ZSM009-1-2023]

In Figure 2-6, an additional entity is also represented x, the Knowledge, which is not a stage but an element that provides storage capabilities, used to maintain CL Goal and configurations and allow the possibility of

communication between the different stages, beyond the main loop sequence. It is important to note that in ZSM the CL is an ME with its lifecycle (creation, configuration, decommissioning, etc.) managed by a specific CL management function called CL Governance. In particular, the CL Governance offers interfaces and functionalities to create, configure, get information, start, pause, and decommission a CL usually consumed by CL creators which can be orchestrators, applications, human operators, or even other CLs or CL management functions. One of the main CL Governance interface consumers is CL Coordination, a CL management function defined by ETSI ZSM for the coordination of multiple CLs. For the level of automation required in 6G systems, multiple CLs may coexist and insist on the same ME (e.g., the same set of resources), take conflicting decisions and move the ME (or the system in general) to inconsistent or unstable states. The objective of the CL Coordination is to avoid/mitigate conflicts by properly coordinating the different CLs e.g., based on the priority of a specific CL. To achieve this, the CL Coordination implements several dedicated functionalities that may also be AI/ML-based. Moreover, this definition contains three aspects of the CL that the automation should respect the followings regarding security:

1. Enable automation of management tasks, continuous optimization, and adaptation of the behaviour of the managed entities in response to changes in internal or external conditions.
2. Closed-loop operation can happen at the management domain level, at the end-to-end service management domain level, and can span across multiple management domains.
3. Support variable degrees of automated decision-making and human oversight with fully autonomous management as the final stage. Closed-loop operation is governed and constrained by operational policies which define the operational conditions under which autonomous operation is allowed. These conditions include levels of human oversight, reporting, and conditions for escalation, delegation and coordination, etc.

In the specific case of security, automation is defined as the use of software to automatically detect, prevent, investigate, and remediate cyberattacks to reduce the number and severity of IT security incidents while minimizing the need for human intervention [RED-23-2]. The automation involves using software in a programmatic way, following a well-defined sequence of coordinated steps that address the various phases of cyberattacks. While the automation of a single task can be easy, the effectiveness of the automated task depends on understating the whole context of the task with the preceding and subsequent steps required in the response process. In this regard, Incident Response (IR) is a key aspect in establishing the processes and technologies used to detect and respond to cyberattacks in a structured manner. The role of IR in security automation is crucial: by defining the needed steps and their relations, IR ensures that the automated tasks are applied effectively and consistently. The most common framework for IR is the NIST 800-61 Computer Security Incident Handling Guide [800-61-r2] which defines IR as a 4-step process, depicted in Figure 2-7, composed of the following steps:

1. **Preparation**: preliminary step in which defences against potential incidents and threats are considered based on the impact they could have on an organization, the likelihood of them occurring, and the criticality of the asset affected.
2. **Detection & Analysis**: the step that includes alerts and notification generation resulting from periodic and/or continuous monitoring.
3. **Containment Eradication & Recovery**: step of reaction to an incident to contain the problem, eliminate it and attempt to restore the system to the state prior to the incident.
4. **Post-Incident Activity**: the post-recovery step that focuses on “lesson learned” and tries to enforce corrective measures to prevent the incident from reoccurring.

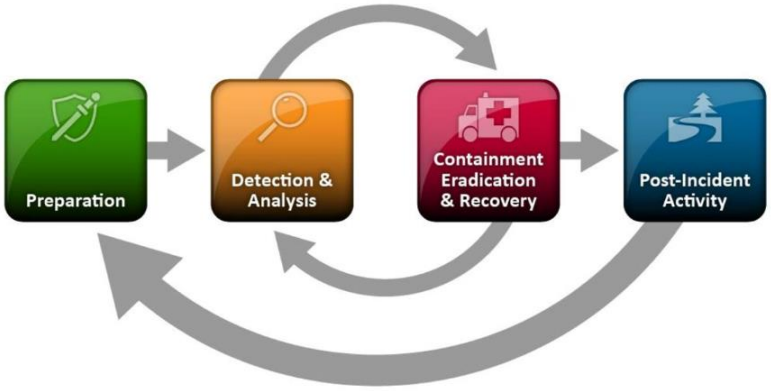

Figure 2-7: Incident Response Lifecycle [800-61-r2]

However, the four-step model no longer reflects the current state of incident response: today, incidents occur frequently and cause far more damage, and recovering from them often takes weeks or months due to their breadth, complexity, and dynamic nature. NIST proposed a revised model in [800-61-r3] in which the new proposed IR model is based on the six functions defined in [CSF2.0] that organize cybersecurity outcomes at their highest level:

- **Govern** (GV): the organization's cybersecurity risk management strategy, expectations, and policy are established, communicated, and monitored.
- **Identify** (ID): the organization's current cybersecurity risks are understood.
- **Protect** (PR): safeguards to manage the organization's cybersecurity risks are used.
- **Detect** (DE): possible cybersecurity attacks and compromises are found and analysed.
- **Respond** (RS): actions regarding a detected cybersecurity incident are taken.
- **Recover** (RC): assets and operations affected by a cybersecurity incident are restored.

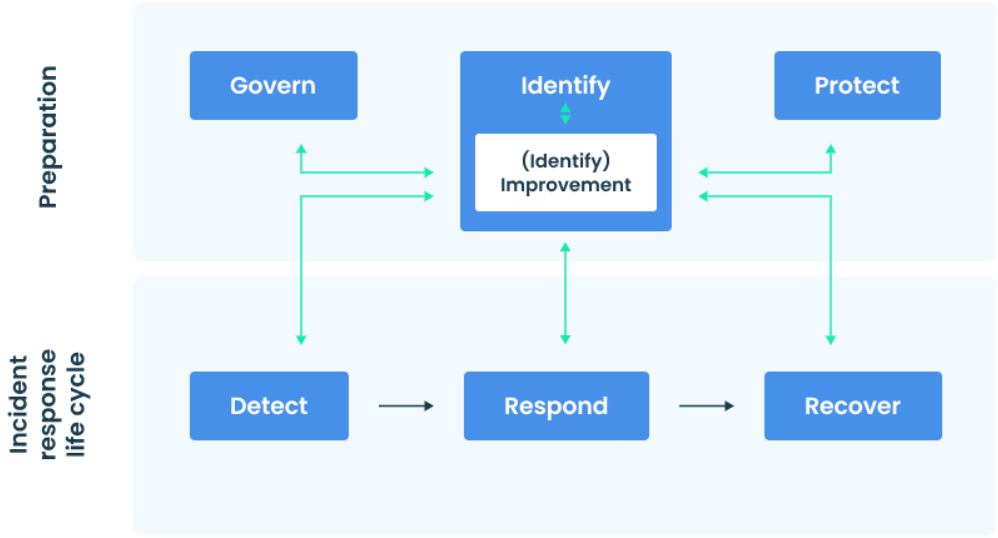

Figure 2-8: Incident Response Model [800-61-r3]

Figure 2-8 represents the new IR model based on the six CSF 2.0 cybersecurity functions. The model is divided into two different sections:

- **Preparation:** reflects activities of govern, identify, and protect that do not fall into the incident response lifecycle, being broader cybersecurity risk management activities that support but are not central to incident response.
- **Incident Response Lifecycle**: composed of detect, respond, and recover representing all the steps that go from the detection of a threat to the restoration of a safe state.

However, all the steps are interconnected to the Identify (Improvement) step outlining the need to learn from the outcomes of all the steps to improve the knowledge about a threat and continuously improve all the activities in incident response. In the following list, the definition of Proactive, Reactive, and Predictive security orchestration given in Section 2.1 is mapped with the [800-61-r3] IR Model steps:

- **Proactive Security Orchestration:** corresponds to the Preparation phase, starting from the risk assessment of a system and ending with the identification of security solutions needed to protect it.
- **Reactive Security Orchestration:** corresponds to the IR Lifecycle part, starting from the detection of a threat that is not covered by the solutions deployed during the proactive operations, and ending with the deployment of new security solutions or the calibration of pre-existing solutions.
- **Predictive Security Orchestration:** is the same as Reactive Security Orchestration but features a "Predict" block instead of the "Detect" one.

In this regard, the execution of proactive and reactive/predictive security orchestration activities will constantly enrich the Identify (Improvements) block allowing the execution of orchestration activities that will be always more fine-tuned and improved. To better understand this concept, it is useful to map the Incident Response Model proposed in [800-61-r2] with the five-stage definition of closed-loop given by ETSI-ZSM. The mapping, depicted in Figure 2-9, is based on the following considerations:

- **Monitoring – Analysis**: these stages can be mapped to the IR steps of detection considering also the monitoring (observe) phase that is needed to perform detection.
- **Decide**: this stage cannot be clearly mapped to one or more IR steps but can be considered as the connection between Detect and Respond. The decision stages take the detection of a threat as an input and select the activities that need to be performed based on the available IRPs.
- **Execute**: this stage can be mapped with the respond and recover steps of the IRP, representing all the actions that are needed to enforce the selected IRP.
- **Knowledge**: this stage, which represents the knowledge shared between all the stages in the ETSI ZSM representation, perfectly fits with the Identify (Improvements) step in being the common knowledge base that allows fine-tuning of next closed-loops executions.

Is important to outline how the stages of Govern and Protect, which are preparation and proactive security orchestration related, are not mapped with the ETSI ZSM concept of CL that is used in this context to describe reactive and predictive security orchestration.

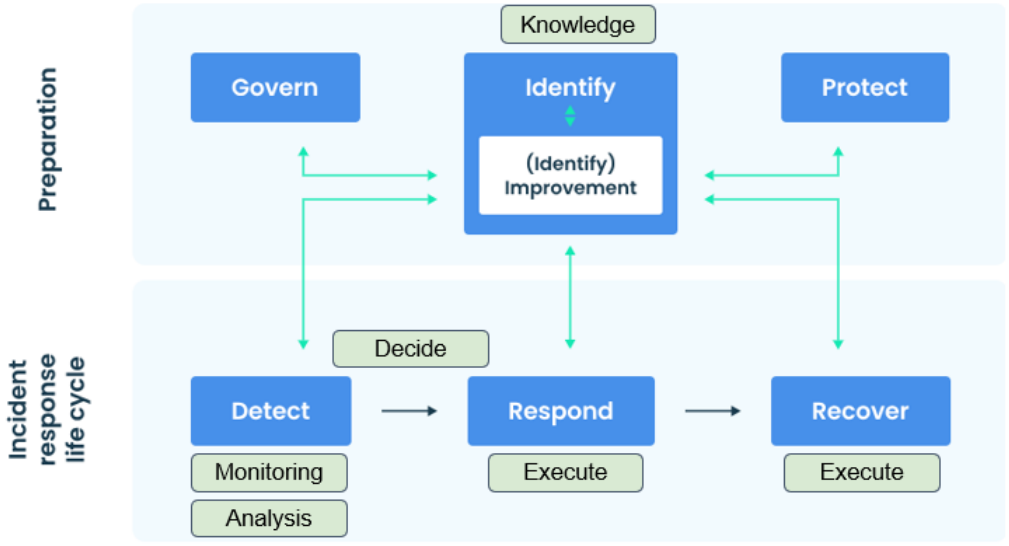

Figure 2-9: NIST IR Model and ETSI ZSM CL Model mapping

ROBUST-6G aims at designing, implementing, and integrating CL-based security automation mechanisms. Early-stage implementation details for this approach are presented in Section 3.6.

3 ROBUST-6G Zero-Touch Security Platform

This section describes the early-stage work on the design and implementation of the key mechanisms that the Zero-Touch Security Platform should encompass: i) security service and resource orchestration, ii) programmable pervasive monitoring, iii) threat/anomaly detection, and prediction iv) closed-loop-based security automation.

An overview of the functional architecture, already discussed in Deliverable D2.2 [R6G24-D22], is provided as an introduction for the technical subsections which detail the work done so far to define the components in charge of implementing all the targeted functions.

3.1 Architecture Overview

One of the main objectives of ROBUST-6G is to provide a dedicated platform for the management of security in 6G environment, capable on the one hand to understand, accept, and address security requirements from 6G stakeholders and, on the other hand, maintain the agreed level of security at runtime, providing specific mechanism for the automatic detection, prediction, and mitigation of threats and anomalies (Zero-touch security automation). It is important to note that the zero-touch automation, i.e., automation without any human intervention cannot be always applicable (especially when automatic decisions are based on AI inference), although remains one of the main target achievements of ROBUST-6G. The zero-touch automation aspect will be clarified in Section 3.6.

In Figure 3-1, the functional architecture of ROBUST-6G Zero-touch security platform is shown. The macro-services of the functional architecture of ROBUST-6G Zero-touch security platform have been already discussed in Deliverable D2.2. In this section and in the rest of the document, the main focus is on a subset of macro-services, highlighted in red in Figure 3-1 and on the components which implement these services.

The core element in the ROBUST-6G Zero-touch Security Platform is the Security Management & Orchestration (M&O) stack which encompasses the Security Service Orchestration (S-SO), the Security Resource Orchestration (S-RO), the Security Closed-Loop management functions (S-CLMgmt), and Network Security Management (NSMgmt).

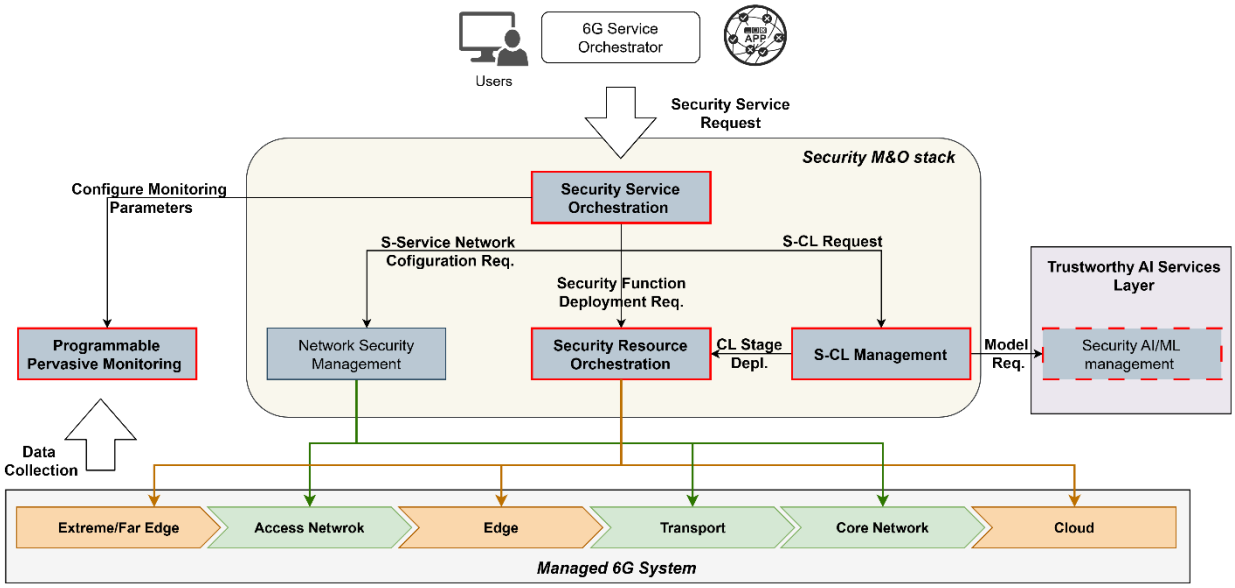

Figure 3-1: ROBUST-6G Zero-touch security platform functional architecture: in red functions discussed in this document

The functionalities of security service orchestration, represented by the S-SO are implemented by the Security Orchestrator, discussed in Section 3.2, which is in charge of the management and orchestration of the security services to be provisioned on the 6G system.

The Security Resource Orchestrator is the component encompassing the S-RO functionalities and is described in Section 3.2: it would include a mechanism for the deployment of security application, part of the Security Service, and algorithms for smart and secure resource allocation for generic 6G vertical services.

The S-CLMgmt, in charge of configuration, lifecycle management, and coordination of multiple CLs, is realized by implementing the CL Governance and the CL Coordination functions (see Section 2.3) as detailed in Section 3.6, where the usage of both functions is described along with the implementation of the CL-based automation for security purposes. In this case, a single macro-service is realized by two different modules. It is worth highlighting that the relation between a macro-service and a SW module implementing it and its functionalities, is not necessarily "1:1". Specifically, a module could implement multiple macro-services and a macro-service could be implemented by multiple modules. For example, all the functionalities provided by the macro-services in the Security M&O are focused on the orchestration of the security. These services can be integrated into a single Security Orchestrator capable of orchestrating service, network, and resource security, along with its automatic preservation in line with the security requirements requested. Hybrid modes are also possible, where the Security Orchestrator encompasses only a subset of the macro-services considered e.g., security service, resource, and network while the automation (CL Management) is provided by a dedicated component. The way to proceed is a SW design and implementation choice which is still under discussion for ROBUST-6G.

Another important point related to the Security M&O is that this document does not discuss the functionalities related to Network Security Management. They are mainly related to the enforcement of specific network configurations in the different network segments. In this project, the focus is mainly on the PHY layer (radio part) and the interaction and integration of PHY layer security mechanisms in the Zero-Touch Security Platform (whose design is not mature enough) will be reported in D4.2. It should also be considered that threat and anomaly detection is always possible by combining monitoring data and specific network traffic analysis applications without a need for specific network configuration.

Outside the orchestration group, Pervasive Programmable Monitoring offers functionalities for collecting monitoring parameters of interest for security, at any level (infrastructure, network, service) and any segment (Edge/Far edge, access, transport, core, cloud). This macro-service is realized through a specific Programmable Monitoring Platform (PMP), described in Section 3.4

Finally, a Security AI/ML management macro-service provides functionalities for the management of AI/ML algorithms for threat detection and prediction. The reason why this service is shown with a dashed board in the architecture in Figure 3-1 is due to the fact that WP4 mainly focuses on the AI/ML models, rather than how they are managed. Since the design and implementation of a dedicated platform/framework for AI/ML operations e.g., training, dataset selection, model selection, etc. are addressed in the Trustworthy AI Services Layer (see Deliverable D2.2) and are out of the scope of WP4. Monitoring, data collection and the implementation of AI/ML algorithms are crucial for implementing AI-driven Zero-touch security automation, as described in Section 3.6

3.1.1 Additional Considerations on Functional Architecture

An important point to highlight is that the Macro-Services presented so far build upon existing elements in 6G M&O, with significant specialization and enhancement to address advanced security requirements. This implies that the enhancement of a 6G system towards security management can be performed by re-using functionalities already part of the existing M&O infrastructure. This section discusses the two extreme integration cases where i) all the security Macro-Services are incorporated in the 6G system as a separate element and its opposite, whereas ii) the majority of the Macro-Service are shared for M&O of both Vertical and Security Services. In between multiple cases of partial Macro-Service sharing are possible. Case (i) is shown in Figure 3-2. The assumption is that the capabilities and functionalities of the 6G system are exposed by specific interfaces, logically grouped in the 6G Exposure, beyond which, the 6G Orchestrator takes care of requests for services from Verticals. The 6G security capabilities are similarly exposed by the Security Capability Exposure and are not directly consumable by Verticals and end users of 6G in general while super users, e.g., SysAdmins, can have direct access to them. In this case, the core set of Macro-Services building the Security M&O encompasses S-SO, S-RO, NSMgmg, and S-CLMgmg. When a Vertical performs a request for a service through the 6G exposure, 6G Orchestrator captures that request and tries to fulfil it. When the vertical service specifies security constraints, the 6G Orchestrator requests the S-SO to provision a specific

security service to fulfil them: such a process is transparent for the vertical as completely hidden by the 6G Orchestrator (SECaaS).

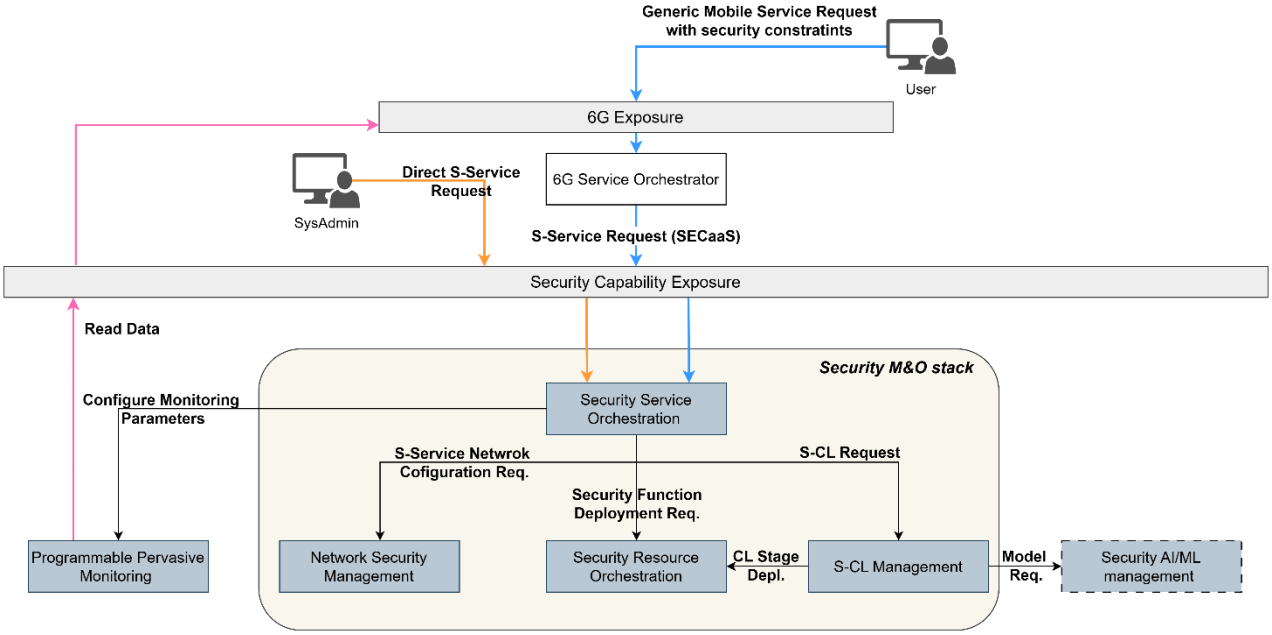

Figure 3-2: Zero-touch security platform integration: all Macro-Services dedicated to security purposes

The monitoring data collected by the PPM can be consumed by the vertical through the 6G exposure while SysAdmins can access it from the Security Capability Exposure.

Case (ii) is shown in Figure 3-3 all the Macro-Services but the S-SO are shared with non-security ones, already belonging to the infrastructure of a Telco Operator. In this case, the Security M&O stack collapses with the S-SO, while the other Macro-Services expose their own functionalities through the two types of exposure: 6G Exposure for the non-security RO, Network management, PPM, CL Management and Security Capability Exposure for the security ones. This type of integration can be seen as follows: a dedicated S-SO integrated into a 6G environment where the common Macro-Services have been extended to provide security functionalities.

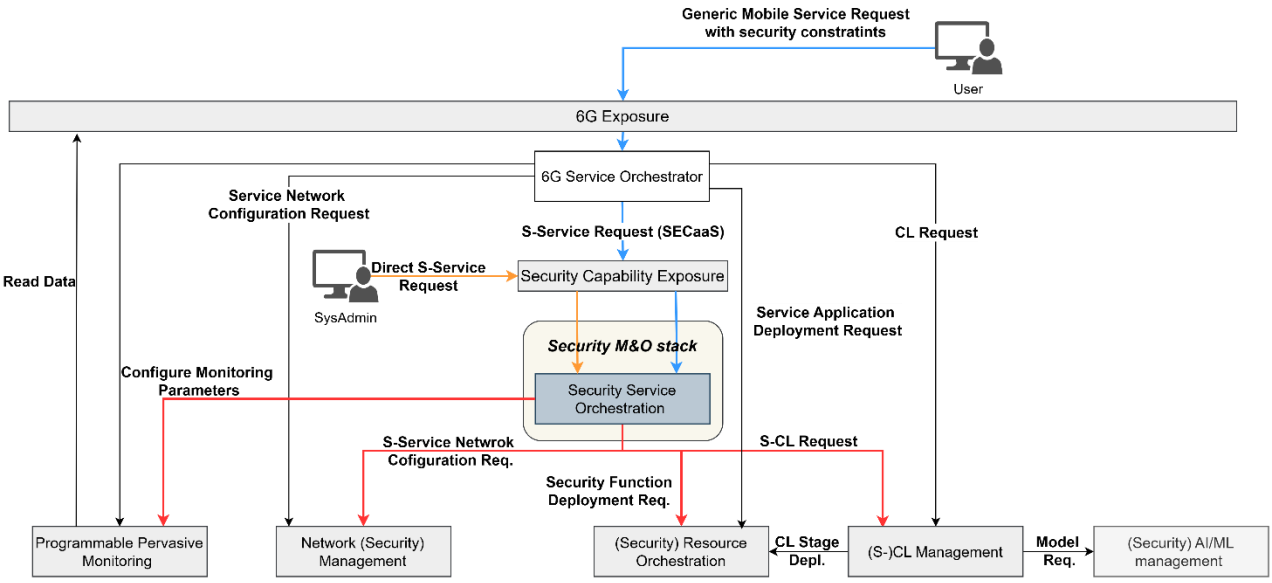

Figure 3-3: Zero-touch security platform integration: all security Macro-Services shared with generic ones

3.2 Security Orchestrator

In this section, we will explore the scope of the security orchestrator, emphasizing how it integrates with other layers of the 6G architecture to ensure the zero-touch management layer's security measures. We will examine its functional architecture, detailing the critical security policies that underpin operations and the definition of ontologies that facilitate the effective selection and inference of security concepts within the system. The discussion will then shift to the role of these ontologies in supporting the orchestrator and shaping security policies, highlighting their significance in ensuring coherent and adaptable security measures. Lastly, we will analyse the processes of security closed loops for proactive and reactive orchestration, which are essential for managing security policies and maintaining an optimal security posture in the face of evolving threats. Together, these sections will provide a comprehensive understanding of the security orchestrator's contributions to the ROBUST 6G architecture.

3.2.1 The scope of the security orchestrator in ROBUST 6G

In the envisaged architecture of ROBUST-6G these requests are addressed to a Zero Touch Security Management Layer and managed by a security orchestrator inside this layer. The security requirements may eventually go through a wide variety of interfaces before reaching the security orchestrator in this layer, this is why we will call the last interface that collects the security requirements for the security orchestrator as the "northbound interface". Thus, the security orchestrator's northern interface is available for consumers who need to address security services requirements towards a service provider. Once both parties agree on the service they wish to achieve, the orchestrator's role is to decompose and translate the requirements to a combination of requests towards orchestrators, or VIMs (Virtual Infrastructure Manager), specialized for the targeted environments. For example, these requests can contain the expression of the security requirements of a consumer for a core network environment, a cloud environment, or an edge environment. In the same way as the northern interface, this combination of requests may eventually go through a wide variety of interfaces before reaching the targeted VIM. We will call the "southbound interface" the first interface responsible for emitting the combination of requests towards the underlying VIMs.

