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

D5.3: Complete 5G security testing infrastructure implementation and final results

Version: v1.0

Deliverable type	R (Document, report)
Dissemination level	Public
Due date	31/10/2022
Submission date	02/12/2022
Lead editor	Maria Christopoulou (NCSRD)
Authors	Pol Alemany, Raul Muñoz, Charalampos Kalalas, Roshan Sedar, Ricard Vilalta (CTTC), Vinh La (MI), Edgardo Montes de Oca, Huu-Nghia Nguyen (MI), Rafal Artych (OPL), Yannick Carlinet, Chrystel Gaber, Yacine Anser, Nancy Perrot, Jean-Phillipe Wary (ORA), Ghada Arfaoui (ORA), Cyril Dangerville, Geoffroy Chollon (TSG), Alejandro Molina Zarca, Rodrigo Asensio (UMU), Klaas-Pieter Vlieg (EURES), Hugo Ramón, Antonio Pastor, Diego Lopez (TID), Wissem Soussi (ZHAW), Gürkan Gür (ZHAW), Vincent Lefebvre (TAGES), Gianni Santinelli (TAGES), Themistoklis Anagnostopoulos, George Xilouris, Dimitris Santorinaios (NCSRD), Anastasios Kourtis (NCSRD), Tharaka Mawane Hewa, Pawani Porambage (UOULU)
Reviewers	Chafika Benzaid (UOULU), Orestis Mavropoulos (CLS), Dhouha Ayed (TSG)
Work package, Task	WP5, T5.3
Keywords	Key Performance Indicators, Test Cases, Security Metrics

Abstract

This deliverable presents the final results of Key Performance Indicators (KPIs) measured in the three INSPIRE-5Gplus demonstrators. In each demo, we detail the storyline, the INSPIRE-5Gplus High-Level Architecture (HLA) mapping, and the 5G security testing infrastructure environment. Finally, we extract the lessons learned from each demonstrator's results to advance security in Beyond 5G networks.

Document revision history

Version	Date	Description of change	List of contributor(s)
v0.1	11/07/2022	Initial release	Maria Christopoulou
v0.2	30/09/2022	Contributions from partners, version sent for review	All authors
v0.3	7/10/2022	Partners address reviewers' comments	All authors
v0.4	26/10/2022	New version for editing based on TM's comments	All authors
v0.8	23/11/2022	Final release	All authors
v0.9	24/11/2022	Final editing	E. Tallås, Eures
1.0	02/12/2022	Final version submitted	U. Herzog, Eures

List of contributing partners, per section

Section number	Short name of partner organisations contributing
Section 1	NCSRD
Section 2	CTTC, UMU, TID
Section 3	UMU, CTTC, TSG, TID, UOULU, MI
Section 4	MI, ORA, OPL
Section 5	ZHAW, NCSRD, MI, TAGES
Section 6	UMU, CTTC, TSG, TID, MI, ORA, OPL, ZHAW, NCSRD, MI, TAGES
Section 7	NCSRD

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.

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

Executive Summary

This deliverable describes the results of the experimentation trials in the final year of INSPIRE-5Gplus. INSPIRE-5Gplus has produced three Demonstrators as representative cases of Beyond 5G scenarios. These demonstrators include security enablers developed during INSPIRE-5Gplus and follow the requirements and services of the INSPIRE-5Gplus High-Level Architecture (HLA).

The three Demonstrators are:

1. <u>Demo 1 "Security Management Closed Loop"</u>

The objective of Demo 1 is the instantiation of the INSPIRE-5Gplus HLA and the INSPIRE-5Gplus closed-loop security management across multiple domains. The demo storyline is based on providing a 5G related service protection scenario facing different types of customers' needs over the *INSPIRE-5Gplus* HLA common framework. The demo instantiates two network slices with specific Security Service Level Agreements (SSLAs) from a 5G network customer. The SSLAs instantiation leads to the connection of slices with services and associated security; and reactively mitigates attacks (*i.e.,* DDoS attacks and cryptomining activities) on the deployed services to maintain the imposed SSLAs.

2. <u>Demo 2 "Trust and Liability Management"</u>

Demo 2 investigates the concepts of trust and liability management on a virtualized infrastructure for a 5G type ecosystem. The demo illustrates vertical SSLA deployment through a specific commitment of vertical service isolation over the proposed infrastructure. Critical services, subject to the Network and Information Security (NIS) directive, demonstrate their fulfillment to slice isolation (*i.e.,* a legal obligation). The objectives of Demo 2 are: *i)* to propose a way to manage and serve those constraints dynamically, and *ii)* to evaluate the SSLA operations on specific infrastructure components.

3. <u>Demo 3 "Moving Target Defense"</u>

The objective of this Demonstrator case is the evaluation of Moving Target Defence (MTD) as an effective mechanism in improving the network's resilience against attacks, by effectively protecting network slices through dynamic reconfiguration of 5G infrastructure properties. The focus of this demonstration is the proactive change of the underlying network configuration to alter the attack surface and impede pre-attack reconnaissance advantages of attackers prior to the attack stage.

These Demonstrators build upon the implementation of components developed or upgraded during INSPIRE-5Gplus and their objectives include: *i)* the rigorous validation of the Project's proposed High-Level Architecture (HLA), and *ii)* the feasibility evaluation of enablers against specific KPIs in real-world scenarios.

Results showcase the feasibility of new technologies and concepts (e.g., Artificial Intelligence, Moving Target Defense, Blockchain, Trust and Liability) in protecting the services of Beyond 5G (B5G) networks. Each Demonstrator also extracts the lessons learned to pave the way for their proper application on future networks.

Table of Contents

List of Figures

List of Tables

Abbreviations

5G-PPP	5G Infrastructure Public Private Partnership
AAA	Authentication, Authorization, Accounting
AAE	Adversarial Autoencoder
AMF	Access and Mobility Function
DDoS	Distributed Denial of Service
DTLS	Datagram Transport Layer Security
E2E	End-to-end
HLA	High Level Architecture
I2NSF	Interface to Network Security Functions
IAM	Identity & Access Management
IoT	Internet of Things
IUC	Illustrative Use Cases
LXD	Linux Container Hypervisor
MANO	Management and Orchestration
MEC	Mobile Edge Computing
MMT	Montimage Monitoring Tool Framework
MS<number>	Milestone <number>
MSPL	My Simple Protocol Language
MTD	Moving Target Defence
NS	Network Service
PCRF	Policy and Charging Rules Function
PoT	Proof of Transit
RAN	Radio Access Network
RCA	Root Cause Analysis
SMD	Security Management Domain
SMF	Session Management Function
SECaaS	Security as a Service
SSLA	Security Service Level Agreement
STA	Smart Traffic Analyzer
TSLA	Trust Service Level Agreement
UDM	Unified Data Management
UDR	Unified Data Repository
UE	User Equipment
V2X	Vehicle-to-everything
VCP	Virtual Channel Protection enabler
VM	Virtual Machine
VNF	Virtual Network Function

1 Introduction

1.1 Scope

This deliverable presents the three Demonstrators that were developed during INSPIRE-5Gplus and the results of Key Performance Indicators (KPIs) measured and validated in each one. We detail the storyline, the INSPIRE-5Gplus High-Level Architecture (HLA) mapping, and the 5G security testing infrastructure environment. We also provide reasoning on how each demo validates the INSPIRE-5Gplus HLA. Finally, we extract the lessons learned from each demonstrator's results to advance security in Beyond 5G networks.

1.2 Target Audience

The target audience of this deliverable are: stakeholders, industry and academic working groups interested in security of 5G technologies, infrastructure providers, Standardization Organizations, the European Commission, Academic and Research stakeholders, as well as non-experts interested in 5G security.

1.3 Structure

The deliverable is structured as follows: Section 2 describes the integration and validation platform to develop, integrate and validate the demonstrations. Sections 3, 4 and 5 detail the implementation, HLA mapping, results and lessons learned of Demonstrator 1, 2 and 3, respectively. Section 6 provides an overview of the HLA coverage by each Demonstrator. Section 7 concludes the document. Appendix A provides the integration tests that were deployed to validate the interoperability of the enablers in each Demonstrator and provides implementation details regarding specific tools used in Demo 1.

2 Final integration and validation platform

This section aims to describe the final used INSPIRE5G-plus infrastructure and the tools shared among the consortium partners in order to develop, integrate and validate the demonstrations presented in the following sections.

2.1 INSPIRE-5Gplus VPN

Since its first description in D5.2 [1], the VPN infrastructure and the number of testbeds involved has kept most of its structure with the CTTC testbed offering evaluation, computing and NFV resources to the rest of partners. Figure 1 shows the changes implemented in the testbed and VPN infrastructure:

– <u>Testbed/Computing resources:</u>

  A new testbed placed in the EURESCOM premises with a small set of generic server resources were available in a temporary agreement with Thales, for the second partner to place its WP3/WP4 enablers as they already do with the CTTC generic testbed.

  Due to an external reason from the INSPRIE5G-plus context, AALTO human resources left that organization and move to UOULU. Due to this fact, AALTO was no more part of the consortium and so its testbed was removed.

  One more aspect is the use of the resource done by Orange. Despite not having their own testbed, they implemented their enablers through the ORANGE POLAND (OPL) testbed. From there, ORANGE offered their enablers for the Demos in which they participate.

– <u>VPN Clients:</u>

  Due to the removal and addition of testbeds, some actions were done with the existing VPN clients. First of all, for security reasons, the AALTO VPN client was erased and could not be used anymore to access the INSPIRE5G-Plus VPN. Then, due to the experimental actions being done in this last part of the project, multiple partners requested new VPN clients in order to test their enablers across the different testbeds/domains. The new clients based on the IP 10.0.37.X were done for UMU (.31), EUR (.33), MI (.35 and .39) and THALES (.37).

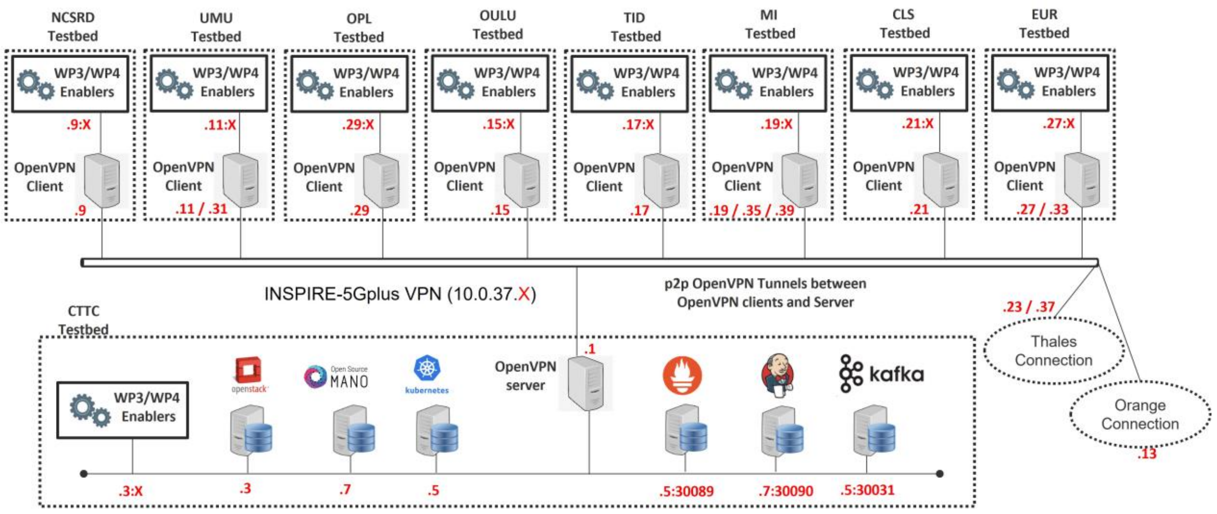


Figure 1: VPN Infrastructure.

Finally, one more action done through the VPN infrastructure is the definition of domain names and so, the use of a Domain Name Server (DNS) was added into the physical infrastructure within the CTTC premises, and the list of names presented in Table 1:

DNS tag	Associated to the:
*openstack.cttc.es	OpenStack resources placed in the CTTC testbed
*k8s.cttc.es	Kubernetes resources placed in the CTTC testbed
*osm.cttc.es	Open Source MANO resources placed in the CTTC testbed
*.demokritos.cttc.es	resources placed in the NCSRD testbed
*.umu.cttc.es	resources placed in the UMU testbed
*.oulu.cttc.es	resources placed in the UOULU testbed
*.tid.cttc.es	resources placed in the TID testbed
*.cls.cttc.es	resources placed in the CLS testbed
*.thales.cttc.es	resources placed in the EUR testbed
*.eurescom.cttc.es	resources placed in the EUR testbed
*.opl.cttc.es	resources placed in the OPL testbed
*.orange.cttc.es	resources placed in the OPL testbed

Table 1: DNS association with INSPIRE-5Gplus physical infrastructure.

2.2 Port forwarding

Since Security Management Domains (SMDs) are interconnected through the VPN exposed in section 2.1, and each SMD manages their own internal network infrastructure, different port forwarding solutions have been adopted. This approach will allow accessing services between SMDs through the VPN.

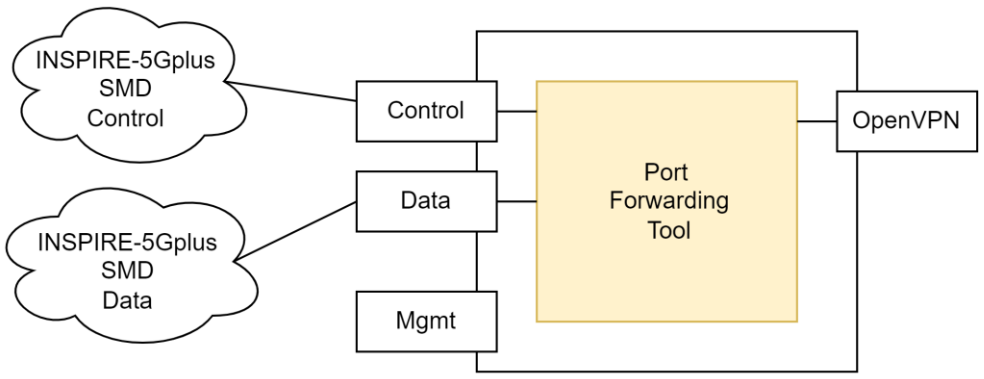


Figure 2: Port forwarding tool.

Figure 2 shows the approach to provide data plane/control plane communication through the VPN to internal SMD services/resources. Port forwarding tools allow specifying listening UDP/TCP ports and forwarding the received traffic to another port. APPENDIX 3.1, 3.2, 3.3 provide guidelines to instantiate the different port forwarding tools used in INSPIRE-5GPlus such as HAProxy, Nginx or iptables.

2.3 Monitoring (Prometheus)

To monitor and obtain the KPIs for each demo, a monitoring service with the Prometheus solution is offered to all the partners to be used with their enablers. Despite this service, in some cases, the outcomes related to the KPIs were obtained without the need to use Prometheus due to the simplicity in the obtention process of that KPI. Instead, other enablers that required to gather higher amounts of information made use of the offered service.

An example of this second case, is the monitoring integration for the Proof of Transit (PoT) and Smart Traffic Analyzer (STA) enablers. From the PoT enabler, metrics are exposed in the PoT controller, and for the STA enabler, they are exposed once the service is launched. Then by using a Prometheus Pushgateway, those metrics are collected to a central point so it can be accessible through the VPN. This will give access to the status and monitoring metrics of those enablers once they are deployed.

Below, it can be found the metrics that are collected for each of the previous enablers:

- Mean time to detect (STA)
- Mean time to detect valid verification (PoT)
- Mean time to detect invalid verification (PoT)

2.4 ICT Fabric (Kafka, Kubernetes, Istio)

Integration Fabric is designed to enable the intra and inter SMD communication, it also performs registration and discovery services among other intra/inter management functionalities. In addition, Integration Fabric has been designed to perform communication-related security features in a service mesh. According to the ETSI ZSM Integration Fabric definition, an integration fabric implementation should cover specific requirements such as service registration, service discovery, service communication or service routing. Figure 3 shows the INSPIRE-5GPlus integration fabric technologies and the deployment on each SMD. The combination of Istio and Kubernetes provides the required service mesh capabilities whereas Kafka integration improves the communication service by providing distributed event streaming features to the platform. Besides, those features are also provided intra domain and inter domain, thus, SMD services are able to reach/communicate each other by using different approaches (e.g., API or events) according to the established security properties. Authentication, access control and other security features like automatic E2E channel protection between control plane services are also intrinsic part of the solution.

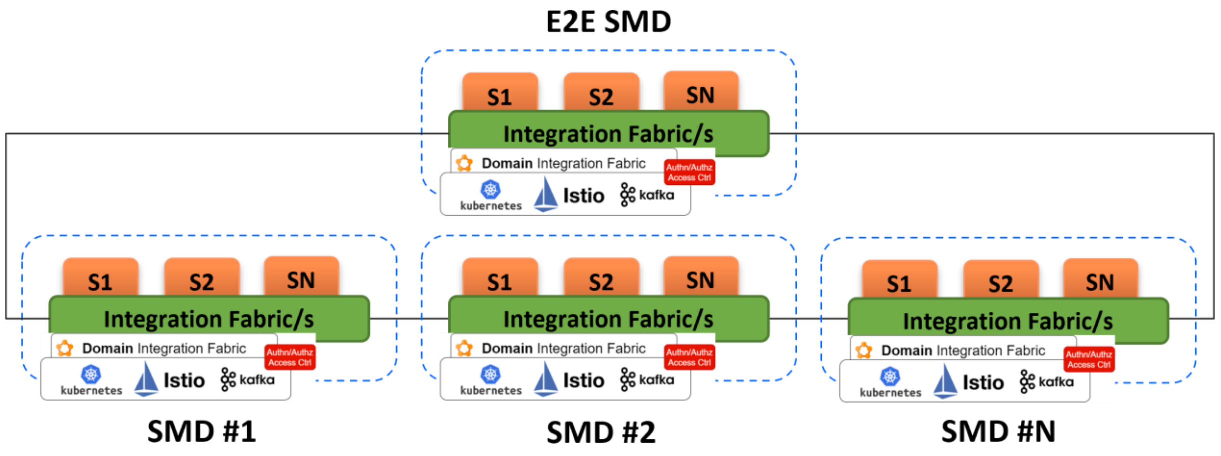


Figure 3: INSPIRE-5GPlus Integration Fabric Technologies.

APENDIX A.6 shows an example of the technical requirements as well as a technical guide to deploy this integration of technologies as Integration Fabric.

2.5 Kubernetes, Openstack and OSM

Finally, within the infrastructure shared among the INSPIRE5G-plus partners, there is a set of NFV resources available at the physical layer ready to be used (if necessary) by the different demos. In

D5.2, it was described that within the CTTC testbed there were three servers offering to the rest of the INSPIRE5G-Plus partners the following services/tools:

- Kubernetes node
- OpenStack node
- Open-Source MANO (OSM) node

Since D5.2, other testbeds had implemented the use of these three tools, especially in Demo1 as OSM was the NFV-O selected to be used on all the SMDs involved. Regarding the use of OpenStack and Kubernetes, each SMD decided which one to use for the services deployment.

3 Demo 1 – Security Management Closed Loop

This Demo aims to demonstrate the instantiation of the INSPIRE-5GPlus High-Level Architecture and the INSPIRE-5GPlus closed-loop security management across multiple domains. To this end, the demo can be summarized as the request and realization of two network slices with specific SSLAs from a 5G network customer. The SSLAs instantiation will lead to the unbundling of slices with services and associated security; and reactively mitigate attacks on the deployed services to maintain the SSLAs defined.

Table 2 and Table 3 summarize the enablers and assets used in Demo 1 and were developed in INSPIRE-5Gplus. A detail description of these tools can be found in the respective INSPIRE-5Gplus deliverables:

Enabler name	Partner
Security Orchestrator - D3.4 Section 2.13 [2]	UMU & TSG
E2E Security Orchestrator - D3.4 Section 2.12 [2]	UMU
Policy Framework - D3.4 Section 2.17 [2]	UMU
E2E Policy Framework - D3.4 Section 2.17 [2]	UMU
Trust Reputation Manager	UMU
E2E Trust Reputation Manager	UMU
Integration Fabric - D3.4 Section 2.20 [2]	UMU
Data Services - D2.4 Section 2.1.12, - D3.4 Section 2.13 [3]	UMU
SSLA Manager - D3.4 Section 2.15, - D3.2 Section 5.2 [2]	TSG
Decision Engine - D3.4 Sections 2.4, 2.10 [2]	TSG
E2E Decision Engine - D3.4 Section 2.10 [2]	TSG
Virtual Channel Protection - D3.1 Sections 4.7, 5.12 [4]	TSG
Security Data Collector - D3.4 Sections 2.3, 2.4, 2.21 [2]	MI
Security Analytics Engine - D3.4 Section 2.4, 2.3 [2]	MI
E2E Security Slice Manager - D3.4 Section 2.16 [2]	CTTC
V2X Misbehaviour Detector - D3.4 Section 2.4 [2]	CTTC
PoT Controller - D4.1 Section 4.2 [5]	TID
I2NSF Controller - D3.4 Section 2.9 [2]	TID
STA - D3.3 Section 3.2 [6]	TID

Table 2: Participating WP3/WP4 Enablers.

Asset name	Partner
I2NSF and PoT Agents - D3.4 Section 2.13 [2]	TID
DTLS Proxy (VCP data plane) - D3.1 Section 5.12 [4]	TSG
IAM - Identity & Access Management (VCP) - D3.1 Section 4.9 [4]	TSG
SFSBroker - D3.4 Section 2.18 [2]	OULU

KACD - D5.2 Section 5.6.3 [1]	OULU & TSG
5G Core & RAN Agent - D3.4 Section 2.21 [2]	UMU

Table 3: Additional tools used in Demo 1.

3.1 Storyline

This demo aims at showcasing the INSPIRE-5Gplus security closed-loop as well as the instantiation of the High-Level Architecture, across multiple domains and sites interconnected through the Integration Fabric. *To that end, the demo showcases the request for two SSLAs between a 5G network slice Broker and a 5G operator piloted by INSPIRE-5Gplus.* On the one hand, the first SSLA will request the protection of the communication between the UE access domain and a 5G service domain located in different and separated SMDs (e.g., through a backhaul connectivity), as well as other security requirements to avoid security issues (e.g., Abnormal behaviour, DDoS protection). The enforcement of this SSLA will involve different domains to deploy a 5G network slice composed by the service itself as well as the security elements (i.e., channel protection IPsec proxies and different monitoring assets) to ensure the SSLA compliance for any traffic. On the other hand, the second SSLA will focus on securing sensor data exchange between IoT sensors in remote sites and a global IoT supervision centre - interconnected via a dedicated 5G network slice - by requesting different types of channel protection between IoT devices and the supervision centre's IoT broker, using DTLS proxies. The enforcement of this SSLA will also involve different management domains, this time to deploy a 5G network slice composed by the IoT supervision service (IoT broker mainly), the IoT edge and the IoT channel protection proxies (as close as possible to the sources and destination) and, finally, the monitoring agents which ensure the SSLA compliance.

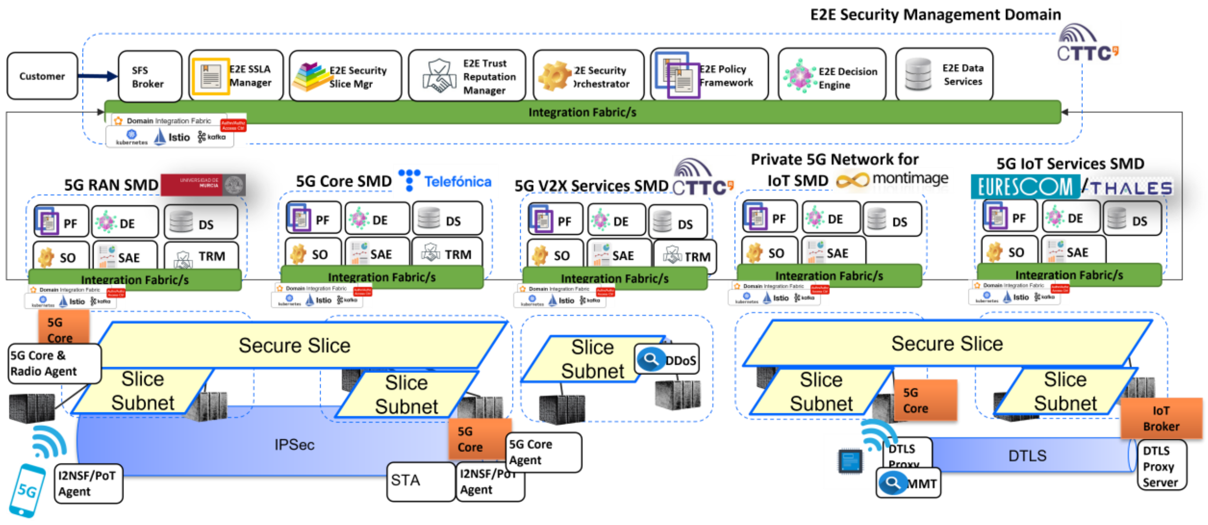


Figure 4: Demo1 overview

Once the SSLAs have been enforced, the reactive capabilities provided by the *INSPIRE-5Gplus* closed-loop will be showcased by simulating different kinds of attacks, internal and external. First, an internal attack will be simulated in which the attacker compromises the 5G Service image with a cryptomining malware, so when the 5G service is deployed as part of the network slice based on the first SSLA, it starts draining the infrastructure resources by mining. In addition, when the attacker verifies that the attack is ongoing (e.g., the compromised node appears in a mining platform), he/she will start an external DDoS attack on the infrastructure to finish depleting the resources. Besides, a second internal attack will be simulated in which the attacker compromises the VCP (Virtual Channel Protection) enabler enforcing the IoT communication channel protection (e.g., DTLS proxy settings).

Since multiple monitoring and security assets were deployed as part of the SSLAs enforcement, different security alerts will be generated which will trigger a series of reactive countermeasures to mitigate the attacks and return to a safe state. Specifically, the abnormal behaviour of the 5G service will be detected and the service will be modified to use the most trustworthy version of the software (trust of current version is dramatically decreased due the attack). This intra-domain countermeasure will be propagated at the E2E level that will trigger the same reactive countermeasure on those management domains that contains the same 5G service image.

The DDoS attack will be also detected, this time in a third domain which will apply reactive policies to filter the malicious traffic. This countermeasure will be also propagated to the End-to-End (E2E) domain that will request the countermeasure enforcement in the 5G RAN management domain to prevent the attack on it.

Finally, if IoT channel protection (DTLS) settings have been violated, the monitoring probe of the Security Analytics Engine enabler detects it and notifies the Decision Engine enabler which triggers the migration (re-deployment) of the affected VCP (DTLS proxy) container by the local SMD Security Orchestrator (SO) to a different server. This will restore the IoT communication to a safe state. The countermeasure will be propagated to the E2E domain but, in this case, it is not required to propagate the countermeasure to other domains.

This demo includes part of proactive and reactive security measures, scenarios and deployments that were previously defined as part of the Use Cases (UC) E, G, O, S and W, in deliverable D2.3 [7] and test cases 1, 2, 3 and 4 in deliverable D5.1 [8].

3.2 HLA Mapping

INSPIRE- 5Gplus High-Level Architecture (HLA) will be covered by providing an E2E SMD and five SMDs, where the closed-loop defined will conduct proactive and reactive processes over the entire system, showing the functional interactions between the enablers of the HLA, carrying out the **enforcement of SSLAs and reactive security policies across multiple SMDs**.

Figure 5 shows the HLA Mapping as well as the participating security enablers. First, **SFSBroker** receives the resource requests from the tenants and performs the brokering operation based on the available resources and pricing offers from the resource providers. The secured service level agreement has been established upon the selection of optimal slice for the consumer request. The slice manager(s) of the resource providers will be invoked from SFSBroker to instantiate the slice. Then, **the E2E Slice Manager** will generate a 5G Security Slice Orchestration policy from the SSLA and the 5G network slice requirements. This policy (e.g., MSPL-OP) will contain different policies to cover different **Security Capabilities** as well as the SMDs where the policies must be enforced to. The MSPL-OP will be sent to the **E2E SO** which will coordinate, validate, and orchestrate the **E2E Security Policy Enforcement,** modifying, splitting, generating new policies (if required) and sending them to the different **SMDs SO**. Then, SMDs Security Orchestrator will intercommunicate with the **Policy Framework** in order to start the SMD orchestration and enforcement process that will comprise policy conflicts and dependencies detection, trust-based policies orchestration (requesting trust metrics to **Trust Reputation manager**) and security assets selection. After that, a translation process is performed obtaining final asset configurations from security policies and selected assets. The SMDs SO will enforce the final configurations, also deploying new VNFs dynamically if required.

To showcase different security capabilities over the HLA, two different SSLAs will be enforced during the demo. Particularly, SSLA 1 will result in the deployment of a multi-domain **Secured Network Slice** composed by a **5G Core** and two **I2NSF Agents,** that will be configured through the **I2NSF Controller** (in which will relay the E2E Channel protection) as well as **three different monitoring assets (PoT Agent, STA, V2X DDoS)** for different security capabilities to ensure that the SSLA requirements are covered as expected. These monitoring assets will feed the SMD **Security Analytics Engine** (and also the Trust Reputation Manager, depending on the asset), which will generate alerts that will be consumed by the **Decision Engine** that will compute new reactive security policies at SMD level. These new policies will be enforced dynamically in the SMD, but they will be also propagated to the **E2E Decision Engine** that will compute E2E reactions if required. Regarding the SSLA 2, it will be enforced by using similar HLA security enablers but also other ones (**MMT**, **IoT-VCP**) able to deal with IoT services and devices and part of the 5G infrastructure.

3.2.1 Data Collection

The security enablers that provide Data Collection features in Demo1 as well as their specific functionality in this demo are described below.

MI Data Collector

The monitoring agent (MMT-Probe) is involved in Demo1 by capturing and analysing online network packets at a given network interface card. It extracts the packet and flow features and sends them to SAE. The features can be grouped into 2 big groups:

- The features related to the detection of any violation of 5G IoT channel protection. Specifically, the attributes of DTLS traffic as listed in the following:
    - Content-Type: a number representing DTLS type of data content. Example: 21 (Alert), 22 (Handshake), ...
    - Version: a number representing the DTLS version. Example: 0xFEFF (version 1.0), 0xFEFD (version 1.1), 0xFEFC (version 1.3)
    - Epoch, Sequence Number, Length: numbers representing respectively epoch, sequence number and total length.
- Cipher Suite: an array of numbers representing the cipher suite proposed by the DTLS client in a Client Hello Handshake packet. Example: 0xC0A4 (TLS_PSK_WITH_AES_128_CCM), 0xC0A5 (TLS_PSK_WITH_AES_256_CCM), etc. This feature is available only in the Client Hello Handshake packets. The features are mostly related to a session or similar sessions to display graphical statistics in SAE, such as: total number of IP packets in a session, session duration, time interval between packets in a session, number of packets per second, average length of IP header, number of bytes per second, sequence of packet lengths, sequence of packet times, number of packets of a certain protocol (e.g., TCP), number of packets of certain types (e.g., TCP: SYN, FIN, RST, PUSH, ACK, URG), etc.

CTTC-V2X Data Collector

For the realization of the data collection HLA functionality in the V2X misbehaviour detector enabler, the HLA component performs the fusion of V2X network traces that are streamed from the data plane using VMs, in which VMs emulate the representation of vehicles within the RAN. These V2X traces are based on an open-source vehicular anomaly-detection dataset[2]. The dataset incorporates various misbehaviour attack types and, for each attack type, a log file per vehicle is generated which contains basic safety messages transmitted by neighbouring vehicles over its entire trajectory. Basic safety messages include -among other features- three-dimensional vectors for vehicle's position, speed, acceleration and heading angle.

TID Data Collector

Smart Traffic Analyzer (STA) is based on a network probe, that received a copy of the signalling traffic on the 5G Core. For the demo 1 purpose, the probe is deployed as a container running in the 5G core service virtual machine, in order to access those signalling traffic. Acting as the data collector in the HLA, the traffic is processed and aggregated in network flows identified by 5-tuples (IP source and port, IP destination and port and protocol) in each direction (client to server and vice versa). For each flow,

[2] VeReMi dataset: https://github.com/josephkamel/VeReMi-Dataset

the following features are distilled and diverted to the ML inference engine in the local Security Analytics engine:

- total number of packets observed
- number of segments with the ACK field set to 1
- number of bytes sent in the payload
- number of segments with payload
- number of bytes transmitted in the payload, including retransmissions

No payload information is processed.

3.2.2 Security Analytics

The security enablers that provide Security Analytics features in Demo1 as well as their specific functionality in this demo are described below.

CTTC-V2X Security Analytics

For the realization of the security analytics HLA functionality in the V2X misbehaviour detector enabler, the HLA component sequentially analyses the incoming streaming basic safety message reports based on the mobility patterns parameters such as position, velocity, and acceleration, to instruct a reinforcement-learning algorithm for the detection of misbehaviour patterns. In particular, the aggregated information constitutes a time-series repository of received basic safety messages with intrinsic temporal and spatial inter-dependencies; the information contained in each basic safety message is constantly evolving over time along the vehicle trajectory while messages from neighbouring vehicles exhibit high spatial dependency. In DDoS attacks, a vehicle transmits basic safety messages 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. The V2X misbehaviour detector is able to detect the abnormal inter-arrival time of such messages (misbehaviour pattern) by comparing against the time threshold specified by the applied policy (conforming to the standard).

MI Security Analytics

The security analysis in the security monitoring framework (MMT) is based on a given set of detection temporal logic-type rules. Complex network event processing performed by the security agents (MMT-Probe) allows detecting when the cipher suite in DTLS packets is violated. Particularly, for each incoming packet, the MI Security Analytics will receive from the Data Collector its cipher suite if the packet is DTLS. The MI Security Analytics then verifies whether the cipher in the suite does not exist in the pre-configured cipher suite that has been provided by the Security Orchestrator. If so, The Security Analytics will generate an alert and send it to the Decision Engine. The alert contains (1) the details about the violated cipher suite, as well as the source and destination IPs of the packet, (2) the additional information given by the SO, such as, the orchestration ID, and (3) the hostname of the current machine that hosts the DTLS proxy and the MI Data Collector.

TID STA

The security analysis engine for the STA is a lightweight inference engine based on a statistical machine learning model. The model is based on the Random Forest algorithm, to detect and classify cryptomining attack. The model has been designed, pre-trained and integrated in the STA using the Network digital twin Mouseworld enabler following the design cycle defined in D3.3, where a representative dataset has been generated for this purpose with signalling 5G traffic (SBA) and cryptomining attacks.

The STA solution co-locates trained inference model in the network probe with the data collector in charge of transforming and feeding the model.

In the demo, the STA will generate, share and alert when a network flow related to encrypted cryptomining malware activity is detected by the inference engine with high level of confidence. The alert includes the complete 5-tuple information (source and destination IP, source and destination port, and protocol) and the timestamp. This information allows to identify the IP address (source IP) of the component infected by the malware, to take corrective actions, by the Decision Engine, such as request redeployment of the component associated to the IP address.

3.2.3 Decision Engine

CTTC-V2X DE

For the realization of the decision engine HLA functionality in the V2X misbehaviour detector enabler, the HLA component, upon detection of misbehaviour, provides the verdict to the security orchestrator to apply the pre-determined reactive security policy, i.e., misbehaving data source to be isolated, dropped, or blocked. For the decision upon detection of DDoS attacks in the SMD, interaction is sought with the security orchestrator (and policy framework) for the applied policy which is represented by the inter-arrival time threshold of the basic safety messages. The issued security policy is expressed using MSPL.

THALES PyrDE DE

The Decision Engine participates in the Demo1 during the DDoS & the cryptomining attacks scenarios and in the IoT channel protection scenario. The functional Decision Engine described in the overall WP2 design is implemented in the component called PyrDE Decision Engine. The PyrDE Decision Engine manages the reactive adaptation of security at the SMD level and at the E2E level.

In each scenario, during an attack, an alert is raised by a SMD Security Analytic Engine feeding the local Thales PyrDE Decision Engine. This alert is first filtered into a set of known types. The PyrDE contains the necessary parsers to support the various SAE used in the project. Then the PyrDE Decision Engine parses the alert and detects the affected service. It generates a new security policy MSPL related to the type of the raised alert. The MSPL is adapted to support the underlying Security Orchestrator API. Finally, the PyrDE Decision Engine pushes the newly generated policy to the SMD security orchestrator. While this first reactive loop runs inside a SMD, the local PyrDE Decision Engine also escalates the alert to the E2E level for further reactions.

3.3 Testbed description

Demo 1 shows an infrastructure composed of five management domains with an E2E management domain on top. As illustrated in Figure 5, the complete Demo1 infrastructure is distributed in the testbed of five different INSPIRE5G-plus partners. On top of it, there is the **E2E SMD placed in the CTTC premises** with those enablers necessary to orchestrate and monitor the services deployed from an E2E point of view. This domain contains the following enablers: the SFS Broker, the E2E SSLA Manager, the E2E Slice Manager, the E2E Security Orchestrator, the E2E Policy Framework, E2E Trust Reputation Manager, Data Services, Integration Fabric, and the E2E Decision Engine.

Below, each of the SMDs may have different enablers and/or tools deployed in order to enforce different security requirements:

The SMD in the CTTC premises contains the Policy Framework, Trust Reputation Manager and Security Orchestrator (UMU) together with the V2X Misbehaviour Detector tool to be used in the detection of the DDoS attack in this domain. Data Services and Integration Fabric are also provided. Moreover, the domain Network Slice Manager and MANO elements will be an instance of the Open-Source MANO (OSM) together with either OpenStack or Kubernetes.

The SMD in the UMU premises contains the Policy Framework, Security orchestrator (UMU), Trust Reputation Manager, Integration Fabric, and Data Services together with the I2NSF/PoT Agent to be used in the IPSec E2E Channel Protection enforcement. Moreover, the domain Network Slice Manager and MANO element will be an instance of the OSM together with either OpenStack or Kubernetes. 5G RAN infrastructure is provided as well as a 5G RAN agent to apply 5G specific configurations as part of the E2E reactive countermeasure for the cryptomining and DDoS attacks. Another instance of the compromised 5G Core will be also deployed to showcase the E2E reactive countermeasure for the mining attack.

The SMD in the MI premises contains IoT devices and the VCP enabler's data plane component (DTLS proxy) to secure the communications with the IoT broker that is installed in the EURESCOM joint premises. The MI premises contain also a MMT-Probe VNF to detect internal attack that tries to downgrade the E2E IoT communication channel protection. The Security Analytic Engine (SAE) is also installed to translate violation alerts to the Decision Engine and to provide control API to control MMT-Probe. The MMT-Probe and SAE are deployed inside a Kubernetes cluster to easily manage and reinitialize MMT-Probe to its pristine original stage as part of the mitigation for the communication channel protection attacks. The DE and SO are also installed inside the Kubernetes cluster. An OSM is installed within MicroStack to provide a bridge to connect the SO and the Kubernetes cluster.

The SMD in the TID premises contains the Policy Framework, Security orchestrator (UMU), Trust Reputation manager, Integration Fabric, Data Services, and the Decision Engine for the closed loop management. Additionally, several security agents are included, the STA, the I2NSF and PoT controllers that deploy the configurations into I2NSF/PoT Agent, also deployed (one here in SMD TID and the other in SMD UMU) for the IPSec E2E Channel protection enforcement. Additionally, OSM and OpenStack will be deployed for supporting the HLA components listed and also a 5G Core System, with software compromising it with cryptomining. The 5G Core will use SMD UMU 5G RAN through the protected slice.

