Grant Agreement No.: 871808
Research and Innovation action
Call Topic: ICT-20-2019-2020: 5G Long Term Evolution

INSPIRE-5Gplus

INtelligent Security and PervasIve tRust for 5G and Beyond

D3.4: Smart 5G Security

Version: v1.1

Deliverable type	R (Document, report)
Dissemination level	PU (Public)
Due date	30/06/2022
Submission date	26/07/2022
Lead editor	Geoffroy Chollon (TSG)
Authors	Geoffroy Chollon, Cyril Dangerville, Dhouha Ayed (TSG), Vincent Lefebvre (TAGES), Rodrigo Asensio, Alejandro Molina Zarca, Pablo Fernández Saura (UMU), Orestis Mavropoulos (CLS), Pol Alemany, Roshan Sedar, Charalampos Kalalas, Ricard Vilalta, Raul Muñoz (CTTC), Chafika Benzaid, Amir Javadpour, Tarik Taleb (OULU), Wissem Soussi, Gürkan Gür (ZHAW), Antonio Pastor, Hugo Ramón Pascual, Diego Lopez (TID), Huu Nghia Nguyen (MI), Maria Christopoulou, Themistoklis Anagnostopoulos, George Xilouris (NCSRD)
Reviewers	Edgardo Montes de Oca (MI), Chafika Benzaid (OULU), Gürkan Gür (ZHAW), Dhouha Ayed (TSG)
Work package, Task	WP 3, T3.4
Keywords	Smart security, attack mitigation, security for 5G, ZSM closed-loop

Abstract

This document describes the result attained in the WP3 within the Task 3.4 "Smart security management". It describes the final state of the WP3 enablers and explains how their combination allows to create multiple closed-loops that follow the ZSM vision.

Document revision history

Version	Date	Description of change	List of contributor(s)
v0.1	08/03/2022	Draft TOC	Geoffroy Chollon (TSG)
v0.2	08/03/2022	First round of contributions for section 2	All partners
v0.3	24/06/2022	Contributions to sections 2 and 3	All partners
v0.5	28/06/2022	Review start	Edgardo Montes de Oca (MI), Chafika Benzaid (OULU), Gürkan Gür (ZHAW), Dhouha Ayed (TSG)
v0.7	06/07/2022	Modifications after review	All partners
V0.8	15/07/2022	Final version	Geoffroy Chollon (TSG)
V0.9	18/07/2022	Final editing, sent for GA approval	Uwe Herzog, Anja Köhler (Eures)
V1.0	26/07/2022	Review by Dhouha Ayed implem., Section 2.14.5 added, further editorial improvements	Dhouha Ayed (TSG), Orestis Mavropoulos (CLS), Uwe Herzog (EURES)
v1.1	20/02/2023	Fixes some figures without text references	Geoffroy Chollon (TSG)

List of contributing partners, per section

Section number	Short name of partner organisations contributing
Section 1	TSG
Section 2	TSG, TAGES, UMU, CLS, CTTC, UOULU, ZHAW, TID, MI, NCSRD
Section 3	ZHAW, CTTC, UOULU, TSG, UMU, TAGES
Section 4	UMU, TSG
Section 5	TSG
Section 6	TSG

Disclaimer

[1] http://creativecommons.org/licenses/by-nc-nd/3.0/deed.en_US

Acknowledgment
The research conducted by INSPIRE-5Gplus receives funding from the European Commission H2020 programme under Grant Agreement No 871808. The European Commission has no responsibility for the content of this document.

Executive Summary

This document presents the design and implementation of different smart 5G security methods and techniques that are essential for achieving Zero-Touch Network and Service Management (ZSM). The link with the High-Level Architecture (HLA) is explained, as well as the future work and gaps that need to be addressed. The ZSM closed loops presented are the results obtained by the task T3.4 of the work package WP3 of the INSPIRE-5Gplus project. They are based on the key enablers that have been developed and are described in more detail with the latest updates.

Table of Contents

List of Figures

List of Tables

Abbreviations

5G-PPP	5G Infrastructure Public Private Partnership
AAA	Authentication, Authorization, Accounting
AAE	Adversarial Autoencoder
API	Application Programming Interface
BSM	Basic Safety Message
DoS	Denial-of-Service
FL	Federated Learning
HLA	High-Level Architecture
MSPL	Medium-level Security Policy Language
NFV	Network Function Virtualization
OODA	Observe Orient Decide Act
RL	Reinforcement Learning
RSU	Road Side Unit
SSLA	Security Service Level Agreement
V2X	Vehicle-to-Everything
VNF	Virtual Network Function
ZSM	Zero Touch Management

1 Introduction

This document is a technical report for the Task T3.4 of INSPIRE-5Gplus project. It provides a smart security management framework that is compliant with the ETSI Zero-touch Network and Service Management (ZSM) reference architecture and the High-Level Architecture (HLA) defined in WP2. The framework integrates the software-defined and cognitive enablers defined respectively in Tasks T3.2 and Tasks T3.3 in order to enable end-to-end (E2E) management in a closed-loop and smart way.

The deliverable describes the final state of technical implementations of the WP3 enablers and the various interactions between them and with WP4 enablers in order to achieve a ZSM closed loop implementation of the HLA. It also details the narrowed scope of each enabler and explains, at their respective level, how to build smaller scale reaction loops, which allows the creation and delivery of multiple "low-scale" and "specific" ZSM closed loops.

The document is organized as follows. Section 2 describes the state of technical implementation reached at the end of the project for the WP3 enablers and the various interactions with each other and with enablers from WP4 in order to illustrate the way they implement the HLA. Section 3 details the various "small-scale" ZSM closed loops built around those enablers. Section 4 elaborates on how the conflicts and the priorities are managed when the closed loops are deployed in a multi-domain context. Finally, the conclusion provides a parallel with the ETSI ZSM vision.

2 INSPIRE-5Gplus Smart Security Management Framework

This section describes the architecture and the implementation of Smart 5G Securities enablers. They are the core enablers that provide the INSPIRE-5Gplus security framework and plays various role in the ZSM loops.

2.1 MOTDEC: MTD controller

2.1.1 Overview

A great challenge in securing 5G and beyond networks is the prevention and mitigation of attacks on large-scale infrastructures. Moving Target Defense (MTD) provides proactive and reactive protection of 5G assets in an NFV environment by reducing the action space of malicious users in the time dimension. MTD consists in moving the assets of the network that might be targeted in various levels of the network: at the virtualization level, at the infrastructure level and at the traffic level. The MTD controller (MOTDEC) enabler allows performing such operations with the main objective of reducing the time window of attackers for intelligence gathering and attack execution, as well as for mitigation of occurring detected attacks. Together with the OptSFC enabler, described in the previous deliverable D3.3, MOTDEC is part of an automated security management system that uses machine learning (ML) to assess the state of the network and optimize the MTD strategy for efficient security and minimized overhead.

2.1.2 Role in the HLA

Figure 1 illustrates the HLA components (in orange) involved in the implementation of MOTDEC.

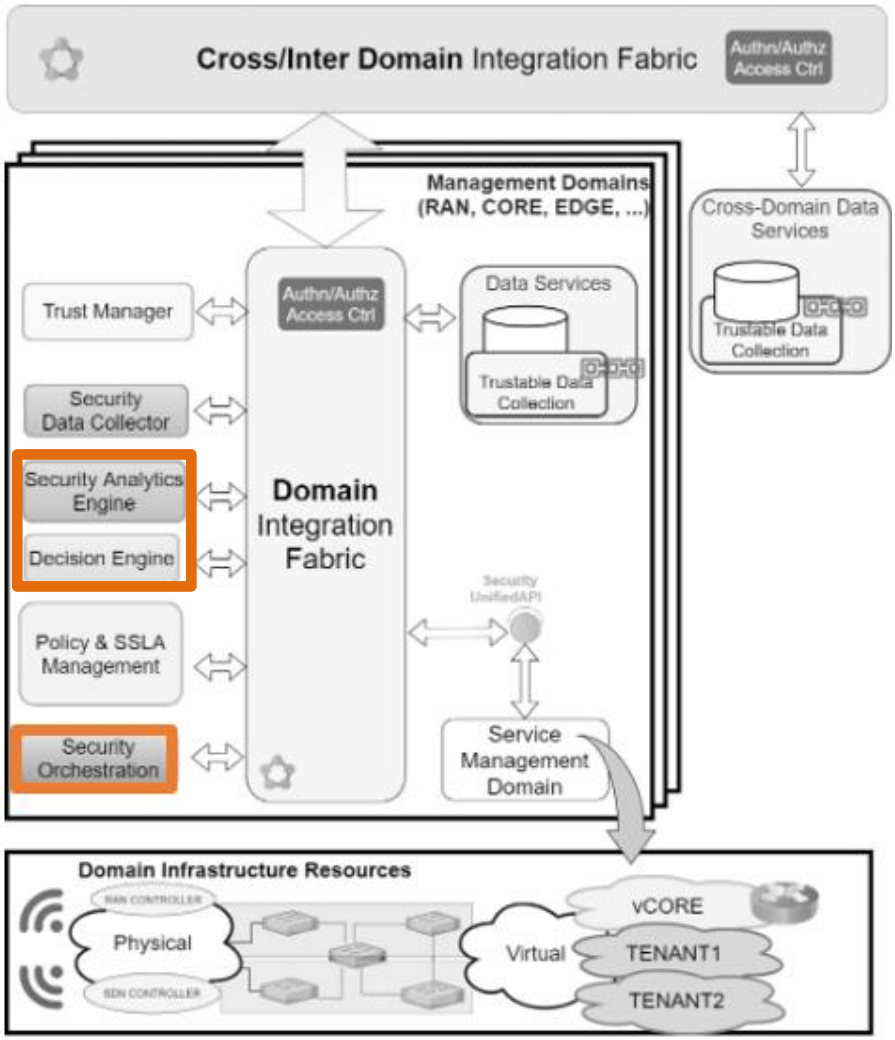

Figure 1: HLA components mapped to MOTDEC and OptSFC

2.1.2.1 SAE Threat assessment Service

MOTDEC integrates a Monitoring System that collects measurements on the network's resource consumption, network performance, and security assessments. This data is fed in near real-time to the OptSFC which models the state of the network as a Markov Decision Process (MDP), described in the previous deliverable D5.3.

2.1.2.2 DE Security Decision Service

The MDP obtained from the Threat assessment Service is used to train a multi-objective deep Reinforcement Learning (RL) agent and get an ML model for the decision making and the MTD strategies optimization.

2.1.2.3 SO MTD Security Orchestrator Service

MOTDEC provides a Security Orchestrator Service, enforcing MTD operations that affect the life-cycle of network slices assets, i.e., Network Services (NS) and Virtual Network Functions (VNF), as detailed in Section 2.1.3.

2.1.3 Sequence diagrams

Figure 2 depicts the initial interactions of MOTDEC with the network slice manager and the NFV orchestrator. These interactions allow the MOTDEC to connect to the management and orchestration

system of the 5G infrastructure, discovering the various network slices, NSs and VNFs running on the network. Through the information on the virtual links MOTDEC acquires various information:

1. the current network topology and dependency between VNF assets in near real-time
2. the resource requirements of the VNF assets from the high-level perspective of network slice to the granular view of single Virtual Deployment Units (VDUs).
3. the different Virtual Infrastructure Managers (VIM) running in dislocated locations, geographically expanding the network and forming an Edge architecture.
4. the resource consumption and resource availability of the infrastructure in near real-time.

Coupled with the network information from primitive network performance monitoring, MOTDEC populate its relational database (RDB) and its time series database (TSDB). Both DBs are shared with the OptSFC cognitive enabler, which uses the information for the assessment of the network state using Markov Decision Process, as described in detail in the previous deliverable D3.3.

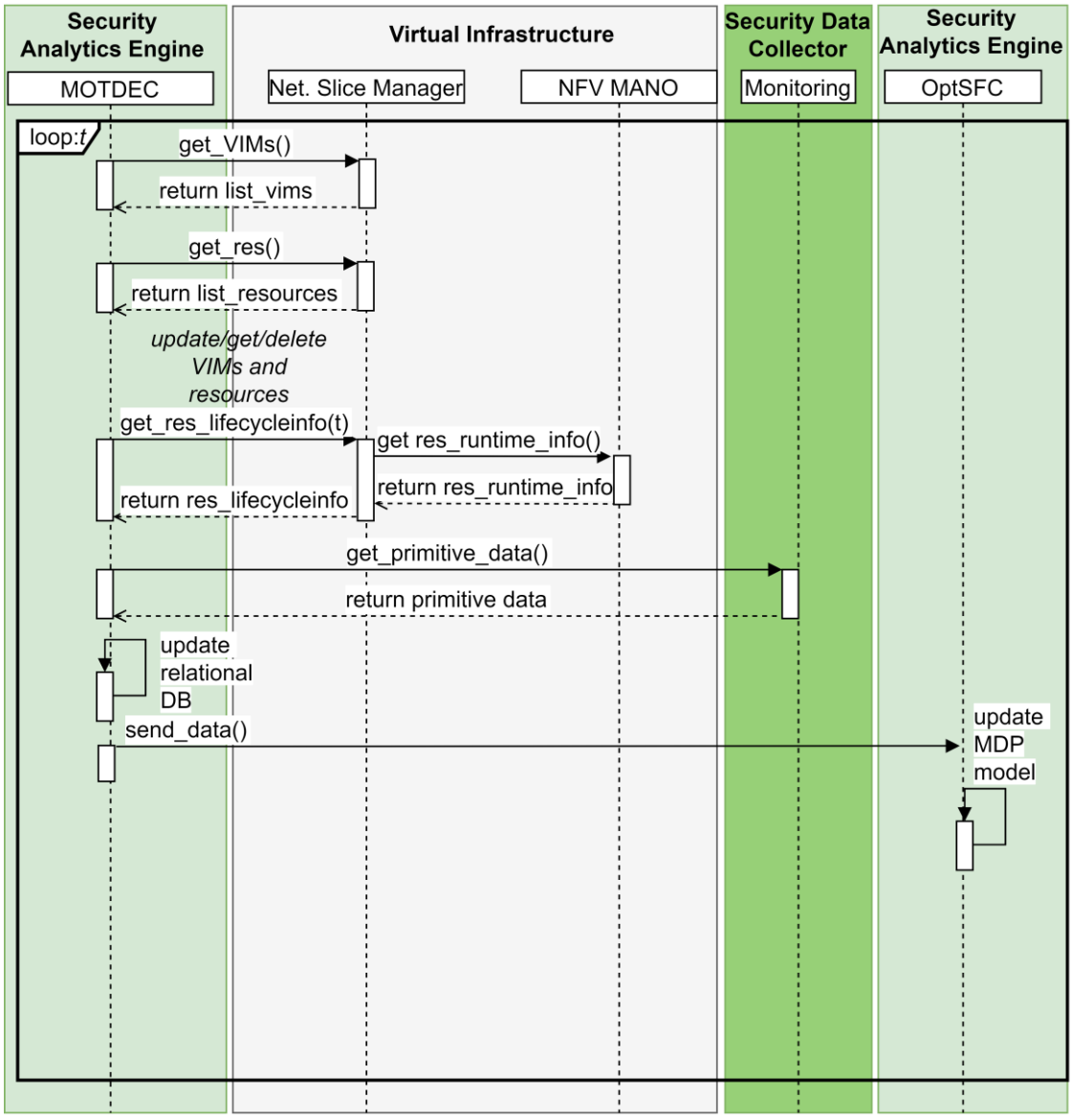

Figure 2: MOTDEC-OptSFC sequence diagram - analytic phase

Figure 3 shows the second phase of the MOTDEC-OptSFC workflow, operating the decision-making for MTD operations and strategy. These can be reactive as, for instance, when a security agent sends an anomaly or attack alert; or proactive, when they are based on the MDP modelling of the state of the network to harden exploitability tasks of possible offenders. The MTD operation proposed by OptSFC is enforced by MOTDEC, who receives a first-hand validation from a possible SSLA orchestrator to verify

conflicts with other orchestrating operations and high-level constraints.

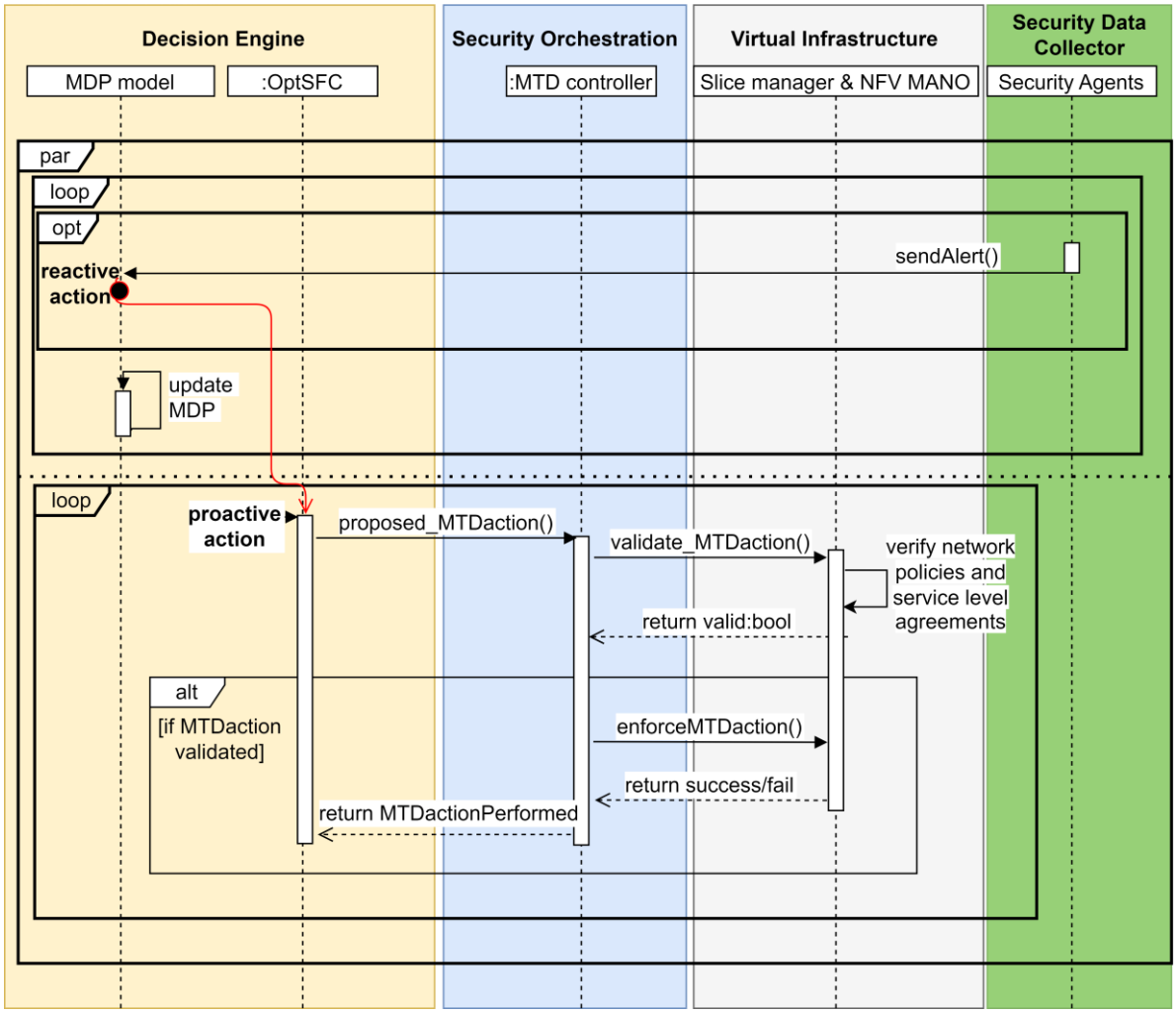

Figure 3: MOTDEC-OptSFC sequence diagram - decision-making phase

2.1.4 Technical implementation

MOTDEC is implemented in Python, using the Django web framework[2]. MOTDEC and OptSFC are setup as two separate applications of one Django project. The OpenAPI interfaces defined for the enablers are implemented using the Django REST framework[3]. The relational database is implemented with SQLite, and is accessible both via REST API calls and a web-based GUI. MOTDEC is enabled to be interact with Open-Source Mano (OSM[4]), used as the NFV orchestrator, Katana[5], as the network slice manager, and MMT probe, as a security agent and network monitoring probe. The interaction between such entities happens also via REST API calls.

The MTD actions that MOTDEC provides are grouped in 2 distinct categories:

- Soft MTD actions: these are SDN-based shuffle operations performed on network interfaces, traffic flow, and network topology on both the internal and the external/public views of the network. In the internal view, MOTDEC could prevent an attacker inside the network slice from easily exploring and further penetrating it. In the external/public view, the resource is meant

[2] https://www.djangoproject.com/

[3] https://www.django-rest-framework.org/

[4] https://osm.etsi.org/

[5] https://github.com/medianetlab/katana-slice_manager

to be always accessible by external devices with a public interface, and it provides a different public IP address to suspicious end-users or UEs, allowing further targeted analysis of their traffic and adding a second layer of security through proxy VNFs. To this scope, MOTDEC integrates an SDN controller (i.e., ONOS) and creates a middle virtual network, called Topology Fuzzer, used to change the node links and network data flow, increasing the difficulty of identifying the network topology. Similar to the work presented by Islam et al. [6], MOTDEC assigns dynamic ephemeral IP addresses to the virtual nodes and redirects the packets to the protected resources with a softwarized address translation (NAT).

- Hard MTD actions: these are operations directly performed on the NFV assets used in the 5G infrastructure, for both assets allocated by the operator's clients and assets deployed by the operator to provide and manage their services. The MTD actions are: 1) MTD restart action: here MOTDEC restarts the NSs or NFs by re instantiating the resource starting from authenticated images. This mitigates security scenarios of attackers introducing themselves in the virtual units to eavesdrop and acquire sensitive data, to block the application running on the unit and resulting in a DoS attack, to encrypt the unit with ransomware, or to create a C&C bot and exploit it as a vector for other chained attacks. The new instance of the service replaces the old one, expelling the intruder from the logical (and physical) resource. The old infected instance is then deleted. 2) MTD cloud diversity action: here MOTDEC moves the protected resource from a virtual infrastructure manager (VIM) to another one with a different cloud execution environment, e.g., from an OpenStack one to a VMware one. This changes the environment of the running resource and reduces the threats due to new specific system's vulnerabilities. To reduce the impact on the the running connections and their quality of service (QoS), the old instance of the service is used until the new instance is ready and operational. This allows a network overhead bound to SDN performances and similar to soft MTD actions. However, additional resources are allocated while the old instance of the service is still running, leading to additional operational costs if compared to soft MTD actions. In practice, the MTD cloud diversity action is similar to the MTD restart action, except that it creates the new instance of the resource in a different VIM; consequently, this can also solve the same threats addressed by the MTD restart action.

  Currently, MOTDEC provides hard MTD actions. The first category of MTD actions is referred to as "soft" because its operations are software-defined network configurations that require minimal resources and are fast to execute. The second category is referred to as "hard" because of its higher resources consumption and time required to execute the actions. However, an interesting aspect of their specific implementations is the fact that old instances stay employed until the new ones are operational.

2.1.5 Summary, lessons learned and guidelines

The need for an autonomous security management system to handle more extensive and complex telecommunication networks is increasing. MTD is a promising technique that shifts the spatiotemporal attack surface of networks at multiple levels: at the application level, the networking level, and the infrastructure level. As a network operator, using all possible MTD operations in a full-stack fashion is complex, especially for the application level, as this requires access to all hosted VNFs, public and private. MOTDEC explored the usage of networking-based and virtualization-based MTD operations as these do not require additional permissions for the hosted services.

Defining an interface that allows application-level MTD to cope with full-stack network security can help create and investigate new combined MTD actions and strategies to improve the network's overall security.

Training an MTD policy solely on a running 5G testbed might require a considerable amount of time, as training episodes may take on the order of seconds, and some ML models might need hundreds of thousands of such episodes. Future work may improve the training phase by using transfer learning,

for example, with models pre-trained in a simulated environment[6]. Such initial training would be faster and less resource-demanding. However, the model might still be non-optimized for real networks due to additional variables, patterns, and KPIs not modelled in the baseline simulation.

Finally, explainability is crucial in cybersecurity decisions, especially when dealing with network slices and VNF resources, underlying elements of critical 5G/B5G networks. Human-in-the-loop will be present for security management, even in closed-loop and autonomous settings, at least in the medium term. Therefore, it is essential to look into the explainability of AI models in future studies, as also mentioned in D3.3.

2.2 SAF: Anomaly Detection in 5G

2.2.1 Overview

The Security Analytics Framework (SAF) provides an anomaly detection service based on deep learning, that was presented in D3.3. In this document, we present its role on the INSPIRE-5Gplus HLA and provide more information regarding its interactions with other enablers. The main objective of SAF is to monitor incoming data from various sources, e.g., compute resources (CPU, RAM, disk), network metrics for profiling the provided QoS (latency, throughput, error rate) and specific application metrics, depending on the deployed services. SAF provides alerts when it detects an anomaly, which could be a potential attack on the network. SAF is based on Keras[7], TensorFlow[8] and TensorFlow Serving[9] for its implementation, as explained in the following sections.

2.2.2 Role in the HLA

SAF is mapped on the Domain Security Analytics Engine in the INSPIRE-5Gplus HLA (see Figure 4), providing the Domain Anomaly Detection Service, as defined in D2.2. SAF retrieves data from the deployed security agents provided by the MMT Framework, pre-processes these data with necessary normalization, standardization and cleaning procedures, trains a deep learning (DL) model and deploys this model into the “production” environment, where it performs inference based on the incoming data. The extracted insights are then forwarded to the Domain Decision Engine, for deriving a decision.

[6] S. J. Pan and Q. Yang, "A Survey on Transfer Learning," in IEEE Transactions on Knowledge and Data Engineering, vol. 22, no. 10, pp. 1345-1359, Oct. 2010, doi: 10.1109/TKDE.2009.191

[7] https://keras.io

[8] https://www.tensorflow.org/

[9] https://www.tensorflow.org/tfx/guide/serving

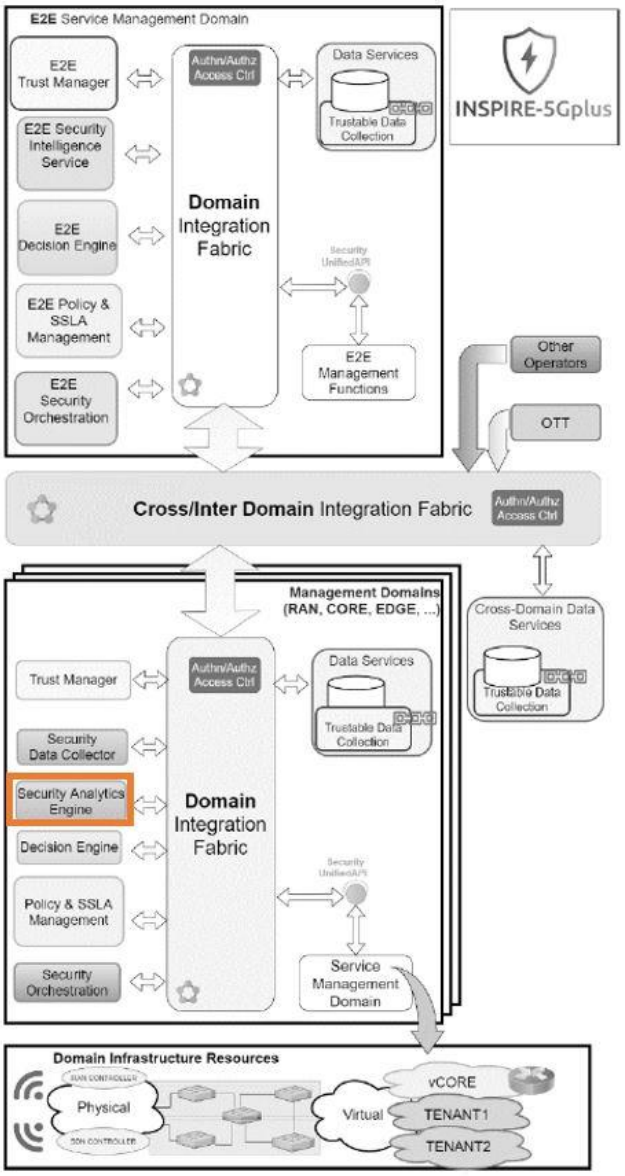

Figure 4: SAF mapping to the HLA

2.2.3 Sequence Diagram

SAF includes submodules for handling data collection, pre-processing, model training, inference and validation. Figure 6 depicts the underlying machine learning pipeline. SAF interacts with clients (see Figure 5) via the TensorFlow Serving API[10], providing predictions or information on the deployed models.

[10] https://www.tensorflow.org/tfx/guide/serving

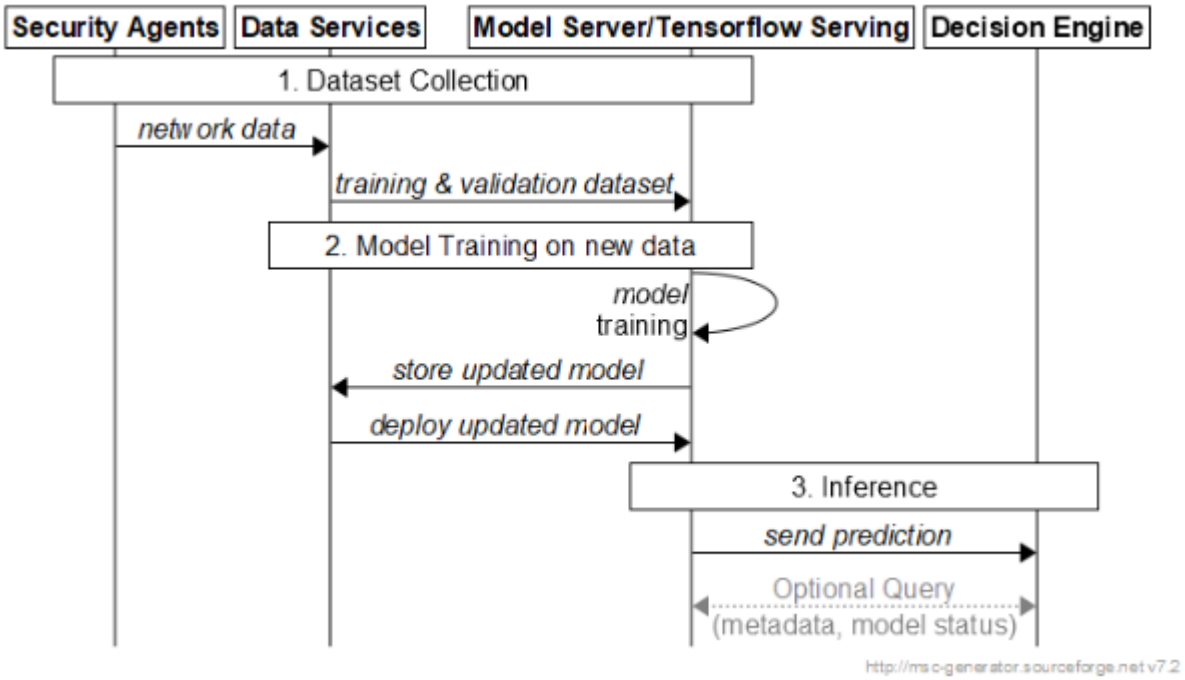

Figure 5 - SAF HLA interactions

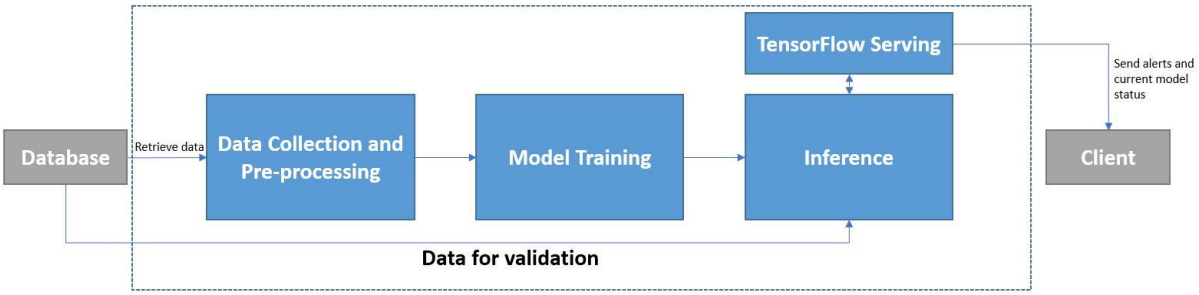

Figure 6: SAF interactions

2.2.4 Technical Implementation

SAF is built upon Keras, TensorFlow and TensorFlow Serving. Keras and TensorFlow are used for data pre-processing, the implementation of the deep learning model, training and inference. The DL techniques used were described in D3.3. TensorFlow Serving is a serving system for deploying models in production environments. “Serving a model” means running a ML model on a server and make their functions available via APIs so that upper-level applications can interact with it[11].

TensorFlow Serving is based on “Servables”, meaning the underlying objects that clients use to perform computation (e.g., inference). TensorFlow Serving handles one or more versions of a servable over the lifetime of a single server instance, enabling the gradual rollout of different algorithm configurations, weights, and other data over time. TensorFlow Serving also includes the “Loaders”, responsible for managing the Servable’s lifecycle. Loaders provide APIs for loading and unloading a servable, separating training-specific parameterization from the infrastructure. TensorFlow Serving runs as a Docker container.

We can use the CURL utility to pass the input values to our model and the response is a JSON object, including the requested prediction and information about the deployed model (e.g., model version, availability, error codes). A server executes the model and provides an API to interact with the deployed model. After initiating TensorFlow Serving, we open the REST API port and bind our local

[11] https://www.tensorflow.org/tfx/serving/architecture

models folder with a folder in the container. TensorFlow Serving provides the Classify, Regress and Predict APIs for providing this information. It is essential to specify the path where we export the model, so that TensorFlow understands all available models and their versions.

2.2.5 Summary, lessons learned and guidelines

Training a deep neural network on changing datasets is a complicated process. Part of this complexity results from fine-tuning the hyperparameters of the neural network which is a time-consuming process. Most papers to date, use ready-made, publicly available datasets and report accuracy metrics of their models, when trained on these specific datasets. However, these metrics only report the accuracy of a model at a specific time instance, if that model had been trained prior with that specific dataset. In a production environment, the dataset will be changing constantly and a dataset cleaning-processing pipeline workflow is needed before training. Therefore, it is important to follow an MLOps approach when validating a new ML model, considering additional factors, including: i) data sampling window and how it affects the model's performance, and ii) the feasibility of the ML pipeline, considering that the ML model should be trained with datasets that show the current status of the network. If the model is trained with past observations that do not reflect the current conditions, then the ML model will make wrong decisions. Another important lesson is the data sampling window and how it affects the training and testing performance of the ML model. In addition, future work will focus on scalable detection in high density environments (over 30 nodes). The current detection method is applicable on individual key performance indicators, so new methods need to be designed that integrate topology features in their decision-making workflow, considering spatial and temporal information in the design of the anomaly detection algorithm.

2.3 MMT: advanced traffic analysis in 5G planes

2.3.1 Overview

The state-of-the-art and challenges in anomaly detection in 5G networks are presented in Section 3.1.3 of the INSPIRE-5Gplus deliverable D3.3. A major issue is the analysis of encrypted traffic since it represents more than 70% of all network traffic today and is widely used by cyber attackers to obfuscate their activity. Machine learning techniques have shown that they can be useful to detect some of the most common attacks as shown in the work presented in Section 3.1.3.2 (Advanced Encrypted Traffic Analysis) of D3.3. Here we will describe how the developed solution fits in the INSPIRE-5Gplus HLA, provide information on the interactions between the different modules, and information on the implementation.

2.3.2 Role in the HLA

The HLA enablers involved are represented in the Figure 7 and their roles are explained in the following subsections.

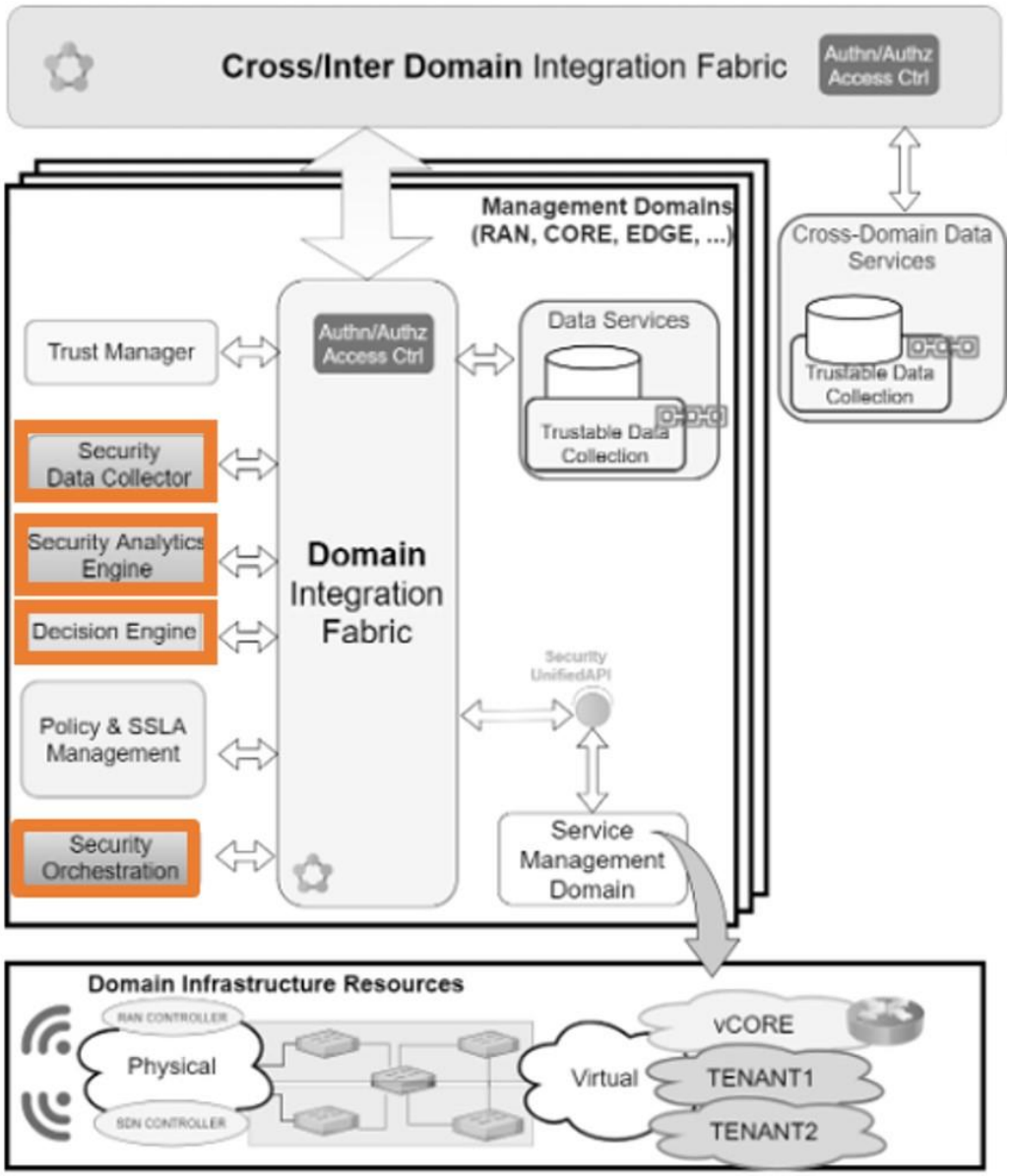

Figure 7: HLA enablers involved in the ML-based traffic analysis

2.3.2.1 Security Data Collector (SDC) and Security Agents (SA)

The solution proposed for analysing encrypted traffic obtains the required data in the form of captured packets (PCAP files or datasets) from the Security Agents deployed in the physical or virtual machines (e.g., MMT-Probes that can capture and compute the features needed for the learning and prediction phases). The packet capture is done in real time and the PCAP datasets and/or computed features are stored by the Data Collector (e.g., a data base management system - DBMS). The features correspond to values in the form of CSV files. In this way the learning phase can be done using the stored information that has been previously labeled as normal or abnormal; and the prediction phase can be done during operation to rapidly detect any abnormal or unexpected behaviour that needs to be acted on.

2.3.2.2 Security Analytics Engine (SAE)

The traffic analysis is a module integrated as part of the Security Analytics Engine (e.g., MMT-Operator). It will periodically receive the computed features from the probes so that in can evaluate them. It will also periodically, or when needed, update the ML model to keep up with any changes in the network topology or the threat landscape. The SAE's main roles are to manage the execution of

the learning and prediction functions, present different results in the form of dashboards, and communicate the results to the Decision Engine in the form of alarms or notifications.

2.3.2.3 Decision Engine (DE) and Security Orchestrator (SO)

The Decision Engine will receive the results of the analysis from the SAE in the form of structured messages (e.g., Kafka messages) and, in accordance with the user defined security policies, perform the appropriate countermeasures in conjunction with the Security Orchestrator.

2.3.3 Sequence diagrams

Figure 8 depicts the interactions between the different functions corresponding to the training phase and the prediction phase during operation. The Learning phase (corresponding to the upper rectangle) is an activity that needs to be performed periodically and each time there are significant changes in the network and the risks. It takes as input datasets containing normal and abnormal traffic that are used to create the ML model in the form of a numeric matrix.

The Prediction Phase (corresponding to the lower rectangle) is performed continuously using the real network traffic to extract the features and provide the verdict using the ML model to determine if a given session involves abnormal or normal behaviour.

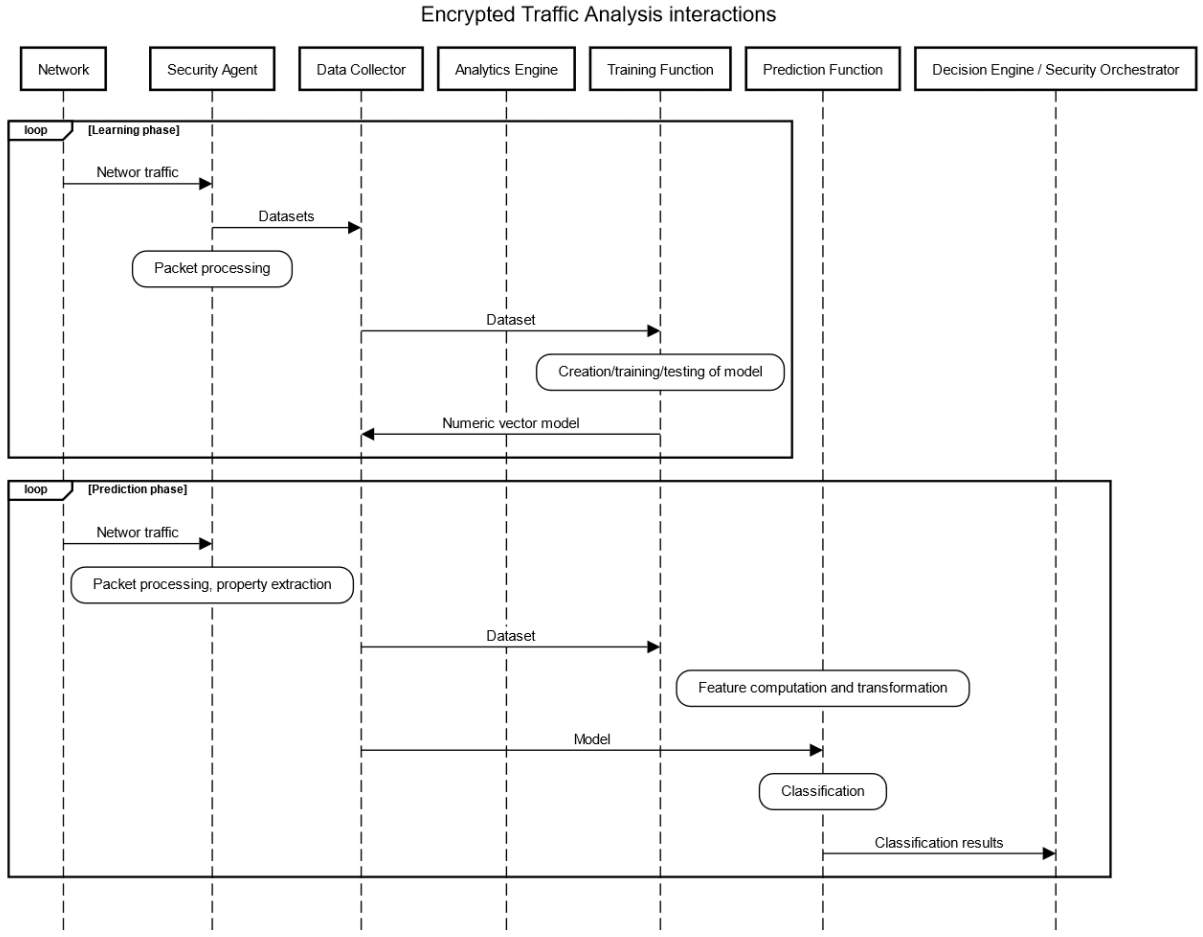

Figure 8: Encrypted traffic analysis training and prediction phases

2.3.4 Technical implementation

The different functions are implemented in Python and using the TensorFlow[12] library. The tensorflow-

[12] https://www.tensorflow.org/

GPU allows speeding up the processing.

The solution relies on the MMT Monitoring Framework to capture the network traffic and extract or calculate the required features that will be used for training the models and performing the predictions. This can be done using saved PCAP files (Packet CAPtures, i.e., datasets containing captured network packets) or by monitoring the traffic during operation.

The solution provides an API that allows other network functions to use the service. The interactions can be performed using several techniques, e.g., Kafka messages, REST, etc. Currently, the API implements the following REST functions:

1. `Get training-results`: retrieve results obtained from the creation of a new model
   - Used for obtaining information on the training process performed by the `post model` function. As an input, it takes the ID number of the model and an option (stats, preds, conf) depending on the information file desired. The stats option returns the KPI metrics of the model based on the testing phase. The preds option returns the predicted values of the testing phase. The conf option returns the confusion matrix values. All the information is structured as ".csv" files.
2. `Get model`: retrieve a new trained model
   - Used to request the model ".h5" file. It takes the ID of the model previously trained by the following post model function.
3. `Post model`: train and test a new model
   - Used for running the training process. It takes two files as input: the training set, and the test set. These files contain calculated and labelled traffic features. The labels 0 or 1, for respectively normal/malicious traffic, are found in the last column of the ".csv" file. The function returns the ID number of the model.
4. `Put model`: upload a new model using an identification number
   - Used for changing the default model used for classification. It takes the ID number of the new model as an input.
5. `Get classification`: return classification results using an identification number
   - Used to obtain the results of a classification. The function takes an ID number that identifies the requested network traffic classification. The classification result consists of a matrix of all the extracted features.
6. `Post classification`: perform a classification on inputted dataset
   - Used for transmitting the network traffic to the service and starting its classification. The feature extraction and Deep Learning modules are executed sequentially, and the function returns an ID number that identifies the given traffic to allow obtaining the classification results.