A similar approach was developed by [NFV018-2024] where users can address their own tenant's requirements in terms of VNFs (*Virtualized Network Functions)* to high-level orchestrators which transfer their needs to one or more VIMs.

In our study, the goal of the security orchestrator is to ensure that the agreed conditions are met in terms of security requirements and agreements between a vertical and a service provider. The security orchestrator must also ensure that they are maintained throughout time and in an end-to-end way, in all the execution environments, by gathering information from the underlying orchestrators. It means that if, when monitoring certain conditions, the analysis detects the presence of a threat to the consumer's services, a specific action should be scheduled to resolve the issue in all the environments concerned by the threat. The actions to be computed and taken by the security orchestrator are policy-based. A *security policy* is the product of the agreement between the service consumer (or the vertical), who wants to deploy its service with security, and a service provider, who owns a catalogue of security services to implement the policy. The security policy is applied over a topology of services, which abstracts the environments where they are deployed. It is regarding this security policy that a security orchestrator observes the state of the services, and alerts then remediates to threats or modifications, when a violation of the policy is detected.

In addition to this security policy maintenance, it is also essential to maintain the minimum acceptable level of security for the vertical's service, which is determined by the security posture of the service provider's organization. This level is deduced from the identified threats for a service, regarding the risk management process from the organization of the service provider. We will define it as "*security posture*", the set of security controls that a service provider must implement to fulfil this identified security level against the service's threats. In other words, the security policy defines the security requirements of a vertical while the security posture defines the security controls that the security provider must implement against wider threats. Both must be implemented by the security orchestrator.

The maintenance of the security policy and the security posture are implemented by security closed loops which monitor, detect, analyse, and respond to security events during the lifecycle of the services to:

- Remediate to security policy violations
- Remediate to unpredictable incidents of the security posture

These two closed loops of the security orchestration aim to implement both the need to satisfy security requirements from a vertical and the need to respond to unpredictable security events according to a security posture.

Additionally, as mentioned in Section 2.1, the security orchestrator has a role in the implementation of proactive orchestration and reactive orchestration. It means that for each of the closed loops, the security orchestrator must ensure the governance of the security but also the detection and reaction plans for given security policies and postures of the system it orchestrates. The following matrix in Table 3-1. describes the relations between the proactive or the reactive security orchestration and the closed loops to maintain the security policies or the security posture.

Table 3-1 Relation matrix between the concepts of closed-loops and orchestration

	Proactive security orchestration	**Reactive security orchestration**
Security policy closed-loop	Implements the security services for the security governance of a vertical's service in every orchestration domain.	Implements the security services of detections and remediation against violations of the vertical's security services policy in every orchestration domain.
Security posture closed-loop	Implements security services for the security governance of a service provider's threat assessment in every orchestration domain.	Implements the security services of detections and remediation against threats to the service provider's assets in every orchestration domain.

From another perspective, the analysis and decision step of the closed loop can consider using AI/ML models to achieve better performance in defining the action to be pursued. More precisely, AI/ML models can be used as services to improve threat detection and policy violations, security alert correlations, context enrichment of security incidents, and remediation accuracy or optimization.

Finally, the security orchestrator will take advantage of a resource management entity to find the best allocation or action plan to execute to mitigate a threat.

3.2.2 Zero-touch Security Orchestrator Functional Architecture

The proposed Zero-touch Security Orchestrator (Z-SO) Functional Architecture, depicted in Figure 3-4, shows the internal functionalities of the Z-SO together with the external modules that it consumes and uses for security applications orchestration. The security orchestrator is composed of the following functional blocks:

- **Security Requirements Translation**: this is the functionality which collects the security requirements and the service descriptions coming from a vertical or a System Administrator. It translates the security requirements and the service descriptions e into an interoperable format which can be ingested by other components of the security orchestrator.
- **Alert Translation:** this is the functionality that collects alerts that come from multiple types of alerting components, Trustworthy AI Services Layer, or Data Management Layer to translate them into an interoperable format that can be ingested by other components of the security orchestrator.
- **Policy Management**: this is the functionality of the security orchestrator that collects the security requirements for a given service and generates a security policy (Preparation Plan PP) that describes the security controls and the remediations to maintain it against threats and violations. It also manages different security policies submitted to the orchestrator for traceability and accountability. Finally, it ingests alerts and compares them to establish security policies to trigger the appropriate remediations (Response and Recovery Plan RRP).
- **Ontology**: this is the "reasoning" component of the Z-SO which supports policy enforcement. It is queried by the Policy Management module and, given a PP or an RRP, it will reason over the ontology entities to select which are the most suitable security application and their configurations. An in-depth of the ontology and the ontology entities is given in Section 3.2.3.

- **Catalogue**: this is an internal database that stores information about target environments and available security applications. This database stores the ontology entities and is used as a knowledge base by the ontology to reason about the available applications and target environments resources/policies.
- **Semantic Translation**: this has the functionality of translating the ontology output, which is a high-level and textual abstraction of the policy enforcement plan, into a set of commands/indications that can be executed by the resource orchestrator and additional domain-specific orchestrators. It also translates high-level abstractions of remediation plans into a set of commands/indications towards closed-loops governance components.

Some of the external components of the security orchestrator were already defined in Section 3.1, in the following is discussed their role and interaction with the Z-SO:

- **Trustworthy and Sustainable AI Services Layer**: This exposes a wide variety of trustworthy AI/ML functionalities that can be consumed by the Z-SO through the Northbound Interface. These functionalities include ML models for improving alert and security requirements management, ML models used as deployable security applications, and ML models used upon the ontology for reasoning the most appropriate way of enforcing a policy.
- **Data Management Platform:** This is a data management layer used by the Z-SO through the Northbound Interface as an exposure interface to provide an endpoint for human investigation.
- **Resource Orchestrator:** This consumes the output of the Z-SO through the Southbound and deploys/manages the needed resources across different target environments.
- **Programmable Monitoring Platform:** This is configured through the Southbound interface by the Z-SO during the execution of proactive security orchestration to monitor the security applications output.

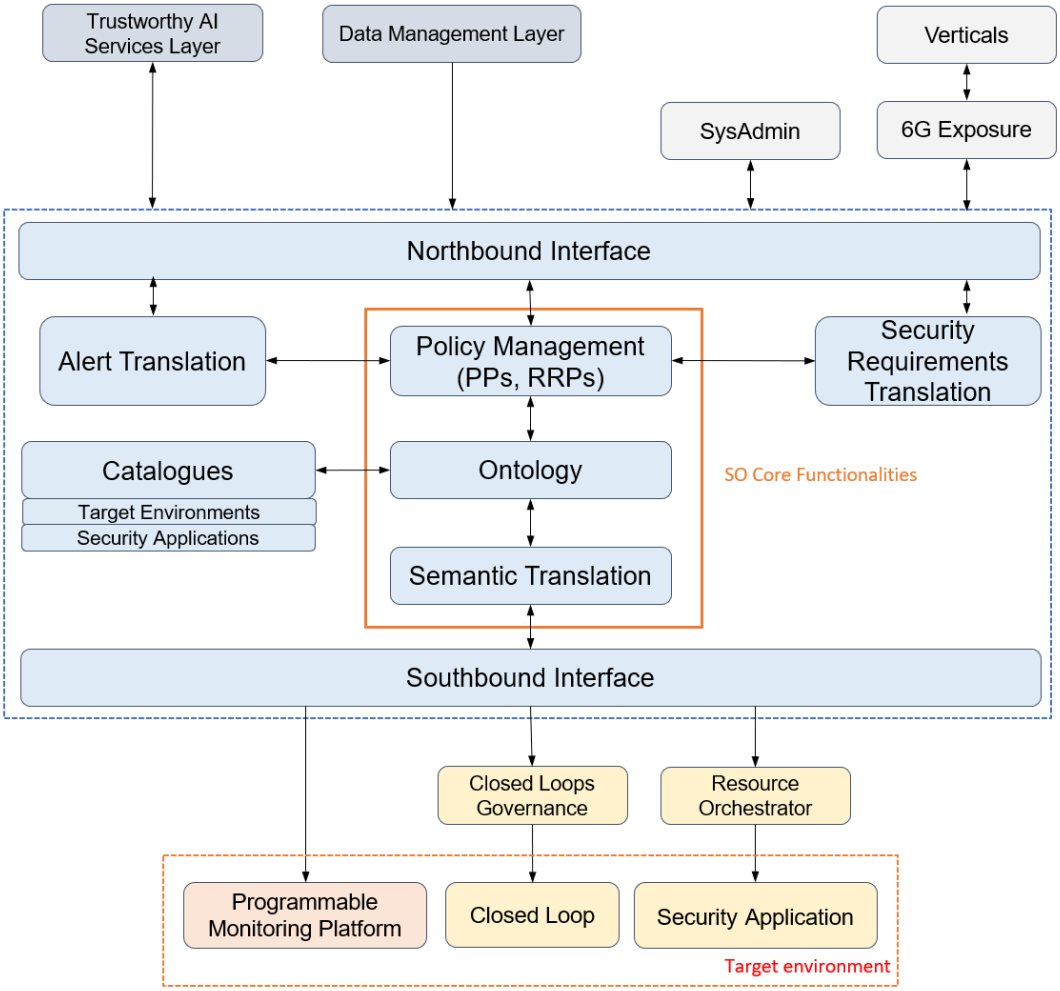

Figure 3-4: Zero-touch Security Orchestrator Functional Architecture

3.2.3 Ontology for Security Orchestration

Based on the OnSOAP architecture and ontology reported in Section 2.2, an Ontology-based approach for Zero-touch Security Orchestration in the multi-domain and dynamic scenario of 6G networks is proposed. The OnSOAP ontology is extended to implement the following characteristics:

- **Multi-domain**: the security orchestrator is capable of managing several cloud, edge, and extreme-edge target environments.
- **Policy-based**: every target environment and target environment owner can define policies about the target environment availability, data privacy, reaction, and recovery time, and the desired security level.
- **Resource management**: the deployment of security systems takes into consideration also the non-functional requirements of the software and the available resources of the target environment.
- **Proactive, Reactive, and Predictive orchestration**: the security orchestrator will not only react to threats but it will also predict future threats and allow user-transparent deployment of security services.

Moreover, the proposed approach adheres to the Incident Response Lifecycle model-based on [800-61-r3]: what was presented as IRP in the OnSOAP ontology is now split into the two different concepts of Response and Recovery Plan (RRP) and Preparation Plan (PP) which are focused respectively on the Respond and Recover steps of the Incident Response Lifecycle and in the part of the Preparation. During all these phases information about the incident is collected and the new knowledge resulting from the incident is used to identify improvements that need to be deployed. The extended ontology is presented below:

- Security Applications $S = \{s_1, s_2, \ldots, s_N\}$
- Response and Recovery Plan $RRP = \{rrp_1, rrp_2, \ldots, rrp_N\}$

- Preparation Plan = {pp_1, pp_2, …, pp_N}
- Security Actions AC = {ac_1, ac_2, …, ac_N}
- Tasks T = {t_1, t_2, …, t_N}
- Functional Capabilities FC = {fc_1, fc_2, …, fc_N}
- Target Environments TE = {te_1, te_2, …, te_N}
- Target Environment Resources TER = {ter_1, ter_2, …, ter_N}
- Service Requirements SR = {sr_1, sr_2, …, sr_N}
- Target Environment Policies P = {tep_1, tep_2, …, tep_N}

The main purpose of the security orchestrator may now be summarized in the following two sentences describing its proactive, reactive, and predictive security orchestration capabilities:

"Given the request of the security service x over the Target Environment TE_x the security orchestrator has to identify the most suited PP_x and security applications S_x who have the Functional Capabilities FC_x that allow performing the set of tasks T_x needed to perform the actions AC_x of PP_x over the Target Environment TE_x such that the needed resources of the service SR_x respects the resources of the Target Environment TER_x and the policies of the Target Environment TEP_x."

"Given the incident or the prediction of the threat x over the Target Environment TE_x and the most suitable Response and Recovery Plan RRP_x for the threat x, the security orchestrator has to find the security applications S_x who have the Functional Capabilities FC_x that allow performing the set of tasks T_x needed to perform the actions AC_x of RRP_x over the Target Environment TE_x such that the needed resources of the service SR_x respects the resources of the Target Environment TER_x and the policies of the Target Environment TEP_x."

To better understand the main differences from the OnSOAP remediation workflow the same example as reported in the section 2.2, addressing target environments and their policies and capabilities is described in Figure 3-5: Extended ontology workflow example

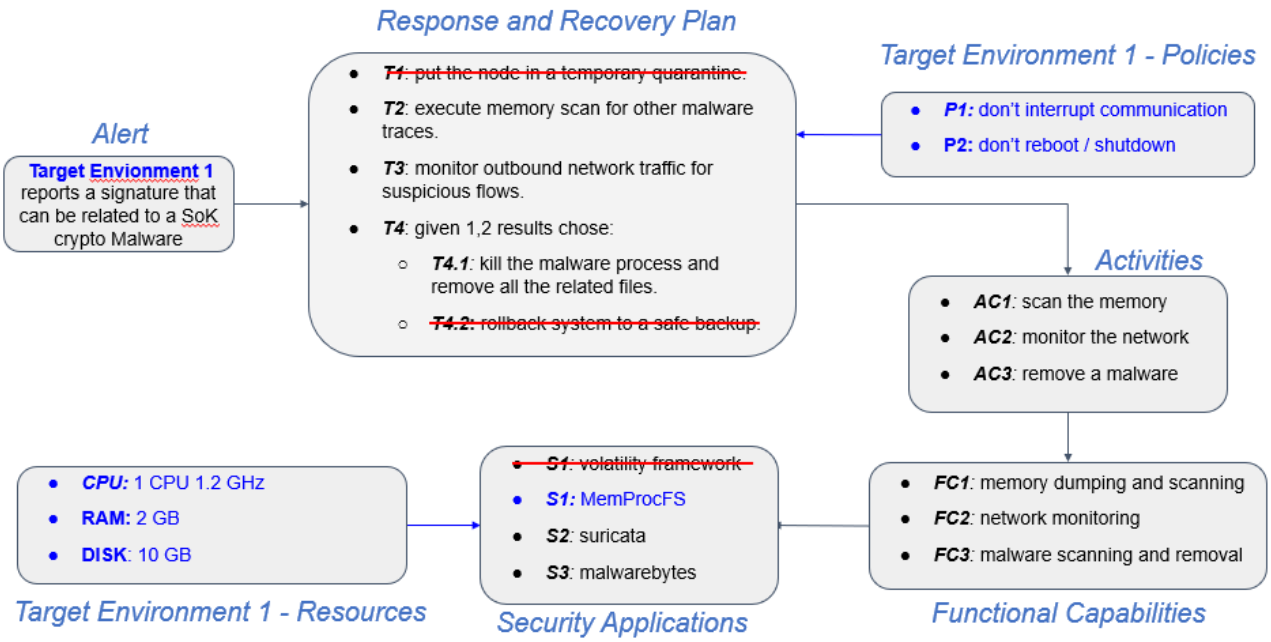

Figure 3-5: Extended ontology workflow example

The major outcomes of the extended ontology are based on the policies of the target environment. Some of the tasks are not taken into consideration based on the target environment resources constraints and the available tools, the most suited tools are selected. A high-level representation of part of the proposed ontology is represented in Figure 3-5: Extended ontology workflow example and Figure 3-7.

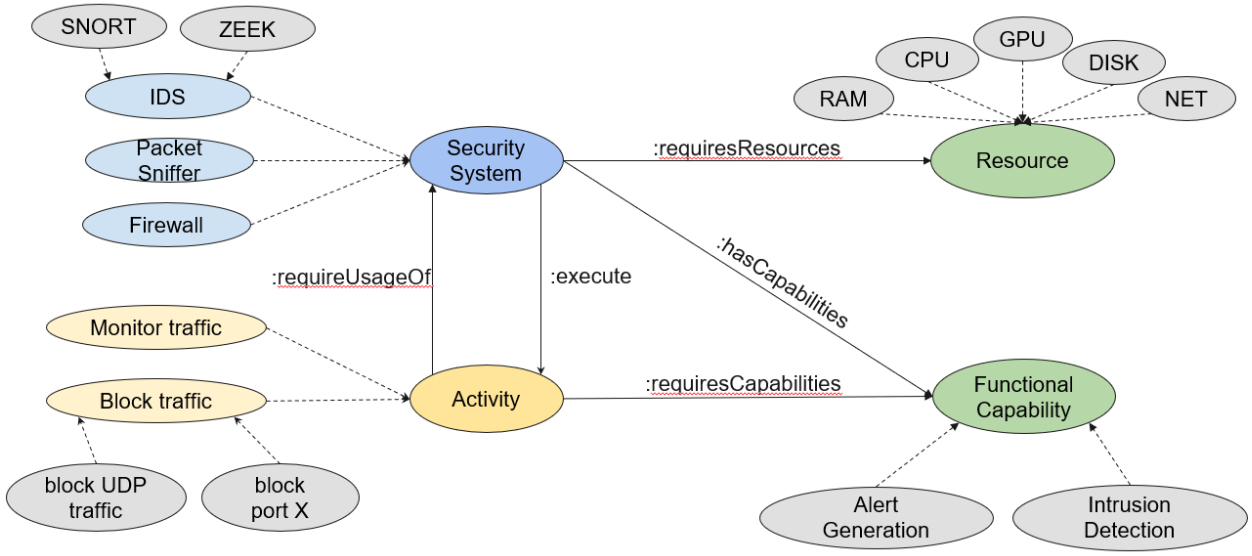

Figure 3-6: SA, AC, NR, FC in the ontology

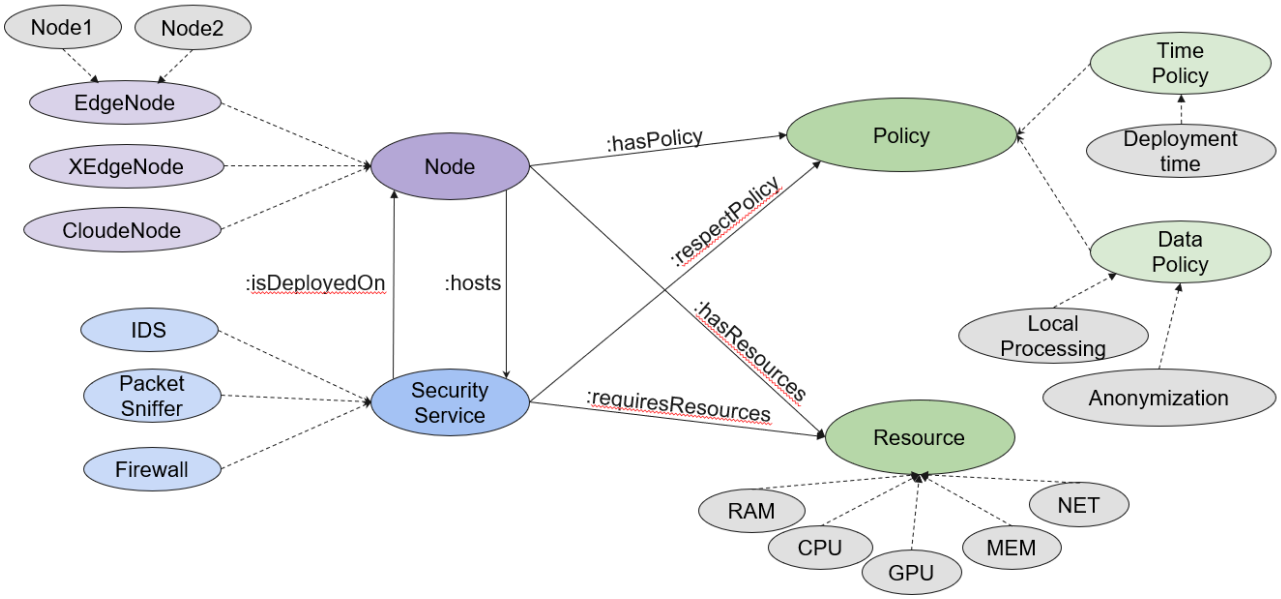

Figure 3-7: TE, SA, P, NR in the ontology

3.2.4 Closed-loops specifications for proactive and reactive orchestration

The security orchestrator needs to maintain two security closed loops to implement proactive and reactive security orchestration:

- A policy maintenance loop to maintain the security policy of a vertical and its services
- A threat response loop to maintain the security posture against identified threats for the services

To implement the policy maintenance loop, the security orchestrator acts as the management point of a policy-based approach for the security requirements coming through the 6G exposure interface and from verticals. In this role, the security orchestrator implements the following assumptions:

- The security orchestrator must establish a new security policy every time a new requirement for security is submitted at T+0 (the first time a security requirement is submitted by a vertical).
- The security orchestrator continuously knows the security policy submitted at T0 and the current state of the services decomposition of the security policy.
- The security orchestrator must maintain the state of the services decomposition to respect the security policy submitted at T+0.

- The security orchestrator must update the security policy and the services decomposition if a new security requirement is submitted at T+1.

To establish a security policy from a vertical as a security services decomposition, the security orchestrator interacts with (Figure 3-8):

- A *Vertical*, through a 6G exposure interface, to collect the security requirements
- At least one *VIM* for each orchestrated domain concerned by the security policy, through a *southbound interface*, as explained in the previous section
- A *Monitoring Platform*, through a *southbound interface*, to implement the monitoring services required to observe continuously the current state of the security services decomposition deployed in each domain concerned by the security policy.
- An *Alerting Management layer*, through a *southbound interface*, to implement the security policy alerting rules, to trigger alerts every time distortion of the security services decomposition is observed in the monitoring events of the security policy.

Furthermore, it is up to each VIM, Monitoring Platform and Alerting Management Layer to deploy respectively the proper security, monitoring, and alerting services according to the security orchestrator's request. Then each one of the former components acknowledges the good health of such services to the security orchestrator. Finally, it is up to the security orchestrator to acknowledge the proper deployment of the services decomposition according to the policy, when every acknowledgement from the underlying domains has been received.

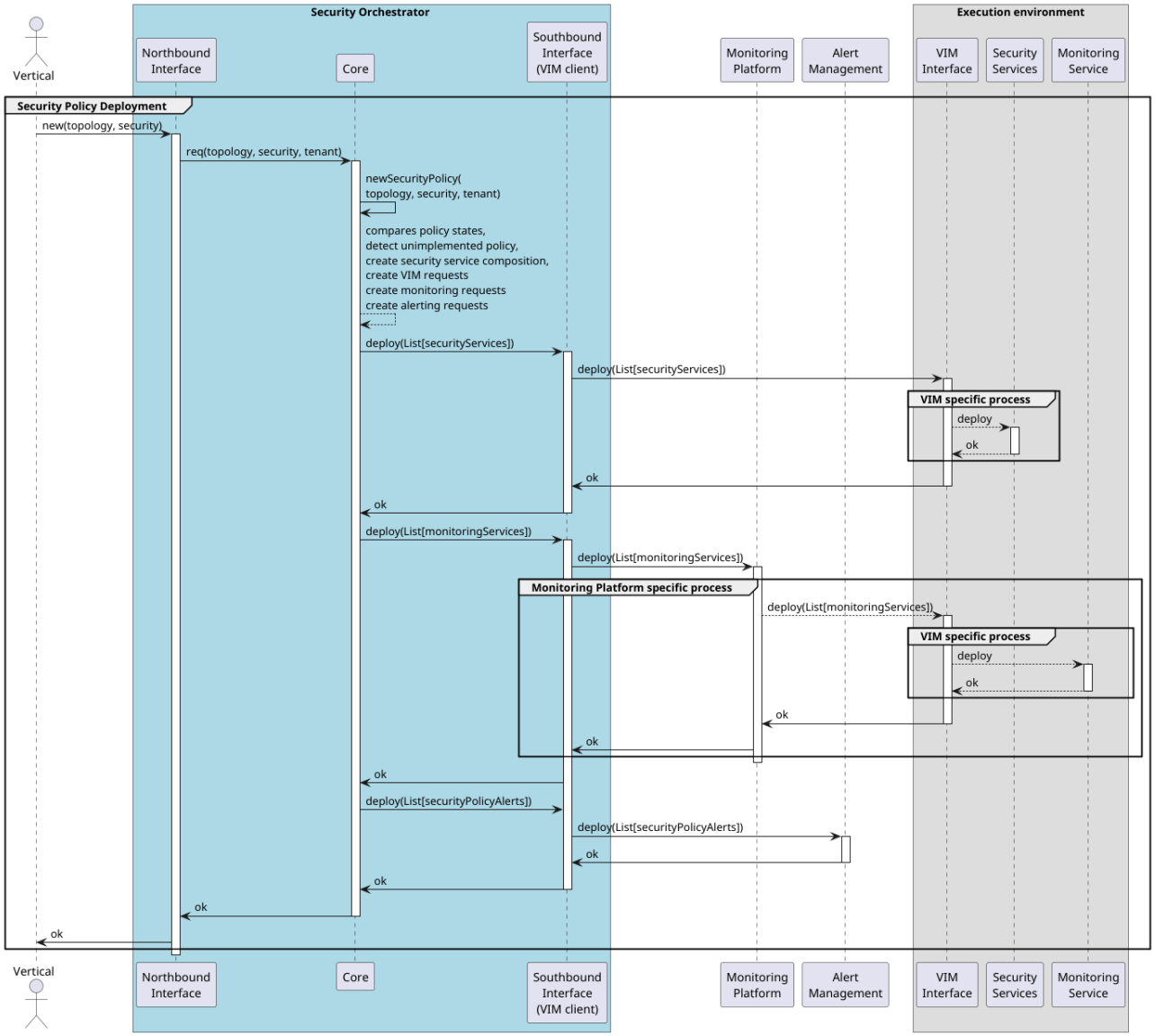

Figure 3-8: Sequence of a security policy deployment from a vertical

To maintain a security policy established at T+0 time, the security orchestrator needs to know continuously the state of the security services decomposition that implements the security policy. Therefore, the security orchestrator is able to collect any alert that concerns a potential distortion of the former state and compute a measured remediation. This remediation is calculated over all the concerned services in all the domains to maintain the policy according to the last security policy requirement expressed by a vertical at T+0. In this perspective, the security orchestrator interacts with (Figure 3-9):

- An *Alerting Management Layer*, through a northern interface, to collect alerts that concern the distortion of the state of the security services decomposition that implement a security policy.
- At least one *VIM* for each orchestrated domain concerned by the alerts, through a *southbound interface*. Each VIM updates the services concerned by the security orchestrator's remediation, following their own processes, but they finally acknowledge the success of their local remediation back to the security orchestrator.
- A *Monitoring Platform*, through a *southbound interface*, to update the monitoring rules of the security policy, if necessary, and to trace the remediation made by the security orchestrator. These updates ensure the detection of future suspicious activities, thus improving the prediction of future distortions.
- An *Alerting Management Layer*, through a *southbound interface*, to update the alerting rules of the security policy, if necessary. These updates ensure the detection of future suspicious activities, thus improving the prediction of future distortions.