The SMD in the EURESCOM premises contains an Integration Fabric instance, the SMD Security Orchestrator (TSG), the IoT broker (CoAP Publish-Subscribe), the VCP enabler's control plane and data plane (CoAP/DTLS proxy) that is enforcing the channel protection policy for all communications to the IoT broker from the IoTs in MI's SMD premises. IAM (Identity and Access Management) services supporting the VCP: identity and access management, authentication, and attribute-based authorization (ABAC) services, data encryption key management (KMS), certificate management (PKI). OSM that fulfils the role of Network Slice Manager and MANO. A Kubernetes infrastructure as deployment environment of the above components (e.g., Integration Fabric) and OSM's target VIM in particular.

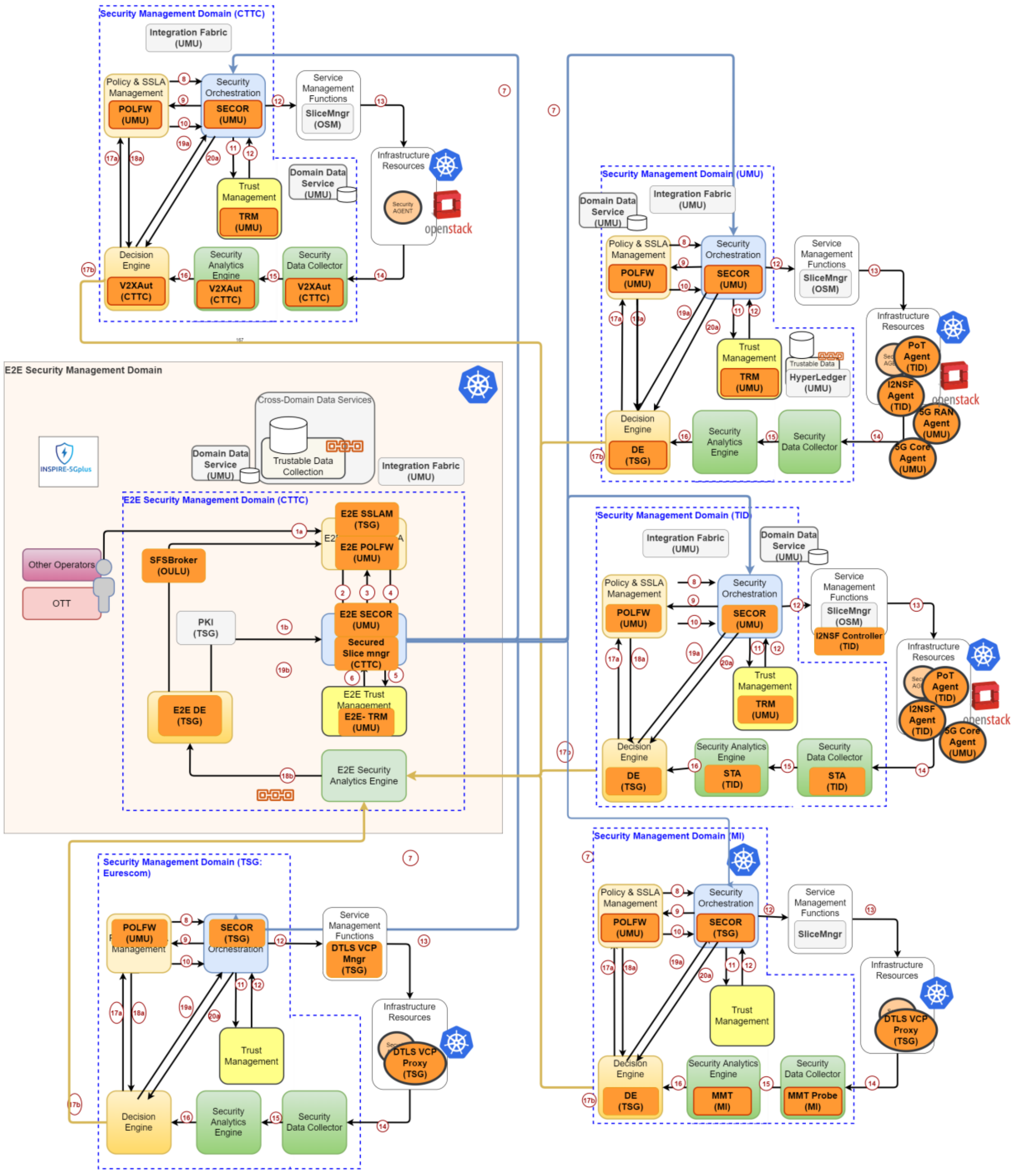

Security Management Domain (CTTC)
E2E Security Management Domain
E2E Security Management Domain (CTTC)
Security Management Domain (UMU)
Security Management Domain (TID)
Security Management Domain (TSG: Eurescom)
Security Management Domain (MI)
Policy & SSLA Management
Security Orchestration
Service Management Functions
Infrastructure Resources
Trust Management
Decision Engine
Security Analytics Engine
Security Data Collector
Domain Data Service (UMU)
Integration Fabric (UMU)
E2E SSLAM (TSG)
E2E POLFW (UMU)
E2E SECOR (UMU)
Secured Slice mngr (CTTC)
E2E Trust Management E2E- TRM (UMU)
E2E Security Analytics Engine
E2E DE (TSG)
SFSBroker (OULU)
PKI (TSG)
Other Operators
OTT
Cross-Domain Data Services
Trustable Data Collection
openstack

Figure 5: Demo1 deployment showing mapping to INSPIRE-5GPlus HLA.

3.4 Interaction diagrams/workflow

3.4.1 Closed-loop Proactive Workflow

The proactive part of the close loop comprises the interactions required for enforcing SSLAs across the infrastructure of the different Management Domains of the operator. Figure 6 represents a customer request that will be transformed into an SSLA as well as network slice requirements. This kind of network slice information and the SSLA IDs will be provided to the E2E Security Slice Manager that will retrieve the SSLAs from the E2E SSLA Manager and it will generate a 5G Security Slice Orchestration Policy that will also contain the required SMDs where to deploy the different network slice components. This policy is then provided to the E2E Security Orchestrator that will validate (by using the policy framework) and orchestrate the distribution of the required orchestration policies (MSPL-OP) from the E2E domain to the final SMDs.

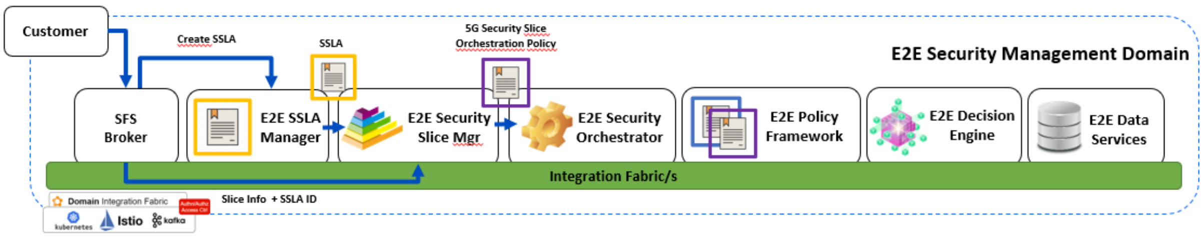


Figure 6: Network Slice and SSLA enforcement request.

Once the policies are received by each SMD Security Orchestrator, they will be in charge of orchestrating, selecting and deploying the necessary assets in the SMD infrastructure, and configuring the required components by translating the MSPL-OP into specific assets configurations. The following subsections provide specific workflow examples for each Demo 1 SSLA. It is important to highlight that whereas the workflow has been separated in subsections according to the security scope (i.e., channel protection, monitoring), both will be enforced simultaneously (or according to the E2E orchestration plan).

Figure 7 shows the detailed workflow for the Network Slice deployment and the SSLA enforcement. It is composed by the following interactions:

1. Customer/tenant requests resources, including the security requirements.
2. SFSBroker verifies whether the customer/tenant requirements can be fulfilled by the available capabilities.
3. If the service requirements cannot be accomplished, the SFSBroker informs back to the customer about the inability to satisfy the request.
4. SFSBroker requests the creation of a new SSLA based on the security requirements defined by the customer.
5. SFSBroker obtains the ID of the SSLA created from the E2E SSLA Manager.
6. SFSBroker generates the E2E Network Slice Template (NST) to be deployed based on the service required by the customer.
7. E2E Slice Manager receives the information regarding the NST with the service to be deployed as well as the IDs of the involved SSLAs that models the security requirements for the NST.
8. E2E Slice Manager retrieves the SSLAs from the SSLA Manager using the specified SSLA IDs from the incoming request in the previous step.
9. E2E Slice Manager generates a Network Slice Instance (NSI) data object and translates it into a 5G Security Slice Orchestration Policy in MSPL-OP format.
10. E2E Slice Manager requests the E2E enforcement to the E2E Security Orchestrator. The request also includes the IDs of the SMDs that will be involved in the 5G security network slice.
11. (and 12) E2E Slice Manager informs the SFS Broker about the acceptance and deployment of the 5G network slice request.

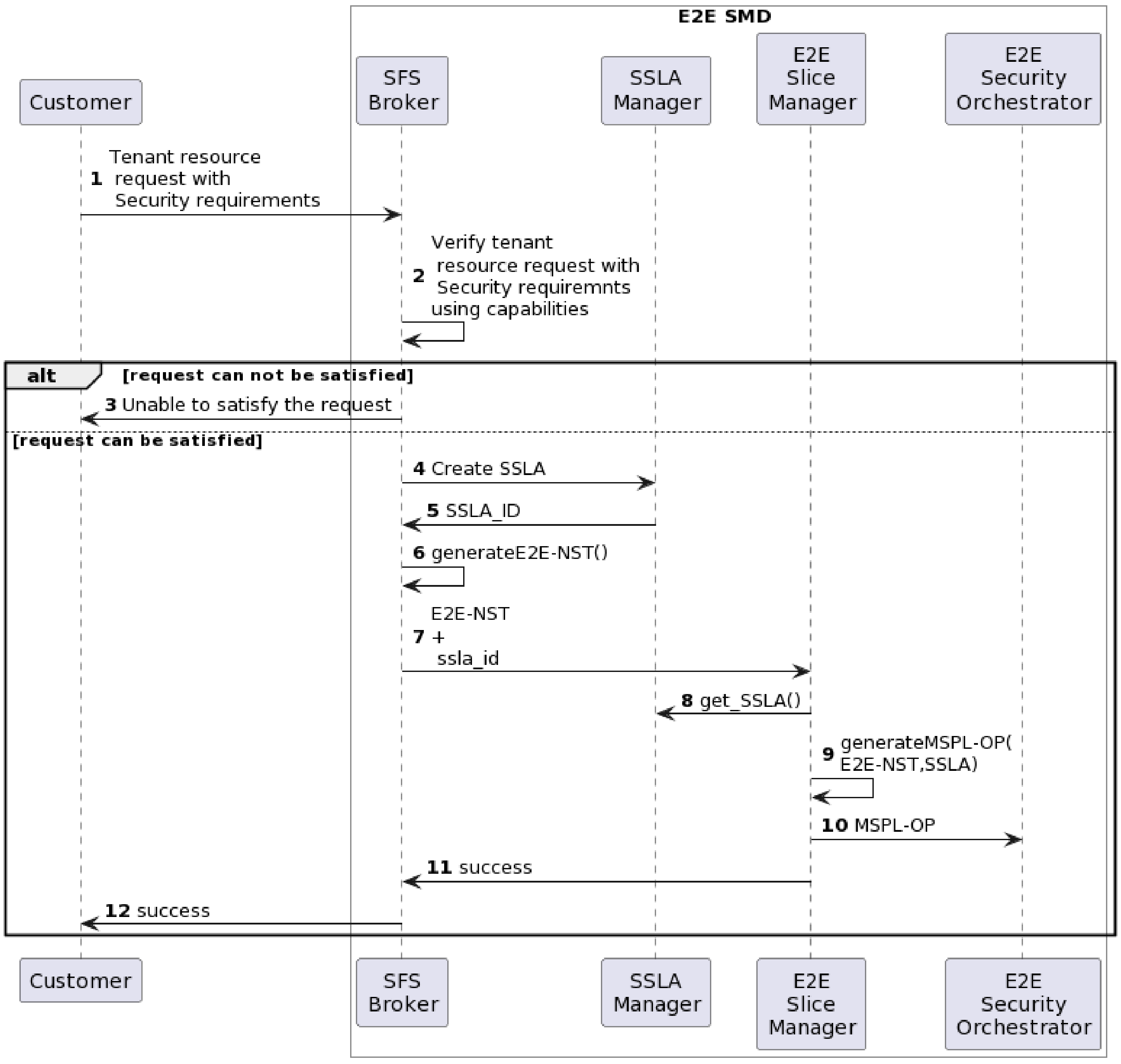

E2E SMD
Customer
SFS Broker
SSLA Manager
E2E Slice Manager
E2E Security Orchestrator
1 Tenant resource request with Security requirements
2 Verify tenant resource request with Security requiremnts using capabilities
alt [request can not be satisfied]
3 Unable to satisfy the request
[request can be satisfied]
4 Create SSLA
5 SSLA_ID
6 generateE2E-NST()
7 E2E-NST + ssla_id
8 get_SSLA()
9 generateMSPL-OP(E2E-NST,SSLA)
10 MSPL-OP
11 success
12 success

Figure 7: E2E SMD SSLA enforcement workflow.

3.4.2 SSLA1 Channel Protection

One part of the first SSLA requires to provide a 5G Core with channel protection as well as other security requirements. To this aim, the E2E Security Orchestrator will generate different 5G Security Slice Orchestration Policies for each involved SMD. In Figure 8, 5G Core SMD (TID) Security Orchestrator will orchestrate the received MSPL-OP which will result in the deployment of the 5G Core as well as the deployment and configuration of an I2NSF/PoT Agents, whereas 5G RAN SMD (UMU) Security Orchestrator will orchestrate the received MSPL-OP which will result in the deployment/configuration of an I2NSF/PoT Figure 8Agent (as the other part of the E2E channel protection tunnel). The configuration of the I2NSF/PoT Agents will be performed through the I2NSF/PoT Controller.

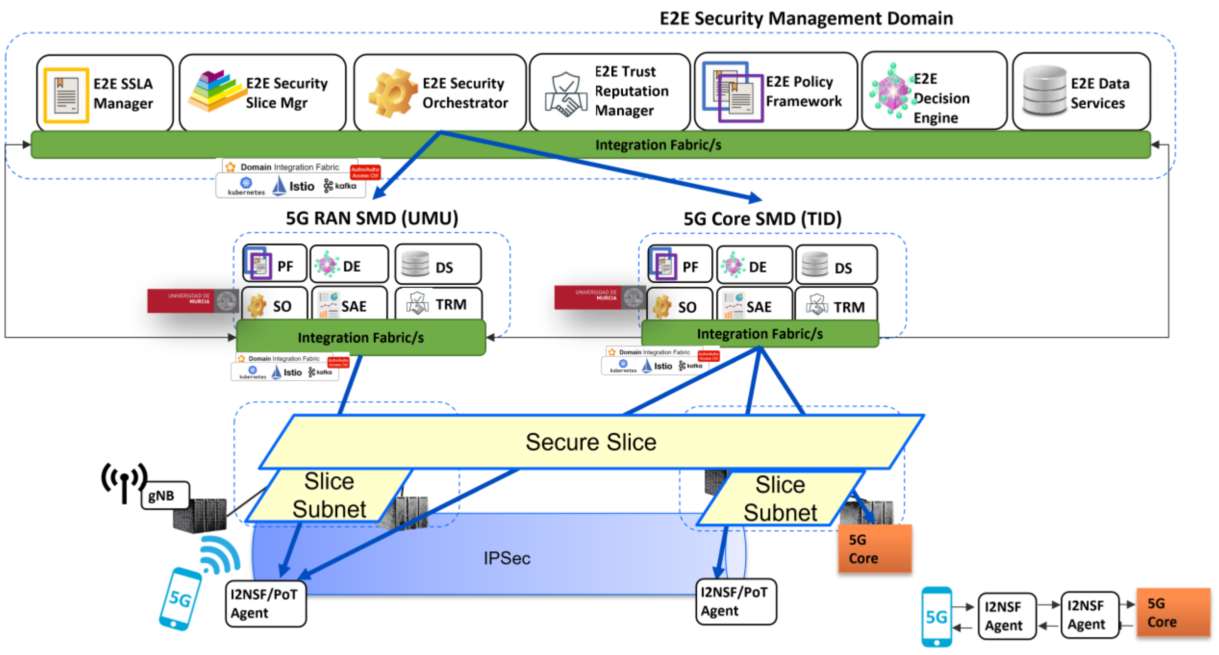


Figure 8: SSLA1 Channel protection enforcement workflow.

Figure 9 shows the detailed workflow for this part of the SSLA enforcement. It is composed by the following interactions:

TID SMD	UMU SMD
1. Security orchestrator receives the MSPL-OP 2. Security orchestrator request to Trust Reputation Manager the trust metrics for the candidate enablers. 3. Security Orchestrator generates the orchestration plan 4. Security Orchestrator requests MSPL-OP translations from the Policy framework for those enablers/assets selected according to the orchestration plan. 5. Security Orchestrator receives assets/enablers configurations. 6. Security Orchestrator requests the 5G slice deployment as part of the orchestration plan. 7. 5G Service is deployed. 8. I2NSF Agent and PoT Agent are deployed (since they can be deployed in the same machine) 9. Security Orchestrator receives the notification which indicates that slice is ready.	1. Security Orchestrator receives the MSPL-OP. 2. Security Orchestrator generates the orchestration plan. 3. Security Orchestrator requests MSPL-OP translations from the Policy framework for those enablers/assets selected according to the orchestration plan. 4. Security Orchestrator receives assets/enablers configurations. 5. Security Orchestrator requests the 5G slice deployment as part of the orchestration plan. 6. I2NSF Agent is deployed. 7. Security Orchestrator receives the notification which indicates that the slice is ready.

10. Security Orchestrator configures 5G service (if required). 11. Security Orchestrator requests the channel protection configuration to the I2NSF Controller. 12. I2NSF Controller configures I2SNF Agent deployed in both sides.	

Table 4: Interactions between TID and UMU SMDs for SSLA channel protection.

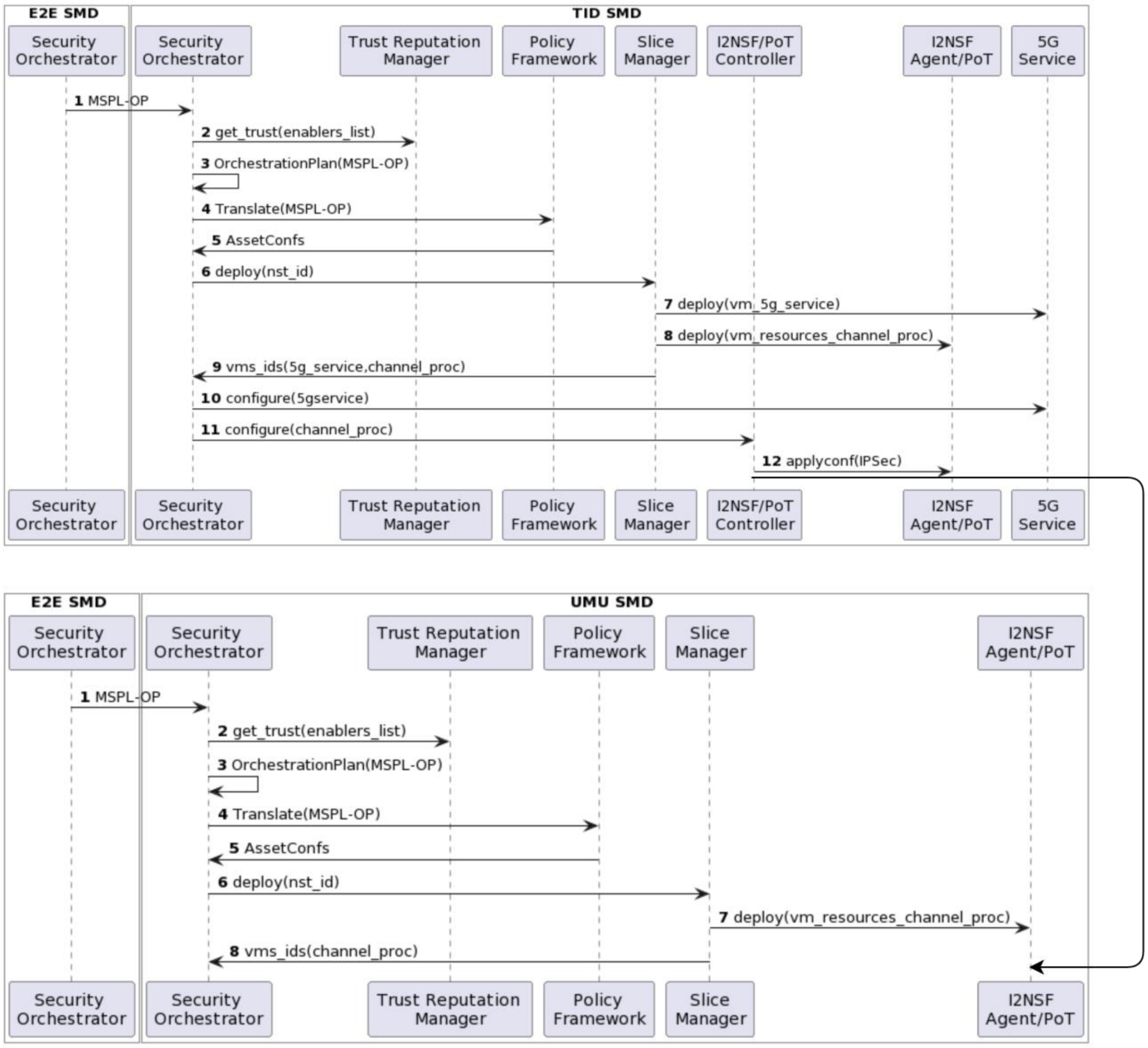


Figure 9: SSLA1 Channel protection enforcement workflow.

It is important to highlight that the 5G slice deployed at step 6 is composed by all the enablers/assets required to cope with the 5G security slice requirements in that domain. Besides, the specific order of the enforcement in both SMDs will depend on the E2E orchestration plan. For instance, in the demo the priorities are higher in those policies that contains the 5G service slice subnet information that will deploy the 5G Core so it will be enforced before the other part of the E2E slice. Moreover, configuration steps depend on subnet slice deployments so the orchestrators will enforce them only when subnet slices are ready.

3.4.3 SSLA1 Monitoring

Regarding the monitoring policies, they are also orchestrated according to the involved domains required to accomplish the SSLA. In this demo, a behavioural monitoring must be performed as near as possible of the 5G Core so the STA (TID) tool will be deployed and configured to this purpose. Since the SSLA also specifies rate limits to avoid DoS attacks in 5G services, a monitoring tool will be also deployed in those SMD that provides 5G services. In this demo, it will be demonstrated by instantiating a monitoring tool for DDoS detection in CTTC SMD, which will play the role of a 5G V2X Services SMD.

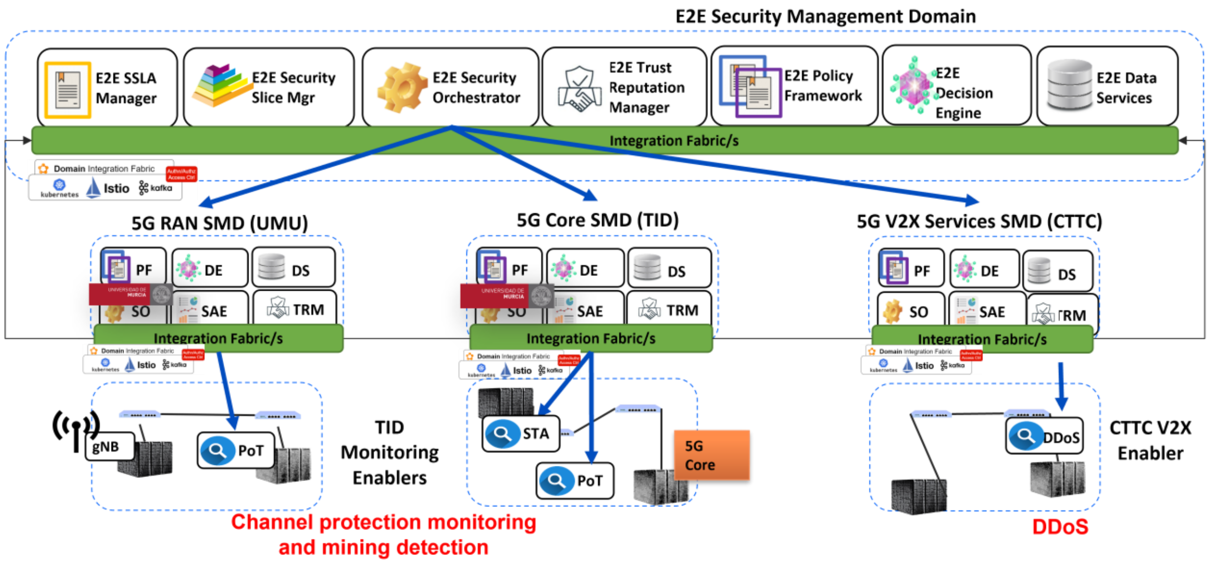


Figure 10: SSLA1 Monitoring enforcement workflow.

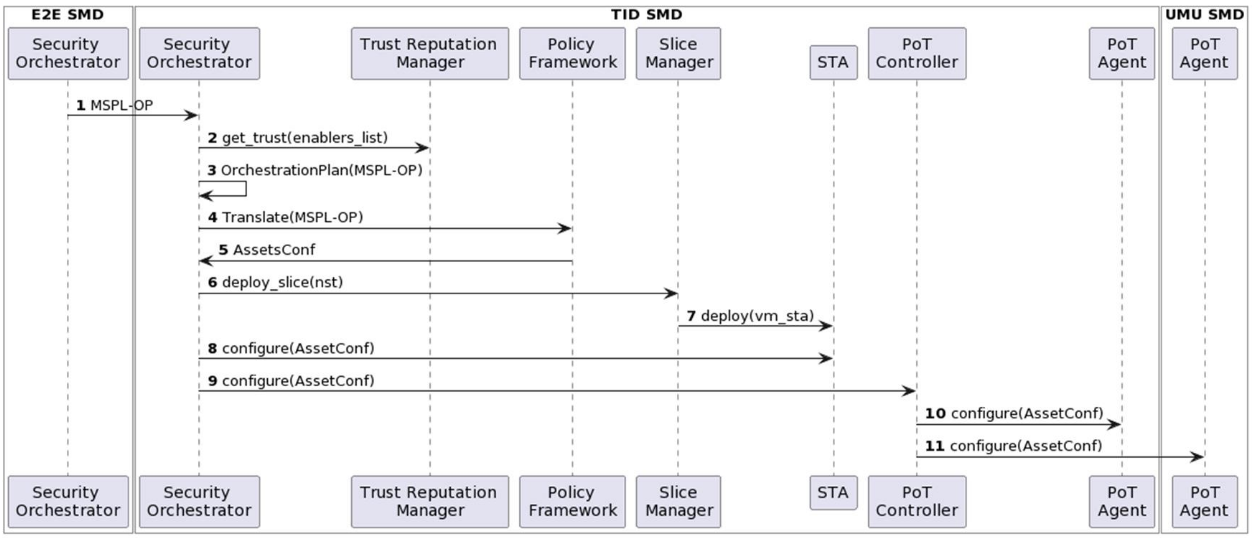


Figure 11: Mining detection and PoT enforcement workflow (TID/UMU SMD).

Figure 11 shows the detailed workflow for 5G RAN SMD and 5G Core SMD monitoring parts of the SSLA enforcement. It is composed by the following interactions:

1. Security orchestrator receives the MSPL-OP.
2. Security Orchestrator retrieves different kinds of information to prepare the orchestration plan, e.g., trust values.
3. Security Orchestrator generates the trust-based orchestration plan.

4. Security Orchestrator requests MSPL-OP translations from the Policy framework for those enablers/assets selected according to the orchestration plan. In this case, STA enabler and PoT.
5. Security Orchestrator receives STA and PoT configurations.
6. Security Orchestrator requests the 5G slice deployment as part of the orchestration plan.
7. STA enabler is deployed.
8. Security Orchestrator configures the STA, and network environment to mirror traffic from 5G service slice.
9. Security Orchestrator configures the PoT agent trough the PoT Controller.
10. TID SMD PoT agent is configured.
11. UMU SMD PoT agent is configured.

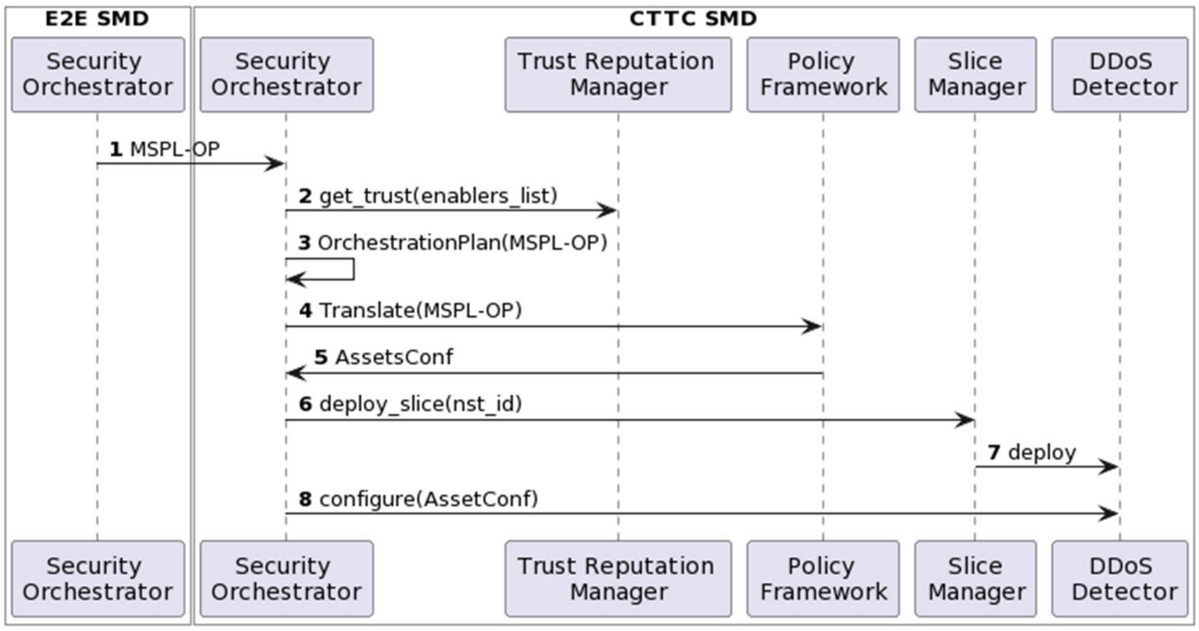


Figure 12: DDoS detection enforcement workflow (CTTC SMD).

Figure 12 shows the detailed workflow for DDoS configuration part of the SSLA enforcement. It is composed by the following interactions:

1. Security orchestrator receives the MSPL-OP.
2. Security Orchestrator retrieves different kinds of information to prepare the orchestration plan, e.g., trust values.
3. Security Orchestrator generates the trust-based orchestration plan.
4. Security Orchestrator requests MSPL-OP translations from the Policy framework for those enablers/assets selected according to the orchestration plan. In this case, V2X DDoS Detector enabler.
5. Security Orchestrator receives V2X DDoS Detector configurations.
6. Security Orchestrator requests the 5G slice deployment as part of the orchestration plan.
7. V2X DDoS Detector is deployed.
8. Security Orchestrator configures the V2X DDoS Detector.

3.4.4 SSLA 2 Secure 5G IoT Supervision Slice (E2E Communication Channel Protection)

One part of this SSLA requires an IoT Broker and channel protection between it and IoT devices. In this regard, 5G Security Slice policies will be provided to those SMDs that manage IoT environments inside the 5G infrastructure, according to the SSLA requirements. Figure 13 shows a high-level overview of the enforcement of this part of the SSLA. On the one hand, EURESCOM SMD will orchestrate the 5G Slice Security Policies by deploying the IoT Broker, as well as one side of the channel protection tunnel by deploying and configuring a DTLS proxy. On the other hand, MI SMD will orchestrate the 5G Slice Security Policies by deploying the other side of the channel protection tunnel as close as possible to the IoT devices.

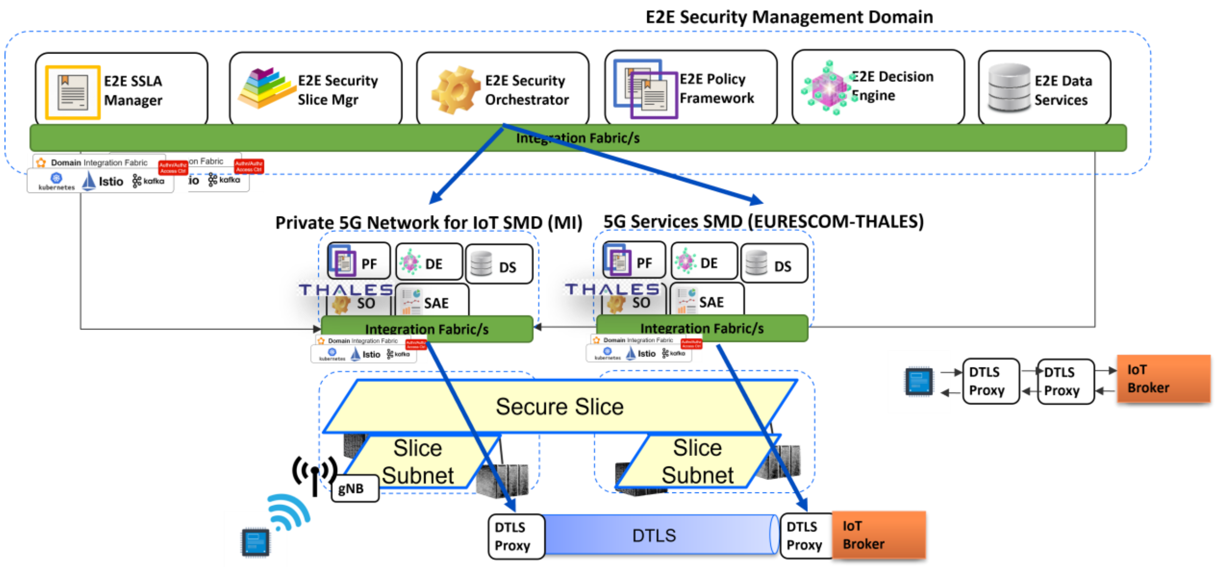


Figure 13: E2E 5G IoT Channel Protection Enforcement.

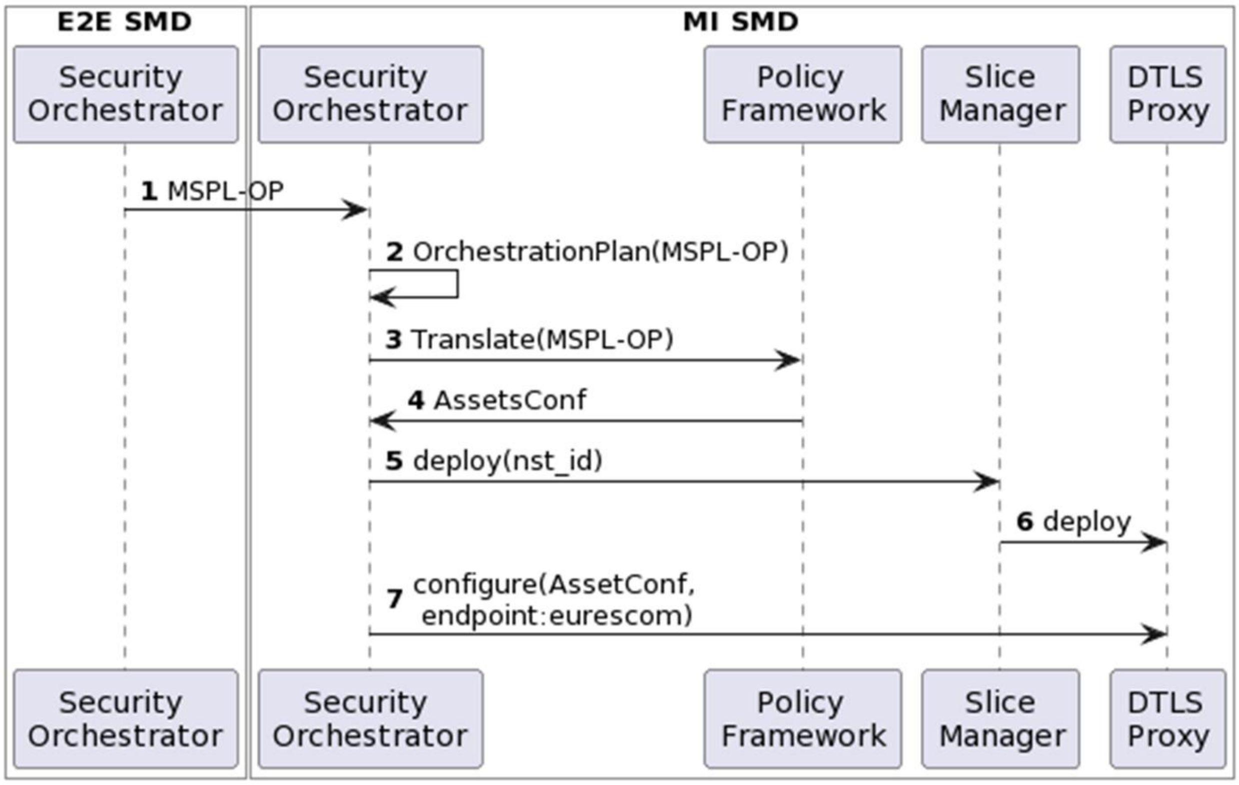


Figure 14: DTLS Channel protection enforcement (MI side).

Figure 14 shows the detailed workflow for this part of the SSLA enforcement. It is composed by the following interactions:

1. Security orchestrator receives the MSPL-OP.
2. Security Orchestrator generates the orchestration plan.
3. Security Orchestrator requests MSPL-OP translations from the Policy framework for those enablers/assets selected according to the orchestration plan. In this case, DTLS Proxy.
4. Security Orchestrator receives DTLS Proxy configurations.
5. Security Orchestrator requests the 5G slice deployment as part of the orchestration plan.
6. DTLS Proxy is deployed.
7. Security Orchestrator configures the DTLS Proxy.