2.3.5 Summary, lessons learned and guidelines

Concerning advanced encrypted traffic analysis, it is important to consider that today more than 80% of the network traffic and attack traffic is encrypted. Current state of the art has shown interesting results when using ML techniques to detect different types of cyber-attacks (e.g., distributed denial of services - DDoS, exfiltration, replay attacks, man in the middle attacks, etc.) in encrypted traffic. However, several challenges remain that need to be addressed for the techniques to become usable in and scalable to real operating environments. The main challenges are: how to avoid needing to implement a different technique for each type of attack; how to manage the dynamicity of the software driven networks and fast evolution of attacks and evasion techniques; how to improve verdict accuracy to avoid the need of handling too many false positives; and how to improve the performance of the techniques used. For this, more work is required to find generic ML solutions that can handle many types of problems, and the security detection functions need to be optimised to concentrate on the higher risks. Furthermore, improving non-supervised or hybrid ML techniques could help address the dynamicity by eliminating the learning phase and improve the accuracy of the results. Finally, the

introduction of quantum key distribution and quantum communications need to be studied to determine how they can improve the resiliency of the networks and applications to attacks.

2.4 V2X misbehavior detection

2.4.1 Overview

Vulnerabilities in large-scale multi-domain automotive environments give rise to a wide range of attack variants, which may result in compromised functional components and end-to-end security issues against vehicular network users. At a first stage, this enabler aims to address security threats originated from attackers against functional elements of a vehicular authentication mechanism tailored for highly dense network scenarios. Through proper cuckoo filter configurations[13], the enabler is shown to improve the authentication resilience against Denial-of-Service (DoS) attacks. At a second stage, the enabler employs a data-driven methodology based on reinforcement learning (RL) to accurately detect misbehaving vehicles which already possess valid credentials and execute insider attacks[14]. The effectiveness of the RL-based approach for vehicle-to-everything (V2X) misbehaviour detection is evaluated by performing experiments using the open-source VeReMi dataset[15] in rapidly changing vehicular environments, and yields superior performance compared to benchmark approaches.

2.4.2 Role in the HLA

The involved HLA components in the enabler implementation are illustrated in the Figure 9.

[13] R. Sedar, C. Kalalas, J. Alonso-Zarate, F. Vazquez-Gallego, “Multi-domain Denial-of-Service Attacks in Internet-of-Vehicles: Vulnerability Insights and Detection Performance," in Proc. of IEEE International Conference on Network Softwarization 2022 (IEEE NetSoft ’22), Milan, Italy, July 2022.

[14] R. Sedar, C. Kalalas, F. Vazquez-Gallego, J. Alonso-Zarate, “Reinforcement Learning Based Misbehaviour Detection in Vehicular Networks," in Proc. of IEEE International Conference on Communications 2022 (IEEE ICC ’22), Seoul, South Korea, May 2022.

[15] [REF-3] J. Kamel, M. Wolf, R. W. van der Hei, A. Kaiser, P. Urien, and F. Kargl, “Veremi extension: A dataset for comparable evaluation of misbehavior detection in vanets,” in 2020 IEEE International Conference on Communications, 2020, pp. 1–6.

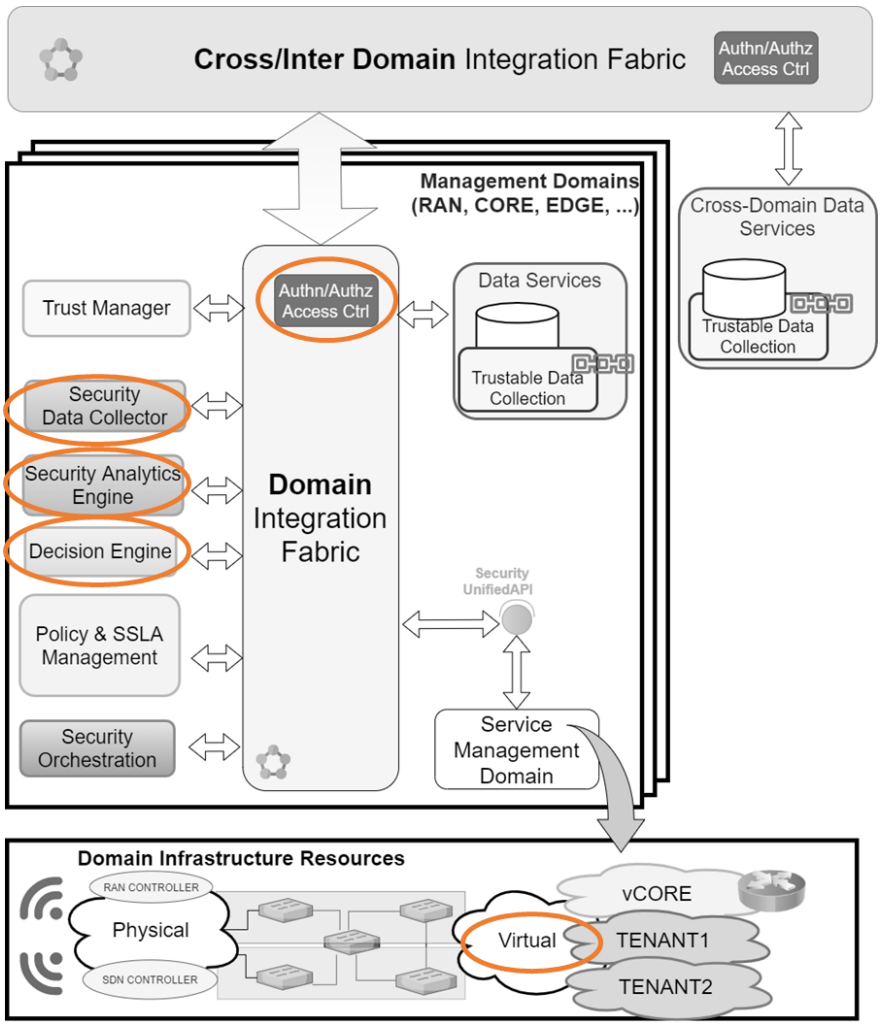

Figure 9: Involved HLA components

2.4.2.1 SDC Data Collection Service

Security data collector performs the fusion of vehicular network traces in .csv format that are streamed from the data plane using virtual machines, which emulates the representation of vehicles within the radio access network. These V2X traces are based on an open-source vehicular anomaly-detection dataset which is simulation-based, and can act as an evaluation baseline for the performance comparison of data-driven misbehaviour-detection techniques.

2.4.2.2 SAE Anomaly Detection Service

The incoming streaming vehicular data reports are sequentially analysed within the security analytics engine based on the mobility pattern parameters of vehicles, such as position, velocity, and acceleration, to instruct the RL algorithm for the detection of misbehaviour patterns. Our approach is shown to be highly effective in detecting various types of attacks, e.g., position falsification, DoS, sudden-stop, Sybil, etc., that can exist in V2X scenarios[14]. We consider the RL-based misbehaviour detector deployed in a road side unit (RSU) where it acts as an agent that interacts with the V2X environment to learn the optimal detection policy. The aggregated information at each RSU constitutes a time-series repository of received basic safety messages (BSMs) with intrinsic temporal and spatial inter-dependencies. The information contained in each BSM is constantly evolving over time along the vehicle trajectory while BSMs from neighbouring vehicles exhibit high spatial dependency.

2.4.2.3 DE Security Policy Proposal Service

Upon detection of misbehaviour, the detection framework in the decision engine provides the verdict to the security orchestrator to apply the pre-determined security policy, e.g., misbehaving data source to be isolated, dropped, or blocked. The issued security policy is expressed using medium-level security policy language (MSPL).

The workflow is illustrated in Figure 10.

2.4.3 Sequence diagrams

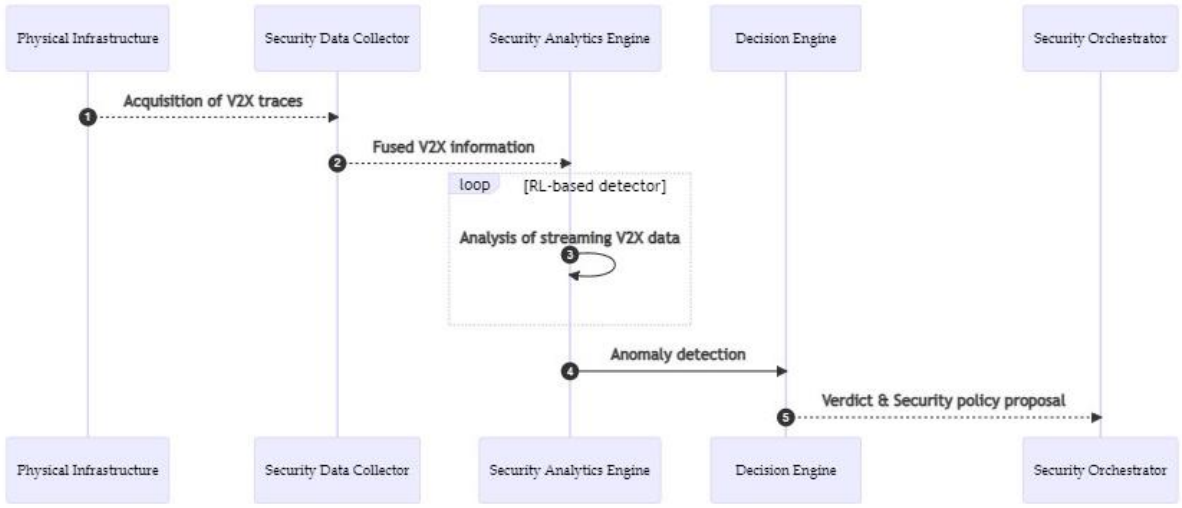

Figure 10: V2X misbehaviour detection workflow

Of particular interest is the detection of DoS insider attacks where a vehicle transmits BSMs at a frequency higher than the limit set by the standard. Such high volume of data transmission would result in extensive periods of network congestion and unavailability to serve other legitimate vehicles. Nevertheless, our enabler has been also successfully evaluated in 14 and 16, considering various types of attacks, acting either as standalone attacks or as a combination of multiple attack vectors. Attacks considered include position falsification, speed falsification, sudden-stop, data replay, delayed messages, traffic congestion and disruptive attacks.

Detection of different DoS attack variants is our focus on INSPIRE-5Gplus Demo 1 implementation. One more feature (not currently included in Demo 1), is the feasibility of a DoS attack execution against the authentication server function (AUSF) in the 5G authentication and key agreement (5G-AKA) mechanism proposed in 17. In 13, we have thoroughly investigated the impact of various design configurations on the resilience of the authentication scheme, and provide useful insights to decrease the effectiveness of DoS attacks against the AUSF.

2.4.4 Technical implementation

The technical implementation details and performance evaluation, including comparative assessment results against benchmark approaches, are documented in 13, 14, 16. A summary is also provided in earlier INSPIRE-5Gplus Deliverable 3.3. The V2X DoS detector constitutes part of the INSPIRE-5G Demo

[16] R. Sedar, C. Kalalas, F. Vazquez-Gallego, J. Alonso-Zarate, "Reinforcement Learning based Misbehaviour Detection in V2X Scenarios", in Proc. of IEEE International Mediterranean Conference on Communications and Networking 2021 (IEEE MeditCom 2021), virtual event, September 2021.

[17] C. Kalalas and J. Alonso-Zarate, “Lightweight and Space-efficient Vehicle Authentication based on Cuckoo Filter," in Proc. of IEEE 5G World Forum 2020 (IEEE 5G-WF ’20), virtual event, September 2020.

1 implementation, and, in particular, it is included in the security management domain at CTTC premises. The operating principle of the V2X DoS detector relies on the real-time detection of BSMs transmitted at a frequency higher than the limit set by the standard. Such high volume of data transmission would result in extensive periods of network congestion and unavailability to serve other legitimate vehicles. For the detection of DoS attacks in this domain, interaction is sought with the security orchestrator (and policy framework) for the applied policy which is represented by the inter-arrival BSM time threshold. In order to deploy the DoS detector, the security orchestrator receives the asset configuration information and requests the 5G slice deployment as part of the orchestration plan. Upon detection of the DoS attack, the security orchestrator instructs the filtering (reactive security policy) of the malicious traffic in the security management domain, which takes place as IP traffic filtering at the application layer.

2.4.5 Summary, lessons learned and guidelines

RL-based misbehaviour detection can be identified as a highly effective approach that can consistently improve the detection experience over time while interacting with unknown and rapidly changing V2X environments without relying on security threshold values. Our evaluation has yielded superior performance for accuracy, recall and F1 metrics compared to benchmark schemes, as demonstrated in 13, 14, 16.

In the path forward, we aim to extend the anomaly detection service of our V2X misbehaviour detection enabler by considering the case where the access to labelled training examples is limited. To this end, our RL-based detector will be complemented with an unsupervised learning layer for discovering hidden patterns from unlabelled V2X traffic traces. Unlabelled data instances will thus be annotated and will be used to train the RL-based detector to discriminate genuine vehicles from misbehaving ones.

2.5 Application-Layer DDoS Detection

2.5.1 Overview

The disruptive capabilities of 5G and beyond networks are envisioned to enable an extensive range of new applications and services. Unfortunately, they are expected to open the door to increased and sophisticated cybersecurity threats. One major security concern that may impede the potential of the anticipated 5G and beyond applications/services is the compromise of their availability through Distributed Denial of Service (DDoS) attacks. Recent years are showing that DDoS attacks are getting stealthier, targeting the application layer rather than the network layer. The complexity of handling application-layer DDoS attacks stems from their ability to mimic genuine behaviour with low-bandwidth usage.

To address this issue, we propose a robust application-layer DDoS self-protection framework; called DDoS Detector. The framework empowers a fully autonomous detection and mitigation of the application-layer DDoS attacks, leveraging Deep Learning (DL) and Software Defined Networking (SDN)[18]. It includes different modules that allow to automatically (i) collect data from network layer (i.e., characteristics of network flows); (ii) analyse the collected data using a DL model to identify malicious patterns in network flows; and (iii) issue a security policy for mitigating the detected malicious traffic pattern, where an SDN controller is used to enforce the generated policy. It is worth mentioning that the DL model we built is not only capable of effectively detecting application-layer DDoS attacks, but also of withstanding adversarial attacks against ML models.

[18] C.Benzaid, M. Boukhalfa, and T. Taleb, “Robust Self-Protection Against Application-Layer (D)DoS Attacks in SDN Environment,” in Proc. IEEE WCNC 2020, Seoul, Korea, Apr. 2020.

2.5.2 Role in the HLA

Figure 11 illustrates the components involved in the implementation of the DDoS Detector enabler.

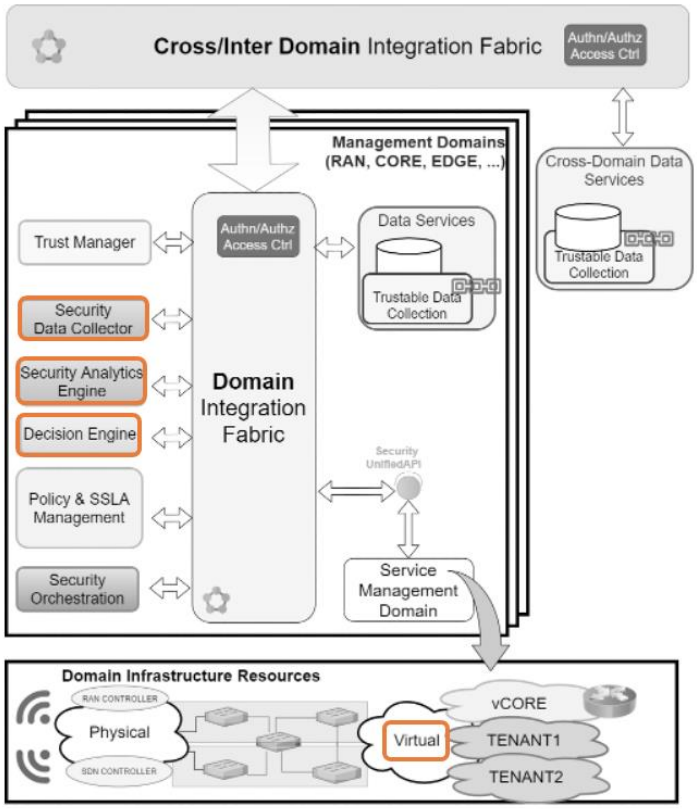

Figure 11: The INSPIRE-5Gplus HLA components involved in DDoS Detector enabler

2.5.2.1 SDC Data Collection Service

The proposed framework is provided with Network Flow Collector and Features Extractor modules which implement the capabilities of the SDC Data Collection Service. The Network Flow Collector provides functionalities to capture in real-time the network flows via port mirroring. To limit the impact of mirroring on the network performance, The Network Flow Collector offers the capability of limiting the mirroring to only traffic flowing from/to the monitored asset (i.e., the asset that needs to be protected against application-layer DDoS attack). The collected network traffic is saved into .pcap files. The Features Extractor module is capable of analysing the PCAP files and retrieve flow's features relevant to application-layer DDoS attack detection. The extracted network flow features are saved into a .csv files that will be fed into the DL model incorporated in the anomaly detection service for attack detection.

2.5.2.2 SAE Anomaly Detection Service

The Detector module has the capabilities of detecting malicious patterns due to application-layer DDoS attacks by analysing the network flow features extracted by the Data Collection Service. To this end, The Detector module relies on an DL model built using Multi-Level Perceptron (MLP) algorithm to detect the anomalous behaviour. If a malicious traffic pattern is identified, the Detector module notifies the Decision Engine (playing the role of anomaly detection service consumer) by sending an alert report containing details on each identified malicious network flow, which include the network flow ID and the prediction confidence level.

2.5.2.3 DE Security Decision Service

To reduce the number of false positives, the implemented decision engine compares the prediction

confidence level to a given threshold. If the prediction confidence level exceeds the threshold, the network flow is considered malicious and a security policy (e.g., flow dropping or steering), expressed using Medium-level Security Policy Language (MSPL), is issued to the Security Orchestrator for mitigation.

2.5.3 Sequence diagrams

Figure 12 illustrates the workflow of application-layer DDoS detection.

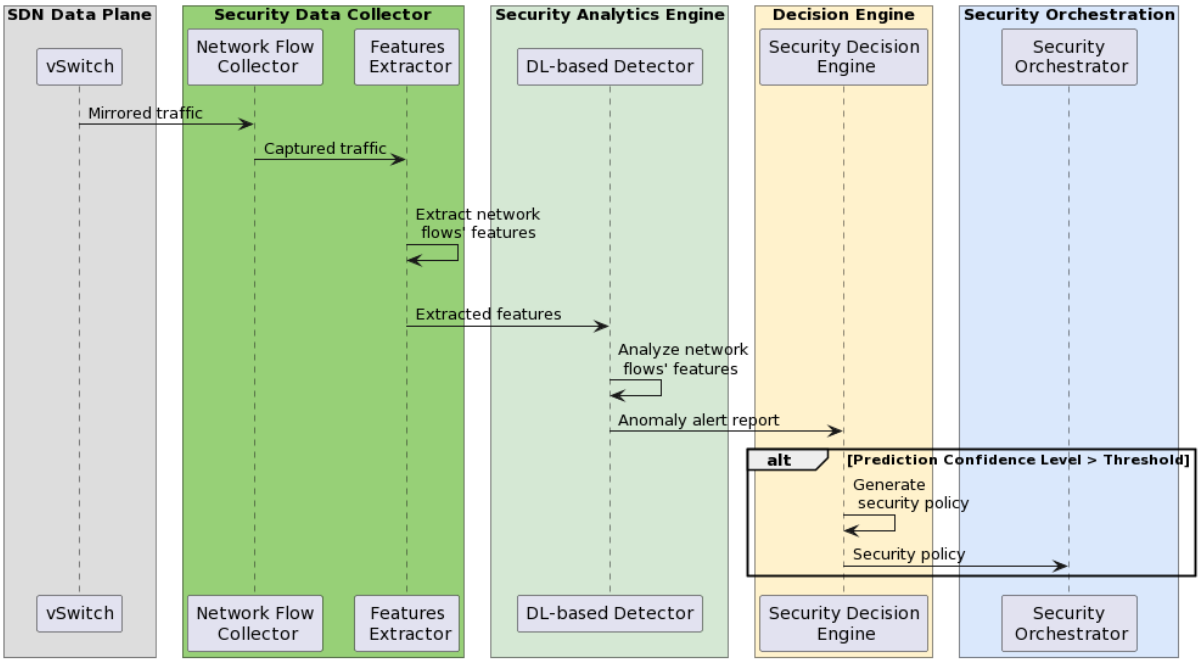

Figure 12: Application-layer DDoS detection workflow

The sequence diagram illustrated in Figure 12 shows the main interaction between the components of the application-layer detection framework. The Network Flow Collector permanently collects network flows via port mirroring. The collected traffic is periodically exported to Features Extractor to retrieve flow's features relevant to application-layer DDoS attack detection. Once extracted, the flow features are passed to the Detector for uncovering suspicious behaviour using a DL model. The DL model is trained to distinguish both high-rate and low-rate application-layer DDoS attacks based on supervised learning approach. Furthermore, the DL model can deal with application-layer DDoS attack even in presence of adversarial attacks, thanks to adoption of adversarial training defence approach. In adversarial training, the DL model is explicitly trained on adversarial examples, generated using white box attack Fast Gradient Sign Method (FGSM)[19], in order to learn how to resist them.

An anomaly alert report is generated by the DL-based Detector and submitted to Security Decision Engine for taking final decision about the maliciousness of a network flow by comparing the detection confidence level reported in the alert report against a pre-set threshold. If the network flow maliciousness is confirmed, the Security Decision Engine uses the malicious network flow's details (e.g., source IP, source Port, protocol ID) enclosed in the alert report to generate a security policy that will be enforced by the Security Orchestrator to block the malicious source.

2.5.4 Technical implementation

The technical implementation details and performance evaluation of the proposed framework are

[19] I. J.Goodfellow, J. Shlens, C. Szegedy. Explaining and Harnessing Adversarial Examples.CooR, abs/1412.6572, 2014

documented in 18. The effectiveness of the DL-Detector against adversarial attacks is presented in 20. The communication between the framework modules is performed using REST APIs. The attack agents are implemented using Hulk21 tool for high-rate DDoS attacks and Slowloris[22] tool for low-rate DDoS attacks. The Cleverhans[23] library is used to craft adversarial application-layer DDoS flows based on FGSM attack. The Application-layer DDoS Detector will be part of the INSPIRE-5Gplus Demo1 implementation; it will be deployed in OULU's Security Management Domain (SMD), providing fully automated detection and mitigation of applications-layer DDoS attack against a video on-demand service. In our implementation, the Security Orchestrator converts the policy into a flow dropping intent and sends it to the SDN controller. Based on the received flow command, a drop flow rule is pushed by the SDN controller to the corresponding virtual Switch (vSwitch) to fulfil the defined security policy.

2.6 Application-Layer DDoS Mitigation

2.6.1 Overview

As introduced in the previous section, DDoS attacks are becoming stealthier, making their detection a challenging task which may not be possible using the characteristics collected from network flows. In a scenario with different network slices deployed on the same infrastructure to provide 5G services to end users, an undetected application-layer DDoS attack may hinder not only the availability of the service provided by the slice under attack, but also the other slices co-hosted on the same infrastructure by exhausting shared resources (e.g., CPU, RAM, etc.) through exploitation of auto-scaling capability. In fact, while auto-scaling is seen as a mean to mitigate the workload that could be caused by DDoS attack if this last can escape detection using the DDoS Detector, it may reshape a (D)DoS attack into an Economical Denial of Sustainability (EDoS) attack, which incurs economic damages to service provider due to the increased elastic use of resources as well as performance degradation due to shared resource starvation.

To address this issue, we propose a new solution that leverages DL techniques to discriminate legitimate auto-scaling requests due to flash events from malicious auto-scaling requests caused by application-layer DDoS attacks. The solution includes different modules that allow to automatically (i) monitor the resource usage and performance metrics of the slice's VNFs; (ii) analyse the collected metrics using a DL model to identify anomalies in resource usage and performance metrics; and (iii) block malicious auto-scaling requests if an anomaly is detected.

2.6.2 Role in the HLA

Figure 13 illustrates the components involved in the implementation of the applications-layer DDoS mitigation solution.

[20] C. Benzaid and T. Taleb,"ZSM Security: Threat Surface and Best Practices," in IEEE Network Magazine,Vol. 34, No. 3, Jun. 2020, pp. 124 - 133.

[21] https://github.com/grafov/hulk

[22] https://github.com/gkbrk/slowloris

[23] https://github.com/tensorflow/cleverhans

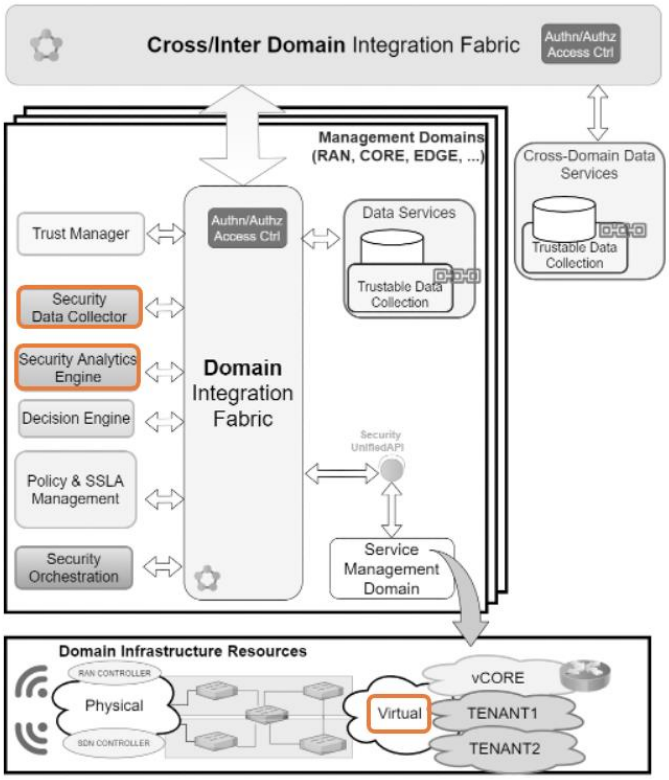

Figure 13: The INSPIRE-5Gplus HLA components involved in DDoS Mitigator enabler

2.6.2.1 SDC Data Collection Service

The proposed solution provides a Monitoring System which implements the capabilities of the SDC Data Collection Service. The Monitoring System offers the capabilities to generate on-demand the dataset for training and testing the anomaly detection model or continuously collect the resource usage and performance metrics values observed in the last x time steps in order to be fed into the Anomaly Detection Service for real-time anomaly detection. Note that the metrics are extracted as time series in CSV files.

2.6.2.2 SAE Anomaly Detection Service

The DDoS Mitigator is built as an anomaly detection model using the unsupervised learning technique LSTM-based Autoencoder on multivariate time series. The LSTM-based AutoEncoder model is trained to reconstruct time-series for normal behaviour. The inputs to the model are the features related to resource usage (e.g., CPU usage, system load, memory usage, I/O network traffic) and system performance (e.g., http response time) collected by the Monitoring System. An anomaly is detected if the reconstruction error is above a given threshold. The DDoS Mitigator generates an anomaly predictions report containing the computed anomaly score for each timestep in the analysed time series, and the decision on whether the values of the VNF's resources usage and performance metrics for a given timestep are anomalous or not.

2.6.3 Sequence diagrams

Figure 14 illustrates the workflow of application-layer DDoS mitigation.

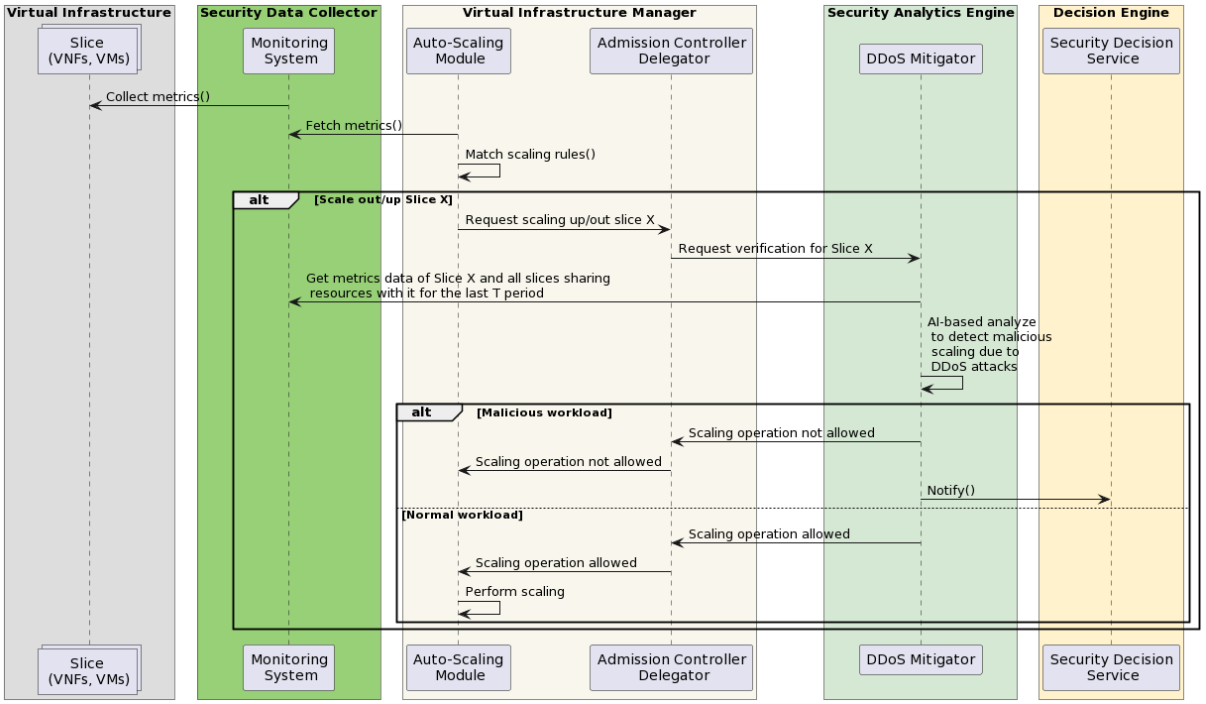

Figure 14: Application-layer DDoS Mitigation workflow

The sequence diagram illustrated in Figure 14 shows the main interactions between the components of the application-layer detection solution. The Monitoring System scrapes periodically the resource usage and performance metrics from the slices' VNFs and their hosting computing nodes via deployed probes. When a scaling-up/out request is issued by the auto-scaling module, the request is intercepted by Admission Controller Delegator which delegates the scaling decision to the DDoS Mitigator module for validation. The DDoS Mitigator uses a DL-based anomaly detection model which can effectively identify anomalous resource usage and performance metrics caused by both high-rate and low-rate application-layer DDoS attacks. If an anomaly is detected, the scaling operation is refused, and the Decision Engine is (optionally)informed to take further measures to mitigate the attack. This could include analysing network traffic to identify the origin of the attack and block the attackers, and/or retraining the DDoS detection system on the new malicious network traffic to improve its capabilities in detecting the attack in the future.

2.6.4 Technical implementation

We built a testbed consisting of two interconnected OpenStack cloud platforms (see Figure 15). A Kubernetes (K8s) cluster with one master node and three worker nodes is set up using four VMs managed by OpenStack. We considered as a case study a virtual Content Delivery Network (vCDN) service deployed as a slice at the edge. In our implementation, the vCDN slice is composed of two Cloud-native Network Functions (CNFs), namely a video streamer and a cache, chained together to provide an HTTP-based on-demand video streaming service. The two CNFs are deployed as K8s services running a NGINX web server and are distributed along two worker nodes. Note that only the streamer service is exposed to the end user for content delivery. To ensure resource isolation between slices, each vCDN slice instance has its own namespace. A fifth VM, running on the second OpenStack cloud, is used to deploy the Monitoring System and the DDoS Mitigator module for anomaly detection. We conducted an experiment where we focused on detecting application-layer DDoS attacks against a vCDN slice by identifying anomalies in the resource usage and performance metrics of vCDN's CNFs.

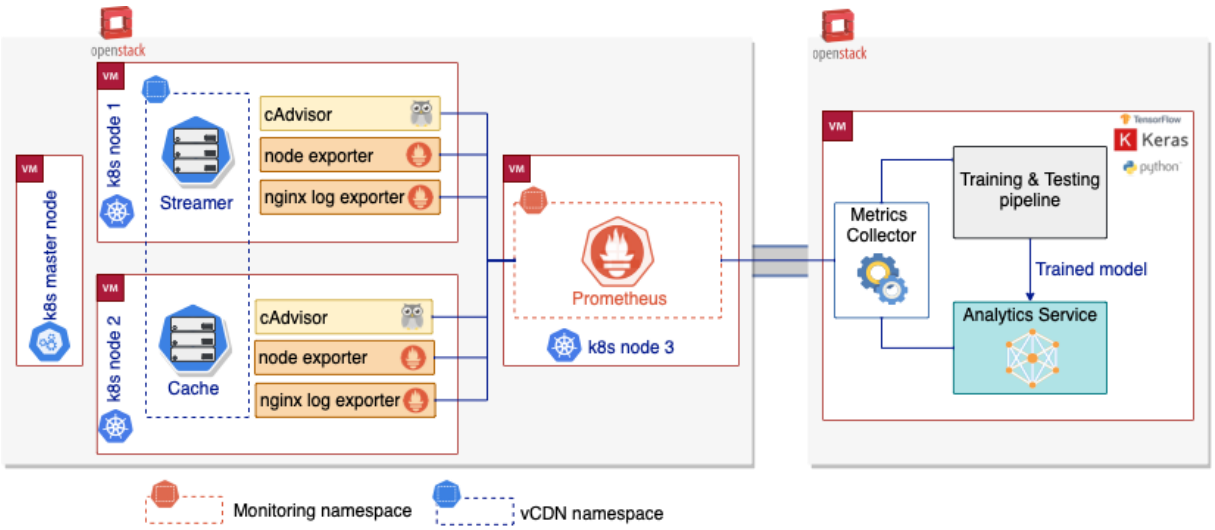

Figure 15: Testbed setup for DDoS Mitigator

The attack agents are implemented using Hulk tool for high-rate DDoS attacks and Slowloris tool for low-rate DDoS attacks. The Monitoring System is implemented using Python and leverages Prometheus API to extract metrics relevant to anomaly detection. For this purpose, Prometheus relies on NGINX-to-Prometheus log file exporter, cadvisor[24], and node-exporter to scrape metrics related to NGINX server, vCDN slice's CNFs and their hosting Worker nodes, respectively.

The preliminary results, reported in D3.3, showed the effectiveness of the DL-based DDoS Mitigator in detecting anomalies related to the launched Hulk and Slowloris attacks. However, we noticed that false positive alarms are also generated. To address this issue, further refinements have been conducted to tune the anomaly threshold for ensuring effective anomaly detection while reducing the false positive alarms. The results depicted in Figure 16 show that the current version of the DL-based DDoS Mitigator model (Figure 16 (b)) outperforms the previous version (Figure 16 (a)) by considerably reducing the number of false positives.

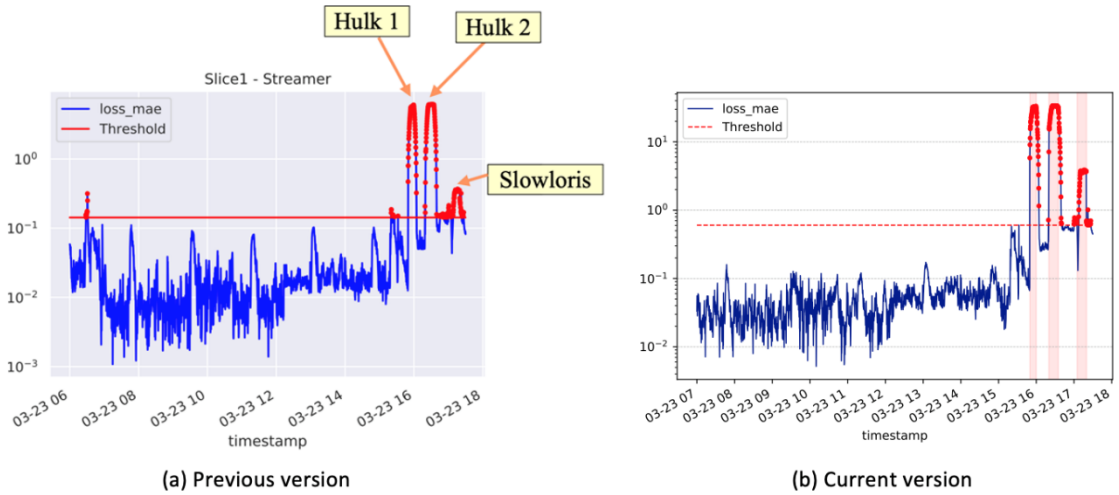

Figure 16: Comparison of anomaly detection performances between the previous version and current version of the DL-based DDoS Mitigator

2.6.5 Summary, lessons learned and guidelines

For DDoS Detector/Mitigator, we intend to investigate the potential of Federated Learning to enable collaborative detection of stealthy DDoS attacks through sharing of knowledge learned locally among

[24] https://github.com/google/cadvisor

cooperating models. The use of FL will also help in fostering data privacy preservation, as the exchanged information is only limited to the model parameters without the need for exchanging any raw data.

2.7 DDoS detection in multi-tenant/domain environment

2.7.1 Overview

5G technology, as it is widely known, will bring very significant improvements in network performance: better bandwidth in terms of connection speeds, a very noticeable reduction in latency, the possibility of connecting many more devices to the network per square meter, or a better network structure, on the basis of SDN and NFV techniques, to provide enhanced support to virtual operators, among other many aspects. However, all these advances, although will allow for an endless list of innovative applications, may also be used for malicious purposes. For instance, distributed denial-of-service attacks (DDoS) benefit greatly from improvements in bandwidth, latency, and the ability to handle more concurrently connected devices.

On this line, a system able to capture and analyse traffic patterns and detect different types of denial-of-service attacks in multi-tenant 5G environments in real-time is proposed. Since network traffic sources could belong to different tenants, the system could also work in E2E covering multiple domains. Developed through a full modular approach, the system will manage all aspects from the monitoring of raw 5G multi-tenant traffic to the detection of a potential DDoS through a machine-learning-based detection engine capable of spotting anomalies in traffic patterns using a combination of two well-known unsupervised learning techniques. The output of the system will enable mitigation agents to enforce the proper countermeasures and alleviate the effects of a potential attack in real-time.

2.7.2 Role in the HLA

In Figure 17, the HLA components related to this asset are highlighted:

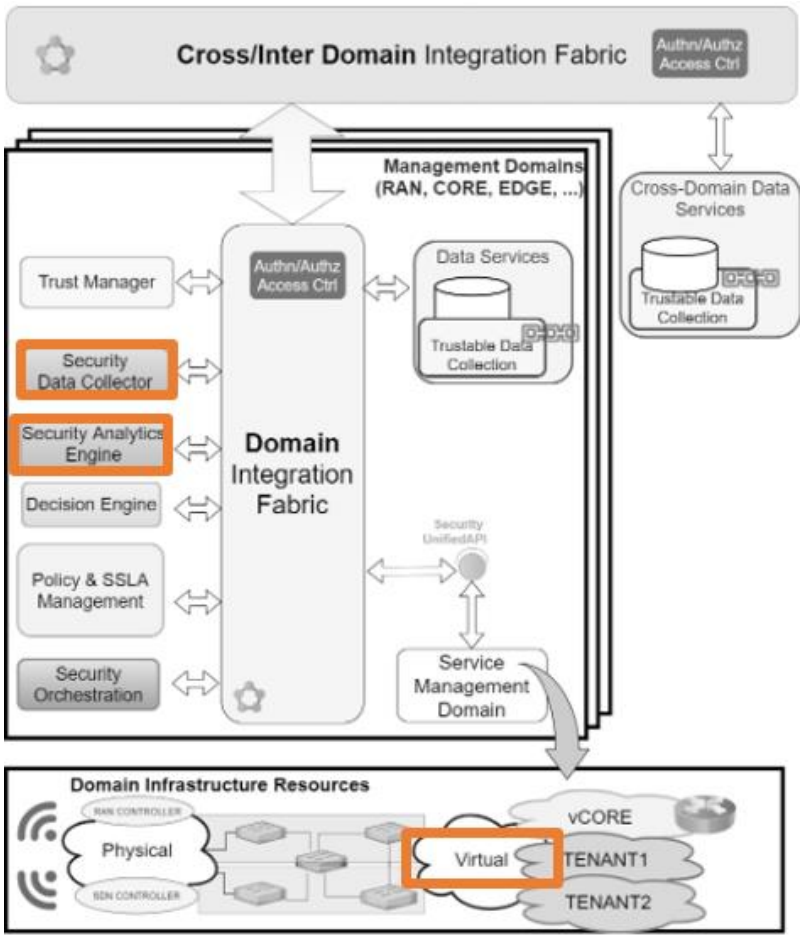

Figure 17: Multi-tenant DDoS detector integration in the HLA

2.7.2.1 SDC Data Collection Service

The system is provided with a monitoring module that covers almost all service capabilities of the SDC Data Collection Service. It is capable of capturing GTP/VXLAN encapsulated multi-tenant 5G traffic in real-time (data capture), extracting substantial information from every captured packet (data extraction) and constructing a JSON-based data model (data translation) that is uploaded, also in real-time, to a cross pub/sub system in order to feed the following modules of the system (data temporal persistence). In addition, this monitoring module offers the option of defining a BPF filter[25] to discard unwanted packets during sniffing phase.

On the other hand, the system also provides a conversations-processing module that, from the information it receives from the previous module, groups packets into conversations and generate a list of 57 DoS-related network features (per conversation) that will be used to feed the final ML-based detection module (data fusion/aggregation capability).

2.7.2.2 SAE Anomaly Detection Service

The ML-based detection engine receives a list of DoS-related network features for every conversation identified by the conversations-processor. Based on that, it offers a joint solution of two unsupervised ML techniques (Gaussian Mixture Models and Autoencoders) to detect anomalies in real-time from conversations data. For every conversation, it generates a final data model containing the conclusion as to whether it is an attack or not. The detection result for every conversation is uploaded to the previously mentioned pub/sub system so that subscribers (e.g., mitigation agents) will receive a notification each time a conversation is classified as an attack. Given these premises, this module is

[25]IBM. (2022). Berkeley Packet Filter (BPF). https://www.ibm.com/docs/es/qsip/7.4?topic=queries-berkeley-packet-filters

considered to cover all the service capabilities of the SAE Anomaly Detection Service (publish results to subscribers and notify consumers of detected anomalies).

2.7.3 Sequence diagrams

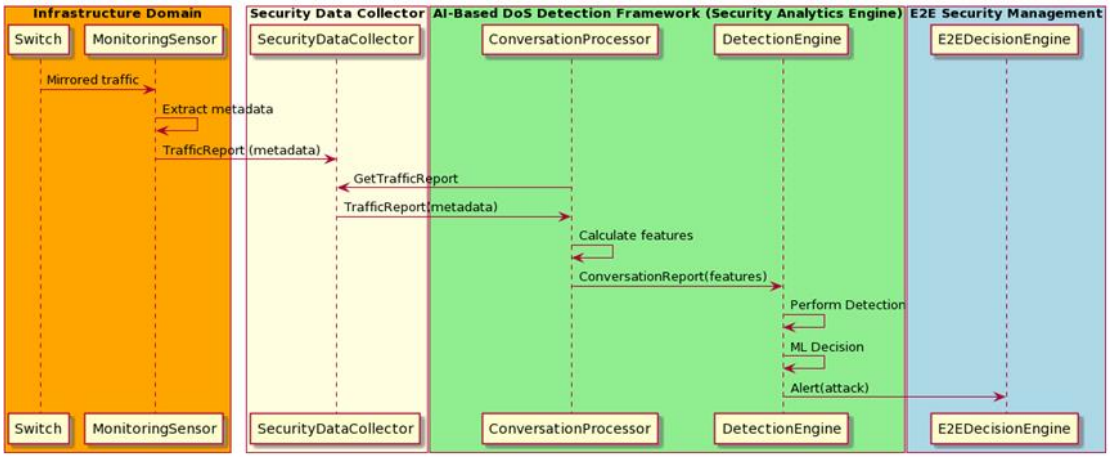

Figure 18: Multi-tenant DDoS detector workflow

In addition to the features shown in the diagram, there are further details of the system which are not covered in it. For example, Figure 18 shows that the conversations-processing component pulls traffic reports as needed from the 'SecurityDataCollector', which runs in a push model. The system is designed to be capable of detecting slow-rate DoS attacks in addition to traditional flooding attacks, and it has proven to do so over application-level encrypted traffic, since all the network features are generated regardless of the application layer. Moreover, the conversations-processing and detection modules are designed to run in a fully distributed way using Apache Spark Streaming, therefore reducing as much as possible the communication delays between the modules as well as detection times. By last, it is also important to add that the ML-based detection module uses only unsupervised-learning models, so the need of labeling the samples during the training phase is completely removed, thus greatly simplifying such task.

2.7.4 Technical implementation

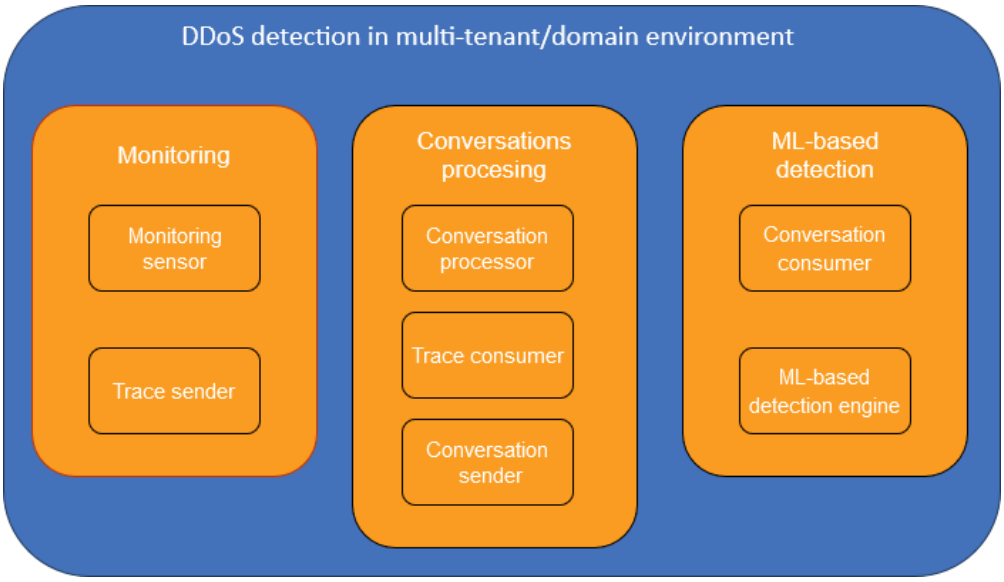

Figure 19: Multi-tenant DDoS detector modules

As shown in Figure 19, the system is composed by three main layers, all connected through Apache Kafka for information exchange. Due to this, each layer is provided with, at least, a module in charge

of communicating with this pub/sub system. In the case of the conversations processing layer, since it needs to either receive packet-related information from the monitoring layer and send conversation-related information to the ML-based detection layer, it implements both consumer and producer or sender modules.