This process of detection, alerting, remediation and prediction reflects the process of the policy maintenance loop of the security orchestration.

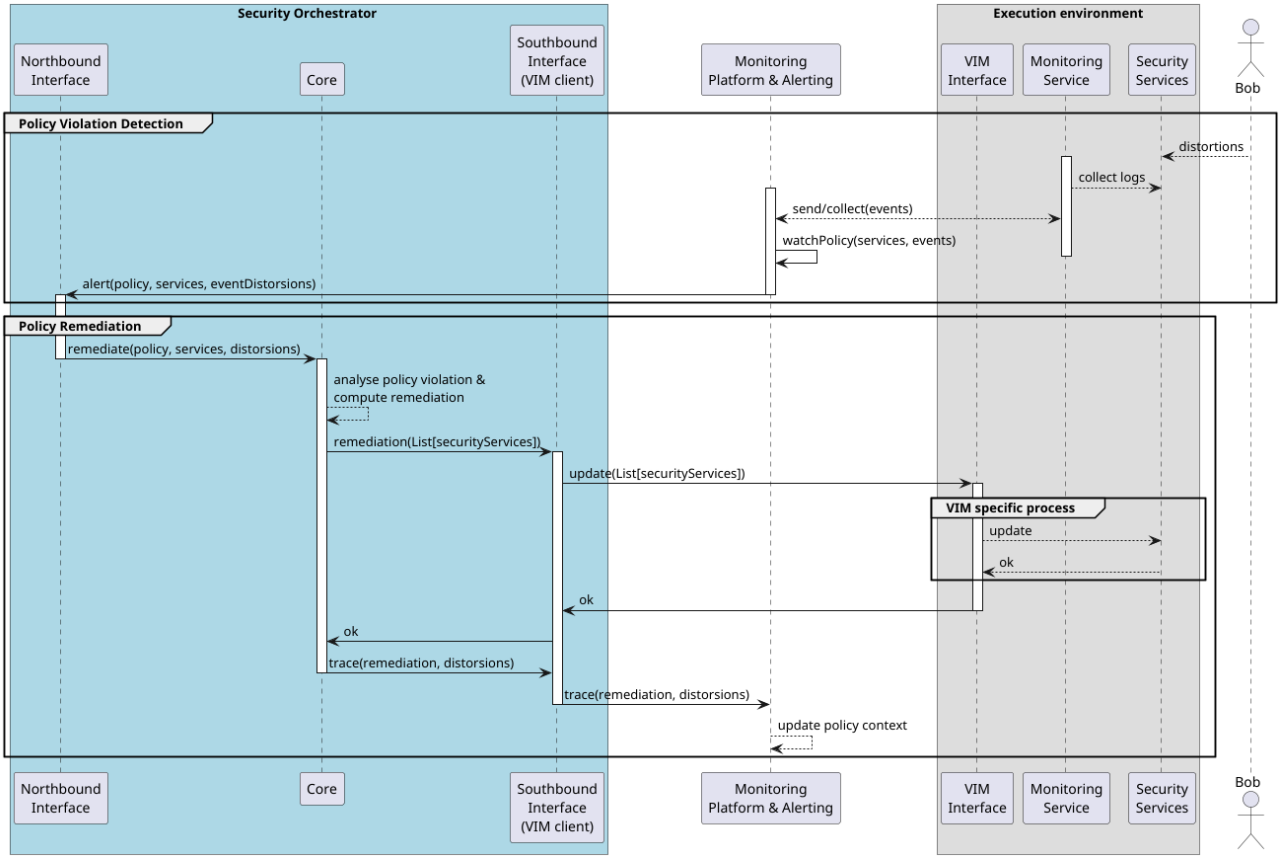

Figure 3-9: Sequence of a security policy remediation in case of policy violation

To implement the threat response loop, the security orchestrator also acts as the management point of security posture requirements coming through a security exposure interface and from system administrators or cybersecurity officers of the service provider's organization. It is important to note that the security posture demand is also expressed as a decomposition of security services. However, this decomposition is dedicated to specific security controls to cover the service provider's orchestrated system against identified threats. In this role, the security orchestrator implements the following assumptions:

- The security orchestrator must establish a new security posture when a new requirement is submitted at T+0 (the first time a security requirement is submitted by a cybersecurity officer).
- The security orchestrator continuously knows the security posture submitted at T0 and the current state of the services decomposition of the security posture.
- The security orchestrator must maintain the state of the services decomposition to respect the security posture submitted at T+0.
- The security orchestrator must update the security posture and the services decomposition if a new security requirement is submitted at T+1.

Since the security posture requirements are also translated to a security services decomposition, the process is similar to the security policy establishment previously described in this section (see Figure 3-10). The main difference is that the service decomposition relates to a threat analysis coming from the cybersecurity management of the service provider, and not from a vertical. Moreover, the security orchestrator interacts with multiple VIMs for each orchestrated domain, so the complexity of the detection of an intrusion or aggression of the security posture can be high. In this perspective, the security orchestrator may require AI/ML models to:

- Assist the deployment of appropriate security services by enriching the services decomposition of the security posture.
- Assist in the detection of the intrusion or the aggression of the security posture by deploying smart agent models into the orchestrated domains.

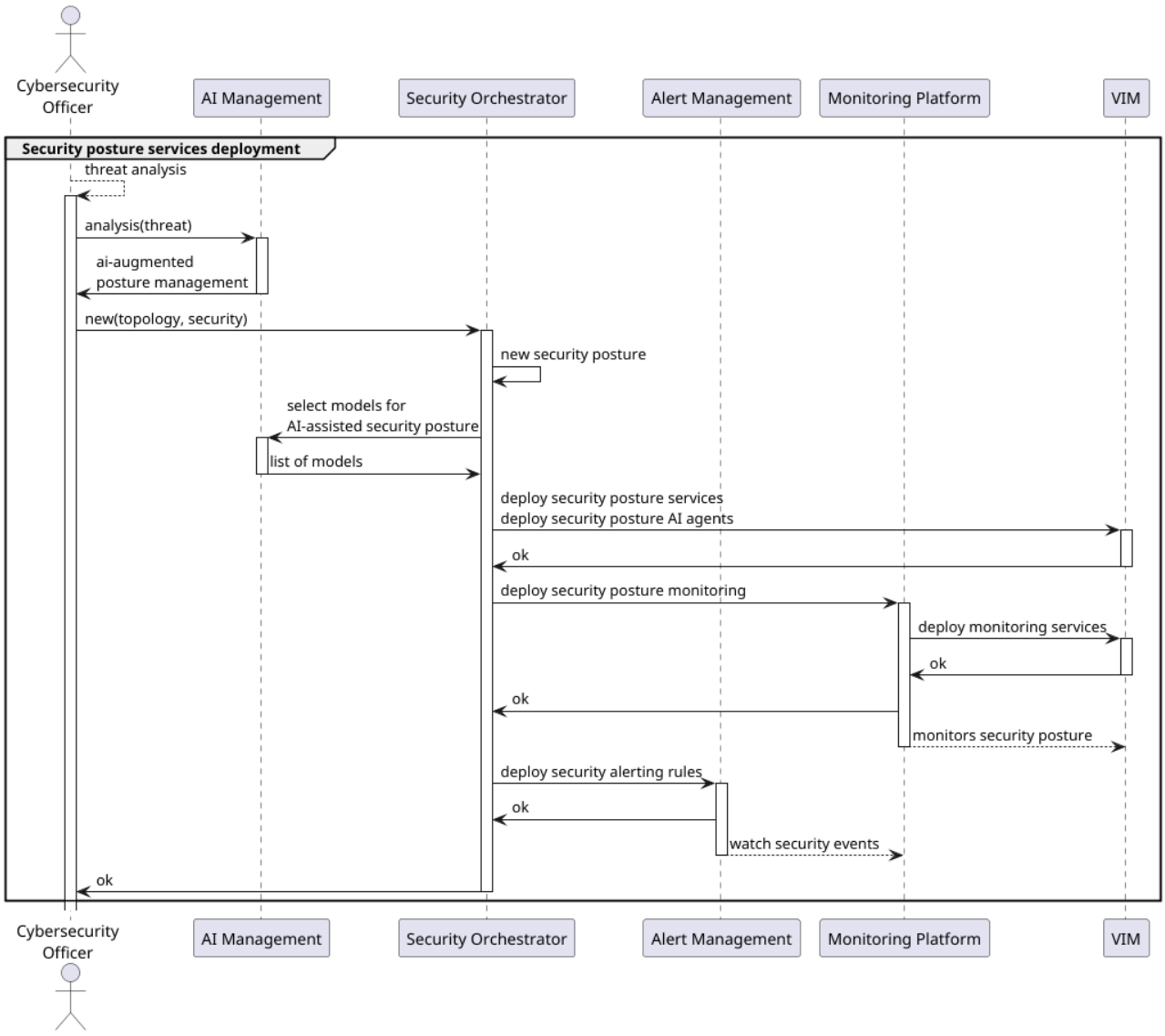

Figure 3-10: Sequence of a security posture deployment from a Cybersecurity Officer

The process of maintaining a security posture established at T+0 time is similar to the process that maintains a security policy. The security orchestrator continuously monitors the state of the security services decomposition that implements the security posture. However, since this monitoring is not configured to detect security services distortion but intrusion events, the process will react to any aggression related to the posture defined at T+0. Since the security orchestrator interacts with multiple VIMs for each orchestrated domain, the complexity of the remediation that concerns the aggression of the security posture can be high. To tackle this problem, the security orchestrator may require further investigation from an *AI and Analytics Management Layer* over collected events, alerts, and enriched Cyber Threat Intelligence (CTI) contexts. In these perspectives, the security orchestrator interacts with (Figure 3-11):

- An *Alerting Management Layer*, through a *northbound interface*, to collect alerts that concern the detection of intrusions or aggression of the security posture.
- At least one *VIM* for each orchestrated domain concerned by the alerts, through a *southbound interface*. Each VIM updates the services concerned by the security orchestrator's remediation, following their own processes, but they finally acknowledge the success of their local remediation back to the security orchestrator.
- A *Monitoring Platform*, through a *southbound interface*, to update the monitoring rules of the security posture, if necessary, and to trace the remediation made by the security orchestrator. These updates ensure the detection of future suspicious activities, thus improving the prediction of future intrusions.
- An *Alerting Management Layer*, through a *southbound interface*, to update the alerting rules of the security posture, if necessary. These updates ensure the detection of future suspicious activities, thus improving the prediction of future distortions.

- A *Cybersecurity officer* authoritative for remediations, revisions, and authorizations, through a northbound interface, to ask for a confirmation to proceed to the remediation in further automated steps. This is an implementation of *human-in-the-loop* for automated security mitigation of the security closed loops. This interaction is optional and can be avoided for fast and non-critical remediations over systems. However, during the threat analysis and security posture definition phases, it is up to the cybersecurity officer to decide whether this step is required for each threat remediation process.

This process of detection, alerting, remediation, and prediction of intrusions reflects the process of the threat response loop of the security posture orchestration.

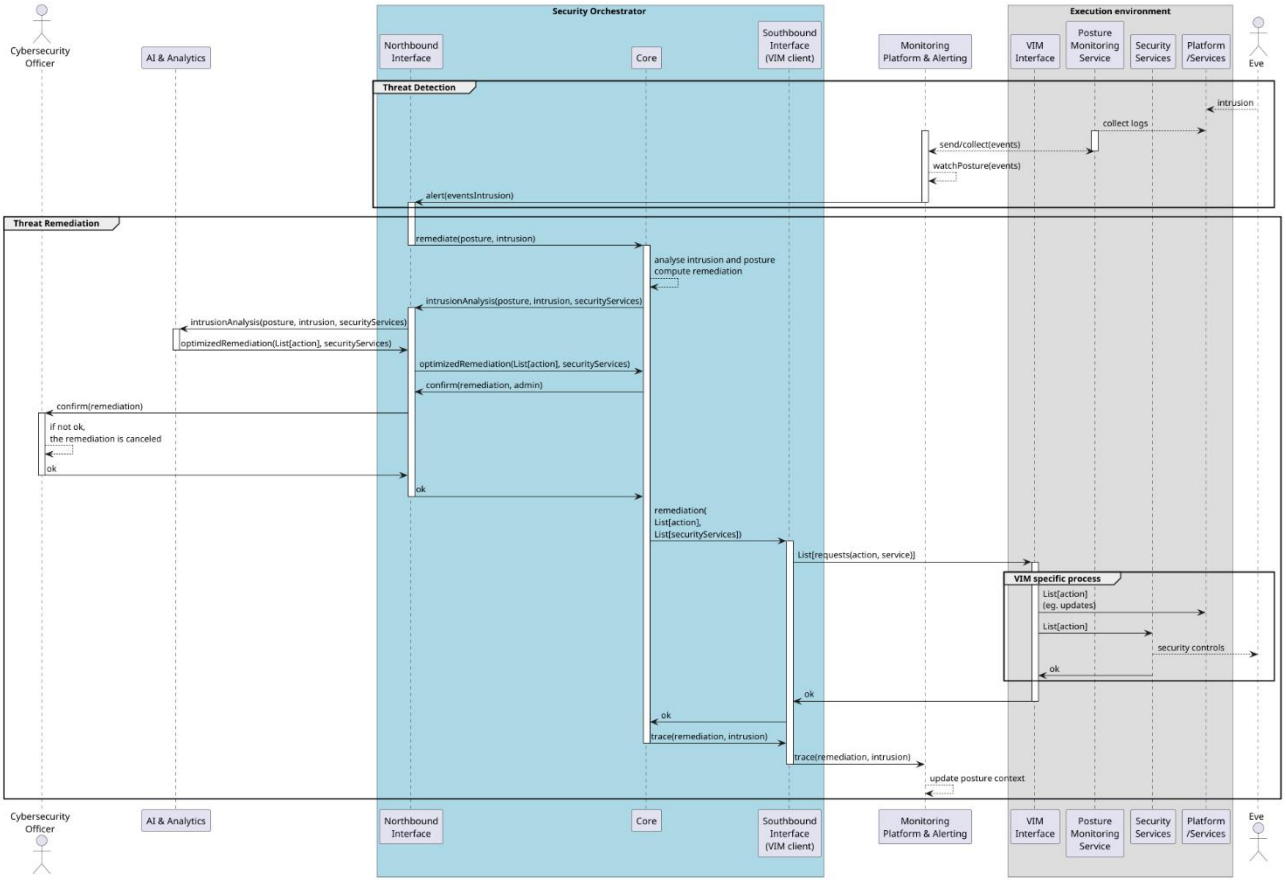

Figure 3-11: Sequence of a threat mitigation in case of intrusion detection

This process can deal with the high complexity of intrusion detections, but it comes with intermediate actions to analyze the context of the intrusion, using the assistance of AI/ML based cognitive tasks (Figure 3-12). Consequently, the process can slow down the whole remediation process and increase the *Mean Time To Respond (MTTR)*, which is crucial for cybersecurity teams. To deal with this limitation, the security orchestrator can rely on the AI/ML models deployed at T+0 time as a part of the security services decomposition. These models implement *AI-assisted fast remediation* for posture maintenance against aggressions and acknowledge the remediation in a second step. Therefore, the security orchestrator can use this acknowledgment to update the security posture, the monitoring rules, and the alerting rules to improve the prediction of future intrusions.

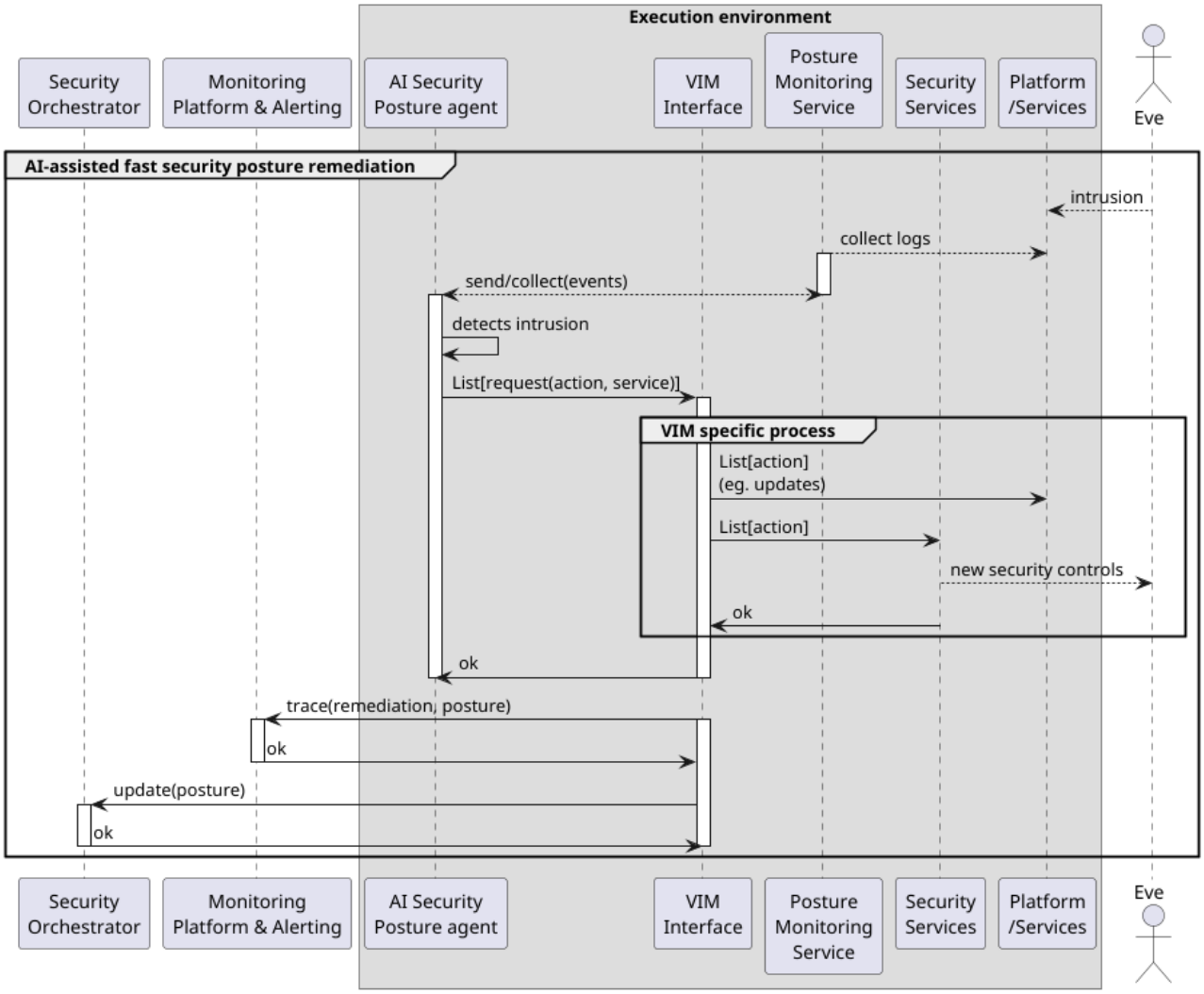

Figure 3-12: Sequence of a fast threat mitigation in case of intrusion detection using an AI agent

3.3 Resource Management for Security

Resource management (RM) represents a complex, critical, and multifaceted task of RAN and edge/cloud operations, which aims to optimize performance, efficiency, and scalability of network infrastructure. Beyond traditional objectives like spectrum and energy efficiency, dynamic resource allocation, and Quality of Service/Experience (QoS/QoE) provisioning, emerging 6G networks introduce amplified challenges for ensuring robust security and privacy, and for performing threat and intrusion detection, often empowered by AI/ML techniques. The increasing complexity and openness of 6G networks demand advanced strategies to address evolving threats while maintaining data integrity and user trust.

3.3.1 Security-Centric 6G Resource Management

Balancing comprehensive security measures with resource limitations across cloud, edge, and far-edge environments is exacerbated by the dynamic and decentralized nature of 6G, which necessitates adaptable and efficient security resource allocation. Resources must be allocated based on security policies tailored to the specific requirements of various network slices and applications, ensuring critical tasks and functions, such as intrusion detection, and encryption are prioritized according to threat levels and latency constraints. Through an intelligent resource orchestration, the system can dynamically adjust to real-time demands, making it scalable and responsive. Achieving optimal security resource allocation requires advanced techniques such as policy-driven orchestration, multi-domain resource coordination, and AI-driven predictive management. These approaches allow networks to anticipate security demands, allocate resources proactively, and maintain optimal performance without overloading the system or requiring overprovisioning. By leveraging closed-loop automation, resource distribution can be continuously adjusted in response to real-time monitoring and threat detection, ensuring proactive and efficient security coverage across diverse 6G nodes.

Although resource management in 6G will build upon and leverage techniques and concepts from current communication systems (such as 4G and 5G), in the context of 6G, traditional security methodologies will be re-evaluated and augmented to address the stringent demands of future network architectures and paradigms. Enhanced and innovative techniques for authentication, encryption, access management, secure communication, and threat detection will be essential to meet the envisioned complex security requirements of 6G networks.

3.3.2 Objectives of Security Resource Management

Effective security resource management in 6G will be centred around the following key objectives, which will shape the resource control and management framework under development in WP4.

- **Efficiency:** this involves optimizing the allocation of security resources, such as those required for intrusion detection, encryption, and access control, as a means to minimize latency and computational overhead. This ensures that security measures are implemented with maximum performance and minimal impact on network operations and waste of resources (e.g., energy, computing).
- **Scalability:** this enables the system to dynamically scale security resources in response to real-time demands, evolving threat levels, and the specific needs of individual network slices or applications. This ensures seamless performance as the network grows and adapts.
- **Adaptability:** this consists of tailoring resource allocation to meet the diverse security requirements and constraints of different network slices, devices, and service types. This flexibility ensures the network ecosystem can effectively address varying demands while maintaining robust protection.
- **Resilience**: this involves the design of resource management strategies that anticipate and recover from disruptions, such as attacks or system failures, and measure potential risk factors, ensuring uninterrupted security operations and network stability under adverse conditions.
- **Prioritization**: this consists of allocating resources based on the criticality of assets, threat levels, and service demands, ensuring that high-priority tasks, such as those protecting critical infrastructure, receive sufficient resources.
- **Transparency**: this ensures clear and understandable resource allocation processes to build trust among stakeholders, such as network operators and service providers, while enabling easier troubleshooting and decision-making.
- **Responsiveness**: this involves the implementation of mechanisms that allow rapid adjustments to resource allocation in response to emerging threats or changes in network conditions, minimizing potential vulnerabilities.
- **Privacy preservation**: inherent in secure resource management tasks, this incorporates privacy-by-design principles to ensure that security measures respect user data privacy and comply with data protection regulations, especially when processing data across distributed nodes.
- **Compliance**: this ensures that resource management aligns with industry standards, regulatory requirements, and organizational policies, enabling the network to meet legal and operational obligations.

3.3.3 Security Solutions for 6G Networks

Addressing these challenges necessitates innovative security solutions that enhance trust, privacy, and resilience across 6G systems. Security strategies must integrate advanced technologies and methodologies, ensuring the network can maintain its robust performance while mitigating risks. For instance:

- **Policy-based resource management**. This defines and establishes security policies that clearly specify resource requirements aligned with the specific needs of each network slice or application. These policies are then enforced via adaptive security orchestration systems that adaptively/dynamically adjust resource allocation in response to changes in resource availability, evolving security demands, and compliance requirements. This ensures that the allocation remains both efficient and aligned with the network's operational priorities.
- **Multi-Domain Resource Coordination:** This effectively manages and distributes security resources across multiple domains (cloud, edge, and far-edge) to ensure efficient distribution of security tasks. Intelligent orchestration tools aware of each network layer's resources will enable the deployment of security services closer to the data source whenever possible. This approach reduces latency, enhances

overall resource utilization, and minimizes the load on centralized infrastructure. Cross-domain orchestration will ensure that dynamic resource allocation meets the needs of delay-sensitive applications without overwhelming individual nodes, maintaining seamless performance across the network.

- **AI-based predictive security:** AI models can be integrated to predict potential security by analysing historical data. This predictive capability enables the system to pre-emptively allocate and adjust resources to mitigate risks before incidents occur. Such a proactive management approach not only helps maintain a robust security posture but also ensures sustained performance and resilience under dynamic conditions since threats are anticipated and countered before they escalate. In ROBUST-6G, an arsenal of explainable and trustworthy AI models and algorithms is being developed for critical security-oriented tasks.

By adopting these advanced techniques, 6G networks will effectively safeguard against emerging threats while maintaining the agility, efficiency, and reliability required for next-generation connectivity.

3.3.4 Security Resource Orchestrator: High-level Design

This section describes the preliminary work on the Security Resource Orchestrator (S-ROr) as the implementation of the S-RO macro-service already presented in Section 3.1 and Deliverable D2.2. As discussed, the S-RO and its implementation could be integrated into the Security Orchestrator, nonetheless, the management and orchestration of cloud resources implies, in general, the interaction with multiple cloud platforms residing in different cloud facilities across the cloud continuum and, in particular in ROBUST-6G, the implementation and integration of specific mechanisms for robust and secure resource allocation (an initial work is described in Section 3.3.5). The high-level architecture of the S-RO implementation is shown in Figure 3-13. Internally there are four main modules:

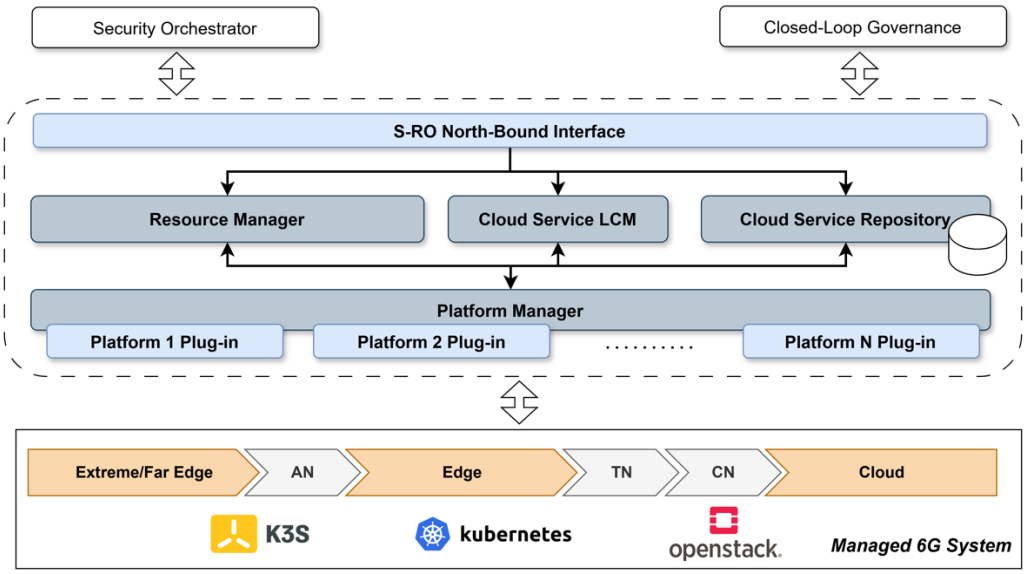

Figure 3-13: Security Resource Orchestrator high-level architecture

- **Resource Manager.** Collects (resource discovery) and maintains (resource inventory) resource information related to the different Cloud Platforms in the cloud continuum consumed by dedicated algorithms to decide the optimal resource allocation for a given service/application. Such algorithms can be integrated into the resource manager or belong to 3rd party components, e.g., the Security Orchestrator or specialized modules for robust and secure resource allocation.
- **Cloud Service LCM.** In charge of the lifecycle management (LCM) of the cloud services and applications, which may include, traffic analysers, anomaly detectors, virtual firewalls, CL stages, etc.
- **Cloud Service Repository.** Maintains the descriptors, i.e., the definition of the Cloud Services. These could be represented either in a uniform manner, e.g., via a unified data model which will be remapped to the specific cloud platform representations, e.g., Kubernetes Deployments YAML, Helm charts, etc., or by the direct use of such specific representation.