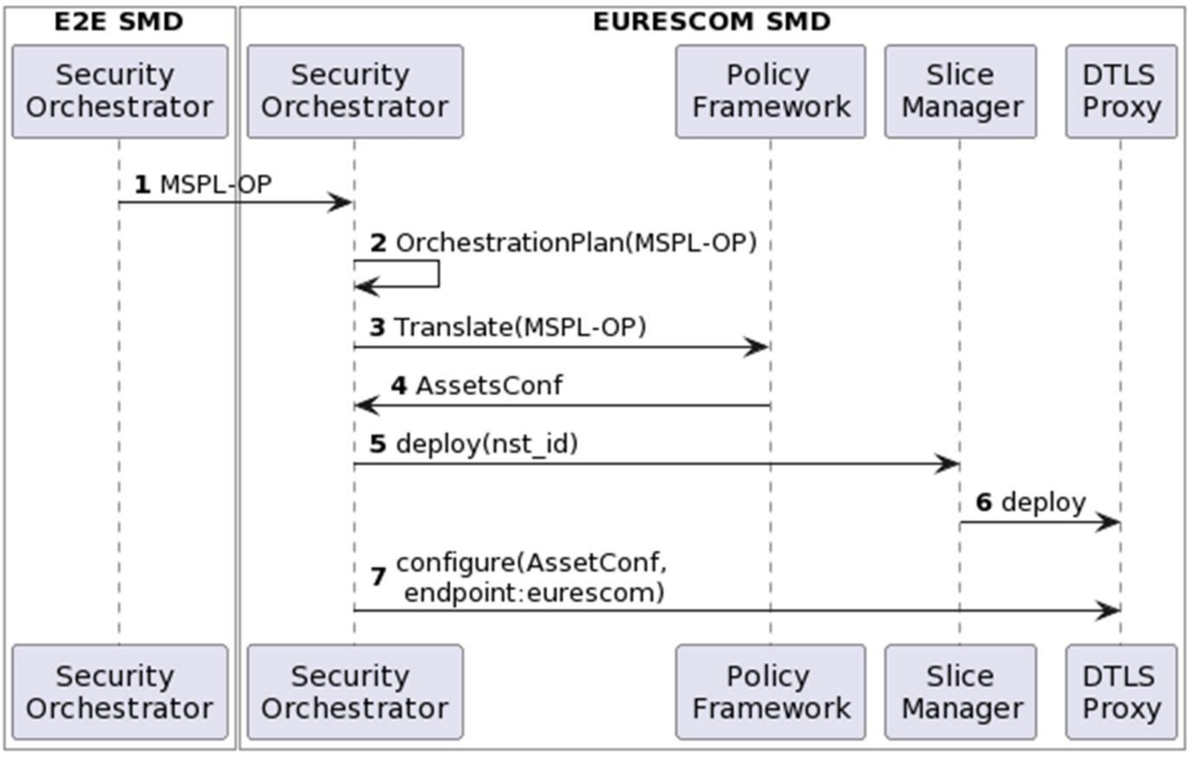


Figure 15: IoT Channel protection enforcement (EURESCOM/THALES side).

Figure 15 shows the detailed workflow for the IoT Channel protection enforcement part of the SSLA enforcement. It is composed by the following interactions:

1. Security orchestrator receives the MSPL-OP.
2. Security Orchestrator generates the orchestration plan.
3. Security Orchestrator requests MSPL-OP translations from the Policy framework for those enablers/assets selected according to the orchestration plan. In this case, IoT Broker and DTLS Proxy.
4. Security Orchestrator receives DTLS Proxy configurations.
5. Security Orchestrator requests the 5G slice deployment as part of the orchestration plan.
6. DTLS Proxy is deployed with the proper security configuration.
7. IoT Broker is deployed.

3.4.5 SSLA 2 IoT Network Slice Monitoring

In this demo, an IoT channel protection monitoring is performed as close as possible to the IoT service so the MMT tool (MI) will be deployed and configured to this purpose in the MI SMD.

The Network Slice deployment and SSLA enforcement for this case, follows the same procedure as illustrated previously in Figure 7.

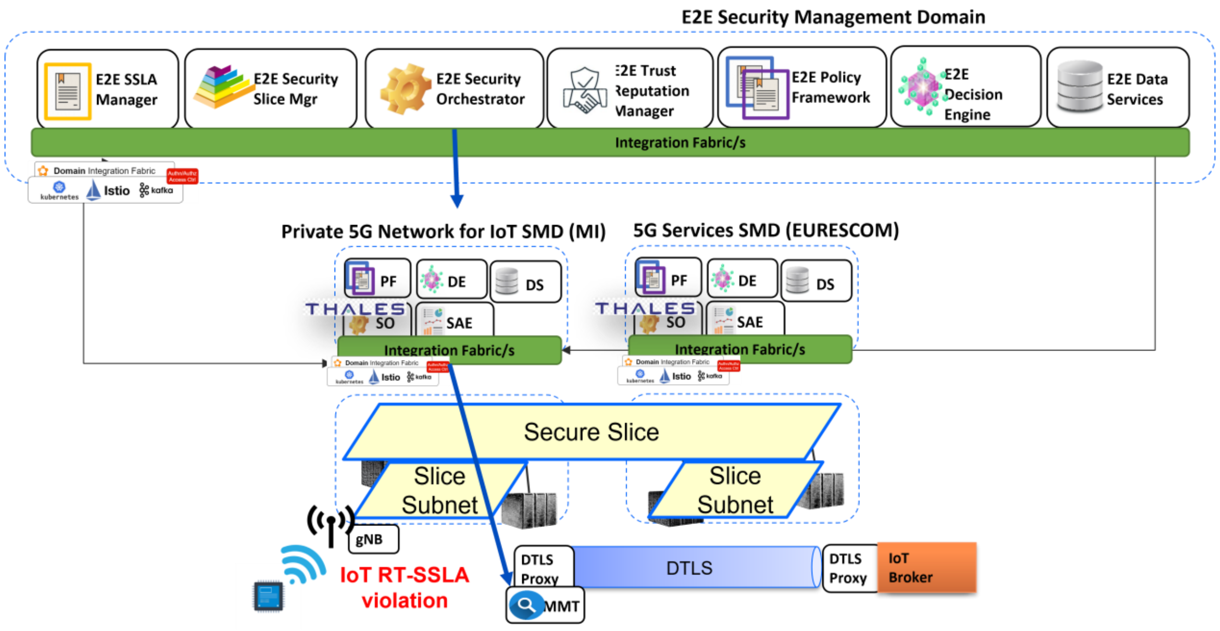


Figure 16: IoT Channel Protection monitoring enforcement.

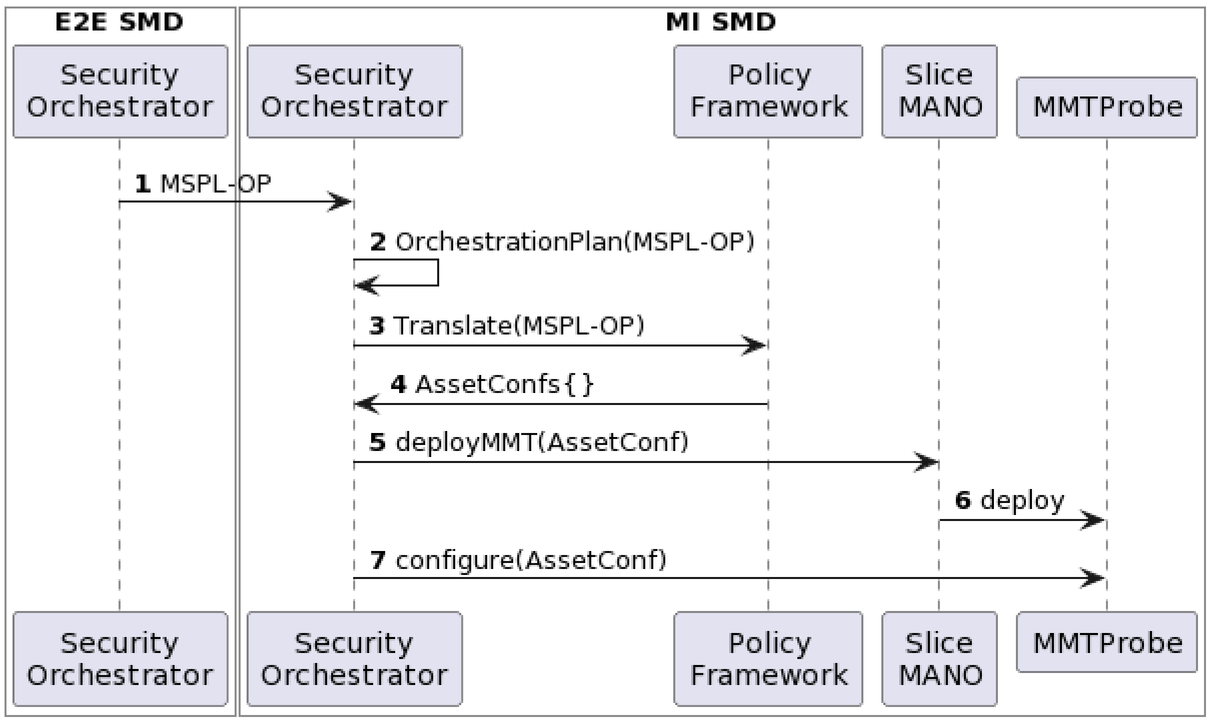


Figure 17: IoT Channel Protection monitoring workflow

Figure 17 shows the detailed workflow for this part of the SSLA enforcement. It is composed by the following interactions:

1. Security orchestrator receives the MSPL-OP.

2. Security Orchestrator generates an Orchestration plan.
3. Security Orchestrator requests MSPL-OP translations from the Policy framework for those enablers/assets selected according to the orchestration plan. In this case, MMT-Probe.
4. Security Orchestrator receives MMT-Probe configurations.
5. Security Orchestrator requests the MMT Probe deployment as part of the orchestration plan.
6. Security Orchestrator configures the MMT-Probe.

3.4.6 Closed loop Reactive Workflow

To show the reactive part of the close loop, different kinds of attacks will be simulated. Then, those monitoring assets that were previously deployed as part of the SSLAs enforcement will notify the issues to the SAE which will process them and will notify the DE that will generate new reactive security policies to close the loop locally (at SMD level), as well as notifying the E2E DE that will also generate new security policies to provide E2E countermeasures (if required). Following subsections detail the attack simulations, detections and reactions considered in this demo.

Cryptomining Attack & Reaction (Internal attack)

Once the 5G service (5G Core) has been deployed, it starts performing mining tasks since the 5G image was compromised[3][4][5][6]. Then, STA asset will detect the issue (Figure 18-step1) and it will notify the SAE. After verifying the attack, it will notify the DE (step2). At this point, TRM is also notified in order to update trust metrics of the involved entities (5G Core). The DE then generates a new security policy to redeploy the 5G Core locally in the SMD. Since the trust has been updated, another version of the 5G Core will be deployed as part of the reactive trust-based orchestration process (step3), closing the loop at SMD level. Finally, the reaction is propagated to the E2E DE that will orchestrate the same decision in the domains that also contain the compromised image (step4), closing the loop at E2E level. To do so, it will resubmit an updated MSPL-OP to the E2E SO which will orchestrate the update in the relevant SMD through the SMD orchestrator.

[3] https://www.theregister.com/2021/10/25/in_brief_security/

[4]https://latestgamestories.com/2022/01/10/the-developer-of-two-of-the-most-popular-open-source-nodejs-libraries-has-decided-to-corrupt-them-affecting-millions-of-users/

[5] https://www.bleepingcomputer.com/news/security/malicious-pypi-packages-hijack-dev-devices-to-mine-cryptocurrency/

[6] https://www.digitaltrends.com/computing/aws-monero-hack-45000-dollars/

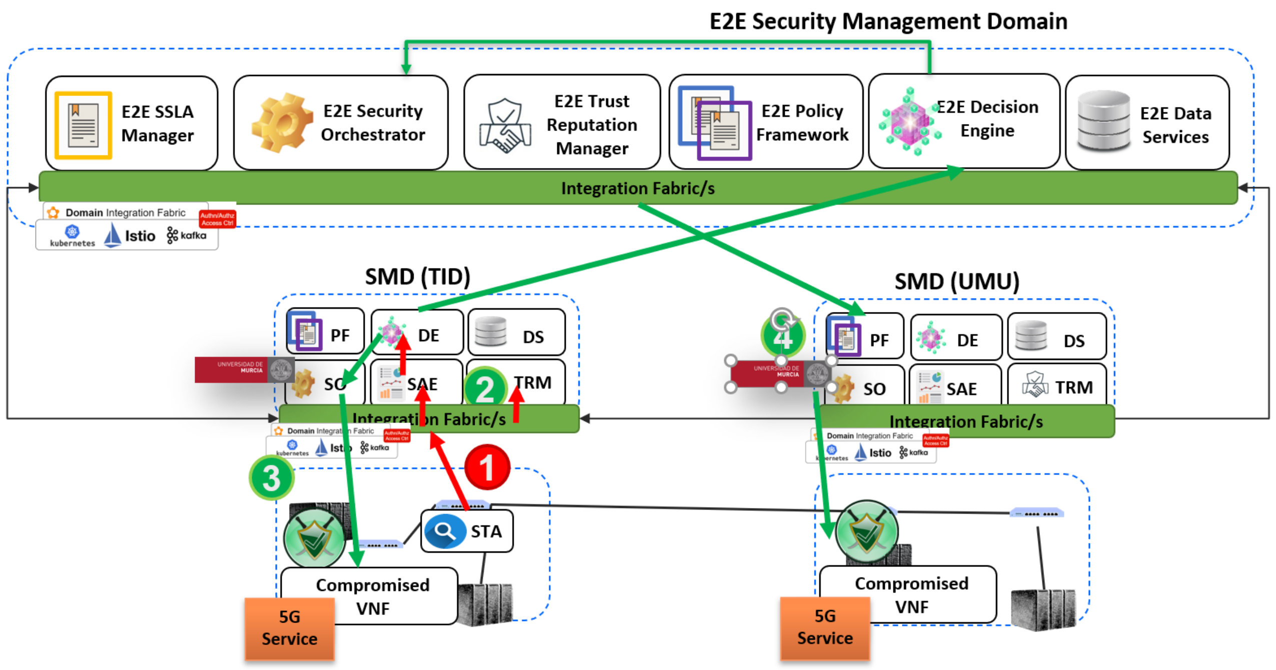


Figure 18: Cryptomining attack and reaction.

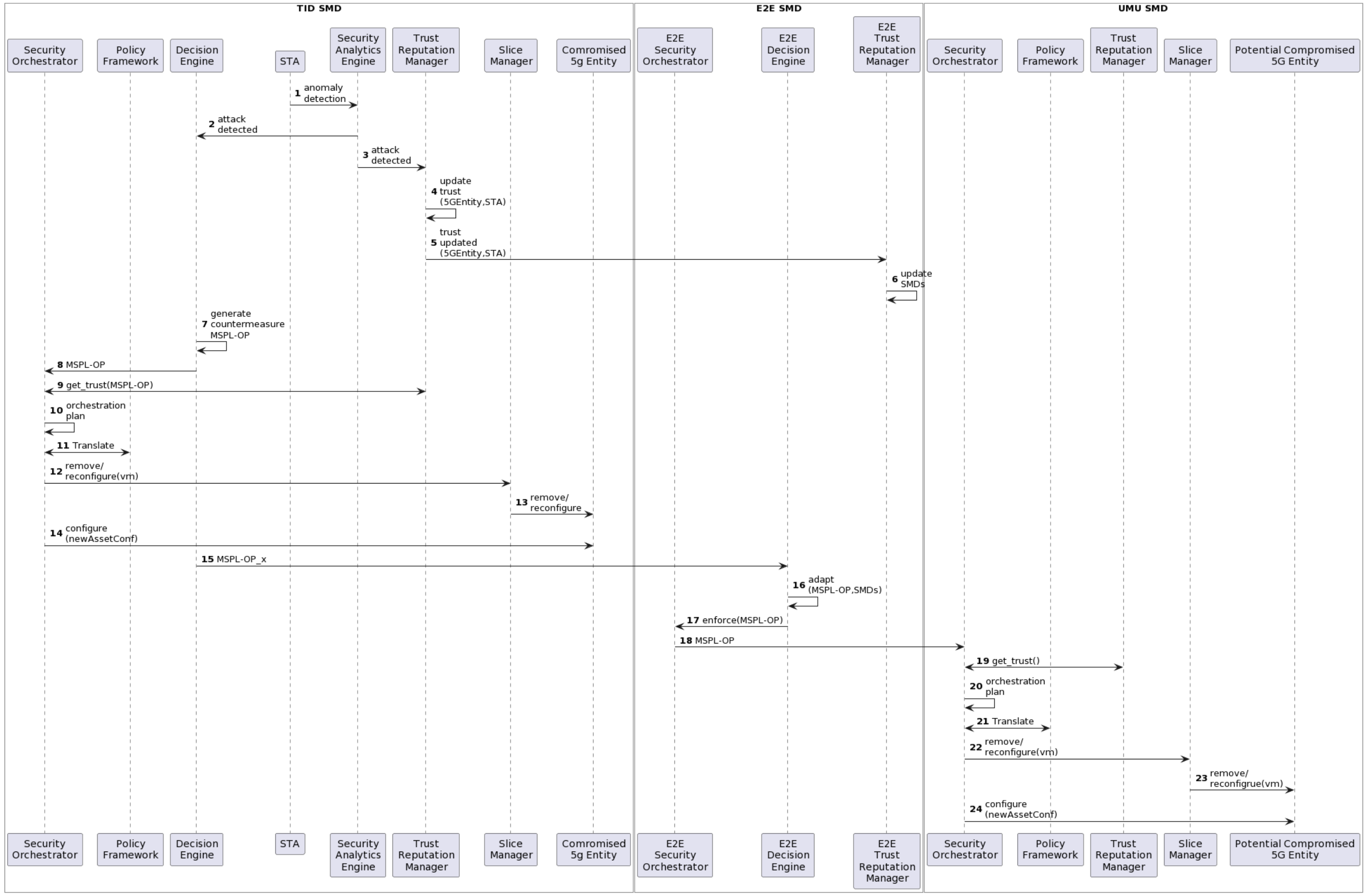


Figure 19: Cryptomining attack reaction workflow.

Figure 19 shows the detailed workflow for this part of the SSLA enforcement. It is composed by the following interactions:

1. STA detects the abnormal behaviour, triggers its internal SAE, that verifies that it is an attack.
2. The SAE sends the alert to the Decision Engine.
3. The SAE also sends the alert to the TRM.
4. TRM updates trust value of the compromised 5G service.
5. TRM notifies the update to the E2E TRM.
6. The E2E TRM computes the new domain trust score based on the information received and updates it in the DLT.
7. Decision Engine generates reactive security policies (redeploy/reconfigure the service).
8. Decision Engine sends the reactive security policies to the Security Orchestrator.
9. Security Orchestrator retrieves different kinds of information to prepare the orchestration plan, e.g., trust values.
10. Security Orchestrator generates the trust-based orchestration plan.
11. Security Orchestrator requests MSPL-OP translations from the Policy framework for those enablers/assets selected according to the orchestration plan. In this case, a more trustable version/configuration of the 5G service. Security Orchestrator receives 5G service configuration.
12. The Security Orchestrator requests to the VNF manager the deletion/reconfiguration of the VM with misbehaviour 5G Service.
13. The VNF Manager enforces the request on affected VM.
14. Security Orchestrator applies the new configuration on the 5G service.
15. Decision Engine notifies the reaction to the E2E Decision Engine.
16. E2E Decision Engine generates new reactive policies for other SMDs that could be affected.
17. E2E Decision Engine requests the E2E policy enforcement to the E2E Security Orchestrator.
18. E2E Security Orchestrator requests the security policy in the affected SMD in order to redeploy the affected VM.
19. Security Orchestrator retrieves different kinds of information to prepare the orchestration plan, e.g., trust values (these values were updated at step 5).
20. Security Orchestrator generates the trust-based orchestration plan.
21. Security Orchestrator requests MSPL-OP translations from the Policy framework for those enablers/assets selected according to the orchestration plan. In this case, a more trustable version/configuration of the 5G service. Security Orchestrator receives 5G service configuration.
22. Security Orchestrator requests to the VNF Manager the reconfiguration/deletion of the VM hosting the potential misbehaviour 5G service.
23. VNF Manager applies the requested operation on the 5G Service.
24. Finally, Security Orchestrator reconfigures the 5G service.

DDoS Attack & Reaction (External)

For the second attack, a DDoS attack simulation is performed involving different SMDs. At this point the DDoS monitoring tool that were previously deployed as part of the SSLA will detect the issue in different ways (Figure 20 - step1). In this case, malicious traffic will be filtered on CTTC 5G V2X Service SMD) (step2), as well as in another domain, UMU 5G RAN SMD, as E2E countermeasure (step3-5).

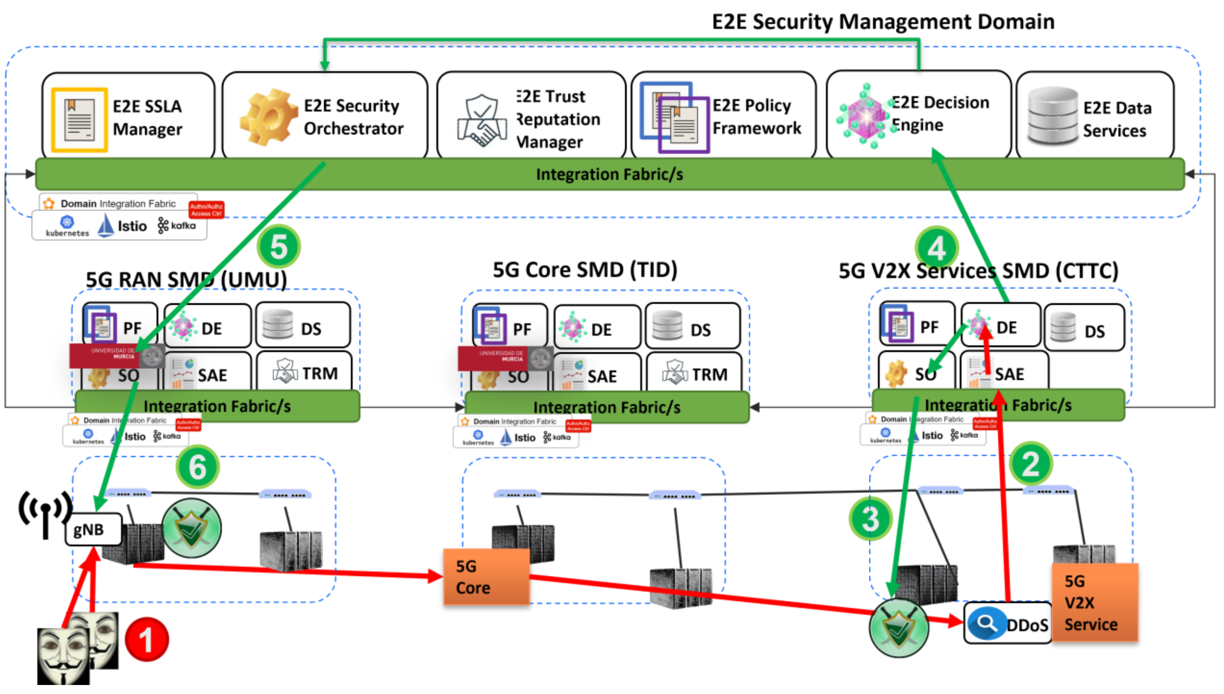


Figure 20: DDoS attack and reaction.

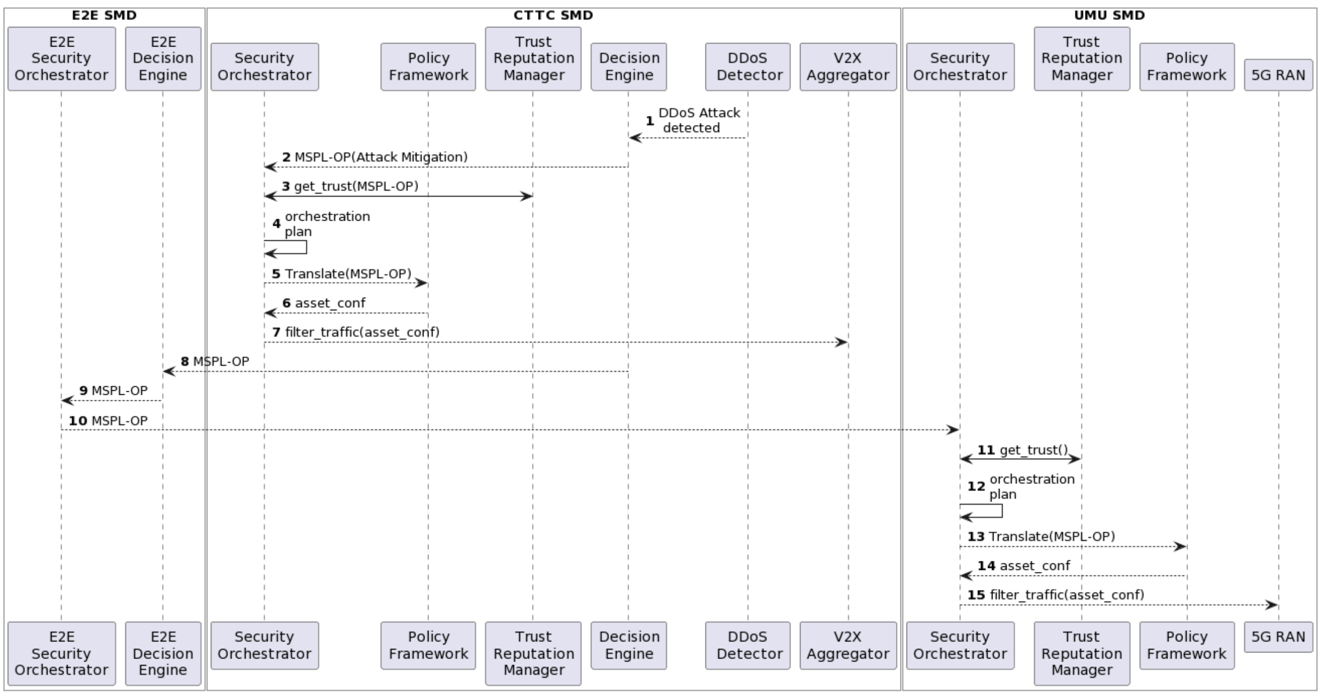


Figure 21: DDoS attack and reaction workflow.

Figure 21 shows the detailed workflow for this part of the SSLA enforcement. It is composed by the following interactions.

1. DDoS detector which also plays the role of Security Analytic Engine detects the **DDoS attack** and notifies it to the Decision Engine.
2. Decision Engine generates **reactive** security policies (**filtering**) and sends the reactive MSPL-OP to the Security Orchestrator.
3. Security Orchestrator retrieves different kinds of information to prepare the orchestration plan, e.g., trust values.
4. Security Orchestrator generates the orchestration plan.

5. Security Orchestrator requests MSPL-OP translations from the Policy framework for those enablers/assets selected according to the orchestration plan. In this case, new filtering configuration over a network filtering asset.
6. Security Orchestrator receives SDN filtering configuration.
7. Security Orchestrator enforces the reconfiguration of the network filtering asset.
8. Decision Engine notifies the reaction to the E2E Decision Engine.
9. E2E Decision Engine generates new reactive policies for other SMDs that could be affected and requests the E2E policy enforcement to the E2E Security Orchestrator.
10. E2E Security Orchestrator requests the security policy in the affected SMD in order to filter the malicious traffic.
11. Security Orchestrator retrieves different kinds of information to prepare the orchestration plan, e.g., trust values.
12. Security Orchestrator generates the trust-based orchestration plan.
13. Security Orchestrator requests MSPL-OP translations from the Policy framework for those enablers/assets selected according to the orchestration plan. In this case, an 5G RAN selected.
14. Security Orchestrator receives 5G RAN configuration.
15. Security Orchestrator reconfigures the 5G RAN to filter the malicious traffic.

IoT Channel Protection attack, RT-SSLA Violation & reaction (Internal)

Figure 22shows the internal attack on IoT Channel protection. It is simulated on MI's and EURESCOM's SMDs. In this scenario, the attack is performed internally from inside MI infrastructure to alter the VCP enabler, i.e., DTLS proxy VNF configuration, in order to downgrade - or modify in any way - the list of DTLS cipher suites used in DTLS encrypted traffic. The alteration of the DTLS proxy security configuration (VNF integrity violation), and resulting potential SSLA violation, is detected by the MMT probe (by analysing the DTLS handshake traffic). The violation will be notified to the SAE by sending an attack alert to the DE. In this case, the DE will react by generating a new security policy which indicates the need to re-deploy (migrate) the MMT Probe and a clean DTLS Proxy (VNF) instance on a new safe server.

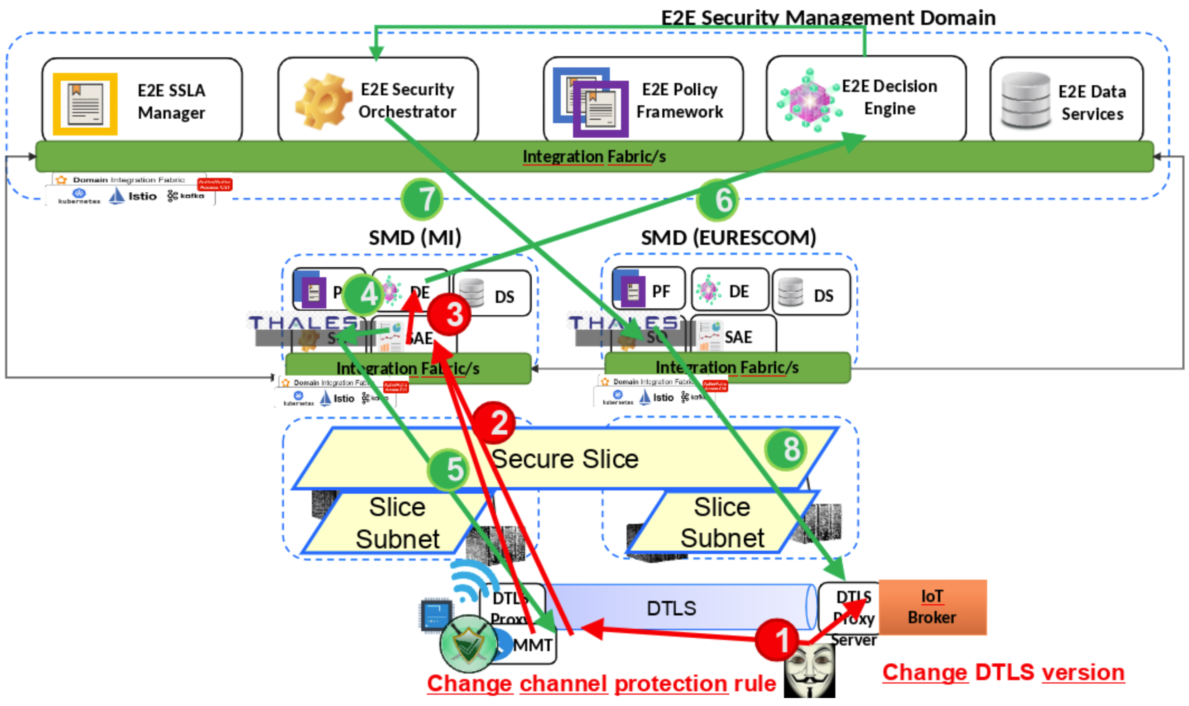


Figure 22: Internal 5G IoT integration attack and monitoring.

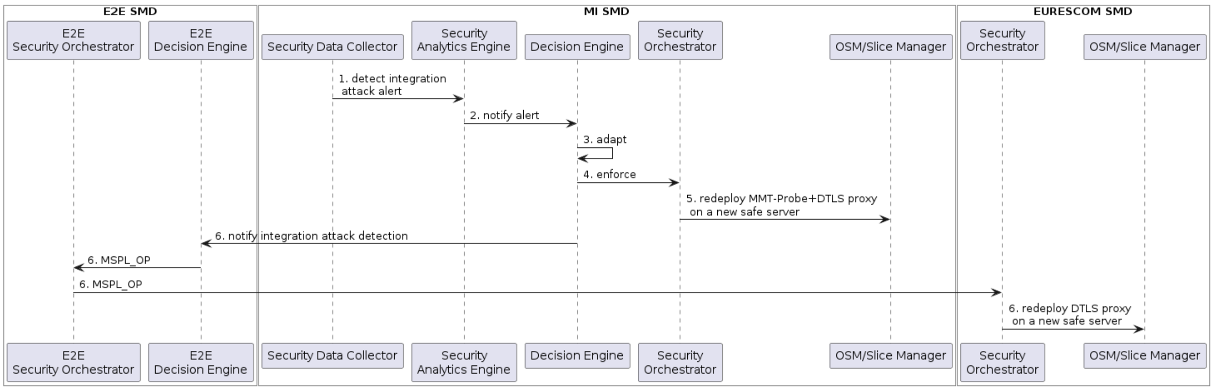

E2E SMD
MI SMD
EURESCOM SMD
E2E Security Orchestrator
E2E Decision Engine
Security Data Collector
Security Analytics Engine
Decision Engine
Security Orchestrator
OSM/Slice Manager
Security Orchestrator
OSM/Slice Manager
1. detect integration attack alert
2. notify alert
3. adapt
4. enforce
5. redeploy MMT-Probe+DTLS proxy on a new safe server
6. notify integration attack detection
6. MSPL_OP
6. MSPL_OP
6. redeploy DTLS proxy on a new safe server

Figure 23: Internal 5G IoT integration attack reaction workflow.

Figure 23 shows the detailed workflow of this part of the SSLA enforcement. It is composed by the following interactions:

1. The Security Data Collector (MMT Probe) detects an internal attack on the DTLS Proxy server, i.e., an unauthorized modification of the DTLS proxy (VNF)'s security configuration (change of cipher suites in DTLS handshake) and notifies the Security Analytic Engine (SAE) of a VNF integrity violation.
2. Security Analytic Engine verifies that it is an attack and sends the alert within the information of the current deployment (SSLA identification, host name) to the Decision Engine.
3. Decision Engine generates **reactive** security policies (migration of the VCP, DTLS proxy, and probe re-deployment).
4. Decision Engine sends the reactive security policies to the Security Orchestrator.
   The Security Orchestrator generates the orchestration plan. The Security Orchestrator requests MSPL-OP translations from the Policy framework for those enablers/assets selected according to the orchestration plan. In this case, it is a re-deployment of the security enablers (MMT Probe and VCP - DTLS Proxy VNF) on a new safe server.
5. The Security Orchestrator might require that the NFV-MANO redeploys (migrates) the compromised VNF (DTLS Proxy) VMs on a new safe machine, and the SDN controller to reconfigure the network to be able to re-deploy MMT-Probe and redirect traffic to the new VNF instance on the safe machine.
6. Decision Engine notifies the reaction to the E2E Decision Engine.
   E2E Decision Engine sends MSPL_OP to E2E Security Orchestrator. E2E Security Orchestrator requires (1) the Security Orchestrator on Eurescom SMD to deploy the DTLS proxy server on a new safe machine and (2) the SO on MI SMD to deploy the DTLS proxy and MMT-Probe on a new safe machine.

3.5 Validation of Demo 1 KPIs

3.5.1 Demo 1 KPIs

Table 5 provides details on the KPIs measured and validated in Demo 1. The following Sections provide the details on measurements and how the target values were calculated.

Demo 1 KPI	Description	Target Value
Initial Time (IT)	IT is calculated as the **time** elapsed from the deployment /enforcement request of a security asset to the moment in which requests performed to it are processed and not discarded	<5 minutes (E2E + 2 Domains)
Mean Time to Detect (MTTD)	MTTD is defined as the **average length of time** between the start of adversary acts and their discovery	<5 minutes <10ms (for MMT non-ML rules)
Number of False Positive (FP)	**Number** of FP is considered as a type of error for a binary classification, false alarms	<10 % <1 % (for MMT-Probe in a controlled experiment)
Number False Negative (FN)	**Number** of FN is considered the samples in a dataset that are not detected despite to be what is searching	<10 % <1 % (for MMT-Probe in a controlled experiment)
Security Service downtime (SSDT)	SDT measures the **percentage of time** a security service is not working or unavailable by the users	<10 % <0.001% (for MMT-Probe and a limited time of operation)
5G Service downtime (5GSDT)	SDT measures the **percentage of time** a 5G related service is not working or unavailable by the users	<10 %
Packet Loss Ratio (PLR)	PLR is defined as the **ratio of** the number of data **packets** lost to the total number of packets that should have been forwarded by a network node	<0.001% (for MMT-Probe and low bandwidth traffic)
Mean Time to Resolve (MTTR)	MTTR can be defined as the **mean/average time** to resolve a security incident	<10 minutes

Table 5: Demo 1 KPIs.

3.5.2 Deployment setup for measurements

Figure 24 shows a high-level diagram of the deployment setup used during demo1 KPI extraction. The deployment is composed of the E2E SMD (CTTC premises), 5G RAN SMD (UMU premises), 5G V2X Service (CTTC premises), Private 5G Network for IoT (MI premises) and 5G IoT Services (Eurescom).

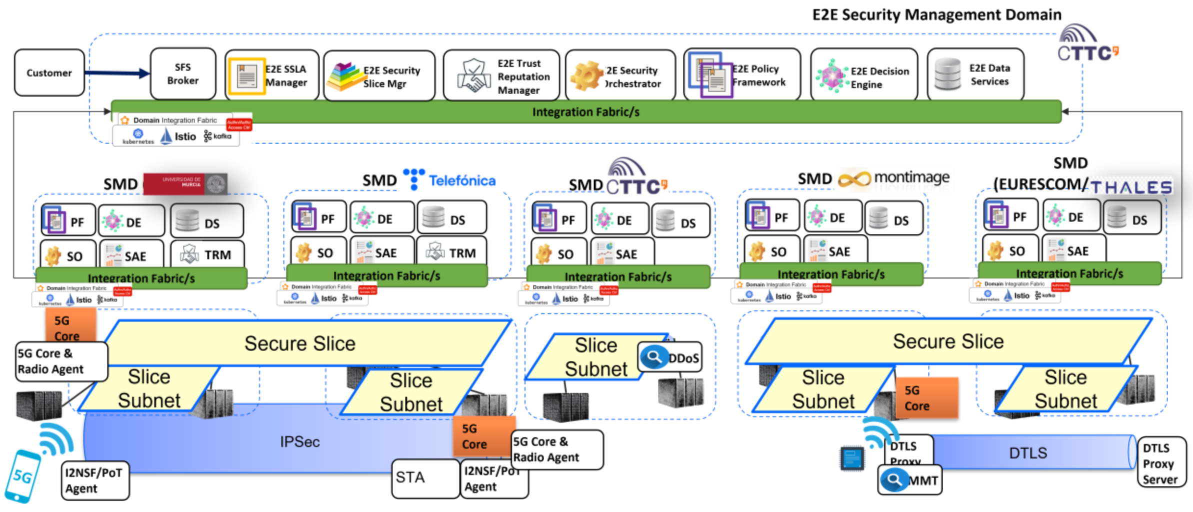


Figure 24: Deployment setup for Demo1 Initial Time KPI.

In this deployment, two different SSLAs have been enforced. The first one enforces a E2E 5G Security Slice that provides on-demand secured 5G Core and E2E 5G protected connectivity as well as protection against different attacks (i.e., cryptomining and DDoS). It involves E2E Security Management Domain (CTTC premises), 5G RAN SMD (UMU premises) and 5G V2X Service (CTTC premises). The second SSLA enforces a E2E 5G IoT Security Slice that provides on-demand secured 5G IoT service and E2E 5G IoT protected connectivity as well as protection against different attacks (i.e., Channel protection cipher-suite modification). It involves E2E Security Management Domain (CTTC premises), Private 5G Network for IoT (MI premises) and 5G IoT Services (Eurescom).

3.5.3 Results of Demo 1 KPIs

Below are the different results obtained for the different SSLAs involved in Demo 1. Each SSLA section provides the involved KPI results as well as detailed descriptions, mainly organised in three blocks; definition, methodology and results.