The **monitoring sensor** module, that is part of the monitoring layer, represents the entry point of the system. It is able to capture network packets and extract relevant information for each one of them in real-time. Once a packet has been processed, it creates a Kafka message containing all this information and publishes it to a dedicated topic.

From that point, the workflow continues through the conversations processing layer. The **conversation processor** module receives packet information messages and, based on the source and destination IP, source and destination ports, it groups packets into conversations. Then, for each identified conversation, it calculates a list of features that includes in a Kafka message which will be sent to another dedicated topic, that is being listened to by a consumer module in the ML-based detection layer.

Based on all the features calculated per conversation by the previous module, the **ML-based detection engine**, after a training phase, will perform the attack detection. Therefore, the output of the system will be a boolean value which indicates if a certain conversation is considered as a DoS attack or not.

2.7.5 Summary, lessons learned and guidelines

With this modular approach, each component can be updated and improved independently from the other. In addition, our proposal for the detection engine based on two complementary unsupervised ML techniques allows high levels of attack detection accuracy to be maintained, while not requiring labelled data for model training. Although implemented ML techniques may be changed in other proposals due to specific requirements, we consider that this is one of the most promising mechanisms to include in a system of this kind.

Lessons learned: During the development we have addressed some issues that have to be taken into account for future developments, such as setting a set of types of packets/payloads for inspection in the monitoring sensor, or selecting the best data pre-processing and feature selection techniques in the detection engine.

Future extensions: As future work, we consider it interesting to extend models and data to further kinds of attacks and use Federated Learning in model training phase for better privacy as well as for improved data quality and quantity.

2.8 Anti GPS spoofing

2.8.1 Overview

Unmanned Aerial Vehicles (UAVs) are set to become an integral part of 5G and beyond systems with the promise of assisting cellular communications and enabling advanced applications and services, such as public safety, caching, and virtual/mixed reality-based remote inspection. However, safe and secure navigation of UAVs is a key requisite for their integration in the airspace. GPS spoofing is one of the major security threats to remotely and autonomously controlled UAVs.

In this vein, a machine learning-based and mobile network-assisted UAV monitoring and control system is developed that allows live monitoring of UAVs' locations and intelligent detection of spoofed positions. The system uses the Convolutional Neural Network (CNN) in the edge UAV Flight Controller (UFC) to locate a UAV and detect any GPS spoofing by comparing differences between the theoretical path loss computed by UFC and the corresponding path loss reported by the connected base station (BS). To reduce the detection latency as well as to increase the detection accuracy, transfer learning is leveraged to transfer the CNN knowledge between edge servers when the UAV

handovers from one BS to another. This system is not only designed for UAVs but also for other types of user equipment.

2.8.2 Role in the HLA

Figure 20 illustrates the components involved in the implementation of the Anti GPS Spoofing system.

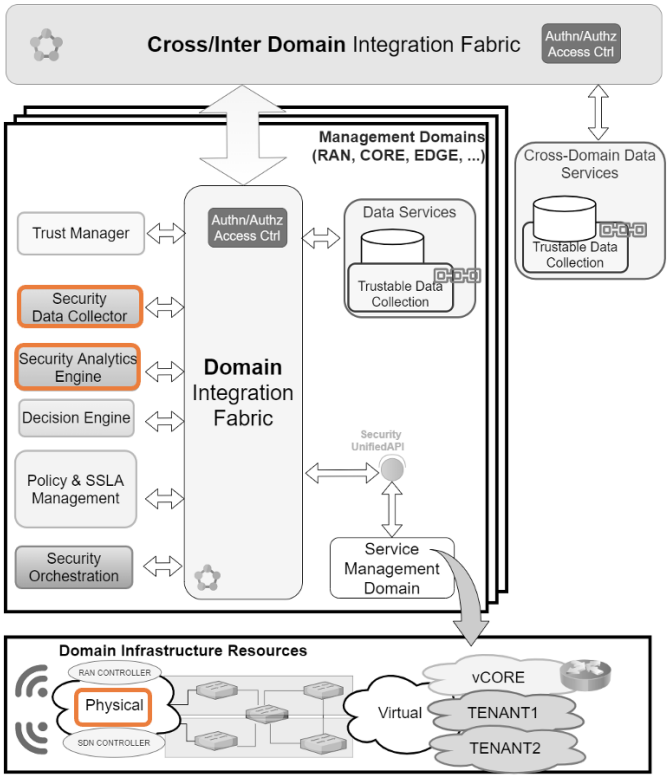

Figure 20: The INSPIRE-5Gplus HLA components involved in Anti GPS Spoofing enabler

2.8.2.1 SDC Data Collection Service

The system is provided with a Location Monitoring module which implements the capabilities of the SDC Data Collection Service. The Location Monitoring module can provide both GPS locations and path loss values for spoofing detection. It collects UAV GPS position data on the edge server from the UAV flight telemetry data, and the path loss data through the physical device, such as base stations. In addition, this module also provides the theorical path loss values computing capability based on base station locations and GPS locations. Specifically, a JSON-based data model, constructed by GPS locations, real-time path losses and theoretical path losses, is sent to anomaly detection service for ML model training and spoofing detection.

2.8.2.2 SAE Anomaly Detection Service

The Anti GPS spoofing system has the capabilities of training and updating ML models as well as detecting spoofed GPS positions using the trained ML models. The ML model training is performed using the data collected from SDC data collection service. Specifically, the Model Trainer module uses a Convolutional Neural Network (CNN) that includes Convolutional layers for feature extraction and Dense layers for the spoofing detection. The Model Updater module leverages Transfer Learning technique to retrain the old model with new data for improving the model performance as well as reducing the model's training time. It is worth mentioning that transfer learning allows sharing the neural network knowledge rather than the raw path losses data, which helps to reduce the amount of data transmission over the network and mitigate network congestion. Finally, the Spoofing Detector module relies on the trained CNN model to detect the spoofed GPS positions. If a GPS spoofing is

detected, the Spoofing Detector can send the landing command to the UAV ground control station, and inform the network controller to release network resources once the spoofing confirmed.

2.8.3 Sequence diagrams

Figure 21 and Figure 22 illustrate the workflow of the training phase and the GPS spoofing detection phase, respectively.

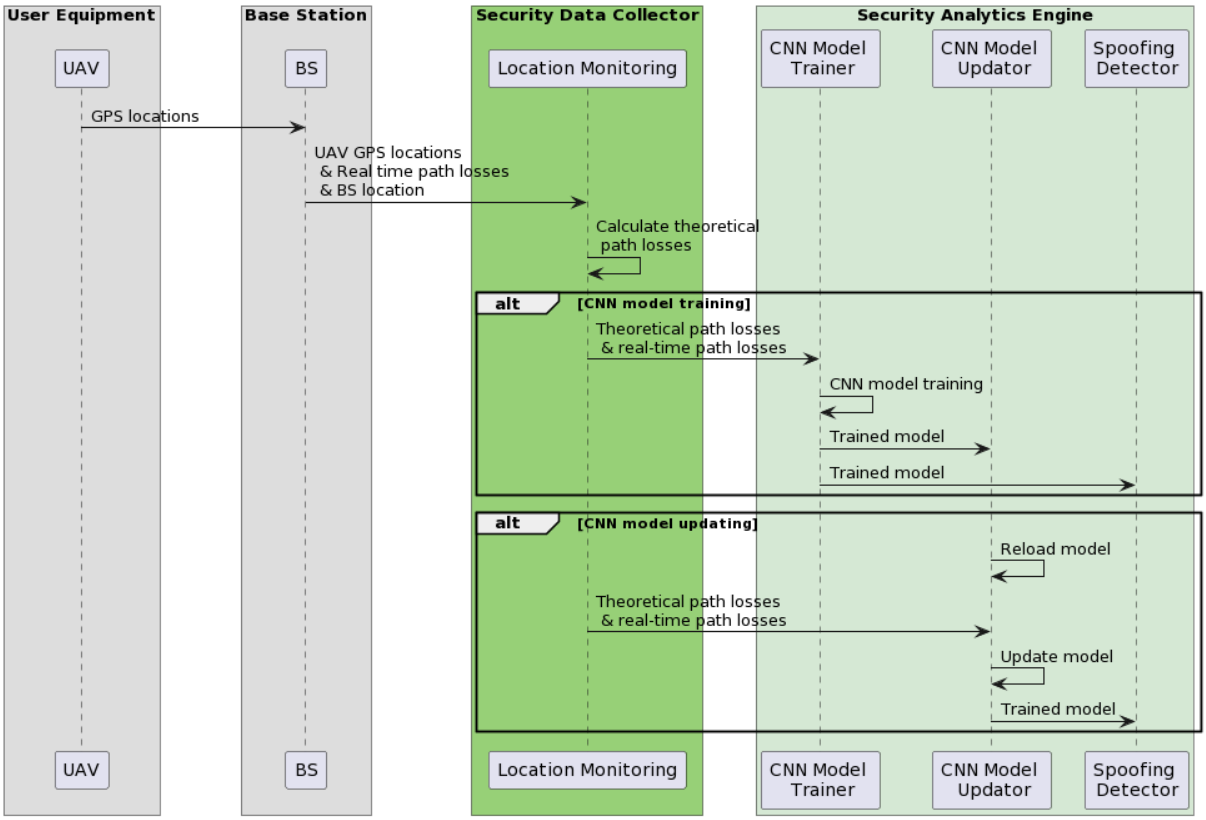

Figure 21 - GPS spoofing detection CNN model training workflow

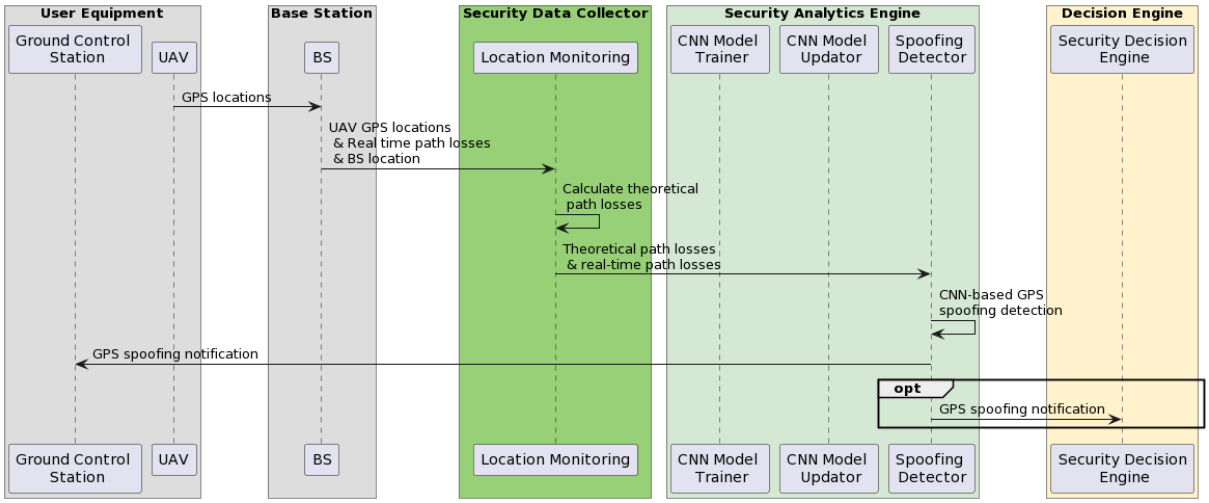

Figure 22: GPS spoofing detection workflow

There are two stages for GPS spoofing detection service, one for the CNN model training (see Figure 21) and the other for spoofing detection (see Figure 22). The model training stage can choose training a new model with raw data or use transfer learning method to update a pre-trained model, where the transfer learning method can reduce the new model training time as well as increasing the detection performance. Specifically, in the first stage, the “Location Monitoring” module provides data to CNN

Model Trainer and Updater modules for training and updating the CNN model, respectively. The data consists of the absolute differences between the real-time path losses reported by the BS and the corresponding theoretical path losses calculated using the BS and UAV locations. The trained model or updated model will be used by the second stage for spoofing detection. In the second stage, the SAE Spoofing Detector uses the trained model to verify the authenticity of the received UAV's GPS position. The Spoofing Detector delivers a report specifying if the location positions are spoofed or not. When the spoofing attack is detected, the UAV will receive a landing command and at the same time the network controller will release network resources once spoofing confirmed (this action could be performed through the Decision Engine).

2.8.4 Technical Implementation

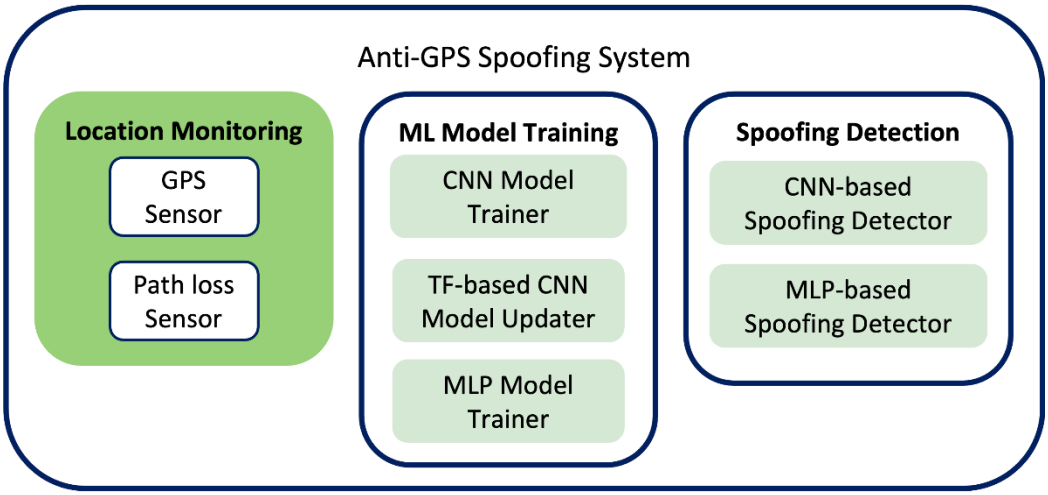

Figure 23: Components of Anti-GPS Spoofing System

As shown in Figure 23, the current implementation of the Anti-GPS Spoofing System is composed of several modules that can be categorized into three blocks based on their capabilities, including, location monitoring, ML model training, and spoofing detection. **For Location Monitoring**, we developed a simulator for data generation, where the theoretical path losses between the BS and the UAV are calculated using the 3GPP path loss model defined in 26. The data simulator can generate dataset for training/updating the Spoofing Detector model or creating data for Spoofing Detector during inference phase.

For ML Model Training, we implemented different modules for training GPS spoofing models using different deep learning techniques. The technical implementation details and performance evaluation of the MLP-based model (already presented in D3.3) and the new TF-based CNN model are documented in 27 and 28, respectively.

The **Spoofing Detector** can use different models to detect spoofed GPS positions. By selecting the model to use and providing the location information, the Spoofing Detector generates a report specifying if the analysed location positions are spoofed or not.

[26] 3GPP, "Enhanced LTE support for aerial vehicles," 3rd Generation Partnership Project (3GPP), Technical report (TR) 36.777, 01 2018. [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?

[27] Y. Dang, C. Benzaid, B. Yang, and T. Taleb, "Deep Learning for GPS Spoofing Detection in Cellular-Enabled UAV Systems," in NaNA 2021, Nov. 2021.

[28] Yanchao Dang, Chafika Benzaid, Tarik Taleb, Bin Yang, Yulong Shen. Transfer Learning based GPS Spoofing Detection for Cellular-Connected UAVs. In Proc. of the International Wireless Communications and Mobile Computing Conference (IWCMC 2022), May 30 – June 3, Dubrovnik, Croatia, 2022. (to appear)

2.8.5 Summary, lessons learned and guidelines

For GPS Spoofing, the proposed solution is devised for one single UAV and does not consider the case of a UAV same. Thus, we are planning to extend the proposed solution to support the detection of GPS spoofing in a UAV swarm. To this end, we will explore the potential of Graphic Neural Networks.

Regarding the Application-Layer DDoS Detector, the Application-Layer DDoS Mitigator and Anti-GPS spoofing enablers:

- Assessing the performances of an ML model only on existing datasets is not enough to validate its effectiveness in detecting attacks. This was the case when we implemented the DDoS Detector. While the model provided high accuracy in discriminating malicious network flows generated by application-layer DDoS attacks on the public dataset CICIDS2017, it failed to achieve the same accuracy when deployed in our testbed. The issue has been solved by using our testbed to generate new training data to augment the one from CICIDS2017. In the future, we intend also to leverage the potential of Generative Adversarial Networks (GANs) for data augmentation. GANs are capable of automatically generating new data that mimic original data with high fidelity
- Training and running ML models, and particularly DL models, are costly in terms of time and resource usage. Through our experience, we noticed that the tuning a model to achieve the desired performances requires to take into account different factors, including the dataset size, the development platform (e.g., Tensorflow, Pytorch, Keras, Caffe, MXNet), the hardware platform (e.g., CPU, GPU, TPU), the ML/DL algorithm (e.g., LSTM, GRU, MLP) and the ML/DL model's hyper-parameters (e.g., the number of hidden layers, the number of neurons in each layer, the learning rate, the batch size, the number of epochs). Reducing the (re)training time of a model is crucial to lower the resource consumption footprint induced by the training process and accelerate its update. Transfer Learning is a solution we explored and implemented to effectively leverage the knowledge of the pre-trained CNN-based anti-GPS spoofing model to improve the detection accuracy while greatly reducing the time spent on training a new model.
- Enabling the ZSM vision of autonomous management in 5G and beyond networks will require running several ML models serving different tasks. This will lead to increased resources consumed by the deployed ML models. To address this issue, we advocate building multi-purpose models; that is, models that can serve different tasks. We start investigation this direction by upgrading the DDoS Mitigator to detect DDoS anomalies based on forecasting errors rather than using reconstruction errors. By doing so, the ML model can server both resource forecasting task and anomaly detection task.

2.9 I2NSF IPsec: encrypted channel protection management

2.9.1 Overview

I2NSF enabler provides the capability of deploying E2E trusted IPsec connections ensuring integrity, confidentiality, and availability based on a Centralized SDN Controller. Basic concepts and technology were described in D3.1, as well of potential applicability on NFV and 5G long term scenarios. To ensure a trusted channel between two endpoints (e.g. between gNB and a 5G Core, between a HPLM and VPLMN in IPX network, etc.) it needs a controller to translate the policies received from the Security Orchestrator (SO) through an API to specific parameters. To this end a YANG data model and its NETCONF implementation was proposed based in the standard RFC 9061[29] for the communication

[29] Marin-Lopez, R., Lopez-Millan, G., and F. Pereniguez-Garcia, "A YANG Data Model for IPsec Flow Protection Based on Software-Defined Networking (SDN)", RFC 9061, DOI 10.17487/RFC9061, July 2021, <https://www.rfc-editor.org/info/rfc9061>

between a Controller and the IPsec engines (agents). Implementation for minimal functional APIs is described in D3.2, jointly with the data model. In this deliverable we introduce the API developed for the interaction between the SO (section 2.12) and the I2NSF Controller, that allows the enforcement of the policies in the "act stage" of the closed loop to provide the necessary automatization mechanisms to handle the deployment and the rekey process during the lifetime of an IPsec tunnel.

2.9.2 Role in the HLA

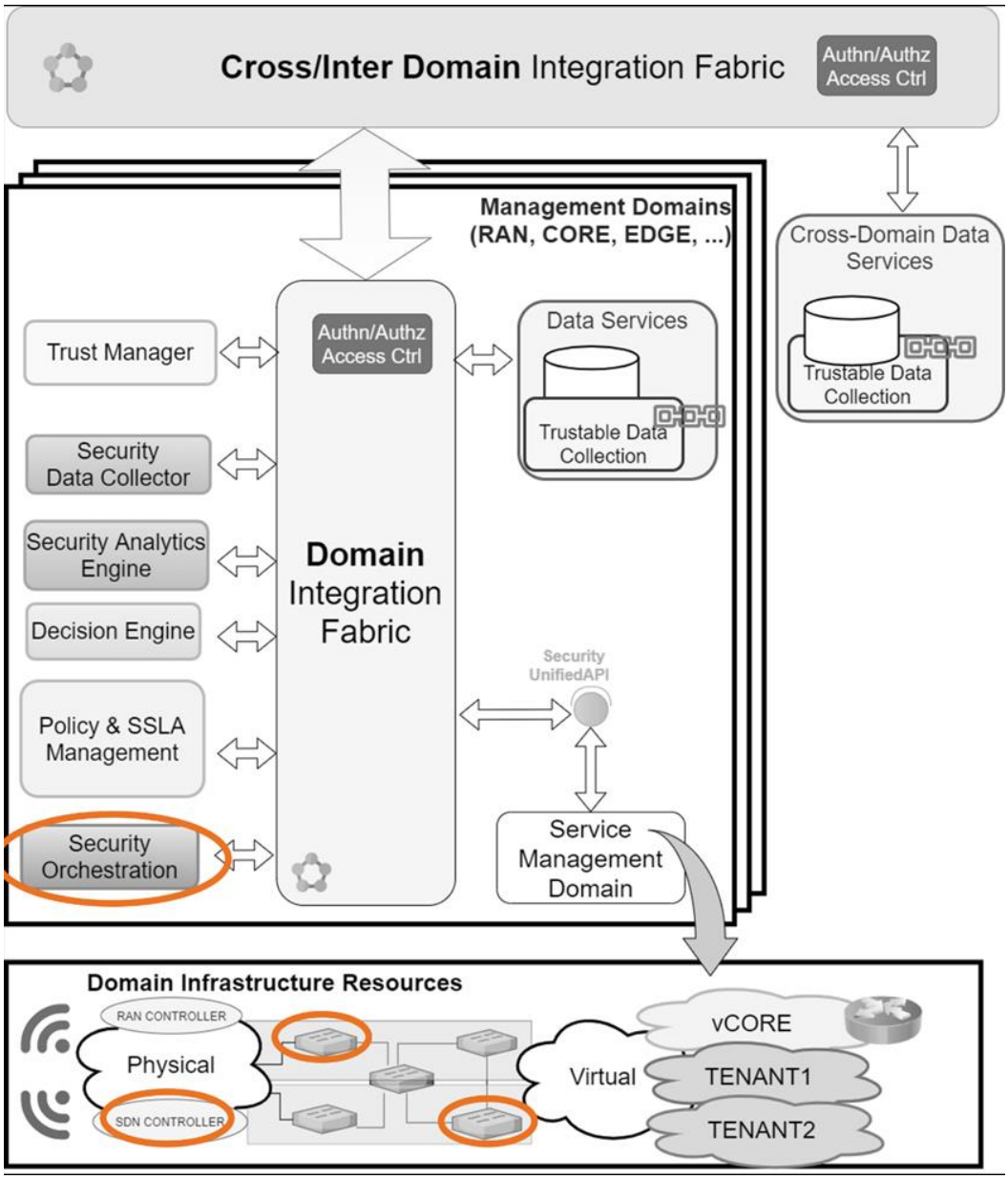

Figure 24: I2NSF IPsec enabler in the INSPIRE-5Gplus HLA

2.9.2.1 Security Agents (SA)

In this solution the I2NSF IPsec agents, take the role of Security Agent (SA). They are in charge of listening requests from the I2NSF controller and translating those requests into specific configurations on the network nodes, which includes the enforcement of the IPsec protocol, with specific received keys and algorithms. As displayed in Figure 24, inside the HLA architecture, the Security Agent are located at the bottom. It also takes care of the periodical refreshment of the keys, creating new IPsec security associations and expiring older ones.

Additionally, one property of the I2NSF agent, is the telemetry capacity, to report the status of the

IPsec tunnel and traffic statistics. This information is delivered to the Security Data Collector and the Trust Manager. The first will feed with network traffic telemetry for security analytics related to traffic patterns to the Security Analytics component. The latter can elaborate relevant reputation metrics based on the security of the IPsec algorithms used.

2.9.2.2 Security Orchestration (SO)

The I2NSF controller extends its own functionality from the Security Orchestrator to the service in the management domain, acting as an application for SDN Controllers in the network infrastructure for specific security agents. In Figure 24, the I2NSF takes the place of the Security Orchestrator in the HLA architecture. The I2NSF controllers can manage specific security policies related to the requirements in terms of number of security agents involved (network nodes) and their connectivity over IPsec, type cryptographic algorithms families to use, the key refreshments, and the activation and deactivation process. The communication between others SO components and I2NSF controller use the Domain Integration Fabric for those messages.

2.9.3 Sequence diagrams

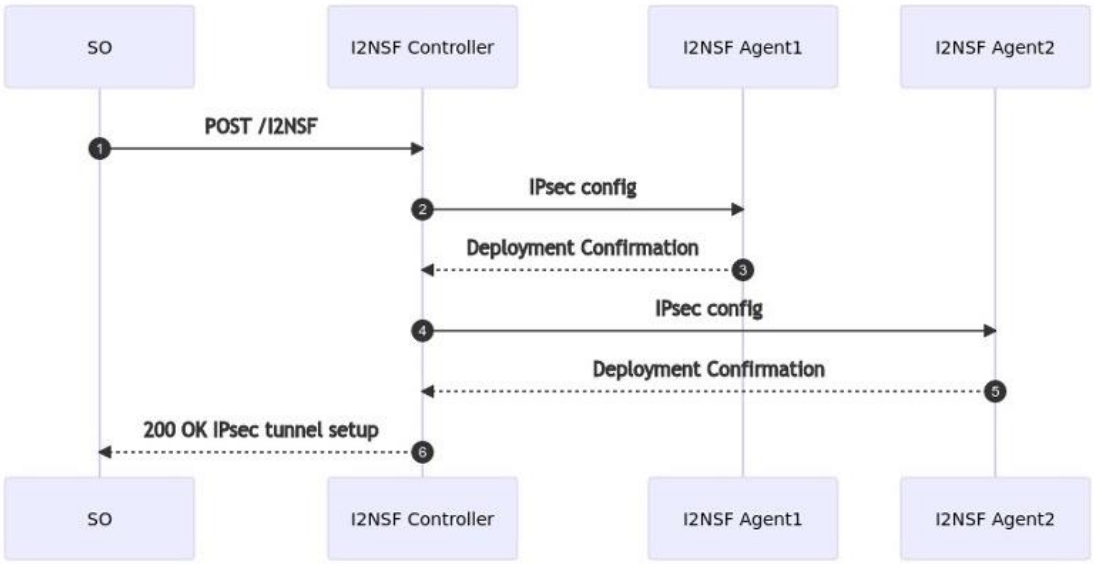

Figure 25: Interaction between the Secure Orchestrator and the I2NSF components

The diagram in Figure 25 represents the interaction between the SO, the I2NSF Controller, and the agents when the channel protection between two endpoints is requested.

First the SO will ask to the I2NSF Controller through an API to deploy the secure channel based on the requested policies. This will start the configuration process where the controller will calculate the necessary parameters to establish the configuration in both agents.

Once both agents have their secure channels running, they will confirm back to the controller that the configuration has been correctly deployed, and both ends of the tunnel are ready.

Finally, the SO will receive a confirmation that the secure channel has been configured and is ready.

2.9.4 Technical Implementation

Over previous implementation several improvements have been made.

API Implementation

A new service has been added to the I2NSF Controller to listen policies request over a new API developed following **OpenAPI 3.0** specification.

Within the API the SO can interact using the following endpoints:

Endpoint	Operation	Description
/i2nsf	POST	Creates an IPsec tunnel.
/i2nsf/{UUID}	GET	Get the status and information of a deployed IPsec tunnel.
/i2nsf/{UUID}	DELETE	Deletes a running IPsec tunnel.

Table 1: I2NSF Controller main API endpoints

To establish the different IPsec tunnels requested by the SO to the I2NSF controller, the latter, through the API, verify and translate the parameters in the request to the standard format of the YANG model defined in D3.2. Figure 26 shows an example of request parameters. The following values are the one included to configure the IPsec tunnel based on the different requirements for the keygen process:

Encryption Algorithm (encAlg): list of encryption algorithms that can be used by the I2NSF agents to establish the IPsec tunnel. The algorithms must be ordered based in the priority.

Integrity Algorithm (intAlg): list of integrity algorithms that can be used by the I2NSF agents to establish the IPsec tunnel. The algorithms must be ordered based in the priority.

Soft Lifetime (softLifetime): time that must pass for the I2NSF agents to launch the rekey process and request new encryption keys.

Hard Lifetime (hardLifetime): max lifetime of an IPsec tunnel when the rekey process has not been done or there was an error during the process. Each time the rekey is achieved the timer is reset.

QRNG Keys (qKeys): this option makes the controller to use a Quantum Random Number Generator to populate the keys used by the I2NSF agents in the IPsec tunnel, instead of local function of random number generator. This optional parameter, is included for high security demands and it has been tested with an external QRNG entropy provider[30].

Also, the controller needs the following information about the nodes:

Internal network (networkInternal): network where the client stands, needed since the agent is running IPsec **Tunnel Mode**.

Control IP (ipControl): address of the node from where the controller is going to send the different requests.

Data IP (ipData): address where the agent is going to be listening to the IPsec packets (those with the ESP header).

NAT functionality:

Since each security agents need to know from which interface and IP address are going to be listening to receive the IPsec packets, this could differ from the public one exposed to the internet if the machine is behind a NAT. Some routers may have the option to redirect incoming IPsec ESP traffic to a specific machine behind NAT.

In such cases, the option “ipDMZ” needs to be specified as the public IP of the interface that is receiving the redirect traffic from the router.

[30] https://www.qrypt.com/

```
{
"nodes": [
    {
      "ipControl": "10.204.4.165",
      "ipData": "155.54.95.206",
      "networkInternal": "10.205.55.0/24"
    },
    {
      "ipControl": "10.204.4.63",
      "ipData": "2.139.184.156",
      "ipDMZ": "192.168.0.153",
      "networkInternal": "10.1.1.0/24"
    }
  ],
  "encAlg": [
    "des"
  ],
```

Figure 26: Example request with the different parameters

Rekey process:

Both agents are running IPsec using the **Ikeless** option. This means that there is no key exchange mechanism between both machines like IKEv2, since keys will be assigned by a third party, in this case the controller. This is designed to provide a centralized control to the Security domain administrator, for example in the key renew frequency, key sizes, algorithms accepted, etc., avoiding the common problem of provision a key and forget forever.

The controller must update the policies of the tunnel with new keys if a notification is received when the soft lifetime expired is received. During this process, the controller will generate new keys which will be sent with the new IPsec policies. Once the agents have confirmed that the new configuration is setup, the controller must remove the previous version of the policies. This process is shown in next Figure 27.

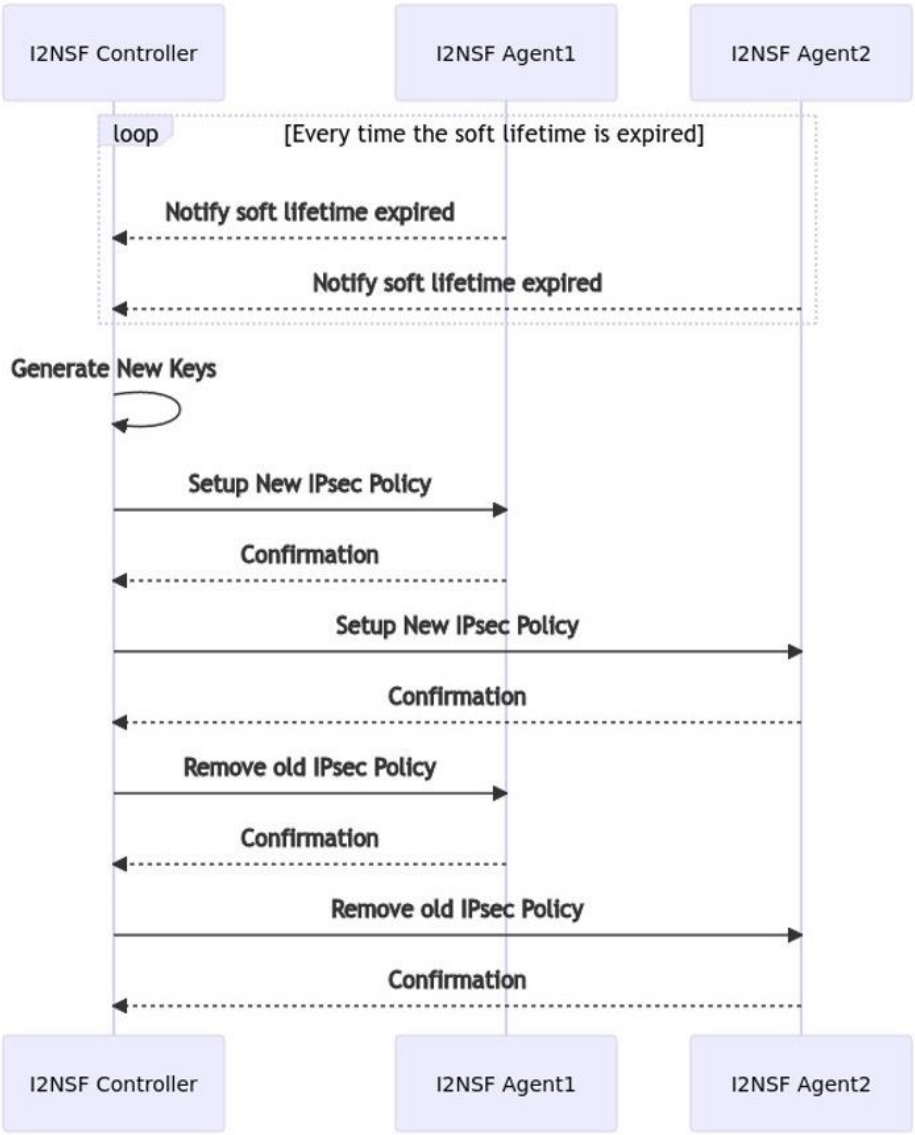

Figure 27: Rekey process between I2NSF components

2.9.5 Summary, lessons learned and guidelines

Guidelines

I2NSF controller is strongly modular so it can be integrated into different network architectures. It can be deployed as a VNF in an NFV infrastructure to setup and control internal network communications in a datacenter that requires multipoint IPsec connections, over the application layer on an SDN Transport Controller for networks to attend northbound interface requests, or embedded as a software component into the Security Orchestrator. The first and second ones, requires the use of API interfaces. Related to the interactions in southbound interfaces between I2NSF Controller and Security Agents, the solution is based on standard (RFC 9061) so it can be integrated with any network components that support NETCONF protocol, such as VNFs or physical devices.

Lessons learned

Support of dynamically and programable encryption services, such as IPsec, and the strict policies related to key renew or cypher suites, are valuable capabilities in future complex multidomain solutions for 5G connectivity. Moreover, the capacity to provide easily relevant information about this functionality and their states, enrich aspects not considered until now such as network trust and reputation.

Future work

The relevant activities envisioned are: Improvement in functionalities related to the RFC9061, such as IKEv2 support; integration with Trust Execution Environment (TEE) to protect key material at execution time; and support QKD technology and interfaces for high security demanding use cases, for example in operators back-haul communications.

2.10 PyrDE: hierarchical security policies management

2.10.1 Overview

The Pyr Decision Engine (PyrDE) is an implementation of some of Decision Engine HLA functionalities. It focuses on reactive adaptation and reconfigure some security deployments depending on live alerts. The Pyr prefix was added to differentiate this implementation with the HLA functional block.

2.10.2 Role in the HLA

The Pyr Decision Engine components relation with the HLA are shown in the Figure 28.

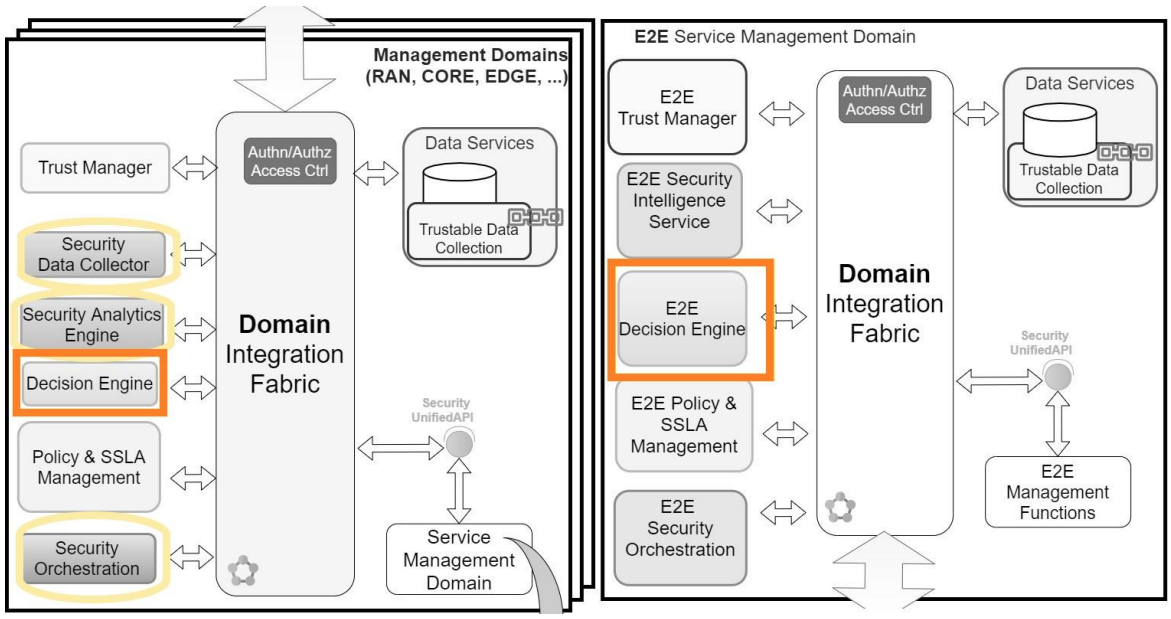

Figure 28: PyrDE overall HLA role

2.10.2.1 DE Security Decision Service

The Pyr Decision Engine provides an API to define the security mitigations. This API is focused on reactive adaptation. Some events or alert trigger the reaction mechanism as the input. A reaction is selected from the event. The output is a manifest push to the Security Orchestrator to enforce the reaction. In the context of the INSPIRE-5Gplus project, this manifest takes the form of an MSPL xml file, pushed using a REST API. This Service should allow the creation of new reaction, their deployment and destruction. Moreover, this API should support the adequate metadata for the priority service described in the next sibling section. Local SMD domains may share common reaction but can also use specific localized reactions.

2.10.2.2 DE Security Decision Priority Service

The Pyr Decision Engine has a notion of priority attached to the reactions to manage decision conflicts. At runtime, a bare Decision Engine could emit multiple reactions from the same alerts. Those may conflict each other by having contradictory side effects. For example, a reaction may disallow a device while another may allow the same device. This fluctuation can affect the applications, as the infrastructure is always fluctuating. This additional protection against this fluctuation is to enforce

some grace period around reactions for extra stability.

2.10.2.3 E2E_DE Security Decision Service

This service is an extension of the equivalent SMD service describe in the previous section. Inside the E2E domain, the domain can be configured with local reaction specific to the domain. Those reactions can affect the E2E domain alone. However, the E2E Security Decision Service can also target the underlying SMD, either by using the synchronization service or by delegating the domain enforcement to the E2E Security Orchestrator. This service enables reactions to be pushed downward to the local SMD.

2.10.2.4 E2E_DE Security Decision Priority Service

The Decision Service at the E2E level shares the same implementation as the local SMD one but it manages an extra layer of priority. As the E2E level is viewed as the top controlling layer, it has by design a total control over the underlying SMD. As such, in the scope of the E2E domain, the priority system inside the E2E DE works the same as the previous DE Security Priority Service. However, for the same raw priority level, the decisions taken at the E2E level have a greater priority compared to the SMD one. Thus, when the decision is enforced from the top E2E level down to the SMD without using the E2E Security Orchestration component, the decision moving downward must be augmented to encompass a greater priority.

2.10.2.5 E2E_DE Security Decision Synchronization Service

The E2E Pyr Decision Engine receives escalation alert from the local SMD DE. Each local SMD DE has the possibility to roll a local reaction given a local alert. Thus, the local SMD DEs need a way to communicate their local mitigation to the E2E layer. Sometime, the local SMD DE may not find adequate reaction at all for a given local event. This service is need to synchronize the local decision up to E2E layer. Moreover, when the E2E Decision Engine does not delegate the multi-domain management to the E2E security Orchestrator, it lacks a way to enforce the E2E reaction back to the underlying domains. Such the service is bi-directional when an E2E Security Orchestrator is not present in the framework.

2.10.3 Sequence diagrams

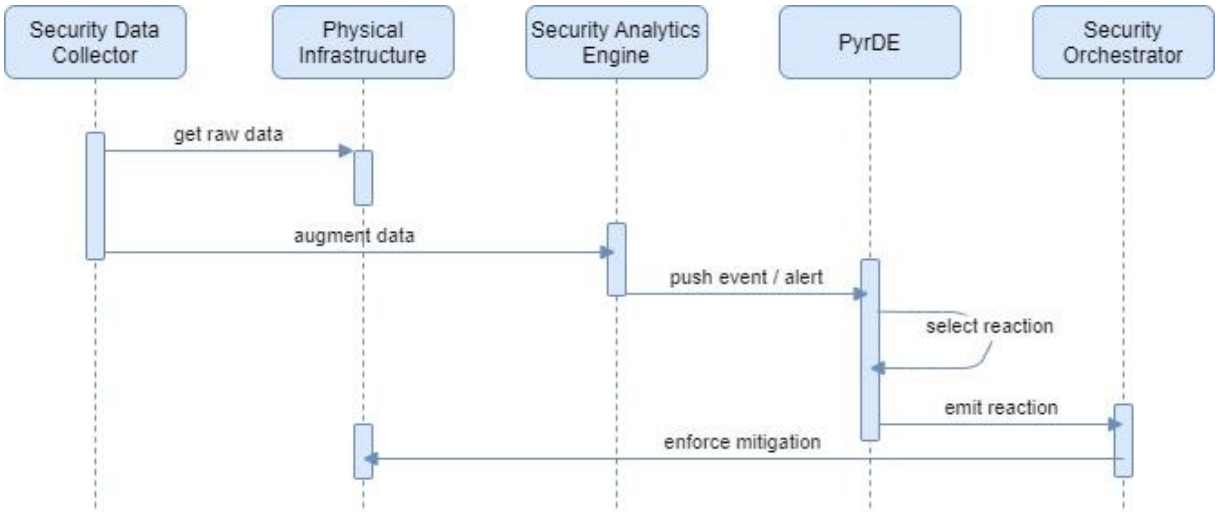

Figure 29: PyrDe HLA workflow

The Figure 29 shows the overall PyrDE workflow in the context of the HLA architecture. For the evaluation in the local POC testbed, the PyrDE manages the security in front of running applications, for example brute-force attack against a SSH server. The underlying infrastructure network is built

using Open vSwitch[31] (OVS) switches, which are software routers compatible with the OpenFlow[32] protocol for configuration. The Security Data collector is a probe using PCAP[33] that capture packet and matches against known attack signature. The Securiy Analytics Engine is backed by ELK[34] for alerts storage and correlation. The Security Orchestrator is high-level SDN controller with a small intent-based REST API that translate security rules (such as blocking an IP, or a flox) into some Faucet manifest. Faucet is the low-level SDN controller that generate the OpenFlow rules and manages the switches. In this small setup, the PyrDe was successfully evaluated by triggering the security rules during attacks.

2.10.4 Technical implementation

The PyrDe is implanted in Golang[35] and provide a REST api using the Echo[36] framework. The PyrDEs at the E2E and SMD level share the same REST API. A local CLI can interact with the PyrDE locally and manages the local reactions. The CLI can also be use on the E2E level to handle the remote reactions running on the SMDs. The APi is published on github[37].

In its design, one defining point is the usage of separate Docker containers for local reactions generation. The PyrDE manages the creation, on-boarding of the local reactions by handling these Docker containers. Those containers have the responsibility to generation the mitigation. A local reaction is described with an augmented docker-compose manifest. The reaction is a set of containers seen as a “black-box” pipeline: in the point of view of the PyrDE the inner details on how the reaction is made or chosen is not known. Some extras metadata fields inside the docker-compose manifest allow describing:

- What kind of alerts are compatible with the reaction.
- The start of the pipeline that consumes the alerts.
- The priority associated with the reaction.

The Figure 30 shows an example of a reaction pipeline manifest.

```
version: "3.9"
x-reaction-meta:
  # The ranking between all reactions
  priority: 1
  # The type of supported alerts
  supported-alert-kinds: ["crypto"]
  # Entrypoint to post alert
  input-endpoint:
    service: "reaction1"
    port: 8080
```

[31] https://www.openvswitch.org/

[32] https://opennetworking.org/sdn-resources/customer-case-studies/openflow/

[33] https://www.tcpdump.org/

[34] https://www.elastic.co/what-is/elk-stack

[35] https://go.dev/

[36] https://github.com/labstack/echo

[37] https://github.com/INSPIRE-5Gplus/i5p-hla-api

```
    prefix: "my/api"
  # The type of output generated by the reaction
  output-kind: ["mspl"]
  # The Env variable pointing to the HTTP endpoint for output submission
  output-receiver:
    service: "reaction1"
    variable: "TARGET"
services:
  reaction1:
    image: "i5g+/reactions1:latest"
          command: "./startreaction --submit-to $TARGET"
```

Figure 30: PyrDe reaction pipeline manifest

The PyrDE is deployed in two flavors: an E2E PyrDE and some SMD PyrDE. The laters control each SMD domain respectively. The E2E PyrDE is just an augmented SMD one.

When a SMD PyrDE starts, it sanitizes the host to reach a stable state. The PyrDE as only one side effect on the host, which is the execution of side Docker containers to run reactions. The containers are tagged using a challenge mechanism. The name of the container is signed using a private static key and store in a label. The PyrDE owns any containers that validate this challenge. As the SMD PyrDE initializes, it lists the current containers, check the label and it removes all owned zombie containers. This small mechanism allows to use the PyrDE safely on host also using containers for other services (such as Kubernetes).

The PyrDE running in a SMD domain must register to the hierarchical E2E PyrDE (see Figure 31) using a REST call. This step allows the discovery of the components. The E2E PyrDE is behind a statically known DNS name, handled by the Integration Fabric and it stores the newly connected SMD PyrDE inside the internal state.

The result of a local reaction is a mitigation. The “blackbox” pipeline computes this mitigation manifest, following the API constraint of the underlying Security Orchestrator. It is injected inside the SMD PyrDE alert handler, which means that another reaction can listen to a mitigation event and act accordingly. For example, an SMD PyrDE can register a local reaction inside a SMD and forwards the computed mitigation event to the E2E domain.

The Figure 31 also displays how alerts are manipulated. The alert events arriving from probes or SAEs (such as ELK) are sent to the alert handler of the PyrDE. Then the PyrDE broadcast the alert to each reactions Docker containers compatible with the alert schema. The alerts are also forwarded to the E2E layer where they follow the same workflow.