- **Platform Manager.** Provides the logic and the interfaces to interact with the different cloud platforms e.g., Kubernetes, K3s, OpenStack, etc., through a set of plug-ins specific for the target interfaces. As shown in the figure, the target segment can be generic Cloud, Edge, Far Edge, and even Extreme Edge, which can include devices e.g., drones, cobots, sensor platforms, etc. with enough resources (CPU, RAM, Storage) to be orchestrated.

The different functionalities provided by the Security Resource Orchestrator are exposed by a unified North-Bound Interface (NBI) and consumed by the Security Orchestrator, to request the deployment of the application part of a security service, and by the CL Governance, which represents the lifecycle manager of the different S-CLs provisioned to enable the security automation. The S-RO is indeed one of the components in charge of deploying the CL stages under a direct request of the CL Governance. This aspect is clarified in Section 3.6 that details the ROBUST-6G implementation of S-CLs.

3.3.5 Risk-averse Resource Management

When considering security threats and privacy leaks as key risk factors, resource-related decisions and orchestration cannot adopt a risk-neutral approach, as it is typical in the expected utility theory (EUT)-based optimization models used in current network control and optimization (e.g., in 4G and 5G). To address this limitation, in ROBUST-6G, we develop a robust, risk-averse, context-aware yet agile resource management framework. This framework evaluates system quality, performance, and constraint metrics – such as security, privacy, and robustness KPIs – through subjective, and usually context-dependent perceptions. Our approach incorporates decision-makers (e.g., human agents) whose preferences and choices align with the principles of cumulative prospect theory (CPT) [Tve92]. By doing so, we capture human-centric aspects of decision-making, including risk perception, loss aversion, and perceptual distortions in the distribution of uncertainty. The result is a comprehensive analytical framework that enables fine-grained and adaptive decision-making under uncertainty, addressing both technical and behavioural dimensions of resource management.

This risk-aware resource control and management 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. By serving as a potential input to the S-RO component, this framework could update the current components by introducing the capability 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. This framework provides the analytical optimization framework and is designed to seamlessly integrate into the S-RO architecture.

The current S-RO architecture focuses on ensuring efficient and secure resource management, and our framework aligns with these objectives by embedding algorithms that evaluate potential risks and decision impacts comprehensively. Through this approach, the framework not only facilitates proactive risk mitigation but also enables the system to account for subjective or context-specific factors that influence decision-making. This added layer of intelligence ensures that resource management decisions are not only optimal but also aligned with broader organizational or operational goals, as well as the end-user's and stakeholders' perceptions, needs, and expectations of robustness, security, and privacy.

We provide below a brief overview of the analytical RM framework, which has the following features and ingredients that are lacking from EUT: (i) reference dependence, (ii) loss aversion, (iii) diminishing sensitivity to returns for both, gains and losses, (iv) probabilistic sensitivity, (v) rank dependence and cumulative probability weighting.

3.3.5.1 Reference Dependence

Users perceive value (semantics) and indicate preferences through deviations from an existing reference point x_0. This reference point may represent an acquired or expected operating level or quantity (e.g., minimum achieved security guarantee operating point under typical system operation) and may differ across application scenarios. The utility function domain is partitioned into two regions relative to x_0.: the loss domain $x < x_0$ and the gain domain $x \geq x_0$. In the loss domain, $u(x) < 0$, $\forall x < x_0$ and in the gain domain, $u(x) > 0$, $\forall x \geq x_0$.

3.3.5.2 Utility function and curvature

Any utility or value function satisfies the following fundamental properties in the framework of classical CPT in terms of curvature and monotonicity: (i) it is continuous and strictly increasing in x; (ii) $u(x_0) = 0$; (iii) $u(x)$ is concave when $x \geq x_0$ and is convex when $x < x_0$ (diminishing marginal utility); (iv) loss aversion.

The curvature of the utility function (marginal utility) characterizes attitudes toward risk (risk aversion) and the user's sensitivity to varying scales of change within each subdomain. Specifically, in the gain domain, a convex utility function suggests that the user perceives changes further from x_0 more intensely, whereas a concave function indicates that the agent is more sensitive to changes occurring closer to x_0. Conversely, in the loss subdomain, the opposite holds. This differs significantly from the typical assumption in EUT that the utility function is concave throughout.

3.3.5.3 Loss Aversion

The CPT utility function exhibits loss aversion, meaning that agents are more sensitive to losses than to equivalent gains. This property is mathematically formulated by requiring that the right-hand marginal derivative of the utility function at the reference point should be smaller than the left-hand derivative, i.e., $u'(x_0^+) < u'(x_0^-)$.

3.3.5.4 Probability Weighting Function

A key attribute of this framework is the non-linear probability distortion, in which objective probability is distorted when being perceived by end users according to a probability weighting function (PWF). The PWF typically models uncertainty perception and captures the effect that users' overweight small probabilities and underweight moderate and high probabilities. The PWFs w are continuous and strictly increasing. As a baseline we will use the behavioral Prelec function [Pre98] $w(p) = e^{-\gamma(-\ln p)^\theta}$, where the parameter $0 < \theta < 1$ controls the curvature of the PWF and the parameter $\gamma > 0$ the location of the inflection point, relative to the line w(p) = p. We will also provide extensions of PWF using generalized entropies.

3.4 Continuous Monitoring and Threat Detection

ROBUST-6G requires advanced monitoring capabilities to detect potential issues in distributed networks, including operational anomalies and emerging security threats. Therefore, we proceed to design an automatic, distributed, and programmable platform for monitoring security characteristics that are subsequently considered by threat detection mechanisms in distributed networks based on the end-user security requirements declared in the Security Service Level Agreements (SSLAs).

Thereby, Section 3.4.1 presents background information on the platforms that can serve as a foundation for initiating a programmable monitoring platform and possible threat detection solutions. In this regard, a study of software tools that meet the necessary design requirements of the platform is also performed, guaranteeing functional and non-functional requirements defined in Deliverable D2.2. By means of main conclusions gathered, an initial architecture design will be presented when continuous monitoring and threat detection (rule-based) activities are depicted via multiple submodules (see Section 3.4.1). In addition, Section 3.4.2 introduces a semantic-aware anomaly detection approach to detect equipment failures or abnormalities without overwhelming systems.

3.4.1 Programmable Monitoring Platform

The Programmable Monitoring Platform (PMP) is an important component for ROBUST-6G architecture that covers the observability phase of the closed-loop generated by the zero-touch security platform. Before starting the design of ROBUST-6G continuous monitoring and threat detection, a review of the state-of-the-art related to zero-touch network and service management (ZSM) is conducted, obtaining know-how to contribute to such a field.

Previous works investigated ZSM for multi-tenant environments based on 5G and Beyond 5G (B5G) networks using techniques based on AI algorithms, security orchestration, and AI analytics [GDZ+23]; others reflected the limitations and the risk security of the AI in this kind of ZSM platforms [BT20]; or even included new features such as blockchain to manage network automation and trustworthiness [XKK+23].

However, none of these articles demonstrated how observability and monitoring actions integrate or interact with other components of ZSM platforms. This is the reason why it is necessary to describe and determine a design and future development of a PMP that can be adaptable and functions as a reference for forthcoming research projects.

The principal objectives of PMP in ROBUST-6G are to support comprehensive network and resource security monitoring. It may collect data at multiple layers of the 6G networks, including services, infrastructure, and network layers. It also may incorporate monitoring agents tailored for far-edge, edge, and cloud domains, enabling selective data distribution and greater system flexibility. In addition, the platform may also include a correlation mechanism to analyse similar data collected in different environments, which will improve the value of the separately collected data, as well as the solutions to be offered to the end-user, as detection and prediction algorithms may have a wider scope of events and relationships.

The PMP could be designed to support extensibility and programmability, potentially allowing new monitoring tools or components to be added without altering its core architecture. Also, it can focus on efficiently aggregating and pre-processing data to optimise resource use and minimise redundancy. Another important characteristic to be contemplated in a PMP solution is related to communication security, which may include the data transmission between the internal components and the external parts of the closed-loop to avoid information exposure. Additionally, the system may dynamically adjust the configuration of its components to meet changing needs or conditions. These objectives will provide competencies to design a PMP to manage future challenges in distributed networks over the Cloud Continuum.

On the other hand, the threat detection component will provide a security capability to closed-loops by consuming the collected data by the PMP, detecting possible anomalies based on rules or trained AI models, and creating alerts to be resolved by other components. This method enhances a continuous monitoring system, improving and reducing the impact of threats.

Taking these considerations in account, tools that provide monitoring microservices on service, network and infrastructure layers are researched, as well as tools that can transmit the information collected by monitoring agents to be pre-processed, aggregated, normalised and, in case of detecting anomalies, to inform other modules by means of an alert system. In addition, a data storage system and a data visualisation system are needed. Also, the platform must be virtualised to improve the scalability, efficiency and lightweight among them.

First and foremost, it is important to cluster the tools by their type of activities, in order to relate them to the considerations that the PMP may have. The main groups of *activities* are listed below:

- Monitors
- Pre-processors and aggregators
- Alert systems
- Visualisers
- Databases
- Event streamers

Once the tools are grouped, the next step is to know what tools may do. Thus, an assessment of each tool is required to check whether they can solve any requirement of the platform. So, in addition to grouping tools by their activities, they will also be classified by their *functionalities*, having subclassified fields to enhance compression (see left column in Table 6-1.

In the group of *Monitoring functionality*, several capabilities are included, such as network monitoring through trace analysis, optionally with the ability to detect devices on the network; host monitoring, which provides general information about devices' operative system; the collection of health metrics related to processes or even the network; and the management of event logs. Under the group of *Anomaly functionality*, the functionalities cover the detection of anomalies in the network, similar to an Intrusion Detection System (IDS). Additionally, it may also include an alert system capable of generating notifications when an issue or an anomaly is detected, as well as the ability to stop anomalies, working as an Intrusion Prevention System (IPS).

The *Virtualisation functionality* details the feature to deploy the tool using containers, with official images created by the developers or sponsored partners available on Docker Hub. It also highlights official Kubernetes compatibility, as reflected in its documentation. Regarding *Infrastructure functionality*, some tools require or may use agents specifically designed by their developers to perform correctly. Similarly, proxies exclusively

created by the tool may also be required. The *Vulnerability detection functionality* includes the ability to detect vulnerabilities or unexpected behaviours in standard devices at the host level, using approaches such as Static or Dynamic Application Security Testing (SAST or DAST). It can also identify issues in images and containers under the same principles.

The *Visualisation functionality* focuses on generating graphs that can be visualised either by the tool itself or by plugins. It also allows the creation of dashboards to display information gathered by the tool or extracted from other sources. Other features include the default storage of data in a database installed in the tool and the availability of several plugins to extend its capabilities.

Performance metrics cover *Throughput functionality*, measured in millions of messages per second, classified as very high, high and moderate, depending on the efficiency of message transmission. *Latency functionality* measures the time it takes to transport information in seconds and is classified as ultra-low, low and moderate. Finally, *Delivery guarantees functionality* reflect the level of certainty in message delivery, divided into high, moderate and low levels.

After knowing which categories and characteristics may be met, this section proceeds to describe a set of tools that may match these characteristics, emphasising that tools are freeware, which mean that the software researched has zero monetary cost. In this regard, multiple security-related tools are briefly described and clustered based on the main groups of *activities*, starting with "Monitors" and ending with "Event Steamers" groups.

1. **Tshark** [Tsh24], a command-line network analyser that performs real-time packet captures. It is based on containers but does not have an official image on Docker Hub. It is the minimalist version of Wireshark, since it only uses a terminal to collect network traces, using filters if necessary, and storing it in files that can be opened later.
2. **Snort++ or Snort3** [Sno24] is a flexible IDS and IPS that excels in real-time traffic analysis, packet logging, support the creation of custom rules to detect and stop threats. Thanks to the new repository of plugin for Snort ++, it can collect event logs. It can be containerised, but do not have an official or sponsored image on Docker Hub. In addition, it implements an alert system using rules. One of the most important capabilities is the deep packet inspection which can handle several types of protocols, in special the GPRS Tunnelling Protocol (GTP) protocol used in 5/6G networks.
3. **Falco** [Fal24a] is a DAST tool that excels in real-time threat detection for containers, Kubernetes, and cloud environments. It monitors log events and the metrics that Falco can collect are only about its own performance, not about other devices. Falco can have a proxy instance using Falcosidekick [Fal24b] daemon to extract all the information centrally from Falco nodes and transport it to third party tools. Falco integrates a real-time alert system. It can be containerised, having an official image on Docker Hub [Doc24g].
4. **Collectd** [Col24] is a monitoring tool designed to collect and store system and service metrics in real-time. It has a plugin to generate alerts, providing a basic alert system using metrics, not threats as Zabbix [Zab24a] or Prometheus Alertmanager [Ale24] can do. Collectd can be containerised, is also officially compatible with Kubernetes environments and has an official image (not appearing as official) uploaded by the organisation account [Doc24h]. It is an official image because is linked to the GitHub project "collectd/ci-docker" [Git24f] associated with the official authors. Collectd cannot generate graphs per se, but it creates RRD files that third party tools such as Kcollectd [Kco17], Grafana [Gra24a] or RRDtool [Rrd24] can interpret and draw.
5. **Fluentd** [Flu24a] is a versatile log and metric collector that excels in unifying log data from diverse sources and streaming it to multiple destinations. In addition, it used for log aggregation and data analytics. Fluentd can be containerised, is compatible with Kubernetes and has an official image uploaded on Docker Hub by the organisation [Doc24m] and other services in its official account [Doc24n]. Fluentd can act as a proxy to transmit logs between agents and can be used also as an agent to gather information.
6. **Fluentbit** [Flu24b] is the lightweight version of Fluentd and is oriented to resource-constrained environments. It has the same features as Fluentd, and its official image is also uploaded on Docker Hub [Doc24o].
7. **OpenTelemetry** [Ope24] is a framework designed for observability, collecting network traces, health metrics and event logs from distributed systems. Despite it collects network traffic, it does not have the capability to detect devices on the network. The services/applications need to expose their own

host information to be extracted by OpenTelemetry. It can be containerised, is compatible with Kubernetes and has an official imagen uploaded by the organisation account on Docker Hub [Doc24p].

8. **Telegraf** [Tel24] is a lightweight, plugin-driven agent for the InfluxData suite and collector of health metrics and events from a wide range of sources. It can also find out the system data of devices thanks to plugins. Telegraf acts as a default agent, but can be configured to send data via a proxy. Such as other tools, it can be containerised and has an official image uploaded on Docker Hub [Doc24r] and maintained by the official organisation.
9. **Prometheus** [Pro24a] is a monitoring, processor, normalizer and aggregator tool. It is one of the most famous tools to collect metrics, information system and manage containers thanks to its exporters. An exporter is a plugin for Prometheus. Event logs cannot be gathered as such but can be processed when they come from multiples sources. In case of anomalies in the information, these can be detected by a rule system that will generate appropriate alerts. Prometheus can be containerised and has and official image from Docker Hub [Doc24s]. In addition, the information collected and extracted from other sources can be visualised by the creating graphs and dashboards. To conclude, the data is stored in an internal Time Series Database (TSDB), but it can be served as a buffer, exporting the information to an external TSDB.
10. **Prometheus Alertmanager** [Pro24b] is an alert manager specializes in handling and routing alerts generated by Prometheus. It provides a high level of alert inhibition, deduplication, grouping and silencing. These features bring the capacity of supress notification for certain alerts if other alerts are already being triggered, remove duplication of content, group categorised alerts of a similar nature into a single notification, and mute alerts for a given time. Prometheus Alertmanager can be containerised and has an official image uploaded on Docker Hub [Doc24s].
11. **Kapacitor** [Kap24] is a real-time data processing and alerting engine designed to work with the InfluxData suite, in special with the TSDB InfluxDB [Inf24]. It provides stream and batch processing capabilities, supporting anomaly detection by means of scripts related with alerts configurations. This is not an IDS per se. As other tools of the InfluxData suite. Kapacitor also can be containerised and has an official image on Docker Hub [Doc24u].

Expanding the list, the following types of tools are outlined in relation to the database, event streaming, and visualisation categories.

1. **Grafana Mimir** [Gra24b] is a horizontally scalable TSDB that is highly optimised for large-scale information and data compression for distributed environments. It can be containerised and has an official image on Docker Hub [Doc24x].
2. **MongoDB** [Mon24] is a flexible NoSQL database that excels in unstructured data management due to its document-based model. The alert configuration related with performance metrics, resource usage, and certain event is carried out with MongoDB Atlas (its cloud service). MongoDB can be containerised, having an official image on Docker Hub [Doc24z]. In addition, MongoDB Compass provides a graphical interface to interact with the database, realise queries and visualise plots.
3. **InfluxDB** [Inf24] from Influx suite is a high-performance TSDB optimized for storing and querying time-stamped data. It supports real-time ingestion with high write and query performance, enabling the storage of metrics, events and IoT data in sophisticated environments. Its alert system works like a classical rule-based system that monitors data, issues a status, creates an alert and notifies certain endpoints. InfluxDB 2.0 has an interface to visualise the data stored in the database and can create simple graphs. Finally, it has a completely compatibility with virtualisation thanks to the official docker image uploaded on Docker Hub [Doc24ab] and its integration with Kubernetes.
4. **Apache Kafka** [Kaf24a] is a distributed streaming platform with a very high throughput and scalability, low latency, and fault tolerance supporting replication and confirmations. Acts as intermediary between a publisher/subscriber, generating topics that external tools can load data to be consumed by other tools. Kafka Connect [Kaf24b] works as an agent to connect Kafka with other components, allowing the ingest and exportation of data to and from Kafka. In addition, Kafka Connect provides a plugin system to extend the operability of Kafka. It can be containerised and has an official image on Docker Hub [Doc24ac].
5. **Filebeat** [Fil24] is a lightweight agent stream forwarder and shipper from various sources to centralise storage. Its main function is collects logs and exports to third party tools, but it can work in isolation as an agent for the transmission of information. Filebeat supports integration in decentralised and distributed environments despite offering moderate throughput and latency, and does not guarantee

delivery or persistence. Moreover, it can be configured to collect and transmit information via a proxy. Filebeat can be containerised and has and official image upload on Docker Hub [Doc24ad].

6. **RabbitMQ** [Rab24] is a message and streaming broker for distributed environment using Point-to-Point and publisher/subscriber communication paradigms. It brings high throughput with low latency and high delivery guarantees, as it is optimised for messages reliability and persistence. RabbitMQ cannot generate graphs by itself, but can visualise, in real-time, information tables about the state of the data and metrics of the queues, connections and nodes. Its functionality is extensible with plugins, and it can be containerised, having an official image on Docker Hub [Doc24af].
7. **Grafana** [Gra24a] is the most common analytic and visualisation platform for creating and interacting with real-time dashboards to monitor metrics from various data sources. It supports a lot of formats and information addressed from software such as InfluxDB, MongoDB, Kafka, Prometheus and Elasticsearch among others. Grafana presents an internal data analyser to evaluate anomalies in the information coming from external sources, generating alerts if necessary. Moreover, Grafana can be containerised and has an official image uploaded on Docker Hub [Doc24v] and is fully compatible with the Kubernetes environment. Its extensibility via plugins enhances further its adaptability in a variety of environments.
8. **Chronograf** [Chr24] is a visualisation and alert management tool. As a main feature, it creates real-time dashboards from InfluxDB, because this tool is one of the InfluxData suite and is driven specially for this suite. On the other hand, alerts are made on data received based on anomalous events. These alerts are simple, due to the fact that Kapacitor extends this characteristic. Chronograf has an official image containerised on Docker Hub [Doc24w]. The plugin system is managed by Telegraf, as it is the tool in charge of it in the InfluxData suite.

In addition to these tools, a comprehensive assessment of tools for each type of activity is provided in the Annex section. Table 3-2 and Table 3-3 show a summary of the described tools to has a human-friendly visualisation. The “X” nomenclature represents default tool features and the “*” are used for special cases that have been previously explained in the text. In addition to this, Table 6-1, Table 6-2 and Table 6-3 are also shown in this Annex section with the rest of the tools evaluated.

Table 3-2 Classification and evaluation of monitoring, pre-processing and aggregation, and alerting tools

		Monitors, pre-processors and aggregators, and alert systems										
		Tshark	Snort++	Falco	Collectd	Fluentd	Fluentbit	OpenTelemetry	Telegraf	Prometheus	Prometheus Alertmanager	Kapacitor
Monitoring	Network	X	X					*				
	Host				X			*	*	*		
	Metrics			*	X	*	*	X	X	X		
	Logs		*	X		X	X	X				
Anomalies	Anomalies detection		X							*		*
	Alert system		X	*	*					X	X	X
	Anomalies stop		X									
Virtualisation	Containerisation	X	X	X	X	X	X	X	X	X	X	X
	Official/Sponsored image			X	X	X	X	X	X	X	X	X
	Official Kubernetes compatibility			X	X	X	X	X	X	X	X	X
Infrastructure	Can use agents			X	X	X	X	X	X	X		
	Can use proxies			*	X	X	X		X			
Vulnerability detection	Host level			X								
	Container level			X								
Visualisation	Graph generation				*					X		
	Graph visualisation									X		
Other features	Default data storage									*		
	Plugins			X	X	X	X	X	X	*		

Table 3-3 Classification and evaluation of databases, event streamers and visualisation tools

		Databases, event streamers and visualiser							
		Grafana Mimir	MongoDB	InfluxDB	Kafka	Filebeat	RabbitMQ	Grafana	Chronograf
Anomalies	Anomalies detection								
	Alert system	X	*	X					
	Anomalies stop								
Virtualisation	Containerisation	X	X	X	X	X	X		
	Official/Sponsored image	X	X	X	X	X	X		
	Official Kubernetes compatibility	X	X	X	X	X	X	*	*
Infrastructure	Can use agents				X	X			
	Can use proxies		*			X		X	X
Visualisation	Graph generation		*	X				X	X
	Graph visualisation		*	X			X	X	X
Other features	Default data storage	X	X	X					
	Plugins				X		X		
Throughput	Very high				X				
	High						X		
	Moderate					X		X	X
Latency	Ultra-low							X	X
	Low				X		X		
	Moderate					X		X	*
Delivery guarantees	High				X		X		
	Moderate								
	Low								

In order to unify and organise the concepts studied in this section with the requirements that the PMP may have in conjunction with threat detection and the tools, a workflow is designed to illustrate the process of its functionality. Additionally, the workflow is going to be associated with the possible tools previously reviewed.

As shown in Figure 3-14, the *Security Orchestrator* acts as an issuer of the necessary features (including security constraints coming from SSLAs) for the *Configuration Manager* to configure the components. After gathering raw data, the *Data Collection Module* transmits the information to the *Data Aggregation and Normalisation Module*. The information is stored in a database, and then the *Exporter* is responsible for transmitting the data to the external components interested in consuming monitoring outcomes. For example, the Data Fabric may receive the information to combine the PMP data with other additional information sources (not considered by the PMP) and infer new knowledge that can finally be leveraged by *Threat Detection* or *Threat Prediction*. In case the *Threat Detection* detects a problem, it can generate an alarm using the *Alert and Notification Module*. It is worth mentioning that the *Alert and Notification, Threat Detection,* and *Threat Prediction Modules* are not reflected on the overall ROBUST-6G architecture since they cover specific activities under WP4. Nonetheless, they consume from the Programmable Monitoring Platform outcomes.

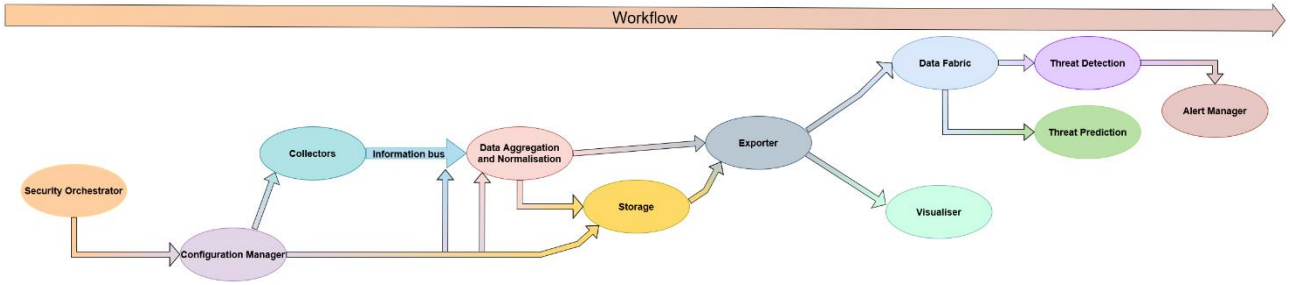

Figure 3-14: Workflow of the Programmable Monitoring Platform

Once the tentative tools have been described, they may be associated with the modules of the PMP to obtain an abstract map of possible associations. This will be a possible mapping of the solution, but it does not mean that it is the combination of tools that will be used in the final design.

Starting with the collectors, the PMP needs to collect health metrics from the infrastructure layer, logs from the service layer, and network traces from the network layer. Therefore, to cover these functionalities, the PMP may use different configurations using Telegraf for health metrics, Collectd for health metrics and logs, OpenTelemetry for health metrics, logs and network traces, Fluentd/Fluentbit for health metrics and logs, Falco for logs but not work as well as other tools and dynamic security, and finally, Snort++ and Tshark for network traces.