SSLA1

Initial Time (IT)

To measure the results of Demo 1 KPIs, we firstly focus on **Initial Time (IT)**, which is calculated as the elapsed time from the receiving of the E2E service and security requirements until they are successfully deployed, including: subslices, services, connectivity and security assets (proactive part). The measurements have been taken from the SFS Broker. Let **T0** be the registered timestamp when SFS Broker receives the E2E service and security requirements. Let **T1** be the registered timestamp when the whole 5G Security Slice has been deployed. This is, all the involved domains have deployed and configured their correspondent security slice, including communication. Then, **IT** can be calculated as the difference between **T0** and **T1**; that is, **IT = T1 – T0**.

This KPI will be measured by terms of the enforcement of E2E service and security requirements involving 4 SMDs (E2E, TID, UMU, CTTC) where the time will be evaluated from the reception of the requirements by the SFS Broker to the successful deployment and configuration in each of the SMDs involved. In particular, in the TID SMD, the deployment of a 5G core and the deployment and configuration of the I2NSF Agent and PoT will be performed. In the UMU SMD, the deployment and configuration of the I2NSF Agent and PoT will be performed. Finally, in the CTTC SMD, the deployment and configuration of the CTTC V2X DDoS monitoring tool will be performed.

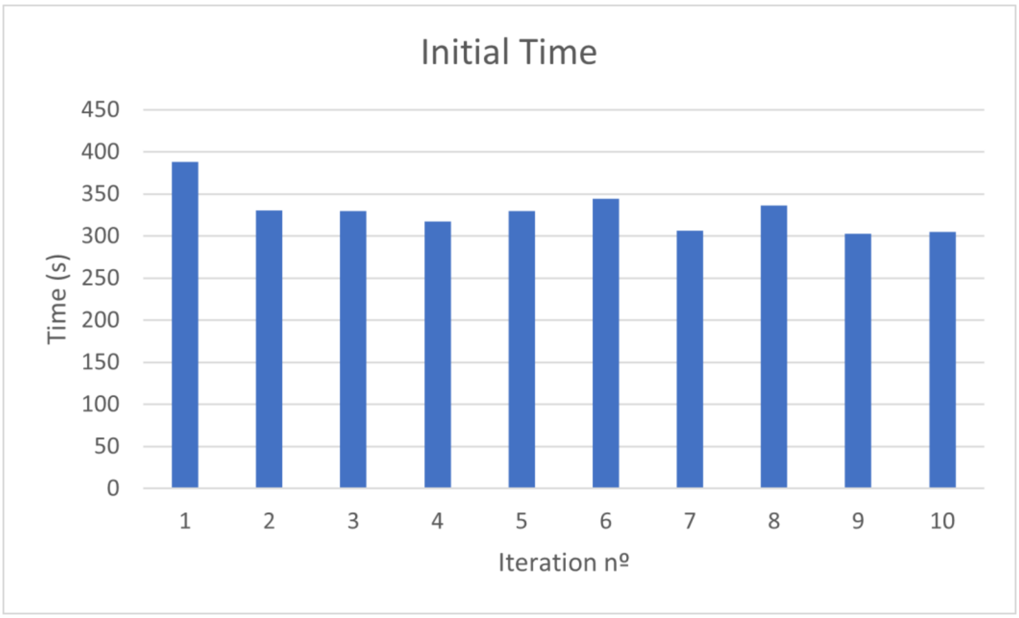


Figure 25: 5G Core Security Slice Initial Time.

Figure 25 shows the obtained values of 10 different executions. The x axis represents the execution iteration, while the y axis represents the IT value in seconds.

Iteration	T0 (timestamp)	T1 (timestamp)	IT (sec)
1	1657879316853	1657879316465	388.357
2	1657880168444	1657880168114	330.194
3	1657881876420	1657882206122	329.702
4	1657882401378	1657882718457	317.079
5	1657882878902	1657883208894	329.992
6	1658911821416	1658912165912	344.4965
7	1658912921123	1658913227579	306.456
8	1658914699593	1658915036264	336.671
9	16589115736489	1658916038957	302.468
10	1658918855018	1658919159648	304.630

Table 6: Initial Time KPIs for SSLA1.

Table 6 shows the following information:

- Iteration represents the number of executions (n)
- T0 represents the initial timestamp in Unix Epoch format
- T1 represents the final timestamp in Unix Epoch format
- IT represents de difference in seconds between T0 and T1 (depicted in Figure 25)

$$\text{IT} = \frac{\sum_0^n (t1-t0)}{n} = 329.0045 \text{ sec} \sim \mathbf{5.48\ min}$$

While providing a 5G core, channel protection and proof of transit across E2E SMD and two SMDs took around 3 minutes, the inclusion of different monitoring features, and adding also a third SMD (V2X SMD), now slightly exceeds the KPI target value. This makes sense, since the more enforcements and configurations are required, the more time it will take.

It is important to highlight that all the domains involved in this enforcement use Virtual Machines for deploying slice VNFs, whose time to deploy is significantly bigger than using containers. Moreover, current dependencies/priorities system uses a sequential approach, this is, SMD enforcements are performed sequentially in a specific order, according to the established priorities, and configuration tasks are also queued until slice components are properly deployed. Thus, the results could be further improved by introducing parallelism in the dependency system to perform parallel deployment operations on each SMD and sequential configuration based on the dependency and priority system.

Mean Time to Detect (MTTD)

SSLA1 enforcement and reaction involves three different detector enablers/assets. PoT (Trust), STA (Behaviour) and V2X detector (DDoS). The measurement methodologies and results for each of them are provided below.

Proof of Transit

- Definition**:** Average time that the PoT Agents needs to detect and validate a PoT packet that has crossed between the agents of the path. This detection applies to valid and invalid packets.

- Methodology: Calculation has been taken in the PoT controller using the scenario of Demo1 between TID and UMU. The measures calculated by the controller are taken from the time difference between when the packet exits the first agent until it is validated after arriving to the second agent. To take different measurements, the interval between two packets have been reduced for each episode. Results**:**

Episode #	Interval (s)	Mean Time to Detect (ms)
1	10	33.64
2	5	33.67
3	1	34.95
4	0.5	34.38
5	0.1	36.23
6	0.05	34.03
7	0.01	33.83
8	0.005	34.65
9	0.001	402.39

Table 7: Mean time to detect the validation of the path by the PoT for a number of episodes.

The values reported in Table 7 represents the current performance of the PoT enabler implementation. From the results it can be extracted that in average it takes **75.31 ms** to make the verification of the path. By using smaller intervals and incrementing the bandwidth consumed in the PoT path, the PoT implementation cannot keep it up with the traffic, increasing the detection time.

STA

- Definition: Average time that takes the STA to detect a cryptomining flow.

Methodology: The calculation has been taken at the output of the STA (alert to the decision engine). For every flow identified as cryptomining traffic, it has been taken the difference from the first time when the STA receives the network flow until it has classified it. The measurement has been made with multiple iterations over pre-recorded traffic captures containing legit and cryptomining traffic where for each flow detected as cryptomining traffic, the time difference has been calculated, since the first packet of the flow was registered until it was classified. The pre-recorded captures have been generated in the Mouseworld lab.

- Results:

Episode #	Mean Time to Detect (ms)
1	81.71
2	97.87
3	90.47
4	90.61
5	85.24
6	75.45
7	93.82
8	84.44
9	85.32
10	85.85

Table 8: Mean time to detect a cryptomining attack by the STA for a number of episodes.

The values reported in Table 8, represent the current capabilities of the STA implementation when detecting cryptomining ciphered traffic. In average, it takes a total of **87.07 ms** to classify a traffic as cryptomining traffic which is considered acceptable in this scenario.

V2X Detector

Definition: Mean time to detect is defined as the average time elapsed between the time the DDoS attack takes place and its discovery by the V2X detector.

Methodology: The calculation of the time to detect a DDoS attack first considers the amount of time elapsed between the arrival of a malicious V2X traffic trace at the V2X service domain until its final detection at the output of the decision engine functional component. To ensure statistical validity, this calculation is repeated for all DDoS attacks present in the dataset, and the mean value is then calculated by dividing the overall (sum) of the time elapsed for all DDoS attacks by the number of DDoS incidents. The mean time to detect performance is assessed for a number of training episodes.

Episode #	Mean Time to Detect (ms)
1	4.13
2	3.95
3	3.96
4	3.92
5	3.91
6	3.92
7	3.98
8	3.89

9	3.87
10	3.94

Table 9: Mean time to detect a DDoS attack by the V2X detector for a number of training episodes.

The values reported in Table 9 corroborate the real-time capabilities of our V2X detector with respect to the time taken to detect misbehaviours. Detection latency is in the order of **4 ms**, which is considered acceptable for many road safety applications, as periodic beacons usually broadcast with frequency of 1-10 Hz.

Number of False Positive (FP)

The measurements, methodologies, and results for false positives of each involved detector are provided below.

STA

- Definition: In the context of the STA enabler, false positives are defined as the incorrect identification of legitimate ciphered traffic that has been identified as cryptomining traffic.

Methodology: The calculation has been taken at the output of the STA enabler, whereby replaying different network captures containing cryptomining and 5G traffic. With the results of the STA, it has been measured the number of incorrect decisions taken, by previously knowing the malicious IPs involved in the attack.

- Then the number of False Positives is divided by the total number of classifications performed by the STA. This process has been repeated in each iteration.

- Results:

Episode #	False positives (%)
1	0.000
2	0.000
3	0.000
4	0.063
5	0.000
6	0.060
7	0.000
8	0.000
9	0.062
10	0.031

Table 10: False positive rate performance of the STA for a number of episodes.

The values reported in Table 10, represent the percentage of false positives obtained from the STA classification output. In average, the percentage of false positives is **0.022%** which makes STA a good estimator, since there is a very low probability in detecting a trusted flow as cryptomining traffic.

V2X Detector

Definition: In the context of DDoS attack detection, false positive is defined as the incorrect identification of a legitimate vehicular behaviour as a malicious behaviour by the V2X detector.

Methodology: The calculation of the false positive rate takes place at the output of the decision engine functional component, where the number of false misbehaviours (equivalent to the concept of false alarms) is divided by the total number of identified misbehaviours. For the calculation, the ground truth information is considered to be available, and for each incorrect DDoS detection, the counter increases. The false positive rate performance is incrementally assessed for a number of training episodes.

Episode #	False positives (%)
1	4.3
2	4.4
3	7.1
4	12.4
5	10.8
6	6.4
7	8.6
8	8.7
9	8.5
10	4.1

Table 11: False positive rate performance of the V2X detector for a number of training episodes.

The values reported in Table 11 demonstrate the low false positive rates achieved by the V2X detector which, in the majority of the training episodes, lie below the threshold of 10%. The average is **7.53%**. It is noted that, during training, the V2X detector is penalized more for false negative actions than for false positives, as the correct identification of misbehaviour is necessary to avoid hazardous situations. Therefore, in an effort to keep false negative outcomes in low values (see next section), false positives are tolerated to an extent that is not excessive.

<u>Number False Negative (FN)</u>

The measurement, methodologies and results for false negatives of each involved detector are provided below.

<u>STA</u>

- Definition: In the context of the STA enabler, false negatives are defined as the incorrect identification of cryptomining traffic that has been identify as legitimate traffic.
- Methodology: The calculation has been taken at the output of the STA enabler, whereby replaying different network captures containing cryptomining and 5G traffic, it has been counted the number of incorrect decisions taken by the STA when making the classifications. Then the number of False Negatives is divided by the total number of classifications performed by the STA. This process has been repeated in each iteration.
- Results:

Episode #	False negatives (%)
1	0
2	0.00
3	0.13
4	0.00

5	0.00
6	0.00
7	0.00
8	0.00
9	0.00
10	0.00

Table 12: False negative rate performance of the STA for a number of episodes.

The values reported in Table 12 represent the percentage of false negatives obtained from the STA classification output. In average the percentage of false positives is **0. 013%**. This is a good estimator since the chances of not detecting a malicious cryptomining traffic are very low.

V2X Detector

Definition: In the context of DDoS attack detection, false negative is defined as the incorrect identification of a malicious behaviour as a legitimate vehicular behaviour by the V2X detector.

Methodology: The calculation of the false negative rate takes place at the output of the decision engine functional component, where the number of missed misbehaviours is divided to the total number of existing misbehaviours. For the calculation, the ground truth information is considered to be available, and for each missed DDoS detection, the counter increases. The false negative rate performance is incrementally assessed for a number of training episodes.

Episode #	False negatives (%)
1	0.5
2	0.4
3	0.4
4	0.3
5	0.2
6	0.8
7	0.4
8	0.1
9	0.4
10	0.5

Table 13: False negative rate performance of the V2X detector for a number of training episodes.

The values reported in Table 13 demonstrate the low false negative rates achieved by the V2X detector which lie below the threshold of 10%. The average is **0.4%**. As mentioned in the previous Section, the V2X detector is trained to be penalized more for false negatives than for false positives, allowing for tolerating false positives to an extent that is not excessive. This is due to the fact that in safety-critical V2X scenarios, the correct identification of DDoS attacks is required to mitigate potential hazardous situations; thus, false negative decisions may be more perilous than false positives.

5G Service downtime (5GSDT)

Definition: In the context of SSLA1, 5G Service downtime is defined as the time required to redeploy a most trusted version of the AMF 5G Service (5G service part of the 5G core).

Methodology: To calculate the AMF service downtime it was measured the elapsed time since the compromised AMF is destroyed until the new AMF is fully ready. This is, it has been deployed and properly registered in the 5G Core.

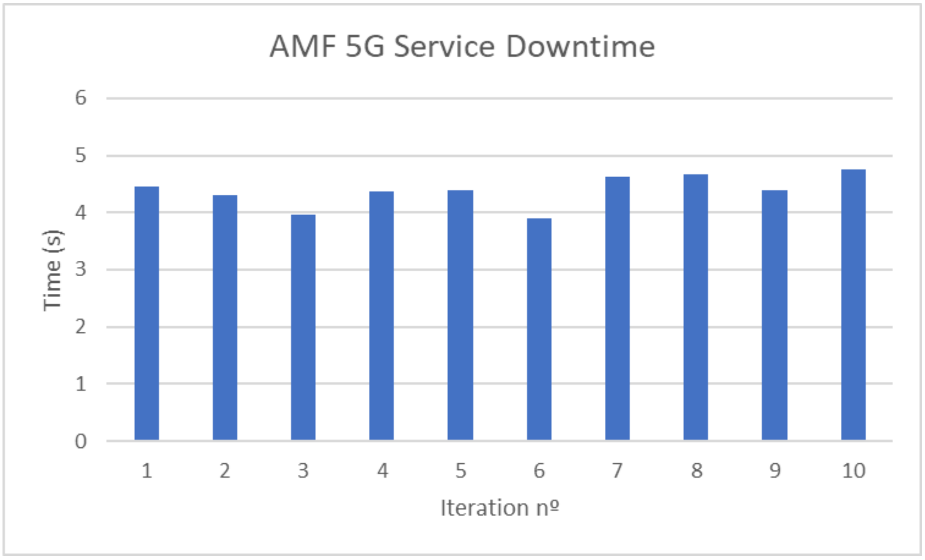


Figure 26: 5G Service downtime measurements over 10 iterations.

Figure 26 shows the obtained values of 10 different executions. The x axis represents the execution iteration, while the y axis represents the Initial Time value in seconds.

Iteration	T0 (timestamp)	T1 (timestamp)	5GSD (sec)
1	1657209995.3651032	1657209999.8092415	4.444138288497925
2	1657210243.4373217	1657210247.7388437	4.3015220165252686
3	1657210359.6538785	1657210363.620423	3.9665446281433105
4	1657210420.5467248	1657210424.9124882	4.365763425827026
5	1657210526.9850607	1657210531.371002	4.38594126701355
6	1657210593.5488973	1657210597.4415584	3.8926610946655273
7	1657210838.04373	1657210842.6701252	4.626395225524902
8	1657211172.1210425	1657211176.7898216	4.668779134750366
9	1657211324.2588537	1657211328.648984	4.390130281448364
10	1657211771.0887496	1657211775.8520083	4.763258695602417

Table 14: 5G Service downtime from compromised AMF removal (T0) until new AMF is fully deployed (T1).

Table 14 shows the obtained measurements over the different 10 iterations. Iteration column represents the number of executions (n). T0 represents the initial timestamp in Unix Epoch format. T1 represents the final timestamp in Unix Epoch format. 5GSDT represents de difference in seconds between T0 and T1.

$$\text{5GSDT} = \frac{\sum_1^n (T1 - T0)}{n} = \textbf{4,38 sec}$$

Assuming a monitoring time (MT) of 24h (86,400 secs) and a single attack since once the trust has been updated and the countermeasure has been enforced, the AMF is not vulnerable to the same threat, the results provided are aligned to the target KPI provided (less than 10%).

$$5GSDT\% = \frac{MT}{5GSDT} \times 100 = \mathbf{0.005\%}$$

Mean Time to Resolve (MTTR)

Since demo1 SSLA1 part involves different attacks, this section provides Mean Time to Resolve KPI results for each of them. Assuming a single attack, since once the trust has been updated and the countermeasure has been enforced, the system is not vulnerable anymore to the same threat, MTTR for each specific attack is measured as the total duration of the operational time for the security service (Total Security Maintenance Time - TSMT). In inspire this measure corresponds to the automated reactive part of the closed loop.

$$MTTR = \frac{Total\ Security\ Maintenance\ Time}{Total\ number\ of\ incidents} = \frac{TSMT}{1} = \boldsymbol{TSMT}$$

Cryptomining Attack

Definition: In the context of SSLA1, local and E2E countermeasures are generated and enforced to mitigate the cryptomining attack.

Methodology: To calculate the local MTTR, the elapsed time was measured from the Decision Engine deployed in TID SMD receiving the cryptomining alert until the countermeasure has been enforced in the SMD. Moreover, E2E MTTR to provide E2E mitigation to the same attack has been also provided.

SMD MTTR

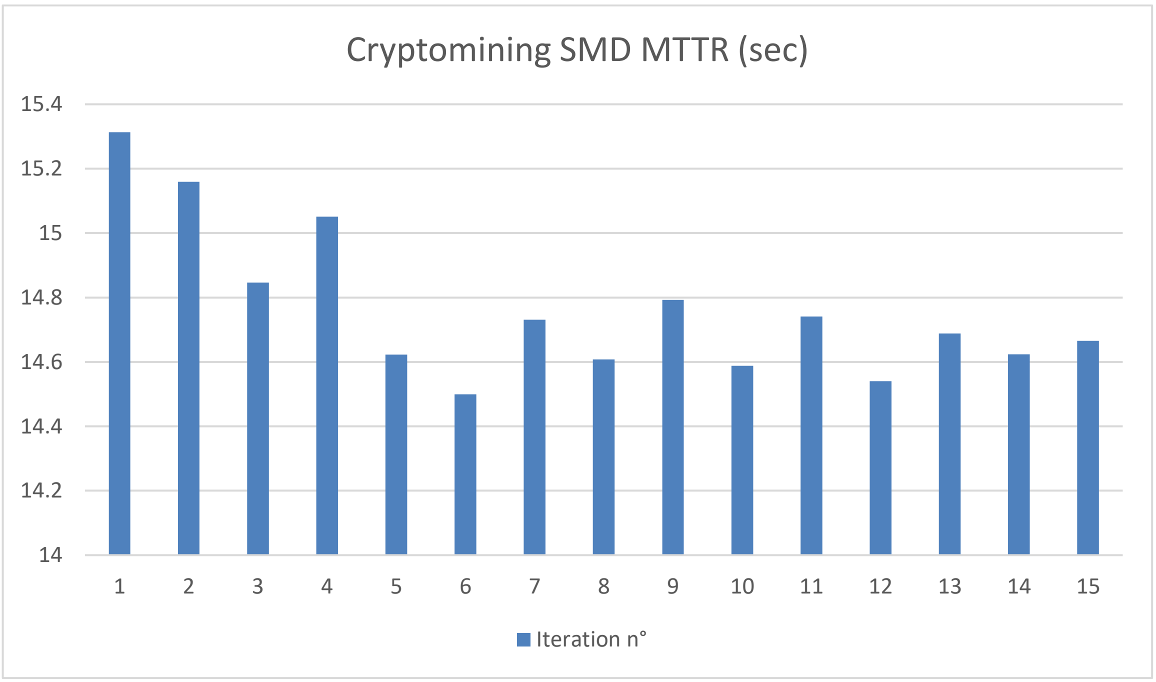


Figure 27: The SMD MTTR of a cryptomining attack.

Figure 27 shows the histogram of the MTTR taken to stop a cryptomining attack in the local TID SMD. The horizontal axis displays the 15 iterations. The vertical axis is the time taken in second from the moment the TID Decision Engine (implemeneted by Thales PyrDe) receives a cryptomining alert to the moment the TID SO responded with a success code implying a mitigation deployment. The raw inputs are reported in Table 18.

Iteration	T0 (timestamp)	T1 (timestamp)	MTTR (sec)
1	1659024138.68449	1659024153.99793	15.313431978

2	1659024376.87756	1659024392.03687	15.159310102
3	1659024466.69596	1659024481.54183	14.845868111
4	1659024553.32936	1659024568.38055	15.051192999
5	1659024643.62788	1659024658.25108	14.623208046
6	1659024722.14425	1659024736.64379	14.499532938
7	1659024844.75917	1659024859.49062	14.731458902
8	1659024916.92308	1659024931.53082	14.607744932
9	1659024980.1988	1659024994.99143	14.792624950
10	1659025043.65082	1659025058.23913	14.588315964
11	1659025112.83434	1659025127.57553	14.741188049
12	1659025185.05466	1659025199.59507	14.540409088
13	1659025245.90651	1659025260.59487	14.688359976
14	1659025427.61679	1659025442.24039	14.623602152
15	1659025512.27601	1659025526.94189	14.665886879

Table 15: Mean Time to Resolve to trigger a mitigation from an cryptomining alert TID SAE -> TID PyrDE.

Table 15 shows the measures made inside the TID SMD when the TID Security Analytics Engine (SAE) forwarded a cryptomining alert to the local TID PyrDE. The T0 column is the timestamps of the operation start time, when the TID PyrDE received the alert in its API endpoint. The T1 column is the timestamps of the operation end time, after the TID PyrDE generated a mitigation MSPL and submitted it to the TID Security Orchestrator. The MTTR column is the delta time in seconds of the 2 previous columns, displaying the elapsed time.

$$\text{SMD MTTR} = \frac{\sum_1^n (T1-T0)}{n} = 14.764809004 \approx \boldsymbol{14.76\ sec}$$

According to the results, cryptomining attack is mitigated in **less than 15 seconds**, thus, we consider that the solution is promising since the KPI target value was less than 10 minutes.

E2E-SMD MTTR

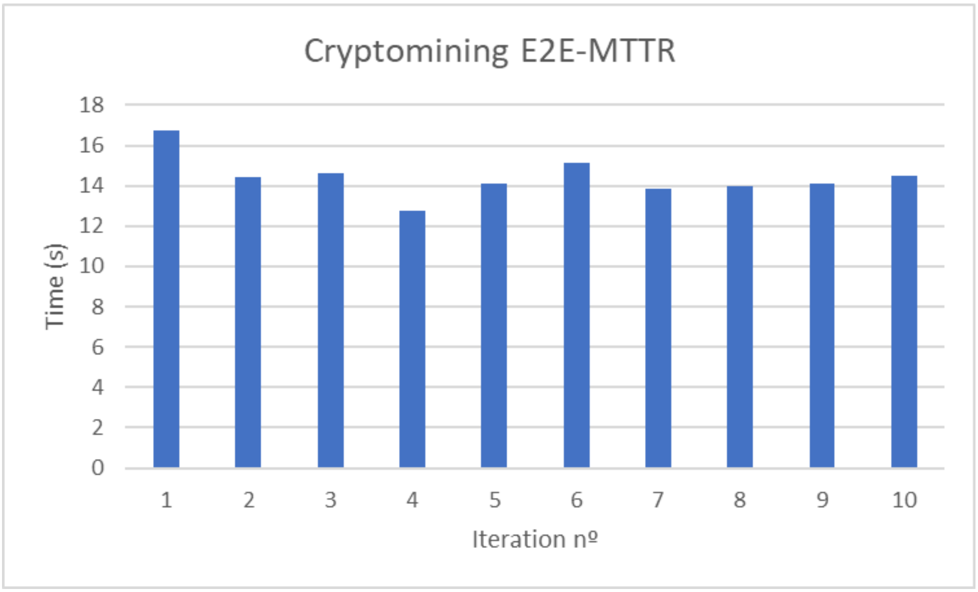


Figure 28: The E2E SMD MTTR of a cryptomining attack.

Figure 28 shows the obtained values of 10 different executions. The x axis represents the execution iteration, while the y axis represents the Initial Time value in seconds.

Iteration	T0 (timestamp)	T1 (timestamp)	MTTR (sec)
1	1658738892.15369	1658738908.87469	16.7210013
2	1658739070.11787	1658739084.55689	14.439022158
3	1658739360.32268	1658739374.91249	14.5898127156
4	1658739460.33463	1658739473.11056	12.7759294656
5	1658739586.21044	1658739600.31644	14.1059957784
6	1658739857.19746	1658739872.32086	15.1234035636
7	1658739996.80151	1658740010.66604	13.864523988
8	1658740069.39718	1658740083.39478	13.9975966572
9	1658740138.13727	1658740152.26572	14.1284530536
10	1658740209.56896	1658740224.05845	14.4894916776

Table 16: Mean Time to Resolve SSLA1 E2E PyrDE -> E2E SecOrch (cryptomining, Trust-based redeployment)

Table 16 shows the obtained measurements over the different 10 iterations for the E2E MTTR KPI. Iteration column represents the number of executions (n). T0 represents the initial timestamp in Unix Epoch format. T1 represents the final timestamp in Unix Epoch format. The MTTR column represents the difference in seconds between T0 and T1.

$$\text{E2E-MTTR} = \frac{\sum_1^n (T1-T0)}{n} = \mathbf{14.42\ sec}$$

According to the results, cryptomining attack is also mitigated in less than 15 seconds, thus, we consider that the solution is promising since the KPI target value was less than 10 minutes. In this case, SMD and E2E mitigations are triggered in parallel and the reactive loop used is the same in both domains so SMD and E2E mitigations provide similar results. Besides, even if SDM and E2E reactions are triggered sequentially the result would round 30 seconds, which is still less than the target KPI value.

DDoS Attack

SMD MTTR

Definition: Mean time to resolve is defined as the average time elapsed between the time the DDoS attack is detected and the enforcement of the mitigation (filtering) policy by the security orchestrator.

Methodology: The calculation of the time to resolve a DDoS attack first considers the amount of time elapsed between the detection of a malicious V2X traffic trace at the V2X service domain until the time the security orchestrator confirms the policy enforcement to trigger remediation actions (filtering) against the malicious traffic. To ensure statistical validity, this calculation is repeated for all DDoS attacks present in the dataset, and the mean value is then calculated by dividing the overall (sum) of the time elapsed for all DDoS attacks by the number of DDoS incidents. The mean time to resolve performance is assessed for a number of training episodes.

Episode #	Timestamp T0	Timestamp T1	Mean Time to Resolve (s)
1	1658492526.0831	1658492533.10376	7.02066
2	1658492536.18289	1658492537.60842	1.42553
3	1658492540.70801	1658492542.17265	1.46464
4	1658492545.22814	1658492546.70427	1.47613

5	1658492549.76289	1658492551.19669	1.4338
6	1658492554.25099	1658492555.84474	1.59375
7	1658492558.89493	1658492560.28269	1.38776
8	1658492563.36715	1658492564.79160	1.42445
9	1658492567.86311	1658492569.47642	1.61331

Table 17: Mean time to resolve a DDoS attack for a number of training episodes

The values reported in Table 17 corroborate the real-time capabilities of our V2X detector with respect to the time taken to resolve misbehaviours. MTTR is kept in the order of few seconds, which is considered acceptable for mitigating detrimental effects on road users and avoiding the propagation of safety-threatening incorrect information by misbehaving vehicles. According to the results, DDoS attack is mitigated at SMD level in less than **2 seconds**, accomplishing the provided KPI target value (less than 10 minutes).

E2E-SMD MTTR

Definition: In the context of SSLA1, the DDoS attack mitigation is defined as the time taken to stop an ongoing DDoS attack by enforcing network rules to block the attacker from the E2E level.

Methodology: To calculate the time to emit and apply the mitigation, a measurement was made from the moment the E2E Decision Engine received an alert containing a notification of a successful DDoS detection to the moment the E2E DE sent a MSPL manifest with the selected mitigation to the E2E Security Orchestrator and received a success code.

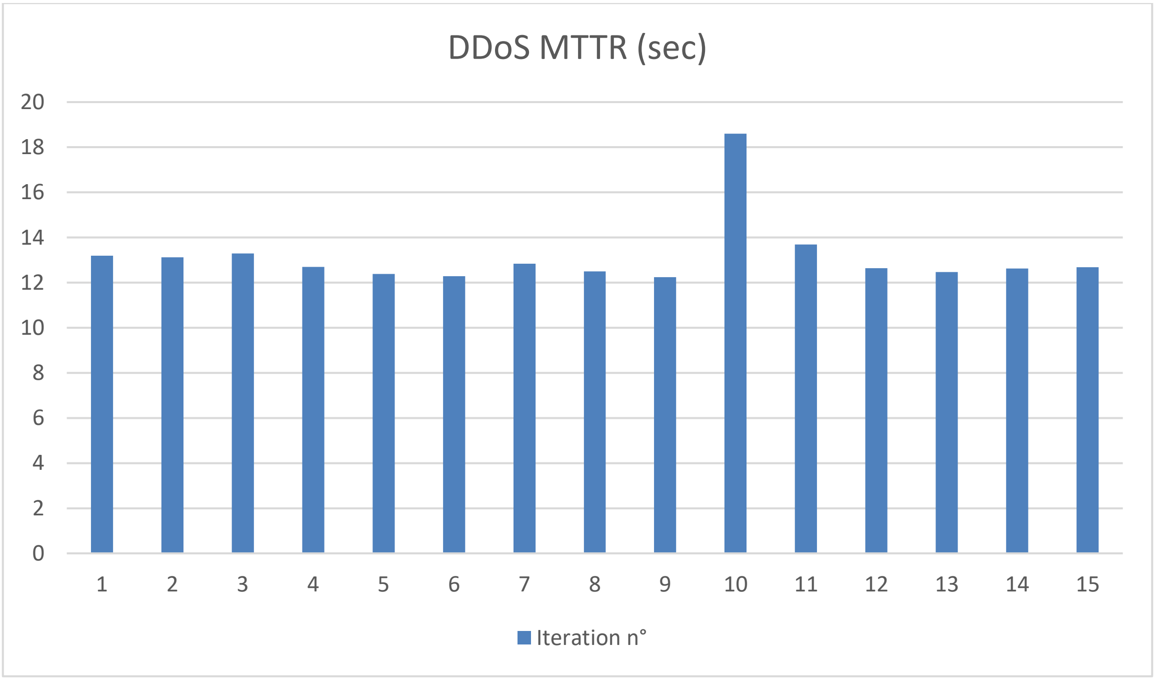


Figure 29: Histogram of Mean Time to Resolve of a DDoS filtering mitigation in the E2E SMD.

Figure 28 shows the MTTR of the DDoS filtering mitigation at the E2E level. The horizontal axis displays the iterations done during tests. The vertical axis displays the time taken in seconds to apply the mitigation.

Iteration	T0 (timestamp)	T1 (timestamp)	MTTR (sec)
1	1658742896.15581	1658742909.34566	13.1898469284

2	1658743290.38509	1658743303.51251	13.1274197388
3	1658743360.2658	1658743373.55321	13.287402906
4	1658743418.98992	1658743431.69045	12.7005304692
5	1658743462.45036	1658743474.83117	12.380813286
6	1658743511.52523	1658743523.81467	12.2894375424
7	1658743550.54741	1658743563.38652	12.8391048168
8	1658743593.92161	1658743606.41288	12.4912713156
9	1658743634.58689	1658743646.82489	12.23799693
10	1658743686.0168	1658743704.61645	18.599649222
11	1658743730.46756	1658743744.15799	13.690428558
12	1658743774.63117	1658743787.27017	12.638992974
13	1658743815.69314	1658743828.1652	12.4720613208
14	1658743857.91586	1658743870.53772	12.6218659836
15	1658743897.5066	1658743910.1863	12.6796953336

Table 18: Raw measurements of MTTR of a DDoS filtering mitigation in the E2E SMD

Table 18 contains the raw measurement of the MTTR of the DDoS mitigation inside the E2E SMD. The T0 column is the timestamps (in seconds) of the moment the E2E PyrDE receives a DDoS alert on its API endpoint. the T1 column is the timestamps (in seconds) of the moment the E2E PyrDE answers with a success code for the Rest API call. The MTTR column is the delta in second between the two timestamps. This delta contains the time taken by the E2E PyrDe to generate the mitigation MSPL, the time to submit it to the E2E UMU SO and the time to receive a success code from the E2E SO.

$$\text{Average DDoS E2E MTTR} = \frac{\sum_1^n (T1-T0)}{n} = 14.1497678217\text{ sec} \approx \mathbf{14.15\ sec}$$

According to the results, DDoS attack is mitigated at E2E level in less than 15 seconds. In this case, E2E SMD mitigation takes more time than SMD mitigation since the same reaction policy (filtering) is enforced in different ways. Filtering policy is enforced through the V2X service in the SMD case whereas the same policy in the 5G RAN Domain is enforced in the physical 5G RAN infrastructure which also requires interactions with the 5G core and data services to retrieve filtering target information.

SSLA2

Initial Time

This KPI will be measured by terms of the enforcement of E2E service and security requirements involving 3 SMDs (E2E, MI, EUR/TSG) where the time will be evaluated from the reception of the requirements by the SFS Broker to the successful deployment and configuration in each of the SMDs involved. In particular in the EURESCOM/TSG SMD the deployment and configuration of a VCP enabler CNF (Containerized Network Function) and IoT Broker in a subnet of the E2E slice will be performed. In the MI SMD, the deployment and configuration of another VCP enabler CNF and MMT Probe will be performed.

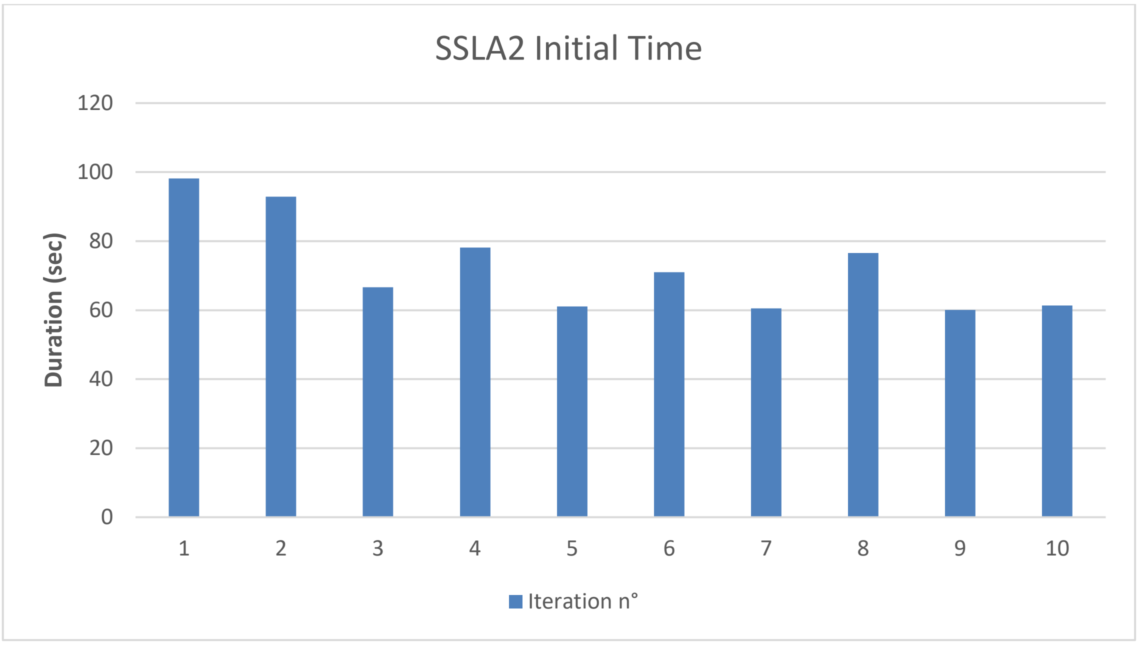


Figure 30: Initial Time for SSLA2

Figure 30 shows the obtained values of 10 different executions. The x axis represents the execution iteration, while the y axis represents the Initial Time value in seconds.

Iteration	T0 (timestamp)	T1 (timestamp)	IT (sec)
1	1666860066.210	1666860164.418	98.208
2	1666861348.113	1666861440973	92.860
3	1666862214.365	1666861281.007	66.642
4	1666863068.244	1666863146.360	78.116
5	1666863475.047	166863536.109	61.062
6	1666863771.005	1666863842.019	71.014
7	1666864104.460	1666864164.964	60.504
8	1666864352.375	1666864428.924	76.549
9	1666864678.055	1666864738.097	60.042
10	1666864868.379	1666864929.746	61.367

Table 19: Initial Time KPIs for SSLA2

Table 19 shows the following information:

- Iteration represents the number of executions (n)
- T0 represents the initial timestamp in Unix Epoch format
- T1 represents the final timestamp in Unix Epoch format
- IT represents de difference in seconds between T0 and T1

The average Initial Time (IT) is calculated with the following formula:

$$\text{IT} = \frac{\sum_{1}^{n}(T1-T0)}{n} = 73.30244444 \text{ sec} \approx \mathbf{73.30\ sec}$$

The KPI target of a maximum of 5 minutes has been met.

Mean time to detect (MTTD) (DTLS cipher violation)

The Mean time to detect SSLA2 KPI is the interval between the moment (T0) when the DTLS proxy in MI SMD sends the first DTLS (handshake) packet using a forbidden DTLS cipher suite (after malicious configuration change) and the moment (T1) when the MMT-Probe raises the alert of this violation.
The two timestamps are available in each alert generated by the MMT-Probe. Indeed, the first timestamp is initially provided by the system via libpcap when capturing the packet, and the second timestamp is generated by the MMT-Probe itself. The following represents an alert:

```
10,3,"eth0",1656510500.967694,79,"not_respected","security","Ensure DTLS traffic is using v1.2 and strong ciphersuite which is set via MMT_SEC_DTLS_CIPHER_ALLOWLIST environment variable which contains a comma-separated list of cipher numbers. Example MMT_SEC_DTLS_CIPHER_ALLOWLIST='0xc0a4,0xc0a5' will allow TLS_PSK_WITH_AES_128_CCM and TLS_PSK_WITH_AES_256_CCM ciphers",{"event_1":{"timestamp":1656510500.967694,"counter":42026,"attributes":[["dtls.packet_count",5]]},"event_2":{"timestamp":1656510500.967694,"counter":42026,"attributes":[["dtls.version",65277],["dtls.client_hello_cipher_suite",[174]]]}}
```

T0 = 1656510500.967694, and T1 = 1656510500.967694.
Consequently, we have the time to detect: T1 - T0 = 0.