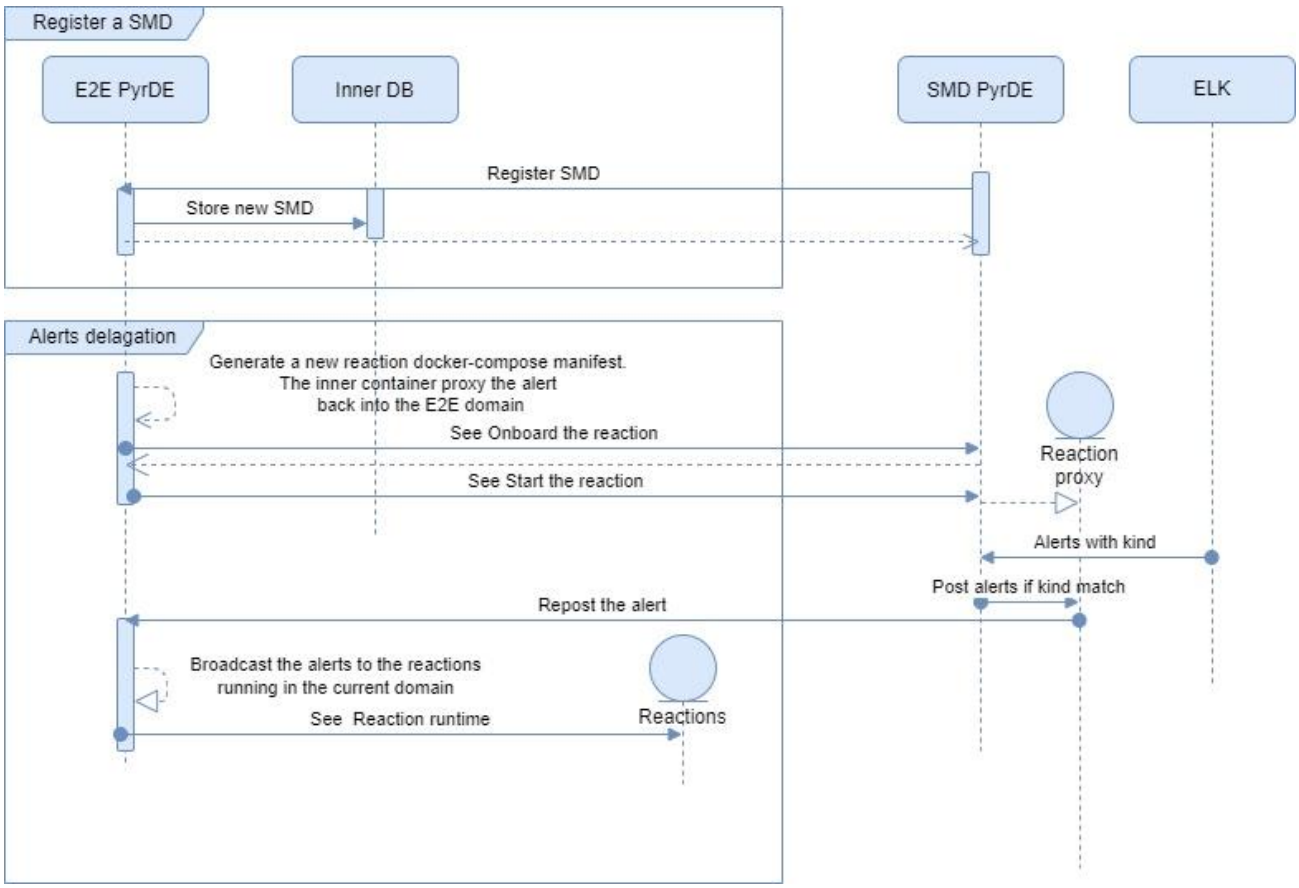

Figure 31: SMD DE registration

A separate call to the local PyrDE enables or disables the local reaction. The Figure 32 describes the starting mechanism of a reaction. A start consists of retrieving the augmented docker-compose manifest, populating the required environment variable and launching the containers using docker-compose. An update to the internal alert proxy inside the local PyrDE configures the broadcast of the desired alert kind to the reaction.

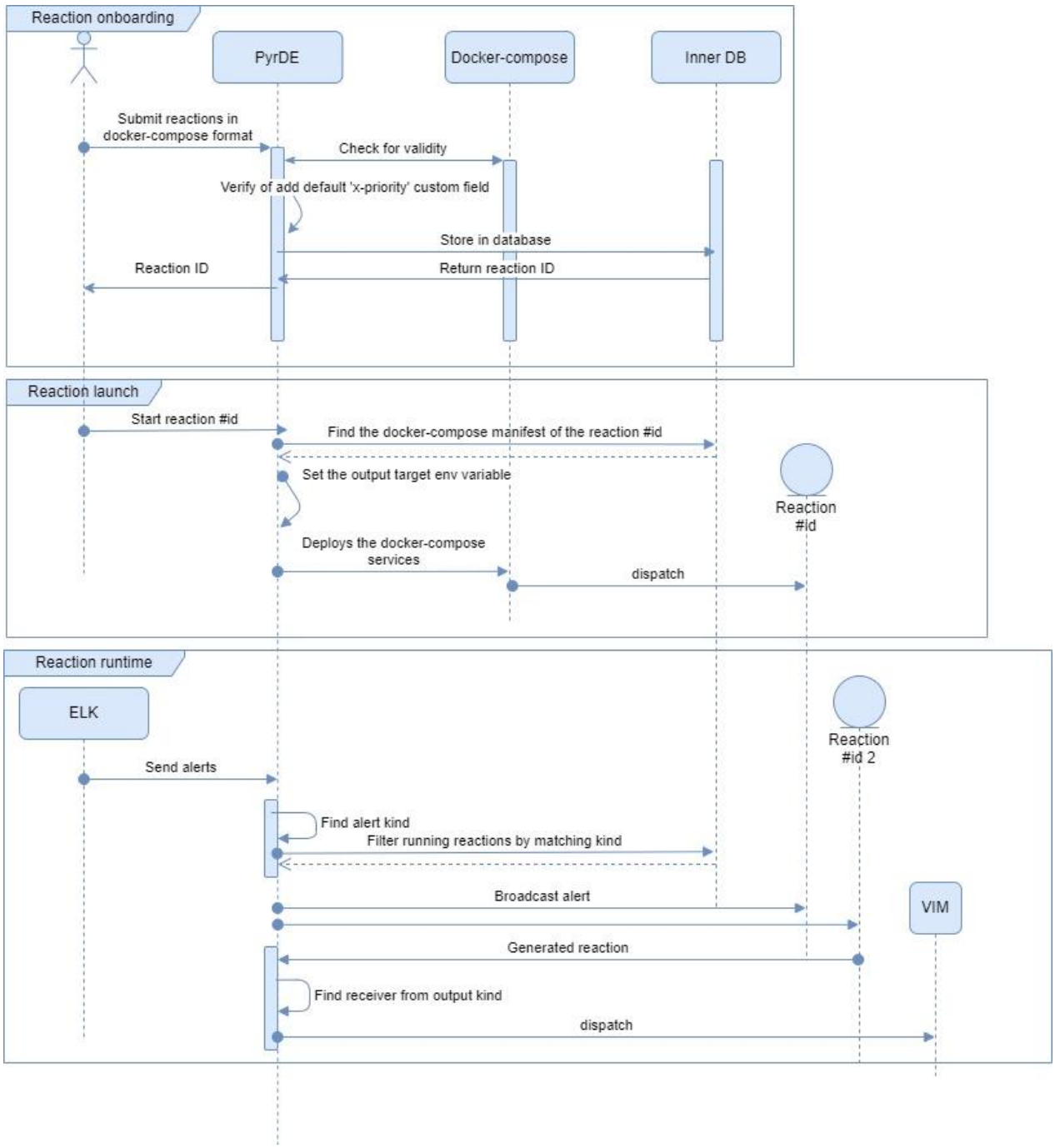

Figure 32: Reactions management

The local reaction uses environment variable to retrieve the endpoint where to submit the computed reaction. The local PyrDE provides this endpoint. This endpoint manages the reaction based on their priority (see the Figure 33). When the PyrDE broadcasts an alert, multiple local reactions may run concurrently. The first reaction to create a mitigation is released to the Security Orchestrator. A time gate period follows a mitigation submission. It allows the system to stabilize while the mitigation bounces inside the architecture. If another local reaction with a higher priority raises a new mitigation, it replaces the current one. An extraneous locking delay is added to the grace period. A lower priority has to wait an extra duration to be allowed to overwrite a more priority one. The REST API supports the modification of the associated priority. Inside the same priority level, the mitigations follow a FIFO model, which means that two mitigations from two separate reactions may intertwine, modulo the grace period.

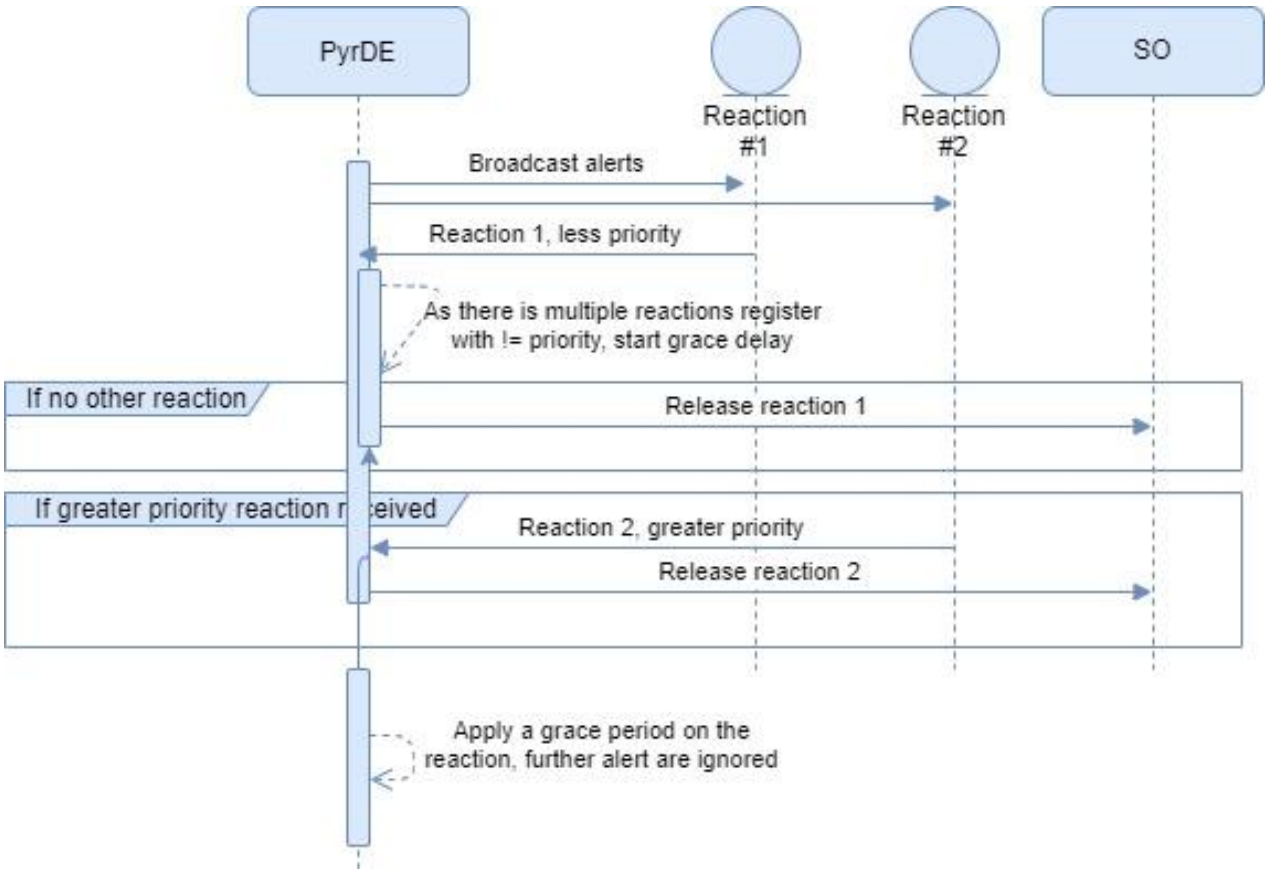

Figure 33 - Reaction priorities

The Figure 34 shows the local testbed implementation used for validation. In it the PyrDE secures the applications by controlling a SDN controller base on Ovs.

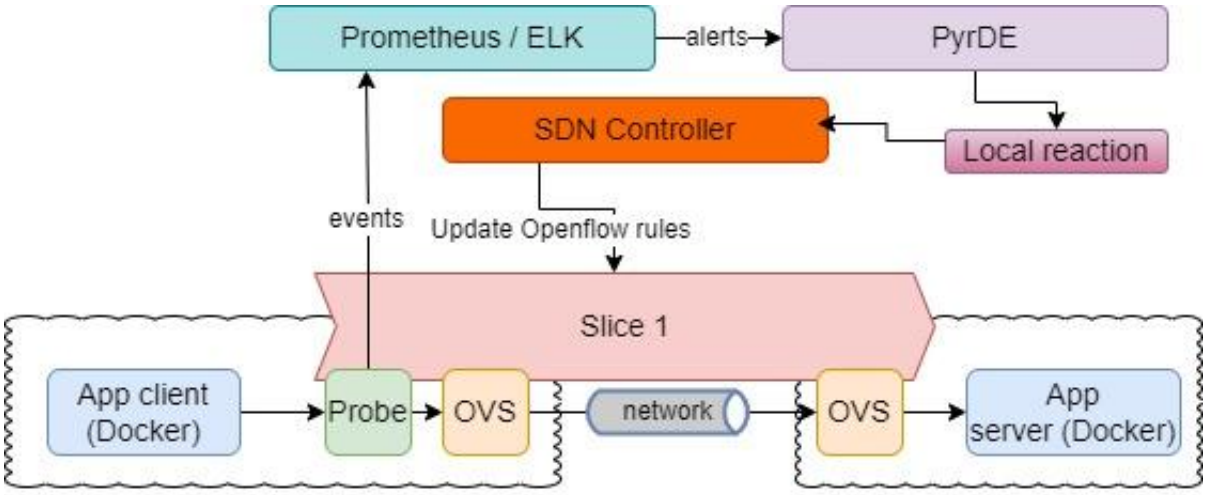

Figure 34: Local testbed example

2.10.5 Summary, lessons learned and guidelines

One of the lessons learned during this project was the numerous formats to support for all the connected enablers. A recommendation would be to adopt the cloudevents[38] specification for describing in a common way all the alert and events generated by the INSPIRE-5Gplus framework.

Some potential next steps for the PyrDE could be to combine it with a 5G Core system. In this context,

[38] https://cloudevents.io/

the PyrDE could manage the autonomous on-boarding on security components bundle as custom User Plane Functions (UPF) on the data plane. Another possibility could be to support SDN controllers as reaction backends. In this context, the PyrDE would create network security mitigations on-top of an OpenFlow compatible network fabric managed e.g. by OpenDayLight.

2.11 Systemic: accurate runtime monitoring

2.11.1 Overview

Between other security properties (e.g., authentication, confidentiality preservation, runtime integrity verification), the SECaaS-protected executables from Systemic SECaaS are self-monitored at runtime with several user-defined perimeters collected per function, for the complete executable or shared object. For that, the executables are appended with the required Systemic security routine which collects measurements produced at protection set-up time and appended on the protected binary file. The run time remote monitoring of the executable constitutes an evolution made during the project, notably resulting from the collaborative work engaged for liability and trustworthiness. This novel remote monitoring facility of any deployed software can integrate the ZSM vision. For the sake of clarity, one shall distinguish Systemic SECaaS server used at protection time and the Systemic routine appended on the protected executables and which generates and transmits heartbeats to the remote monitoring entity. Heartbeats content contains confirmed unambiguous tampering attack detection or authentication failures or, as part of future works, metrics that can be used to detect a large set of potential control flow-type of attacks on the software.

In all cases, heartbeats can be transmitted to other enablers as extraneous elements to infer anomalies on a processing node, location or part of the network with a more global vision to take the ad hoc reaction. Typically, Moving Target Defence will consider this information to avoid the redirection of the traffic to this corrupted node eventually before the pristine version of the code is re-installed. The periodicity of the integrity monitoring is adjusted automatically by the routine to conform with the performance overhead induced on the protected software in operation. An alternative to this sequence is to solicit the integrity check from an external security orchestrator. Systemic routine will produce the integrity monitoring on this event-demand and will thereafter return the test result to the request emitter.

2.11.2 Role in the HLA

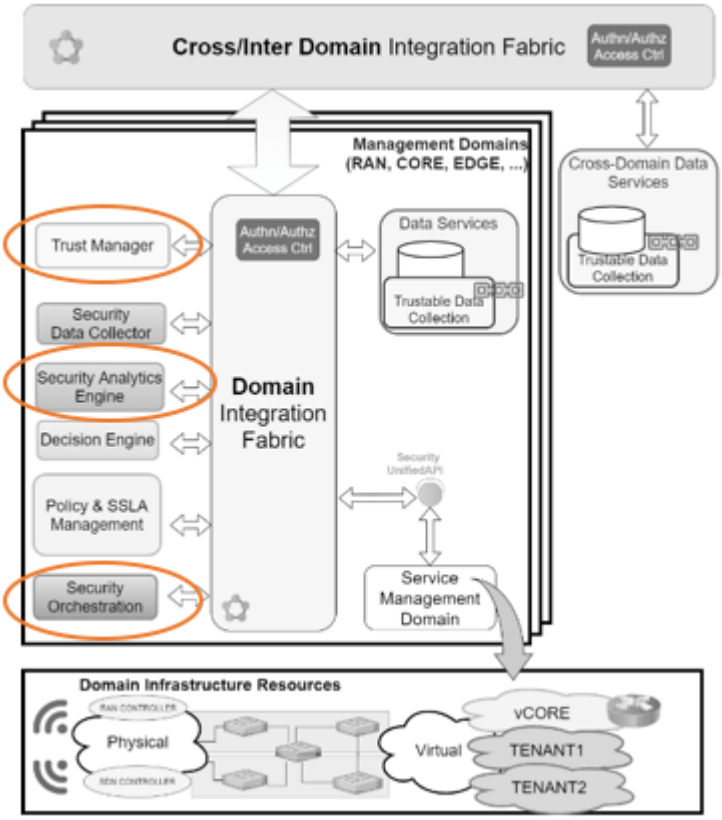

Figure 35: Systemic position the HLA

Per-se, Systemic SECaaS enabler shall be viewed as part of the Trust manager (top left in Figure 35), as being directly providing trust properties to a VNF or a VSF or any vertical application software committed in the slice. With respect to Systemic functionality, its insertion in the Trust Manager is the more relevant (as offering trust to the protected software, notably by its run time integrity continuous verification) but Systemic SECaaS can also be considered as offering security services (and be located in the Security Orchestration) and notably with respect to the deep run time monitoring service resulting from the project. Deep monitoring is a security service.

Systemic first interacts with the Domain integration fabric where the original software which requires protection shall be stored. Systemic collects the binary file, checks its protection associated metadata (if present) and protects the software if it is not yet protected. The protection can be defined by pre-defined templates eventually modified by a user through the user interface or directly be ordered through APIs by any security management utility (security orchestration, security management, being in the security domain or from the end-to-end security domain).

On a general perspective, since Systemic applies software security assurances, it brings higher trustworthiness on the deployed payload as delivering the confirmation that the code is integrated and authenticated before start, therefore taking part of the Trust Manager. The trust manager can be instructed by the security orchestration. At last, delivering security alerts or pre-alerts, it interacts with the Security Analytics Engine and indirectly with the Security Orchestration which will finally take the ad hoc reactions.

2.11.3 Sequence diagram

Systemic SECaaS sequence diagram is split into two parts corresponding to two phases of protection setup and protected software runtime activity when deployed.

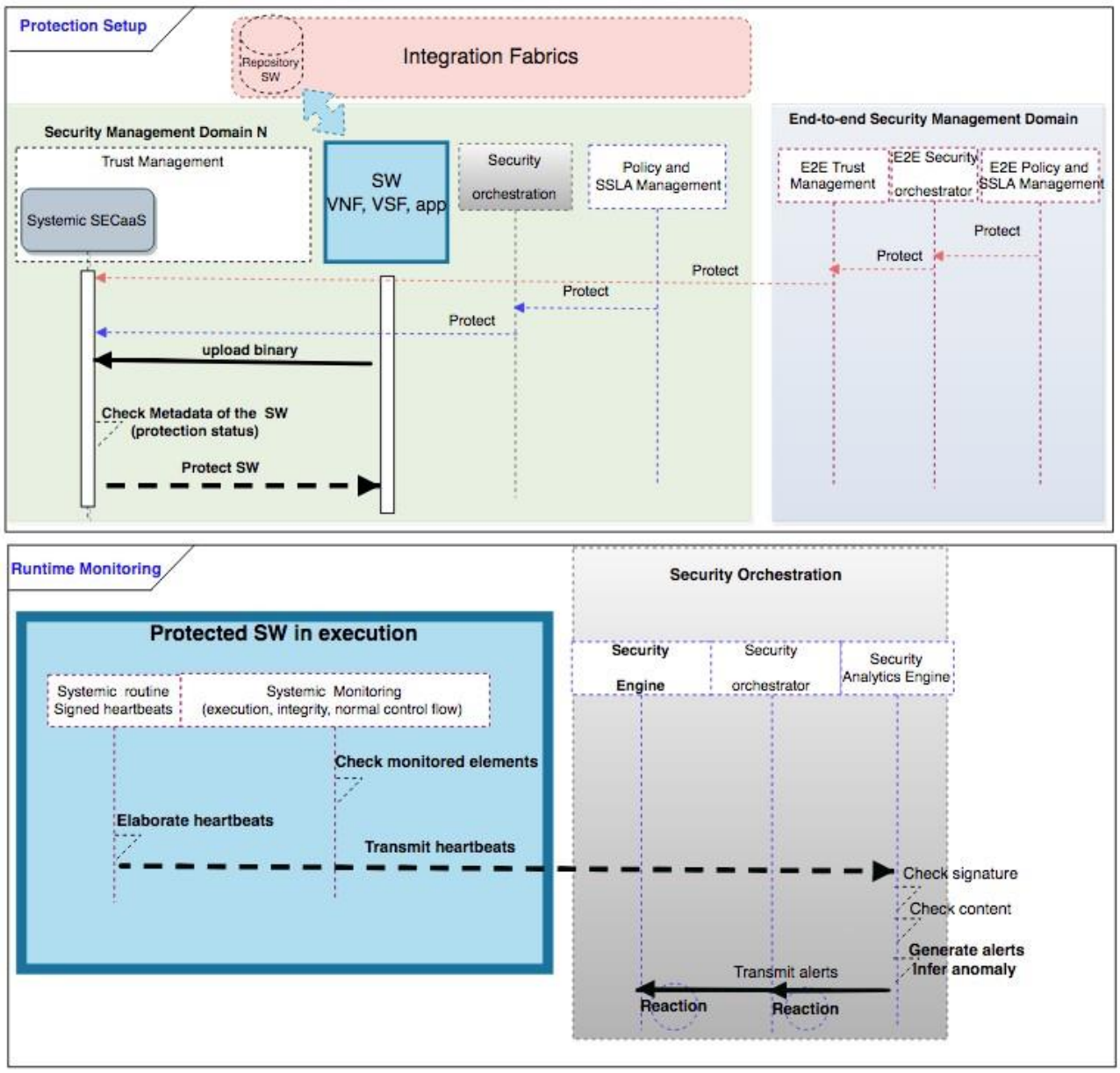

Figure 36: Systemic (SECaaS and routine) sequence diagram

Figure 36 depicts a standard setup workflow in these two main phases of protection setup and deep monitoring at runtime.

The upper part of the figure shows the interactions with the different enablers possibly interacting with the SECaaS, from the E2E Policy and SSLA manager, E2E Security Orchestrator, E2E Trust manager and their counterparts in the lower-level security domain. As stated above, the SECaaS is defined as taking part of a domain Trust Manager, although it could also be viewed as taking part of the security orchestration. These enablers can instruct and interact between them to order the protection of any deployed software taking part of the SLA. The security functions offered by Systemic confer trustworthiness and security of the software by enforcing its authentication, integrity, confidentiality and deep runtime monitoring. Henceforth, security and trust management of the SLA will benefit from the security services offered by Systemic SECaaS and may interact to commit their enforcements on a designated payload under deployment, leveraging the SECaaS APIs and by designating the payload stored in the integration fabrics. The SECaaS is able to check the metadata resulting from a former protection, check the reality of the protection settings of formerly protected version and of course protect a version.

The lower part of the figure shows the interaction between the appended Systemic routine located on

the protected software, once protected and which emits signed heartbeats, by collecting measured elements (integrity, presence of a key on the platform for geolocation purposes, control flow extracted elements (per code block time and frequency metrics). The signed and encrypted heartbeats deliver very valuable and exclusive markers of the correct functioning state of the protected software (e.g., the software executes (effectively, in real), at the correct location, is integrated and its execution trace does not deviate from the normal. These deep monitoring confer a novel mean to continuously validate the correct functioning of the software. From these heartbeats, a Security Analytics engine can elaborate alerts (i.e., self-authentication test failure, runtime integrity failure, absence of a required key on the machine) or pre-alerts (i.e., control flow trace elements), processed at the decision engine or security orchestrator which elaborate the ad hoc reaction.

2.11.4 Technical implementation

SECaaS implementation

The SECaaS current implementation packaging has been improved in the course of the Project. This packaging is a progress as stated just above and with the objective of easing Systemic installation in a typical 5G infrastructure. The implementation follows the same logics of Systemic itself. The installation must be automatically worked out in any encountered infrastructure software structures.

As displayed at the bottom of Figure 37, the wrapper is either offered inside a Docker container or as a monolithic statically linked native application (aka Solidbox) in order to comply with environments without container. As Solidbox can be possibly packed inside a VM, the three types of deployments (e.g., VM, CM and bare metal) shall comply with all types of processing and infrastructure contexts.

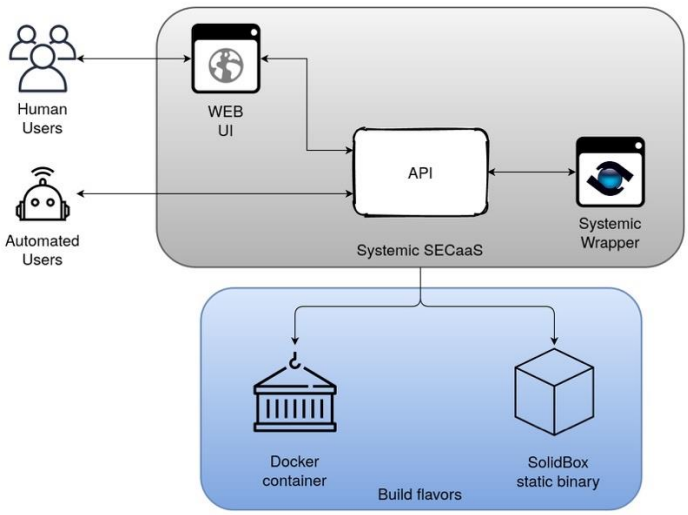

Figure 37: SECaaS implementation details.

The packaging has been engineered for the SECaaS model with exposed APIs to be inserted and used by a security orchestration or any type of server.

The system requirement is minimal as the VM-based, the CM-based or the native wrapper application can be installed on any commodity server of common and standard performance. The wrapper platform does not require Intel's SGX but if this option is selected by the user, the machines which execute the protected software shall be SGX-enabled.

The user documentation is available within the WEB-UI of the SECaaS server. The current version shall be updated and enriched with the new features resulting from the Project (e.g., SGX enablement, control flow shadowing obfuscation, expansion of authentication coverage to the protected file dependencies, etc.). This later update is a consequence of the complete reshuffling of the SECaaS server packing which turned the Windows native UI into HTML5 web interface. The current version of the wrapper is v21.11.01, available on TAGES company internal repository.

Systemic routine current implementation

Systemic routine is currently under progress to integrate the heartbeat monitoring facility as depicted here. This work has been engaged as part of demo 2 and 3. In the course of the project, the heartbeats are encrypted and signed using the TLS protocol and include a json file resulting from the SECaaS protection process and a mask reflecting the state of the software at the time of the heartbeat generation.

2.11.5 Summary, lessons learned and guidelines

A progress derived from this project is our own elaboration of our reasoning that it is far easier to collect the assurance that one software is in good functioning state than to protect against the endless attacks spawn at many different levels. When something wrong is detected, the ad hoc action shall be taken elsewhere, at a distant and integrated level.

Whereas one or a very limited quantity of methods can be used to check the correct functioning state accumulating a very low impact on performance, protecting the software against any types of attack requires an important number of mitigation techniques, all having their own impact on the software performance. The lesson learnt from this reasoning is that our technology can be used to elaborate the deep functioning state monitoring.

In the course of the project first period, our solution evolved in several directions to confer a higher security level (e.g., employ of Intel's SGX, enlargement of the security coverage to called dependencies) before their load and call by the main or a facilitated use (i.e., Solidbox solution packaging). Moreover, resulting from the project second period, a **deep monitoring** concept and development is engaged. The concept brings several promising novelties, all exploiting the by-default interconnection of any deployed software. Deep monitoring consists of checking continuously the correct functioning state of any deployed (Systemic protected) software. The vision is to keep track of each individual running instances, globally and with accuracy. The very first collected information is a proof of execution which results from Systemic appended routine, capable to trace the control flow and to validate that this specific instance of the software is effectively running. To meet that goal, a change of paradigm in software deployment is envisioned. It consists of generating different singular software instances with the aim of unambiguous identification and monitoring, deviating from the standard software deployment scheme based on identical deployments. In our vision, the most innovative deployment scheme is to generate each singular instance on-demand, directly when the software is called for the first time. For that, the SECaaS server will receive the generation orders on-the-fly which also confers a major progress as the software is deployed only when required, reducing its exposure to malevolent eyes. The deep monitoring concept necessitates security provision conferring trustworthiness and security (confidentiality and integrity) to the heartbeats messages

Second, our vision also integrates software zoning, a novel concept to restrict the execution of a software into a pre-defined perimeter of machines. **Software zoning**, preventing software cloning or use without right, is enforced by creating a functional dependence between the software and a provisioned key. It is notably considered as a mean to prevent IP-spoofing which is a persisting threat for telecom node impersonation leading to various types of network attack.

2.12 E2E Security Orchestrator

2.12.1 Overview

The 5G proposes a complex scenario where we find the need to manage the security of several domains when deploying services, then it is necessary to have an entity capable of directing, coordinating and validating the E2E security service orchestration process taking into account the Slicing capabilities characteristic of 5G systems. The E2E Security Orchestrator proposes a solution as part of ZSM approach, that driven by a closed-loop and with a global vision of the domains is capable of deploying a service as part of a 5G E2E security slice, which will be logically divided into sub-slices

and correctly deployed in each Security Management Domain (SMD). The E2E Security Orchestrator is responsible for performing the entire orchestration process, building each sub-slice policy with associated security according to the SMD.

2.12.2 Role in the HLA

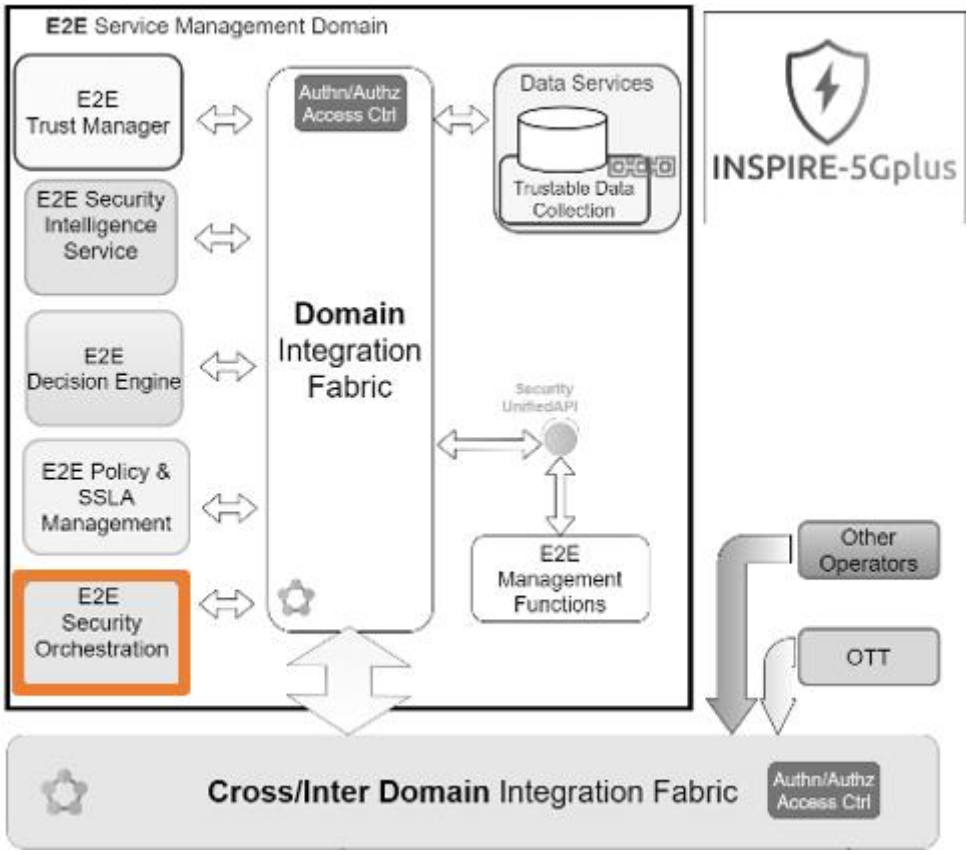

Figure 38: E2E Security Orchestration in HLA architecture

E2E Security Orchestrator showed in Figure 38 is in charge of directing, validating and coordinating the enforcement of security policies among the system with an E2E perspective. It will receive policies from the SSLA manager as part of the proactive enforcement and from the E2E Decision Engine as part of the proactive enforcement.

2.12.2.1 E2E_SO Security Policy Enforcement Service

E2E SO Security Policy Enforcement Service can receive multiple types of inputs for starting the orchestration & enforcement process, in particular it can receive HSPL-OP or MSPL-OP; once the policy has been received, the E2E SO identifies the Security Enforcement Domains and build the enforcement plan which will be forwarded accordingly to each SO.

2.12.3 Sequence diagrams

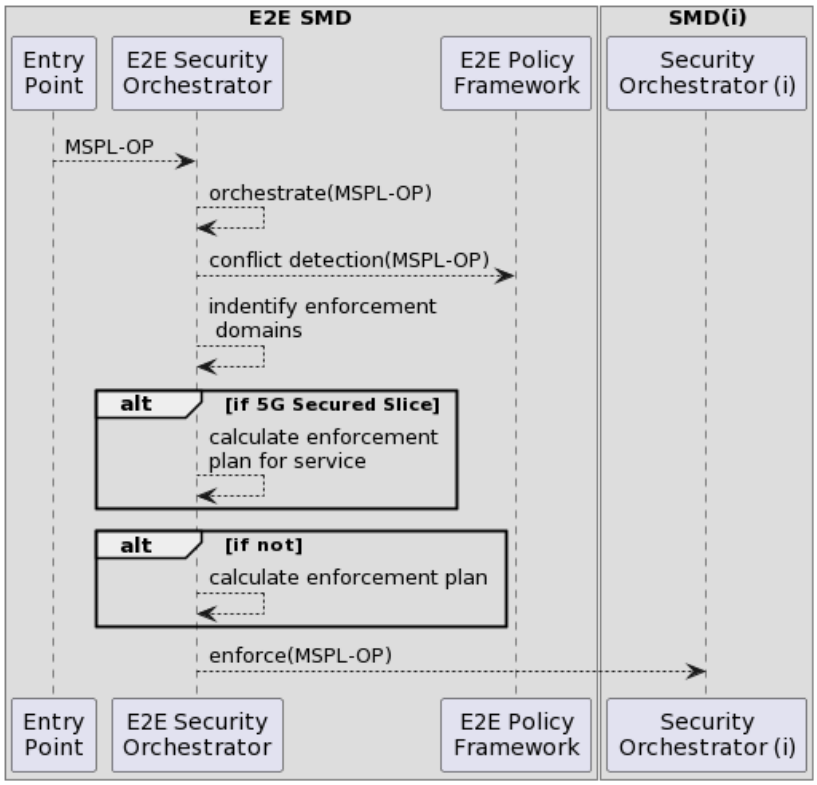

Figure 39: E2E SO orchestration sequence diagram

The E2E SO (as seen in the Figure 39) receives an MSPL-OP from one of the two possible entry points (SSLA Manager or Decision Engine), then the E2E SO starts the orchestration process by performing a conflict detection, after that it will extract the enforcement domain of the policies with the aim of calculating the enforcement plan. Depending on the type of MSPL-OP required to enforce (5G Security Slice or Security Requirement), the enforcement plan is calculated in a different manner. Once the enforcement plan is calculated, the E2E SO communicates the required MSPL-OP to each involved SMD SO to eventually enforce the policy as part of the E2E policy.

2.12.4 Technical implementation

Figure 40 shows the functional block that compose the E2E Security Orchestrator required to perform the orchestration procedure of security policies among the system with an E2E perspective.

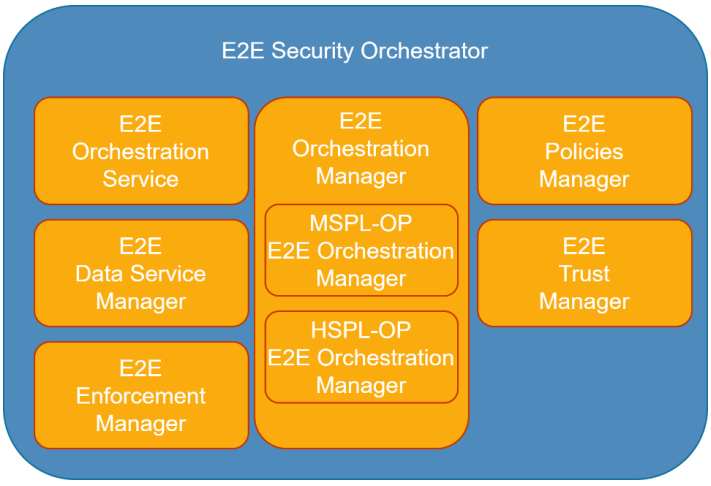

Figure 40: Current technical implementation and software blocks of the current deployment

- **E2E Orchestration Service:** This module provides the entry point for the E2E orchestration and enforcement of security policies. Specifically, it implements different classes of REST API endpoints that allow the E2E enforcing of different kinds of security policies. For instance, *MSPLEnforcementService* implements the E2E orchestration and E2E enforcement of MSPL policies by using specific E2E*MSPLOrchestrationManager* and *E2EMSPLEnforcementManager*. These classes implement the specific workflow for E2E orchestrating and E2E enforcing MSPL policies.
- **E2E Data service manager:** It implements a data services client that allow to access E2E data services in a common way. The specific data service driver is provided by auto-generated code by using OpenAPI tools (swagger-codegen).
- **E2E Enforcement Manager:** It implements the way that the E2E enforcement plan is enforced. Different E2E enforcement managers can be developed by including an *"enforce"* method. For instance, the *E2EFiveGSecuritySliceEnforcementManager* requests the sub-slice deployment to the calculated SMDs.
- **E2E Orchestration Manager:** It implements the E2E Orchestration Procedure; from the building of the Orchestration Policy until the beginning of the Enforcement Procedure, Different E2E orchestration managers can be developed by including an orchestration method. In particular for this block the E2E MSPL orchestration manager have been implemented which includes special cases for E2E 5G slice orchestration.
- **E2E Policies Manager:** This module implements different managers that allow managing different policy models. Since in the scope of INSPIRE-5GPlus MSPL models are used as main security policies model, A *MSPLManager* has been extended to include support to 5G slice capabilities. It offers methods and functions to ease the orchestrator manipulating MSPL policies (e.g., load_policies, detect_and_amend_amibiguity, translate).
- **E2E Trust Manager:** It implements common methods and functions that allow requesting trust assessment operations independently of the underlying (and available) Trust Reputation Manager implementations. Besides, specific driver for UMU E2E TRM was developed. It implements the specifics to get trust values of SMDs from UMU E2E TRM

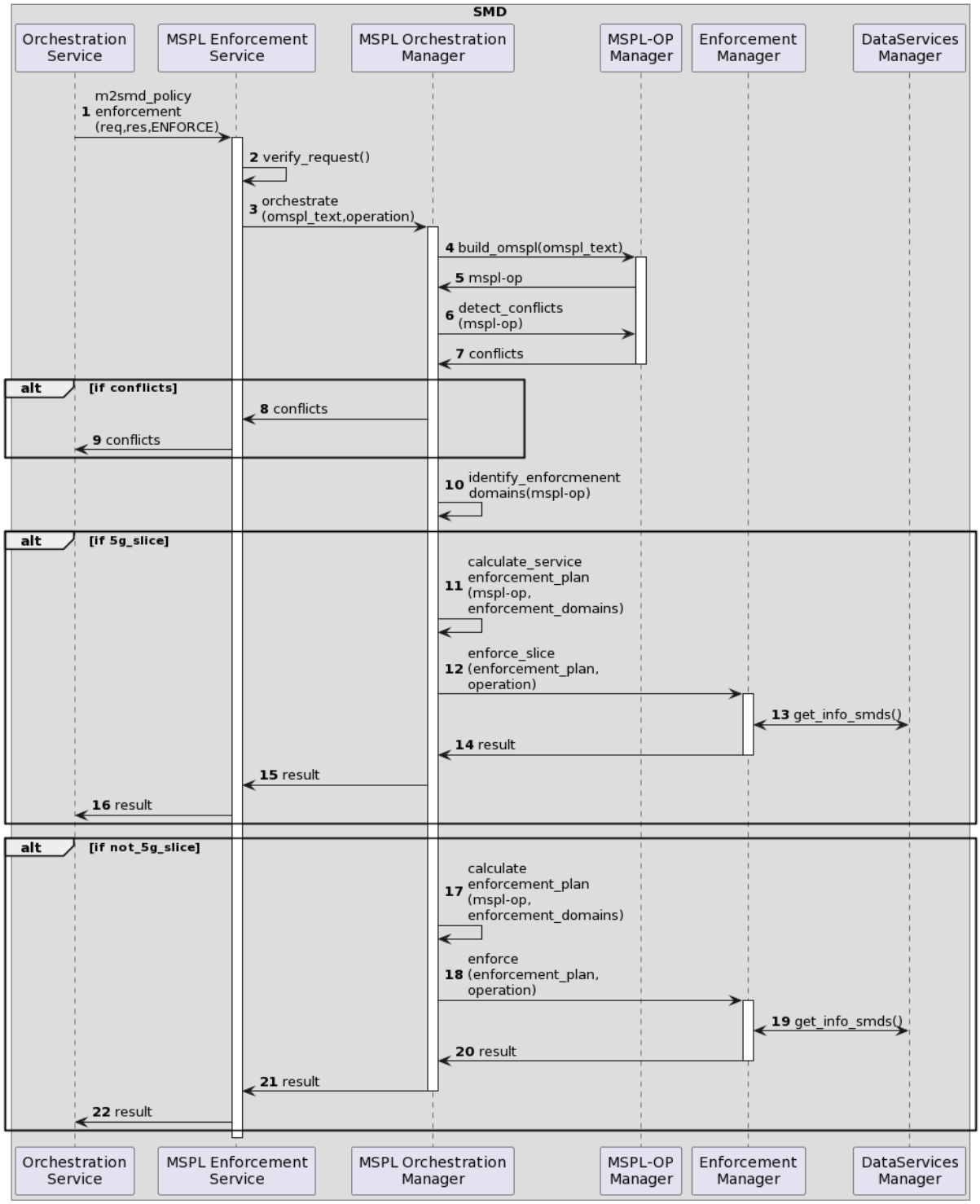

Figure 41 - Security Orchestrator SMD workflow details

1. In Figure 41, the first step for the **Orchestration Service** is the entry point of the E2E Security Orchestrator, it can receive HSPL-OPs and MSPL-OPs but it can be extended to other formats. As the Demo 1 is MSPL based, we will focus only on MSPL-OP flow. When the orchestration request is received at the orchestration service, the E2E SO starts the E2E orchestration process dictated by the entry point used in the service, in this case the MSPL Enforcement Service is used by calling the m2smd_policy_enforcement function with the operation ENFORCE.
2. The MSPL Enforcement Service verifies the API request and if no troubles have appeared
3. The MSPL Enforcement Service calls the **MSPL-OP Orchestration Manager** that will be used as

main building block to perform the orchestration process, including the calculation of the enforcement plan.

4. The MSPL-OP Orchestration Manager block is supported by the **Policy Manager (MSPL-OP Manager)**, starting by supporting the module to transform the requests from the API
5. The result is a well-defined MSPL-OP that is returned to the MSPL-OP Orchestration Manager block.
6. Then Policy Manager performs a policy conflict detection.
7. The result is returned to the MSPL Orchestration Manager.
8. If a conflict is detected, the conflicts are passed to the MSPL Enforcement Service
9. That is also forwarded to the Orchestration Service
10. Returning to the main flow, in this step the MSPL Orchestration Manager takes the MSPL-OP and identifies the enforcement domain of each part of the policy. Among this process, the manager identifies if the MSPL is 5G Slice related
11. If it detects that MSPL is 5G slice-related, in which case, slicing capabilities have been added so policies belonging to a sub-slice in an SMD are bundled and sent together. As services are not a security policy and they have security policies related to it, a specific treatment is made when the service is detected following a new flow, the E2E SO will create the MSPL-OPs accordingly to the SMD, specifying the sub-slice with the security policies that must be enforced and if the service must be deployed in that domain or it belongs to another sub-slice. The Orchestration Block makes use of the **Policy Manager** block to perform MSPL-OP operations, in particular, it is used to detect when the policy is service-related and it is in charge of adapting the 5G Secured Slice to each domain, by selecting from the original MSPL-OP only the security policies that affects the involved domain for which the MSPL-OP is being constructed at that moment. Once the enforcement plan has been created, it has prepared one MPSL-OP for each enforcement domain involved.
12. The **Enforcement Manager** block starts at this point
13. The Enforcement Manager will use the **Data Services Manager** to get the entry point of each SMD, and send each MSPL-OP to its SMD correspondingly.
14. Returning to the Orchestration Manager the result of the enforcement on each SMD.
15. That is transmitted to the Enforcement Service
16. And eventually from the Enforcement Service to the Orchestration Service.
17. In case there is no 5G Slice in the MSPL-OP, the enforcement plan is slightly different, as the policies are not bundled per domain, and they are sorted by the priority included as a field. So the lowest the priority is, the faster is received at the domain level.
18. As mentioned, the Enforcement Plan is sorted so, the Enforcement Manager starts sending MSPLs corresponding to each domain. In case that the same MSPL is shared by two domains (that is specified by the enforcement plan), a split process occurs that renames the ID of the policies based on the domain that is going to be sent to.
19. Again, the Enforcement Manager is supported by the Data Services which retrieves the information required to communicate with required SMDs.
20. Repeated process 14-16.

2.12.5 Summary, lessons learned and guidelines

Guideline

With this approach we enhanced the benefits of using policies, such as performing the E2E

orchestration procedure with the maximum flexibility which is an indispensable requirement for a mobile network security system, not only in terms of being able to add more security capabilities to the Orchestrator but also in terms of being able to add new orchestration algorithms without affecting previous behaviours, as well as easing the integration with new functional blocks.