The information collected needs to be exchanged with other internal modules to be pre-processed and aggregated. The responsible for transmitting the information is a bus that can use tools such as Kafka, RabbitMQ, and Filebeat.

The data in distributed environments comes from multiple sources. Due to this phenomenon, the PMP relates the information through aggregation techniques. In this case, only one tool was assessed, so the configuration is limited to Prometheus.

The PMP may store the information in a database to preserve the information that will be used by several components such as the Threat Detector and the Thread Predictor for model training. Grafana Mimir, MongoDB and InfluxDB may be used to develop this work, but TSDB is the optimal type for conserving log-related information.

An API exporter transmits the information to other components and a visualiser that may be represented by Grafana or Chronograf to show the data through graphs and plots.

Related with the threat detection component, it may use Snort++ to detect anomalies in the network and generate alerts with Snort++, Prometheus Alermanager or Kapacitor. The last tool may be combined with the others InfluxData suite software.

In the context of continuous monitoring, a number of modules have been designed for implementing the PMP, as shown in Figure 3-15. These modules are intended to provide a more comprehensive and detailed view of

system operations, with the objective of improving operational efficiency. The modular approach allows for an adaptative and programmable solution to be adopted in accordance with the specific needs of each environment.

The **Configuration Manager GUI or API** is the point of contact between end users and the PMP that would be used to interact with the PMP configuration module. In addition, the Security Orchestrator and the end-user will be able to make data requests. The configuration requests would be received by the **Configuration Manager**, which would allow end users to define monitoring parameters, thresholds, and policies. It also supports role-based access control for configurations. Additionally, it would allow the Security Orchestrator to verify security requirements in terms of SSLAs. Thereby, the **Configuration Manager** may configure or reconfigure each monitoring tool in accordance with real-time security constraints to check if they are being fulfilled. The **Configuration Data Storage** module, as its name suggests, would store configuration information related to the monitoring tools as well as configuration requested by end users via the Security Orchestrator.

The **Data Collection Module** collects data from various sources, such as network devices, servers, applications, and security appliances. It would support multiple data types, protocols, and network segments. Data collected will be raw, not providing processing techniques. Information would be transmitted over the **Communication Bus**. It will share monitored data with two different modules. The first one, named **Near Real-time Data Retrieval**, offers the possibility of retrieving a small piece of data stored in the current communication bus to be consumed by external modules. Then, external modules may process and correlate the information located in the Communication Bus with further techniques. The second module is the **Data Aggregation and Normalization Module,** which is responsible for consuming and aggregating data from multiple sources and normalising it into a consistent format for processing. This ensures compatibility and simplifies downstream processing.

On the other hand, the **Data Correlation and Feature Extraction Module** will be related to these two last modules described, employing the raw or processed data to apply semantic techniques to understand the context, meaning, and significance of the monitored data. This involves interpreting the data beyond its raw form to derive higher-level insights. Besides, it generates new features or variables from the raw and correlated data that provide a greater wealth of information for prediction algorithms.

Once the data processing is performed, the **Long-term Data Storage** will be able to store collected and processed data in a scale and secure manner, supporting long-term storage for historical analysis that can be consumed by other modules. A **Data Storage GUI** may implement a visualiser to observe the monitored data regardless of the type of database. This module would be used by an end-user or a third party to supervise in a straightforward approach the previously monitored information and obtain an overview of system status.

The PMP offers its outcomes to be consumed by external modules via different modules. One of the modules was already described to expose raw data. Alternatively, the **Data Exporter** module will gather the processed information and enable the export of data and report to external systems or for offline analysis. Furthermore, it would support various formats (e.g., CSV, JSON, XML, YANG) to ensure a wide range of choices. Finally, the **Historical Data Retrieval** module allows retrieving large amounts of data or historical data storage in the long-term database.

A representation of the external modules has also been reflected in Figure 3-15, expressing several external modules that may consume the monitoring data to perform their work. Focusing specifically on the **Analysis Engine** and **Alert and Notification Module**, the functionalities regarding threat detection and subsequent alerting can be determined, since these external modules will be in charge of acting reactively on the system.

Starting from the **Analysis Engine** module, it can consume the monitored information from the **Near Real-time Data Retrieval**, the **Data Exporter,** or the **Historical Data Retrieval**. The data would be analysed through a rule-based system, determining potential threats and who is affected. In case a threat is detected, the **Analysis Engine** module will inform the **Alert and Notification Module,** which has the competencies to generate the alert and notify the other modules.

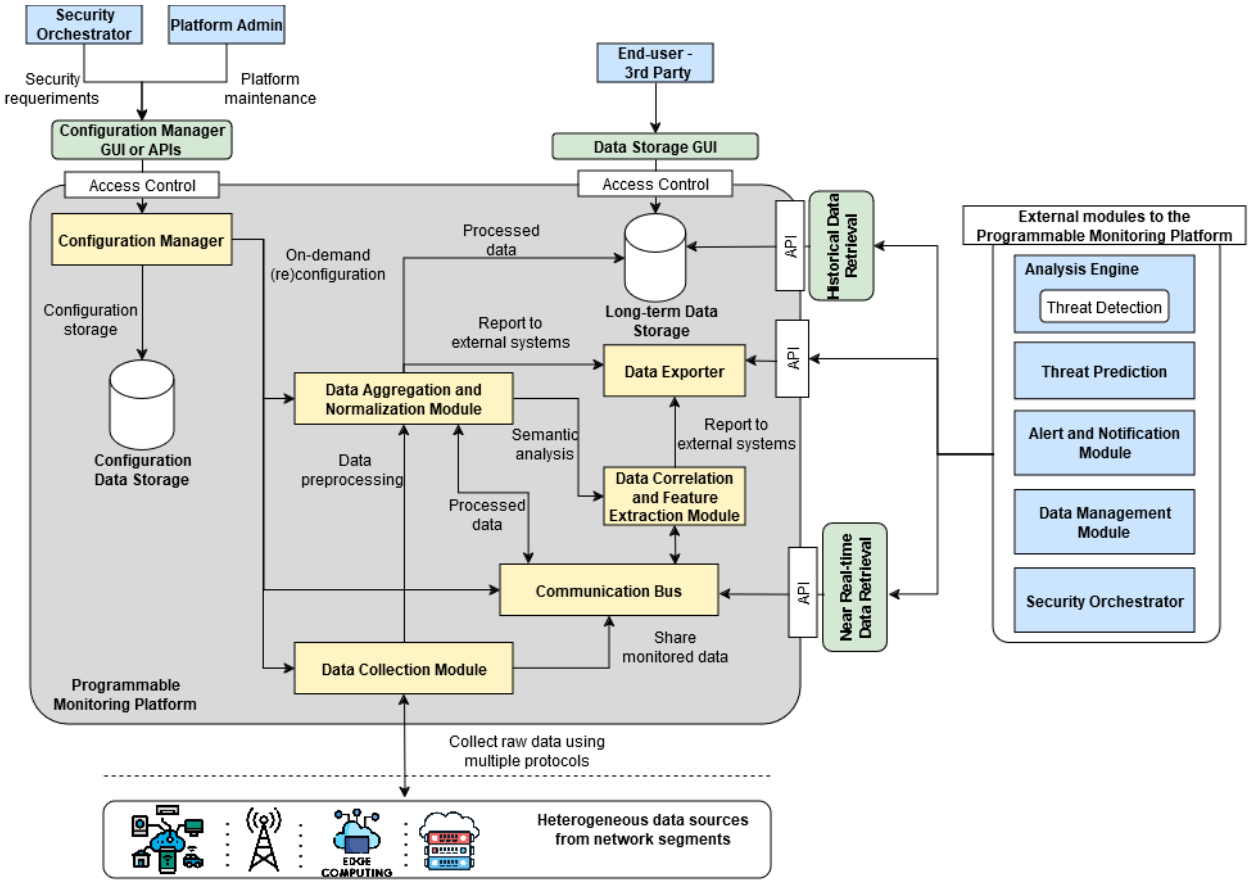

Figure 3-15: PMP architectural design

Regarding the choice of tools, different combinations will be made, which will be analysed and justified. The most realistic combinations and those closest to the initial software design will be shown as possible candidates, though it may vary during the next months. Additionally, the descriptions will have a reference table to improve their visualisation. The table will be sorted by activity type. A sub-activity type focused on the monitoring and communication fields will be specified, so this sub-activity will be placed in the table below its corresponding activity. In case a tool realises the same functionality in more than one combination, it shall be named but not justified, unless it performs additional or different work.

As shown in combination 1 column of Table 3-4, the monitoring activity has four sub-activities to cover the layers of Service, Infrastructure, and Network. Falco would work on the Service layer since it can collect logs events, and its DAST functionality provides security. Fluentd may also work as an event logs collector/event log unifier and can gather metrics from devices. It is oriented to environments with a huge computing power. Therefore, in other environments, such as Far Edge, a tool like Telegraf is more efficient in acquiring health metrics from these types of sources. Snort++ can be used as a good solution to extract network traffic due to its open feature to decapsulate GTP packages used in 5G/B5G/6G. An event streamer could use Filebeat to solve a problem with a real-time exportation that Snort++ has. Filebeat can transmit the network layer data to Kafka, which is responsible for communicating the information between the internal and external modules using its topics and a publisher/subscriber policy. Prometheus might be used as an aggregator and normalisation tool to send the information to the data exporter or other modules to correlate data. In addition, Prometheus can store the information internally as a buffer and export it to a long-term database. InfluxDB may serve as a database due to the fact that it is efficient and third-party tools can obtain information easily from it. The visualisation of information may be shown with Grafana dashboards, considering that they can represent data from a large number of third-party tools. On the other hand, Snort++ might realise the threat detection using the data monitored thanks to its IDS rule-based system and provide the alerts with Prometheus Alertmanager.

The combination 2 column starts with OpenTelemetry as the service layer log provider as an alternative to Falco, losing the securitization but enhancing the observability over event logs. Fluentbit is a lightweight version of Fluentd, so it presents the same functionality. The same tools will perform the monitoring of the Network layer, event streamers, pre-processing and aggregator, visualizer, threat detector, and alert system.

Databases would be changed to Grafana Mimir, which can manage data directly from Prometheus and expose it to Grafana for visualisation.

Finally, the combination 3 columns of Table 3-4 will maintain the event streamers, the pre-processing and aggregator, the database, the threat detector, and the monitoring tools from the first combination but reduce each layer to one unique tool. This combination is oriented to an InfluxData suit, using Telegraf, InfluxDB, Chronograf, and Kapacitor (TICK stack). Chronograf visualizes the information extracted from the InfluxDB database, generating simple alerts in case it is needed. The alert system is extended by Kapacitor, implementing stream and batch processing capabilities and supporting anomaly detection by means of scripts related to alert configurations.

Table 3-4 Combinations of tools for the PMP, threat detector, and alert system.

Type of activity		Combination 1	Combination 2	Combination 3
Monitoring	Service	Falco	OpenTelemetry	Falco
	Service /Infrastructure	Fluentd	-	-
	Infrastructure	Telegraf	Fluentbit	Telegraf
	Network	Snort++	Snort++	Snort++
Event streamer	Internal	Filebeat	Filebeat	Filebeat
	Internal/External	Kafka	Kafka	Kafka
Pre-processing and aggregation		Prometheus	Prometheus	Prometheus
Database		InfluxDB	Grafana Mimir	InfluxDB
Visualiser		Grafana	Grafana	Chronograf
Threat Detector		Snort++	Snort++	Snort++
Alert system		Prometheus Alertmanager	Prometheus Alertmanager	Kapacitor

3.4.2 Semantics-aware Anomaly Detection

Rethinking accuracy in remote estimation is crucial for efficient resource management and change detection, which covers anomaly detection—both of which are core objectives of Task 4.2.

Classical remote estimation systems have traditionally focused on accuracy, assuming that all data is equally important. In this setup, errors are measured by the difference between the true and estimated states, regardless of the data's significance. Nonetheless, as systems increasingly rely on massive sensory inputs and have limited processing power, this approach faces significant challenges. Consequently, it is crucial to consider whether accuracy is the only metric that can be adopted in estimation. Data can carry context, making some data points much more valuable. In anomaly detection, for example, identifying significant data at the right time is far more impactful than maximizing accuracy across all data.

The measurements may convey richer information beyond simple physical amplitudes, leading to the need to revisit the definition of estimation quality and incorporate data significance into system design. Our approach exploits the value of information through the history of system realizations to determine the optimal timing of transmission, thereby reducing the amount of uninformative data transmitted in the network.

In manufacturing systems, regarding an exemplary real-world context, it is essential to detect equipment failures or abnormalities without overwhelming the system with irrelevant data. Recognizing critical status updates in real-time is more valuable than continuously monitoring stable conditions in networked control of robot swarms. Both examples show the need to prioritize significant data over continuous accuracy.

3.4.2.1 Semantics of Information

The Semantics-aware estimation shifts the focus from just accuracy to the significance of the information. At first, a Markov model is considered to start with the most intuitive example, where the system alternates

between normal and alarm states. The types of errors addressed in the approach are missed alarms—where an alarm is overlooked—and false alarms—where the given system mistakenly signals an alert in a safe state. Missed alarms carry higher costs than false alarms because they indicate a failure to detect a potential issue. Additionally, the effect of consecutive errors can compound over time if unaddressed. Thereby two new metrics are introduced: the Age of Missed Alarm (AoMA) and the Age of False Alarm (AoFA) [LP24], each capturing the lasting impact of these errors.

Breaking down the components, a sensor will monitor the process, deciding whether to transmit a state in each time slot. Importantly, these actions do not take effect immediately, and the communication channel is subject to packet losses. The receiver then uses these transmissions to update its estimate and sends acknowledgements or rejections back to the sensor. By focusing on these semantic aspects, the transmission of critical data is prioritized, which helps improve both resource efficiency and system safety. This selective approach implies that instead of sending every piece of data, the clients concentrate on transmitting information that could indicate a developing anomaly or risk.

3.4.2.2 Optimal Policy Structure

The key contribution is to enhance threshold-based switching policies by introducing two new optimization metrics: Age of Missed Alarms (AoMA) and Age of False Alarms (AoFA), as illustrated in Figure 3-16. These metrics capture the temporal impact of estimation errors, considering both their immediate costs and their long-term consequences. By optimizing these metrics in addition to traditional error and communication trade-offs, we enable more sophisticated transmission strategies for systems that prioritize accurate state estimation, especially in scenarios where the timing of errors is crucial.

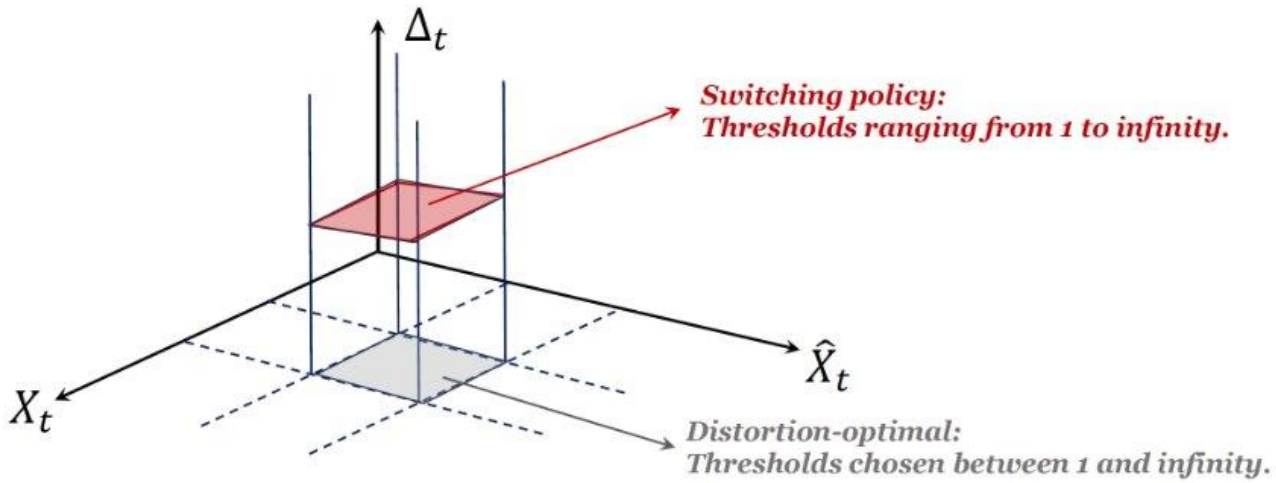

Figure 3-16 A schematic view of the semantics-aware switching policy in 3-dimensional space.

In practical terms, this model significantly reduces the frequency of transmissions while ensuring that critical data is still sent promptly. By aligning the policy with semantic significance, it is possible to achieve a more efficient and resilient system, directly supporting the goals of anomaly detection and context-aware control.

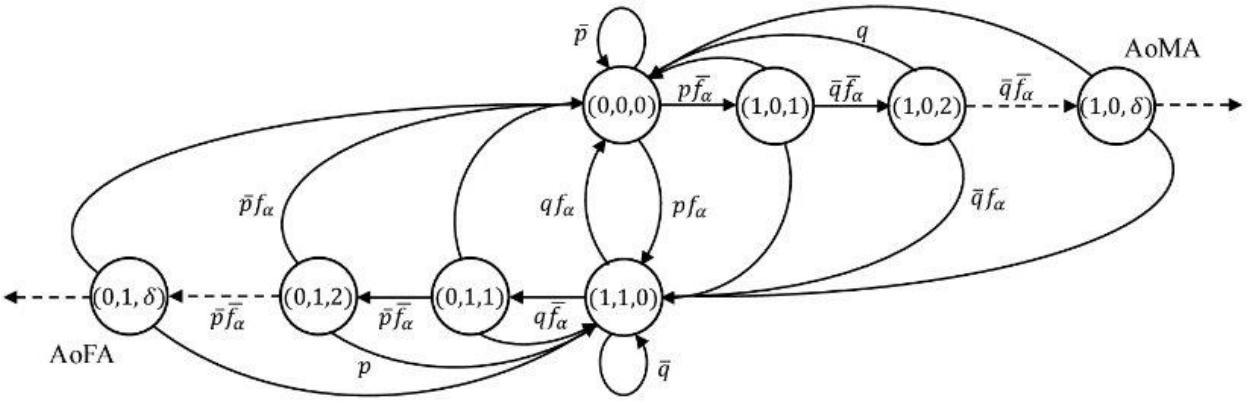

Figure 3-17 Discrete-time Markov chain representing the truncated decision process under a switching policy.

When the source is symmetric and the states are equally important, the optimal policy degenerates from a switching policy to a simple threshold policy. The approach can be further extended by covering multiple states in change detections, as described in Figure 3-17, eventually proposing a more generally applicable solution for semantics-aware estimation.

In summary, the semantics-aware estimation approach challenges traditional accuracy-focused methods by prioritizing data significance, which in turn supports efficient and timely anomaly detection. This shift will improve resource allocation and control capabilities in systems that need to process high volumes of sensory data effectively.

3.5 Incident Prediction and Continuous Mitigation

Due to the rapid increase in volume and complexity of cyber threats, traditional reactive measures that rely on human intervention are no longer effective. This has led to the rise of closed-loop security orchestration, a modern paradigm that integrates threat prediction, detection, mitigation, and response into a unified, automated process. The "closed-loop" aspect highlights the system's capability to adapt and improve its responses over time through a feedback mechanism. By incorporating advanced machine learning and deep learning models, the orchestrator can effectively predict threats based on historical network traffic patterns and emerging anomalies.

Once a threat is detected by the network detection system, pre-emptive actions such as traffic filtering, blocking the attacker, and isolating compromised network segments are carried out by the mitigation component of the security orchestrator. These automated detection and response capabilities not only accelerate threat mitigation but also reduce the likelihood of human error and response delays, thereby minimizing the impact of attacks and reducing downtime.

In this subsection, we discuss incident prediction and mitigation within the framework of a closed-loop Zero Touch Network and Service Management (ZSM) system. Various cybersecurity threats and intrusion detection datasets that can be utilized to develop the predictive and mitigation models for this work package have been reviewed. Threats and corresponding mitigation actions are defined using a mitigation matrix. To address the challenges of mitigation and prediction, we propose multi-label classification algorithms such as binary relevance, classification chains, and sequential generative models. The detailed processes—including dataset utilization, preprocessing, feature selection, and model development are outlined as follows.

3.5.1.1 Cyber Threat and Potential Mitigation by Closed-loop of Security Orchestrator

In this subsection, we define the scope of cyber threats and outline potential mitigation actions that can be executed by a closed-loop security orchestrator.

The security attacks to be mitigated by the closed-loop of security orchestration are Denial of Service (DoS), Distributed Denial of Service (DDoS), unauthorized access, reconnaissance, and cryptojacking. Table 3-5 illustrates these threats, their associated activities, and the relevant IDS datasets that can be used to develop mitigation and predictive models.

For mitigation, as shown in Table 3-6, the security orchestration mitigation actions are decomposed into several subtasks (M1 to M11), which can be combined to effectively address security threats. A value of 1 in the table indicates that the action will be taken when the threat is detected. In contrast, a value of 1* specifies that the action will only be taken when the threat's severity level is high.

Table 3-5: Summary of Threat and Dataset

Threat/Attack & Description	Attack Activities	Dataset
Reconnaissance • Attacks aimed at gathering information about the target network in preparation for a more significant attack.	• Port Scanning • Ping • Sweeping • Footprinting	CICIDS 2017, CICIDS 2018, UNSW-NB15, TON_IoT, etc.
Access Attack • Attacks where an attacker tries to gain unauthorized access to a system or network.	• Brute Force Attack • Password Cracking • Privilege Escalation	CICIDS 2017, CICIDS 2018, UNSW-NB15, TON_IoT, etc.

Denial of Service (DoS) and Distributed Denial of Service (DDoS)
• Attacks that overwhelm the network or service with traffic, making the system/services unavailable to legitimate users
	• Traffic flooding
	• SYN flood, Ping of death
	• Application layer attacks
	  ○ slowloris,
	  ○ http request flooding
		CICIDS 2017, CICIDS 2018, UNSW-NB15, TON_IoT, etc.

Cryptojacking • A cyberattack where a hacker secretly uses someone else's computing resources (like a computer, smartphone, server, IoT devices) to mine cryptocurrencies without the owner's consent	• Cryptocurrency mining	Cryptomining dataset [Man24]

Table 3-6 Attacks and Mitigation Actions by Security Orchestrator

Code	Mitigation of Attack	Crypto-jacking	Reconnaissance	Access	DoS / DDoS
M1	Isolating infected system	1		1	1
M2	Close all unused ports via the firewall		1		
M3	Segmentation/isolation of critical system (to separate VLAN)		1		1
M4	Block attacker (MAC)		1	1	1
M5	Blacklist and whitelist attacker (MAC)	1*	1*	1*	1*
M6	Enhance system security credential			1	
M7	Rate limiting and throttling to limit the incoming connection attempts		1		1
M8	Traffic filtering and scrubbing				1*
M9	Perform patching/update	1			
M10	Perform system reinstallation	1*			
M11	Notify network operator/provider	1*			1*
M12	Notify vendor	1*			1*
	1 – Mitigation on all Severity Level				
	1* – Mitigation on High Severity Level				

A brief description of each dataset is given as follows:

- **CICIDS2017 Dataset:** the Canadian Institute for Cybersecurity Intrusion Detection System 2017 dataset (CICIDS2017) [Int17] is created by the Canadian Institute for Cybersecurity (CIC), to offer a realistic depiction of contemporary network traffic, encompassing both benign and malicious activity. The dataset is designed to simulate real-world network attack scenarios. The implemented attacks include DoS, DDoS, Brute Force FTP, Brute Force SSH, Heartbleed, Web Attack, Infiltration and Botnet. The dataset includes packet capture (PCAP) files, as well as over 80 network flow features generated using CICFlowMeter [AppCic] stored in CSV files.
- **CICIDS2018 Dataset:** the CICIDS2018 [Ima18] is another comprehensive dataset created by the Canadian Institute for Cybersecurity (CIC) to emulate realistic network traffic and attack scenarios in a modern IT environment. It is widely utilized in cybersecurity research for benchmarking intrusion detection systems (IDS) and evaluating machine learning models in network security contexts. The dataset includes seven distinct types of attacks: Brute Force, Heartbleed, Botnet, DoS, DDoS, Web-based attacks, and internal network infiltration. The simulated environment features 50 attack machines targeting victim organizations comprising 5 departments with a total of 420 user machines and 30 servers. CICIDS2018 contains network traffic captures and system logs from each machine, with 80 features extracted from the traffic using the CICFlowMeter-V3 tool [AppCic].
- **UNSW-NB15 Dataset:** it is a network intrusion detection dataset created by the Australian Centre for Cyber Security (ACCS) at UNSW Canberra in 2015 [Mou15]. The dataset was designed to address the limitations of older datasets like KDD99 by providing more modern, realistic network traffic and attack scenarios. The dataset contains nine types of attacks, namely, Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode and Worms.
- **ToN-IoT Dataset:** it is a benchmark dataset designed to reflect modern and realistic IoT cyberattack scenarios [MouToN]. The dataset was generated using a comprehensive testbed built on a multi-layer architecture, including Edge, Fog, and Cloud components. The infrastructure was implemented using technologies like NSX-VMware, Kali Linux, Node-RED Server, Hive-MQTT broker, and cloud services. The dataset includes nine types of cyberattacks: backdoor, DoS, DDoS, injection, Man-In-The-Middle (MITM), password cracking, ransomware, scanning, and cross-site scripting (XSS). In total, 43 features were extracted from the network traffic data. Additionally, two extra features, 'label'

and 'type,' were added: the 'label' feature is used for binary classification (normal or attack), while the 'type' feature supports multi-class classification for identifying specific attack categories.

- **Cryptomining data set [Man24]:** The cryptomining data set [Man24] was created by Mannella et al. (2024) to be used to build cryptojacking detection module for *IoT Proxy* – a solution to enhance the security of resource-constraint IoT devices. The dataset consists of XMR-Stacks plain and encrypted traffic, MadoMiner Startum connections, and Coinhive mining sessions. The test case consists of a PC and a Rasberry Pi where PCs are used to mine Monero coin via Coinhive using a variety of browsers. The system is intentionally infected by Madominer. XMR-Stack are executed on the Raspberry Pi to gather some Stratum traffic. Wireshark are used to capture and analyze network traffic to produce PCAP files. The dataset is available on Kaggle[2].