The following table presents 10 other measurement results:

Iteration	T0 (timestamp)	T1 (Timestamp)	Time to detect (microsecond)
1	1656596803.473524	1656596803.473524	0
2	1656596809.854516	1656596809.854516	0
3	1656596816.128613	1656596816.128613	0
4	1656596822.414565	1656596822.414565	0
5	1656596828.854669	1656596828.854669	0
6	1656596835.218616	1656596835.218616	0
7	1656596841.510589	1656596841.510589	0
8	1656596848.026065	1656596848.026065	0
9	1656596854.555693	1656596854.555693	0
10	1656596860.986725	1656596860.986725	0

Table 23: Raw measurement results of Mean Time to Detect of DTLS cipher violation

In all measurements, we always have time to detect is zero microsecond. Since the measurement time was not able to be precise enough to measure less than a microsecond interval, the real value of time to detect could be less than a microsecond. Thus, mean time to detect SSLA2 KPI would be less than one microsecond. The values are acceptable because the tested traffic is really small, e.g., which concerns only the DTLS handshake moment. Furthermore, the violation detection would be also simple by checking the intersection of two sets of cipher suites.

Number of False Positive (FP)

FP = 0

The FP is always **zero** as the detector (MMT-Probe) is doing an exact byte (integer) comparison against fixed numeric values (flagged positive if and only some value in DTLS handshake message does not match the valid ones from SSLA), so there is no probability of error.

Number False Negative (FN)

FN = 0

The FN is always **zero** for the same reason as FP.

5G Security Service Downtime

The Security Service Downtime (SSDT) KPI is to measure the interval during which the IoT DTLS Proxy is not available to end users. The interval starts from the moment the user cannot connect to the DTLS proxy (because it is compromised by attackers) until the moment the user can access the proxy again.

The KPI is measured manually by the following steps:

- As an attacker, who can access internal service, modifies the DTLS proxy configuration to downgrade to a weaker ciphersuite. As an end user, we trigger periodically several DTLS handshakes until we cannot access to the Proxy (and note this moment T0)
- MMT-Probe should detect one or several violations, trigger the alert chain to SAE --> DE ---> SO in order to migrate the DTLS proxy and MMT-Probe to a new safe compute node
- The end user continues triggering DTLS handshakes until successfully connecting to the proxy (and note this moment T1)
- KPI measurement = T1 - T0
- As administrator, request the SO to reinitialize the slice by (1) removing the Proxy and MMT-Probe from the worker node, and (2) deploying them into the first node, and repeat the steps above to perform another measurement.

Figure 31 presents the measured values of 10 different executions, in which, the x axis represents the execution iteration, and the y axis represents the measured values in milliseconds.

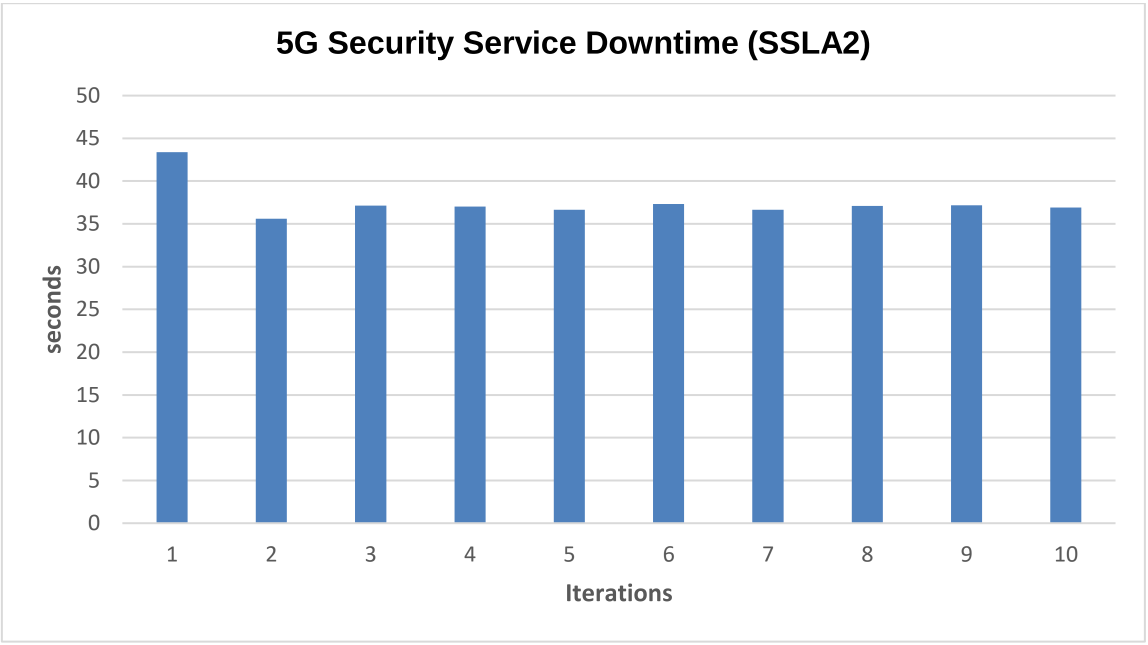


Figure 31: 5G Security Service Downtime measurements over 10 iterations for SSLA2.

The following table represents the results of our 10 measurements:

Iteration	T0 (ms)	T1 (ms)	SSD KPI Result (s)
1	1658847911497	1658847954888	43.391
2	1658848475597	1658848511202	35.605
3	1658854935538	1658854972686	37.148
4	1658855293169	1658855330212	37.043
5	1658855560181	1658855596845	36.664
6	1658855896960	1658855934274	37.314
7	1658856075091	1658856111752	36.661
8	1658856273867	1658856310964	37.097
9	1658856460221	1658856497413	37.192
10	1658856637291	1658856674217	36.926

Table 20: 5G Security Service Downtime measurements over 10 iterations for SSLA2.

Iteration column represents the number of executions (n). T0 represents the initial timestamp in Unix Epoch format. T1 represents the final timestamp in Unix Epoch format. 5GSDT represents de difference in seconds between T0 and T1.

$$\text{SSDT} = \frac{\sum_1^n (T1-T0)}{n} = \textbf{37,504 sec}$$

Packet Loss Ratio (PLR)

The PLR KPI is defined as the ratio of the number of data packets lost to the total number of packets that should have been forwarded by a network node.

These two numbers are provided by the system Kernel via libpcap. MMT-Probe reports periodically these values via its reports having ID = 200, in which:

- the number of data packets lost is the total of the number of packets rejected by NIC, T1 (6th field), and number of packets rejected by MMT-Probe, T0 (8th field).
- the total number N of packets received by NIC is in 5th field

For example, given the following report: 200,3,"eth0",1656603406.810439,1181,0,1160,0,426134,0; we have: T1 = 0, T0 =: (T1 + T0)/N*100 = 0/1181*100 = 0

We have some other results as in Table 21:

Report ID	Probe ID	NIC	Timestamps (second)	# Packets received by NIC	#Pkt dropped pkt by NIC	# Pkt received by MMT	#Pkt dropped by MMT	#Bytes received by MMT	#Bytes dropped by MMT
200	3	eth0	1656604114.44305	390	0	379	0	126211	0
200	3	eth0	1656604119.44321	405	0	394	0	131790	0
200	3	eth0	1656604124.44333	420	0	409	0	137369	0
200	3	eth0	1656604129.44382	437	0	426	0	143032	0
200	3	eth0	1656604134.44464	452	0	441	0	148611	0
200	3	eth0	1656604139.44477	467	0	456	0	154190	0
200	3	eth0	1656604144.44485	482	0	471	0	159769	0
200	3	eth0	1656604149.44502	497	0	486	0	165348	0
200	3	eth0	1656604154.44545	517	0	506	0	173335	0
200	3	eth0	1656604159.4457	537	0	526	0	181078	0
200	3	eth0	1656604164.44609	555	0	544	0	189108	0

Table 21: Packet Loss Ratio measurements for SSLA2.

Based on the measurements above, we conclude that the PLR value is **0%**.

Mean Time to Resolve (MTTR)

To calculate the local MTTR in case of SSLA2, we measure the elapsed time between the instant T0 when the Decision Engine in MI SMD receives the alert from the SAE, and the instant T1 when the countermeasure has been enforced in the MI SMD, in our case the migration of the DTLS Proxy (VCP enabler) to a safe compute node.

The figure below shows the obtained values of 10 different executions. The x axis represents the execution iteration, while the y axis represents the Initial Time value in seconds.

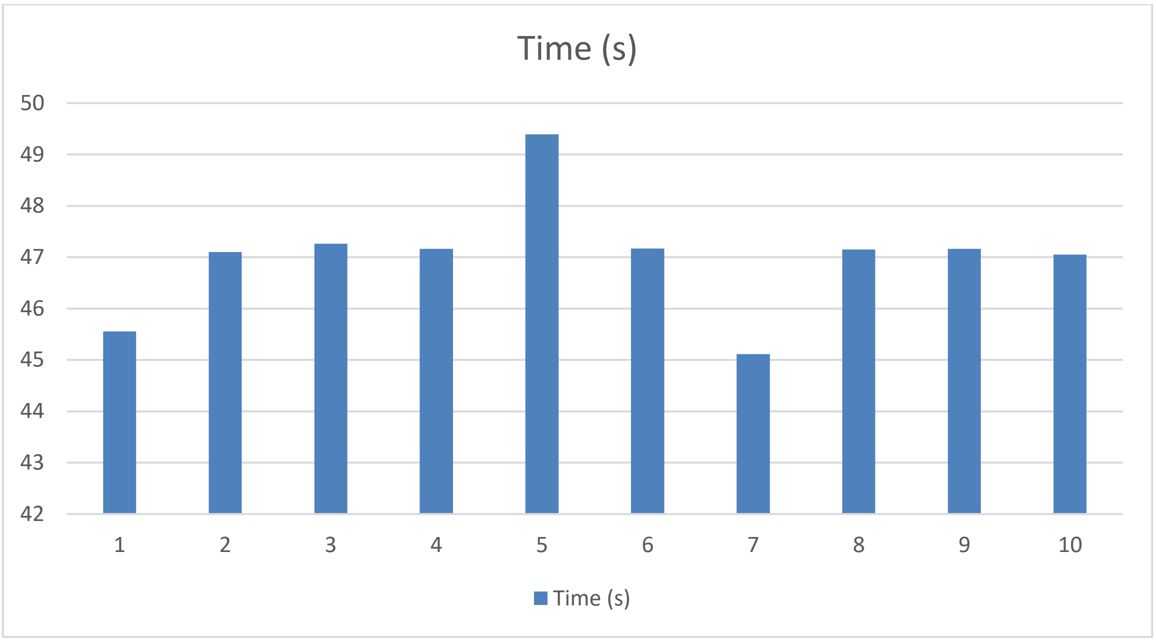


Figure 32: MTTR of SSLA2 over 10 iterations.

Iteration	T0 (timestamp)	T1 (timestamp)	MTTR (sec)
1	1657388355.429	1657388309.875	45.554
2	1657388469.279	1657388516.379	47.100
3	1657388581.569	1657388628.833	47.264
4	1657388652.626	1657388699.786	47.160
5	1657388812.619	1657388862.009	49.390
6	1657388880.009	1657388927.179	47.170
7	1657388957.025	1657389002.135	45.110
8	1657389022.718	1657389069.870	47.152
9	1657389104.600	1657389151.760	47.160
10	1657389522.207	1657389569.260	47.053

Table 22: Time samples from DE (T0: alert received by DE in MI SMD).

Table 22 shows the obtained measurements over 10 iterations for the MTTR KPI. Iteration column represents the number of executions (n). T0 represents the initial timestamp in Unix Epoch format. T1 represents the final timestamp in Unix Epoch format. MTTR represents de difference in seconds between T0 and T1.

$$\text{MTTR} = \frac{\sum_1^n (T1 - T0)}{n} = \mathbf{47{,}011\ sec}$$

3.6 Conclusions and Lessons learned

Demo1 has provided KPIs for measuring Initial time, mean time to detect, number of false positives, number of false negatives, security service downtime, 5G service downtime and mean time to resolve in two different scenarios with two different SSLAs, as well as for different kind of attacks. All the results accomplished the proposed KPI targets values. We identified that there are some KPIs such as initial time or Mean time to resolve whose results depend on the amount of dependencies and SMD involved. The more dependencies and SMD involved, the more time they will take since the enforcement will be sequential. Moreover, it is also important to highlight that each SMD can enforce the same security policy in a different way, using different techniques, so the enforcement time will depend on the available technology and the time it takes to be deployed and configured. In fact, this difference can be seen in the initial time results for SSLA1 and SSLA2, where the first one is enforced using VMs in OpenStack whereas the second one is enforced through docker containers in Kubernetes. Finally, the performance of the security enablers/assets are also a key point here. For instance, the same filtering orchestration policy was enforced in two different domains in two different ways, where the 5G RAN reconfiguration required more time than the V2X filtering.

4 Demo 2 – Trust and Liability Management

Demo 2 focuses on trust and liability management on 5G virtualized infrastructure. The demo illustrates vertical SSLA deployment through specific commitments of vertical service isolation over the deployed infrastructure and demonstrates the fulfillment of critical services to slice isolation.

Demo 2 defines a scenario showcasing the security of end-to-end services by means of the enablers as summarized in Table 23, which are further detailed in D4.1 [5], D4.2 [9], D4.3 [10] and D4.4 [11].

Demo2 Enablers	Owner
Security by Orchestration for MEC (OpenStack)	OPL
Security by Orchestration (Kubernetes)	Orange
LASM & TRAILS MANIFEST	Orange
Deep Attestation	Orange
Montimage Monitoring Framework (M-RCA)	MI
Systemic Software Security SECaaS	TAGES

Table 23: Demo 2 participating enablers.

4.1 Storyline

Demo2 is based on 4 different Domains as illustrated hereafter.

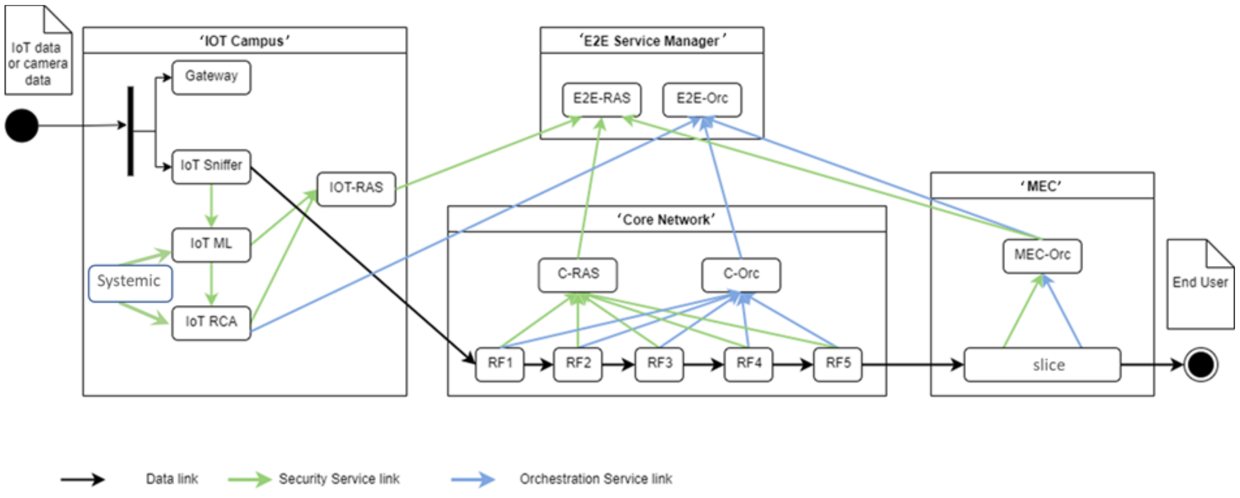


Figure 33: Demo2 topological view.

The 4 domains of Demo2 are:

- The '**IOT Campus domain**': this environment operates real IOT equipment to deliver an INSPIRE5GPlus equivalent Industrial campus. Directly inside this campus, a monitoring framework called MMT proposed by Montimage is implemented. It is worth noting that MMT is monitoring the IoT wireless network communications and not the environment (i.e., the campus). The tools are protected when critical mode is required by the service delivered by Systemic enablers.
- The '**Core Network domain**': this environment, operated in Orange's premises, is based on a real Kubernetes infrastructures operating 5 real applications to illustrate real slice conditions. It has to be noted that Kubernetes infrastructure will be charged with 50 other slices and

operated over 16 workers in order to evaluate in real condition the performance of the proposed orchestration system.

- The ‘**MEC domain**’: this environment, operated in OPL’s premises, is operated over OpenStack and implements elements of a 5G MEC infrastructure.
- The ‘**E2Eservice Manager domain**’: This domain is representing the Command Control Center of Demo2. The aim of Demo2 is to demonstrate efficiency of security enablers and not to provide industrial automation systems. That is why the system management will propose a set of menu and potential actions, to be chained or chosen manually by an operator, who operates the whole demonstration.

Demo 2 articulates two modes (i.e., normal and critical) corresponding to two different Trust Service Level Agreements (TSLAs) complying with different security threat levels. Systemic interaction with the two modes follows:

Normal mode: MMT-Probe surveys IoT traffic (in IOT Campus domain) and detects malware or misbehaviour. The reports (e.g., statistics, RCA results) generated by MMT are propagated through a core network slice to an application operated inside a MEC environment via MQTT.

Note: in this state, the MMT Probe is not protected by the Systemic SECaaS.

Critical mode: After qualification of a critical incident by a technician at the E2E Service Manager (Command Center level), the technician decides to go to a critical state and activate an additional video connection to control the safety of the industrial environment. The E2Eservice Manager Domain instructs that the MMT-Probe needs to be re-deployed in a secure environment for its protection. The Systemic SECaaS wraps the MMT-Probe. The protected version is installed instead of the original software and embedded a security routine running into Intel’s SGX enclave. Signed heartbeats are transmitted to establish security properties as: In parallel the critical mode activation forces the slice operated in Core Network to be re-orchestrated in order to be isolated as defined by the Client (owner of Industrial Campus). After activating the critical mode, the technician may collect evidence of the end-to-end operation in critical mode as defined by the condition of usage, thanks to the deep attestation enabler that will perform measures as established with the Industrial Client.

Services offered by Systemic to protect the MMT probe

- Code was self-authenticated before start
- Code executes and at the right place
- Code is integrated during run time.

4.2 HLA Mapping

The HLA mapping for demo 2 is illustrated in figure 32. Given the low TRL level of enablers, their interconnections are limited.

With the LASM and TRAILS enablers, we captured the topology and the responsibility layout of the Demo 2 service. M-RCA tool detects and identifies an anomaly and signals to the Security Orchestrators present in the 5G Core and MEC Domains, Systemic Software Wrapping tool and Deep Attestation to activate the critical modes as described above. The HLA components illustrated are Policy & SSLA management, Decision Engine, Security Orchestration, Trust Management.

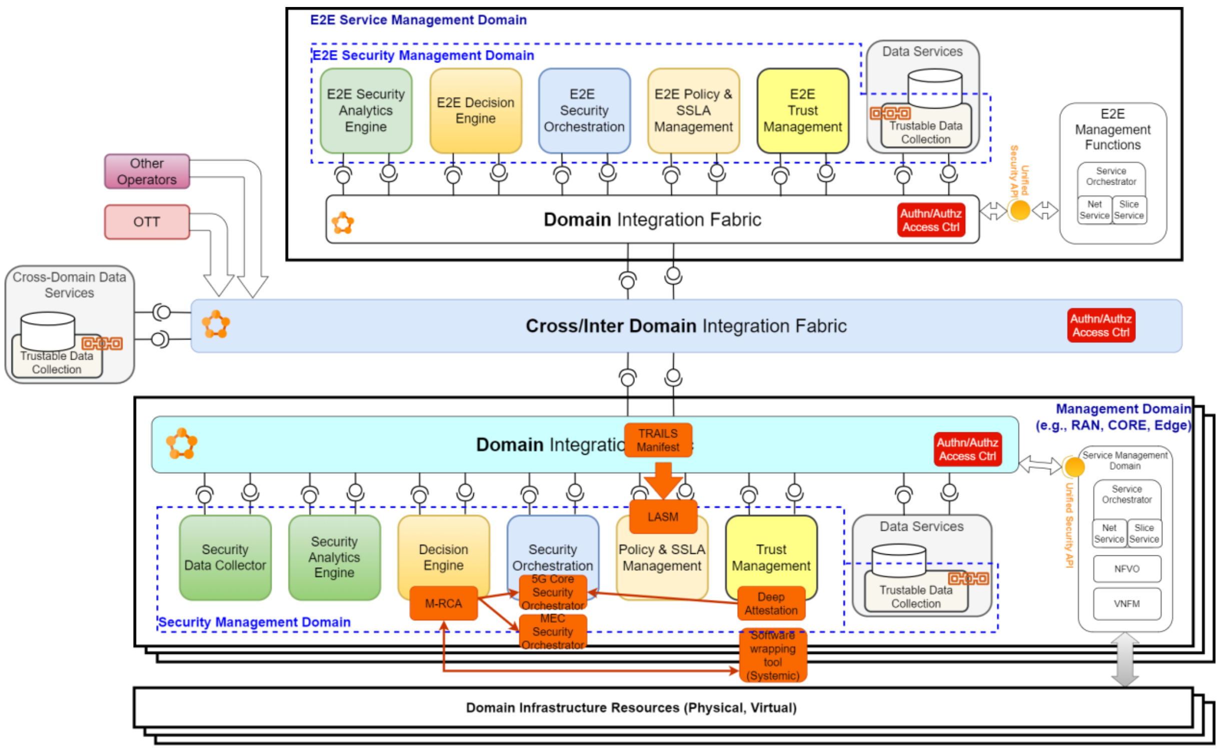

E2E Service Management Domain
E2E Security Management Domain
E2E Security Analytics Engine
E2E Decision Engine
E2E Security Orchestration
E2E Policy & SSLA Management
E2E Trust Management
Data Services
Trustable Data Collection
E2E Management Functions
Service Orchestrator
Net Service
Slice Service
Domain Integration Fabric
Authn/Authz Access Ctrl
Unified Security API
Other Operators
OTT
Cross-Domain Data Services
Trustable Data Collection
Cross/Inter Domain Integration Fabric
Management Domain (e.g., RAN, CORE, Edge)
Domain Integration Fabric
TRAILS Manifest
LASM
Service Management Domain
NFVO
VNFM
Security Data Collector
Security Analytics Engine
Decision Engine
M-RCA
Security Orchestration
5G Core Security Orchestrator
MEC Security Orchestrator
Policy & SSLA Management
Trust Management
Deep Attestation
Software wrapping tool (Systemic)
Security Management Domain
Domain Infrastructure Resources (Physical, Virtual)

Figure 34: HLA mapping of demo 2

4.3 Testbed description

A specific web interface is proposed to allow technician to interact and control Demo2 operation.

For the sake of having realistic settings, a real Kubernetes platform is used to operate enablers over the "Core Network" domain in real conditions. The Kubernetes environment (managed at Orange's side for the Core Network domain) takes advantage of GIT (https://git-scm.com/) and Argo (https://argoproj.github.io/cd/) tools for the pre-compute of container's orchestration. The platform also contains a custom Kubernetes orchestrator, that implements an optimization module. This optimization module is based on a combinatorial optimization Integer Linear Program model. It makes sure the security isolation, latency and resource constraints are all satisfied when selecting the placement of the functions.

4.3.1 Network Deployment Description and Interaction Diagrams

As illustrated in Figure 35, the complete Demo2 infrastructure is distributed in the testbeds of 3 different INSPIRE5G-plus partners.

4.3.1.1 Montimage Monitoring Framework

Figure 35 represents the IoT network (left rectangle) where the Sniffers are deployed for capturing the traffic. The information sent by the IoT devices is sniffed and transmitted to the MMT-IoT module that will analyse and extract the statistics feeding the MMT-RCA (M-RCA). Once the fire is detected, the MMT-RCA module will notify the Orchestrator indicating the potential root-causes based on its analysis via MQTT pub/sub channels. The technician is able to view the Node-RED based dashboards of M-RCA as well as to activate the camera to obtain visual information.

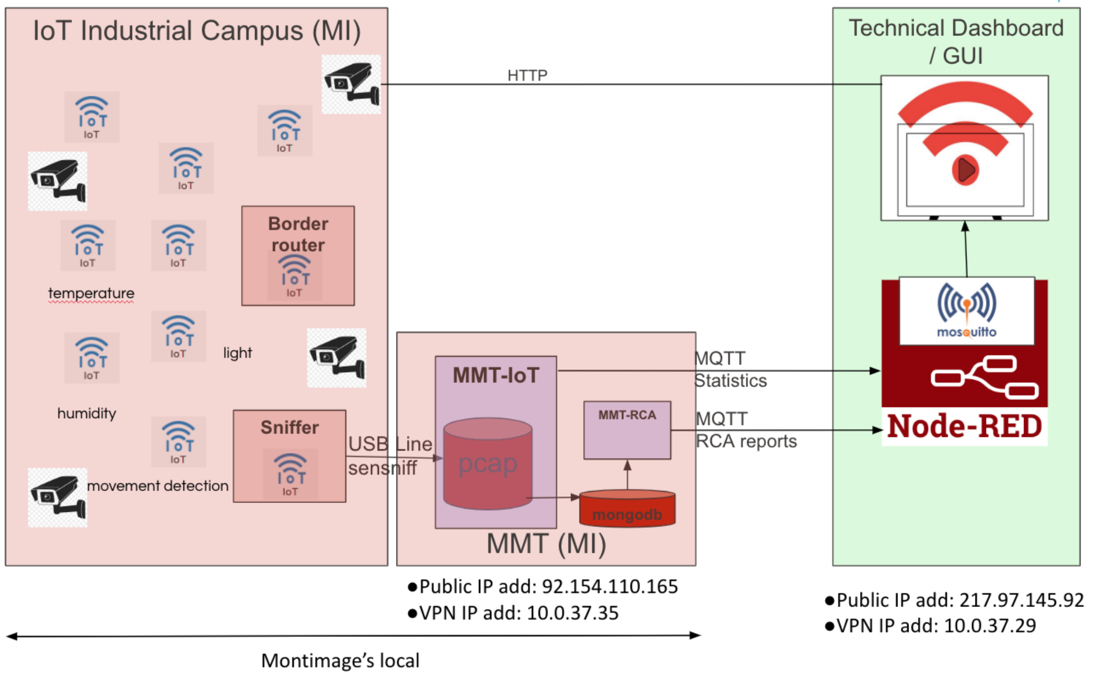
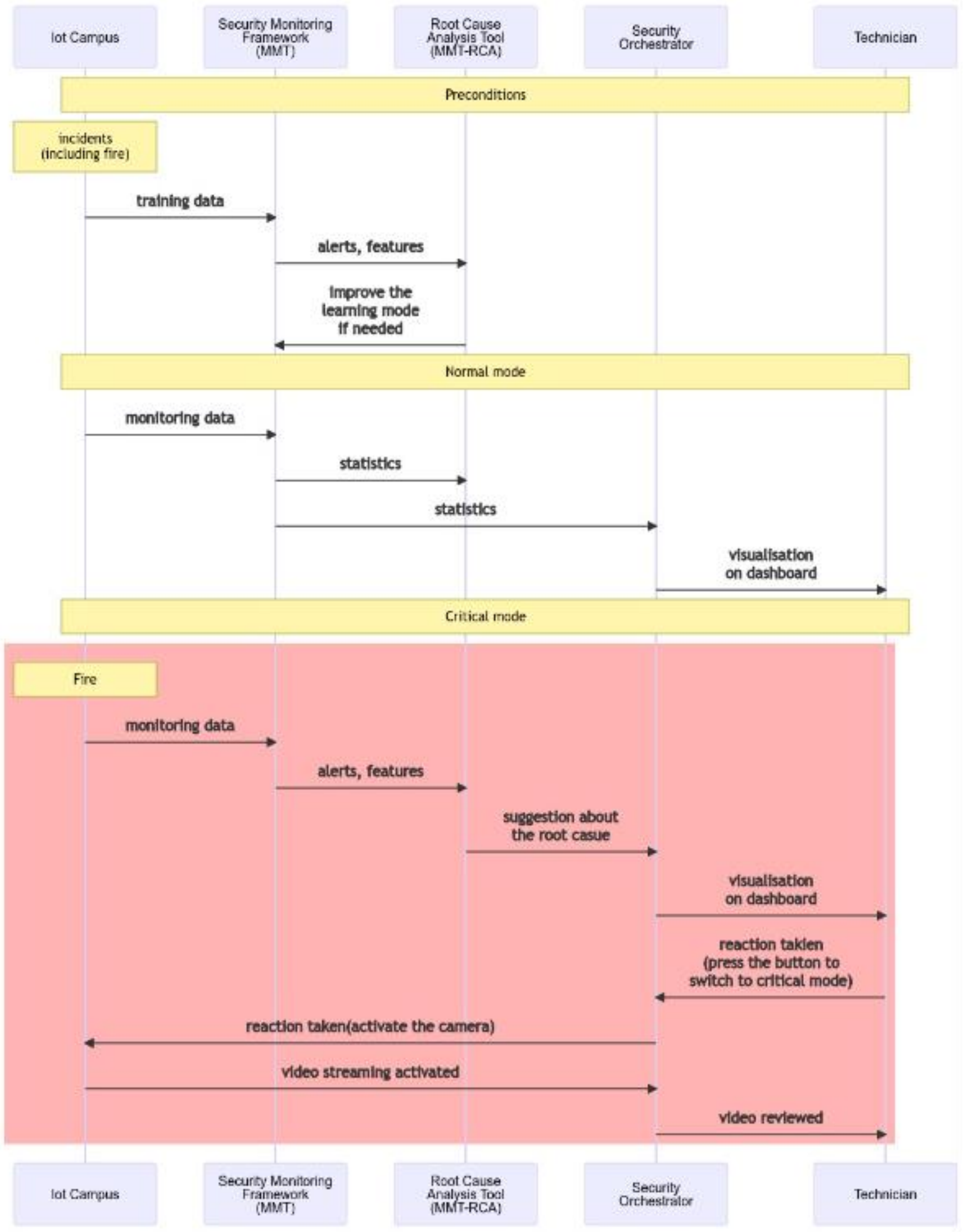


Figure 35: Industrial Campus and M-RCA.


Iot Campus
Security Monitoring Framework (MMT)
Root Cause Analysis Tool (MMT-RCA)
Security Orchestrator
Technician
Preconditions
incidents (including fire)
training data
alerts, features
improve the learning mode if needed
Normal mode
monitoring data
statistics
statistics
visualisation on dashboard
Critical mode
Fire
monitoring data
alerts, features
suggestion about the root casue
visualisation on dashboard
reaction takken (press the button to switch to critical mode)
reaction taken(activate the camera)
video streaming activated
video reviewed
Iot Campus
Security Monitoring Framework (MMT)
Root Cause Analysis Tool (MMT-RCA)
Security Orchestrator
Technician

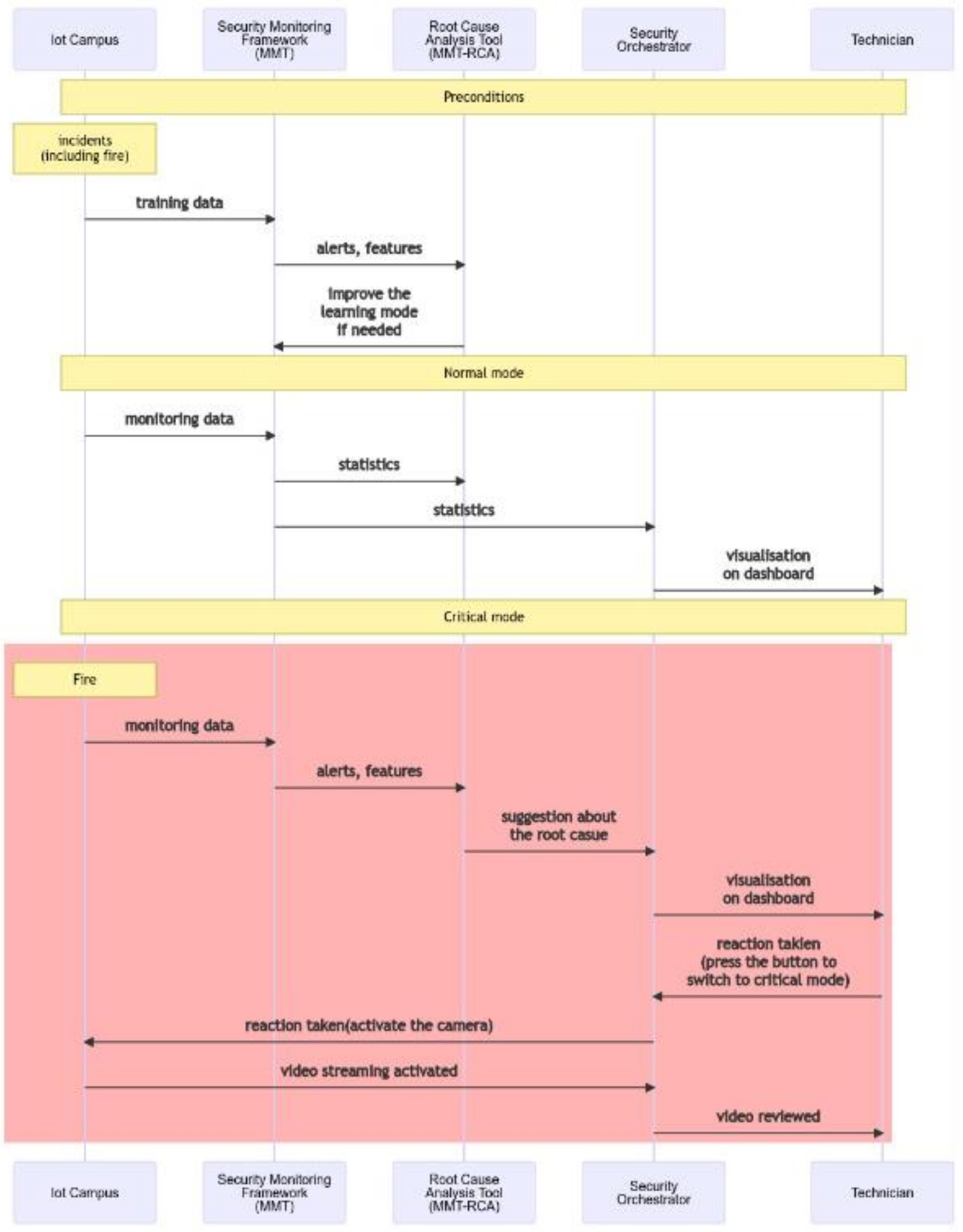

Iot Campus
Security Monitoring Framework (MMT)
Root Cause Analysis Tool (MMT-RCA)
Security Orchestrator
Technician
Preconditions
incidents (including fire)
training data
alerts, features
improve the learning mode if needed
Normal mode
monitoring data
statistics
statistics
visualisation on dashboard
Critical mode
Fire
monitoring data
alerts, features
suggestion about the root casue
visualisation on dashboard
reaction takken (press the button to switch to critical mode)
reaction taken(activate the camera)
video streaming activated
video reviewed

Figure 36: M-RCA workflow in Demo2.

Figure 36 demonstrates the workflow of M-RCA in Demo2. The actors involved in this use case are the following:

- **IoT Campus** sends monitoring data to MMT-RCA. Several **IP cameras** can be activated and manipulated/rotated from distance to focus on the source of the anomaly (i.e., fire).
- **MMT monitoring framework with the MMT-RCA module** analyses collected data and raises alerts if needed.

- **Security Orchestration** receives analysis results from MMT-RCA and displays on web-based dashboards
- **Technician Command Center** clicks the button on the web-based interface to activate the CRITICAL mode once receiving an alert from MMT-RCA.

4.3.1.2 LASM & TRAILS Manifests

Administrators, either at the MEC or E2E level interact with the LASM UI.

As depicted in figure 36, the administrator in the MEC domain creates the TRAILS manifest for the MEC service. For this, he retrieves the TRAILS manifests of network components which are already listed in the catalog with a SPARQL query[7]. Then, he resolves the requirements and capabilities constraints, updates the commitments taken on the overall new service. Once completed, he generates the TRAILS archive which corresponds to the MEC service and sends it to its customer, the administrator of the E2E Service. Upon reception, the administrator of the E2E service adds the TRAILS manifest to its catalog and applies a referencing policy expressed in SWRL (Semantic Web Rule Language)[8].

[7] SPARQL is a query language for ontologies and standardized by W3C

[8] SWRL (Semantic Web Rule Language) is a language used to express inference rules on ontologies.

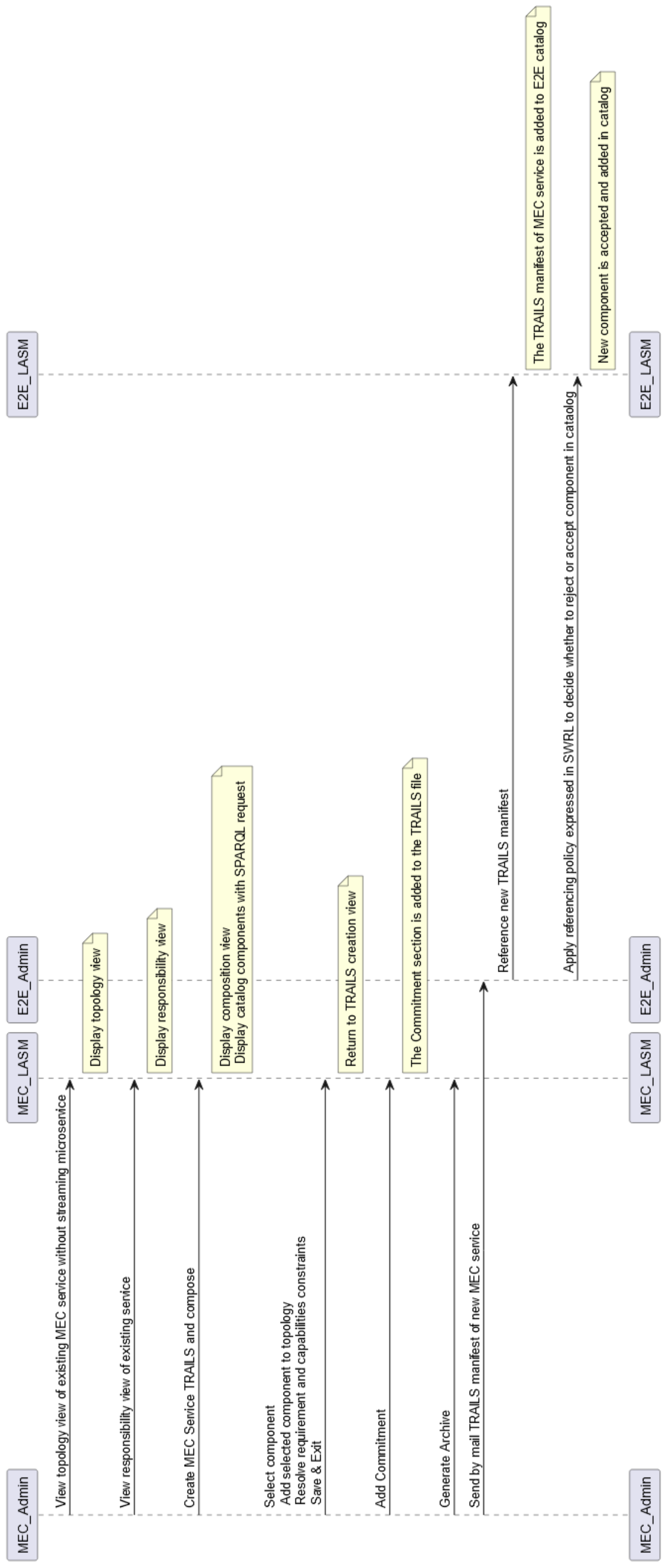

MEC_Admin
MEC_LASM
E2E_Admin
E2E_LASM
View topology view of existing MEC service without streaming microservice
Display topology view
View responsibility view of existing service
Display responsibility view
Create MEC Service TRAILS and compose
Display composition view
Display catalog components with SPARQL request
Select component
Add selected component to topology
Resolve requirement and capabilities constraints
Save & Exit
Return to TRAILS creation view
Add Commitment
The Commitment section is added to the TRAILS file
Generate Archive
Send by mail TRAILS manifest of new MEC service
Reference new TRAILS manifest
The TRAILS manifest of MEC service is added to E2E catalog
Apply referencing policy expressed in SWRL to decide whether to reject or accept component in catalog
New component is accepted and added in catalog
MEC_Admin
MEC_LASM
E2E_Admin
E2E_LASM

Figure 37: LASM and TRAILS manifest workflow in demo2

4.3.1.3 Systemic

Systemic software security is used to protect MMT-Probe software, part of the MI's MMT IoT solution. Systemic is a Security as a Service solution made available to Montimage using credentials, using either the WEB-based Graphical User Interface, or via direct API requests. As a result of INSPIRE-5Gplus project, MMT-probe is protected with Intel's SGX flavor.