Lessons learned

The E2E security orchestrator is characterized by complicated functionality, especially when it comes to coordinating with the other enablers and creating the enforcement plan as part of the orchestration process. The creation of the enforcement plan varies depending on the type of policy to be enforced. This means that when it is detected that a slice is to be deployed, it is necessary to check all the security requirements associated with the service, generate a customized MSPL-OP for each domain and order them by priorities, so that the process can be carried out correctly.

Future work

Future enhancements as explained in the following Section 2.13 will be focused on the security orchestration for B5G/6G networks, planning to integrate new AI-based orchestration algorithms that could collaborate with previous implemented algorithms, as well as easing the integration with: i) new technologies in the form of new capabilities. ii) New languages that could extend MSPL to represent more precisely certain requirements, so the E2E SO would be extended to support orchestration processes with the new models. iii) New functional blocks that could represent novel advancements in terms of security, as the TRM is in this project.

2.13 Security Orchestrator

2.13.1 Overview

5G features require (among others) a dynamic coordination and allocation of different resources according to the requirements (and security requirements). Services automation, network and security management through orchestration capabilities play a key role in accomplishing this task. In this regard, INSPIRE5G-Plus security orchestrator has been implemented as a ZSM-aligned solution to improve security in 5G infrastructures and beyond through a Policy-based security orchestration. Thus, the security orchestrator is in charge of orchestrating and enforcing security policies across the SMD infrastructure, considering their current status. The policy-based approach together with the plugin/driver/managers implementation allow including new security features in a modular way.

Since its last description in D3.2[39], the security orchestrator has been evolved to perform a trust-based orchestration and enforcement of 5G Security Slice policies. To this aim, different allocation types, algorithms, managers and drivers have been developed.

2.13.2 Role in the HLA

Figure 42 shows the Security Orchestrator module inside the HLA. Its main role is about orchestrating security policies across the SMD infrastructure. To this aim, it provides the Security Policy Enforcement Service.

[39] https://www.inspire-5gplus.eu/wp-content/uploads/2021/05/i5-d2.2_initial-report-on-security-use-cases-enablers-and-mechanisms-for_v0.14.pdf

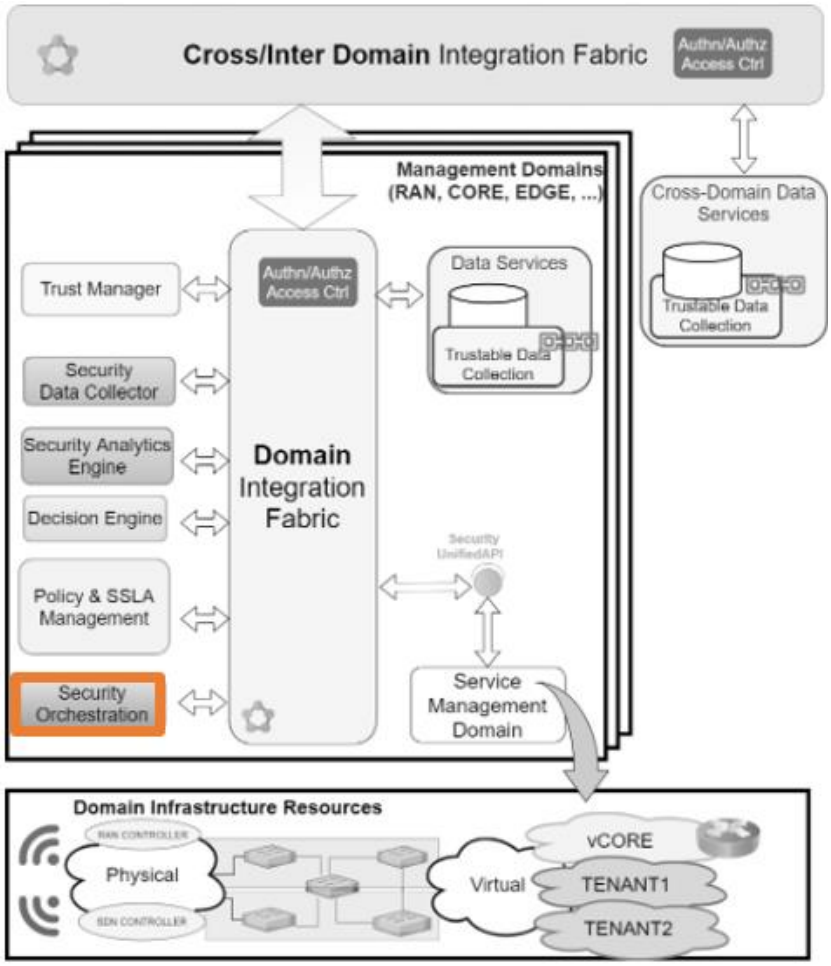

Figure 42: SO role in the HLA

2.13.2.1 SO Security Policy Enforcement Service

In the Figure 42, the Security Orchestrator Policy Enforcement Service provides a REST API to the rest of the INSPIRE-5GPlus components. It allows requesting security policies enforcement. As part of the policies enforcement, a trust-based orchestration process generates a trust-based enforcement plan. The orchestrator then create/configure/reconfigure assets/enablers by using different MANOs. As the result of this service, the security policies will be enforced across the SMD as expected.

2.13.3 Sequence diagrams

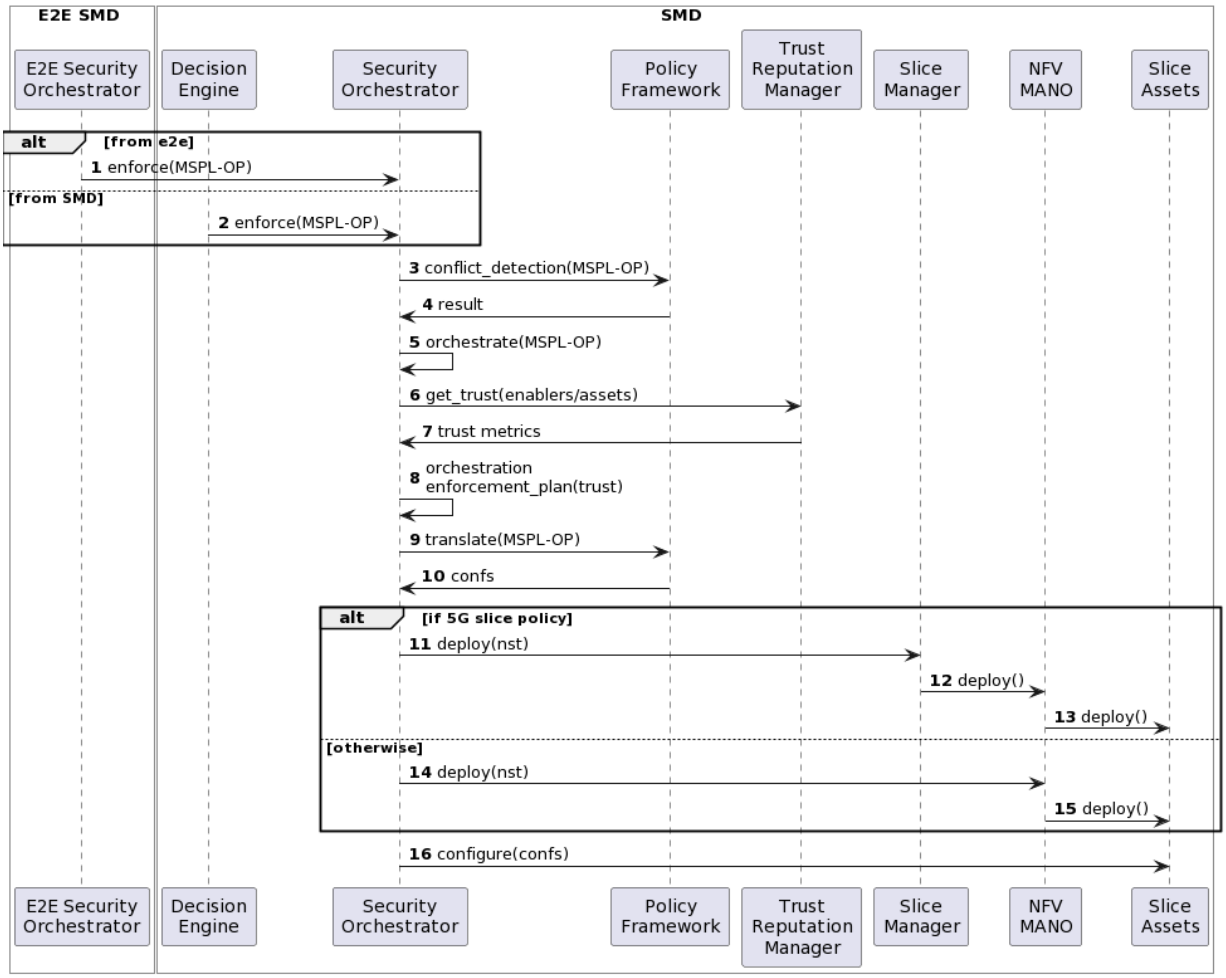

Figure 43: Orchestration process

Figure 43 shows the main workflow that allows a SMD Security Orchestrator orchestrate and enforce proactive and reactive security policies. The Security orchestrator can receive proactive/reactive enforcement requests from E2E Security Orchestrator (step 1), or reactive policies from the Decision Engine (step 2). Once the security orchestrator receives the enforcement request, a policy conflict/dependencies detection is performed (steps 3-4). In case there are no conflicts, a trust-based orchestration algorithm calculates the best enablers/assets that should be used to enforce the security policies, as well as an enforcement plan to determine the enforcement points (steps 5-8). After that, each security policy will be translated (steps 9-10). If new slices or VNFs deployments are required, the security orchestrator will command it through Slice Manager/VNF manager. After slice or VNFs are deployed, security services will be configured according to the enforcement plan.

2.13.4 Technical implementation from UMU

The Security Orchestrator implements different services and modules (python3) that allows orchestrate different kind of security policies (e.g., 5G security slices) across multiple enablers/assets of the SMD infrastructure.

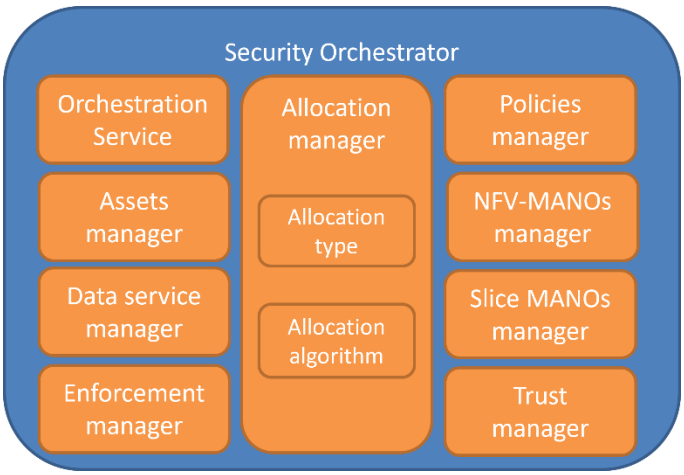

Figure 44: SO modules

Figure 44 shows the main modules (services and managers) that have been implemented/extended inside the Security Orchestrator in the scope of INSPIRE-5Gplus. In general, the implementation follows a manager/driver approach that allows the orchestrator to be able to orchestrate new technologies by adding new specific managers and drivers to their modules.

1. Orchestration Service: This module provides the entry point for orchestrate and enforcing security policies. Specifically, it implements different classes that REST API endpoints that allow enforcing different kind of security policies. For instance, *MSPLEnforcementService* implements the orchestration and enforcement of MSPL policies by using specific *MSPLOrchestrationManager* and *MSPLEnforcementManager*. These classes implement the specific workflow for orchestrating and enforcing MSPL policies.
2. Allocation Manager: It implements the calculation of the orchestration and enforcement plans according to the selected allocation type and allocation algorithms. This selection is performed as first step according to the available features. For instance, if the infrastructure is NFV-enabled, an nfv-enabled allocation type can be selected. Otherwise, conf-only allocation will be selected. Besides, specific *FiveGSliceAllocation* type has been also implemented to enforce 5G Security slice policies. Regarding the algorithms, different algorithms based on load, weight and trust have been implemented.
3. Policies Manager: This module implements different managers that allow managing different policy models. Since in the scope of INSPIRE-5GPlus MSPL models are used as main security policies model, A *MSPLManager* has been implemented. It offers methods and functions to ease the orchestrator manipulating MSPL policies (e.g., load_policies, update_candidate_assets, translate).
4. NFV-MANOs manager: It implements common methods and functions that allow requests NFV operations independently of the underlying (and available) NFV-MANOs. For instance, operations like *deploy, configure, remove, get_current_descriptors, get_current_instances* are provided. Besides, specific driver for OSM was developed. It implements the specifics to perform those common operations in OSM.
5. Slice-MANOs manager: It implements common methods and functions that allow requests Slice operations independently of the underlying (and available) Slice-MANOs. For instance, operations like *deploy, configure, remove, get_available_slice_templates* are provided. Besides, specific driver for OSM was developed. It implements the specifics to perform those common operations in OSM.
6. Trust manager: It implements common methods and functions that allow requests trust assessment operations independently of the underlying (and available) Trust Reputation Manager implementations. Besides, specific driver for UMU TRM was developed. It implements the specifics to get trust values from UMU TRM.

7. Assets/Enablers Manager: It implements a set of drivers that allow the Security Orchestrator configuring Assets/Enablers (using the results of the policy translation). Different drivers have been implemented:
    1. AmarisofRANDriver: It allows enforcing configurations on Amarisoft-based 5G deployments.
    2. 5GCoreAgentDriver: It allows enforcing/managing configurations and 5G core components.
    3. I2NSFControllerDriver: It allows enforcing I2NSF configurations in the I2NSF Controller.
    4. PoTControllerDriver: It allows enforcing PoT configurations in the I2NSF Controller.
    5. STADriver: It allows enforcing STA monitoring configurations.
    6. V2XDoSDriver: it allows enforcing v2x monitoring configurations into the v2x enabler.
    7. V2XAggregatorDriver: it allows enforcing v2x aggregator configurations like v2x data filtering into the v2x enabler.
8. Data Service manager: It implements a data services client that allow access data services in a common way. The specific data service driver is provided by auto-generated code by using OpenAPI tools (swagger-codegen).
9. Enforcement Manager: It implements the way that the enforcement plan is enforced. Different enforcement managers can be developed by including an *"enforce"* method. For instance, the *FiveGSecuritySliceEnforcementManager* requests the slice deployment to the selected SliceMANO, and once the different elements that compose the slice have been deployed, it updates the data services and uses the new endpoints to enforce final security configurations obtained during the translation process.

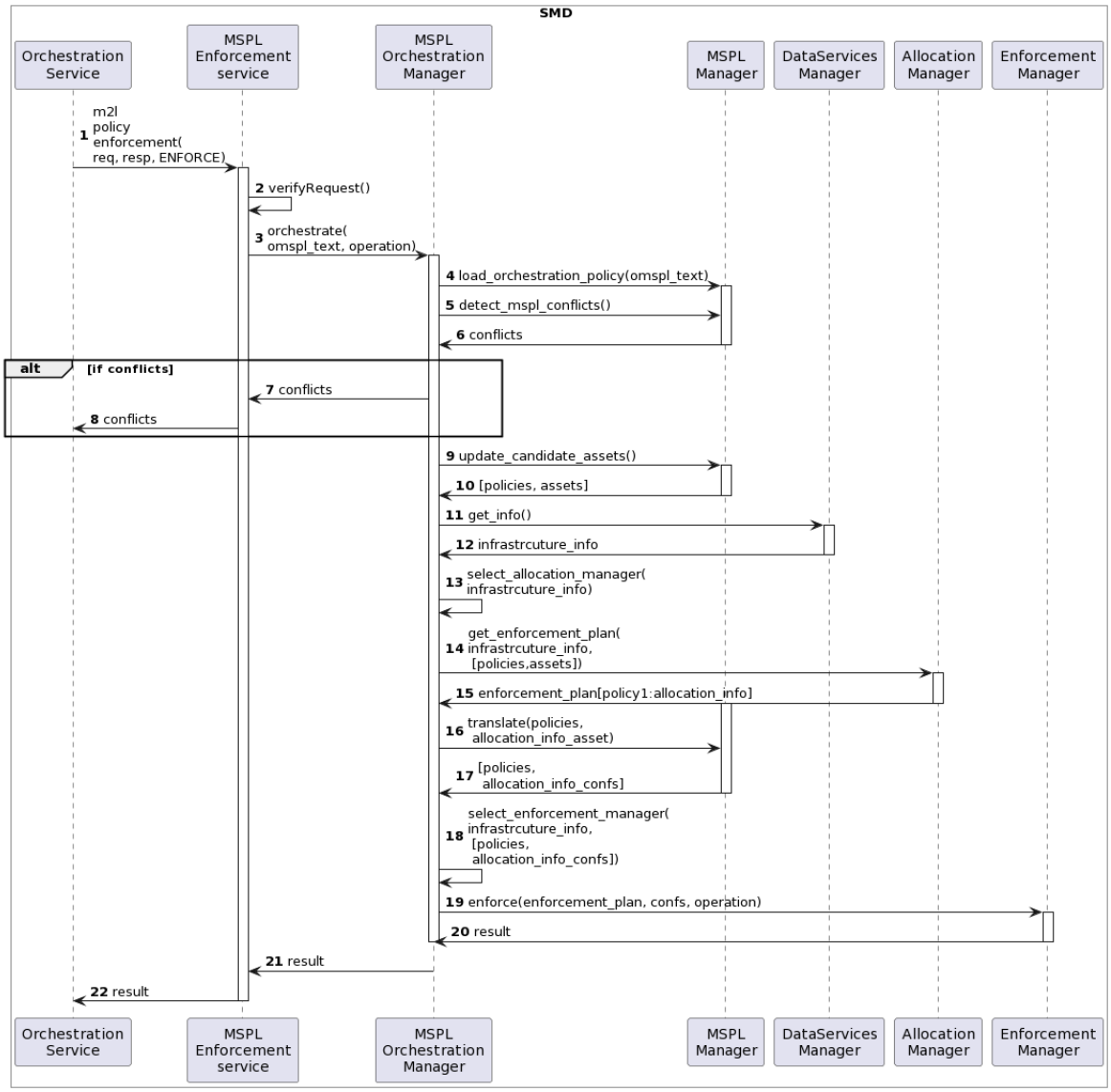

Figure 45: MSPL Orchestration

Figure 45 shows the MSPL Orchestration workflow that has been implemented for INSPIRE5G+, mainly focused on the interactions between the MSPLOrchestrationManager and the rest of the managers that provides information/features to that process.

1. The Orchestration service implements different endpoints to enforce different kind of security policies. Current implementation allows orchestrating and enforcing MSPL-OP, and a subset of MUD files. Depending on the policy type, different implementation of the enforcement service is provided. For instance, for MSPL-OP, *MSPLEnforcementService* object is created and the enforcement is requested.
2. MSPLEnforcementService verifies that the enforcement request is well formed.
3. MSPLEnforcementService requests the orchestration and enforcement to the MSPLOrchestrationManager.
4. MSPLOrchestrationManager uses the MSPLManager to load the MSPL-OP policy as a MSPL-OP object by using the mspl module, that implements the MSPL-OP XML parser by using the pyxb[40] tool.

[40] https://pypi.org/project/PyXB/

5. MSPL conflicts are requested. Conflict detection is performed by using pyke[41], a knowledge-based inference engine. Different rules for intra-policy conflict detection we defined in a previous project are available.
6. Conflict detection result is provided.
7. In case of conflict, MSPLEnforcement service is notified.
8. In case of conflict, OrchestrationService is notified.
9. Otherwise, MSPLManager is used to gather the available assets/enablers with the required security capabilities, able to enforce the requested policies.
10. MSPLManager updates the enabler/asset candidates list for each security policy (e.g., Filtering Policy -> [ONOS, Firewall...]), according to the available ones, where available means that they are already deployed or that they could be deployed dynamically (e.g., as part of a slice deployment). MSPLManager uses DataServices to retrieve that information.
11. MSPLOrchestrationManager retrieves information from data services that will be as part of the orchestration process (e.g., available MANOS, available slice managers...)
12. MSPLOrchestrationManager receives the requested information from data services trhough the DataServicesManager.
13. MSPLOrchestrationManager uses data services information to decide the suitable allocation manager. For instance, if a 5G security slice is required and a slice MANO is available, *FiveGSecuritySliceAllocationManager* will be used.
14. MSPLOrchestrationManager uses the selected AllocationManager to orchestrate an enforcement plan. Infrastructure info, as well as policies and the candidate assets/enablers are also provided.
15. AllocationManager returns the allocation info (including the selected enabler/asset) for each security policy. This allocation info contains information such as, it the enabler/asset must be deployed, descriptors, NFV-MANO/Slice-MANO, orchestrator driver required and dynamic configuration APIs that will be used to enforce the policy.
16. Once the enforcement plan has been defined for each policy, policy translation is requested to the MSPLManager.
17. MSPLManager returns the final configurations for each policy according to the enforcement plan.
18. In order to proceed with the enforcement stage, the MSPLOrchestrationManager decides the suitable EnforcementManager according to the enforcement plan, and the infrastructure information (different orchestration approaches could require different enforcement approaches. For instance, regular security policy enforcement is managed in a different way that 5GSecuritySlice enforcement).
19. The enforcement is requested to the selected *EnforcementManager*
20. *EnforcementManager* provides the result of the enforcement.
21. MSPLOrchestrationmanager provides the result of the orchestration and enforcement.
22. MSPLEnforcementService provides the result of the whole operation.

[41] http://pyke.sourceforge.net/

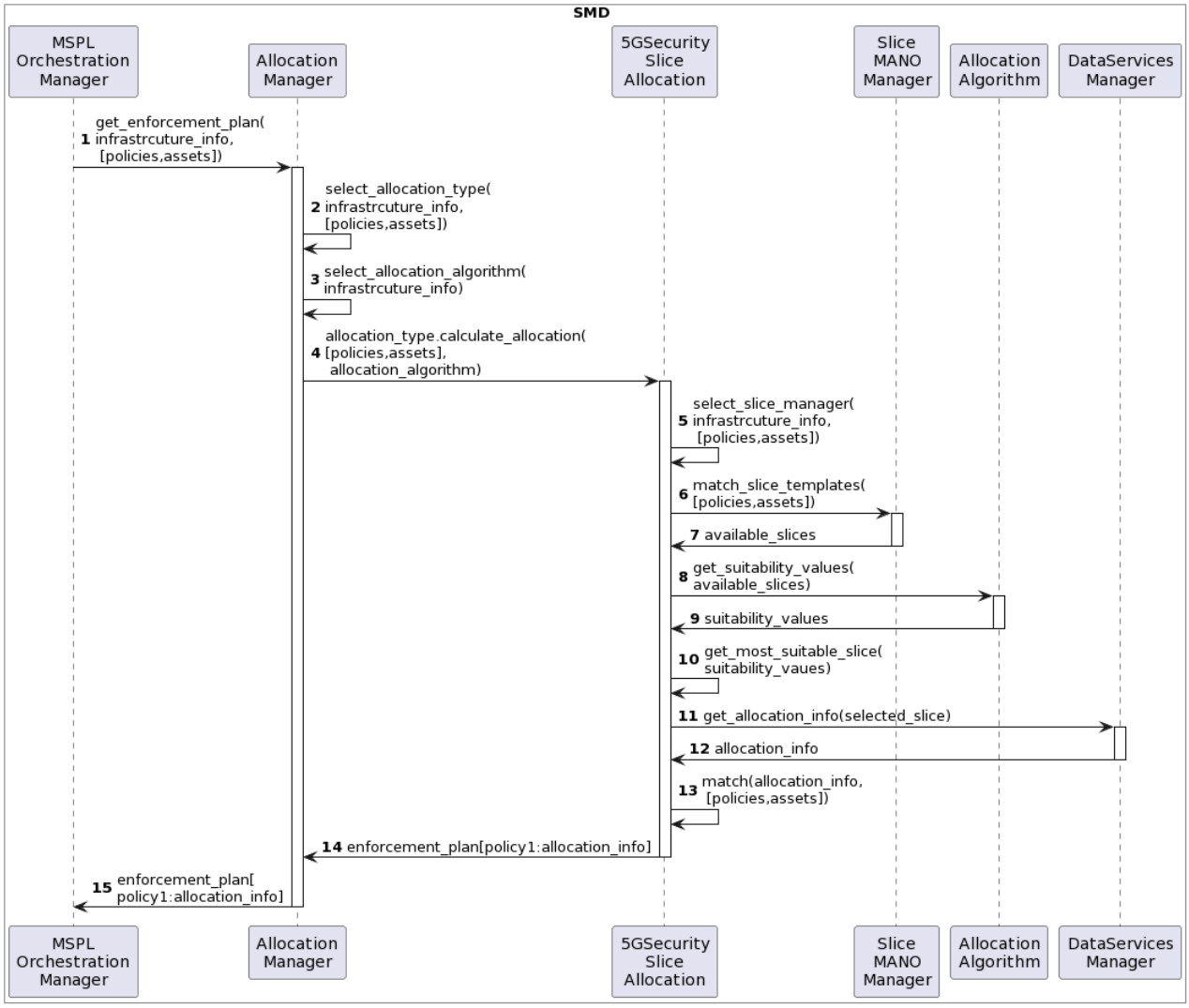

Figure 46: 5GSecuritySliceAllocation implementation

Figure 46 shows the specific flow implementation for the 5GSecuritySliceAllocation process:

1. MSPLOrchestrationManager requests the enforcement plan to the AllocationManager.
2. AllocationManager selects an allocation type based on the enforcement requests as well as the information about the current infrastructure. Current implementation considers:
    1. 5GSecuritySliceAllocation.
    2. Regular Conf-onlyAllocation.
    3. Regular NFV-enabled Allocation.
3. AllocationManager selects the allocation algorithm according to the resources availability (e.g., Trust-based algorithm cannot be selected if TRM is not available, or load based algorithm cannot be selected if there is no place where load metrics can be retrieved).
4. AllocationManager uses the selected allocation type and allocation algorithm to requests the enforcement plan calculation.
5. 5GSecuritySliceAllocation algorithm selects a slice MANO manager. This selection is performed by considering available MANOS as well as the required capabilities.
6. 5GSecuritySliceAllocation algorithm requests a match between the required features and the available pieces inside the available slice templates.
7. SliceMANOManager returns those slices that are able to cope with the requested capabilities.
8. For each slice template, the 5GSecuritySliceAllocation algorithm requests the suitability asset value to the Allocation Algorithm (in this case, Trust-based Allocation algorithm due TRM is available). Current implementation also considers LoadBasedAllocationAlgorithm and WeightBasedAllocationAlgorithm. Besides, new algorithms can be added easily.

9. Trust-based allocation algorithm relies to TRM to calculate trust, and it returns the results.
10. 5GSecuritySliceAllocation algorithm selects the best slice template according to the trust-based score.
11. Allocation info is requested from data services through DataServiceManager.
12. Allocation info is retrieved.
13. 5GSecuritySliceAllocation algorithm matches policies with the enablers/assets allocation info inside the slice.
14. 5GSecuritySliceAllocation returns the enforcement plan composed by the selected slice template as well as the mapping between each policy, the required enabler/asset (part of the slice) that will be used to enforce it, as well as their allocation info.
15. Enforcement plan is provided to the MSPLOrchestrationManager.

Figure 47 shows the specific flow implementation for the 5GSecuritySliceEnforcement process:

1. MSPLOrchestrationManager requests the policies translations according to the enforcement plane.
2. MSPLOrchestrationManager receives the required configurations.
3. MSPLOrchestrationManager requests the enforcement to the enforcement manager. According to the involved policies, this manager will be instantiated to a 5GSecuritySliceEnforcement manager.
4. 5GSecuritySliceEnforcement manager gets the selected slice template (nst_id) from the enforcement plan.
5. 5GSecuritySliceEnforcement manager requests the slice deplyment to the SliceMANOManager.
6. SliceMANOManager requests the slice deployment to the selected SliceMANO (OSM in the testbed).
7. SliceMANOManager polls the slice status.
8. SliceMANOManager receives the current status of the slice deployment operation.
9. Once the slice is ready, SliceMANOManager requests the deployment information (nsi_info)
10. Nsi info is provided to the SliceMANOManager.
11. Nsi info is provided to the 5GSecuritySliceEnforcement manager.
12. 5GSecuritySliceEnforcement manager updates the slice information into data services trhough the DataServicesmanager.
13. For each security policy, part of the security slice, 5GSecuritySliceEnforcement manager requests the enforcement of the configurations according to the enforcement plan information.
14. 5GSecuritySliceEnforcement manager provides the result to the MSPLOrchestrationManager.
15. MSPLOrchestrationManager update the policies status though the MSPLManager.

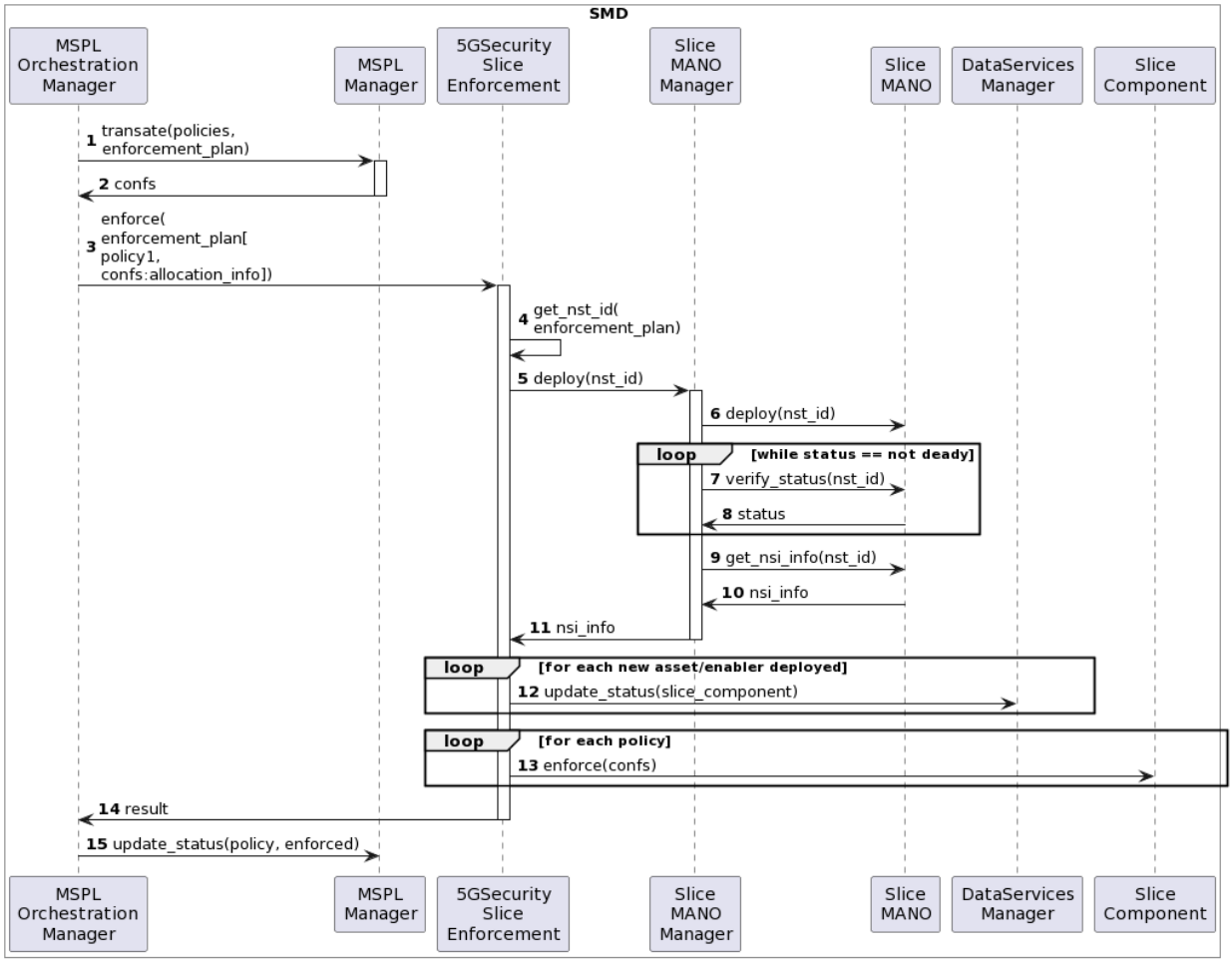

Figure 47: 5G Security Slice Enforcement

To provide part of the functionality of Data Services support during 5GSlice orchestration processes, previous implementation of a system model has been extended to consider new assets, enablers as well as the domain info. Figure 48 shows a subset of different models used for storing/retrieving infrastructure information.

```
GET /
```

```
HTTP 200 OK
Allow: GET, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

{
    "domain": "http://sysmodel-api.k8s.gaialab/domain/",
    "tenant": "http://sysmodel-api.k8s.gaialab/tenant/",
    "device": "http://sysmodel-api.k8s.gaialab/device/",
    "device-group": "http://sysmodel-api.k8s.gaialab/device-group/",
    "flavor": "http://sysmodel-api.k8s.gaialab/flavor/",
    "location": "http://sysmodel-api.k8s.gaialab/location/",
    "software": "http://sysmodel-api.k8s.gaialab/software/",
    "api": "http://sysmodel-api.k8s.gaialab/api/",
    "capability": "http://sysmodel-api.k8s.gaialab/capability/",
    "enabler": "http://sysmodel-api.k8s.gaialab/enabler/",
    "credential": "http://sysmodel-api.k8s.gaialab/credential/",
    "network-interface": "http://sysmodel-api.k8s.gaialab/network-interface/",
    "sixlowpan-network-interface": "http://sysmodel-api.k8s.gaialab/sixlowpan-network-interface/",
    "network-interface-info": "http://sysmodel-api.k8s.gaialab/network-interface-info/",
    "connection": "http://sysmodel-api.k8s.gaialab/connection/",
    "flow": "http://sysmodel-api.k8s.gaialab/flow/"
}
```

Figure 48: Subset of models

To ease different modules accessing/retrieving data, the client is auto-generated by using swagger tools, so common formats and methods are provided.

Finally, the Security Orchestrator (and the complementary modules) runs in docker containers. Docker-file have been elaborated to ease testbed/debug deployment. Besides, multiple k8s manifests (per micro service) have been created to deploy and integrate the solution with the integration fabric, following the "service-deployment-routing" approach.

2.13.5 Technical implementation by TSG

TSG's technical implementation of the SO has a similar architecture to UMU's one described in the previous section, but with the following main differences:

1. It does not either include the Trust Manager module in the current stage, nor does it support any integration with UMU TRM.
2. Since TSG's implementation targets mainly Kubernetes (K8s) environments for deployment in the context of INSPIRE-5Gplus, i.e. using K8s as VIM (Virtual Infrastructure Manager) in ETSI terms, the Slice MANOs and NFV MANOs manager modules only support OSM as Slice MANO at this stage, with K8s VIM, and the only supported kind of VNF (Virtual Network Function) is CNF (Containerized VNF), which OSM calls more precisely in this case KNF (Kubernetes VNF); with particular support for INSPIRE-5Gplus security enabler VCP (Virtual Channel Protection) packaged and deployed as a KNF. It is worth mentioning that TSG implementation needs to go beyond ETSI VNF standards - not mature yet regarding CNFs - to specify and implement mechanisms for CNF instantiation and configuration.

2.13.6 Summary, lessons learned and guidelines

Guidelines: We recommend using policy-based orchestration since it improves automation and management. Policy-based orchestration approach allows to homogenize infrastructure management since orchestration processes feed on well-known and common formats. Multiple orchestration algorithm can give better orchestration results since orchestrator can be adapted according to the infrastructure or the situation requirements. To separate "what" and "how" can increase orchestration flexibility, different stages allow providing different constraints to determine what should be enforced, how should be enforced and their relationship. Finally, driver-based orchestration mitigates technology heterogeneity and provides extensibility, besides, it allows orchestrator managing new technologies on demand.

Lessons learned: After implementation and testing, we identified different possible improvements in terms of orchestration algorithms and dependencies management. Different AI based approaches could provide improvements on the results.

Future extensions: We expect to continue the security orchestration research line (E2E and SMD level) for B5G/6G, especially focused on providing new AI-supported orchestration/allocation algorithms. Besides, new managers could be included, extending the ones developed for INSPIRE5G-Plus in order to provide interoperability with future technologies. For instance, a new policy, slice mano, or data services mano could be provided. Also, to include new orchestration capabilities, new orchestration drivers could be provided. Finally, dependencies management could be also improved as part of new research efforts. Interoperability between SMDs at the orchestration level is also considered as a priority and the usage of an extended version of TOSCA to support the security policies for orchestration will be studied.

2.14 DiscØvery

2.14.1 Overview

DiscØvery has been described in D3.2 and D4.2 as a tool to facilitate the assessment of cybersecurity and trust among networks. DiscØvery provides automated functions that help the security analyst to holistically view the assets of network, determine the attack surface, and suggest mechanisms and other insights to improve the security posture.

2.14.2 Role in the HLA

DiscØvery is part of the E2E policy and SSL management of the HLA as shown in Figure 49.

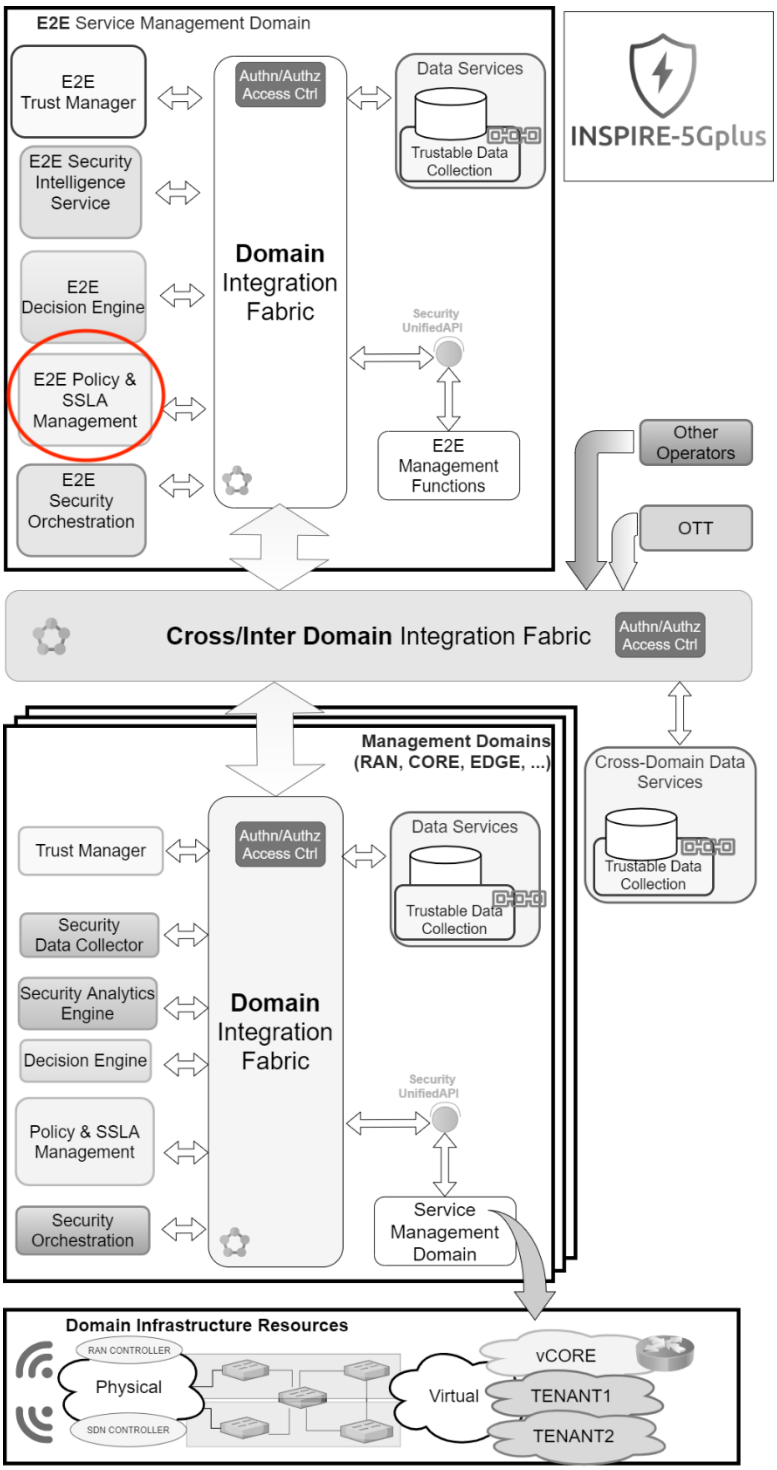

Figure 49: DiscØvery in the INSPIRE-5Gplus HLA

2.14.2.1 Threat and Vulnerability Identification service

The Threat and Vulnerability identification service is used to automatically identify threats that are targeting assets and the vulnerabilities that can be used to exploit them. To identify threats the tool uses an internal database of threat types and network assets that they can target. The database had been populated with information identified in WP2, specifically the D2.1 5G Security – Current status and Future Trends. For the identification of vulnerabilities, the tool supports linking to external vulnerability databases, such as the CVE database provided by Mitre[42]. Once the Threat and

[42] https://cve.mitre.org

Vulnerability service is used, the tool generates a graph-based representation of the network with the network's components, assets, identified threats and vulnerabilities shown as nodes in the graph. A security analysts can then visually inspect the graph to better assess the security posture of the network.

2.14.2.2 Cyber Security Insights Service

The Cyber Security Insight service is used to provide a list of suggestions to improve the security posture of the network based on its characteristics and configuration. The insights range from suggesting additional high-level security mitigation strategies, such as policies, to low-level security mechanisms, such as firewall rules. The provided insights cannot be automatically applied to the network as they to be assessed and tested by a security analyst. However, a security analyst can experiment with different network topologies and configuration through the DiscØvery tool since different topologies will output different insights.

2.14.3 Sequence Diagrams

The sequence diagram of the data of the DiscØvery is shown in Figure 50 below. DiscØvery gathers the asset information of the network to generate a model to be used for security assessment. The main process for the security assessment is the Cyber Security Insight service, that once it is used it can generate a report that can be shared with the Policy and SSL Management. The report includes the data from all the services supported by the tool such as the Threat and Identification Service and the Cyber Security Insight Service.

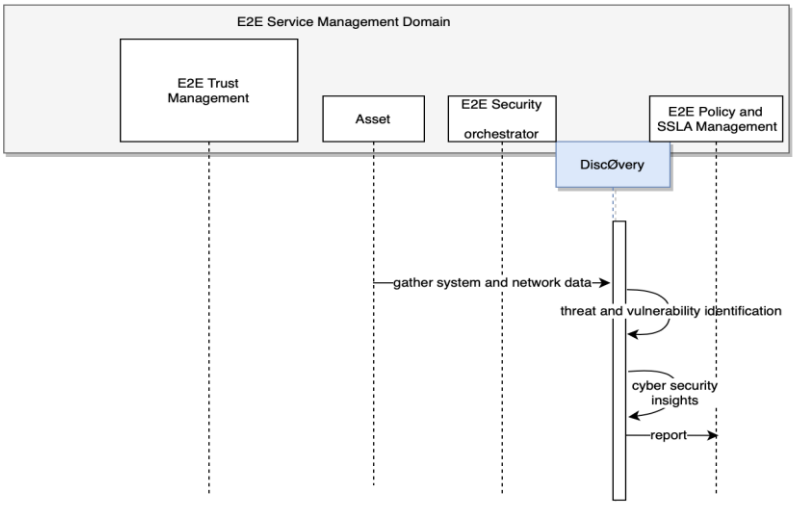

Figure 50: Sequence Diagram of DiscØvery

2.14.4 Technical Implementation

DiscØvery's technical implementation is based on the Apparatus Modelling language[43] that provides a framework for security analysis of complex systems and networks. The DiscØvery tool can model network in the following engineering phases:

1. design phase (model the high-level concepts of a network)
    1. identify threats

[43] Mavropoulos, O., Mouratidis, H., Fish, A., & Panaousis, E. (2019). Apparatus: A framework for security analysis in internet of things systems. Ad Hoc Networks, 92, 101743.

    2. identify assets
    3. identify constraints
    4. outward facing nodes
    5. model social engineering attacks
    6. develop a high-level security policy
    7. model access control tokens
2. implementation phase (model the low-level concepts of the network)
    1. identify threats
    2. identify vulnerabilities
    3. identify assets
    4. identify constraints
    5. identify mechanisms
    6. outward facing nodes
    7. model social engineering attacks
    8. develop a low-level security policy
    9. model access control tokens
3. state diagrams (model the different states of a network)
    1. model different configurations of a network based on detected events
    2. model different configurations of the network based on the security mechanisms
    3. identified events can be used as alerts in system monitoring applications

DiscØvery makes uses of a software component to consume network traffic to automatically generate models for analysis. Additionally, DiscØvery can be linked to vulnerability databased to identify vulnerabilities on network components.

2.14.5 Summary, lessons learned and guidelines

Guidelines

The DiscØvery enabler has a dedicated help function that provides a quick overview of the enabler's features. The help can be accessed by typing 'help' in the command line of the enabler. The 'help' action also displays the short keys and other commands that can be used to quickly access features. Additionally, the 'help' action also provides a quick menu to access the guide of the tool that is hosted on its Github page[44].

Lessons learned

During the development of the enabler, we streamlined the process of customising the modelling language as well as the functions of the enabler that facilitate the security analysis and rely on the expertise of the end-user. For example, end-users may need to extend the modelling language of the enabler to express additional concepts or update the description of certain threats and constraints that are provided by the enabler's analysis options. All the configurable options are easily accessible by the enabler's settings menu.

Future extensions

[44] https://github.com/CyberLens/Discovery

For the future work of the enabler, we aim to extend its visualization and security analysis logic to other domains such as the energy infrastructure. Additionally, we plan to refine the software architecture of the enabler to be more easily integrated with other cybersecurity tools to improve their network visualization and security assessment.

2.15 SSLA Manager

2.15.1 Overview

SSLAs (Security Service Level Agreements) have been defined in D3.2 as a way to capture the technical security requirements of an end-to-end secure Slice, driving the deployment of the necessary security controls to enforce it by enriching or configuring the services of the slice Service Providers (SPs); and the necessary monitoring techniques to verify that the security functions are working according to the agreed SSLA. In order to automate the security life cycle of a Slice, a machine-readable SSLA format is adopted based on the SPECS SSLA model that is extended to support slicing in the 5G context. This extension introduces security-related information allowing to specify the following Sections in a Slice term description:

1. Slice resource providers that describe the available infrastructure of the resource providers (appliances, networks, etc.);

2. Security capabilities required in a Slice service. A capability is defined as a set of security controls from industry standards, e.g. the NIST SP 800-53;

3. Security metrics referenced in the Slice service properties and used to define Security Service Level Objectives (SLOs) in the guarantee terms Section. A metric specification includes information about it and also information to process the SLOs, such as the metric name and definition, its scale of measurement, and the expression used to compute its value.

2.15.2 Role in the HLA

The Figure 51 shows the role taken by the SSLA Manager’s component in the HLA architecture.