3.5.1.2 Features of Mitigation and Predictive Models

In the initial phase where the security orchestrator is still under development, datasets from the system, CICIDS2017, CICIDS2018, UNSW-NB15, ToN-IoT, and Cryptomining datasets will be used to build the mitigation and predictive models. Once the test cases are prepared, the corresponding datasets will be leveraged to retrain the model, improving accuracy. To ensure compatibility, the network traffic should be in a network format similar to the features generated by CICFlowMeter shown in Table 6-4 of Subsection 6.2. Each sample will be appended with threat detection results along with a brief description of the identified threat. To detect anomalies in IoT devices, telemetry data (shown in Table 6-5 of Subsection 6.3) is required. The features from Table 6-4 and Table 6-5 are to be combined to form a hybrid dataset which will be used to build the prediction and mitigation models.
In addition to the above, multiple binary labels representing mitigation actions (as described in Table 3-6) will be incorporated into the dataset to support the development of threat mitigation and prediction models.

For the predictive dataset, data must be further processed and consolidated to ensure a fixed number of records for each interval. Threats occurring in the subsequent interval will be recorded using binary labels to indicate potential threats expected in the next period.

3.5.1.3 Architecture of Mitigation and Prediction Models

Once a threat is detected through the threat detection process, a mitigation model is deployed to determine the appropriate action(s) for execution by the security orchestrator. Since multiple mitigation steps may be necessary for a single threat, mitigation labels must be added to the dataset. Similarly, for predictive algorithms where multiple threats might be anticipated within a given period, multi-label classification models, such as Binary Relevance (BR) [Bou04], Classification Chain (CC) [Rea11], and Sequential Generation Model (SGM) [Yan18], will be used to identify the next course of action. The methodology of mitigation and predictive models is illustrated in Figure 3-18 and Figure 3-19, respectively.

[2] https://www.kaggle.com/datasets/danielecanavese/cryptomining-data-set/

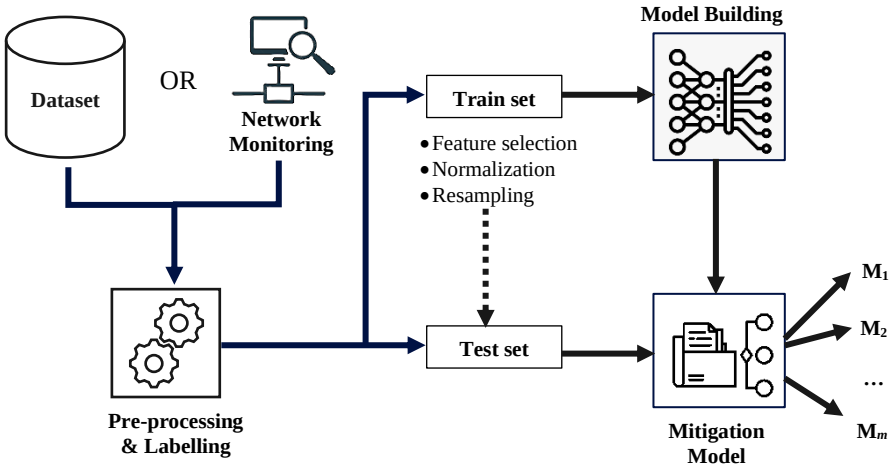

Figure 3-18: Threat Mitigation Process Flow.

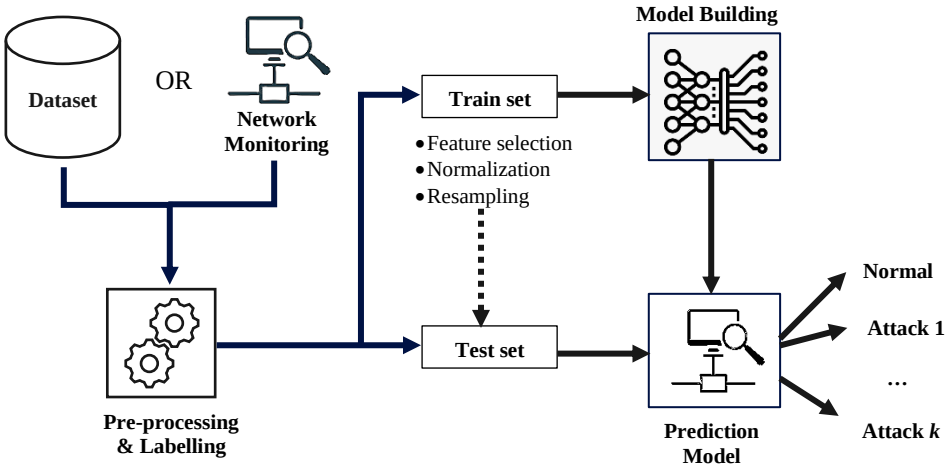

Figure 3-19: Threat Prediction Process Flow

3.5.1.3.1 Preprocessing and Labelling

In this phase, the data will first be cleaned to address any missing or erroneous values. Following this, labelling will be applied. For the mitigation model dataset, multiple binary labels (M1 to M11), as outlined in Table 3-6, will represent specific mitigation actions. A label of 1 indicates that the action is required, while 0 signifies it is not needed.

For the predictive model dataset, further processing and consolidation will ensure a consistent number of records per interval. Threat presence in the upcoming interval will be annotated using binary labels, indicating the likelihood of potential threats in the next period.

Finally, the dataset will be split using stratified sampling, dividing it into an 80-20 split for training and testing, ensuring balanced representation across all classes.

3.5.1.3.2 Feature selection, Normalization and Resampling

On the training dataset, data normalization and encoding of categorical variables will first be applied. Next, data transformation and feature selection or extraction will be performed to improve the discriminative power of the features. If the dataset is imbalanced, resampling techniques may also be used to achieve balanced representation across classes.

The Min-Max normalization [Hen21] that transforms feature values in a range of [a, b] is

$$x' = a + \frac{(x - \min(x))(b - a)}{\max(x) - \min(x)}$$

Several feature selection algorithms will be deployed for the feature selection process. One of them is the Boruta feature selection algorithm [Kur10]. Boruta is a powerful, non-parametric feature selection method that works well with high-dimensional datasets and helps remove irrelevant or redundant features. It is an extension of the random forest algorithm and works by comparing the importance of original features to the importance of randomly shuffled (shadow) features.

Suppose $f_1, f_2, \cdots, f_n$ and $s_1, s_2, \cdots, s_n$ are original features and corresponding shadow features (permuted versions of the original features), respectively.

The feature importance: Mean Decrease Impurity (MDI) has expression

$$I(f_i) = \sum_{t \in T} \frac{\Delta G_t(f_i)}{|T|}$$

where $I(f_i)$ is the importance of feature f_i and $\Delta G_t(f_i)$ is the decrease in impurity (such as Gini impurity) for feature f_i in decision tree t in the random forest with T trees.

The importance of original feature f_i is decided as follows:

1. If $I(f_i) > \max(I(s_i))$ (where $\max(I(s_i))$ the importance of the most important shadow feature s_i), then feature f_i is deemed relevant.
2. If $I(f_i) < \min(I(s_i))$, then feature f_i is irrelevant.
3. If $\min(I(s_i)) \leq I(f_i) \leq \max(I(s_i))$, the feature is classified as tentative.

In machine learning, imbalanced datasets arise when the classes in a classification problem are not equally represented, with one class having significantly more samples than the other. This class imbalance can severely impact the performance of many machine learning models, particularly those that are sensitive to class distribution, such as decision trees, logistic regression, and neural networks. To address this issue, oversampling can be applied to the minority class to increase its representation, while under-sampling can be used on the majority class to reduce its dominance, thereby balancing the class distribution and improving the model's ability to learn from both classes.

The Synthetic Minority Over-sampling Technique (SMOTE) [Cha02] is the oversampling technique that generates synthetic data points for the minority class by interpolating between existing instances. Suppose x_1 and x_2 are two instances of a minority class, the new synthetic instance x_{new} can be generated as:

$$x_{new} = x_1 + \lambda(x_2 - x_1)$$

where λ is a random number in a range of 0 and 1.

In the down-sampling of the majority class, the Tomek Link (T-Link) [Tom76] technique will be deployed. Tomek Link not only reduces data points of the majority class, it also removes borderline or noisy samples.

Tomek Link Algorithm

1. Input: A dataset $E = \{(x_1, y_1), (x_2, y_2), \cdots, (x_N, y_N)\}$ where $x_i \in D \subseteq R^m$ and $y_i \in \{C_0, C_1\}$ and C_0 is the majority class.
2. For each $x_i \in D$ $(where\ D$ is the domain of $E)$:
   - Find its nearest neighbour $x_{nn(i)} = \arg\min_{x_j \in D} d(x_i, x_j)$ for $i \neq j$ and

$$d(x_i, x_j) = \sqrt{\sum_{k=1}^{m} \left(x_i^{(k)} - x_j^{(k)}\right)^2}$$

   - If $y_i \neq y_{nn(i)}$, i.e. they belong to different classes, then the pair $(x_i, x_{nn(i)})$ is a Tomek Link.
3. Output: A new dataset E' after removing instances from the majority class C_0 in each Tomek Link pair.

The encoding process, normalization settings and selected features from the feature selection process will be applied to the testing dataset to ensure consistent input for the mitigation and predictive models.

3.5.1.3.3 Mitigation and Predictive Model

Multi-label classification techniques like binary relevance (BR), classification chain (CC), and sequential generation model (SGM) will be deployed to build the mitigation model. The model will be tested and evaluated on the testing set of the IDS dataset.

Binary Relevance (BR)

Binary Relevance (BR) [Bou04] treats each label as an independent binary classification task, training a separate classifier for each label using a shared feature set. In this context, individual classifiers are used to determine the applicability of each mitigation action. As shown in Figure 3-20 and Figure 3-21, the BR-based mitigation process can be implemented either with or without incorporating threat detection features.

In the first approach, incorporating threat detection features can enhance the classification accuracy of the mitigation model, as the mitigations are closely correlated with specific threats. Conversely, training classifiers without threat labels may slightly reduce classification accuracy but can potentially increase the model's robustness. Without relying on explicit threat labels, the classifier may still respond effectively to previously unseen threats, leading to a more adaptable and resilient mitigation model.

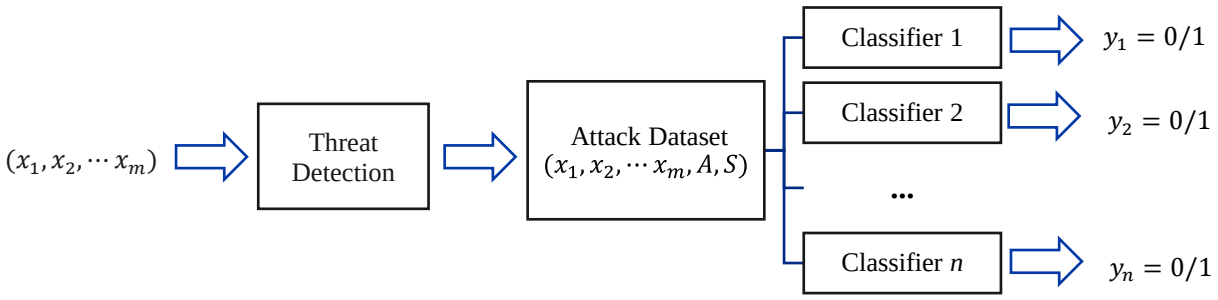

Figure 3-20 Threat Mitigation / Prediction using Binary Relevance with Threat Detection.

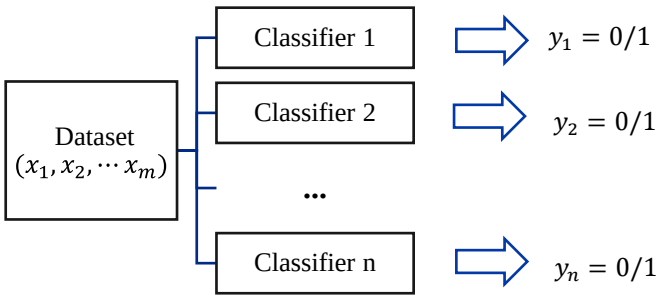

Figure 3-21 Threat Mitigation / Prediction using Binary Relevance without Threat Detection.

Classification Chains

One of the limitations of binary relevance is it does not consider relationships or dependencies among labels, treating each label as an independent binary classification task. In many real-world applications, such as text categorization or threat detection, labels are often correlated (e.g., "spam" and "phishing" might occur together). Ignoring these dependencies can reduce the model's accuracy and fail to capture important contextual information. The classification chains (CC) method [Rea11] is thus proposed to solve BR issues. Classification chains involve training a sequence of binary classifiers, one for each label, where each classifier uses the original features and predictions from all previous classifiers in the chain (Figure 3-22). This approach captures label dependencies effectively, allowing for more accurate prediction.

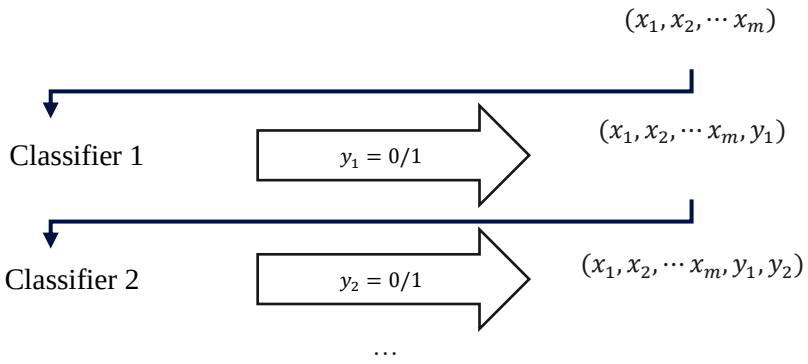

Figure 3-22: Threat Mitigation / Prediction using Classification Chains.

Supervised learning algorithms such as decision tree, random forest (RF), and artificial neural networks (ANN) may be employed to perform the classification.

Classification using Decision Tree

A decision tree is a commonly used algorithm for classification and regression tasks. It works by recursively splitting the dataset based on certain conditions or features to create a tree-like structure of decisions. The goal is to partition the data into subsets that are as homogeneous as possible with respect to the target variable. A decision tree consists of nodes, edges/branches and leaves. Nodes represent decisions or tasks based on feature values. Edges/branches represent the outcomes of the decision or test and leaves represent the predicted label (in classification) or value (in regression). The commonly used splitting criteria of decision tree algorithm on classification tasks are Gini impurity and entropy (information gain).

The Gini Impurity is a measure of how often a randomly chosen element would be incorrectly classified if it were randomly labelled according to the distribution of labels in the dataset. The Gini Impurity of a dataset *S* with *n* classes is defined as

$$Gini(S) = 1 - \sum_{i=1}^{n} p_i^2$$

Where p_i is the probability of class *i* in *S*.

Entropy on the other hand measures the impurity based on the concept of information theory to quantify the uncertainty or disorder in the data. The entropy of a dataset S with *n* classes is:

$$Entropy(S) = -\sum_{i=1}^{n} p_i \log_2 p_i$$

Where p_i is the probability of class *i* in *S*.

Information gain (*IG*) when feature f_i is used as a split expression

$$IG(S, f_i) = Entropy(S) - \sum_v \frac{|S_v|}{|S|} \times Entropy(S_v)$$

Where S_v is the subset of S with a feature value f_i equal to v.

Classification using Random Forest Algorithm

The Random Forest algorithm is a powerful ensemble learning method primarily used for classification and regression tasks in machine learning. It builds multiple decision trees and combines them to achieve more accurate and stable predictions.

The process starts with bootstrap sampling, where multiple samples are created from the dataset by randomly selecting data points with replacement. Each sample is then used to build an individual decision tree. This approach introduces diversity among the trees, making the ensemble more robust.

Random feature selection is applied at each split within each tree, where a random subset of features is chosen to determine the best split. This further reduces correlations between the trees and improves overall model performance. Each decision tree is typically grown to its maximum depth, enabling it to capture complex patterns within its subset of the data.

In classification tasks, the final prediction is determined by a majority vote across the predictions of individual trees, making Random Forest a reliable choice for accurate and stable classification results.

$$y_{RF}(x) = \text{mode}(\hat{y}_1(x), \hat{y}_2(x), \cdots, \hat{y}_T(x))$$

Where $\hat{y}_i(x)$denotes the prediction of tree-i.

Binary relevance (BR) and classifier chains are required to train different classifiers for each label. The methods are therefore computationally intensive for high-order labels. As such, the Sequence Generation Model (SGM) is proposed to address the issue. Inspired by Yang et al. (2018) [Yan18] where the model has successfully applied to text sequence generation, we use it to predict threat mitigations/prediction on the multivariate dataset generated from the security orchestrator.

Sequential Generation Model (SGM)

Let the label space be $L = \{0,1\}$, a data $\boldsymbol{x}$ contains m features and N samples are assigned to $\boldsymbol{y}$ containing n labels in space L. The objective of SGM is to find an optimal label sequence $\boldsymbol{y}^*$ that maximized the conditional probability $p(\boldsymbol{y}|\boldsymbol{x})$ as follows:

$$p(\boldsymbol{y}|\boldsymbol{x}) = \prod_{i=1}^{n} p(y_i | y_1, y_2, \cdots, y_{i-1}, \boldsymbol{x})$$

The overview of the proposed model architecture is shown in Figure 3-23. First, the label sequence of each sample is sort according to the frequency of the labels in the training set. High-frequency labels are placed in the front.

Encoder:

In the encoder process, bidirectional LSTM by Hochreiter (1997) [Hoc97] are used to read data $\boldsymbol{x}$ from both directions and compute the hidden state of each mitigation/attack

$$\vec{\boldsymbol{h}}_i = \overrightarrow{LSTM}(\vec{\boldsymbol{h}}_{i-1}, \boldsymbol{x}_i)$$

$$\overleftarrow{\boldsymbol{h}}_i = \overleftarrow{LSTM}(\overleftarrow{\boldsymbol{h}}_{i+1}, \boldsymbol{x}_i)$$

The hidden representation of the *i*-mitigation is obtained by concatenating hidden states from both directions, $\boldsymbol{h}_i = [\vec{\boldsymbol{h}}_i; \overleftarrow{\boldsymbol{h}}_i]$.

The attention mechanism assigns the weight α_{ti} to the *i*-label at time-step *t* as follows:

$$e_{ti} = \boldsymbol{v}_a^T \tanh(\boldsymbol{W}_a s_t + \boldsymbol{U}_a \boldsymbol{h}_i)$$

$$\alpha_{ti} = \frac{\exp(e_{ti})}{\sum_{j=1}^{m} \exp(e_{ti})}$$

where $\boldsymbol{W}_a, \boldsymbol{U}_a, \boldsymbol{v}_a$ are weight parameters and s_t is the current hidden state of the decoder at time-step *t*. The final context vector $\boldsymbol{c}_t$ which passed to the decoder at time-step t is calculated as follows

$$c_t = \sum_{i=1}^{m} \alpha_{ti} \boldsymbol{h}_i$$

Decoder:

The hidden state s_t of the decoder at time-step *t* is computed as follows

$$s_t = LSTM(s_{t-1}, [g(\hat{\boldsymbol{y}}_{t-1}); \boldsymbol{c}_{t-1}])$$

where $[g(\hat{\boldsymbol{y}}_{t-1}); \boldsymbol{c}_{t-1}]$ means the concatenation of vectors $g(\hat{\boldsymbol{y}}_{t-1})$ and $\boldsymbol{c}_{t-1}$. $g(\hat{\boldsymbol{y}}_{t-1})$ is the embedding of the label which has the highest probability under the distribution $\hat{\boldsymbol{y}}_{t-1}$. The probability distribution $\hat{\boldsymbol{y}}_t$ over the label space *L* at time-step *t* and is obtained by

$$\boldsymbol{o}_t = \boldsymbol{W}_o f(\boldsymbol{W}_d \boldsymbol{s}_t + \boldsymbol{V}_d \boldsymbol{c}_t)$$

$$\hat{\boldsymbol{y}}_t = softmax(\boldsymbol{o}_t)$$

where $\boldsymbol{W}_o, \boldsymbol{W}_d$ and $\boldsymbol{V}_d$ are weight parameters.

The loss function is the binary cross-entropy loss function

$$Loss = -\frac{1}{N} \sum_{j=1}^{N} \sum_{i=1}^{m} \left[y_i(j) \cdot \log_2\left(\hat{y}_i(j)\right) + (1 - y_i(j)) \cdot \log_2(1 - \hat{y}_i(j)) \right]$$

where $y_i \in \{0,1\}$ and $\hat{y}_i$ are the actual and predicted label vectors of each sample

$(j = 1,2, \cdots N)$, respectively.

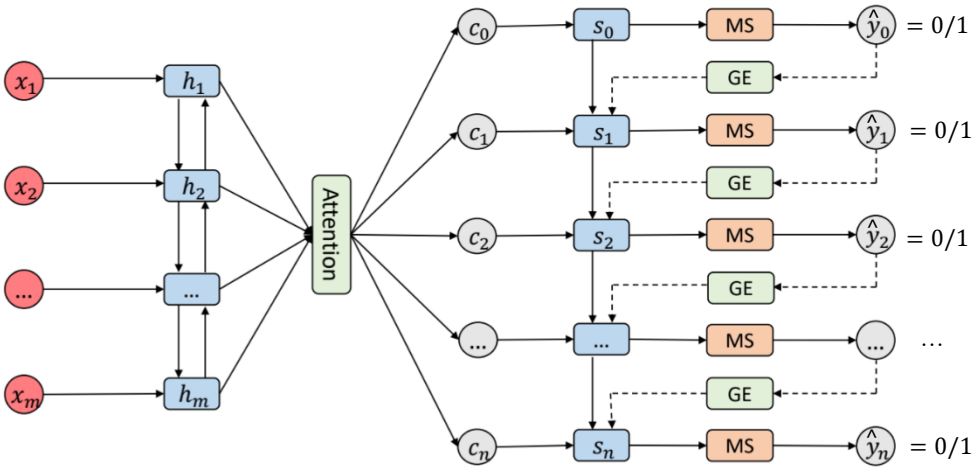

Figure 3-23: The overview of SGM model for threat mitigation. MS denotes the masked softmax layer. GE denotes the global embedding.

Evaluation Matrix

The evaluation metrics used in the empirical study include accuracy, recall, precision, and F1-score. These metrics are defined and interpreted as follows:

1. Accuracy

   Accuracy measures the overall correctness of the model by evaluating the proportion of correctly classified instances (both malicious and benign) to the total instances.

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + FN + TN}$$

2. Recall
   Recall assesses the model's ability to correctly identify malicious attacks (true positives) out of all actual malicious instances.

$$\text{Recall} = \frac{TP}{TP + FN}$$

3. Precision
   Precision measures the proportion of true positive predictions (accurately identified malicious instances) out of all instances predicted as malicious.

$$\text{Precision} = \frac{TP}{TP + FP}$$

4. F1-Score
   The F1-score is the harmonic mean of recall and precision, balancing the trade-off between them. It is particularly useful when the class distribution is imbalanced.

$$\text{F1} - \text{Score} = \frac{2 \times \text{Recall} \times \text{Precision}}{\text{Recall} + \text{Precision}}$$

The definitions of *TP*, *TN*, *FP*, and *FN* are shown in Table 3-7 are illustrated as follows:

True Positive (*TP*): The number of malicious data instances correctly classified as malicious.

True Negative (*TN*): The number of benign data instances correctly classified as benign.

False Positive (*FP*): The number of benign data instances incorrectly classified as malicious.

False Negative (*FN*): The number of malicious data instances incorrectly classified as benign.

These metrics collectively provide a comprehensive understanding of the model's performance in detecting and mitigating network attacks.

Table 3-7: The confusion matrix of TP, TN, FP and FN

		Actual Class: Attack	Actual Class: Normal
Predicted Class	Attack	*TP* (True Positive)	*FP* (False Positive)
	Normal	*FN* (False Negative)	*TN* (True Negative)

3.6 Security Closed-Loop modelling and management

As discussed in Section 2.3, the CL is not a novel concept and it is actually applied for a number of automation tasks, event in mobile network systems. While the idea behind the CL is basically the same, the way the loop can be implemented is actually evolving, due to the growing need for automation requested nowadays, not only by Telco Operators.

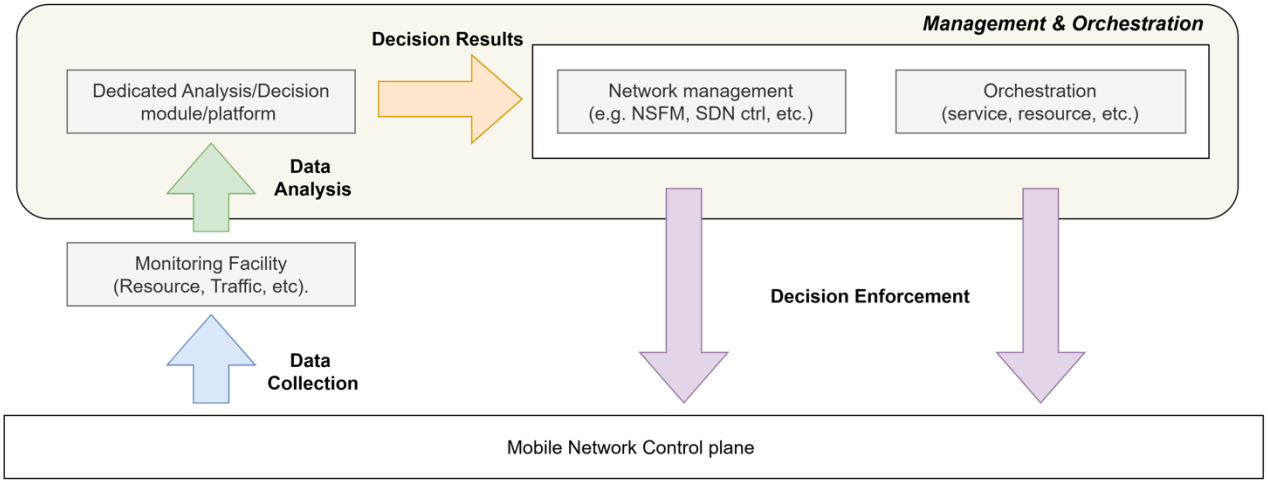

Figure 3-24: CL built with existing Management, Orchestration and Control entities

A CL can be implemented in a full static manner, i.e., with the different stages pre-deployed and even pre-configured to automatize specific tasks. In the simplest case, such loops re-use the existing M&O components to build some stages while the others are implemented by specific modules. In Figure 3-24 is shown as an example of a static CLs.

The Monitoring stage is implemented by exploiting the existing monitoring facility, i.e., not dedicated to the loop, the Analysis and Decision can be implemented with a dedicated module placed in the M&O, while the Execution is provided by the orchestrator/controller that performs the corrective actions by enforcing specific configuration on the Managed Entity (ME), i.e., the entity targeted by the automation task. This type of "built-in" loop can be definitely effective for specific tasks, but since the stages are static components, the scalability of the automation mechanism is limited and the degree of flexibility mainly depends on the configurability of different stages at runtime. Since the loops are in this case, mostly pre-provisioned, Governance and Coordination are usually simple or could even be not required at all.

A concrete example of CL built in the Management, Orchestration, and Control architecture is provided is shown in Figure 3-25, where is depicted the automation concept in H2020 SLICENET [SLICENET20].