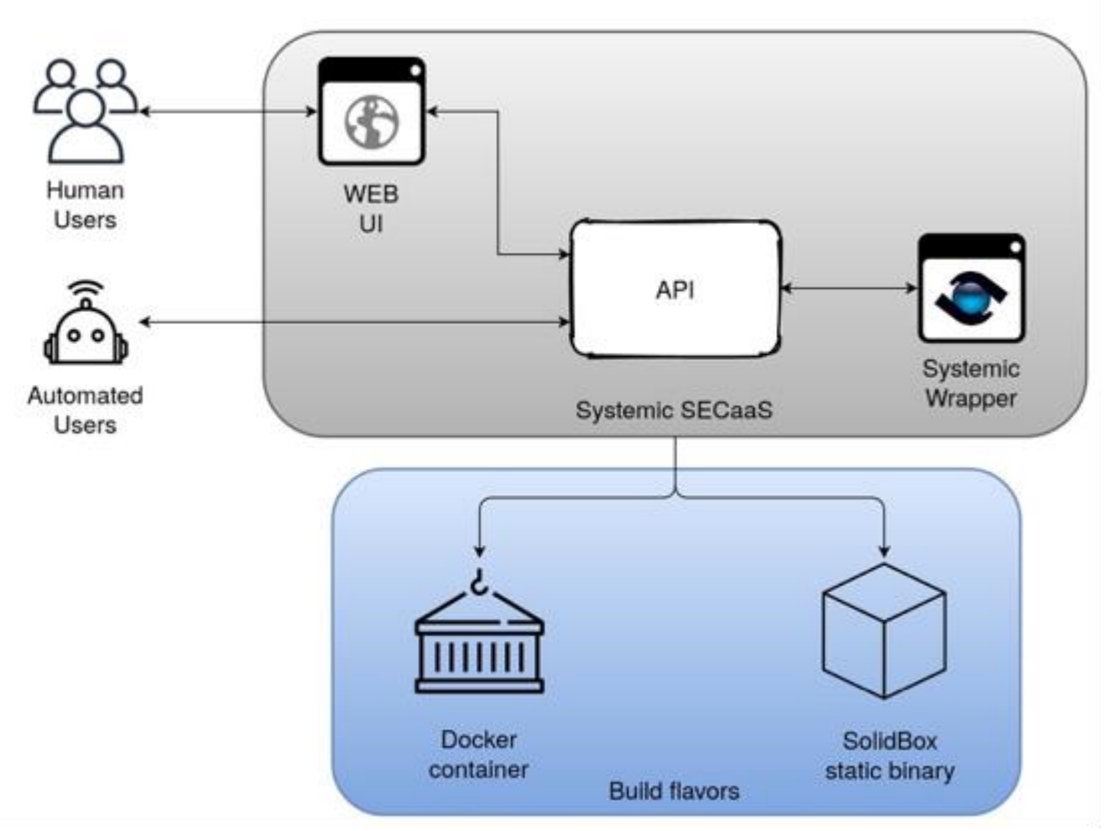


Figure 37: General Layout of Systemic wrapper and interfaces

As a second result of INSPIRE-5Gplus project, a monitoring facility has been developed, , it is made of the run-time integrity measurement routines, the server to collect such measurements and the WEB interface to display them to the administrators. The goal is generating periodic heartbeats which attest the correct conditions of operations of the protected software during its operations. The heartbeats measure the integrity of the application at run-time and are encrypted and signed from within Intel's SGX to confer trust to their content. They bring a novel type of deep monitoring, deeply inserted into the protected program control flow. The contribution of Systemic into Demo 2 is shown in the following sequence diagram, split in two phases of (i) the protection and instantiation phase of MMT-probe and (ii) its continuous monitoring by the novel deep monitoring facility.

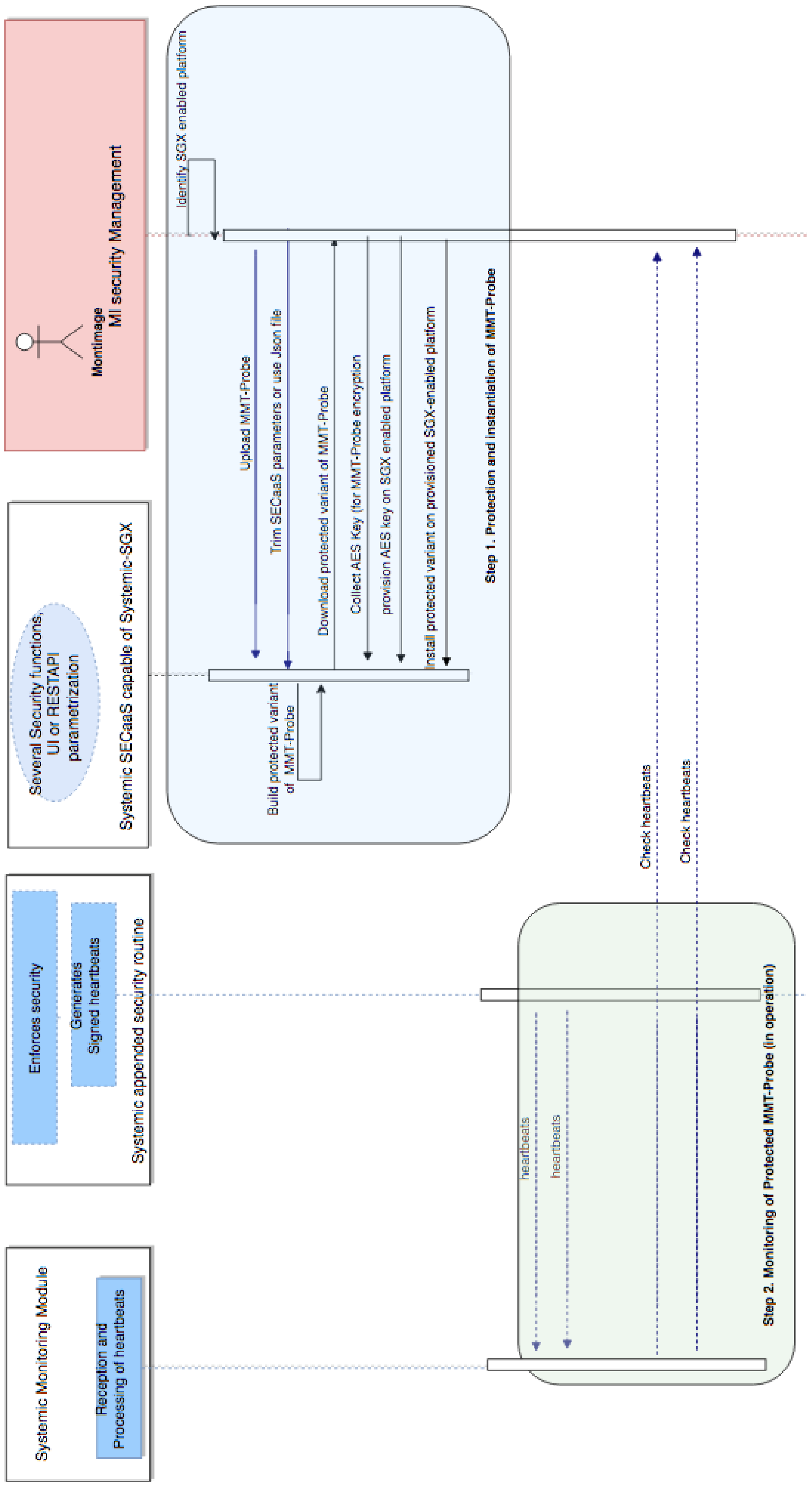

Montimage
MI security Management
Several Security functions, UI or RESTAPI parametrization
Systemic SECaaS capable of Systemic-SGX
Enforces security
Generates Signed heartbeats
Systemic appended security routine
Systemic Monitoring Module
Reception and Processing of heartbeats
Identify SGX enabled platform
Upload MMT-Probe
Trim SECaaS parameters or use Json file
Download protected variant of MMT-Probe
Collect AES Key (for MMT-Probe encryption
provision AES key on SGX enabled platform
Install protected variant on provisioned SGX-enabled platform
Step 1. Protection and Instantiation of MMT-Probe
Build protected variant of MMT-Probe
Check heartbeats
Check heartbeats
heartbeats
heartbeats
Step 2. Monitoring of Protected MMT-Probe (in operation)

Figure 38: Systemic hardening and monitoring of MMT-Probe in Demo2 (critical mode)

The figure shows how Montimage's MMT-Probe is protected and monitored by Solidshield's Systemic-SGX variant. Noticeably, all of these actions take place in Demo 2's critical mode which is triggered by the service operator to confer a higher trustworthiness of the MMT-IoT surveillance service. Higher trustworthiness of the service first derives from the installation at a deemed trustworthy platform, enabled with Intel's SGX. Then a protected variant of MMT, hardened by Systemic in confidentiality and integrity is generated by Systemic's SECaaS using its SGX elevated security assurances to Systemic appended routine, along with the generation of an AES key used for the variant decryption. As a result, the variant can only be executed on one of the platforms provisioned with this AES key. Montimage will install the key into the SGX-enabled trusted platform, which will be the only one able to operate its protected MMT-Probe variant executable file appended with Systemic's routine. This routine will be sheltered and protected into Intel's SGX trusted execution environment. Once MMT-probe protected variant launches and operates, periodic heartbeats are generated and signed by the Systemic routine and transmitted to Systemic's monitoring collector, capable to verify their authenticity and collect tem. The following monitored elements are carried by the heartbeats: (i), the program is authenticated (therefore is integrated and originated from Systemic's SECaaS), (ii) it has been decrypted before its launch (which attests that the program executes on the trusted platform), (iii) the program truly executes and is integrity is verified during its execution (as a result of the periodic integrity checks generated by one of the program thread).

4.3.1.4 Deep Attestation

Figure 38 gives an overview of the API that can be used to verify a given property.

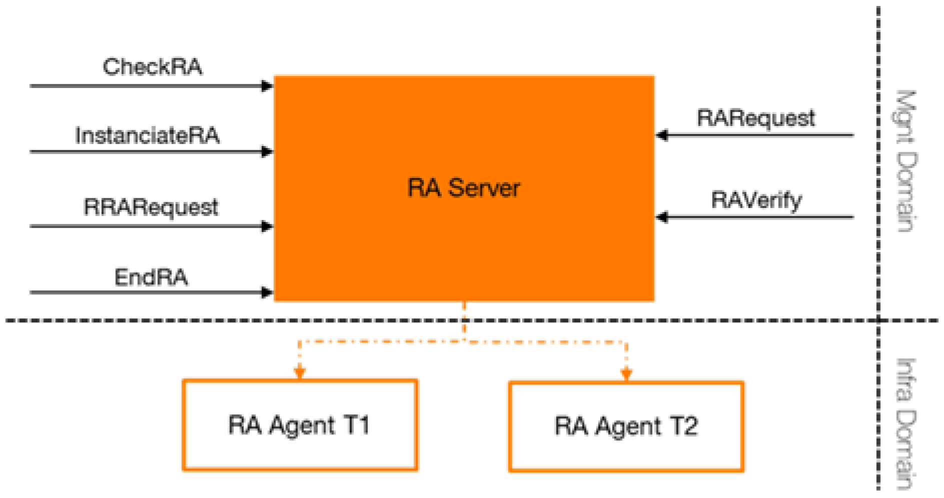


Figure 38: The RA API that can be called

<u>Management APIs</u>

- CheckRA (targetID, param): This command enables to check if the node « targetID » is able to run a RA.
- InstantiateRA (targetID, type, param): This command triggers an instanciation of the required code in the node « targetID ». If the instanciation succeeded, the « targetID » is added to the internal register « RunningRA »
- RRARequest: This command returns the list « RunningRA ».
- EndRA (targetID, param): This command triggers the deletion of the RA related code running in the node « targetID ». If the deletion succeeded, « targetID » is deleted from the internal register « RunningRA »

Service APIs

- RARequest (targetID, type, param): This command run the RA code on the node « tragetID » if the node « targetID » existes in the internal register « RunningRA ».
- RAVerify (SIG, targetID, param): This command enables the verification of the result of a « RARequest » if the node « targetID » existes in the internal register « RunningRA ».

“param” is an empty field that can contain additional information.

In Figure 39, we show the exchanges between a Vertical and the Infrastructure Operator (RA Server, Agents) to have an attestation from a target. The attestation will help the Vertical to check whether the target ensures a given property. For the sake of simplicity, we schematize only two agents, but we can have more agents associated to an RA server. The agents can be on the same target or in different targets. We also omit the parameter of the commands.

-

Step 1: Check if the target is RA compliant

Sequence (1-3)

The vertical provides a target ID to the RA server to check if this target can perform an RA. The target ID can be for instance an IP address.

-

Step 2: Instantiation of the RA service

Sequence (4-12)

If the target is RA compliant, the vertical can ask to instantiate the RA service on it. Upon receiving this request, the RA server checks first whether the target is already instantiated and ready to perform attestations (6). If it is the case, the vertical can ask for an attestation (Next sequence). Otherwise, the RA server will upload and instantiate the target.

-

Step 3: Attestation request

Sequence (13-20)

In this sequence, the vertical should precise the type of the requested attestation (e.g., deep attestation). He can also provide some parameters needed to compute and verify the attestation like a nonce. Upon receiving the request, the RA server triggers the corresponding agents running on the target.

-

Step 4: End of the service

Sequence (21-24)

When the vertical receives the requested attestation, he can decide to clean up the target from all the RA materials.

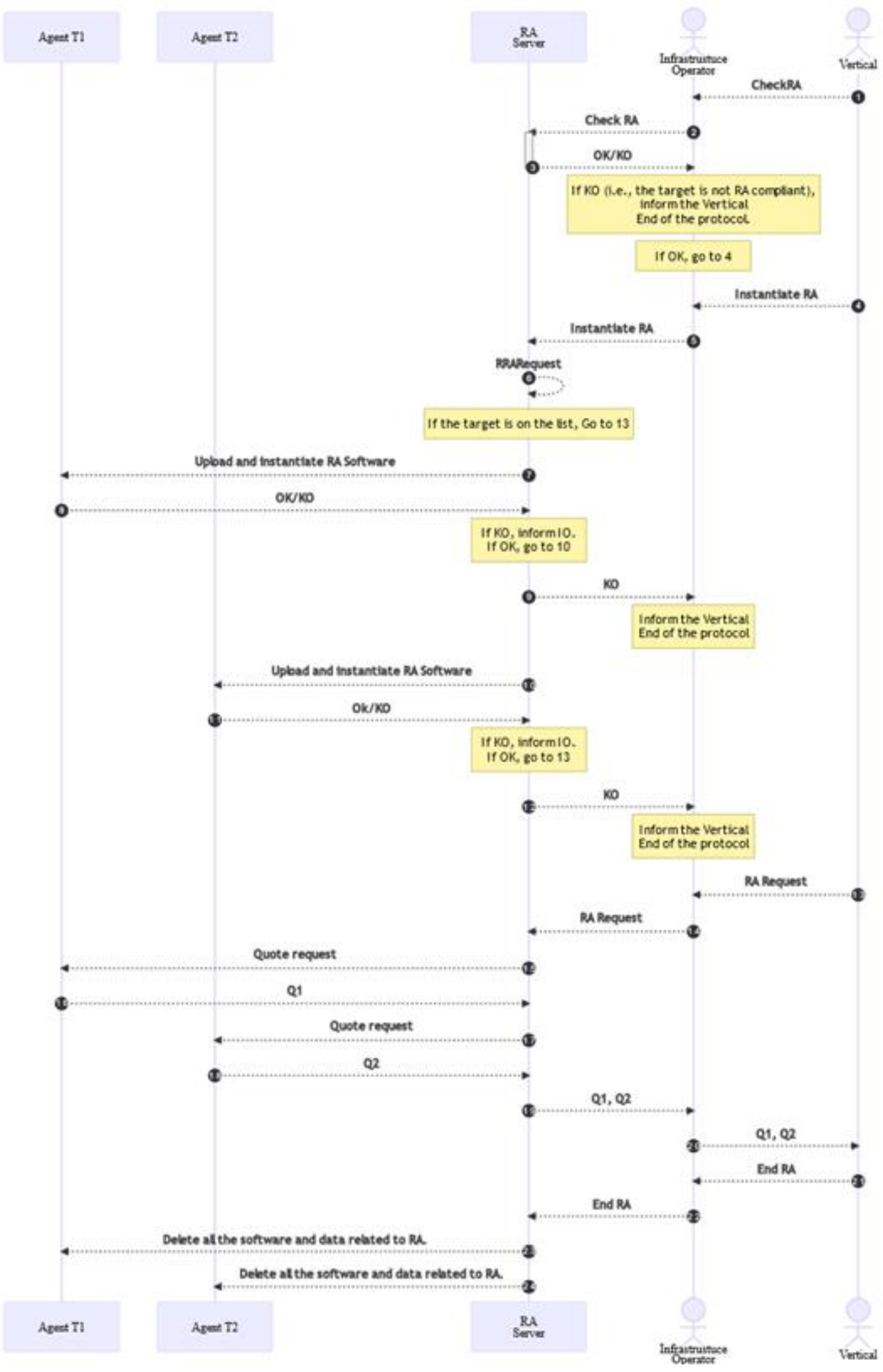

Agent T1
Agent T2
RA Server
Infrastructure Operator
Vertical
CheckRA
Check RA
OK/KO
If KO (i.e., the target is not RA compliant), inform the Vertical End of the protocol.
If OK, go to 4
Instantiate RA
Instantiate RA
RRARequest
If the target is on the list, Go to 13
Upload and instantiate RA Software
OK/KO
If KO, inform IO. If OK, go to 10
KO
Inform the Vertical End of the protocol
Upload and instantiate RA Software
Ok/KO
If KO, inform IO. If OK, go to 13
KO
Inform the Vertical End of the protocol
RA Request
RA Request
Quote request
Q1
Quote request
Q2
Q1, Q2
Q1, Q2
End RA
End RA
Delete all the software and data related to RA.
Delete all the software and data related to RA.

Figure 39: Basic Flow of RA - Deep Attestation

4.3.1.5 Security-by-Orchestration for Kubernetes

The security by Orchestration enabler is implemented inside a Kubernetes infrastructure and is used to isolate specific chain of NFs in Demo 2 (on demand). The Kubernetes infrastructure implements the whole framework of Deep Attestation, with a specific agent instantiated at each system node level.

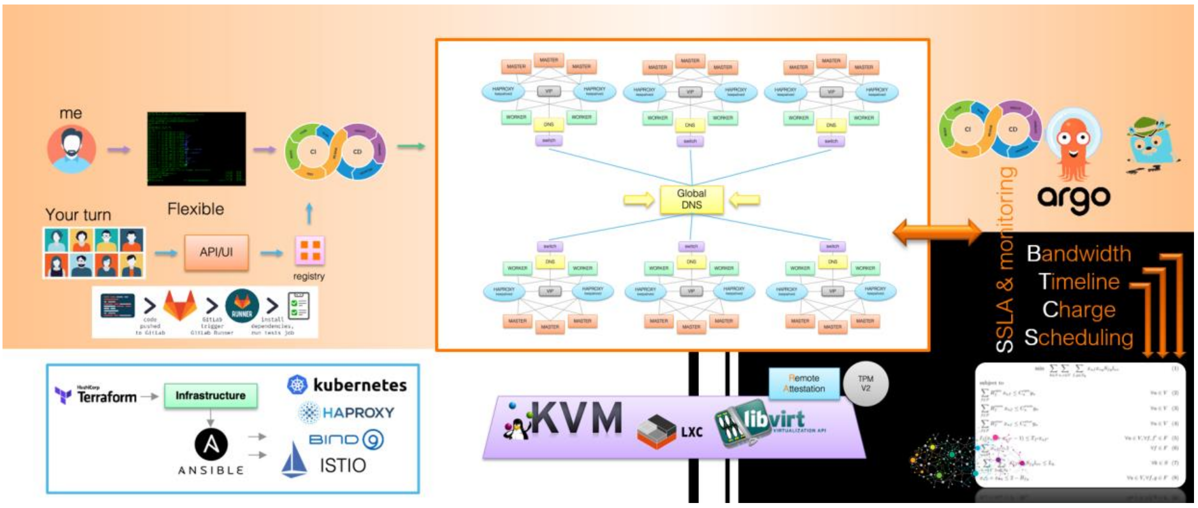


Figure 40: Demo 2 Kubernetes infrastructure

4.3.1.6 Security-by-Orchestration for MEC

Security by Orchestration for MEC enabler is integrated into OPL MEC experimental environment based on Opentack. The environment represents the Edge SMD in INSPIRE5G-Plus HLA. The enabler is a part of MEC Orchestrator and receives security policy parameters in the "additionalArtifacts" object the MEC Application Package data. It performs the placement of MEC applications using Infastructure Orchestrator (Openstack), MEC applications serve the IoT Campus devices over 5G Core domain.

4.4 Validation of Demo 2 KPIs

4.4.1 Demo2 KPIs

This section presents the KPI delivered by Demo2. Table 24 summarizes the Demo 2KPIs, their descriptions, target values and results. Sections 4.4.2.1 – 4.4.3.2 provide more information on the measurements conducted.

Demo 2 KPI	Description	Target Value	Results
Initial Time (IT)	Duration of Systemic's SECaaS wrapping process (ie, time to protect MMT-Probe)	Systemic : <60 sec MMT-Probe: <60 sec	Systemic's initial time on MMT Probe: **32 sec**
Mean Time to Detect (MTTD)	Time to detect an anomaly (from the occurence of the anomaly)	Systemic: Average time 7,5 seconds MMT Probe: <10 sec	Systemic: The average Period of the heartbeats is **15 sec**. The Standard deviation is less than 0.1 **sec**. MMT Probe average **8.4 sec**
False Positive		Systemic's measures on MMT-probe (code self-authentication test succeeds; code executes and code is integrated (periodic test): 0% MMT Probe: 0%	Systemic: **0%** (by nature of the test) MMT Probe: **0%**
False Negative		Systemic's measures on MMT-Probe (code self-authentication test succeeds, code executes and code is integrated (periodic test): 0% MMT Probe: 0%	Systemic: **0%** (by nature of the test) MMT Probe: **0%**
Service Response Time	Time to run a deep attestation of an infrastructure (Hypervisor + VM)	Deep Attestation: <15 sec Security-by Orchestration: <1min Security by Orchestration for MEC: <1min LASM – SPARQL: <1 sec LASM – SWRL: <1 min	DA: The maximum duration is **11.55 sec**; the median is **5.33 sec** and the mean duration is **5.68 sec** (for 100 trials) 5 seconds 0.26 seconds LASM-SPARQL : Average **1.6 sec** LASM-SWRL : Average **0.76 sec**
% Packets lost		MMT Probe: 0%	MMT Probe: **0%**

		Security-by-Orchestration: <1%	
% of security isolation requirements not satisfied		Security-by-Orchestration: <1% Security by Orchestration for MEC: <1%	**0**
Time to go in critical mode	Time to compute placement of micro-services for critical mode	Less than 5 seconds	
Packets lost during reorchestration (normal to critical mode):		Less than 1%	**0%**
Ratio of security isolation requirements not satisfied		Less than 1 %	**0%**

Table 24: Demo 2 KPIs, target Values and Results

4.4.2 Results of Demo 2

4.4.2.1 Systemic enabler KPI

The enabler KPIs are as follows:

- **Initial Time:** Cycle duration for wrapping an executable: Less than 60 seconds

Mean Time To Detect: (code tampering, code was not modified before being loaded, code executes outside a pre-defined White Execution Zone (WEZ): Freshness of the alerts=frequency of the heartbeats: **7.5 seconds** (defined as the average value between the measurement and the next heartbeat generation (Period is defined at 15 seconds).
Side notice: The frequency of the heartbeats is user-defined, as the amount of time required for the integrity measurement integrity time depends on the size of the protected software. Improvements on Systemic user interface will guide the user to select the ideal frequency.

- **Mean Time To Resolve**: Complete cycle including:
    - Reception of the security alert,
    - decision taking by the security orchestrator, and
    - instruction at the MANO to re-install a pristine version of the code.
- **False Positive and False negative:** 0% (by the very nature of all tests produced)

KPI	Value	Measurement on MMT probe
Initial time	less than 60 seconds	32 seconds (average over 100 wrapping), Beyond 60 seconds: None
Meant Time to Detection	Average time 7,5 seconds	Measurements made over a batch of 100 independent integrity tampering attempts, each performed at a random delay after the startup of the respective MMT process. The average Period of the heartbeats is 15 seconds. The Standard deviation is less than 0,1 second.

Table 25. Systemic KPIs

4.4.2.2 Deep Attestation enabler KPIs

The enabler KPI is:

- **Service Response Time:** Cycle duration for operating: Less than 15 seconds.

For the measurements, our hypervisor is a laptop running Ubuntu 20.04.1 (kernel version 5.4.0-58) with an Intel i5-10210U CPU, 8GB RAM and a Nuvoton TPM NPCT75X. We used KVM to turn this laptop into a hypervisor. We used a full virtual TPM implementation, using QEMU with libtpms version 0.7 and swtpm [15] version 0.5. All virtual machines are QEMU virtual machines (version 4.2.1) with 2 cores and 4G RAM running Ubuntu 20.04.1. We run 100 trials of the hypervisor attestation and 100 trials of the VM attestation. In table 29, we provide the minimum, median, mean and maximum.

	min	*median*	*mean*	*max*
Hypervisor	*3.22*	*5.33*	*5.68*	*11.55*
VM	*0.66*	*0.97*	*1.03*	*1.41*

Table 26: Deep attestation enabler KPIs: Minimum, median and maximum time in second for attestation of a hypervisor and a virtual machine (100 Trials).

4.4.2.3 MMT KPIs:

- **Packet Loss Ratio:**
    - Number of packets processed by MMT/ Number of packets captured by the IoT Sniffer: **0%**
    - Number of packets captured by the IoT Sniffer vs Number of packets exchanged/ collected by the IoT Border Router: **0%**
- **Mean Time To Detect (MTTD):**
    - The average length of time between the start of an incident's recurrence and their discovery based on the similarity score: 8.4160945 (s)

Test	T0 Y-M-D H:M:S	T1 Y-M-D H:M:	MTTD (s)
1	2022-9-16 10:9:47.779791	2022-9-16 10:9:55.992788	8.212997
2	2022-9-16 10:10:0.997936	2022-9-16 10:10:9.439580	8.441644
3	2022-9-16 10:10:14.443337	2022-9-16 10:10:22.735571	8.292234
4	2022-9-16 10:10:36.346088	2022-9-16 10:10:44.604182	8.258094
5	2022-9-16 10:10:54.609375	2022-9-16 10:11:2.953540	8.344165
6	2022-9-16 10:11:7.958411	2022-9-16 10:11:16.339423	8.381012
7	2022-9-16 10:11:21.341970	2022-9-16 10:11:29.715265	8.373295
8	2022-9-16 10:11:34.720396	2022-9-16 10:11:43.985112	9.264716
9	2022-9-16 10:11:48.103644	2022-9-16 10:11:56.421281	8.317637
10	2022-9-16 10:12:1.425856	2022-9-16 10:12:9.701007	8.275151

Table 27: MMT KPIs

- **Accuracy:**
    - False positive: 0
    - False negative: 0
- **Initial time:** about 12 seconds

- **Confidence:**
    - Similarity score >= 85% (The higher similarity score represents a higher confidence)
    - Number of supervised learning datasets: 13 datasets of about 2500 records each.

4.4.2.4 Security by Orchestration enabler KPIs:

We started with a Service Chain already deployed on our testbed Kubernetes platform with the default Kubernetes orchestrator. The placement of the 10 micro-services does not meet security isolation constraints (defined in the SSLA). Then we request to enter critical mode and run the placement algorithm to compute the optimal placement of micro-services. We measure the time to compute this optimal placement.

- **Service Response Time:** Cycle duration for operating a normal to critical mode migration: Less than 5 seconds.
- Our measurements indicate an average time to compute the placement of 4.35 ms**% packets lost during reorchestration (normal to critical mode): less than 1%**

  The packet loss was at 0%
- **% of security isolation requirements not satisfied: less than 1 %**

  The security isolation requirements was at 0%

4.4.2.5 Security by Orchestration for MEC enabler KPI:

- **Mean Ratio of Time Functions are Not isolated In Critical mode**: 0 seconds

  Current implementation of Security by Orchestration by MEC offers static application placement under constraints
- **Mean Observation Report Request Response Time**: 0.258483 seconds

4.4.2.6 TRAILS enabler KPI:

We modeled the Demo2 5G End to End Service which is composed of an IoT campus, 5G core, MEC and a E2E management service reached 1,0 MB.

When we enter this model in the TRAILS ontology, we obtain the metrics described in Table 28:

	Demo 2 End to End Service
Axiom	2579
Logical axiom count	642
Declaration axioms count	607
Class count	58
Object property count	36
Data property count	49
Individual count	490
Annotation Property count	31

Table 28: TRAILS demo 2 KPI

4.4.2.7 LASM enabler KPI:

The LASM provides a smart directory and advanced search functionalities thanks to the use of an ontology. We provide hereafter some metrics measuring the performance.

- **Time to respond to a SPARQL query**

SPARQL is a language used to query an ontology and standardized by W3C. While composing a new service, the administrator at the level of the MEC domain or E2E Service performs a SPARQL query to retrieve a list of components from the catalog.

A request consists of searching for a TRAILS according to characteristics using the ontology, the evaluation has been done with three queries. The results are presented in the table 30. This is useful for reorchestration.

	Description	Result
Query 1	Query to find the TRAILS of a component type Network Service under the responsibility of Orange.	Less than 1 second
Query 2	Query to find the TRAILS of a component that has a “container” capability and doesn’t have a requirement.	Between 1 second and 1.50 seconds
Query 3	Query to find the TRAILS of a component in a MEC service that has as capability “Critical Mode Activation” and an SLA on the MORRT (Mean Observation Response Request Time, defined in D4.4 [12]) that is less than 10ms.	Between 1.80 seconds and 2 seconds

Table 30: LASM Time To Respond to SPARQL query

- **Time to respond to a SWRL reasoning:**

In our context, the SWRL rules are referencing policies that will evaluate the TRAILS before its referencing. The following rules have been used:

- Rule n°1 : Accept only the TRAILS that have “high” as validation score .
- Rule n°2 : Reject the TRAILS that have an energy consumption above 0.0018kw/h.
- Rule n°3 : Assigns a scaling policy to a specific VNF model for which cybersecurity tests showed the need of scaling up resources such as CPU, RAM, energy.

The results are described in Table 29:

	Demo 2 E2E TRAILS manifest
Rule n°1	0.75 seconds
Rule n°2	0.75 seconds
Rule n°3	0,78 seconds

Table 29: Time To Response to a SWRL reasoning request

4.4.3 Deployment setup for measurements

4.4.3.1LASM enabler KPI:

The measurements were performed five times on an Intel® Xeon® W-2133 Processor with 32 GBytes of available RAM and the results presented above corresponds to the average times measured over the 5 experiments. To compute the KPIs, specific web interfaces are proposed to interact with the ontology. The **Time to respond to a SPARQL query** and the **Time to respond to an SWRL reasoning request** are the delay in seconds between the HTTP request being sent and the HTTP response being received.

4.4.3.2Security by Orchestration for MEC enabler KPI:

The measurements of MEC ORTT were performed 20 times using the "curl" tool for response time calculation and the presented result is the average value.

To collect measurements, Security by Orchestration enabler was used to deploy test application on the experimental OPL MEC environment.

After launching MEC Application under the security constraints, requests were sent to enabler’s REST API interface. This way the report about placement of MEC Application according the SSLA was requested

4.4.4 Dissemination of Demo 2 Results

The architecture of Demo 2 was presented in the paper “The Owner, the Provider and the Subcontractors: How to Handle Accountability and Liability Management for 5G End to End Service” published in the Emerging Network Security Workshop within the ARES 2022 conference.

Part of Demo2 have been publicly presented during the exhibition week (17 to 20 of October 2022) at Orange- Chatillon (“Salon de la Recherche & Innovation 2022”). The enablers and demo2 parts presented during this event were:

- The ‘**Core Network domain**’
- The ‘**MEC domain**’
- The ‘**E2Eservice Manager domain**’

the Enablers: Security by Orchestration, Deep Attestation Framework, Systemic Software Security SECaaS, LASM & MANIFEST enablers

4.5 Conclusions and Lessons Learned

Through its Demo2, INSPIRE5GPlus project illustrates a novel way to propose, deliver and provide evidence of security commitments that could be requested on-demand by Verticals. Demo2 targets essentially the NIS2 Directive on NIS2 verticals especially addressing Operator of Essential Services and Operators of Critical services and their new regulation constraints and safety obligations. More specifically, for NIS2 directive's targeted operators, the fact to be in capacity to operate On-Demand and to deliver evidences that security services are really delivered is a valuable and instrumental step to allow them to delegate and control part of their Regulation constraints and safety obligations.

As the main result of all work worked out in WP4 relating to Liability aware Trusted 5G security, the issues related to Trust versus Liability and their duality have been illustrated and discussed. They are resolved in a simply way inside Demo2, in particular regarding the request of Slice components isolation (Security by Orchestration linked to a Deep Attestation framework) or remote proof that a software is operated in a critical mode (SYSTEMIC to protect MMT-IOT server). The SLA concept used in Demo2 is described in the section: D2.4- § 6.1 ‘5G imposes technological trust models discontinuity’. Precisely, as a result, for each proposed SLA to be committed by the parties, the Demo2 establishes a trust and liability mechanism linking together 3 major commitments (i) description of a security SLA, (ii) a way to collect evidence or measure the effectiveness of the SLA over the infrastructure and referred as ‘the convention of proof’ to be agreed between the stakeholders and (iii) an attestation framework accessible by the Client and operated by the infrastructure owner, both together defining the parties , leveraging the Deep Attestation feature used to collect and signed the collected evidence of SLA realization.

The first benefit of WP4 integrated research, exemplified in Demo2, is the elaboration of a simple method “convention of proof” to establish transparency and accountability for security measures associated to trust and security obligations. The second benefit is to establish crypto-proven or solid evidences as specified in the "convention of proof” and transmitted to the operators that all implied security enablers (operated directly or sub-contracted) are effectively operating as they should. The third benefit is the elaboration of a versatile and scalable attestation framework, capable to attest integrity of virtualized payloads and supporting platform as well as any collected points of liability measures as defined in the “convention of the proof”.

Part of Demo2 have been demonstrated in the Industrial event organized by Orange[1] , and we now investigate (discussion between INSPIRE5GPlus project partners) under which conditions we could continue, operate, deliver and standardized those approaches and propose new schemes of certification (ENISA EUCC / EUCS / 5G) in order to allow Clients / Verticals to receive the collected evidences as facts opposable in court or in the litigation phase with CyberInsurance for instance.

[1] INSPIRE-5Gplus enablers highlight the new capacities of security commitments – INSPIRE-5Gplus

5 Demo 3 – Moving Target Defense

The third demonstrator of the INSPIRE-5Gplus project (demo 3) showcases network slices' proactive and reactive security by using Moving Target Defense (MTD) techniques implemented on top of the 5GENESIS testbed [13]. To this end, demo 3 presents two attack scenarios for reactive MTD protection and a real-time modelling system that includes risk assessment, operational-cost evaluation, and Quality of Service (QoS) overhead for cost-efficient proactive MTD.

The enablers and assets involved in Demo 3 are listed in Table 30 and Table 31, which are fully described in D3.4 [2]:

Enabler name	Partner
Montimage Monitoring Framework	Montimage
Systemic	Solidshield
Anomaly Detection Framework	NCSRD
Optimizer for Security Functions	ZHAW
MTD Controller	ZHAW
Katana Network Slice Manager	NCSRD

Table 30: Demo 3 participating Enablers.

Asset name	Partner
Network Topology Fuzzer (TopoFuzzer) [14]	ZHAW

Table 31: Demo 3 Additional Tools.

5.1 Storyline

The objective of this Demonstrator case is the evaluation of Moving Target Defense (MTD) as an effective mechanism in improving the network's resilience against attacks, by effectively protecting network slices through dynamic reconfiguration of 5G infrastructure properties. The focus of this demonstration is the proactive change of the slice configuration and VNF deployed instances to alter the attack surface and impede pre-attack reconnaissance advantages of attackers prior to the attack stage. The cooperation of the MTD Controller and the Slice Manager will be mainly based on network slice monitoring, especially of critical slices that will trigger their reconfiguration proactively and reactively based on a defined threat and cost model.

The MTD mechanisms deployed should be adapted corresponding to the threat under consideration, ranging from no action to simple indirection or even multiple stacked indirections. The levels of MTD actions applied consider the end-user cost of applying the action in order to avoid penalizing legitimate users and make progressively the path to the protected resources more complex. In addition, MTD can protect security functions in a slice to maintain their configuration integrity and increase their robustness against reconnaissance and advanced persistent threats (APT).

An important aspect of Demo 3 is the collection and joint analysis of heterogeneous data from points of interest within the 5G infrastructure for integrated monitoring. Security Agents will act as distributed probes that will be deployed on-the-fly and adapted to changing requirements and topology. These probes will extract data from packets, flows, system and applications logs that will be subsequently used by the Security Analytics Engine and the MTD mechanism. The Security Analytics Engine will focus on detecting and classifying anomalies associated with security incidents and will inform the MTD enabler for their subsequent mitigation and resolution for protecting the deployed slices.

Demo 3 includes the scenarios and deployments that were previously defined as part of the illustrative use case (IUC) 6 in deliverable D2.2 [15], use case (UC) N in D2.3 [7] and test case 5 in deliverable D5.1 [8] HLA Mapping.

5.2 HLA Mapping

The scenarios displayed in Demo 3 and described in deliverable D2.3, iUC6 indicates the coverage of the INSPIRE-5Gplus HLA in order to provide an autonomous closed-loop proactive and reactive MTD security management and orchestration.

Deliverable D3.1 introduced the first mapping between demo 3 enablers with the HLA components, but this got extended with additional enablers that implement several other components. In total, demo 3 enablers implement the functionalities of the following HLA components:

1. The Security Data Collector: implemented by the enabler MMT for probing and network monitoring (detailed in Section 5.2.1).