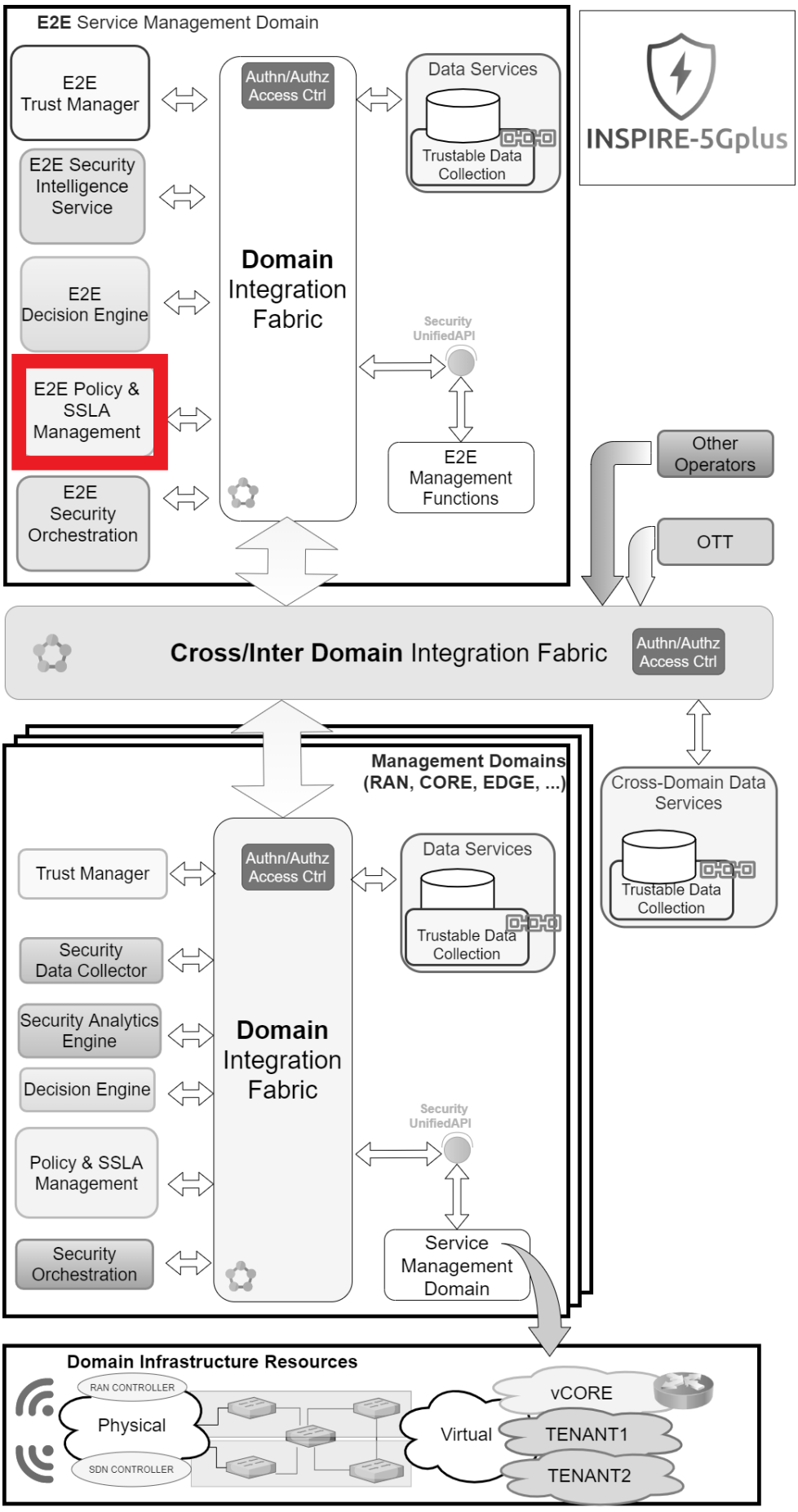

Figure 51: The SSLA Manager role in the HLA

2.15.2.1 E2E_PSM SSLA Storage Service

The SSLA Manager provides the E2E_PSM SSLA Storage Service (cf. HLA service table), i.e. handles the storage of the SSLA documents (WS-Agreement format extended by SPECS SLA model[45]) and versioning, and ensures their validity, especially their consistent relationships with the catalogues of services, security capabilities, security controls and security metrics. It also provides the necessary REST API to create (with syntactic and semantic validation), read, update and delete them.

2.15.3 Sequence diagrams

In a typical SSLA Manager use case, the customer initially defines high-level security requirements on an end-to-end secured slice that he/she wishes the E2E-secured Service Provider to provide; based on these requirements, and after negotiation with the Service Provider, the customer and Service Provider agree on a SSLA document (as shown in Figure 52).

Then, for a particular secured slice deployment, the customer sends this SSLA to the SFS Broker which then interacts with the SSLA manager for validating and registering active – enforced - SSLAs (for which SSLA-compliant slices are deployed). All the E2E services can now refer to the SSLA Manager for getting the full validated SSLA document when only the SSLA ID is passed as argument in the various interactions between them; and only the SSLA ID shall be used from now in the service calls (after initial SSLA registration in the SSLA manager). In particular, when the SFS Broker calls the E2E Slice Manager for slice instantiation (Network Slice Instance creation), it only passes the SSLA ID, and the E2E Slice Manager is able to get the complete SSLA document from the SSLA Manager given the ID. Also, later on, when a possible security incident is detected, the SSLA Manager may be queried to check for a potential SSLA non-compliance.

[45] https://bitbucket.org/specs-team/specs-utility-interoperable-data-model/src/master/resources/schemas/xml/SLAtemplate.xsd

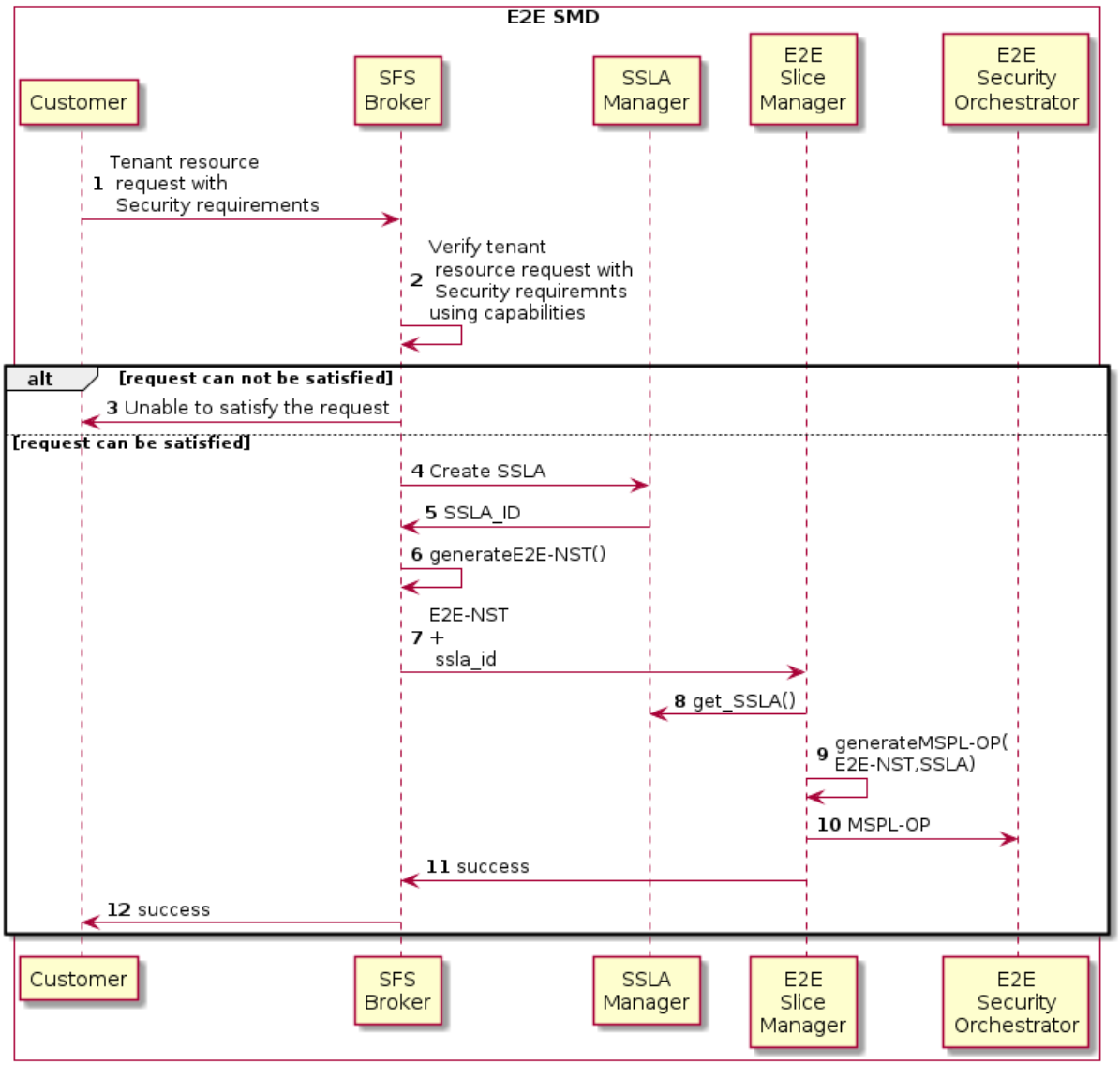

Figure 52 - SSLA Manager E2E SMD workflow

2.15.4 Technical implementation

The Figure 53 displays the current technical implementation and software blocks of the currently deployed SSLA Manager.

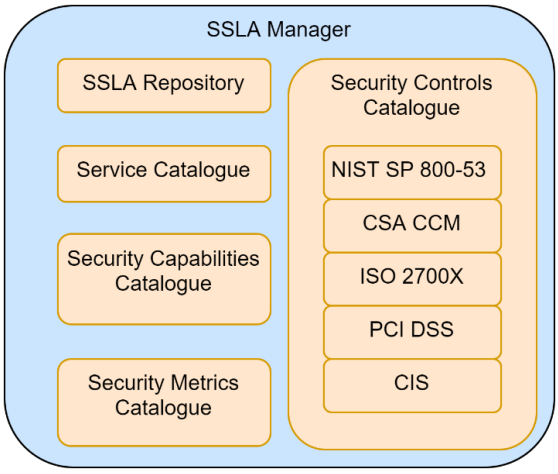

Figure 53 - The SSLA Manager main components

The SSLA Manager implements the REST API specified in OpenAPI format on the Swagger website:

1. https://app.swaggerhub.com/apis/INSPIRE-5Gplus/api-sla_management/

The API provides SSLA lifecyle management operations (creation, update, removal) and access to the managed SSLA content. The main managed objects are actually SSLA templates as defined by the SLATemplate XML schema based on the WS-Addressing standard and work from the SPECS project.

The implementation relies on:

1. SSLA Repository: a repository of SSLA templates applying and therefore referring to Services in a Service Catalogue, and associated security capabilities in the Capabilities catalogue and metrics in the Security Metrics catalogue;
2. The Service catalogue: services with security features to which SSLAs may apply.
3. The Security Capabilities catalogue: possible security features services may have, each capability grouping a set of security controls in the Security Controls catalogue;
4. Security Controls catalogue: set of security controls from the standard control framework: NIST SP 800-533, Cloud Security Alliance's Cloud Control Matrix, ISO 27000 series, PCI (Payment Card Industry) DSS (Data Security Standard), Centre for Internet Security controls.
5. The Security Metrics Catalogue: set of metrics definitions (metadata, unit, scale, etc.) relevant to the Capabilities.

2.15.5 Summary, lessons learned and guidelines

The concept of Security SLA has an important role in the provisioning of a 5G slice resources and its orchestration. Providing a framework that offers APIs to manage the whole Security SLA life cycle and to provide all the functionalities needed to automate their enforcement and to monitor the security features and their compliance to the SSLAs is a priority for the development of 5G security. As a future work, we intend to work on intelligent distribution algorithms of SSLAs overs SMDs during the orchestration process. The integration of SSLAs specification in TOSCA will also be studied to insure interoperability between SMDs.

2.16 Secured Network Slices for SSLA

2.16.1 Overview

Since its last description in D3.2[46], this enabler has evolved in terms of the management domain where it is going to be used. Originally, the enabler was thought to be placed in a (lower) Management Domain but due to the WP5 Demo1 requirements of multiple domains being used, it was requested to evolve this enabler in order to be used in the E2E Service Management Domain.

This requirement did not imply a big difference as the main functionalities are kept and what it really changed is the interaction with the other enablers. So, now, the interactions are with the (E2E) SSLA Manager, the E2E Security Orchestrator and the Policy Framework. For simplicity within the WP5 Demo 1 context, the name of this enabler was changed to "E2E Network Slice Manager".

2.16.2 Role in the HLA

The involved HLA components in the enabler implementation and their main relationship with other HLA components are illustrated below, in Figure 54.

[46] Documents download module (europa.eu)

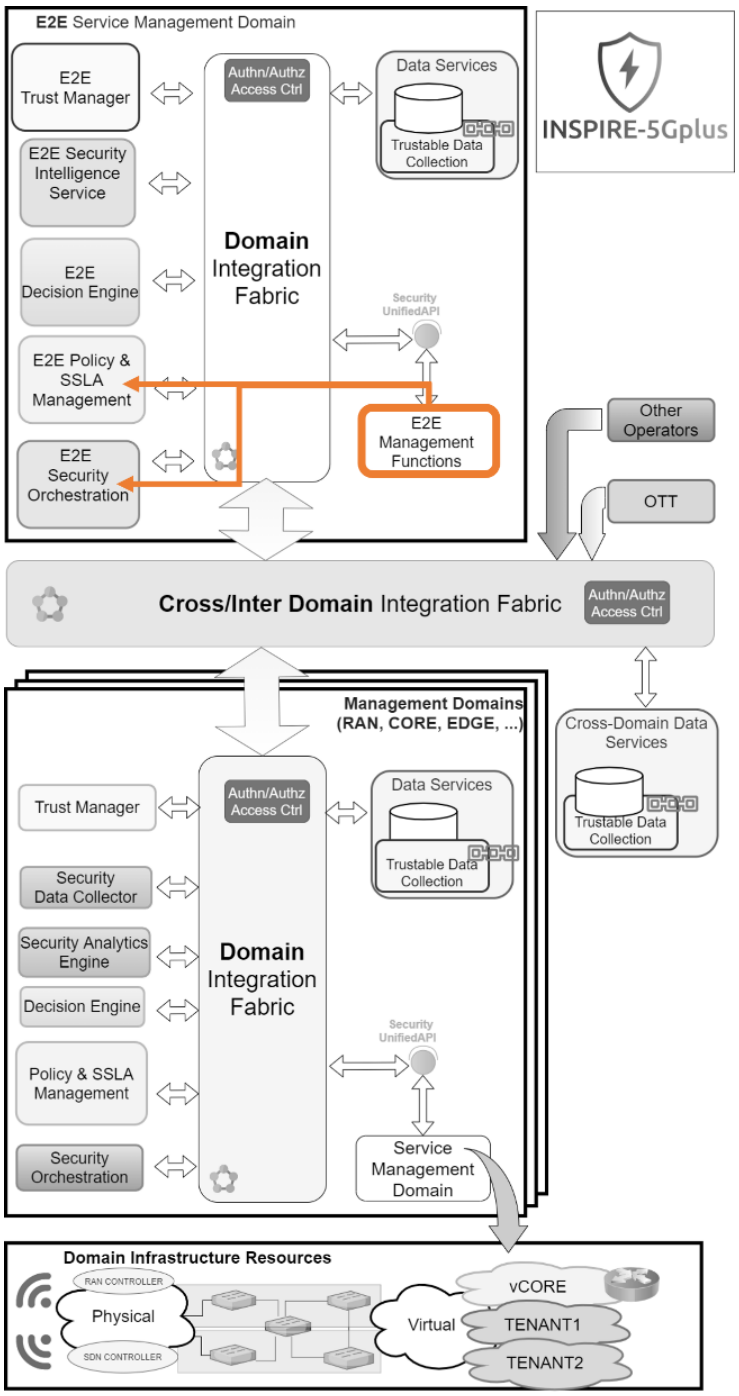

Figure 54: Secured Network Slices for SSLA enabler within the HLA.

2.16.2.1 E2E Network Slicing Management Service

This enabler provides to the system the capability to pre-define a set of networking and computing resources descriptors (i.e., Network Slice Template (NST)) that once instantiated creates a logical virtual network (i.e., Network Slice) specifically dedicated for a 5G vertical service or others like an Internet of Things (IoT) service. Due to the variability of services co-existing within the same infrastructure and each of them with its own specific Quality of Service (QoS) or Security (QoSec) requirements the enabler was specially designed to deal with the SSLA Manager in order to enforce the security elements around the Network Slice.

Based on the available service and security, the “Secured Network Slices for SSLA” (or E2E Network Slice Manager) enabler deals with the NSTs and the SSLAs stored in the SSLA Manager to create the Network Slice Instances (NSIs) with the information about the deployed resources and security metrics to be checked. Based on the requested SSLA, the “Secured Network Slices for SSLA” enabler requests

the most appropriate policy to the E2E Policy Framework. Then, with all the received information, it sends to the E2E Security Orchestrator the required information (in the shape of an MSPL object) which distributes the multiple domain MSPL requests to the different Security Orchestrators based on the domains involved. Finally, once the E2E Network Slice is deployed, the "Secured Network Slices for SSLA" enabler will receive the confirmation of all expected resource are ready to be used.

Similarly, when the users are done with the service and this has to be terminated, a similar procedure will be done with the "Secured Network Slices for SSLA" triggering the distribution of requests to terminate all the NSI elements.

2.16.3 Sequence diagrams

The previous service process is illustrated in more detail in Figure 55. The Vertical desiring a Secured Network Slice sends the request defining the NST and SSLA identifiers (step 1) to the Secured Network Slices for SSLA (SNS4SSLA) enabler. Then, the SSLA information is obtained from the SSLA Manager (steps 2/3) and the NSI based on the NST and the SSLA is generated (step 4) and from it, the MSPL (i.e., policy) element for the E2E Security Orchestrator (SO) is created and forwarded (step 5). The E2E SO orchestrates the corresponding deployment with the multiple domain SOs (steps 6/7) and once done, it informs back to the SNS4SSLA enabler (step 8). Finally, this updates the NSI object (step 9) and informs the tenant (step 10) about the e2E network Slice being ready to be used.

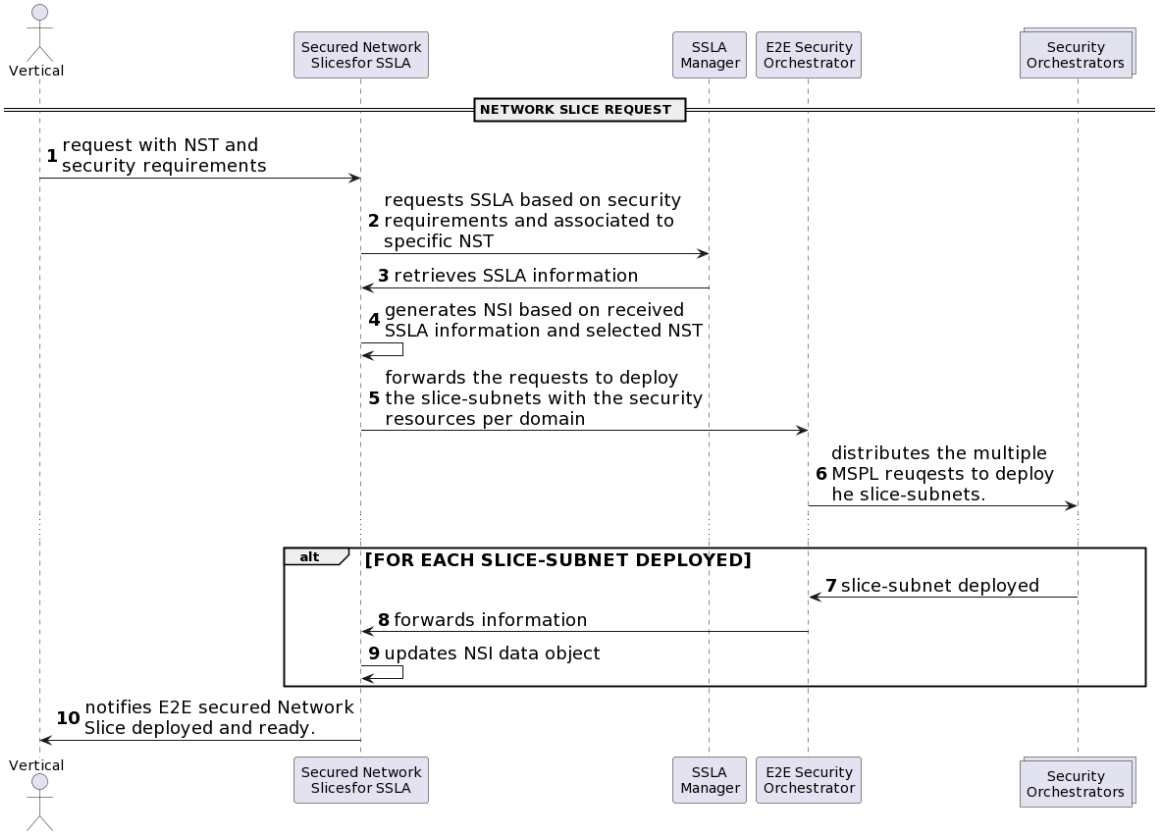

Figure 55: E2E Network Slicing Management Service Workflow.

2.16.4 Technical implementation

As previously commented, due to change in WP5 from the validation of multiple test cases to the validation of three demonstrators, this enabler has evolved its internal architecture and relationships with the external enablers from the one presented in D3.2[47].

47 https://ec.europa.eu/research/participants/documents/downloadPublic?documentIds=080166e5dd8df242&appId=PPGMS

As presented in Figure 56, now the architecture of this enabler has the following components:

1. main - access point for the enabler to receive the requests to deploy and manage network slicing related-actions.
    1. System Configuration: in charge to prepare the internal enabler processes such as the logging, threads to manage the received requests independently, etc.
    2. Settings: it defines specific enabler parameters such as its own port or other enablers' ports.
2. Sec Slice Mngr. - This module is composed by:
    1. Slice Management: core ef the enabler, it manages the deployment of the Network Slice Templates (NSTs) with the associated SSLAs to generated the corresponding Secured Network Slice Instance (NSIs).
    2. MSPL generator: it takes care to convert the generated NSI information into an MSPL data object for the E2E SO.
3. Repositories - It contains the data objects necessary to deploy and manage Secured network Slices. This modules contains a Database (DB) for the NSTs and a DB for the NSIs.
4. slice_pf: in order to accomplish the HLA architecture functionalities described in the INSPIRE5G-plus architecture and for the correct development of the WP5 Demonstrator1, a basic E2E Policy Framework functionality for Network Slices was added in this enabler. In there, a set of policies are defined and with the data coming from the E2E Data Services the right policy is selected based on the capabilties available in the multiple domains below the E2E domain.
5. Mappers - Composed by a set of individual sub-modules, each with the specific requests to send information to external enablers. These are:
    1. E2E_SSLA: to retrieve the information from the E2E SSLA Manager enabler.
    2. E2E_SO: to request the deployment of the Network Slices across all the involved Management Domains through the E2E Security orchestrators.
    3. E2E_DS: to rquest the capabilities available in the multiple domains.

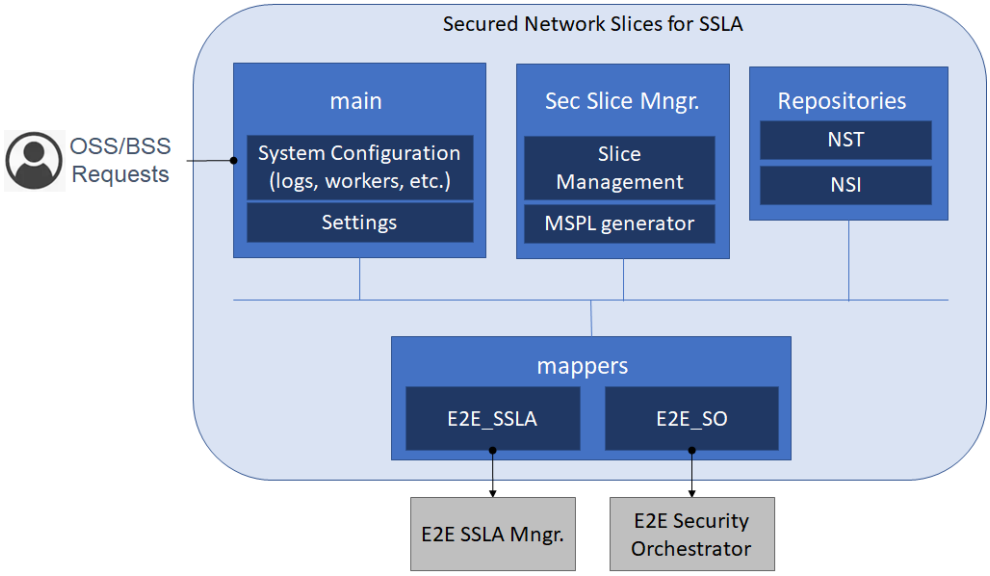

Figure 56: Secured Network Slices for SSLA internal architecture.

2.16.5 Summary, lessons learned and guidelines

Guidelines

As each element within the enabler has its specific functions, the enabler architecture follows a modular design. This allows to improve and update each module independently from the others. Without affecting those elements that are working properly while improving other modules or solving possible issues. The ode ok this enabler is in the Inspire5G-plus Github respository, available to be used

together with the other enablers.

Lessons learned

During the development we have addressed some issues that have to be taken into account for future developments, such as the development of a basic Policy Framework functionality for network slices. Moreover, the enabler has added new data object elements such as the use of the MSPL for the E2E SO implemented in XML.

Future extensions

As future work, we consider that the Policy Framework functionality with policies for network slices should be improved, either with a new enabler or within the current one but its internal procedures will need for sure an improvement.

2.17 Policy Manager

2.17.1 Overview

The Policy Manager (PM) component transforms the abstract Protection Level requirements and constraints expressed by consumers and providers into specific parameters that indicate, to the Security Orchestrator, the security services to configure, deploy and manage. The PM provides a framework defining the language and semantics to define Security Service Level Agreement (SSLAs) based on policies. These policies will be refined from a high abstraction level description to deployment-ready representations performing in its way conflict detection procedures to avoid inconsistencies. These values will finally be enforced in real time in cooperation with other INSPIRE-5Gplus functions.

2.17.2 Role in the HLA

Figure 57 shows the Policy manager module inside the HLA. It is in charge of refining, translating and detect policy conflicts. To this aim, it provides the refinement service, MSPL translation service, policy storage service, conflict detection service.

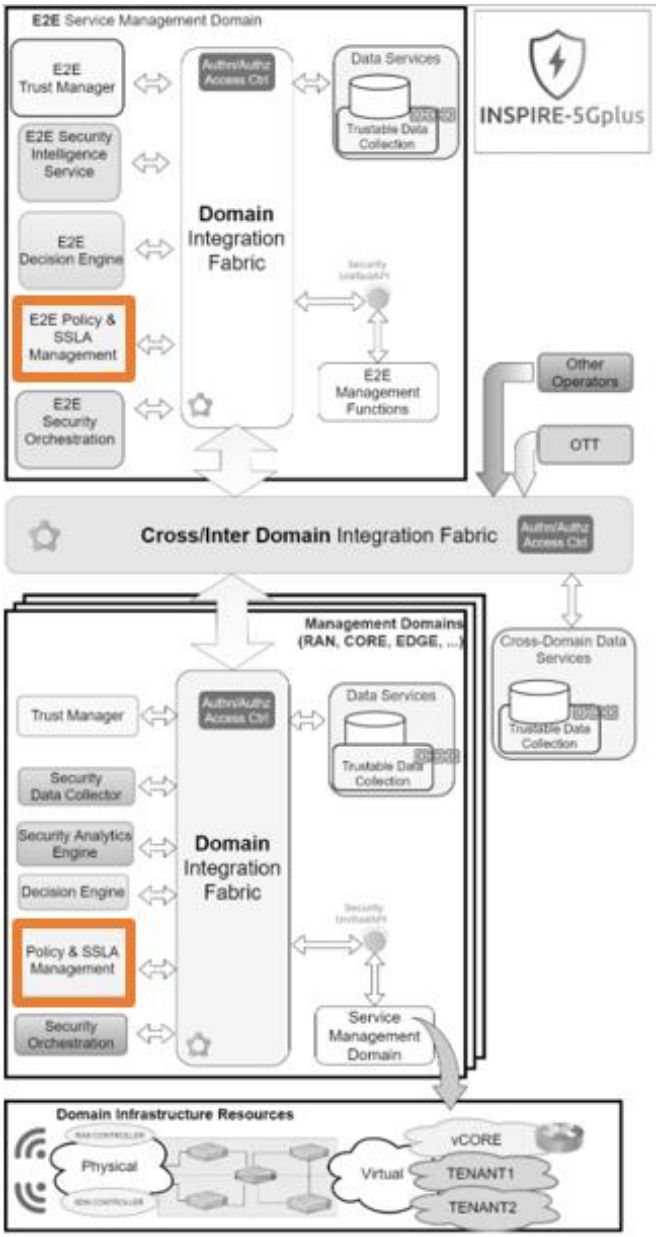

Figure 57: Policy Manager roles in HLA

2.17.2.1 PSM HSPL Refinement Service

The Refinement Service refines an HSPL-OP policy into MSPL-OP policy

2.17.2.2 PSM MSPL/TOSCA Translation Service

The Translation Service is an API entry point that receives an MSPL-OP policy as input and performs a translation process which returns the final precise configuration to be enforced on calculated asset/enabler. Also a translation could be made from MSPL to TOSCA in order to perform orchestration operations to other orchestrators (e.g., OSM, ONAP...).

2.17.2.3 PSM Security Policy Storage Service

The Security Policy Storage Service is in charge of storing the enforced security policies among the system, it is clearly useful to state the context of the system, and to perform further effectiveness analysis and conflict detection procedures.

2.17.2.4 PSM Policy Conflict Detection Service

Policy Conflict Detection Service performs the conflict detection by taking the current status of the system and the requested enforcing security policy, and detecting and notifying potential inconsistencies between them. Also, the Policy Conflict Detection Service perform an analysis of incompatibilities inside the security policy itself.

2.17.2.5 E2E_PSM HSPL Refinement Service

The Refinement Service refines an HSPL-OP policy into MSPL-OP policy

2.17.2.6 E2E_PSM Security Policy Storage Service

The E2E Security Policy Storage Service is in charge of storing the enforced security policies among the E2E system, it is clearly useful to state the context of the system, and to perform further effectiveness analysis and conflict detection procedures.

2.17.3 Sequence diagrams

Figure 58 shows the main interactions with the Policy Framework. In case that E2E security orchestrator receives High-level Security Policy Language Orchestration Policies (HSPL-OP), they must be orchestrated by performing a conflict detection and refining the HSPL-OP to MSPL-OP. Then the following steps are the same given the case that a MSPL is received as first input. The MSPL-OP orchestration process starts by performing a conflict detection at this abstraction level, then the normal orchestration procedure is performed as described in section 2.12.3. Once the SMD SO have received the MSPL-OP it starts the orchestration procedure by performing a conflict detection at this level and translating the MSPL-OP into final asset configuration, finally the enforcement procedure is performed as described in section 2.13.4.

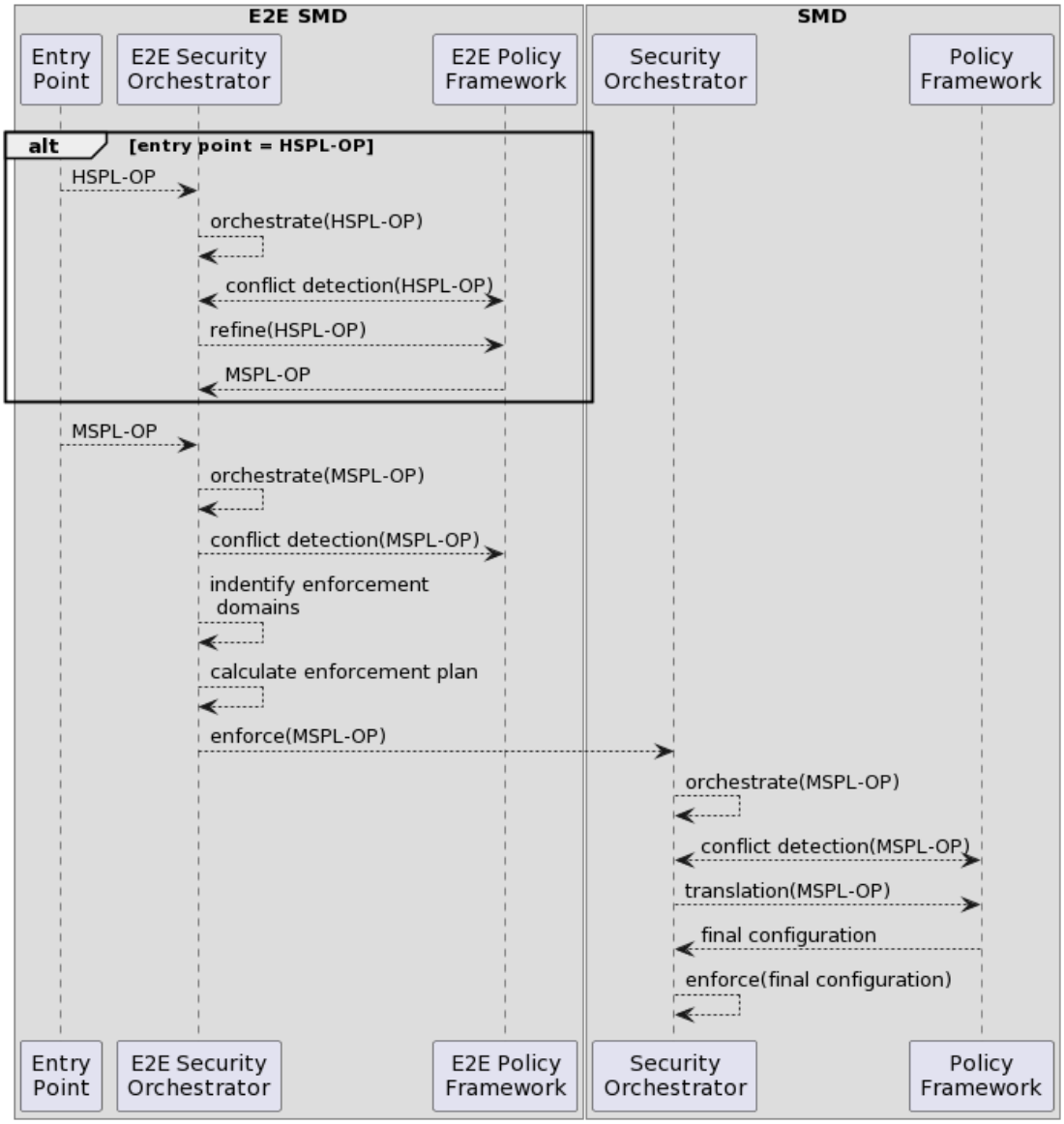

Figure 58: Policy Framework sequence diagram

2.17.4 Technical implementation

Figure 59 shows the current technical implementation and software blocks of the current deployment: Policy Repository with the Policy Repository Service. Policy Interpreter: with Conflict Detector, Policy Refiner & Policy Translator Services. and the Security Enablers Providers: with Plugins as services to provide Final Configuration Plugins.

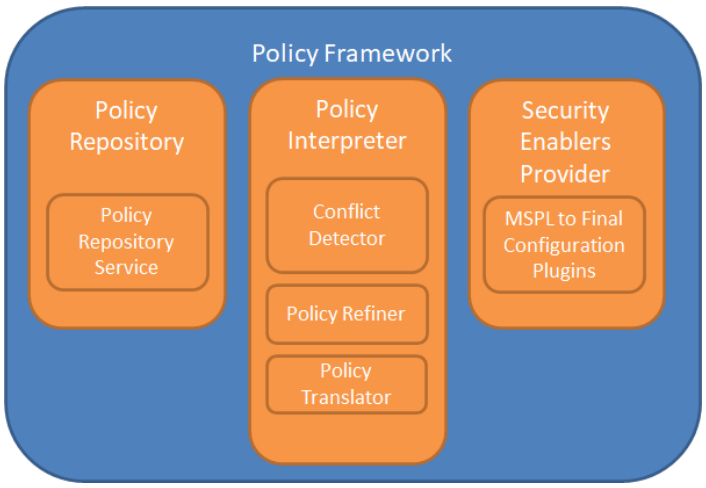

Figure 59: implementation and software blocks of the Policy Framework

1. Policy Repository Service: This fundamental block maintains a record of the policies enforced along the system and the current status of them, it eases the traceability and management.
2. Policy Interpreter: This fundamental block performs all kind of operations required for the orchestration procedure and is divided into 3 main blocks.
    1. The Policy Refiner: is in charge of refining High Level abstraction Policies into Medium Level abstraction Security Policies, and also it has been extended to perform refining of MSPLs-OP into more detailed MSPLs-OP in case some ambiguity in any capability is found (e.g. Channel Protection Capability into IPSec capability), to resolve this ambiguity the ambiguity is sent to the Conflict Detector. This module is mainly used at E2E level.
    2. The Policy Translator: translates MSPLs-OP into final specific asset configuration, used at SMD level. This block is supported by the Security Enablers Providers which offers plugins that ease the translation process depending on the security asset / enabler.
    3. The Conflict Detector: performs conflict and dependency check inside the MSPL-OP received. This module is based on an IA engine that via inference process is capable of detecting conflicts and dependencies between policies to be enforced and already enforced policies. This module has been extended to resolve ambiguities in certain capabilities where the domain's capabilities referred by the policy are consulted to select the best fitted option (e.g., domain 1 & 2 have Channel Protection capability, the Conflict Detector select among all options available, one that can be enforced by both domains).
3. Security Enablers Provider: Provides required plugins by the Policy Translator to perform the precise translation into specific security asset.

Policy models have been extended in order to represent services as part of a 5G E2E Security Slice, these services are in accordance with the E2E Security Orchestrator and the NFV MANO controller developed for the Security Orchestrator. The services are intended to be deployed as part of a 5G Security Slice, where they will be part of an E2E Network Slicing with Security Services associated to each sub-slice in which the E2E slice is divided for each domain that crosses the slice. Further details about model extension are explained below. As part of the extension of the policies new capabilities have been modelled to represent capabilities related to 5G infrastructure and E2E services deployment. In particular the following capabilities have been added:

1. Network Slicing: deploying a network slicing in a given domain.
2. 5G_Security_slicing: deployment of an E2E network slicing with associated security capabilities, in

general will be given by a secured service and the underlying network infrastructure with the security services given by the policies. Proof of Transit: the ability to monitor a protected channel with integrity,

3. 5GCore: the ability to deploy a 5G core as a 5G service
4. 5G_Secured_Service, ability to deploy a 5G service with associated security

Every Configuration rule of a policy is composed of: An Action, a condition and a series of policies necessary to apply them. Actions and conditions represent within a policy the behavior that the system must follow in order to apply the policy and some restriction that must be met in order to carry out the action, respectively. The actions and conditions have also been extended according to the extension of the policies in order to represent the 5G Security Slice and Secured Service. In particular, as possible actions of both we have those of: DEPLOY, UPDATE and REMOVE. While the conditions for a 5G Security Slice are that a Secured Service is present.

```
<ITResourceOrchestrationid="mspl_9f1a88b4fc67421b98de270d5a63d35f"
  <ITResourceid="mspl_eef61525d1594412bdcef34a4bfb7fc9"
    orchestrationID="mspl_9f1a88b4fc67421b98de270d5a63d35f" tenantID="1">
   <configuration xsi:type="RuleSetConfiguration">
     <capability xsi:type="FiveGSecuritySlice">
       <Name>5G_Security_Slice</Name>
       <sliceID>1548</sliceID>
     </capability>
     <configurationRule>
       <configurationRuleAction xsi:type="FiveGSecuritySliceAction">
        <fiveGSecuritySliceActionType>DEPLOY</fiveGSecuritySliceActionType>
        <securedService>
        <service id="62">
             <name>5GService</name>
            <type>5GCore</type>
            <domainID>1</domainID>
         </service>
         <securityRequirements id="asb6723bdqw87hd">
          <sloID>1</sloID>
           <sloID>2</sloID>
          <sloID>3</sloID>
        </securityRequirements>
        <securityPolicies >
          <msplID>mspl_9f1a88b4fc67421b98de270d5a63d35a</msplID>
          <msplID>mspl_eef61525d1594412bdcef34a4bfb7fc9</msplID>
          <msplID>mspl_eef61525d1594412bdcef34a4bfb7fc7</msplID>
        </securityPolicies>
         </securedService>
       </configurationRuleAction>
      <configurationCondition>
           <isCNF>false</isCNF>
        </configurationCondition>
        <Name>Rule0</Name>
        <isCNF>false</isCNF>
       </configurationRule>
       <Name>Conf0</Name>
     </configuration>
 </ITResource>
  <ITResource id="mspl_9f1a88b4fc67421b98de270d5a63d35a "
      orchestrationID="mspl_9f1a88b4fc67421b98de270d5a63d35f" tenantID="1">
     …
 </ITResource>
</ITResourceOrchestration>
```

This example represents the extension made on the policy model with some of the new capabilities added. In particular this example shows the refinement made from the SSLA1 of DEMO1 to MSPL-OP. This is an Orchestration Policy which capability is a 5G Security Slice, as show in the example, the model has been extended to have a sliceID that represents the E2E Slice among the system. A 5G Security Slice is formed by, the proper slice, an action and a secured service. The action dictates what the security orchestrator must do with the information received, in this particular case, it is to deploy the slice. The secured service, is a 5G Service, which usually could be virtualized, a 5G Core is represented as a Service to be deployed as part of a 5G Secured Slice, but others could be also deployed (V2X Service, VoIP, Extended Reality...). The service has also related the domain in which the service has to be deployed, this is important because the treatment the policy will receive with respect to that domain. The Security Requirements that are part of the Secured Service are the requirements presented by the SSLA, while the Security Policies are the ids of the MSPLs that represent the security capabilities for enforcing the security requirements of the SSLA.

2.17.5 Summary, lessons learned and guidelines

Guidelines

Policy Manager make use of the HSPL/MSPL-OP language to model all kind of security requirements, this language is derived from other standardized languages, enhancing their flexibility to express detailed security requirements. The Policy Management also include a conflict detection module that by the inference characteristic of the Rule Systems is able to detect conflicts between policies that are required to be enforced and the already deployed policies. Rule Engines are an AI approach to detect conflicts that enable the continuous growing of the Base Knowledge and perform Learning techniques.

Lessons learned

We have learned how to represent services as part of slices using the MSPL-OP modeling keeping a logical structure, but this was not easy because MSPL language was not designed to represent services but security requirements.

Future work

Regarding Policy Management, UMU expects to continue the security policy management research line (E2E and SMD level) for B5G/6G, especially focused on providing new models, not only for security management but also for service management. In this regard, it is expected to establish a synergy with other extended open-source languages like Topology and Orchestration Specification for Cloud Applications (TOSCA). In addition, new translation plugins and conflict detection rules could be defined to consider new policy models, capabilities as well as new conflict detection strategies.

2.18 SFSBroker

2.18.1 Role in the HLA

In the context of INSPIRE-5Gplus, a security enabler is presented for slice brokering, named as, SFSBroker (shown in the Figure 60). The security enabler is developed as a blockchain service and mapped as a Slice Service in the End-to-End Management Functions in INSPIRE-5Gplus HLA. SFSBroker facilitates the tenants to select the optimum network slice from the Mobile Network Operators (MNOs) for a resource request and utilizes a game theory-based algorithm encoded as smart contracts for the selection algorithm.

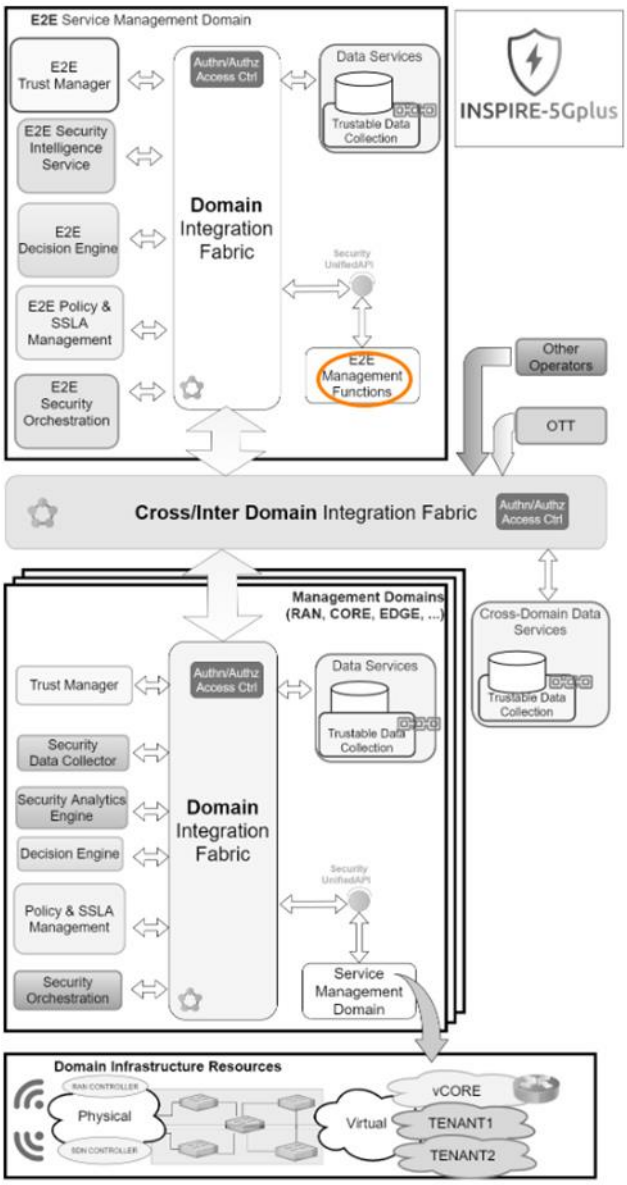

Figure 60: SFSBroker role in the HLA

2.18.2 Overview

Secure and Federated Network Slice Broker (SFSBroker) is a blockchain based service that governs the secure and privacy preserved resource trading between network operators/service providers at one end, and multiple network tenants at the other end. The SFSBroker architecture and its core modules are presented in D3.2. Figure 61 displays the SFSBroker components where the Prime Mover collects the resource requests coming from different production sites and forwards them to Mediator. Security Manager is a security service blockchain (SSB) to protect the entire brokering service from Denial of Service (DoS) attacks. When the resource requests are coming from the tenants, Mediator runs the slice selection algorithm by taking the updates of resource availability and price, creating the Network Slice Template (NST), and sending NST to Global Slice Manager. Finally, the Global Slice Manager in SFSBroker sends the recommendation of NST to blockchain-based SSLA manager to process further deployment of SSLA and then for the actual network slice deployment.