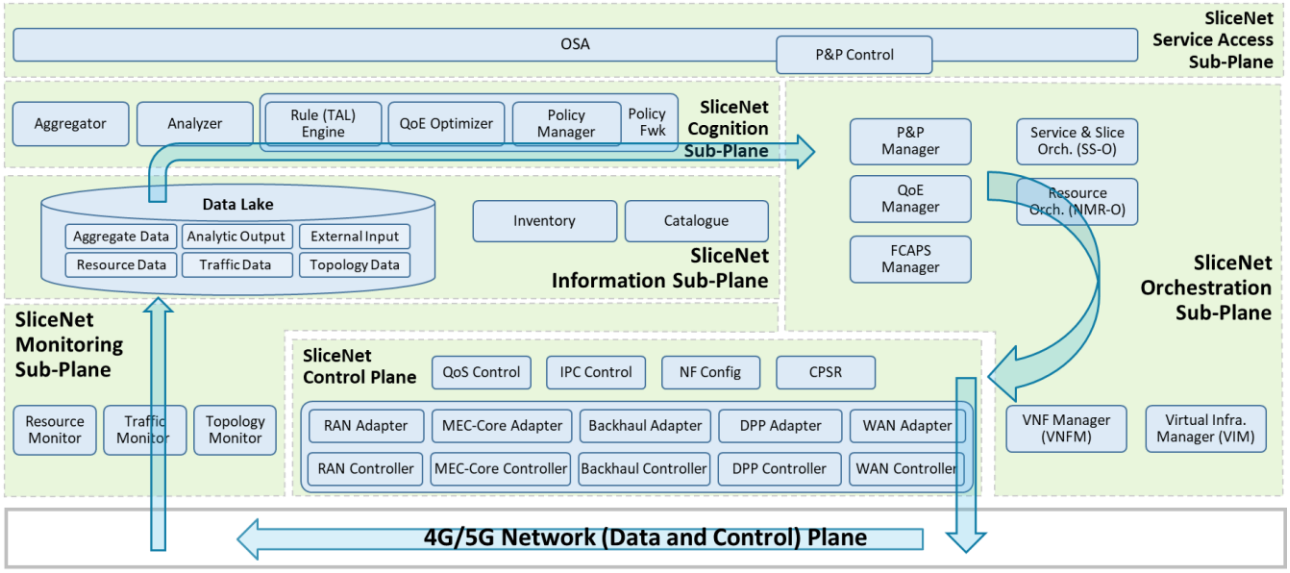

Figure 3-25: CLs in H2020 SliceNet realised with modules belonging to management and control ecosystem [SLICENET20]

The data are collected by several monitoring components and stored in a Data Lake. Specialized components belonging to a dedicated Cognition Sub-Plane perform analysis and take decisions that are finally enforced by the components in the Orchestration and Control sub-planes.

The more modern, flexible, and in line with the needs of automation of 6G systems way to implement a CL, is to consider it as a cloud application, where all the stages are services or micro-services of such an application, completely configurable and orchestrable. This would improve:

- Flexibility: CL tuned and configured to specific target ME. The stages can differ based on the type of loop, e.g., in an AI-driven CL, the analysis stage can be a specific AI/ML model trained for the given ME. The ME can be therefore selected at runtime and can be anything whose control can be automatized, e.g., services, compute resources, network slices, etc.
- Dynamicity: CL deployable close to the ME and/or close to the data to be monitored. Scalable and movable to other locations if required.
- Controllability: As a dynamic SW, a CL can be easily managed at runtime, although this requires specific CL management functions such as CL Governance and CL Coordination.

In Figure 3-26 is shown an example of CL as a cloud application, assuming an orchestration architecture with multiple capabilities.

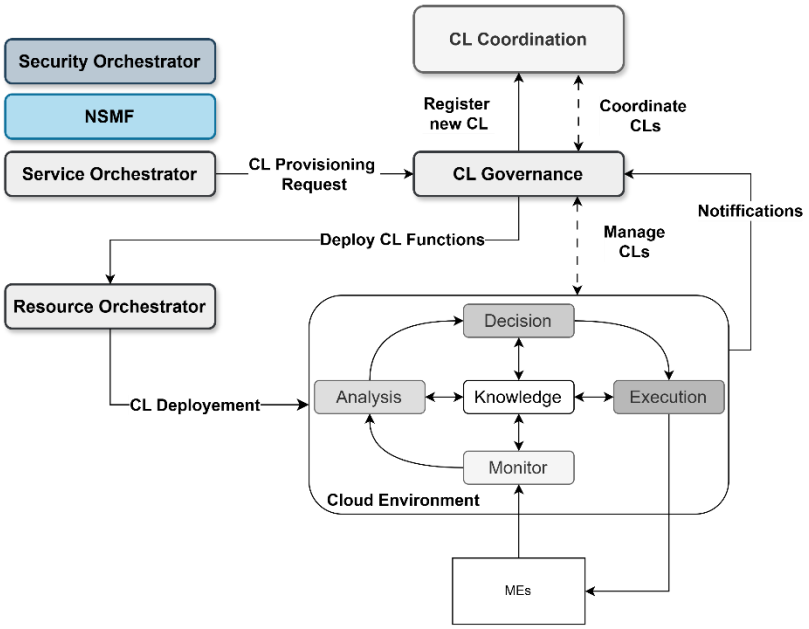

Figure 3-26: CL deployed as a cloud application

The loop is requested by one of the orchestration actors, e.g., service, NSMF (Network Slice Management Function), security, etc., by querying the CL Governance that selects the most suitable set of stages for the given request. Here the CL Governance is also assumed as in charge of the lifecycle management of the CLs. At this point, the CL Governance i) requests the Resource Orchestrator to deploy the stages and ii) registers the CL to the CL Coordination in order to make it aware of the existence of a new CL to be coordinated. The stages are then deployed to the cloud environment (public cloud, edge, far/extreme edge) that better fits the service requirements.

ROBUST-6G security CLs use a hybrid approach where the Monitoring stage is realized through the Programmable Monitoring Platform described in Section 3.4, while Analysis, Decision, and Execution are dynamically deployed. The PMP should offer a high degree of configurability in data selection and manipulation, suitable for the needs of a security CL. In the case in which an additional level of data manipulation would be required for the effectiveness of the Analysis, and additional Monitoring stage can be deployed to consume data from the PMP and prepare it for the Analysis. The Monitoring stage, along with Analysis, and Decision provides the security characterization of the CL, while the Execution stage is mainly an entity that translates the decision in requests towards the Security Orchestrator or sends notification to a human operator, as clarified later on. In particular:

- **Monitoring.** The PMP is properly configured to collect parameters related to the level of security agreed upon and the threat/anomaly the security service aims to avoid/mitigate.
- **Analysis.** It is an AI-driven application, trained for detecting (reactive loop) or predicting (predictive loop) anomalies and threats. In the case of predictive loops, the AI/ML techniques used are the ones described in Section 3.5. For detection, DL techniques such as Long short-term memory (LSTM), Autoencoders (AE), and Generative adversarial networks (GAN) are well-suited to the context due to the complexity and amount of data that must be processed [AAF+24]. It is important to highlight that the detection could not necessarily be AI-based and implemented with the use of proper rules.
- **Decision.** It is usually based on policies and rules for selecting the proper mitigation plan for a given threat/anomaly detected or predicted. The execution of such a plan can be directly requested to the Security Orchestrator, in case of zero-touch automation or communicated to a human operator in case of supervised security decision.

The possibility of performing human supervised decision is an important aspect of the ROBUST-6G security automation, especially when the actuation decisions are taken by an AI-based process: potential inaccurate decisions adopted in critical tasks may cause harm to the system (and people). EU commission debates the topic in the AI-Act [EUREG2024/1689], where it is introduced the key concept of “Human Oversight”: a human operator should be able to prevent and/or minimize risk given by the usage of AI in critical tasks, while the AI-based systems must provide dedicated human/machine interfaces to perform such an oversight.

For that reason, although the demonstration of a full (zero-touch) security automation is one of the main objectives of ROBUST-6G, the S-CL instantiated by the Zero-Touch Security Platform will be always configurable to include the human in the loop. Below a set of figures explains how the S-CL is instantiated exploiting the Zero-Touch Security Platform and how the automation is implemented. The functional architecture discussed in this document is used for the sake of simplicity.

Figure 3-27 shows an example of S-CL deployment into an IoT (Edge/Far-Edge) environment exploiting the functionalities of the Zero-Touch Security Platform. The S-CL is created and deployed in the target environment together with the security service: in this case, it is considered one of the scenarios of UC2, where an attacker penetrates and exploits the IoT devices (smart lamps) to mine cryptocurrency (Cryptojacking attack). A more detailed discussion on the UC is reported in Deliverable D6.1 [R6G24-D61]. The sequence is similar to the one exposed in Figure 3-26, where the fully orchestrable CL is presented. In this specific case, the PMP is programmed to collect network traffic data and resource consumptions from IoT devices, while Analysis, Decision, and Execution are deployed at the Edge of the network. The Analysis stage is a specific AI/ML application properly trained for the scenario by the AI/ML management while Decision and Execution are containers that can be requested by the S-RO.

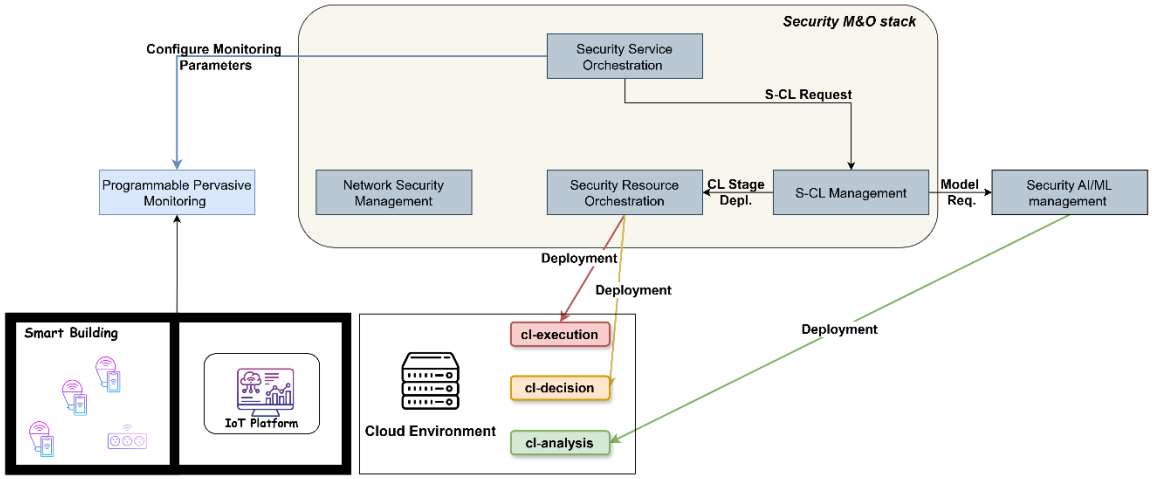

Figure 3-27: Example of S-CL Deployment in ROBUST-6G

Once the Analysis infers the anomaly, the Decision shall decide if and how to react, by selecting a proper mitigation plan to be executed. At this point, the Execution can act in two ways, depending on whether the S-CL is configured for full automation or for supervised decision. In Figure 3-28 is shown in the case of full automation (zero-touch).

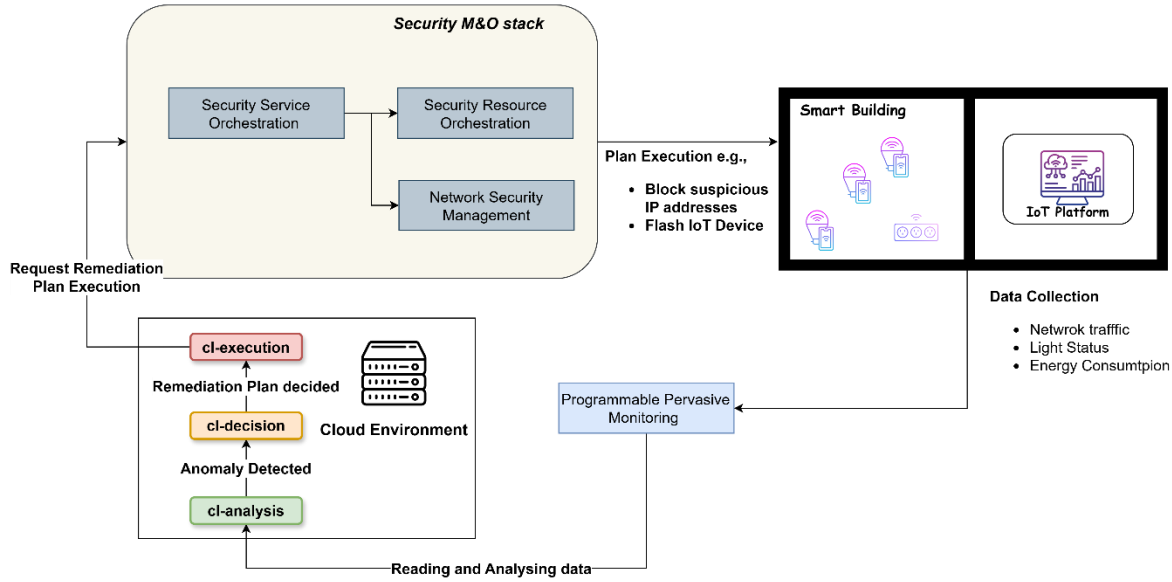

Figure 3-28: Automation in ROBUST-6G (UC2 scenario)

In this case, the execution translated the Decision's output in one or more request for the interface of the S-SO, which execute the remediation plan proposed by the Decision, e.g., block suspicious IP addresses, flash the firmware of the IoT device, provision a fresh and updated version of the IoT platform, etc.

In the case of supervised decision, the Execution notifies the human operator with a proper message whose content may affect the human intervention. In the case the message would contain only a minimum set of information e.g., the anomaly detected, when and where, the countermeasure would be fully on the hands of the operator while, in the case the message contains the mitigation plan selected by the Decision, the operator may check and decide to apply it simply by requesting its execution towards the S-SO interface. An example is shown of Figure 3-29, where the SysAdmin is notified through a dedicate interface (Security Capability Exposure) by the Execution stage.

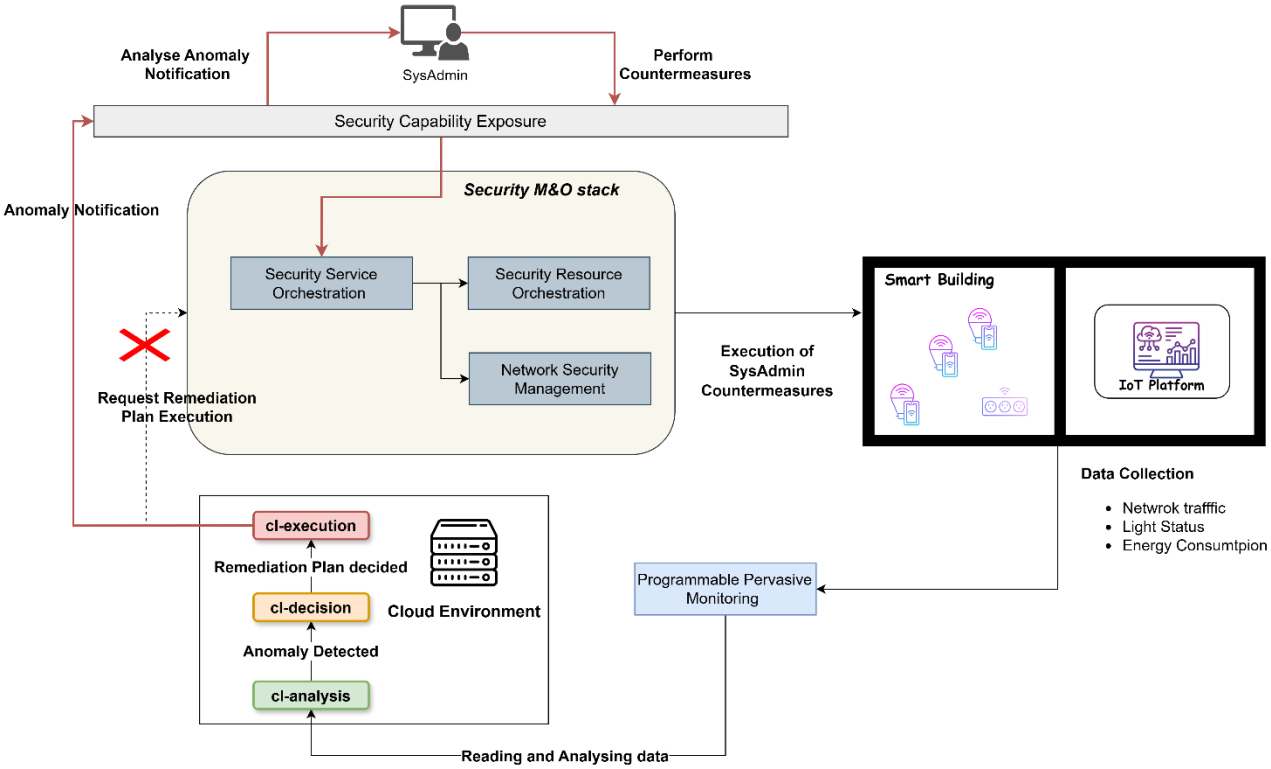

Figure 3-29: Automation in ROBUST-6G with Human in the loo

4 Conclusions

The document covered several aspects of zero-touch security management. Initially, the state of the art concerning the Security Orchestration and Automation through Security Closed-Loops has been investigated, as core concepts of the ROBUST-6G Zero-Touch Security Platform. The document discussed the role and the functionalities of a Security Orchestrator, the types of security orchestration (proactive, reactive, and predictive) and how the Security Orchestrator can model the security system and the related workflows by exploiting the OnSOAP architecture. Security Automation proposed by NIST has been compared with the ETSI ZSM Closed-Loop concept, trying to identify a mapping between the steps for implementing and maintaining a given level of security into a target system, proposed by the NIST, and the stages of Closed-Loop as defined by ETSI ZSM.

Then, an overview of the Zero-Touch Security Platform functional architecture has been provided, as an introduction to the detailed discussion related to its key pillars i) security service and resource orchestration, ii) programmable pervasive monitoring, iii) threat/anomaly detection, and prediction iv) closed-loop-based security automation. Potential integrations of these functionalities in 6G systems are also discussed. For (i) and (ii), the components in charge of implementing the respective functionalities have been presented, Security Orchestrator for the Security Orchestration, Resource Orchestrator for the Resource Orchestration, and Programmable Monitoring Platform for the Programmable Pervasive Monitoring. For all of them, scope, early-stage architecture, and working logic have been discussed. In particular, for the security Orchestrator, an extension of the OnSOAP ontology has been proposed, to meet the ROBUST-6G security orchestration requirements. Similarly, for the Resource Orchestration, studies on techniques for secure resource allocation have been proposed. For the Programmable Monitoring Platform, the more mature component, specification of internal modules, potential SW baseline, and workflow, encompassing the interaction of the platform's internal modules has been presented. Placement and interaction with the other components of the Zero-Touch Security Platform have also been described.

Techniques for anomaly detection and prediction, based on the use of proper AI/ML algorithms have also been discussed, with an investigation of the potential cyber threats to be addressed, suitable datasets for training, and workflows for mitigation. The AI/ML algorithms and the Programmable Monitoring Platform will be part of Security Closed-Loops, implementing the Monitoring and the Analysis stages respectively, along with the Decision and Execution stages. In this sense, the description of the Security Closed-Loop implementation in ROBUST-6G is provided: an entity conceived as totally configurable to address the security constraints specified by the Security Orchestrator.
The results presented in this document represent the baseline towards the first stable software design of the ROBUST-6G Zero-Touch Security Platform targeted for the second year of the project, whose early prototype will be reported in D4.2.

5 References

[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.

[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.

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

[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.

[GDZ+23] S. C. Garcia, E. G. De La Calera Molina, A. M. Zarca and A. F. Skarmeta Gomez, “Dynamic ZSM Multi-Operator Policy Based Security Framework for B5G Infrastructures”. 2023 IEEE Future Networks World Forum (FNWF), Baltimore, MD, USA, 2023, pp. 1-6.

[BT20] C. Benzaid and T. Taleb, "AI-Driven Zero Touch Network and Service Management in 5G and Beyond: Challenges and Research Directions”. IEEE Network, vol. 34, no. 2, pp. 186-194, March/April 2020.

[XKK+23] M. Xevgenis, D. G. Kogias, P. A. Karkazis, H. C. Leligou, “Addressing ZSM Security Issues with Blockchain Technology”. Future Internet 2023, 15, 129

[Bet24] Bettercap, 2024, https://www.bettercap.org/ .

[Doc24a] Bettercap image from Docker Hub, 2024, https://hub.docker.com/r/bettercap/bettercap .

[Kis24] Kismet, 2024, https://kismetwireless.net/ .

[Air22] Aircrack-NG, 2022, https://www.aircrack-ng.org/index.html .

[Doc24b] Aircrack-NG image from Docker Hub, 2024, https://hub.docker.com/r/aircrackng/git .

[Git24a] Aircrack-NG repository from GitHub, 2024, https://github.com/aircrack-ng/aircrack-ng .

[Mun21] Munin, 2021, http://munin-monitoring.org/ .

[Doc24c] Munin image from Docker Hub, 2024, https://hub.docker.com/r/dockurr/munin .

[Git24b] Munin repository from GitHub, 2024, https://github.com/munin-monitoring/munin .

[Tsh24] Tshark, 2024, https://www.wireshark.org/docs/man-pages/tshark.html .

[Wir24] Wireshark, 2024, https://www.wireshark.org/ .

[Doc21] Wireshark image from Docker Hub, 2021, https://hub.docker.com/r/wireshark/wireshark-opensuse-15.2-dev .

[Git24c] Wireshark repository for GitLab image containers, 2024, https://gitlab.com/wireshark/wireshark-containers .

[Tcp24] Tcpdump & Libpcap, 2024, https://www.tcpdump.org/ .

[Sno24] Snort++/Snort3, 2024, https://www.snort.org/snort3 .

[Zee24a] Zeek, 2024, https://zeek.org/ .

[Doc24d] Zeek image from Docker Hub, 2024, https://hub.docker.com/r/zeek/zeek .

[Zee24b] Zeek Package Manager, 2024, https://packages.zeek.org/ .

[Zab24a] Zabbix, 2024, https://www.zabbix.com .

[Doc24e] Zabbix base image from Docker Hub, 2024, https://hub.docker.com/r/zabbix/zabbix-build-base .

[Zab24b] Zabbix + Kubernetes, 2024, https://www.zabbix.com/la/integrations/kubernetes .

[Go24] Go programming language, 2024, https://go.dev/ .

[Sur24] Suricata, 2024, https://suricata.io/ .

[Tri24] Trivy, 2024, https://trivy.dev/ .

[Doc24f] Trivy image from Docker Hub, 2024, https://hub.docker.com/r/aquasec/trivy .

[Git24d] Trivy-operator for Kubernetes, 2024, https://github.com/aquasecurity/trivy-operator .

[Git24e] Clair, 2024, https://github.com/quay/clair .

[Fal24a] Falco, 2024, https://falco.org/ .

[Fal24b] Falcosidekick, 2024, https://falco.org/tags/falcosidekick/ .

[Doc24g] Falco image from Docker Hub, 2024, https://hub.docker.com/r/falcosecurity/falco .

[Col24] Collectd, 2024, https://www.collectd.org/ .

[Ale24] Prometheus Alertmanager, 2024, https://prometheus.io/docs/alerting/latest/alertmanager/

[Doc24h] Collectd image from Docker Hub, 2024, https://hub.docker.com/r/collectd/ci .

[Git24f] Ci-docker subproject from Collectd repository, 2024, https://github.com/collectd/ci-docker .

[Kco17] Kcollectd, 2017, https://www.forwiss.uni-passau.de/~berberic/Linux/kcollectd.html .

[Gra24a] Grafana, 2024, https://grafana.com/ .

[Rrd24] RRDtool, 2024, https://oss.oetiker.ch/rrdtool/index.en.html .

[Elk24] ELK (Elasticsearch, Logstash, and Kibana), 2024, https://www.elastic.co/ .

[Pac24] Packetbeat, 2024, https://www.elastic.co/es/beats/packetbeat .

[Met24] Metricbeat, 2024, https://www.elastic.co/es/beats/metricbeat .

[Wat24] Kibana Watcher, 2024, https://www.elastic.co/guide/en/kibana/current/watcher-ui.html .

[Ale24] Kibana Alerting, 2024, https://www.elastic.co/es/kibana/alerting .

[Doc24i] Elasticsearch image from Docker Hub, 2024, https://hub.docker.com/_/elasticsearch [Doc24j] Logstash image from Docker Hub, 2024, https://hub.docker.com/_/logstash .

[Doc24k] Kibana image from Docker Hub, 2024, https://hub.docker.com/_/logstash .

[Doc24l] Elastic official repository on Docker Hub, 2024, https://hub.docker.com/u/elastic .

[Flu24a] Fluentd, 2024, https://www.fluentd.org/ .

[Doc24m] Fluentd image from Docker Hub, 2024, https://hub.docker.com/_/fluentd .

[Doc24n] Fluentd official repository on Docker Hub, 2024, https://hub.docker.com/u/fluent .

[Flu24b] Fluentbit, 2024, https://fluentbit.io/ .

[Doc24o] Fluentbit image from Docker Hub, 2024, https://hub.docker.com/r/fluent/fluent-bit .

[Ope24] OpenTelemetry, 2024, https://opentelemetry.io/ .

[Doc24p] OpenTelemetry image from Docker Hub, https://hub.docker.com/r/otel/opentelemetry-collector-contrib .

[Zip24] Zipkin, 2024, https://zipkin.io/ .

[Doc24q] Zipkin image from Docker Hub, 2024, https://hub.docker.com/r/openzipkin/zipkin .

[Tel24] Telegraf, 2024, https://www.influxdata.com/time-series-platform/telegraf/ .

[Doc24r] Telegraf image from Docker Hub, 2024, https://hub.docker.com/_/telegraf .

[Oss24] OSSEC+, 2024, https://www.ossec.net/ .

[Doc20] OSSEC image from Docker Hub, 2020, https://hub.docker.com/r/atomicorp/ossec-docker/tags .

[Pro24a] Prometheus, 2024, https://prometheus.io/ .

[Doc24s] Prometheus image from Docker Hub, 2024, https://hub.docker.com/r/prom/prometheus .

[Pro24b] Prometheus Alertmanager, 2024, https://prometheus.io/docs/alerting/latest/alertmanager/ .

[Kap24] Kapacitor, 2024, https://www.ifluxdata.com/time-series-platform/kapacitor/ .