2. The Security Analytics Engine: implemented by the enabler Anomaly Detection Framework (ADF) and the enabler Systemic for binary tampering detection, and pro reactive MTD operations. The enabler Optimizer for Security Functions (OptSFC) also performs threat analysis and risk assessment needed for proactive MTD operations (detailed in Section 5.2.2).

3. The Decision Engine: implemented by the enabler OptSFC for cost-efficient MTD decision making (detailed in Section 5.2.3).

4. The Security Orchestration: implemented by the enabler MTD Controller (MOTDEC) (detailed in Section 5.2.4).

5. The Service Orchestrator: implemented by the enabler Katana Network Slice Manager (also detailed in Section 5.2.4).

Figure 41 depicts the mapping:

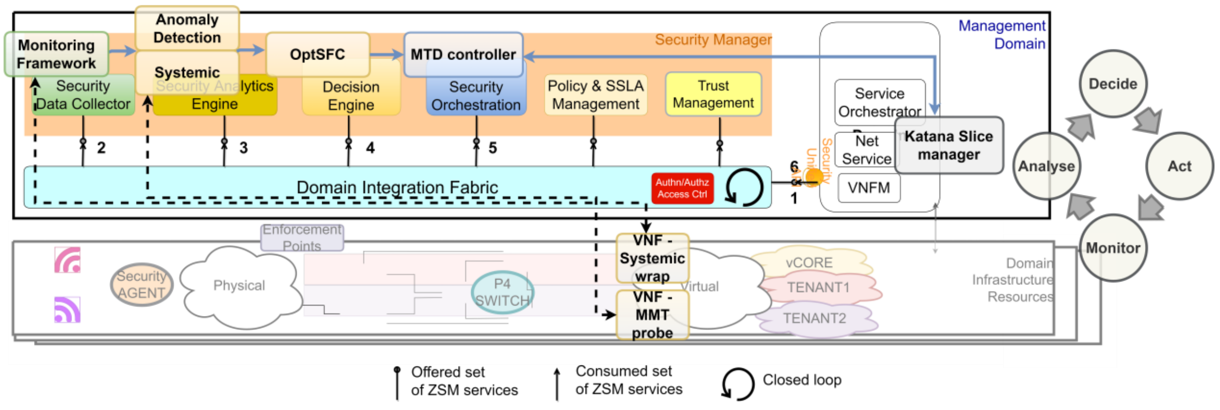


Figure 41: HLA mapping with Demo 3 enablers in one Service Management Domain.

5.2.1 Data Collection

MI Data Collector (MMT-Probe)[MI] is used in Demo 3 to capture packets in the data plane. It then analyses the QoS features that will be used by other enablers in Demo3. To do so, we installed MMT-Probe inside the machine which contains UPF so that MMT-Probe can capture all ingoing and outgoing packets of UPF. MMT-Probe uses DPI technique to extract attributes of UEs' packets that are encapsulated inside GTP protocol in the data plane. MMT-Probe calculates then the QoS attributes, such as latencies of uplink and downlink packets based on the extracted attributes. Beside the QoS attributes, MMT-Probe provides also general attributes concerning each UE, such as, number of packets, payload volumes, active sessions, etc.

MMT-Probe sends the attributes to MI Security Analytics Engine (MMT-Operator) which simply stores the attributes in a Mongo database. Other enablers can query the database to get the attributes.

5.2.2 Security Analytics

Systemic [Solidshield], through its binary re-construction process known as the wrapping process, hardens *VNFs* against confidentiality and integrity attacks, including on running executables (i.e., by memory introspection and tampering). Tampering is detected and is reported via a REST HTTP API call to a ingestion endpoint of OptSFC.

Anomaly Detection Framework (ADF) [NCSRD] handles the anomaly detection part, based on data retrieved by the Security Data Collector (MI). The core part of ADF is the Deep Learning (DL) algorithm that provides the decision on whether a flow is an anomaly or not and sends an alert to MOTDEC whenever such an anomaly is detected.

MI Security Analytics Engine is not involved in Demo 3 as a security detector but a receptor which receives the attributes from MI Data Collector, then makes them available to other enablers by storing them in a Mongo database. It also provides a graphical user interface to users to view the attributes.

Optimizer for Security Functions (OptSFC) [ZHAW] performs vulnerability checking for the threat analysis and risk assessment. This is done by using the open-source vulnerability scanner OpenVAS[9] tool that allows to configure, schedule, and manage vulnerability scans on a networked system. It scans TCP and UDP ports, fingerprint and identify running services using Common Platform Enumeration (CPE) naming, and performs active and passive vulnerability scans against maintained public and private vulnerability databases such as the Common Vulnerability Enumerations (CVEs) database and Network Vulnerability Test (NVTs) database. The first allows to find possible vulnerabilities based on the CPE of the services running in the targeted host. These scans are passive and prompt, but may contain multiple false positives. NVTs instead allow to perform active and more precise vulnerability scans using local security checks of a range of operating systems and vendors.

OptSFC schedules scans for all VNFs in one or multiple network slices, periodically and every time a VNF is reinstantiated. OptSFC then groups the CVE details of vulnerabilities based on 3 general types of threats: *1)* Advanced Persistent Threats (APTs), grouping CVEs of vulnerabilities that allow to remotely execute code or malwares detected with local security checks; *2)* Data Leak Threats, grouping CVEs of vulnerabilities that allow to gain sensitive information, such as SQL and XSS injection, directory traversals and local_file_inclusion; *3)* DoS Threats, grouping CVEs based on Buffer Overflow vulnerabilities and NVTs finding network based DoS vulnerabilities.

For each of the three major threat groups, OptSFC aggregates the Common Vulnerability Score System (CVSS) exploitability score and base score of the CVES, as well as the number of ports used by vulnerable services. This data is later used for the optimization of the decision policy used in the Decision Engine and detail in the relative next section.

5.2.3 Decision Engine

OptSFC optimizes the MTD proactive decision policy using deep Reinforcement Learning. It groups the collected data/features based on three optimisation objectives:

[9] OpenVAS - Open Vulnerability Assessment Scanner

5.2.4 Security Orchestration

1. Reduction of the operational cost of MTD: as some MTD actions like the hard MTD action (described in deliverable D3.4) need to use additional resources to reinitiate or to migrate a VNF, the operational cost of MTD actions is defined as *mtd_cost = resources_cost * deployment time.* To measure such KPI, an empirical measurement of the cost of virtual resources is done to find the coefficients between CPU cost, RAM cost, and disk cost, based on definition of *resource_cost* = β + α1 cpu+ α2 ram_gb + α3 $/hour.

2. **Improvement of the network performances (and reduction of MTD network overhead)**: as some MTD actions, like migrating a VNF to a different edge platform, requires to redirect the traffic of end users, and might affect mid-term performances of the service based on the distance of the new location, we need to consider the network performance in OptSFC decision policy. To measure it, OptSFC collects from **MI's Security Analytics Engine** the following network metrics with a regular frequency of 5 seconds for every protected VNF: number of UEs connected to a VNF, connection latency (derived from the RTT values of packets), connection throughput, packet loss rate (derived from packets' retransmission requests) and the number of packets in and out. The MTD overhead on the QoS of a VNF is defined as *mtd_QoS_overhead= (1+p_loss_rate_increase) * latency_increase* (this allows to only consider the latency increase if no packet was lost during the traffic redirection). OptSFC also considers VNF's QoS SLAs. If an MTD action violates such SLA, the overhead is increased by a penalty factor, hyperparameter of the ML-training phase.

3. Improvement of the risk assessment for proactive security:

The risk assessment for proactive security uses the data from the vulnerability scans previously described (Section 5.1.2). For each of the three major threat groups, an attack success probability (ASP) value is calculated from the relative maximum exploitability score. ASP values are increasing in time, modelling the reconnaissance advantage of attackers when they have a static target. When an MTD action is performed, the ASP is reset to its original value (*i.e.,* ASP only reconsiders the vulnerabilities a VNF has). Similar to the QoS SLAs, OptSFC considers VNF's SSLAs generalizing it in terms of a *vnf_impact* value. The security risk is then defined as *sec_risk = maximum(ASP * cvss_score) * vnf_impact* for each VNF.

The collected data is then used to have a real-time observation of the network, modeled as a Multi-Objective Markov Decision Process (MOMDP) and implemented in OpenAI Gym[10]. The MOMDP serves as the observation of the deep RL agent, where different deepRL algorithms are benchmarked as described in deliverable D3.3 [6].

5.2.5 Security Orchestrator and Service Orchestrator

MOTDEC [ZHAW]

MOTDEC implements two types of MTD actions, *soft* MTD actions and *hard* MTD actions, as described in deliverable D5.3.
To enforce both types of MTD actions, MOTDEC uses a separate module named **Network Topology Fuzzer (TopoFuzzer)**. This module allows live migration of TCP connections, not feasible with Destination Network Address Translation (DNAT). SDN redirections only work when values at the IP level are modified to change the TCP connection state machine of the new server, which causes increased latency overhead. For more efficient TCP live migrations, TopoFuzzer uses a two-sockets reverse proxy that receives the traffic via the VNFs' public IPs on one socket and establishes a dynamic connection to the VNF via its private IP.
TopoFuzzer also allows to redirect UDP traffic with simple DNAT rules as no session is maintained between the two end-points at the transport layer. TopoFuzzer creates a different redirection proxy

[10] https://www.gymlibrary.dev/

for each VNF, isolating the traffic of each VNF and each network slice. In parallel, such proxies can be implemented in Mininet hosts, allowing for lightweight scalability of the system for hundreds of VNFs. TopoFuzzer also implements a virtual network using Mininet, positioned before the redirection proxies and after the User Plane Function (UPF) of the 5G network.With this network, TopoFuzzer implements soft MTD actions such as traffic route reconfiguration, and topology modification (by removing or adding virtual switches and gateways) in a resource-efficient manner.

As Hard MTD actions have more consequent resource costs and QoS overhead but also have considerably higher security gains, demo 3 focuses on evaluating the deployment and optimization of such MTD actions, namely the re-instantiation and the migration of VNFs and NSs. These would allow to remove any infection and all ranges of malware, from spyware and botnet C&Cs to installed backdoors and ransomware (when combined with conventional microservices architectures where the main VNF service is a separate virtual resource from the database and in-memory cache).

MOTDEC is interfaced with the Katana network slice manager for the management and orchestration of the VNFs at the NFVO level, required in *Hard* MTD actions.

KATANA Network Slice Manager [NCSRD]

Katana is responsible for performing CRUD operations on slices. It receives a Network Slice Template which defines the specifications of the slice to be deployed by MOTDEC and enforces these requirements across the end-to-end 5G infrastructure, i.e., from the gNBs to the Edge/Transport and Core Network Domains.

5.3 Testbed description

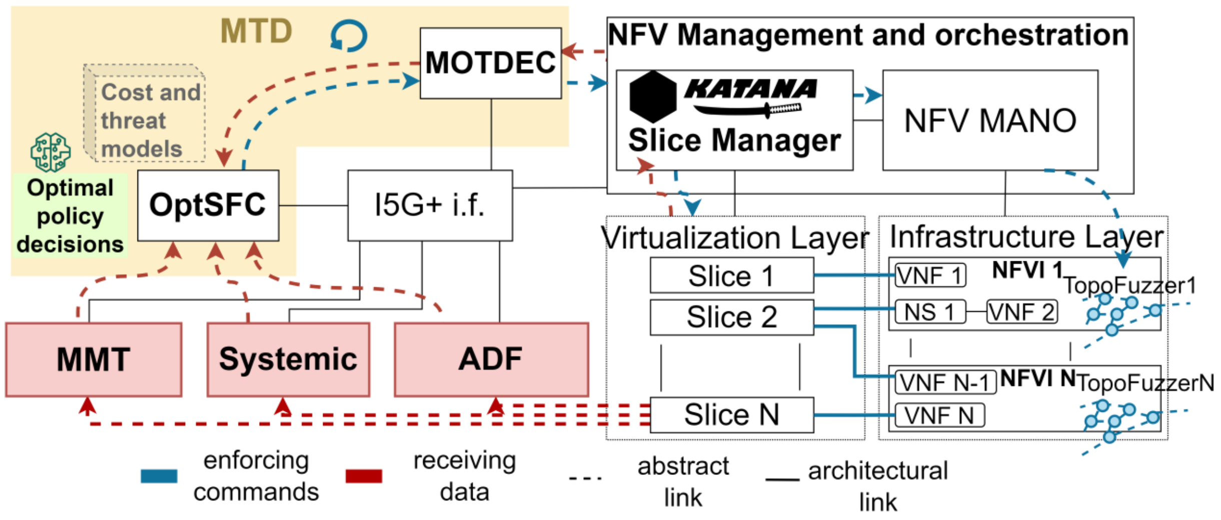


Figure 42: Demo 3 enabler's integration and architecture deployment.

The enablers are deployed and integrated on the 5GENESIS testbed in NCSRD premises, in Greece [13]. The testbed is further described in Section 5.5.2. The following lists the interfaces used for the integration of the enablers into the Demo 3 security framework:

INTERFACES:

MMT - ADF and MMT - OptSFC:

The ADF and The OptSFC obtains network measurements from the MI monitoring system every five seconds via its Mongo database which is populated by MI Data collector in near-real time.

ADF - OptSFC and SYSTEMIC - OptSFC:

Systemic performs periodic integrity checks and upon failure, sends a JSON formatted alert to OptSFC REST endpoint, via an HTTP POST request. The ADF sends anomaly alerts to OptSFC using the same REST API interface.

MOTDEC -KATANA:

MOTDEC sends a Network Slice Template to Katana to enforce the MTD action on the deployed slices. This is done with a JSON formatted Slice Template sent via HTTP POST to Katana's REST API interface.

OptSFC - MOTDEC:

OptSFC decides periodically which MTD action to perform. Then, it sends the decision to MOTDEC's REST endpoint via an HTTP GET request.

MOTDEC - TopoFuzzer:

MOTDEC interacts with the Topofuzzer to allow live migration of TCP connections and redirection of UDP traffic. MOTDEC sends HTTP POST requests to TopoFuzzer when adding and updating VNF private IPs.

5.4 Interaction diagrams/workflow

Figure 39 depicts the proactive closed-loop security management of Demo 3, characterized by the initial interactions of MOTDEC with the Katana network slice manager. 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 metrics from network probes of the MMT, 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 and model the MOMDP.

Then, proactively, based on the risk assessment formulated in the MOMDP, OptSFC proposes the MTD operation that MOTDEC will enforce, in coordination with the Katana slice manager and the NFV MANO.

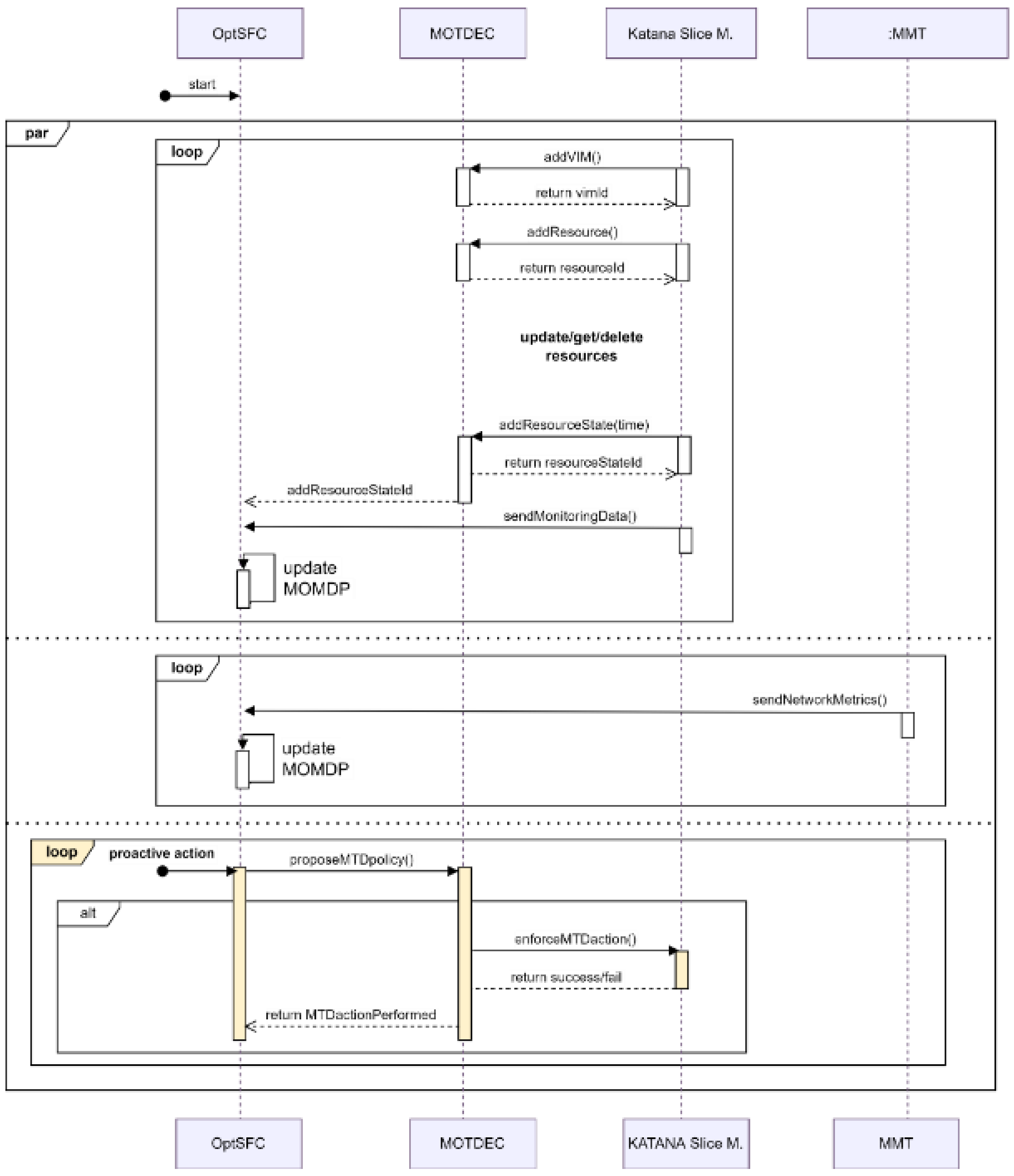


Figure 43: Proactive closed-loop workflow

In Figure 44, the workflow diagram depicts the reactive closed-loop process, triggered when tampering attack is detected by Systemic and/or when the ADF detects the anomalous traffic generated by a C&C malware. In both cases, the alert triggers an immediate reaction of OptSFC, which decides on the MTD action to perform to mitigate the attack. The MOMDP is then used to keep track of the attack and analyse we the alerts are still received or not.

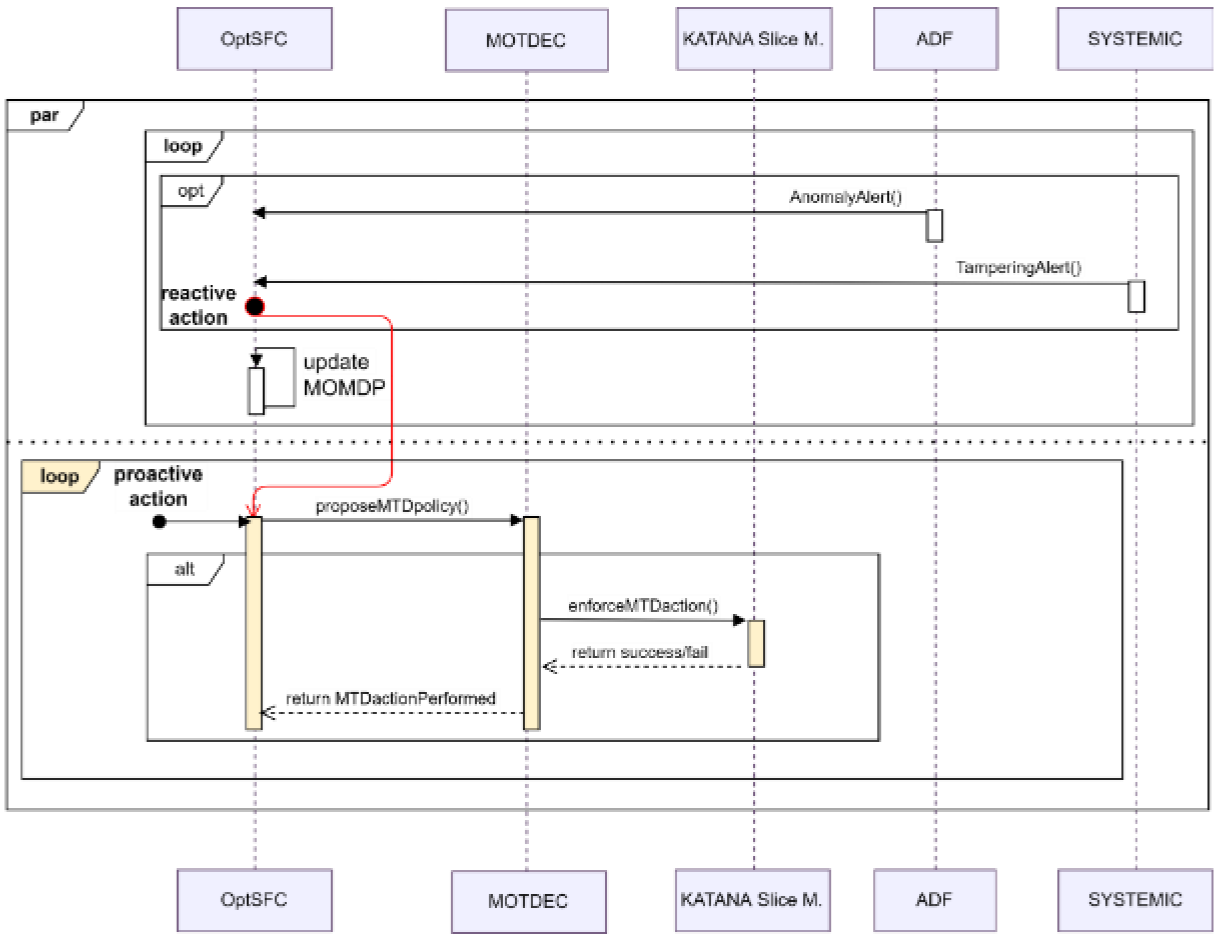


Figure 44: Reactive closed-loop workflow for C&C and tampering attack

5.5 Validation of Demo 3 KPIs

5.5.1 Demo 3 KPIs

Table 32 summarizes the Demo 3 KPIs that were measured during the experimentation period, their target values and their results. Sections 5.5.2 and 5.5.3 provide details on the measurement methodology and the conditions of measuring each KPI.

KPIs	Description	Target Value	Results
Mean Time to implement the MTD action (MTID)	How long it takes an MTD action (e.g., IP change) to be relayed to the action enforcer	MTID < 5 s.	MTD Action: VNF migration (Openstack): Migrate from Edge to Core: **1.15min** Migrate from Core to Edge: **1.57min**
MTD action cost • Worst-case (C_w) • Mean (C_m)	A comparative value showing the overhead of MTD action (example metrics to monitor change in CPU load, change in response time for the protected function)	C_w< 50% increase C_m< 20% increase	MTD Action: VNF migration with **10 connected UEs**: • Packet loss: **7% increase** (instantly, then it goes to normal values) • Latency: **100% increase in 1sec timeframe** (instantly, then it goes to normal values) MTD Action: VNF migration with **1 connected UE**: • Packet loss: **0%** increase • Latency: **0%** increase
Mean decision time for MTD action (MDTA)	The mean time it takes for the optimization engine to come up with a new MTD policy	MDTA < 500 ms.	**220ms** for DRL inference

Table 32: Demo 3 KPIs and Target Values.

5.5.2 Deployment setup for measurements

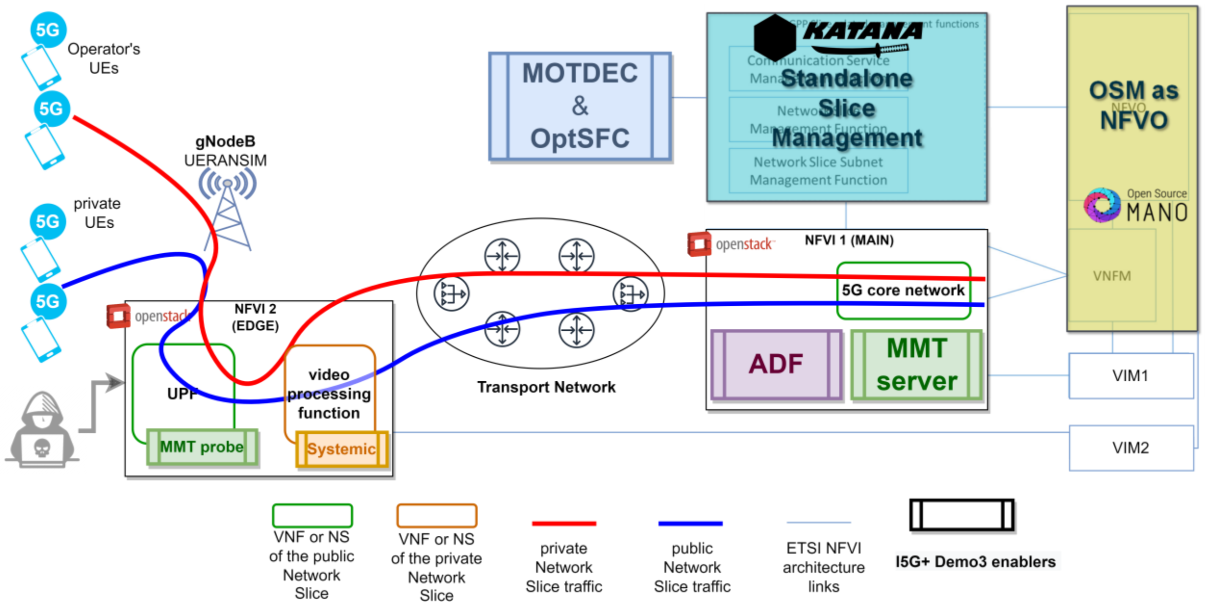

Operator's UEs
private UEs
5G
gNodeB UERANSIM
openstack NFVI 2 (EDGE)
UPF
MMT probe
video processing function
Systemic
Transport Network
MOTDEC & OptSFC
KATANA Standalone Slice Management
Communication Service Management Function
Network Slice Management Function
Network Slice Subnet Management Function
openstack NFVI 1 (MAIN)
5G core network
ADF
MMT server
OSM as NFVO
Open Source MANO
VNFM
VIM1
VIM2
VNF or NS of the public Network Slice
VNF or NS of the private Network Slice
private Network Slice traffic
public Network Slice traffic
ETSI NFVI architecture links
I5G+ Demo3 enablers

Figure 45: Demo 3 5G testbed.

To perform the KPI evaluations mentioned in the previous section, Demo 3 includes the implementation of a real 5G testbed with a core and edge platform, depicted in Figure 45. The setup showcases an infrastructure hosting two end-to-end network slices, a private one allocated by the operator's provider, and a public one managed by the operator's client. The private network slice simulates a MEC application, while the 5G core network (implemented with Open5gs) distributes its UPF on the edge platform (implemented with Openstack).

MOTDEC, OptSFC, Katana, and the NFV orchestrator (OSM) are hosted in the core platform. The TopoFuzzer and the ADF are hosted at the edge node/s, while the Systemic tampering detector is installed in each running VNF (four VNFs in total). the MI monitoring probe is installed in the same gateway resource as TopoFuzzer, thus allowing to capture both external traffic (from the UE to the Topofuzzer proxies) and external traffic (from the Topofuzzer proxies to the VNFs). UEs are emulated using UERANSIM, allowing to evaluate the scalability of the testbed with the creation of multiple 5G end-devices.

ATTACK SCENARIO

An edge VNF is used for time critical applications. The connected UEs exchange traffic with the VNF which is located behind a proxy. The attack scenario assumes that a spyware is installed on the edge VNF of interest and sends traffic to one of the connected UEs in regular time intervals. The other connected UEs exchange normal traffic throughout the attack.

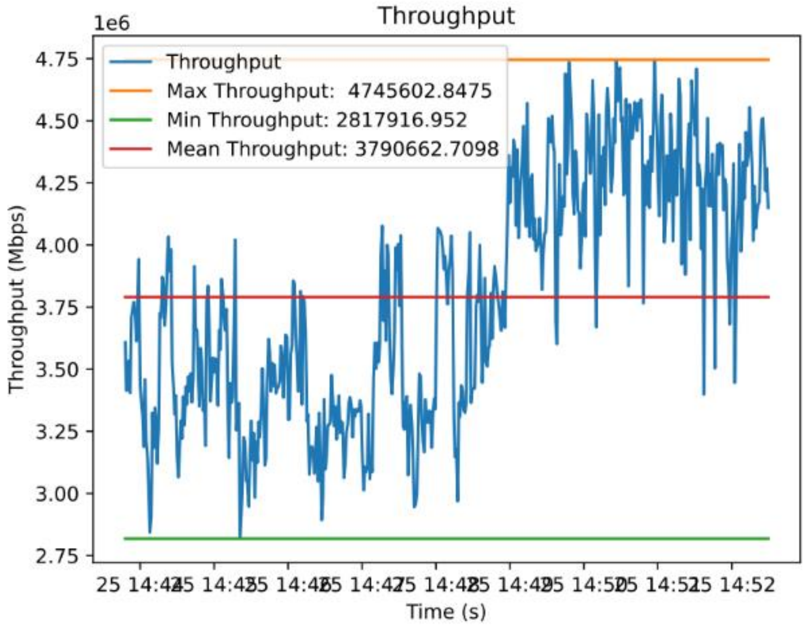
5.5.3 Results of Demo 3 KPIs

MTD Network Performance:

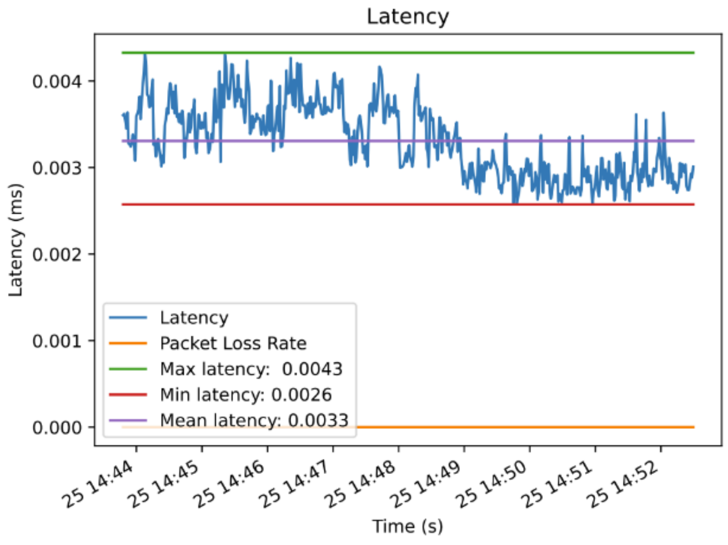
The following measurements have been taken on http/2 traffic connecting up to 10 user equipments (UEs) to the four running VNFs in the 5G testbed. UEs send traffic via the VNF's public IP. As previously detailed in section 5.1.3, the TopoFuzzer redirects the traffic to the current instance of the VNF via the private IP. This redirection is made in an average time of 0.7 milliseconds (measured for TCP traffic). In the worst case, when there is a single connected UE, from an average of 2.6 ms of RTT without TopoFuzzer,we go up to 3.3 ms when using it; this represents a 26% increase in latency. However, when connecting 10 UEs, the average latency of direct connections increases to 11.3 milliseconds, reducing the TopoFuzzer latency overhead to 10%.

This redirection is further measured when performing re-instantiation and migration Hard MTD actions. Tests alternate restart and migrate operations on a certain VNF 30 times.

Figure 46 shows the measured traffic of a VNF during a [reinstantiate - migrate - reinstantiate] sequence.



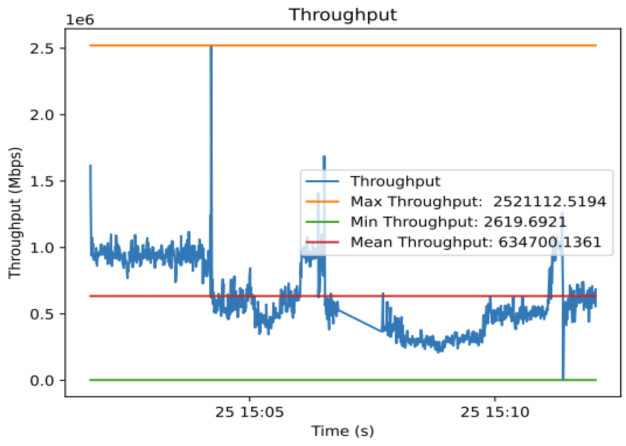
(a)



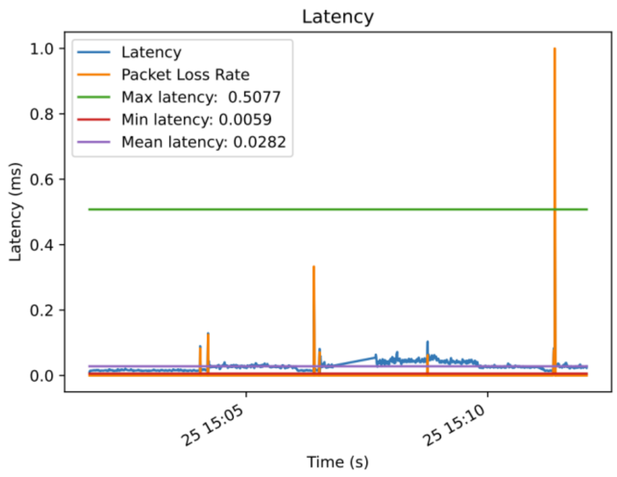
(b)

Figure 46: Throughput (a) and Latency (b) during Hard MTD actions.

The MTD migration of the VNF shows both bandwidth and QoS differences between the core and the edge locations. While the edge has better latency (second half of both measurements in figure X) due to its closeness with the UE, the core shows higher bandwidth capacity. With respect to the QoS overhead of *Hard* MTD actions, there is no packet loss or statistically noticeable latency and throughput overhead, indicating a smooth redirection of the UE connection.



(a)



(b)

Figure 47: Throughput (a) and Latency (b) during Hard MTD actions with 10 connected UEs

Figure 47 shows the http traffic of 10 UEs connected to the VNF during a [re-instantiate - migrate - re-instantiate - migrate] sequence. With about 10 connected UEs, a QoS overhead of *Hard* MTD actions becomes noticeable. Packet retransmissions happen in one second frame. For the re-instantiation, the average packet loss rate in one second frame-window (7%) is inferior to the packet-loss rate of VNF migrations (33%). Latency changes accordingly in a second time frame. However, after the migration, we notice the change in the QoS related to the core location being more distant to the UE than the edge (from an average of 0.013 we have a latency average of 0.042).

Throughput changes after the migration from the edge to the core for the same reason, and actually shows an increase of throughput after Hard MTD actions. This may be related to the retransmitted packets increasing the throughput before going back to normal traffic values.

MTD Operational Cost:

Mean Time to Implement a Hard MTD action:

The re-instantiation time of a VNF depends not only on the VNF size, but also on the hosting platform hardware. The edge node instantiates a VNF in 1.57 minutes on average, while the core platform, equipped with better CPU and disk, does it on an average of 1.15 minutes as shown in Figure 48. The restart time shows an average time of *1.88 minutes for the edge* and an average time of *1.09 minutes* *for the core*. The general average computed on 80 *Hard* MTD actions (40 migrations and 40 restarts) is *1.49 minutes*.

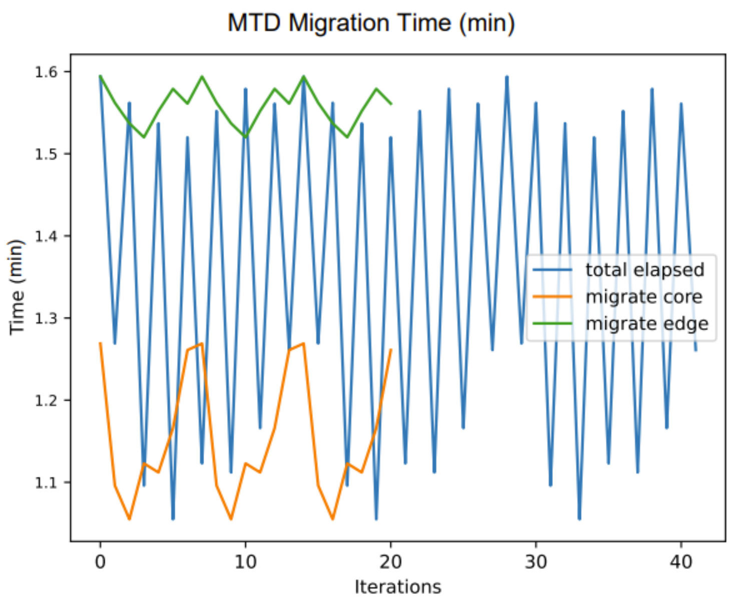


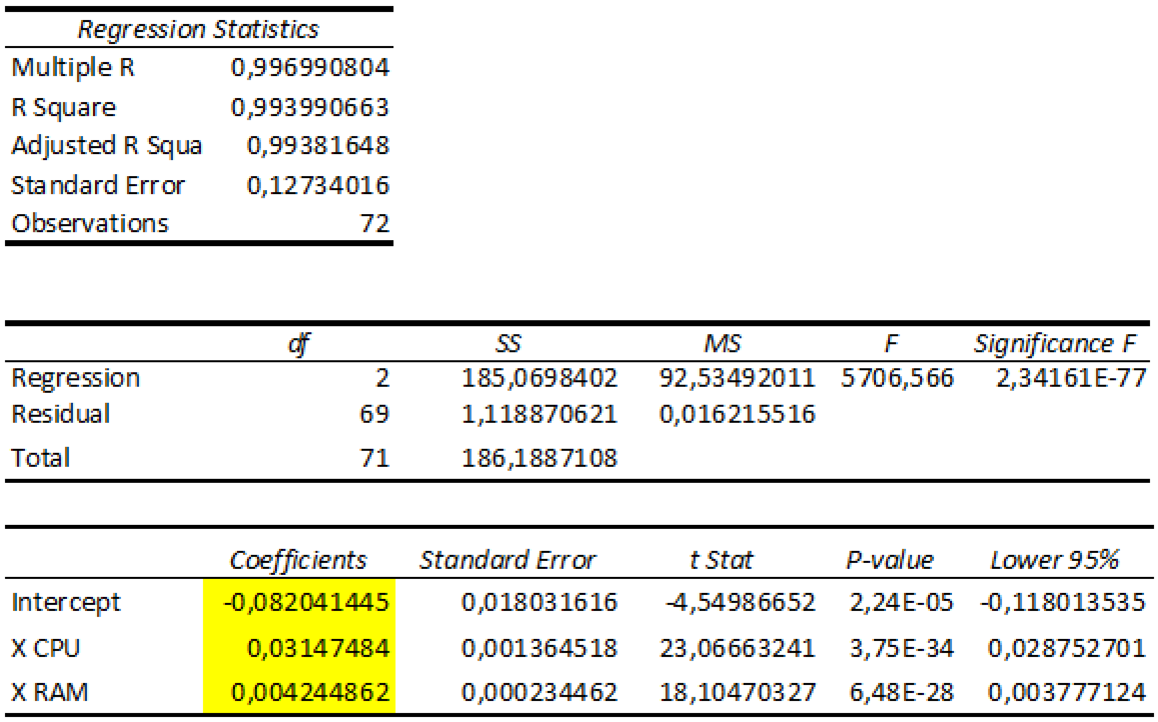
Figure 48: Mean time to implement Hard MTD actions.