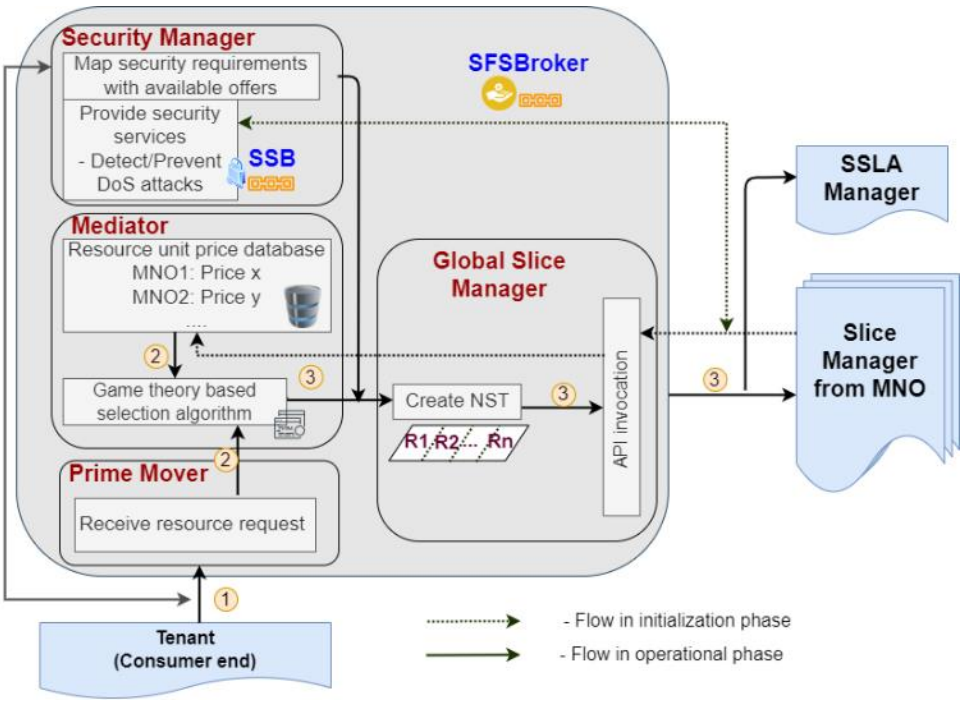

Figure 61: SFSBroker overview diagram

2.18.3 Sequence Diagram

Figure 62 explains the inner working of the SFSBroker as a sequence diagram involving its components.

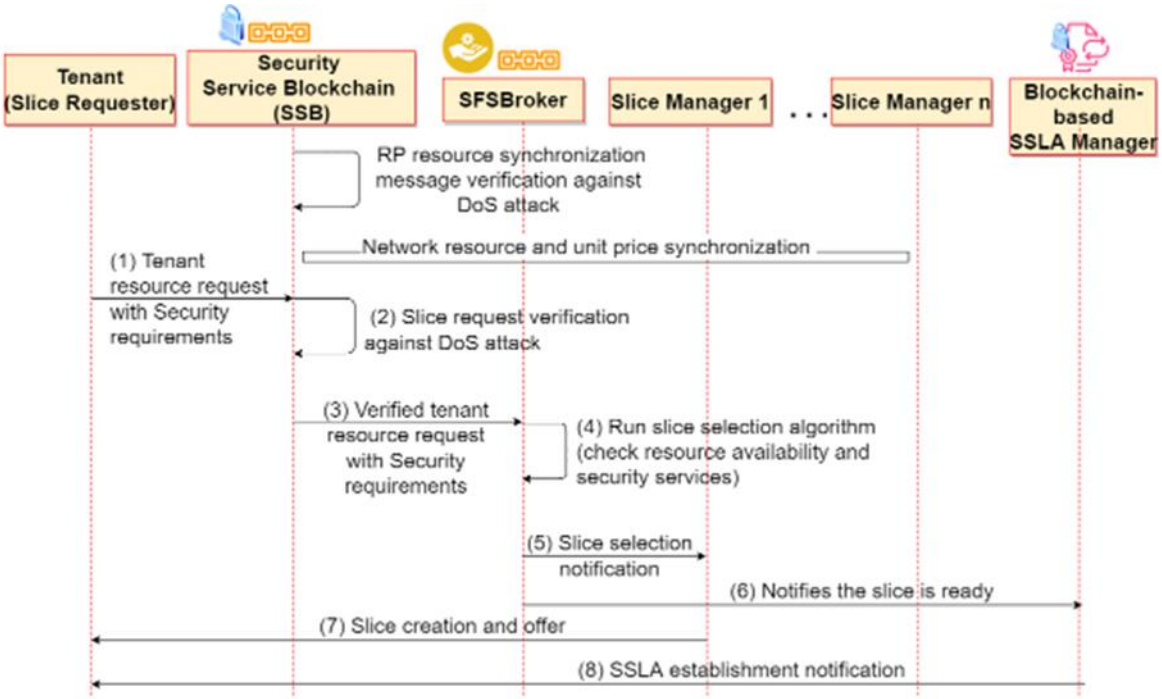

Figure 62: SFSBroker workflow

2.18.4 Technical Implementation

The proposed architecture leverages the slice brokering operation by encoding the selection algorithm into smart contract in the consortium blockchain. The consortium comprises of tenants and resource providers. As indicated in the Figure 63 below, the Tenants, slice managers and SSLA managers connected using APIs.

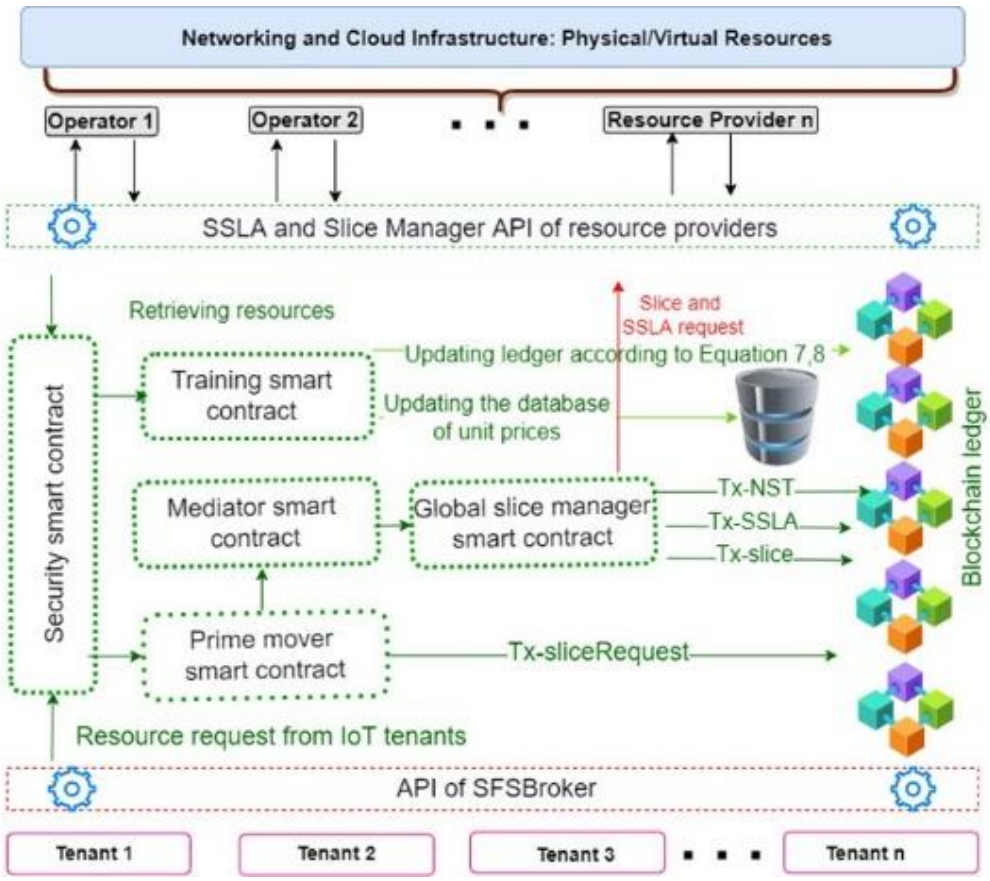

Figure 63: SFSBroker blockchain usage

2.19 Katana Slice Manager

2.19.1 Overview

Katana Slice Manager[48] is a software component that creates, modifies, monitors and deletes network slices. Katana includes a North Bound Interface (NBI) for receiving the Network Slice Template (NEST) and provides the API for managing and monitoring them. The NEST a descriptor where the network slice characteristics and parameters are described, such as the list of NFV components (Network Services) that need to be instantiated, WAN configuration, QoS, monitoring level, Life-Cycle stages, etc. The NEST must be written in a human/machine readable language, i.e., JSON or YAML. Based on the on-boarded NEST, Katana maps the available data plane resources and the described slice requirements, in order to create a requested slice. Katana communicates with the Network Sub-Slice Manager components of the Management Layer, namely the Virtual Infrastructure Manager (VIM), the NFV Orchestrator (NFVO), the Element Management System (EMS), and the WAN Infrastructure Management (WIM) through the South Bound Interface (SBI).

Katana Slice Manager is based on a mesh of microservices, each running on a docker container. The containerized approach allows a highly modular architecture that makes the applications independent of the underlying system while providing flexible maintenance and scalability.

2.19.2 Role in the HLA

Katana's mapping into the HLA components is depicted in Figure 64 and provides the E2E and domain Network Slicing Management Services:

[48] https://github.com/medianetlab/katana-slice_manager

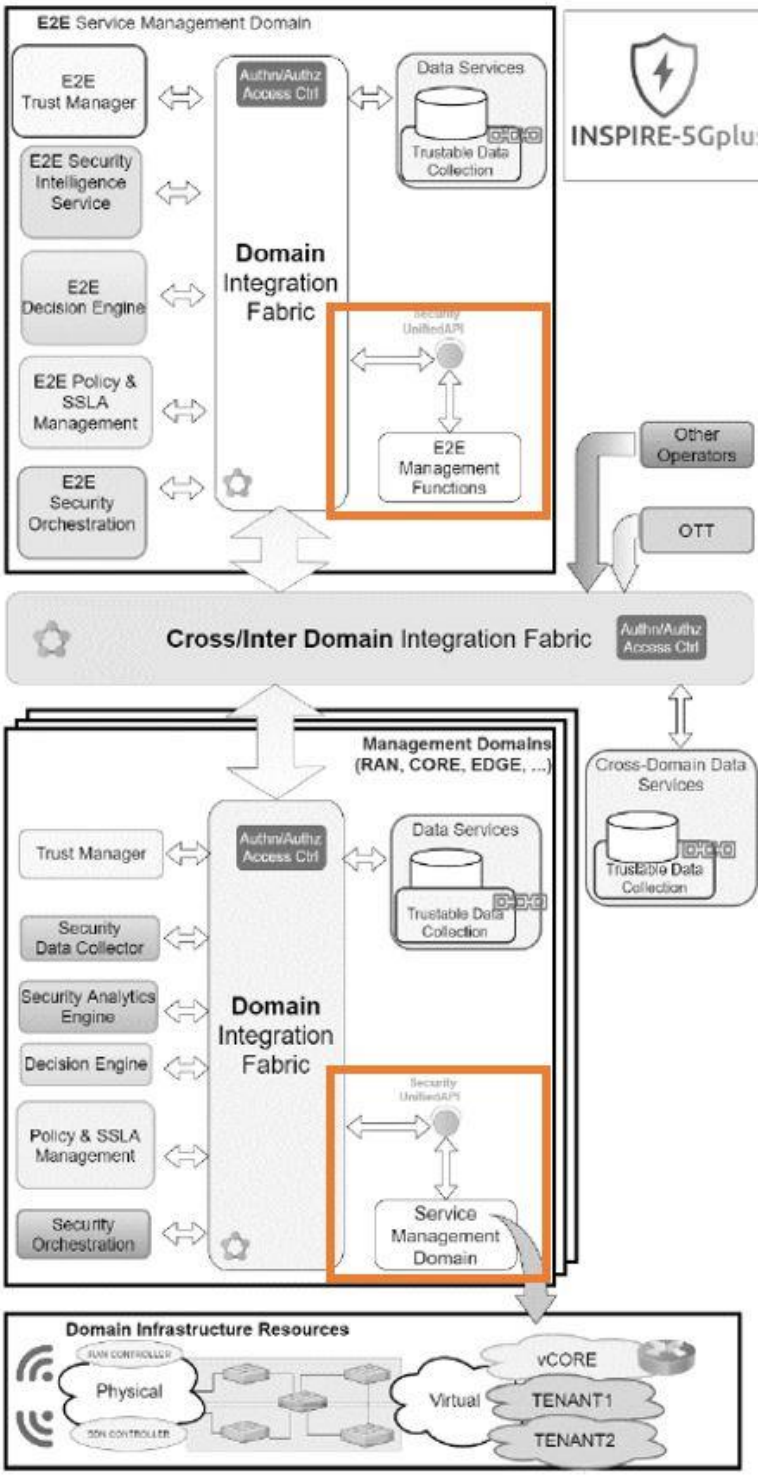

Figure 64: Katana Slice Manager mapping on the HLA

2.19.3 Sequence diagram

There are sequence diagrams of Katana for instantiating, modifying or deleting a network slice. In this document, we provide the modification sequence diagram. For reference, the interested readers can learn more on the Katana Github[49]. The Figure 65 depicts the slice update sequence diagram that contains four steps: i) instantiate a new network slice, ii) update/delete running network slices, iii) update transport network parameters, and iv) update/start/stop the radio network services. These

[49] https://github.com/medianetlab/katana-slice_manager/wiki/arch

steps cover the E2E network, including the transport and radio/core/edge management domains. Katana has been updated to handle Day-2 operations, meaning updating specific network parameters on a specific management domain without deleting the deployed E2E network slice. Katana receives requests via the HLA Security Orchestrator through its northbound API regarding slice creation, modification, or removal and interacts with the underlying components of the infrastructure via its southbound API.

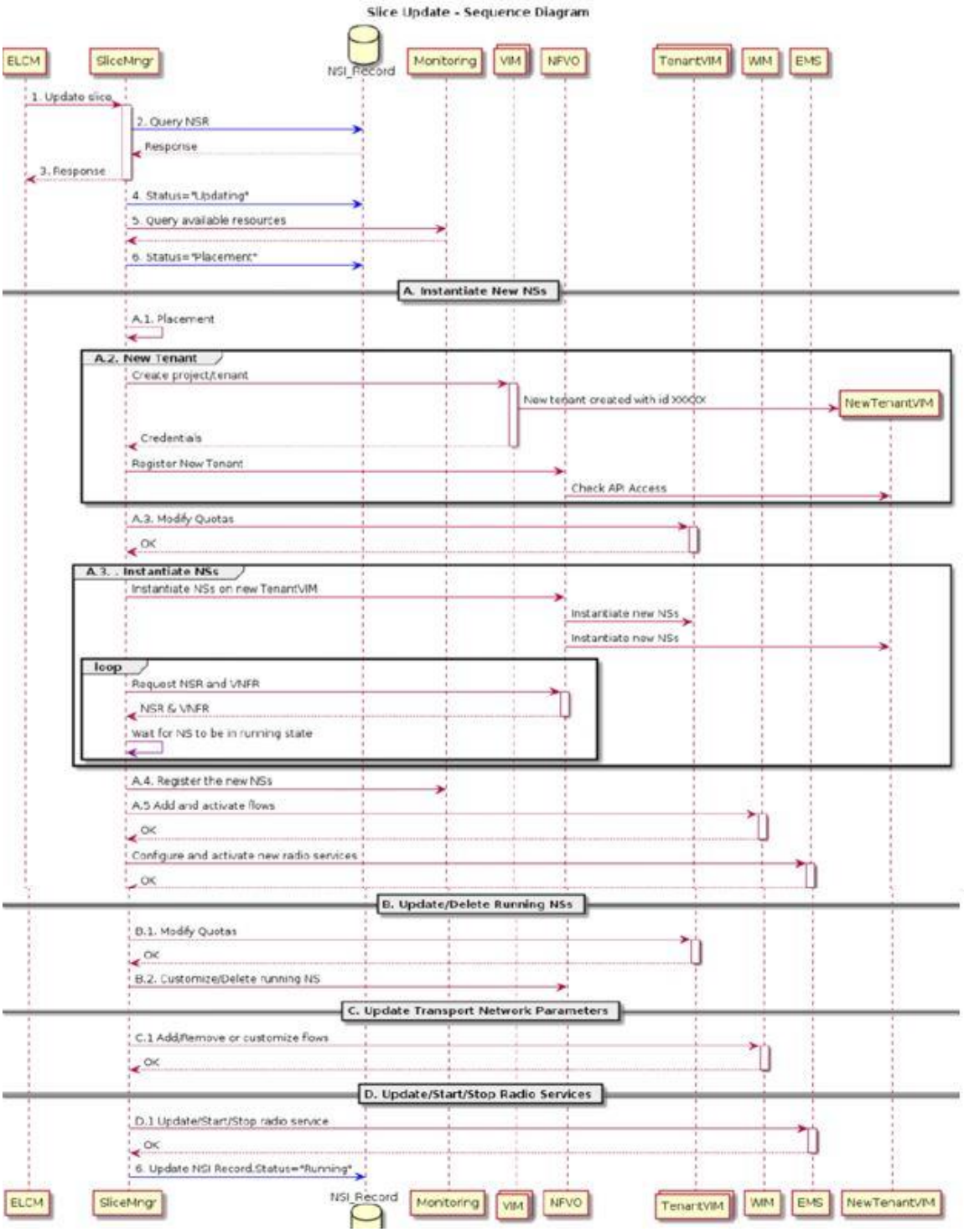

Figure 65: Katana Slice update sequence diagram

2.19.4 Technical Implementation

The Figure 66 illustrates the architecture of the Katana Slice Manager. Katana contains distinct microservices running as Docker containers. Katana is continuously updated with additional features

and interested readers can download the code in Katana's Github page: https://github.com/medianetlab/katana-slice_manager

To make a brief summary, the responsibilities of the major Katana components are:

1. Slice Monitoring: This module monitors the status of all the deployed network slices and alerts the Slicing Lifecycle Manager of potential changes.
2. Slicing Lifecycle Manager: This module receives the requests from the NBI, implements CRUD operations, and receives messages from the Slice Monitoring module regarding the status of the deployed network slices.
3. Slice Mapping: This module maps the GST request with Network Slice Templates (NEST) supported by the underlying infrastructure.
4. Slice Provisioning: This module receives requests from the Slicing Lifecycle Manager to setup, configure or delete the WAN paths.
5. Adaptation Layer: This module includes the VIM, NFVO, NMS and Monitoring plugins and abstracts the underlying infrastructure technology to the Slice Manager, separating its main functionality features from the underlying infrastructure.

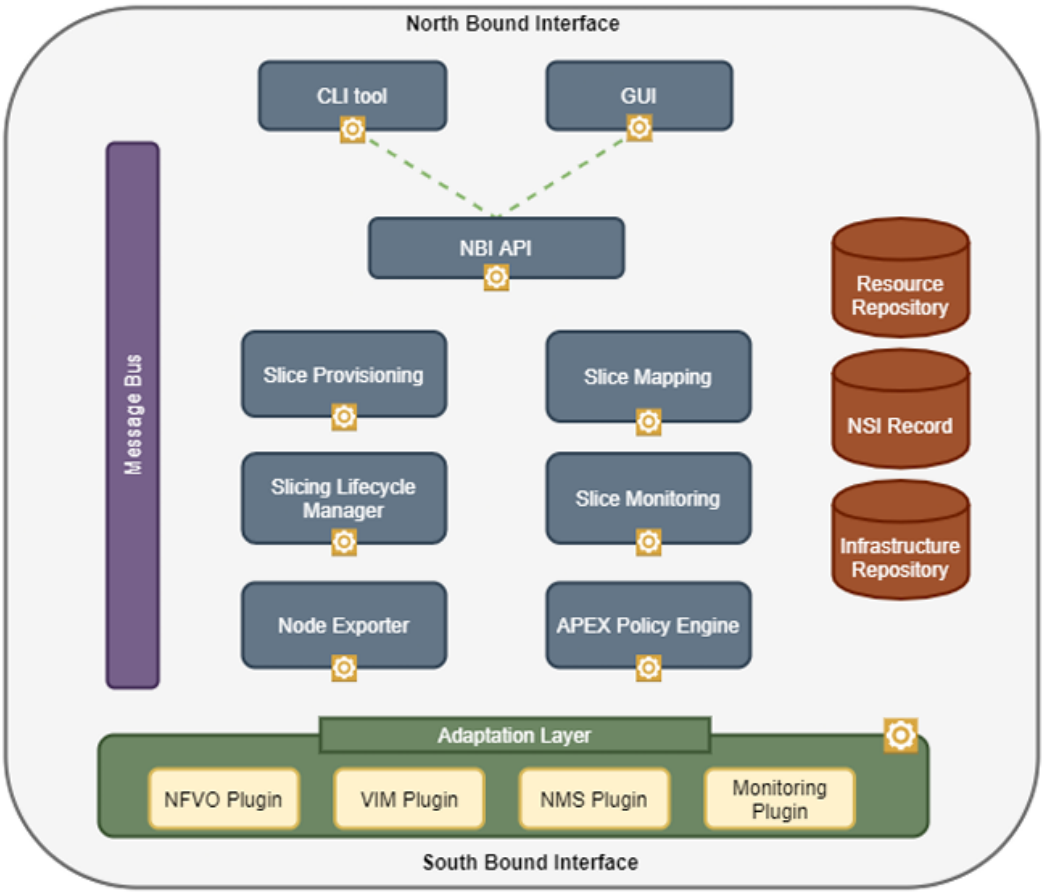

Figure 66: Katana Slice Manager architecture

Katana also provides a Swagger UI tool to document the NBI RESTful APIs, available at 'http://<katana_ip>:8001'.

The latest release of Katana Slice Manager introduces some new modules that, in conjunction with advancements in the existed modules, offer an enhanced set of operations related to the slice management processes. The most noteworthy new functionalities are the slice monitoring and the Day-2 operations.

Slice monitoring involves tracking and analysing network components and functions that the Slice

Manager has created and configured as part of a network slice. This process provides insights into the performance while ensuring continuous uptime and good health of the services realized by every network slice. Platform administrators can utilize the slice monitoring feature to quickly detect networking failures and determine in real-time whether the platform is running optimally. Katana Slice Manager capitalizes on Prometheus and Grafana to create a toolkit that offers monitoring, visualization, and alerting capabilities. In addition to these modules, an exporter tool has been developed for collecting metrics from the underlying infrastructure components. These tools are packaged in Docker containers and delivered as part of Katana microservices architecture.

Regarding the Day-2 operations on running slices, the Slice Lifecycle Management (LCM) module of Katana has been further developed to enable some specific actions on slices that have already been deployed. More specifically, it allows the creation and deletion of Virtual Network Functions (VNFs) within the context of the running slice. Moreover, it allows restarting VNFs created as part of the slice and optionally following any defined constraints (e.g., move to another cloud computing environment). These Day-2 operations are supported by the respective endpoint APIs, providing access to external users and services. Figure 67 shows the Swagger UI of those APIs endpoints.

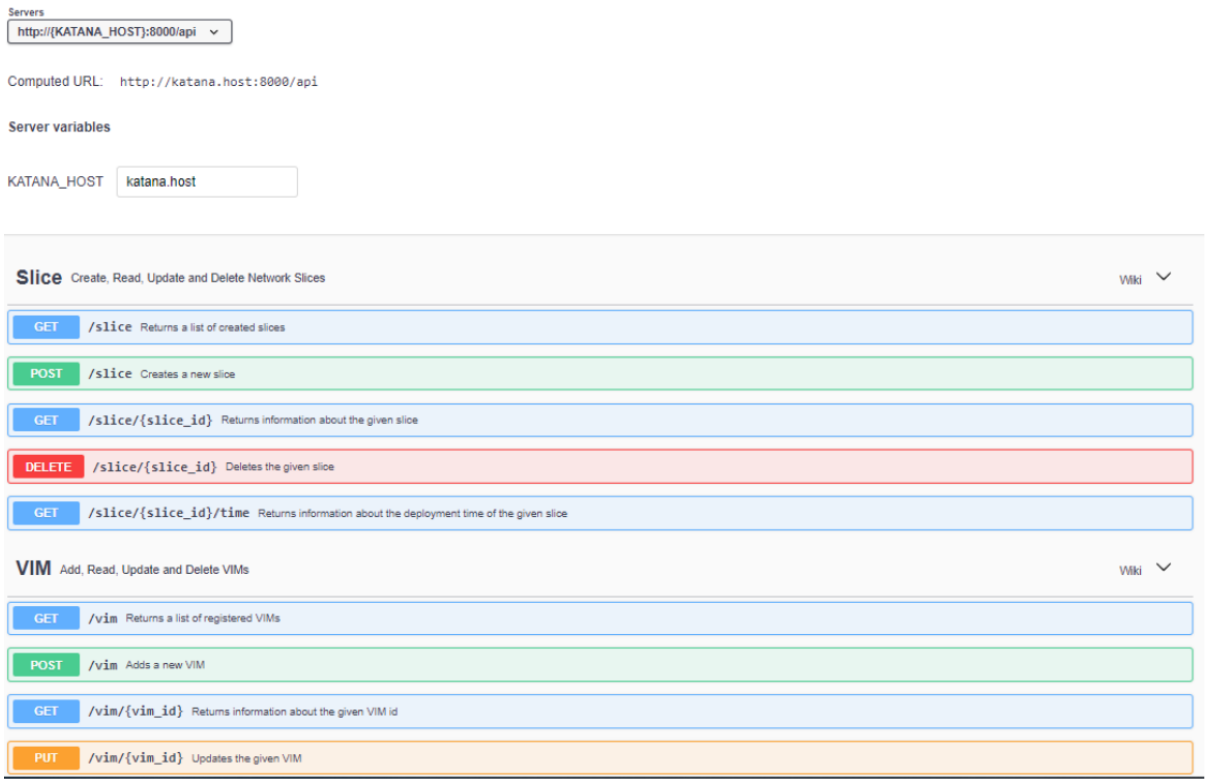

Figure 67: Katana Swagger tool

2.19.5 Summary, lessons learned and guidelines

Guidelines:

Slice Management in the future will be the cornerstone for enabling control and mitigation loops in software networks, service deployment and lifecycle management in future deployments should be driven almost exclusively through by Slice Manager.

Lessons Learned:

In order to get the full potential of the Slice Manager, a tighter integration and fine tuning with the OSS components available in a potential real implementation is required. In order to cope with dynamic changes during the operation state of the deployed services, extensions to Slice Manager were required to handle day-2 operations.

Future Extensions:

Future versions of the SLice Manager need to implement cloud native features and APIs, in addition unified approach has to be followed considering the policy descriptions and policy engine. The Slice Manager evolution will consider improvements under the following topics:

- Security Orchestration
- Dynamic Network Slicing support
- Mobility Management in Network Slicing

2.20 Integration Fabric

2.20.1 Overview

The ZSM approach relies on the use of integration fabrics. These fabrics provide communication and security capabilities between and within the SMDs as well as other service management features such as registration, discovery that needs to be performed inter/intra-domain. For instance, it allows access to data services but it is also able to ensure certain security properties in the fabric communications. This is, Integration fabric provides not only communication capabilities in different ways but also other interesting security and management features in a service mesh.

2.20.2 Role in the HLA

Figure 68 shows the integration fabric modules inside the HLA. It is in charge of providing different communication and security aspects intra and inter domains. Specifically, it provides domain and cross domain data access services, registration services, discovery services, invocation services and communication services among others.

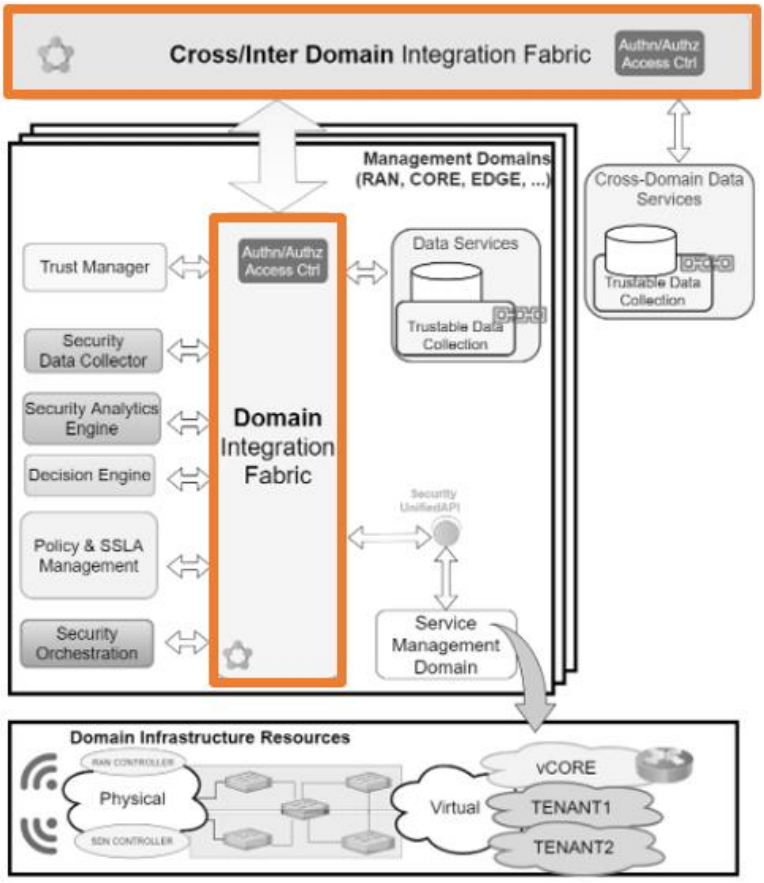

Figure 68: Integration Fabric role in HLA

2.20.2.1 DomLvl/CrossDom Data Access Service

The implementation allows accessing cross domain and single domain data services through management communication service and management service invocation routing capabilities.

2.20.2.2 IF Registration Service

The implementation allows register services, even if they are outside of the service mesh.

2.20.2.3 IF Discovery Service

As part of the implementation of the integration fabric, Istio provides service discovery capabilities[50].

2.20.2.4 IF Invocation Service

Current implementation allows services invocation through services routing. Besides, security properties can be applied dynamically for the services invocation such as traffic shifting, circuit breaking, mirroring or load balancing among others.

2.20.2.5 IF Communication Service

The implementation allows communicating services in different ways such as, REST APIs, custom flows or messaging queues. Security properties can be also applied to these communication services.

2.20.3 Sequence diagrams

Figure 69 shows an example workflow extracted from istio, as part of the messaging capabilities. It shows a kafka service that provide messaging support to internal and external services (e.g., Inter/Intra domain). The Kafka service is deployed together with an Istio sidecar that enhances the solution with Istio capabilities. As the figure shows, traffic can be received from internal services or from outside the service mesh (gateway). Security properties, routing, and other properties can be applied in both case before and after the traffic enters to the service. In the image, the PasstrhoughCluster represents and endpoint outside the service mesh where the messaging traffic is sent. Security properties can be also applied at this point.

[50] https://istio.io/latest/blog/2020/multi-cluster-mesh-automation/

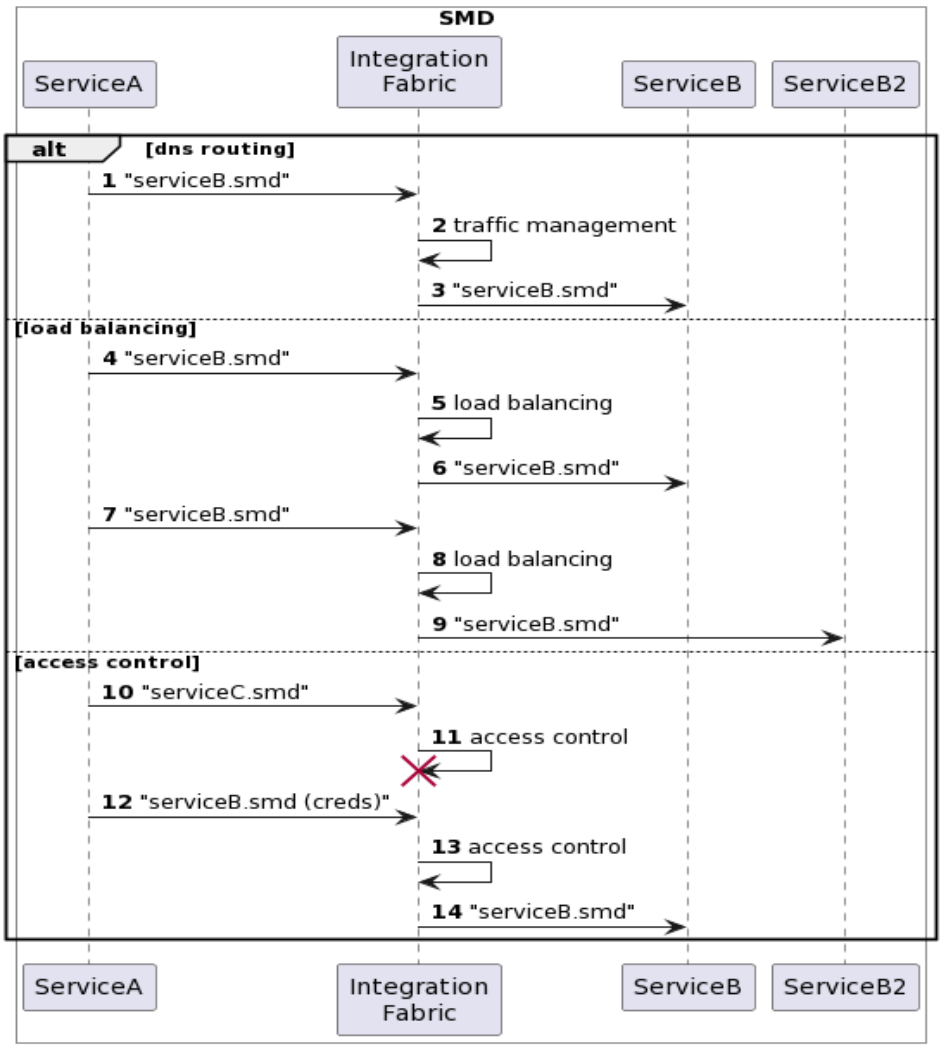

Figure 69: Integration Fabric flow example

2.20.4 Technical implementation

Figure 70 shows a deployment example of the integration fabric to communicate ISNPIRE-5GPlus control plane services with dynamic agents and services deployed dynamically across the SMD infrastructure. The fabric has been deployed a as a combination of different technologies to cope with the ETSI ZSM Integration Fabric requirements:

2. Management services registration service
3. Management services discovery service
4. Management communication service
5. Management service invocation routing service
6. Management capability exposure configuration service

Specifically, it has been deployed as a combination of Kubernetes, Istio and Kafka. Kubernetes provides a container-based high-availability (among others) environment for services. The deployment of Istio simplifies the observability, provides traffic management, security, and policy with the leading service mesh. It enhances basic Kubernetes deployment to provide most of the ETSI ZSM Fabric requirements. In addition, the deployment of Kafka inside of the Istio service mesh provides event streaming features to the fabric.

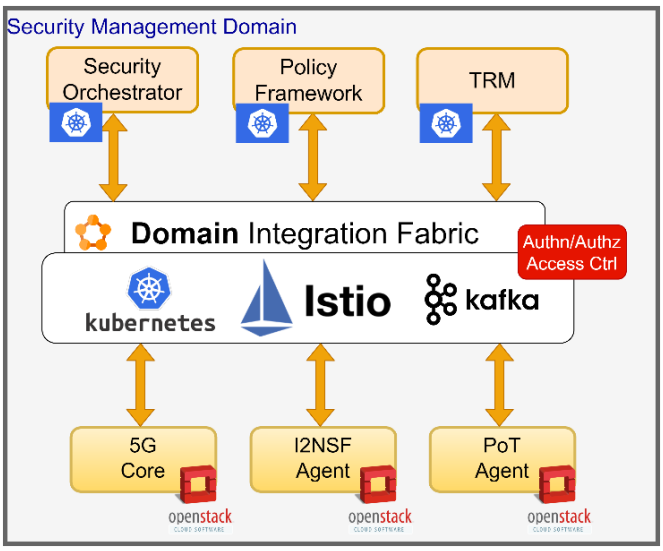

Figure 70: Integration Fabric deployment example

2.20.5 Summary, lessons learned and guidelines

Guidelines:

UMU advices using integration fabric approach proposed by ETSI in order to achieve features such as registration, discovery, communication, invocation, routing, exposure as well as providing a security overlay (e.g., AuthN/AuthZ).

Lessons learned:

After implementation and testing, we advise using an integration fabric implementation based on Kubernetes, Istio and Kafka since It copes with the mandatory features identified by ETSI among others.

Future extensions:

UMU expects to continue the Integration fabric research line (E2E and SMD level) for B5G/6G, especially in terms of collaborating to standardize technical proposals, as well as applying them to new projects. While right now the UMU proposal comprises multiple technologies it is possible that in the near-future appears new enabling technologies that implement natively all the integration fabric requirements.

2.21 Security Agent - 5G Core & Radio Agent

2.21.1 Overview

INSPIRE-5Gplus requires of security assets for monitoring and managing security at a local point. In particular these two agents are focused on granting monitoring and management of resources for security at the 5G Core and the Radio Access Network. They have the capability of extracting relevant information on demand, and performing actions required by the Security Orchestrator through the drivers introduced in section 2.13.4.

2.21.2 Role in the HLA

Figure 71 shows the role of the 5G Core & Radio agent on the HLA, as both are agents for monitoring and managing local resources of 5G components, they both are present in the Domain Infrastructure Resources, at the 5G Core and Radio systems respectively.

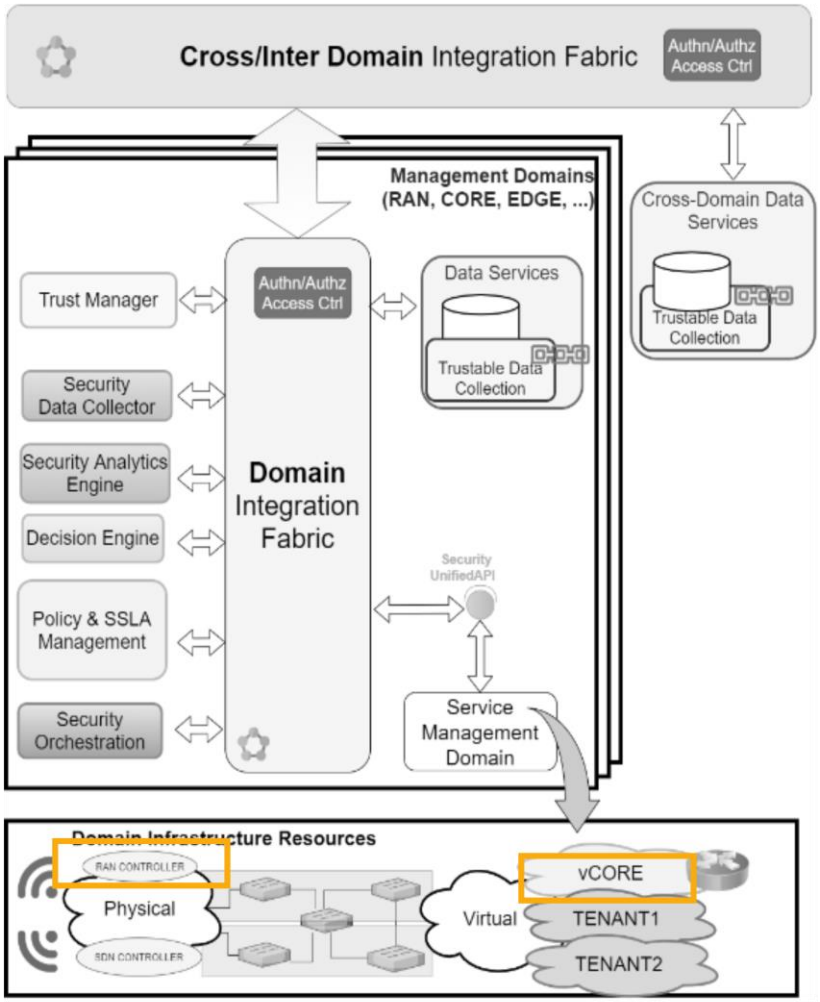

Figure 71: HLA role of 5G Core & RAN agents

2.21.2.1 Security Policy Local Enforcement Service

The implementation enables the enforcement of Security Policies through the application of determined operations at the 5G Core and 5G Radio infrastructure.

2.21.2.2 Network Monitoring and Telemetry Service

The implementation enables the extraction of useful data on-demand through the monitoring capabilities of the agents.

2.21.3 Sequence diagrams

The sequence diagrams in Figure 72 & Figure 73 shows the simplified action flow of both agents where they can receive the configuration to be enforced and execute them in the 5G system. Possible operations available in the 5GC agent are:

- deploy: deploy a Network Function (NF) in the 5GC.
- redeploy: redeploy a Network Function with a specified new version in the 5GC.
- get_info: extract required info from determined NF of the 5GC.
- stop: stop determined NF of 5GC.

Both agents are interconnected as they belong to the same 5G system, and they can exchange information to perform a 5G operation (e.g. new AMF registered). The 5G Radio agent is provided by

Amarisoft web socket. Thus, we have implemented a driver to manage the Amarisoft agent dynamically and autonomously.

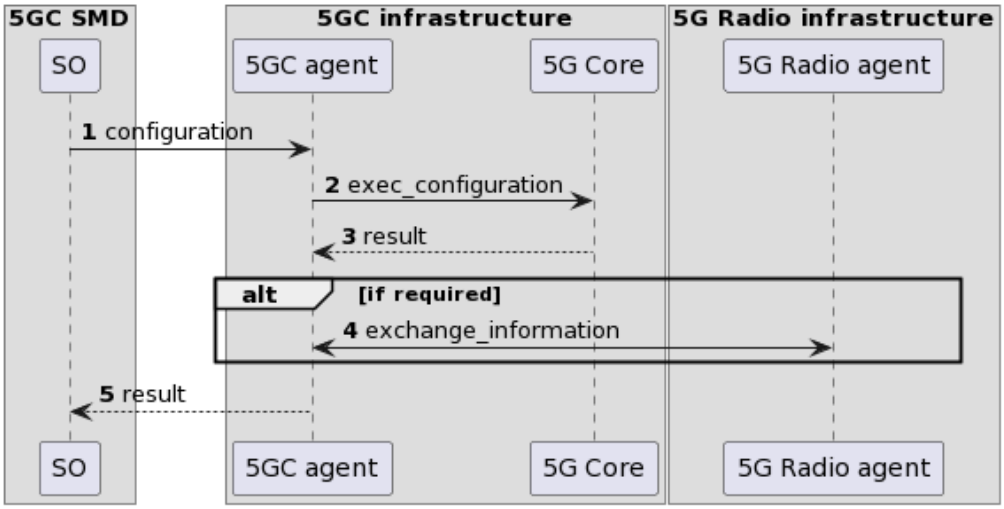

Figure 72: Sequence diagram of 5G Core agent

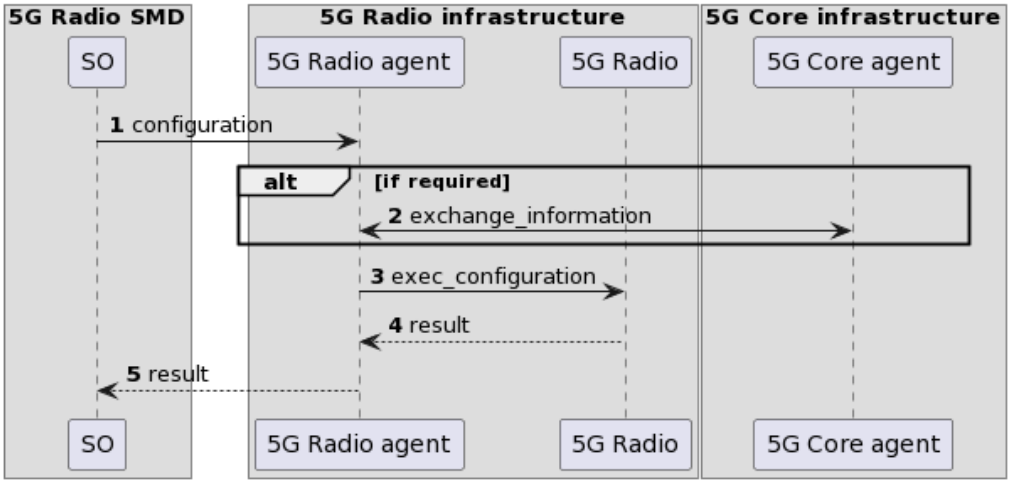

Figure 73: Sequence diagram of 5G Radio agent

2.21.4 Technical implementation

In Figure 74 functional split is depicted. Where both agents have only one functional block, in charge of receiving the requests and depending on the content of the request, execute the required action.