[Doc24u] Kapacitor image from Docker Hub, 2024, https://hub.docker.com/_/kapacitor .

[Doc24v] Grafana image from Docker Hub, 2024, https://hub.docker.com/r/grafana/grafana .

[Chr24] Chronograf, 2024, https://www.influxdata.com/time-series-platform/chronograf/ .

[Doc24w] Chronograf image from Docker Hub, 2024, https://hub.docker.com/_/chronograf .

[Gra24b] Grafana Mimir, 2024, https://grafana.com/oss/mimir/ .

[Doc24x] Grafana Mimir image from Docker Hub, 2024, https://hub.docker.com/r/grafana/mimir .

[Mar24] MariaDB, 2024, https://mariadb.org/ .

[Doc24y] MariaDB image from Docker Hub, 2024, https://hub.docker.com/_/mariadb .

[Mon24] MongoDB, 2024, https://www.mongodb.com/.

[Doc24z] MongoDB image from Docker Hub, 2024, https://hub.docker.com/r/mongodb/mongodb-community-server .

[Pos24] PostgreSQL, 2024, https://www.postgresql.org/ .

[Doc24aa] PostgreSQL image from Docker Hub, 2024, https://hub.docker.com/_/postgres .

[Inf24] InfluxDB, 2024, https://www.influxdata.com/ .

[Doc24ab] InfluxDB image from Docker Hub, 2024, https://hub.docker.com/_/influxdb .

[Kaf24a] Apache Kafka, 2024, https://kafka.apache.org/ .

[Kaf24b] Apache Kafka Connect, 2024, https://kafka.apache.org/documentation/#connect .

[Doc24ac] Apache Kafka image from Docker Hub, 2024, https://hub.docker.com/r/apache/kafka .

[Fil24] Filebeat, 2024, https://www.elastic.co/beats/filebeat/ .

[Doc24ad] Filebeat image from Docker Hub, 2024, https://hub.docker.com/r/elastic/filebeat .

[Zer24] ZeroMQ, 2024, https://zeromq.org/ .

[Doc15] ZeroMQ image from Docker Hub, 2015, https://hub.docker.com/r/zeromq/zeromq .

[Nat24] NATS, 2024, https://nats.io/ .

[Doc24ae] NATS image from Docker Hub, 2024, https://hub.docker.com/_/nats .

[Rab24] RabbitMQ, 2024, https://www.rabbitmq.com/.

[Doc24af] RabbitMQ image from Docker Hub, 2024, https://hub.docker.com/_/rabbitmq .

[VPC+22] Karima Velasquez, David Perez Abreu, Marilia Curado, Edmundo Monteiro, “Resource orchestration in 5G and beyond: Challenges and opportunities”, Computer Communications, no. 192, 2022, p. 311-315.

[ETS002] ETSI GS ZSM 002, “Zero-touch network and Service Management (ZSM); Reference Architecture”, 2019. Available: https://www.etsi.org/deliver/etsi_gs/ZSM/001_099/002/01.01.01_60/gs_ZSM002v010101p.pdf

[HUA23] Huawei, “6G White Paper”, 2023. Available: https://www.huawei.com/en/huaweitech/future-technologies/6g-white-paper

[LJN+23] L. Li, J. Jones, and N. Nee, "On End-to-End Intelligent Automation of 6G Networks," Future Internet 14, no. 6: 165, 2022.

[MIC23] Microsoft, “What is SOAR?”, 2023. Available: https://www.microsoft.com/it-it/security/business/security-101/what-is-soar

[RED23] Red Hat, “What is orchestration?”, 2023. Available: https://www.redhat.com/en/topics/automation/what-is-orchestration

[SJ23] S. Smith and J. Jones, "A multi-vocal review of Security Orchestration", ACM Computing Surveys, apr, no. 37:45, 2023.

[KA21] J. Kinyua and L. Awuah, “AI/ML in Security Orchestration, Automation and Response: Future Research Directions”, Intelligent Automation & Soft computing, no. 28(2), p. 527-545, 2021.

[Splu24] Splunk. Retrieved December 20, 2024, from https://www.splunk.com/

[Fire24] FireEye Developer Documentation. Retrieved December 20, 2024, from https://fireeye.dev/

[Simp24] SOAR Marketplace Integrations - Siemplify. Retrieved December 20, 2024, from https://cloud.google.com/chronicle/docs/soar/marketplace-integrations/siemplify

[Res24] Security Verify Documentation - Resilient Integrations. Retrieved December 20, 2024, from https://www.ibm.com/docs/it/security-verify?topic=integrations-security-resilient

[MZZ+23] J. M. B. Murcia, J. F. P. Zarca, A. M. Zarca and A. Skármeta, "By-default Security Orchestration on distributed Edge/Cloud Computing Framework," 2023 IEEE 9th International Conference on Network Softwarization (NetSoft), Madrid, Spain, 2023, pp. 504-509.

[HFT+22] “An automated closed-loop framework to enforce security policies from anomaly detection”, João Henriques, Filipe Caldeira, Tiago Cruz, Paulo Simões, Computers & Security Volume 123, December 2022.

[FLU22] Fluidos, “FLUIDOS (Flexible, scaLable, secUre, and decentralIseD Operating System)”, 2022. Available: https://www.fluidos.eu/

[GYC+22] Priyatham Ganta, Kicho Yu , Dharma Dheeraj Chintala, and Younghee Park, “Adaptive Network Security Service Orchestration Based on SDN/NFV”, 22nd International Conference, WISA 2021 Jeju Island, South Korea, August 11–13, 202, pp 231-242.

[Pos23] Dmitry Pospelov, “Ontologies in the digital age: Advantages, Limitation, and Alternativa Development Paths”

[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.

[NFV-MAN 001 V4.4.] GS NFV-MAN 001 V4.4.1: Network Functions Virtualisation (NFV); Management and Orchestration. Sophia Antipolis: European Telecommunications Standards Institute, December 2022. Available at: https://www.etsi.org

[AAF+24] Alyazia Aldhaheri, Fatima Alwahedi, Mohamed Amine Ferrag, Ammar Battah, Deep learning for cyber threat detection in IoT networks: A review, Internet of Things and Cyber-Physical Systems, Volume 4, 2024,.

[800-61-R2] National Institute of Standards and Technology. Computer Security Incident Handling Guide (SP 800-61 Rev. 2). Gaithersburg, MD: U.S. Department of Commerce, August 2012.

[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.

[CSF2.0] National Institute of Standards and Technology. "The NIST Cybersecurity Framework (CSF) 2.0," Cybersecurity White Papers, NIST CSWP 29, February 26, 2024

[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.

[Cha02] Chawla NV, Bowyer KW, Hall LO, Kegelmeyer WP (2002) SMOTE: Synthetic Minority Over-sampling Technique. Journal of Artificial Intelligence Research 16: 321-357.

[Hoc97] Hochreiter, S. & Schmidhuber, J. (1997). Long short-term memory. Neural Comput, 9(8), 1735-1780.

[Hen21] Henderi, H., Wahyuningsih, T., & Rahwanto, E. (2021). Comparison of Min-Max normalization and Z-Score Normalization in the K-nearest neighbor (kNN) Algorithm to Test the Accuracy of Types of Breast Cancer. International Journal of Informatics and Information Systems, 4(1), 13-20.

[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

[Int17] Intrusion detection evaluation dataset (CIC-IDS2017). Available online:

https://www.unb.ca/cic/datasets/ids-2017.html (accessed on 2 October 2024).

[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).

[Mou15] Moustafa, N. UNSW-NB15 Datasets. Available online: https://research.unsw.edu.au/projects/unsw-nb15-dataset (accessed on 2 October 2024).

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

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

[Tom76] Tomek Ivan (1976) An Experiment with the Edited Nearest-Neighbor Rule. IEEE Transactions on Systems. Man, and Cybernetics 6: 448-452.

[Tve92] A. Tversky and D. Kahneman, "Advances in prospect theory: Cumulative representation of uncertainty," J. Risk Uncertain., vol. 5, no. 4, pp. 297–323, 1992.

[Pre98] D. Prelec, "The probability weighting function," Econometrica, vol. 66, no. 3, pp. 497–527, 1998.

[Yan18] Yang, P., Sun, X., Li, W., Ma, S., Wu, W., & Wang, H. (2018). SGM: sequence generation model for multi-label classification. arXiv preprint arXiv:1806.04822.

[SLICENET20] SLICENET, "End-to-end Cognitive Network Slicing and Slice Management Framework in Virtualised Multi-Domain, Multi-Tenant 5G Networks," Horizon 2020, European Commission, Project ID: 761913, 2017-2020. Available: https://slicenet.eu/

[ZSM009-1-2023] ETSIZSM 009-1, "Zero-touch network and Service Management (ZSM); Closed-loop automation; Part 1: Enablers." Version 1.1.1, 2023-01.

[EUREG2024/1689] REGULATION (EU) 2024/1689 OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL of 13 June 2024 laying down harmonised rules on artificial intelligence and amending Regulations (EC) No 300/2008, (EU) No 167/2013, (EU) No 168/2013, (EU) 2018/858, (EU) 2018/1139 and (EU) 2019/2144 and Directives 2014/90/EU, (EU) 2016/797 and (EU) 2020/1828 (Artificial Intelligence Act), online (December 24) https://eur-lex.europa.eu/eli/reg/2024/1689/oj

[R6G24-D22] Deliverable D2.2: Use Cases, Requirements, ROBUST-6G Initial Architecture and Initial ROBUST-6G Dataspace. Horizon Europe Project, Grant Agreement No. 101139068.

[R6G24-D61] Deliverable D6.1: Use Case Validation Plan and Testbed Design. Horizon Europe Project,

Grant Agreement No. 101139068

[NFV018-2024] GR NFV-EVE 018 - V5.1.1 - Network Functions Virtualisation (NFV) Release 5; Evolution and Ecosystem; Report on Multi-tenancy in NFV (etsi.org)

6 ANNEX

6.1 PMP - State of the art of tools

This section of the annex shows the rest of the tools evaluated for the PMP.

1. **Bettercap** [Bet24], is a traffic analyser and a real-time traffic package handler that can detect devices through Man-in-the-Middle and sniffing technics. It can be used in wired and wireless networks, including IoT protocols such as Bluetooth among others. Also, it can be containerised, having an official imagen on Docker Hub [Doc24a]. In most cases, it is oriented towards network penetration testing but is quite efficient in collecting data from the network.
2. **Kismet** [Kis24] that proposes a wireless network detector, with monitoring and traffic analysis of Wi-Fi, Bluetooth and wireless protocols used in IoT devices. It can function as a Wireless Intrusion Detection System (WIDS), detecting anomalies and attacks in wireless networks, such as spoofing and Denial of Service (DoS), and generating alerts.
3. **Aircrack-NG** [Air22], a comprehensive suite for auditing, key cracking, packet capture, monitoring and replay attacks in wireless networks. It is primarily used for penetration testing and wireless security analysis. It can be containerised and has an official (although not shown as official) image on Docker Hub [Doc24b]. The tool has an official version released in 2022 but is currently updated on its GitHub in 2024 [Git24a].
4. **Munin** [Mun21], a network monitoring and metrics tool that can be containerised [Doc24c] and has official support for Kubernetes technology. The last version released is from 2021 but is currently updated on its GitHub in 2024 [Git24b]. It can visualise the collected information, store it in its own Round Robin Database (RRD) and report in case of anomalous behaviour may occur, but it is not an IDS. In addition, it has a long list of plugins to cover other purposes. As a final feature, it uses a Master-Node system, so it relies on nodes (agents) to extract information that the master will consume.
5. **Wireshark** [Wir24] which is similar to Tshark, but with an interface. It is containerisable and in this case, it has an official imagen on Docker Hub, which is not shown as official, but is uploaded by the official Wireshark account [Doc21], not being the latest version. The latest version is uploaded on its GitLab [Git24c].
6. **Tcpdump** [Tcp24] is a command-line packet analyser that provides detailed network traffic capture in real-time oriented to monitoring, network diagnostics and security analysis. It can be containerised, but does not have an official or sponsored image on Docker Hub. Its main functionality is to collect traffic network, filter it, and store it in PCAP files or transmit the data to other tools.
7. **Zeek** [Zee24a] is a powerful network analysis framework specializing in deep packet inspection, network traffic and log monitoring. It can be containerised, having an official image on Docker Hub [Doc24d]. Zeek provides a high-level view of network behaviour, allowing users to define customised detection and analysis policies thanks to its scripting language. Regarding the use of agents and proxies, it can be optionally integrated with these components. Agents can be used to collect information and transmit it to Zeek for further analysis. On the other hand, proxies can be used to forward logs and communication between multiple instances, especially in large-scale and distributed environments. Zeek Package Manager [Zee24b] enables this tool to install plugins.
8. **Zabbix** [Zab24a] is a real-time monitoring of servers, networks, and applications, among them. Zabbix collects many data types, from network traces to logs or health metrics. It can be containerised, having an official image on Docker Hub [Doc24e]. Also, it has a Kubernetes official integration [Zab24b] to be implemented. It supports agent-based monitoring and proxy-based for distributed environments. Zabbix offers a flexible alerting system and extensive visualization tools for generating dashboards and reports. The information is stored in a database defined during the installation of the tool. Finally, Zabbix has some plugins and a tutorial to create new ones using the Go [Go24] language.
9. **Suricata** [Sur24] is a high-performance network threat detection engine that functions as an IDS, IPS and network security monitoring tool. It also collects network traces and generates logs such as network event logs, intrusion attempts, anomalies, traffic patterns, and detected connection. Suricata can be containerised and used in a Kubernetes environment but does not have an official image.
10. **Trivy** [Tri24] is a SAST that scans container images, file systems, application dependencies and Git repositories searching vulnerabilities, misconfigurations and exposed secrets without executing the code. Also, it can detect vulnerabilities in local and cloud virtual machine (VM) images (.vmdk or

AMI files), in other words, static vulnerabilities in operative systems in VMs. Trivy can be containerised, it has an official image uploaded on Docker Hub [Doc24f] by the official organisation Aqua Security. In addition, it has compatibility with Kubernetes using Trivy-operator [Git24d].

11. **Clair** [Git24e] is also a SAST that realises static analysis of the container images, components and dependencies to identify potential vulnerabilities. It has several images on Docker Hub, but none with the current version. By default, Clair implements a PostgreSQL database to store scans reports and vulnerability information.
12. **ELK** stack (Elasticsearch, Logstash, and Kibana) [Elk24] is the sixteenth tool evaluated. ELK cannot collect network traces or system and services metrics but can use other agents developed by Elastic to cover these features such as Packetbeat [Pac24] to analyse packets and Metricbeat [Met24] to extract metrics and send them to ELK. Logstash is responsible for gathering event logs. Kibana Watcher [Wat24] (available in the commercial version) can set up alerts based on rules or thresholds. Also, Kibana Alerting [Ale24] allow to create and manage alerts when data meets certain conditions. Docker Hub has an official image for each tool [Doc24i] [Doc24j] [Doc24k], but Elastic also has an organisation account from which it uploads their own images [Doc24l]. Kibana provides the dashboards and graphs from its graphic user interface (GUI). Elasticsearch implements a database NoSQL by default.
13. **Zipkin** [Zip24] is a distributed tracing system. It provides a collecting network traces across distributed systems and can create dashboards for the visualisation of the information gathered. This tool can be containerised and has an official repository on Docker Hub [Doc24q]. In addition, Zipkin can operate with plugins called “extensions”.
14. **OSSEC+** [Oss24] is a Host-based Intrusion Detector System (HIDS) with the ability to collect event logs and information of the host. It provides file integrity, rootkit detection and proactive security management to stop threats using scripts. Despite this method, it is not considered an IPS solution, because these scripts act as a response to a trigger according to their creators. Additionally, OSSEC+ can be containerised and has an official image in Docker Hub [Doc20], but it is not up to date.

Continuing the list, the following types of tools are described in association with databases and event streamers.

1. **MariaDB** [Mar24] is a relational database with high scalability and security. It supports SQL queries and provides a storage engine making it versatile for a variety of tools. MariaDB can be containerised and has an official image on Docker Hub [Doc24y]. Also, can work with proxies such as MaxScale or HAProxy for load balancing, routing data security. This tool has a list of plugins to extend its functionalities.
2. **PostgreSQL** [Pos24] is a relational database with high robustness and SQL compliance. PgBouncer and HAProxy bring load balancing using proxies to PostgreSQL. As with other tools, it can be containerised and has an official image upload on Docker Hub [Doc24aa]. Thanks to its plugin system, PostgreSQL can extend features such as cryptographic functions, partition creation and maintenance or statistical data tracking.
3. **ZeroMQ** [Zer24] is an asynchronous messaging library that provides ultra-low latency with very high throughput but does not integrate guarantee delivery and persistence. It is designed to communicate microservice process in complex systems such as distributed or concurrent environments. ZeroMQ allows several message paradigms like publish/subscribe, push/pull and request/reply. It has an official containerisable image uploaded on Docker Hub [Doc15], but it is not up to date.
4. **NATS** [Nat24] is a lightweight messaging tool for real-time communication in distributed environments. Designed with a very high throughput, with ultra-low latency and moderate fault tolerance due to the optional message acknowledgement. It supports request/reply and publish/subscribe paradigms. NATS can be containerised and has an official image on Docker Hub [Doc24ae].

		Monitors						
		Bettercap	Kismet	Aircrack-NG	Munin	Wireshark	Tcpdump	Zeek
Monitoring	Network	X	X	X	X	X	X	X
	Host							
	Metrics				X			
	Logs							X
Anomalies	Anomalies detection		X					X
	Alert system		X		X			X
	Anomalies stop							
Virtualisation	Containerisation	X	X	X	X	X	X	X
	Official/Sponsored image	X		X	X	X		X
	Official Kubernetes compatibility				X			
Infrastructure	Can use agents				X			X
	Can use proxies							X
Vulnerability detection	Host level							
	Container level							
Visualisation	Graph generation				X			
	Graph visualisation				X			
Other features	Default data storage				X			
	Plugins		X		X			X

Table 6-1 : Monitors tools categorised and evaluated

		Monitors						
		Zabbix	Suricata	Trivy	Clair	ELK	Zipkin	OSSEC+
Monitoring	Network	X	X			*	X	
	Host	X				*		X
	Metrics	X				*		
	Logs	X	*			X		X
Anomalies	Anomalies detection		X					X
	Alert system	X	X			*		X
	Anomalies stop		X					*
Virtualisation	Containerisation	X	X	X	X	X	X	X
	Official/Sponsored image	X		X		X	X	X
	Official Kubernetes compatibility	X		X	X	X	X	
Infrastructure	Can use agents	X				*		X
	Can use proxies	X				X		
Vulnerability detection	Host level			*				
	Container level			X	X			
Visualisation	Graph generation	X				X	X	
	Graph visualisation	X				X	X	
Other features	Default data storage	X			X	X		
	Plugins	X				X	*	

Table 6-2 : Monitors and alert system tools categorised and evaluated.

		Databases and event streamers			
		MariaDB	PostgreSQL	ZeroMQ	NATS
Virtualisation	Containerisation	X	X	X	X
	Official/Sponsored image	X	X	X	X
	Official Kubernetes compatibility	X	X	X	X
Infrastructure	Can use agents				
	Can use proxies	X	X		
Visualisation	Graph generation				
	Graph visualisation				
Other features	Default data storage	X	X		
	Plugins	X	X		
Throughput	Very high			X	X
	High				
	Moderate				
Latency	Ultra-low			X	X
	Low				
	Moderate				
Delivery guarantees	High				
	Moderate				X
	Low				

Table 6-3 : Databases and event streamers tools categorised and evaluated.

6.2 TCP/IP Network Flow Statistics

Table 6-4: TCP/IP Network Flow Statistics [Ima18]

Feature Name	Description
Flow ID	ID of the network flow
Source IP	Source IP address of the network flow
Source Port	Source TCP or UDP port number
Destination IP	Destination IP address of the network flow
Destination Port	Destination TCP or UDP port number
Protocol	Type of protocol associated to the network flow
Timestamp	Timestamp of the network flow
Flow Duration	Duration of the network flow
Total Fwd Packets	Total number of packets in the forward direction
Total Backward Packets	Total number of packets in the backward direction
Total Length of Fwd Packets	Total size of packet in forward direction
Total Length of Bwd Packets	Total size of packet in backward direction
Fwd Packet Length Max	Maximum size of packet in forward direction
Fwd Packet Length Min	Minimum size of packet in forward direction
Fwd Packet Length Mean	Average size of packet in forward direction
Fwd Packet Length Std	Standard deviation size of packet in forward direction
Bwd Packet Length Max	Maximum size of packet in backward direction
Bwd Packet Length Min	Minimum size of packet in backward direction
Bwd Packet Length Mean	Mean size of packet in backward direction
Bwd Packet Length Std	Standard deviation size of packet in backward direction
Flow Bytes/s	Number of bytes transferred per second - Flow byte rate
Flow Packets/s	Number of packets transferred per second - Flow packets rate
Flow IAT Mean	Average time between two flows
Flow IAT Std	Standard deviation time two flows
Flow IAT Max	Maximum time between two flows
Flow IAT Min	Minimum time between two flows
Fwd IAT Total	Total time between two packets sent in the forward direction
Fwd IAT Mean	Mean time between two packets sent in the forward direction
Fwd IAT Std	Standard deviation time between two packets sent in the forward direction
Fwd IAT Max	Maximum time between two packets sent in the forward direction
Fwd IAT Min	Minimum time between two packets sent in the forward direction
Bwd IAT Total	Total time between two packets sent in the backward direction
Bwd IAT Mean	Mean time between two packets sent in the backward direction
Bwd IAT Std	Standard deviation time between two packets sent in the backward direction
Bwd IAT Max	Maximum time between two packets sent in the backward direction
Bwd IAT Min	Minimum time between two packets sent in the backward direction

Fwd PSH Flags	Number of times the PSH flag was set in packets travelling in the forward direction (0 for UDP)
Bwd PSH Flags	Number of times the PSH flag was set in packets travelling in the backward direction (0 for UDP)
Fwd URG Flags	Number of times the URG flag was set in packets travelling in the forward direction (0 for UDP)
Bwd URG Flags	Number of times the URG flag was set in packets travelling in the backward direction (0 for UDP)
Fwd Header Length	Total bytes used for headers in the forward direction
Bwd Header Length	Total bytes used for headers in the backward direction
Fwd Packets/s	Number of forward packets per second
Bwd Packets/s	Number of backward packets per second
Min Packet Length	Minimum length of a flow
Max Packet Length	Maximum length of a flow
Packet Length Mean	Mean length of a flow
Packet Length Std	Standard deviation length of a flow
Packet Length Variance	Minimum inter-arrival time of packet
FIN Flag Count	Number of packets with FIN flag was set
SYN Flag Count	Number of packets with SYN flag was set
RST Flag Count	Number of packets with RST flag was set
PSH Flag Count	Number of packets with PUSH flag was set
ACK Flag Count	Number of packets with ACK flag was set
URG Flag Count	Number of packets with URG flag was set
CWE Flag Count	Number of packets with CWE flag was set
ECE Flag Count	Number of packets with ECE flag was set
Down/Up Ratio	Download and upload ratio
Average Packet Size	Average size of packet
Avg Fwd Segment Size	Average size observed in the forward direction
Avg Bwd Segment Size	Average size observed in the backward direction
Fwd Avg Bytes/Bulk	Average number of bytes bulk rate in the forward direction
Fwd Avg Packets/Bulk	Average number of packets bulk rate in the forward direction
Fwd Avg Bulk Rate	Average number of bulk rate in the forward direction
Bwd Avg Bytes/Bulk	Average number of bytes bulk rate in the backward direction
Bwd Avg Packets/Bulk	Average number of packets bulk rate in the backward direction
Bwd Avg Bulk Rate	Average number of bulk rate in the backward direction
Subflow Fwd Packets	The average number of packets in a sub flow in the forward direction
Subflow Fwd Bytes	The average number of bytes in a sub flow in the forward direction
Subflow Bwd Packets	The average number of packets in a sub flow in the backward direction
Subflow Bwd Bytes	The average number of bytes in a sub flow in the backward direction
Init_Win_bytes_forward	Number of bytes sent in initial window in the forward direction
Init_Win_bytes_backward	# of bytes sent in initial window in the backward direction

act_data_pkt_fwd	# of packets with at least 1 byte of TCP data payload in the forward direction
min_seg_size_forward	Minimum segment size observed in the forward direction
Active Mean	Mean time a flow was active before becoming idle
Active Std	Standard deviation time a flow was active before becoming idle
Active Max	Maximum time a flow was active before becoming idle
Active Min	Minimum time a flow was active before becoming idle
Idle Mean	Mean time a flow was idle before becoming active
Idle Std	Standard deviation time a flow was idle before becoming active
Idle Max	Maximum time a flow was idle before becoming active
Idle Min	Minimum time a flow was idle before becoming active
Attack	Benign, Reconnaissance, Access, DoS/DDoS, Cryptojacking, etc.
Severity	Levels 1 (low) to 5 (highest)
Description	Brief description about the threat

6.3 Telemetry Data Features of IoT/IIoT Devices

Table 6-5: Telemetry Data Features of IoT/IIoT Devices

Service profile: IoT Heater activity			
ID	**Feature**	**Type**	**Description**
1	date	Date	Date of logging IoT telemetry data
2	time	Time	Time of logging IoT telemetry data
3	heater_temperature	Number	Temperature measurement of a heater sensor linked to the network
4	temp_condition	String	Temperature conditions of a heater sensor linked to the network, where temperature is high of low based on a predefined threshold value
Service profile: IoT Thermostat activity			
ID	**Feature**	**Type**	**Description**
1	date	Date	Date of logging IoT telemetry data
2	time	Time	Time of logging IoT telemetry data
3	current_temperature	Number	Current Temperature reading of a thermostat sensor connected with the network
4	thermostat_status	Boolean	Status of a thermostat sensor is either on or off
Service profile: IoT Lighting activity			
ID	**Feature**	**Type**	**Description**
1	date	Date	Date of logging IoT telemetry data
2	time	Time	Time of logging IoT telemetry data
3	light_status	Boolean	Status of a light sensor is either on or off
Service profile: IoT Occupancy sensor activity			
ID	**Feature**	**Type**	**Description**
1	date	Date	Date of logging IoT telemetry data
2	time	Time	Time of logging IoT telemetry data

3	Presence of human	Boolean	Detect the presence of human
Service profile: IoT Weather activity			
ID	**Feature**	**Type**	**Description**
1	date	Date	Date of logging IoT telemetry data
2	time	Time	Time of logging IoT telemetry data
3	temperature	Number	Temperature measurements of a weather sensor linked to the network
4	pressure	Number	Pressure readings of a weather sensor linked to the network
5	humidity	Number	Humidity readings of a weather sensor linked to the network
6	Rainfall	Number	The depth of rain collected by the rain gauge