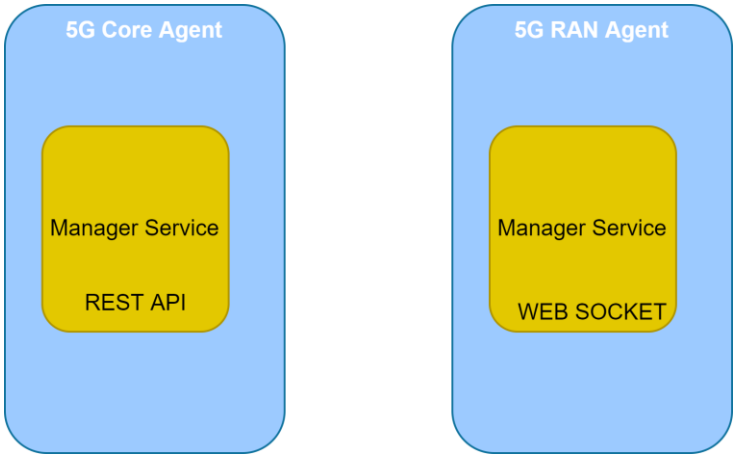

Figure 74: Technical functional blocks of 5G Core & Radio Security Agents

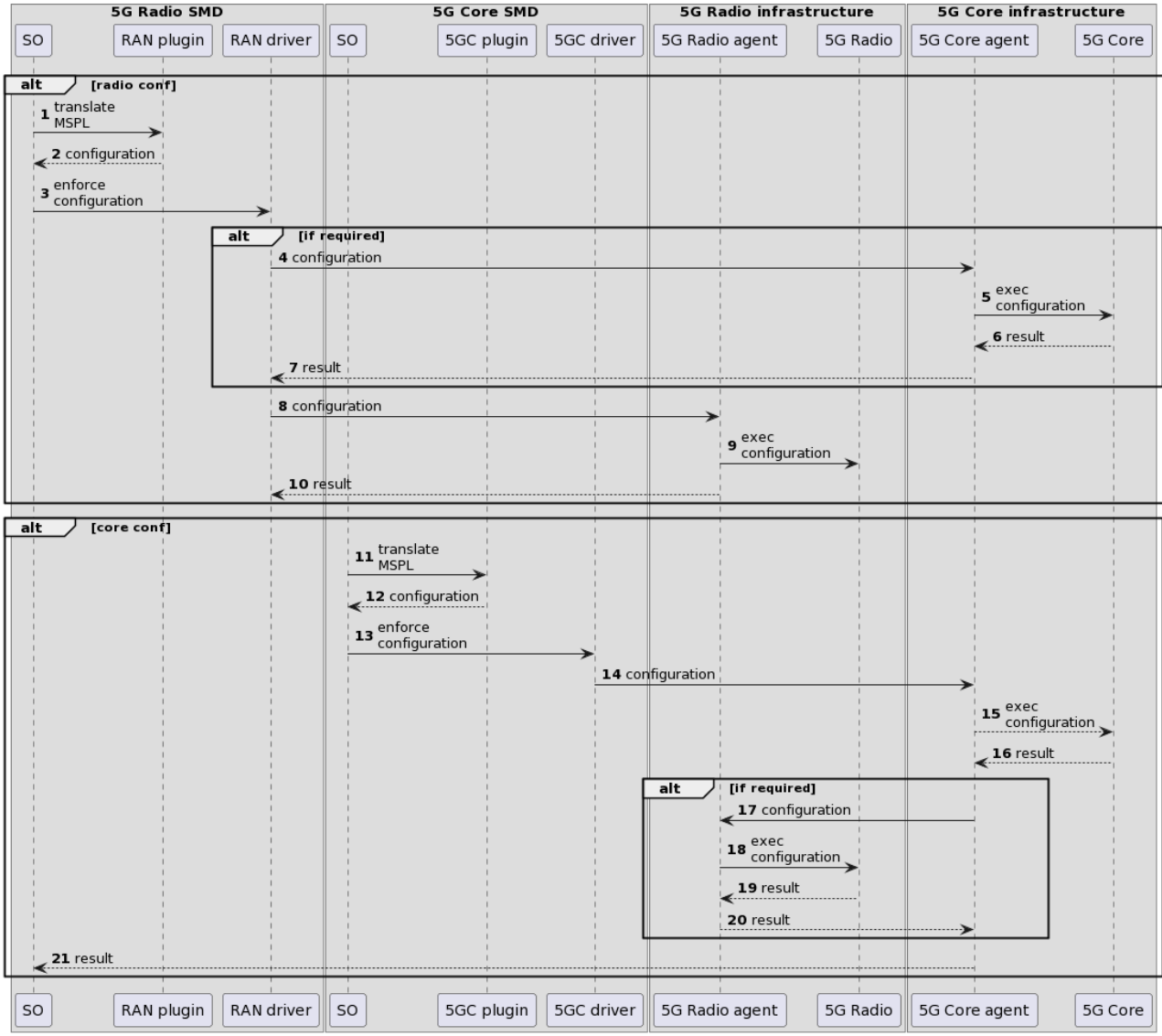

Figure 75 - Technical implementation sequence diagram

The Figure 75 depicts the sequence diagram that occurs for enforcing a security policy by using both Security Agents:

1. Given the case that we need to configure some RAN related capability (e.g., isolation of a user): the SO sends the corresponding received MSPL to the RAN plugin at the SO, that performs a translation procedure to translate the requirements specified in the MSPL, to a final configuration of the RAN.
2. The RAN plugin sends the configuration to the SO.
3. The SO starts the enforcement procedure, sending the configuration to the RAN drive
4. The RAN driver will then evaluate if further exchange of information is required with other 5G component (e.g. ,5G Core agent)
5. If some operation is required from the 5G Core agent, it will execute the operation in the 5G Core (e.g., get information about AMF)
6. The result is sent back to the 5G Core agent.
7. The 5G Core agent will forward the result to the 5G RAN driver.
8. This information could be used to influence the final configuration sent to the RAN agent.
9. The RAN agent finally enforces the configuration in the Amarisoft Radio Equipment.

10. The result is forwarded to the SO.
11. Given the case that we need to configure a 5G Core. First of all, the SO send to the 5G Core plugin the MSPL to perform a translation procedure.
12. The 5G Core plugin returns the final configuration to be enforced.
13. The SO sends to the 5G Core driver the enforcement operation to apply the 5G Core configuration in the 5G system.
14. The driver extracts the entry point of the 5G Core agent and send the configuration using json format.
15. The 5G Core agent execute the operation (deploy, redeploy, get_info, stop) in certain NFs of the 5G Core.
16. The result is sent back to the 5G Core agent
17. If required, the 5G Core agent can communicate to the RAN agent to take further actions to complete the enforcement of the policy (e.g., Register the AMF in the RAN)
18. The RAN agent applies the configuration on the RAN.
19. The result is sent back to the RAN.
20. The received result is forwarded to the 5G Core agent.
21. Finally, the result is forwarded to the SO.

2.21.5 Summary, lessons learned and guidelines

Guidelines:

We have used Security Agents as part of the enforcement process to perform fine-grained enforcement over 5G System, enabling the orchestration of configurations in 5G Components such as 5G gNB and the dockerized 5G Core thus using 5G native components as security assets through the use of Security Agents.

Lessons learned:

The 5GC open-source enable the distribution of the 5G NFs as the closed-source options that we have at our disposal are represented as a black box, not distributable. But on the other hand the open-sourced versions are few documented and it is required to in-deep study 5G to understand the functionality of the configuration files and configure them properly.

Future work:

The 5G Core Security Agent is planned to be evolved to keep covering more security procedures supported by the 5G standard, and to enable dynamic orchestration of security requirements for specific users/tenants.

The 5G RAN Security Agent as is a closed-source is planned to evolve the implemented driver to use the offered features by the security agent to enable the security through the Radio Access Network.

2.22 HLA roles summary

The following Table 2 displays a summary of the HLA roles defined in D2.2 versus the previously presented enablers. Some services are missing, for example, the root cause analysis services or the trust related (TM) services as they are covered by WP4 enablers. In this case, they are displayed in italics.

HLA Services	2.1 MOTDEC	2.2 SAF/ADF	2.3 MMT	2.4 V2X Mis. detect.	2.5 App layer DDoS	2.6 App layer DDpS	2.7 DDoS detect	2.8 Anti GPS	2.9 I2NSF	2.10 PyrDE	2.11 Systemic	2.12 E2E Sec ORch	2.13 Sec Orch	2.14 Discovery	2.15 SSLA Mgr	2.16 Sec Net Slci	2.17 Policy Mngr	2.18 SFSBroker	2.19 Katana	2.20 Integration Frab	2.21 Sec Agents
SDC Data Collection Service			X	X	X	X	X	X													
SAE Anomaly Detection Service	X	X		X	X	X	X	X			X										
SAE RooT Cause Analysis Service																					
DE Security Decision Service	X			X	X					X											
DE Security Decision Priority Service										X											
SO Security Policy Enforcement Service									X		X		X								
PSM HSPL Refinement Service																	X				
PSM MSPL/TOSCA Refinement Service																	X				
PSM Security Policy Storage Service																	X				
PSM SSLA Storage Service														X	X						
PSM Policy Conflict Detection Service																	X				
TM Trust Reputation Manager Service																					
TM Component Certification Service																					
TM Slice Trustworthiness Service																					
TM Ordered Proof of Transit Service																					
TM Service Trust Manager Service											X										
TM Wrapper Service																					
E2E_SAE Anomaly Detection Service							X														
E2E_SAE RooT Cause Analysis Service																					
E2E_DE Security Decision Synchro. Service										X											
E2E_DE Security Decision Service										X											
E2E_DE Security Decision Priority Service										X											
E2E_SO Sec. Policy Enforcement Srv.												X									
E2E_PSM Security SLA Refinement Service																					
E2E_PSM HSPL Refinement Service																	X				
E2E_PSM Security Policy Storage Service															X		X				
E2E_TM Collaborative E2E Network Slice Mgmt																					
E2E_PSM SSLA Storage Service																			X		
SMD Network Slicing Management																X					
E2E_SMD Network Slice Brokering																		X			
DomLvl/CrossDom Data Access Service																				X	
IF Registration Service																				X	
IF Discovery Service																				X	
IF Invocation Service																				X	
IF Communication Service																				X	
SA Security Policy Local Enforcement Service																					X

HLA Services	2.1 MOTDEC	2.2 SAF/ADF	2.3 MMT	2.4 V2X Mis. detect.	2.5 App layer DDoS	2.6 App layer DDpS	2.7 DDoS detect	2.8 Anti GPS	2.9 I2NSF	2.10 PyrDE	2.11 Systemic	2.12 E2E Sec ORch	2.13 Sec Orch	2.14 Discovery	2.15 SSLA Mgr	2.16 Sec Net Slci	2.17 Policy Mngr	2.18 SFSBroker	2.19 Katana	2.20 Integration Frab	2.21 Sec Agents
SA Network Monit. and Telemetry Service																					X
Unif_Sec_API Network Slice Tmpl. Actions														X		X					
Unif_Sec_API Net. Slice Instance Actions														X		X					

Table 2: Global coverage of the HLA by the enablers

3 Smart Closed Loops

The WP3 work package finishes by providing multiple enablers implementations (as described in the section 2). These enablers were used to create multiple small-scale closed-loops that manage the security of live running applications. The loops combine the WP3 enablers in a smaller scope compare to the final WP5 demonstration and sometime are bound to different components. This section describes those smart closed loops. They all follow the general closed-loop design described in the D2.2 deliverable. They all attest of the various ZSM management obtained within the WP3 work package.

The Figure 76 is a reminder of the OODA principles applied to the targeted INSPIRE-5Gplus reference architecture, where a control loop runs on top of the infrastructure. In the first Observe phase, the Security Data Collector gathers infrastructure data. In the second Orient phase, the Security Analytics Engine augments the data with extra information or correlations. Then in the third Decide phase, the Decision Engine finds the mitigation from the event relative and the current infrastructure state. In the next fourth Governance phase, the Policy and SSLA Manager adapts the security mitigation relative to the security policies rules. Then in fifth Act phase, the Security Orchestrator enforces the mitigation inside the infrastructure.

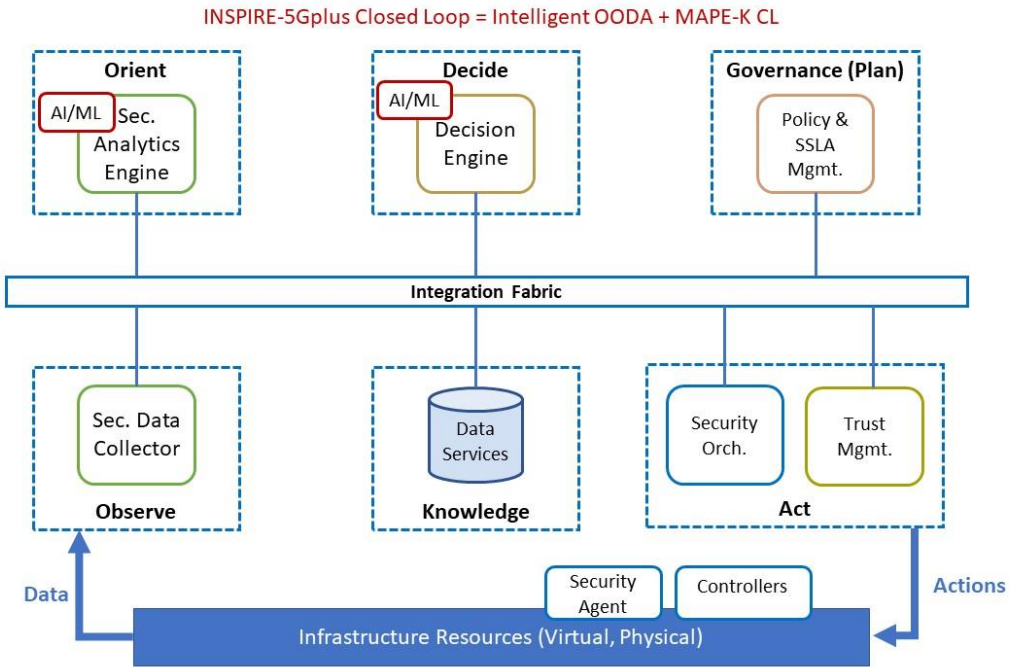

Figure 76 - INSPIRE-5Gplus closed-loop reference

3.1 Service protection in a single management domain

3.1.1 MTD Controller

3.1.1.1 Observe

As MOTDEC and its cognitive component (OptSFC) evaluate and enforce proactive and reactive MTD strategies, they collect various data for the evaluation of the attack surface and attack success probability of possible attacks (proactive case), and attack detection alerts from security agents and anomaly detection systems (reactive case).

For the proactive case, MOTDEC collects network performance measurements, consumption of computing resources, and vulnerability scans of the VNFs used in the infrastructure. Network measurements come from monitoring probes that can be installed directly in the VNFs or in the Virtual

Infrastructure Managers (VIMs), via port mirroring (or port forwarding). As specified in Section 2.1.3, the probe used in the implementation is the MMT from Montimage. The consumption of computing resources comes from the VIMs, such as Openstack, observing CPU cores, RAM, and storage consumption, keeping count of the remaining available resources. the Vulnerability scans are performed by MOTDEC and are implemented using the OpenVAS vulnerability scanner.

For the reactive case, MOTDEC collects attack alerts from the security agents Systemic and the SAF. Both are implemented and integrated with MOTDEC in the scope of Demonstration 3 as described in D5.2 in the INSPIRE-5Gplus. Systemic detects binary tampering of services deployed in VNFs, possibly disrupting or hijacking the normal functioning of the service. SAF/ADF instead detects anomalies that vary from the normal behaviour of the VNFs. Such anomalies can be generated by Advanced Persistent Threats (APTs), such as installed malware (e.g. spyware), C&C, or backdoors. Attack alerts are received together with information on the VNF target and the attack type.

3.1.1.2 Orient

In the proactive case, OptSFC evaluates the attack surface of each resource in the network by finding related vulnerabilities, listed as Common Vulnerability Enumeration (CVE) entries. CVEs found by the vulnerability scans are classified as vectors of different types of attacks, such as APTs, data leaks, and DoS attacks. This classification is done based on their mapping with Common Weakness Enumeration (CWE) entries and on keyword matching. The CVSS base scores and exploitability scores of such CVEs are then aggregated based on this classification. Finally, we estimate the reduction of the attack success probability (ASP) on the resources based on their attack surface and the estimated effect of each MTD action on each attack type (considering the frequency of the MTD action as a parameter).

For the evaluation of the operational cost of the MTD actions, we define Cost/hour = β + $\alpha 1$ cpu+ $\alpha 2$ ram_gb + $\alpha 3$ disk_gb, where we consider the additional resources used and time required for the enforcement of the action. We made an empirical analysis on the prices of cloud resources to find such coefficients by using the May 2022 catalogues of 4 major cloud providers: Amazon AWS, Microsoft Azure, Google Cloud, and OVH Cloud (covering around 70% of the cloud market in the first quarter of 2022). In the reactive case, attack alerts are used as they are.

3.1.1.3 Decide

In the proactive case, these data and their augmented evaluations are used for the multi-objective deep RL training, which identify 3 objectives: operational cost of MTD operations using the cost/hour formula, security improvement by reducing the ASP and increasing the attack surface shifts, and the network overhead by keeping network performances stable. the RL agent is implemented in a continuous learning setup as it updates the ML model periodically based on the MTD operations that has been decided and the measurements received from the environment.

In the reactive case, OptSFC can rely in 2 mitigation policies: learned mitigation policy based on the specific attack detected, and manually defined mitigation policy that the network administrator gives to MOTDEC. The learned mitigation policy is part of the autonomous RL optimization policy, where the agent learns the optimal mitigation policy based on whether the attack stopped its effect or not, which will lead the security agents to stop sending attack alerts. The manually defined mitigation policy can be used when the best mitigation to an attack is known by the network administrator and forces OptSFC to use it instead.

3.1.1.4 Act

After the decision of OptSFC on the MTD action to be performed, MOTDEC enforces the action coordinating operations imparted to the Katana network slice manager and the OSM NFVO.

3.1.2 V2X misbehaviour detection

3.1.2.1 Observe

During this phase, the aggregated BSMs exchanged among vehicles are inspected for potential existence of DDoS traces which are transmitted at a frequency higher than the limit set by the standard. The aggregated BSMs constitute a time-series repository evolving over time, and the inter-arrival time of anomalous information is lower for DDoS attack messages. DDoS can take various forms such as DDoS random, DDoS disruptive, DDoS random sybil, and DDoS disruptive sybil attack variants.

3.1.2.2 Orient

The aggregated BSMs constitute a time-series repository evolving over time, and the inter-arrival time of anomalous information is lower for DDoS attack messages. The IP address of each vehicle is also embedded in the BSMs in order to assist the network filtering asset for the service remediation.

3.1.2.3 Decide

Security analytics engine sequentially analyses the incoming streaming BSMs based on mobility patterns, to instruct the RL algorithm (V2X-DDoS detector) for the detection of DDoS attacks. The decision refers to the identification of either genuine or malicious vehicular behaviour at each time-step.

3.1.2.4 Act

The decision engine, upon detection of DDoS traces, provides the verdict to the security orchestrator to apply a pre-determined security policy, e.g., malicious traffic to be isolated, dropped, or blocked, through proper IP filtering at application layer.

3.1.3 Anti-GPS spoofing

3.1.3.1 Observe

In this stage, the Location Monitoring module captures in real-time the UAV's GPS positions from the UAV flight telemetry data and their corresponding path losses reported by the base stations, and computes the theoretical path losses based on the base station locations and UAV's GPS locations. The collected data are exported to the Spoofing Detector for uncovering spoofed GPS positions.

3.1.3.2 Orient

In this stage, the Spoofing Detector analyses the data received from the Location Monitoring to identify spoofed positions. The analysis is performed leveraging different deep learning techniques; specifically, MLP and CNN models. A notification is sent to the Decision Engine if a GPS spoofing is detected.

3.1.3.3 Decide

Receiving the GPS spoofing notification, the Decision Engine can decide on the action to perform such as sending the landing command to the UAV ground control system and asking the MANO to release resources allocated to UAV.

3.1.3.4 Act

The Ground control system (deployed as a service at the edge[51]) sends a control command to UAV for safely landing it. Once done, the MANO can proceed to free the resources reserved for the UAV; for example, the network slices created for the UAV.

3.2 System protection

3.2.1 DDoS detection and mitigation inside the application layer

3.2.1.1 Observe

In this stage, the Monitoring System continuously collects the resource usage and performance metrics values observed in the last x time steps. The data are collected from the slice's VNFs and their hosting computing nodes via deployed probes.

3.2.1.2 Orient

In this stage, the DDoS Mitigator analyses the resource usage and performance metrics collected by the Monitoring System over the time to assesses whether a scaling-out/up request is legitimate or caused by an application-layer DDoS attack. The analysis is performed using a LSTM-based AutoEncoder model trained to reconstruct multivariate time-series for normal behaviour. The reconstruction error is calculated for each time step in the analysed time series.

3.2.1.3 Decide

Using a preset anomaly threshold, the DDoS Mitigator can decide if the values of the resource usage and performance metrics are anomalous. When an anomaly is detected, the DDoS Mitigator notifies the Admission Controller Delegator to refuse the scaling-out/up request expressed by the auto-scaling module.

3.2.1.4 Act

The Admission Controller Delegator performs the appropriate actions to stop the scaling-out/up operation.

3.2.2 DDoS detection in multi-tenant/domain environment

Figure 77 displays the OODA cycle of the DDoS detection. The steps are explained in the following

[51] O. Bekkouche, T. Taleb and M. Bagaa, "UAVs Traffic Control Based on Multi-Access Edge Computing," 2018 IEEE Global Communications Conference (GLOBECOM), 2018, pp. 1-6, doi: 10.1109/GLOCOM.2018.8647421.

subsections.

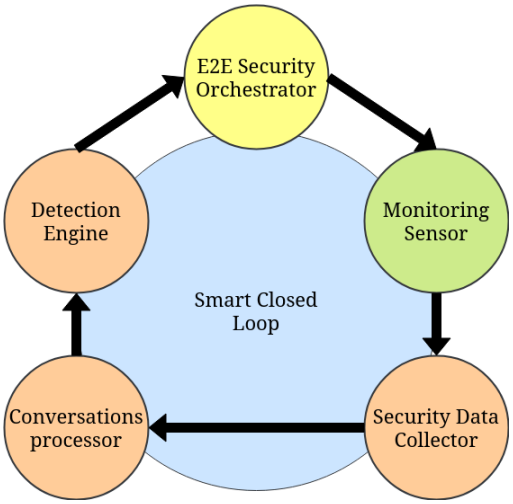

Figure 77: Smart loop for DDoS detection

3.2.2.1 Configure

As a first step, the E2E Security Orchestrator configures each one of the modules based on a list of parameters. For instance, traffic filters in the Monitoring Sensor, conversation timeouts in the Conversations Processor, or ML parameters such as number of epochs to be executed in the Detection Engine.

3.2.2.2 Observe

In this stage, traffic is captured in real-time by the Monitoring Sensor. For each packet, relevant fields are extracted and a summary of each one is sent to the Security Data Collector, which will forward the information to the Conversations Processor.

3.2.2.3 Orient

At the conversations processor, packets information is grouped into conversations, and a list of features for each one is generated and sent to the Detection Engine.

3.2.2.4 Decide

The Detection Engine, based on the combination of two unsupervised ML techniques, decides whether a conversation represents a DoS attack or not. The conclusion for each conversation is sent back to the E2E Security Orchestrator, where the process began.

3.2.2.5 Act

Once the E2E Security Orchestrator receives the list of conversations provided with a normal or attack traffic label for each one of them, it enforces some policies to respond against the threat. Some feasible countermeasures could be blocking traffic from attacker's IP or redirecting it to a honeynet for further investigation.

3.2.3 Software protection

3.2.3.1 General

Systemic software protection is a SECaaS service offering software security, hardening un-protected executables into hardened variants and enabling the runtime monitoring of the deployed protected instances. The protection and monitoring services enable to insert Systemic in a closed loop layout as

a security breach detected in monitoring will trigger either the deployment of the original version or a change of the security policy, during the execution and without requiring the re-installing and launch of a new variant. The automated protection regulation at runtime is the next research step requiring the collecting and processing of granular runtime metrics during the execution of the running program. These metrics are time and frequency measures related to the different executed code blocks of the running program. Their collection will be enriched along several runs at one location or at different locations to elaborate the execution profile (i.e., Control Flow Graph profile) which corresponds to a normal software execution, enabling to discern deviations signalizing potential attacks.

The close-loop arrangement corresponds to the central monitoring which enables to modify the security policy according to transmitted signed heartbeats as shown in Figure 78.

The central monitoring is able to collect and process heartbeats originated from several deployed protected software, elaborating an integrated averaged CFG profile. This profile enables to spread the protection costs at the right locations, avoiding to create overhead surge at highly over-loaded segments of the software. It also enables to discern attacks on the control graph (e.g., code injection, ROP type of attack) and the elaboration of the most adapted reaction.

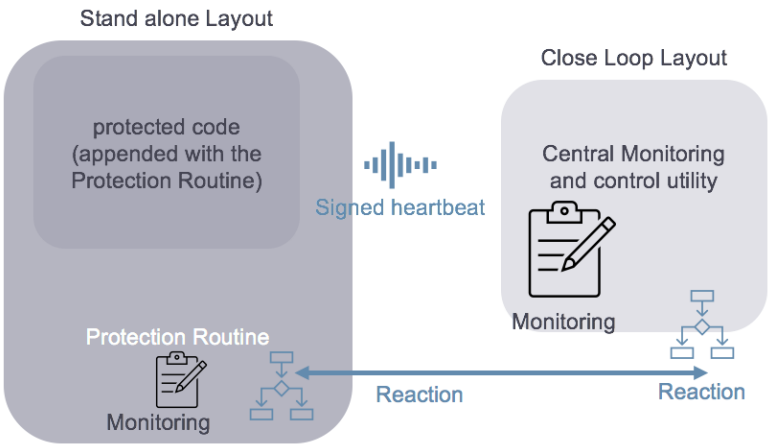

Figure 78: System protection layout

The OODA cycle is given below.

3.2.3.2 Observe

In this stage, the security routine appended into the protected software detects integrity failures and deviations from the normal CFG. These elements are collected locally by the protected program itself but can also be transmitted in the form of signed heartbeats to a central monitoring utility.

3.2.3.3 Orient

As a refinement, the local or reversely centralized processing of the decision-taking can be oriented by a level of severity of the attacks and be threat context dependent. If an attack is recurrent and has been detected on other deployments, a modification of the decision-taking and associated reaction can be forwarded to the monitored software.

3.2.3.4 Decide

As stated above, a centralized or individually and locally processed decision-taking and reaction can be designed. Local and individual processing of these collected data is simpler to deploy, without a communication link with a remote utility (with its potential security weaknesses) but comes with an automatized processing of detected failures, leaving no to low room for elaborating discrete reactions. The best cyber-defence is the one executed covertly and un-detected by the attacker. A decision taking from a remote location which ranges over a different perimeter and at a deferred timing does not

bring the information to the attacker that its attack has been detected.

3.2.3.5 Act

Two types of reaction can be worked out: the tear down of the tampered executable or a change of the security policy (e.g., elevation of the frequency of measurement, transfer of more code section from an exposed execution environment to a more trusted execution environment). The second alternative does not interrupt the service offered by the software, keeping its running execution seamlessly. The second alternative modifies the security pattern applied on the software, ideally with fine-grained adjustment.

3.3 Multi-domain

3.3.1 Robust-fed learning

The proposed two-tier anomaly detector is made up of two sets of detectors for each management domain. Every anomaly detector analyses network flows and discards those that are identified as anomalous. The proposed detectors are dispersed throughout the network. Each detector has a database that stores the captured network flows. This information is then used to train the models in each detector. The algorithm for anomaly detection has been formed using a federated learning-based mechanism.

FL is itself insecure because the centralized server can be poisoned if there is at least one adversary who wants to manipulate the learning process. A typical adversarial attempt to poison the FL process is illustrated in Figure 79 below. Considering the nature of the false information, these attacks are called either data poisoning attacks or model poisoning attacks. Since the centralized server cannot guarantee that the distributed entities provide accurate local models, defending mechanisms are also needed in FL. As the centralized server has little or no control over the level of security of each distributed node, the centralized server must impose security mechanisms centrally to distinguish poisonous and honest model updates. And then use only the honest updates to continue with the learning process. This task is challenging as the centralized server does not possess any validation data for verification.

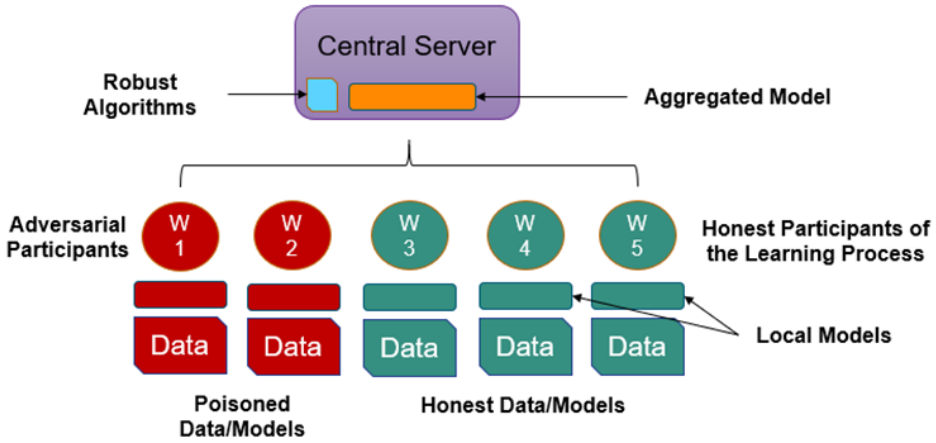

Figure 79: Poisoning attacks on FL systems

Hence, sophisticated algorithms should be designed that can impose strict security at the centralized server to make sure that the FL environments is safe. The adversaries may have a common goal, and their updates are similar in nature and their objective is to deviate the learning process towards adversarial objective. Therefore, further improvement of the defence mechanisms at the centralized server is needed to increase robustness and resiliency.

The existing defence strategies are still vulnerable against coordinated and intelligent attacks. A coordinated attack means, the adversary has control over multiple nodes that poison the system and

the adversary manipulates these nodes to work together to perform an attack that is less detectable. Generally, the robust algorithms consider two properties of the model/gradient updates received from the distributed nodes. Those two are the magnitude and the direction of the updates. Simultaneously, two different approaches have been used to identify these adversaries based on the received updates, i.e. euclidean distance based detection mechanisms and angular based detection mechanisms. So far, the angular based detection mechanisms yield better results empirically. This is because in poisoning attacks, the adversary controls a set of nodes to manipulate the model to update along a desired direction rather than the true objective of the learning problem. The existing defence mechanisms are vulnerable against the noisy adversaries in a following manner.

Since the algorithm uses the similarity index to rule out the adversary-controlled nodes from the system, the coordinated adversaries are capable of introducing noisy updates to break the similarity of the updates and remain undetected. To tackle this problem, the defence mechanisms should not merely consider the angle of the updates but also a mix of angle and the magnitude of the updates. Moreover, rather than analysing the angle and the magnitude of the model update, the defence mechanisms should consider the model parameters and their behaviour throughout the learning process to identify and eliminate the adversarial models from the system. The two-stage anomaly detector proposed for Inspire 5G+ architecture using federated learning is vulnerable for these type of poisoning attacks. The nodes in a given domain may provide poisonous training data to deteriorate the accuracy of the trained model and finally causing incorrect predictions in anomaly detection.

3.3.2 Reactive security protection

Outside of the final integration, the Pyr Decision Engine is deployed in a local scope to create a small reactive loop for security protection.

3.3.2.1 Observe

The Pyr Decision Engine focuses on the reactive case. It provides reactive mitigations for protecting running services. It delegates the observing phase to other components for threats appraisal. In this context, some probes are monitoring the infrastructure and the services. In the current local deployment, the probes survey the flow in front of potential targets. The targets are Docker containers acting as VNFs. The probe is transparent for the application. The observation can be pushed directly to the Pyr DE or to an intermediate fabric (Kafka) or to an Analytic Engine (such as ElasticSearch).

3.3.2.2 Orient

After capturing packets, the flow is assessed against known threat. For example, in the current deployment, the target is a SSH server under a password brute force attack. In the current deployment, to avoid sending header around during this orient phase, the analysis is done on the probe.

The anomalous flows are detected by computing a signature hash on the packet five tuple and comparing the signature to a pre-computed set of malicious signatures.

3.3.2.3 Decide

In the covered reactive phase, the Pyr DE create quick mitigations using a rule-based approach. For a given event, an adequate reaction is selected.

3.3.2.4 Act

The Pyr DE, at this final phase, can emit a MSPL manifest to configure the infrastructure. However, in the current local deployment, the Pyr DE can also emits some OpenFlow rules to OVS switches using a small intent REST API. The rules creation follows pre-filled templates. The fact that reactions are Docker containers, separates from the core of the Pyr DE harness, allows for flexible reaction creation.

4 Closed loop & E2E coordination

This section describes the coordination that happens at the core of the INSPIRE-5Gplus Smart closed-loops. In the multi-domains context targeted by the INSPIRE-5Gplus project some issues can appears. The closed loops need to deal with the enforcement of security policies in multiple domains with various capabilities. A conflict can arise when applying a policy or a decision. The local decision from an SMD can be escalated.

4.1 Hierarchical organization & Conflict management

4.1.1 Decision delegation/escalation

The local SMD Decision Engine may forward the alerts to the E2E level. It does that while running its local closed loop. It provides an autonomous local ZSM closed loop in the domain that enforces fast reactions. In an escalation context, the forwarded event can trigger in the E2E Decision Engine more mitigations. For example, a local mitigation stopping a brute force attack can be broadcasted to the others domains. Sometimes, the local SMD closed loop can ignore an event due to a missing reaction. In this context, SMD Decision Engine cannot take any mitigation and by forwarding event to the E2E level, it delegates the decision.

4.1.2 Policies hierarchical enforcement for E2E security

The E2E Decision Engine delegates the global enforcement to the global Security Orchestrator residing at the top level. The E2E SO manages the technical details on how to split and forward the global policies downward. The E2E Security Orchestrator verifies the capabilities of a subordinate domain before translating the policies to the required format.

4.2 Conflict Management

4.2.1 Policies conflict detection

The conflict detector module belongs to the Policy Framework. This module is in charge of maintaining policy consistency during the closed management loop through a system of rules provided by Pyke engine. Specifically, the conflict detector is a step during the orchestration process, prior to policy enforcement. The module receives a MSPL-OP (e.g., Filtering, Channel Protection, ...) and evaluates all the parameters included in the MSPL and try to match with the corresponding the rules defined for them in the rule engine. This process takes as the input the MSPL-OP and check that the parameters can be applied to the specific policy type, but also it will review that the new policy doesn't conflict any already enforced policies. The policies maintain a status, the new ones are incorporated as "pending" and those that are retrieved from the policy repository, if they are still applied, their status will also indicate it. First those that were already applied are added to the rules system, and then the MSPLs belonging to the MSPL-OP will be added. Thus, each "pending" MSPL will be passed through the rule engine matched against already enforced policies and against the other MSPLs of the MSPL-OP. During this process, for example, it is checked that the policy has not already been applied or that the action does not contradict another already applied (e.g., ALLOW and DENY). After this process called Inference, new conclusions can be added to the knowledge base, and can be further used to detect new conflicts.

Through the knowledge base facts, we can add objects with different characteristics necessary to correctly perform the conflict detection, for example: we add to the fact base the different capabilities that a given domain has, focusing on the E2E perspective, the conflict detector performs a search for inter-domain conflicts, such as the possibility that between two domains that are going to establish a channel protection using the same key and protocol parameters (IPsec/DTLS). To do this, the

capabilities of each domain are added to the fact base, and when the Channel Protection policy is received, it is verified that each domain involved in the MSPL has the necessary capabilities to apply the policy. In case the Conflict Detector detects a conflict, an alert is triggered indicating the risk and the orchestration process is stopped so that the Decision Engine can take appropriate action. Below the rules to perform the E2E Channel Protection conflict detection can be found

MSPL(?m1)∧(hasnot(m1.dest.domain1, m1.capability) ||

hasnot(m1.dest.domain, m1.capability))

→ ”Capability in Domain Missing(?m1)”

The Conflict Detection module allows to ensure the robustness of the system while maintaining and benefiting from the flexibility of the policies since it can be extended to cover new security assets or domains, or even adapt it to new conflicts discovered.

4.2.2 Decision conflict

Inside a closed-loop, a reaction conflict is possible. First inside a sole SMD domain, some mitigation pipeline can run concurrently, each targeting a specific threat. They may emit a reactive decision on the same resource with contradictory side effects. The Decision Engine coordinates this conflict by using a priority system to order the mitigation. It also applied a grace period to provide stability in the infrastructure by avoiding too much fluctuation.

4.3 E2E Security Policies Management

Policy management at the E2E level is part of the closed management loop that has two possible entry points on the E2E SO, the closed policy management loop maintains the objective of ensuring compliance with the SSLAs on 5G services contracted with a customer through the security management of 5G network slices. The first entry point, the proactive one, serves to deploy the 5G service and security services that ensure compliance with the security requirements set forth in the SSLAs, while the other entry point, the reactive one, serves to generate countermeasures that maintain SSLA compliance when a change in context (e.g., an attack) compromises them. The **proactive** part receives the policy from the E2E Slice Manager, this policy is an Orchestration Policy that contains in MSPL-OP language the translation of the SSLA: a 5G service that must be deployed and some security requirements that must be associated to that 5G service, all this as part of an E2E 5G Security Slice. When the E2E SO, receives this policy, it first examines it to detect that it is a 5G security slice (remember that the E2E SO is prepared to receive also another type capabilities or languages, which would make the flow go differently, but INSPIRE-5GPLUS focuses on the deployment of the 5g security slice), this makes the E2E SO build a policy per domain, where each policy is considered a sub-slice, this sub-slice contains: the service in case this domain is the target of the service, and the security requirements to be deployed in the domain. In other words, for each domain, the E2E SO builds an orchestration policy that accumulates the necessary requirements to be deployed so that the security requirements established in the SSLA are fully met. The construction of each orchestration policy per domain is called "enforcement plan", each policy has an associated priority level that allows the orchestrator to dictate in which order the policies should be applied, as each orchestration policy of the enforcement plan may contain multiple policies, the highest priority of the set of policies belonging to that orchestration policy will be applied, so the enforcement plan is sorted by domain priority. The **reactive** part refers to the entry point from the E2E DE, triggered as part of a reactive process given from the SAE when an anomaly is detected. The main difference of the reactive orchestration policies generated by the E2E DE is that they do not have to contain inside as a capability the 5g_security_slice, in which case the flow varies since the enforcement plan does not generate a single orchestration policy per domain but generates several orchestration policies per domain and they are applied (i.e. sent to the SMD SO) depending on their individual priority.

5 ETSI ZSM closed loops support and coverage

The ETSI, in the "Closed-Loop Automation; Part 1: Enablers"[52] document, defines two types of closed loops:

- Made-to-order closed loops (M2O-CL) which are built and deployed on demand by the ZSM framework.
- Ready-made closed loop (RM-CL) which are created by the ZSM vendors without any further dynamism.

In such context, the ZSM loops presented in this report are inside the Ready-made loop, as each component was selected and assembled around a main enabler. A subset of the M2O-CL is also supported as some components of a closed loop are dynamically deployed (for example the monitoring probes). The commissioning phase is prepared statically and the operational phase is static. Moreover, in term of governance, the INSPIRE-5Gplus does not integrate a global management life cycle of the closed loop or their models. The closed-loops of the INSPIRE-5Gplus project focus on the lifecycle of the managed entities (network slices, security policies, ...). The ETSI defines the ability of the ZSM framework to monitor KPIs and adapt the closed loops and their models to optimise the closed-loops set goals. In this regard, on the two basic type of policies, the INSPIRE-5Gplus framework uses the service policy type to control the security around a running service based on the externally observable behaviour: attack pattern or security warning. The resource policy type optimizing the delivery of constraints around a service is used during on-boarding with the refinement of SSLAs into lower level MSPLs manifests.

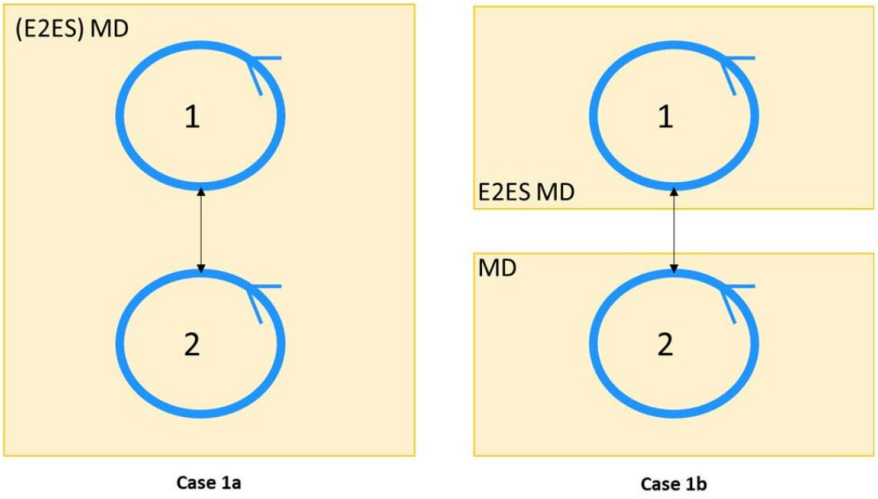

Figure 80: ETSI Hierarchical Closed Loops

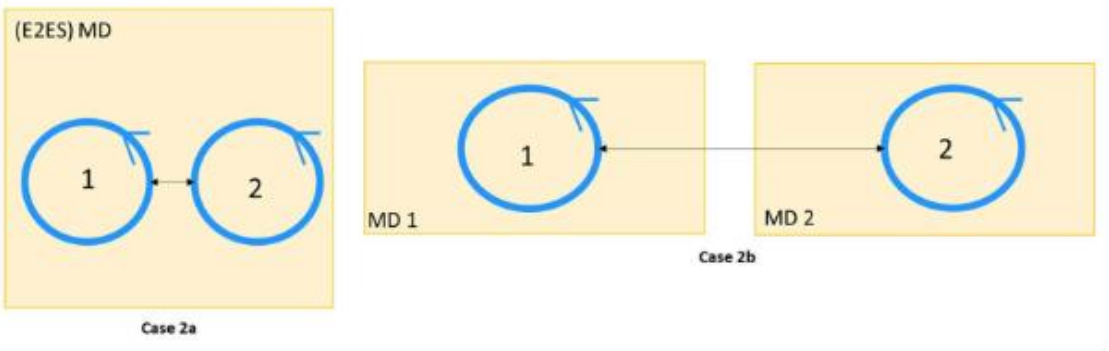

Figure 81: ETSI Peer Closed Loops

[52] https://www.etsi.org/deliver/etsi_gs/ZSM/001_099/00901/01.01.01_60/gs_ZSM00901v010101p.pdf

The ETSI introduces two types of coordination between multiple concurrent running closed-loops that defines the delegation and escalation done between them. Moreover, these types can also be classified based on the domain where the closed-loop are deployed. The hierarchical model, shown in the Figure 80, where a top-level closed-loop manages a set of underlying closed-loop, running either in the same domain or in a separate domain. In the peer model (see Figure 81), each closed-loop is independent but can influence each other behaviour with again a flexibility in their domain deployment. The ETSI refines the hierarchical model as a mode where the subordinate closed-loops are managing the local domain targeting a self-healing goal and local optimum. Whereas the top-level domain focuses on a global or end-to-end optimum. In this scenario, the subordinate might take conflicting decision, which the top-level can coordinate and orchestrate. The coordination can involve:

- Delegation: where the top-level pushes global order down to the subordinates closed-loop.
- Escalation; where the subordinate close-loops inform the top-level loop of local events.

In the INSPIRE-5Gplus project, the project ZSM closed-loop framework is of a hierarchical type. Each local Security Management Domain (SMD) contains a closed-loop. This loop manages the local security deployment. Then it escalates the local decision to the top-level End-to-End (E2E) domain for further interactions. Yet the final coordination is hybrid as, in the E2E level, the side-effect is managed by a global Security Orchestrator which crafts and delegates the policies down to each SMD Security Orchestrator.

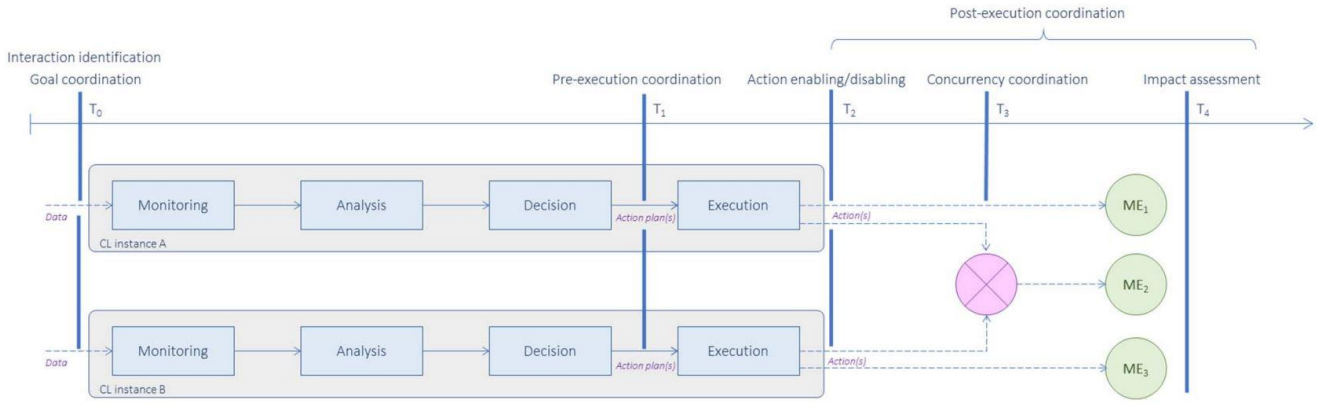

Figure 82: ETSI exemplary Closed Loop Coordination timeline

The Figure 82 displays an example timeline of some coordination between two closed-loops where multiples Managed Entities (ME) are manipulated by concurrent closed-loop. Each closed-loop has its computing model ingesting data from monitoring probes, analysing it, taking a decision based on it and executing it. With that, some conflicts might appear in the executed actions. The INSPIRES-5Gplus ZSM framework follows the same pattern with equivalent OODA loops (described in this document). As for the conflicts, the hierarchical model has some priority level parameters and the Security Orchestrator has a conflict detection component for the security policies. As for the impact assessment, as the project focuses on security, the goal is achieved when the originating probe acknowledges the disappearance of the threat.

The ETSI also introduces some services for the ZSM closed-loop governance. The INSPIRE-5Gplus framework covers a subset. For example, in the Closed Loop Governance (CLG) service, the “Escalate issue” capability is covered by the local Decision Engine to the E2E level. The “Manage Closed Loop goal” covered by the local Security Orchestrator which handles the local security goal achievement. In the Pre-execution coordination service, the “Provide notifications of conflicting action plans” capability is also done by the local Security Orchestrator within the scope of the security policies manifests.

Finally, the ETSI, in the “Means of Automation”[53] document, introduces the concept of entities governance trough policies. Whereas, in the ETSI, those policies can cover multiple aspect from QoS

[53] https://www.etsi.org/deliver/etsi_gr/ZSM/001_099/005/01.01.01_60/gr_zsm005v010101p.pdf

to energy, the INSPIRE-5Gplus targets the management of security. For the ETSI, The Policy Continuum preserves the policy relationships in each level of abstraction. The policies are translated between each level with a set of potential dependencies between them.

Figure 83: ETSI Schematic representation of policy levels

Figure 83 shows those the possible policy level from the ETSI point of view:

- The Business Level Policies: correspond to the "Market, product & customer" layer with global and generic constraints. This level is the SSLAs layer in the INSPIRE-5Gplus framework.
- The Service level and AF level: is the intermediate level where generic resources are configured. This level is the MSPLs layer in the INSPIRE-5Gplus framework.
- The AFpolicies level: is where the entities are manipulated and the policies enforced. In the INSPIRE-5Gplus framework, it translates to the infrastructure management component (for example OSM).

With that report, the INSPIRE-5Gplus project approach was successful to build several “specialized” ZSM closed-loop. Each one was described in this document. The pay-off was a local management of a security aspect (relative to each enabler / partner). The next step is for that framework to be tested in the WP5 demonstration where a subset of enablers is combined to create an overall ZSM framework across multiple domains.

6 Conclusions

This document is the last deliverable of WP3. It describes the enablers of the INSPIRE-5GPlus smart security management framework. These enablers implement the most important HLA capabilities starting from intelligent data collection and analysis and accurate runtime monitoring of virtualized 5G environments to cognitive and efficient prevention, detection and mitigation of security in multi-tenant, multi-domain network infrastructures. The document details a set of ZSM closed loops based on those enablers, analyses the level of their coverage to the ETSI ZSM specification and provides solutions to manage conflicts and priorities when the closed loops are deployed in a multi-domain context. A set of lessons learned, guidelines, and future work has been included for each enabler of the framework.

In the upcoming INSPIRE-5Gplus WP5 activities, these enablers will be integrated into the demonstrators 1 and 3 to illustrate various reactive and proactive closed loops based on the smart security framework resulting from WP3 and to work on its integration and performance measurements.