MTD Resource overhead cost:

To determine the operational cost of the MTD actions, we performed an empirical study on the cost of cloud resources to identify the cost ratio between CPU, RAM and disk resources.

For simplicity, we use the cloud providers' convention of $/hour as a measurement unit for virtual resources. On 25th May 2022, we collected the prices of over 70 VM offers, ranging from 1 CPU and 0.5GB RAM to 128 CPUs and 864GB RAM, from 4 major cloud providers: AWS, AZURE, GOOGLE CLOUD, and OVH (covering at least 65% of the world market in Q1 2022[11]). We did not distinguish high-tier

[11] https://www.theregister.com/2022/05/02/cloud_market_share_q1_2022/



Regression Statistics	
Multiple R	0,996990804
R Square	0,993990663
Adjusted R Squa	0,99381648
Standard Error	0,12734016
Observations	72

	df	*SS*	*MS*	*F*	*Significance F*
Regression	2	185,0698402	92,53492011	5706,566	2,34161E-77
Residual	69	1,118870621	0,016215516		
Total	71	186,1887108			

	Coefficients	*Standard Error*	*t Stat*	*P-value*	*Lower 95%*
Intercept	-0,082041445	0,018031616	-4,54986652	2,24E-05	-0,118013535
X CPU	0,03147484	0,001364518	23,06663241	3,75E-34	0,028752701
X RAM	0,004244862	0,000234462	18,10470327	6,48E-28	0,003777124

Figure 49: Logistic Regression of prices for 65% of the cloud market in Q1 2022.

hardware from low entry ones, and as the various prices have different coefficients, a polynomial system is not solvable. We use logistic regression to find the approximate coefficients with a low P-value, showing a strong correlation between the coefficients of the different prices (see Figure 49). The final costs are 0.03147 $/h per CPU and 0.004244 $/h per GB of RAM. As cloud storage services are provided separately, the disk cost has been measured from the average of 37 storage prices from the same cloud providers, giving the final price of 0.000066 $/h per GB. These coefficients find the average cost for disk, CPU and RAM without distinction between a high-tier hardware and a low-tier one.

Deep RL training and Results on the OptSFC (optimization): Deep RL agents use a static timestep to periodically observe and act on its environment. For OptSFC, the timestep duration to analyze and decide on proactive MTD operations is fixed to 15 seconds. This value takes into consideration that the Mean Time To implement a Hard MTD action, which goes over one minute. As MOTDEC cannot enforce a new MTD action on a VNF that did not complete the implementation of the previous one, the 15 seconds allow the OptSFC to analyze and possibly apply parallel MTD operations on different VNFs. When an anomaly or attack alert is received, the deep RL agent embedded in OptSFC takes on average 220 ms to decide on the reactive MTD action to perform.

The training of a deep RL model generally requires tens of thousands or millions of timesteps. On the real observation with the data collected from the testbed, this would take around 15 x 10^6 / (3600 x 24) = 174 days, almost 6 months. This is a common RL limitation faced in fields such as robotics. One solution to this issue is to train the RL model in a simulation, and then test it in the real world.

To this end, we create a simulated environment using the network measurements and observations derived from the real 5G testbed, such as the normal traffic metrics with 1 UE, the increased values per UE, and the MTD actions QoS overhead and time to implement. The random metric values in the simulation are generated using the Gaussian distribution from the mean and the standard deviation, and bounded by their minimum and maximum values. The rewards computed in the MOMDP for the different optimization objectives are scaled and integrated in a singular reward score using hyper-parameters are used to weight each objective and give importance to one objective over the other.

Systemic measurements

Systemic requires less than 10 seconds to wrap the VNF code when this is initialized. This, together with other security solutions in the VNF, have to be considered before redirecting the traffic from the old instance to the new one when performing Hard MTD actions, avoiding giving attacker a blind spot for undetected attacks.

5.6 Conclusions and Lessons Learned

– OptSFC Deep RL optimization

Despite using model-free RL algorithms the model can still stagnate in local minina. Various methods can be applied to avoid it, such as adding Bayes Net Noises. The dynamic growth of networks is properly simulated but the dynamic growth of the action space is learned through the reward system, adding complexity to the optimization task of the RL to an additional objective. To resolve these issues, action masking methods [5] could be applied to remove the action space problem from the actual objectives the RL agent needs to optimize.
The trained Model however is able to find an optimal strategy and outperforms random MTD operations when dealing with cost-efficient policies.

– MOTDEC

MOTDEC redirection method, using TopoFuzzer, solves the issue of live TCP connection handovers with negligible QoS overhead. A drawback of a proxy-based redirection when it comes to security is the management of TLS certificates to allow the authentication of the proxy with the VNF's UE clients. This is de-facto accepted limitation as all cloud providers offer proxy services that require the sharing of TLS certificates. However, with the advent of the new http/3 protocol standard, built on top of the QUIC protocol, this limitation is dropped as sessions and TLS handshakes are entirely handled at the application layer. This is demonstrated by the successful redirection of https traffic based on http/3 with TopoFuzzer.

6 Validation of INSPIRE-5Gplus HLA by the Demos

6.1 Demo 1 validation of the INSPIRE-5Gplus HLA

Demo 1 validates the HLA through proactive and reactive stages of ZSM-aligned closed-loops, locally and E2E. Two different SSLAs and slice information are used for this purpose. The enforcement of both of them validates proactive and reactive stages of the autonomous closed-loops, involving more than 20 enablers/assets across 6 SMDs that map directly with HLA functional blocks (Figure 5). On the one hand, the first SSLA/slice info requires the deployment of a 5G network as a service which must also be secured under certain requirements, such as: protection of the communication channel with encryption, protection of the 5G core against cryptomining and protection of V2X services against DDoS. SSLA and slice info are refined into operator policies which will be transformed into several Orchestration Policies, one for each domain involved, and each of these policies will represent a sub-slice belonging to the E2E slice. Those policies are then orchestrated across specific SMDs to satisfy the requirements. These processes involve enablers instantiated at the E2E SMD that implements E2E Policy & SSLA Management, E2E Security Orchestrator, E2E Trust management, Data Services and Integration Fabric functionalities. Once each SMD receives their per-domain security policy, a trust-based orchestration and the local closed loop will deploy and configure each of the elements specified in the policies, in an appropriate and orderly manner. These processes involve enablers at SMD level that implements SMD Security Orchestrator, Policy & SSLA Management, Trust management, Data Services, Integration fabric, Service Management Functions, Security Data Collector, Security Analytics Engine and Decision Engine functionalities. On the other hand, the second SSLA focuses on securing the sensors of a private 5G IoT network and the IoT Broker by securing the communication channels and DDoS protection. Once both SSLAs have been deployed the system is accomplishing the security requirements, thus if a security requirement violation occurs, Security Data Collectors together with the Security Analytics Engine will trigger a reactive closed-loop which starts with the Decision Engine elaborating policy-based countermeasures that will be send to the Security Orchestrator and subsequently the countermeasures will be applied in order to mitigate the attack and maintain the SSLA. If required, countermeasures are scaled to the E2E Domain in order to perform actions on potential affected domains. Demo1 also validates the E2E reaction by using threat knowledge received from the SMDs and enforcing different countermeasures across the whole multi-domain infrastructure.

Security Req. ID	Security Requirement Description
SEC-REQ-01	The 5G network has monitoring agents capable of extracting relevant information from the 5GCore, 5G RAN, 5GServices and Security Services in order to provide relevant information of the correct functioning of the system and serve as input to the auditing services capable of detecting anomalies and trigger the reactive mitigations.
SEC-REQ-02	The 5G network has the UDR and UDM that maintain a database of subscribers, that when the AMF receives a Registration Request will perform the Registration Procedure verifying the subscriber data, and only allowing the access to the network when it has a valid

Security Req. ID	Security Requirement Description
	subscription.
SEC-REQ-03	The 5G network is based on NFs containerized deployment enabling the scalability and management operation of the NFs, grating an adaptation to the requirements specified.
SEC-REQ-04	The 5G network through the deployment of different sub-slices as part of an E2E Network Slice allocate the required resources to ensure the correct operation of the deployed services in each of the domains that have to pass through.
SEC-REQ-05	The 5G network has enforced the security capabilities specified by the customer in the SSLA to fulfil the security level required by the vertical services in addition to other security capabilities native derived from the nature of 5G Networks.
SEC-REQ-06	The 5G network coordinate the 5G Management Procedures with the Docker NFV operations which enables the management of the entities following the 3GPP standard.
SEC-REQ-07	The 5G network uses the 3GPP standards to perform operations on the managed entities and users to recover from an attack keeping the services actives and minimizing the impact in case that a service should be migrated or reinstantiated (e.g., AMF Planned Removal TS 23.501 5.21.2.2).
SEC-REQ-08	The 5G network's security mechanisms monitors and protect confidentiality, integrity and the proper operation of the services without degrading the quality nor affecting to the functional requirements of the services offered.
SEC-REQ-09	The security mechanisms of the 5G network are deployed as a VNFs which leverages the abstraction requirements of the virtualization to be deployed in any hardware that supports this technology.
SEC-REQ-10	The 5G network is supported by the Trust Reputation Manager which maintains updated a repository of trust calculations of the NFVs deployed with associated version a location.
SEC-REQ-12	The 5G ecosystem through the monitoring agents is capable of extract KPIs of the infrastructure to guarantee that in every moment the SSLA contracted with the costumer is met.
FC-REQ-01	Demo 1 through the instantiation of monitoring agents (e.g., MMT Tool, PoT...) is capable to extract telemetry data from the infrastructure.
FC-REQ-02	When specified in the SSLA, Demo 1 through the instantiation of Security Analytics Engine agents ensure the premature detection of security breaches by analyzing the data collected from the security functions or the infrastructure.

Security Req. ID	Security Requirement Description
FC-REQ-03	If due to a vulnerability or attack the SSLA is compromised, Demo 1 will trigger the reactive closed-loop to mitigate the attack through security policies and re-establish the fully enforcement of the SSLA requirements in run time.
FC-REQ-04	Demo 1 through the escalation is capable of coordinate multiples domains to perform required security actions.
FC-REQ-06	Demo 1 monitors the KPIs of the deployed network slices, performing an anomaly detection over anomaly results and a anomaly prediction in other domains susceptible to similar attacks
FC-REQ-07	Demo 1 is capable of monitoring different layers of the infrastructure, from the Link layer to the Application Layer or Virtualization Layer.
FC-REQ-08	Demo 1 through the different enablers is capable of assess compliance with the SSLA, Security Analytic Engine to detect any violation and Decision Engine and Security Orchestrator to deploy countermeasures to maintain the compliance.
FC-REQ-09	Demo 1 deploys by demand Security Analytics Engine which receives the information gathered by monitoring agents and perform security analysis, in case any incident or threat is detected an alert is raised to the Decision Engine to elaborate further actions.
FC-REQ-10	Demo 1 defines the SSLA as its High-Level Abstraction Policy which will be refined into MSPL-OP and eventually translated into final security asset configurations, following the ZSM policy levels scheme.
FC-REQ-11	Demo 1 framework support Security Analytics Engines as STA to detect anomalies over encrypted traffic.
FC-REQ-12	Demo 1 shall support passive access to continuous up-to-date traffic in the network.
FC-REQ-13	Demo 1 monitoring agents that perform the role of Security Data Collector store the information retrieved from the infrastructure and services and maintain updated to further analysis. The E2E Trust Reputation Manager receives trust value data from different domains and aggregates in a unique trust values repository.
FC-REQ-14	Demo 1 Security Analytics agents can retrieve monitored information pre-applying filters to select only valuable information depending on the type of analysis.
FC-REQ-15	Demo 1 through the Security Orchestrator with the policy-based orchestration and different VNF MANOs deploys VNF in a virtualized infrastructure capable to chain the VNF to offer E2E Secured Services.
FC-REQ-16	Demo 1 through the policy-based orchestration is capable of configure deployed VNFs adapting the configuration to the environment

Security Req. ID	Security Requirement Description
	and specific requirements (e.g., IPSec Channel Protection with specific crypto suite).
FC-REQ-17	Demo 1 framework performs a validation of certain security assets to ensure that they are performing required capability.
FC-REQ-18	Demo 1 through the Slice Manager, Policy Framework and Decision Engine is capable of generating Security Policies of different level of abstraction.
FC-REQ-19	Demo 1 uses MSPL-OP and have support of HSPL-OP, that are policy languages with the required flexibility and specificity to declare services with security associated.
FC-REQ-20	Demo 1 through the Policy Framework and its main building blocks is capable of performing required policy management operations.
FC-REQ-21	Demo1 manages (retrieve, translate, enforce, remove) different policy models (5G security slice, Monitoring, Channel protection, Proof of transit, Filtering, Secured Service MANO).
FC-REQ-22	Security policy conditions are analysed and considered during translation and orchestration processes to generate and enforce configurations according to them since conditions and actions models the desired behaviour of the system.
FC-REQ-23	E2E Security Orchestrator and Security orchestrator prepares de orchestration and enforcement plan to decide on security policy execution.
FC-REQ-24	Security policy actions are analysed and considered during translation and orchestration processes to generate and enforce configuration actions according to the conditions.
FC-REQ-25	Policy conflict detector service (E2E and SMD) provides policies conflict and dependencies detection capabilities. The service is invoked during orchestration and enforcement stages but it can be invoked as standalone process.
FC-REQ-26	Security Orchestrator implements NFV-MANO/Slice-MANO drivers to interoperate with the NFV MANO APIs for management of NFV network services.
FC-REQ-27	Security Orchestrator retrieves network slice status information multiple times during the orchestration and enforcement process. New services information is collected in data services. OSM provides network slice status information.
FC-REQ-28	The closed-loop implementation allows demo1 detect and mitigate different kind of attacks, including those related to performance degradation such as Cryptomining or DDoS attacks.

Security Req. ID	Security Requirement Description
FC-REQ-29	Demo1 provides specific automated management for compute, storage, and network resources, VNFs, slices and services for an automated MTD operation. It corresponds to the SSLA2 part of the demo.
FC-REQ-30	STA, V2X DDoS and MMT are able to detect abnormal behaviours of the managed networks and services. For instance, in demo1 STA detects Cryptomining attack in the 5G Core as part of the abnormal behaviour detection.
FC-REQ-31	Collected telemetry data are governed at different points, e.g., MI Security analytics, STA, and data services.
FC-REQ-32	Data services/Integration fabric provide common access to the collected up-to-date telemetry data. Event steaming platform provides common topics to retrieve telemetry data.
FC-REQ-34	STA, V2X DDoS, MMT and TRM store historical data needed for the prediction and analytics.
FC-REQ-35	Integration fabric/data services allows the collection of data from ZSM managed entities to perform automated network and service security management based on AI.
FC-REQ-36	Decision Engine provide reactive security policies according to the detected threat that are valuable to a security analyst.
FC-REQ-37	Trust Reputation Manager keeps updated trust values to provide automated security assessment based on identified vulnerabilities.
FC-REQ-38	STA, V2X DDoS and MMT tools include threat identification based on best practices, network configurations, user activities.
FC-REQ-39	STA, V2X DDoS and MMT tools use network related information and other relevant information for security analysis.
ZFC-REQ-01	STA and V2X DDoS enablers deployed in demo1 provide root cause identification.
ZFC-REQ-02	Automated management (i.e., detection, identification, prevention, and mitigation) of Cryptomining, DDoS and crypto-suite downgrade security incidents/attacks. Different automated mitigations are provided at SMD and E2E SMD levels; Trust-based redeployment, filtering (V2X SMD and RAN SMD) and Moving Target Defense.
ZFC-REQ-03	Automated security management is based on AI/ML techniques provided by STA, V2X DDoS, MMT security enablers.
ZFC-REQ-04	Closed-loop security management is supported at SMD and E2E SMD levels. Proactive and reactive parts of the closed loops are demonstrated across 5 different SMDs.
ZFC-REQ-05	Most of the involved entities support open interfaces. OpenAPI interfaces were provided.

Security Req. ID	Security Requirement Description
ZFC-REQ-06	Access control to services exposed by the security management domains is supported by Istio (as well as by the services themselves) as part of the integration fabric.
ZFC-REQ-07	It supports security management of end-to-end services that cross boundaries between multiple domains. Different E2E policies enforcement is provided across different domains.
ZFC-REQ-08	Decision Engine support bounding the automated decisions-making by the established SSLA and security policies. Reactive security policies are available at the policy repository.
ZFC-REQ-09	Security management services provided can be registered in the integration fabric. New instantiated services are registered automatically.
ZFC-REQ-10	New instantiated services are registered automatically. Besides, Integration fabric/data services provide services end point information.
ZFC-REQ-11	Demo1 Integration fabric support the capability to invoke the discovered security management services.
ZFC-REQ-12	Demo1 Integration fabric support the capability of communication between the security service producers and the security service consumers in different ways (e.g., Rest APIs, Websocket, Event streaming platform).
ZFC-REQ-13	V2X enabler, integrity of incoming data traces is checked and validated in the analytics engine where we deep dive into the semantics of the messages. STA and PoT enabler verify the integrity, the first one using a specific format to send the alerts to the DE and the PoT by establishing a format with gRPC between the agents and the controller.

Table 33: Demo 1 INSPIRE-5GPlus Requirements

Functional Module	Service
Security Data Collector (SDC)	Data Collection Service
Security Analytics Engine (SAE)	Anomaly Detection Service
	Root Cause Analysis Service
Decision Engine (DE)	Service Policy Proposal Service
	Security Asset Priority Service

Security Orchestrator (SO)	Security Policy Enforcement Service
	MSPL/TOSCA Refinement Service
	Security Policy Storage Service
	Policy Conflict Detection Service
	SSLA Storage Service
Trust Management (TM)	Trust Reputation Manager
E2E Decision Engine (E2E DE)	Security Policy Synchronization Service
	Security Policy Proposal service
	Security Asset Priority Service
E2E Security Orchestrator (E2E SO)	Security Policy Enforcement Service
E2E Policy and SSLA Management (E2E PSM)	Security SLA Refinement Service
	Policy Conflict Detection Service
	Security Policy Storage Service
E2E Trust Management (E2E TM)	Trust Reputation Manager Service
Domain-Level & Cross-Domain Data Services	Data Access Service
Integration Fabric (IF)	Registration Service
	Discovery Service
	Invocation Service
	Communication service
Security Agent (SA)	Enforcement Point Service
	Network Monitoring and Telemetry Service
Unified Security API	Network Service Actions List
Service Management Domain (SMD)	Network Slicing Management
E2E Service Management Domain (E2E SMD)	Network Slice Brokering

Table 34: HLA Services Provided by Demo1

6.2 Demo 2 validation of the INSPIRE-5Gplus HLA

Demo 2 illustrates the HLA with a perspective of establishing transparency, understandability and liability-awareness. Actually, Demo 2 and associated collaborative work have introduced the concept of transparency to allow third party to control the effectiveness or reality of security service commitments (SLAs). Automation, as demonstrated in Demo1 and transparency, as demonstrated in Demo2 aspects are complementary. In practice, Demo 1 demonstrates the capacity, by leveraging the Close loop approach, to operate a specific SLA over multi-technical domains. In contrast, Demo2 allows external third parties (i.e., critical Verticals under NIS2 Directive) to be in capacity to delegate some of their security constraints. In that sake, specific control means are allocated to these verticals to measure and qualify the effectiveness of SLA committed on demand and in real time.

As a result of the two Liability workshops organized within INSPIRE5GPlus project, we have established that a Client needs to control and to monitor that committed SLAs are effective and delivered as contractualized, irrespectively to whatever security measures be taken by the service provider. As a matter of fact, the client does not directly consider and/or has the visibility on the technical means used to deliver the service (e.g., security orchestration, machine learning).

Demo 2 enablers specifically foster the capacity to elaborate the effectiveness of the security proposed service. In particular, Systemic novel feature of deep monitoring used in combination with the deep attestation framework establishes the effectiveness of the isolation properties (i.e., security by orchestration), the effectiveness of IoT campus monitoring system hardening and current execution (i.e., Systemic hardening and monitoring of MMT probe service).

INSPIRE5GPlus project enablers have all their utility and position in the HLA and its trust and liability management component. Demo2 exemplifies a new way to operate security constraints delegation to infrastructures, and then clear the way for HLA evolution towards multi-parties and multi domains infrastructures used for SLAs delivery to Client. In future extension, HLA would have to integrate new tools to model precisely one-to-one stakeholders' liability and inter SLAs commitments, to control and monitor effectiveness with proven and accepted evidences the committed security services between parties to deliver the operate the whole Client service (with respect to their regulations and SLA commitment).

Security Req. ID	Security Requirement Description
SEC-REQ-01	Demo_2 has monitoring agents capable of extracting relevant information from the proposed domains, in order to provide relevant information of the correct functioning of the system.
SEC-REQ-03	The Demo_2 is based on NFs containerized deployment (over Kubernetes) enabling the management operation of the NFs, grating an adaptation to the requirements specified.
SEC-REQ-08	The Demo_2 's security mechanisms monitor and protect confidentiality, integrity and the proper operation of the services without degrading the quality nor affecting to the functional requirements of the services offered.

Security Req. ID	Security Requirement Description
SEC-REQ-12	The Demo_2 system through the monitoring agents is capable of extract KPIs of the infrastructure to guarantee that in every moment the SSLA contracted with the costumer is met.
FC-REQ-01	Demo 2 through the instantiation of monitoring agents (e.g., MMT Tool, attestation agent...) is capable to extract telemetry data from the infrastructure.
FC-REQ-05	Demo_2 supports the capability to ensure only validated/certified resources should be used isolated.
FC-REQ-07	Demo 2 is capable of monitoring different layers of the infrastructure, from the Link layer to the Application Layer or Virtualization Layer (depending of the capacity of deployed attestation agents).
FC-REQ-08	Demo 2 through the different enablers is capable of assess compliance with the SSLA of isolation.
FC-REQ-17	Demo 2 performs a validation of certain security assets to ensure that they are performing required capability.
FC-REQ-30	MMT are able to detect abnormal behaviours of the managed networks and services. For instance, misbehave of several IOT components over an industrial campus.
FC-REQ-31	Collected telemetry data are governed at different points, e.g., MI Security analytics and data services.
FC-REQ-39	MMT tools use network related information and other relevant information for security analysis.
ZFC-REQ-03	Security monitoring and detection is based on AI/ML techniques provided by MMT security enablers.
ZFC-REQ-05	Most of the involved entities support open interfaces. OpenAPI interfaces were provided.
ZFC-REQ-06	Access control to services exposed by the security management domains is supported by Istio (as well as by the services themselves)
ZFC-REQ-07	It supports security management of end-to-end services that cross boundaries between multiple domains owned by different legal entities.
ZFC-REQ-12	Demo2 supports the capability of communication between the security service producers and the security service consumers in different ways (e.g., Rest APIs, Event streaming platform).
ZFC-REQ-13	Demo 2 supports the capability to check / validate the integrity of telemetry data.

Table 35: Demo 2 INSPIRE-5GPlus Requirements

Functional Module	Service
Security Analytics Engine (SAE)	Root Cause Analysis Service detection of critical event in IOT campus
Decision Engine (DE)	Service Policy Proposal Service activation of specific SLA of isolation or critical mode protection
Security Orchestrator (SO)	Security Policy Enforcement Service the isolation of operated through a specific orchestration under constraints of physical resources
E2E Policy and SSLA Management (E2E PSM)	Security SLA Refinement Service manual activation of critical SLA over domains
Security Agent (SA)	Enforcement Point Service Attestation agent deployed over Kubernetes PoD
Service Management Domain (SMD)	Network Slicing Management Security by orchestration of the Client slice when activating critical isolation SLA

Table 36: HLA Services Provided by Demo 2

6.3 Demo 3 validation of the INSPIRE-5Gplus HLA

Demo 3 focuses on the Moving Target Defense paradigm, so it highlights the security, functional and non-functional requirements, as well as the services of the INSPIRE-5Gplus HLA that enable the proactive security of the underlying infrastructure. Table 37 summarizes the requirements addressed by Demo 3 and Table 38 highlights the HLA services required to run this Demo. Demo 3 requirements cover successive steps of the overall workflow: *i)* data telemetry, i.e., data collection, processing, *ii)* anomaly detection, *iii)* optimal policy selection by the decision engine, *iv)* orchestration of the selected policy, *v)* network slice update/re-instantiation, and *vi)* verification of the deployed images (where services are running).

Each step is implemented by participating enablers of Demo 3:

1. Data collection and processing: MMT
2. Anomaly detection: ADF, Systemic

3. Optimal Policy selection: OptSFC
4. Network Orchestration: MOTDEC
5. Network Slice update/re-instantiation: Katana Slice Manager
6. Verification of the deployed images: Systemic

Security Req. ID	Security Requirement Description
SEC-REQ-01	The 5G network shall provide telemetry and other auditing information relevant to the security mechanisms of the system.
SEC-REQ-02	The 5G network shall only allow authenticated users to consume the services provided by the 5G system.
SEC-REQ-03	The 5G network shall warrant measurable level of availability of its services to the relevant stakeholders.
SEC-REQ-04	The 5G network shall ensure the necessary network capacity and network resources for the critical operations of the 5G services.
SEC-REQ-05	The 5G network shall enable a secure platform for vertical services to be deployed.
SEC-REQ-06	The 5G network shall enable the state management of its platform components.
SEC-REQ-07	The 5G network shall be able to revert to previous states with minimal service disruption of deployed application in case of malicious compromise.
SEC-REQ-08	The 5G network’s security mechanisms should not impact the functional requirements of critical operations for vertical applications.
SEC-REQ-09	The security mechanisms of the 5G network shall be able to be deployed in any potential 5G hardware provider without any impact on their performance or functionality.
SEC-REQ-10	The security mechanisms of the 5G network shall be able to measure/evaluate trust level of its components and platforms and share this information with verticals in a safe and trustable way.
SEC-REQ-12	The 5G ecosystem shall be able to publish security KPI measuring the compliance of stakeholder with their Security Level Commitments.
FC-REQ-01	INSPIRE-5Gplus framework shall support the capability to collect up-to-date telemetry data.

Security Req. ID	Security Requirement Description
FC-REQ-04	INSPIRE-5Gplus framework shall support the capability to allow multi-domain interaction.
FC-REQ-05	INSPIRE-5Gplus framework shall support the capability to ensure only validated/certified resources should be used.
FC-REQ-06	INSPIRE-5Gplus framework shall support the capability to perform anomaly prediction based on the required KPIs of the managed network slices
FC-REQ-07	INSPIRE-5Gplus framework shall support the capability to monitor different structured data (network, operating systems, applications, nsi).
FC-REQ-09	INSPIRE-5Gplus framework shall support the generation of alerts that can be processed by the Security Orchestrator.
FC-REQ-11	INSPIRE-5Gplus framework support Security Analytics Engines as STA to detect anomalies over encrypted traffic.
FC-REQ-12	INSPIRE-5Gplus framework shall support passive access to continuous up-to-date traffic in the network.
FC-REQ-15	INSPIRE-5Gplus framework shall support the capability to automatically deploy virtualized network functions software (including virtualized security functions)
FC-REQ-16	INSPIRE-5Gplus framework shall support automatic configuration of virtualized network function parameters (including virtualized security functions)
FC-REQ-17	INSPIRE-5Gplus framework shall support the capability of automatic verification of virtualized network functions normality after deployment.
FC-REQ-26	INSPIRE-5Gplus framework shall support the interoperation with the NFV MANO APIs for management of NFV network services.
FC-REQ-27	INSPIRE-5Gplus framework shall support the collection of data on network slices status.
FC-REQ-28	INSPIRE-5Gplus framework shall support the capability to take action to mitigate performance degradation due to security issues.
FC-REQ-29	INSPIRE-5Gplus framework shall support automated management for compute, storage, and network resources, VNFs, slices and services for an automated MTD operation.
FC-REQ-30	INSPIRE-5Gplus framework shall support the detection of abnormal behaviours of the managed networks and services.
FC-REQ-32	INSPIRE-5Gplus framework shall support the capability to common access to the collected up-to-date telemetry data.
FC-REQ-34	INSPIRE-5Gplus framework shall support the capability to store historical data needed for the prediction and analytics.

Security Req. ID	Security Requirement Description
FC-REQ-36	Decision Engine provide reactive security policies according to the detected threat that are valuable to a security analyst.
FC-REQ-39	STA, V2X DDoS and MMT tools use network related information and other relevant information for security analysis.
ZFC-REQ-02	INSPIRE-5Gplus framework shall support automated management (i.e., detection, identification, prevention, and mitigation) of security incidents/attacks.
ZFC-REQ-03	INSPIRE-5Gplus framework shall support automated security management based on AI/ML techniques.
ZFC-REQ-04	INSPIRE-5Gplus framework shall support closed-loop security management.
ZFC-REQ-05	INSPIRE-5Gplus framework shall support open interfaces.
ZFC-REQ-07	INSPIRE-5Gplus framework shall support security management of end-to-end services that cross boundaries between multiple domains.

Table 37: Demo 3 INSPIRE-5GPlus Requirements

Functional Module	Service
Security Data Collector (SDC)	Data Collection Service
Security Analytics Engine (SAE)	Anomaly Detection Service
Decision Engine (DE)	Service Policy Proposal Service
Security Orchestrator (SO)	Security Policy Enforcement Service
Domain-Level & Cross-Domain Data Services	Data Access Service
Security Agent (SA)	Network Monitoring and Telemetry Service
Service Management Domain (SMD)	Network Slicing Management

Table 38: HLA Services Provided by Demo3

7 Conclusions

This deliverable showcased the results of the three INSPIRE-5Gplus Demonstrators. These Demonstrators include representative scenarios of activities and attacks in future networks and demonstrate the feasibility of key technologies in protecting their services and their consumers.

Demo 1 showcased the feasibility of the INSPIRE-5Gplus HLA and the closed-loop management in two scenarios with different SSLAs. Demo 2 investigated and presented the need for Trust and Liability in an ever-connecting network environment. Demo 3 presented the feasibility of AI and MTD as key technologies for the proactive protection of deployed slices considering cost as a significant constraint in the operations.

All Demonstrators showcased that INSPIRE-5Gplus has completed its objective: to provide a zero-touch security framework made from advanced enabling technologies, e.g., emerging AI techniques, ZSM, TEE, MTD, considering the requirements of advanced 5G use cases. The feasibility of its provided solution was demonstrated in complex scenarios that will be part of upcoming deployments. The most important result of this activity was that INSPIRE-5Gplus completed its objective and delivered a high-end security framework with an extensive range of features that will act as enablers for more research and experimentation in upcoming Beyond 5G networks.

References

[1] INSPIRE5G-plus Consortium, “D5.2 First 5G security testing infrastructure implementation and preliminary results,” October 2021. [Online]. Available: https://www.inspire-5gplus.eu/wp-content/uploads/2021/11/i5-d5.2_first-5g-security-testing-infrastructure-implementation-and-preliminary-results_v1.0.pdf.

[2] INSPIRE-5Gplus Consortium, "D3.4 Smart 5G Security," July 2022. [Online]. Available: https://www.inspire-5gplus.eu/wp-content/uploads/2022/07/i5-d3.4_smart-5g-security_v1.0.pdf.

[3] INSPIRE-5Gplus Consortium, "D2.4 Final Report on Enablers and Mechanisms for Liability-aware Trustable Smart 5G Security Management Framework," October 2022. [Online]. Available: https://www.inspire-5gplus.eu/wp-content/uploads/2022/11/i5-d2.4_final-report-on-enablers-and-mechanisms-v1.0.pdf.

[4] INSPIRE-5Gplus Consortium, "D3.1 5G security assets baseline and advancements," May 2021. [Online]. Available: https://www.inspire-5gplus.eu/wp-content/uploads/2021/05/i5-d3.1_5g-security-assets-baseline-and-advancements_v0.7.pdf.

[5] INSPIRE-5Gplus Consortium, "D4.1 Trust mechanisms for 5G environments," August 2021. [Online]. Available: https://www.inspire-5gplus.eu/wp-content/uploads/2021/09/i5-d4.1_trust-mechanisms-for-5g-environments_v1.0.pdf.

[6] INSPIRE-5Gplus Consortium, “D3.3 5G security new breed of enablers,” March 2022. [Online]. Available: https://www.inspire-5gplus.eu/wp-content/uploads/2022/03/i5-d3.3_5g_security_new_breed_of_enablers_v1.0.pdf.

[7] INSPIRE-5Gplus Consortium, "D2.3 Final Report on Advanced 5G Security Use Cases," October 2022. [Online]. Available: https://www.inspire-5gplus.eu/wp-content/uploads/2022/05/i5-d2.3_final-report-advanced-5g-security-uc_v1.0.pdf.

[8] INSPIRE5G-plus Consortium, "D5.1 5G security test cases," February 2021. [Online]. Available: https://doi.org/10.5281/zenodo.4569524.

[9] INSPIRE-5Gplus Consortium, "D4.2 Trust management in multi-tenant/multi- party/multi-domain 5G environment," May 2022. [Online]. Available: https://www.inspire-5gplus.eu/wp-content/uploads/2022/05/i5-d4.2_trust-management-in-multi-tenant-5g-environment_v1.0.pdf.

[10] INSPIRE-5Gplus Consortium, "D4.3 Liability mechanisms for 5G environments," October 2021. [Online]. Available: https://www.inspire-5gplus.eu/wp-content/uploads/2021/11/i5-d4.3_liability-mechanisms-for-5g-environments_v0.91.pdf.

[11] INSPIRE-5Gplus Consortium, "D4.4 Liability management in a 5G environment," August 2022. [Online]. Available: https://www.inspire-5gplus.eu/wp-content/uploads/2022/08/i5-d4.4_liability-management-in-a-5g-environment_v1.0.pdf.

[12] INSPIRE-5GPlus, “D4.4 - Liability Management,” 2022.

[13] 5GENESIS Consortium, "Deliverable D4.3 The Athens Platform (Release C)," 2021. [Online]. Available: https://5genesis.eu/wp-content/uploads/2021/09/5GENESIS-D4.3_v1.0.pdf. [Accessed 23 December 2021].

[14] "TopoFuzzer," [Online]. Available: https://github.com/wsoussi/TopoFuzzer.

[15] INSPIRE-5Gplus Consortium, "D2.2 Initial Report on Security Use Cases, Enablers and Mechanisms for Liability-aware Trustable Smart 5G Security," May 2021. [Online]. Available: 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.

[16] INSPIRE-5Gplus Consortium, "D5.2 First 5G security testing infrastructure implementation and preliminary results," October 2021. [Online]. Available: https://www.inspire-5gplus.eu/wp-content/uploads/2021/11/i5-d5.2_first-5g-security-testing-infrastructure-implementation-and-preliminary-results_v1.0.pdf.

[17] 5GENESIS Consortium, "D4.2 The Athens Platform (Release B)," January 2020. [Online]. Available: https://5genesis.eu/wp-content/uploads/2020/02/5GENESIS_D4.2_v1.0.pdf. [Accessed 19 July 2021].

[18] INSPIRE-5Gplus Consortium, "D4.1 Trust mechanisms for 5G environments," September 2021. [Online]. Available: https://www.inspire-5gplus.eu/wp-content/uploads/2021/09/i5-d4.1_trust-mechanisms-for-5g-environments_v1.0.pdf.

Appendix A

Appendix A includes supplementary information which can be used as reference by interested reader to ease information sharing. Appendix A includes the following Sections:

- A.1: INSPIRE-5Gplus requirements and services
  - A.1.1: 5G Security Management: Functional and Non-Functional Requirements and Services
  - A.1.2: INSPIRE-5Gplus HLA's services
- A.2: Integration Tests in Demonstrators
  - A2.1.: Demo 1 Integration Tests
  - A2.2: Demo 2 Integration Tests
  - A2.3 Demo 3 Integration Tests
- A3: Port Forwarding – HA Proxy
- A4: Port Forwarding – Nginx
- A5: Port Forwarding – iptables
- A6: INSPIRE-5Gplus SMD Control Fabric from scratch

A.1 INSPIRE-5Gplus HLA requirements and services

Appendix A.1 provides the full list of requirements and services of the INSPIRE-5Gplus HLA for reference.

A.1.1 5G Security Management - Functional & Non-Functional Requirements

Security Req. ID	Security Requirement Description
SEC-REQ-01	The 5G network shall provide telemetry and other auditing information relevant to the security mechanisms of the system.
SEC-REQ-02	The 5G network shall only allow authenticated users to consume the services provided by the 5G system.
SEC-REQ-03	The 5G network shall warrant measurable level of availability of its services to the relevant stakeholders.
SEC-REQ-04	The 5G network shall ensure the necessary network capacity and network resources for the critical operations of the 5G services.
SEC-REQ-05	The 5G network shall enable a secure platform for vertical services to be deployed.
SEC-REQ-06	The 5G network shall enable the state management of its platform components.
SEC-REQ-07	The 5G network shall be able to revert to previous states with minimal service disruption of deployed application in case of malicious compromise.
SEC-REQ-08	The 5G network's security mechanisms should not impact the functional requirements of critical operations for vertical applications.
SEC-REQ-09	The security mechanisms of the 5G network shall be able to be deployed in any potential 5G hardware provider without any impact on their performance or functionality.
SEC-REQ-10	The security mechanisms of the 5G network shall be able to measure/evaluate trust level of its components and platforms and share this information with verticals in a safe and trustable way.
SEC-REQ-11	The security mechanisms used in a complex 5G ecosystem shall be able to identify, distribute and allocate responsibilities between 5G ecosystem stakeholders.
SEC-REQ-12	The 5G ecosystem shall be able to publish security KPI measuring the compliance of stakeholder with their Security Level Commitments.
SEC-REQ-13	Technologies used to distribute over 5G ecosystem (end to end) and evaluate post security incident root cause of failure are trustable.
SEC-REQ-14	The 5G system must provide security mechanisms to ensure that user (and endpoints) data are securely processed and stored wherever it is processed or stored. Both confidentiality and integrity guaranties shall be brought all along the full lifecycle of the data in transit, process

Security Req. ID	Security Requirement Description
	and storage.

Table 39: 5G security requirements.

Functional Security Req. ID	Security Requirement Description
FC-REQ-01	INSPIRE-5Gplus framework shall support the capability to collect up-to-date telemetry data.
FC-REQ-02	INSPIRE-5Gplus framework shall support the capability to specify the SSLA to detect security breaches and assess security functions.
FC-REQ-03	INSPIRE-5Gplus framework shall support the capability to ensure the SSLA during run-time.
FC-REQ-04	INSPIRE-5Gplus framework shall support the capability to allow multi-domain interaction.
FC-REQ-05	INSPIRE-5Gplus framework shall support the capability to ensure only validated/certified resources should be used.
FC-REQ-06	INSPIRE-5Gplus framework shall support the capability to perform anomaly prediction based on the required KPIs of the managed network slices
FC-REQ-07	INSPIRE-5Gplus framework shall support the capability to monitor different structured data (network, operating systems, applications, nsi).
FC-REQ-08	INSPIRE-5Gplus framework shall support the real-time assessment of SSLAs.
FC-REQ-09	INSPIRE-5Gplus framework shall support the generation of alerts that can be processed by the Security Orchestrator.
FC-REQ-10	INSPIRE-5Gplus framework shall support the translation of high-level policies to verifiable SSLAs and actionable remediations.
FC-REQ-11	INSPIRE-5Gplus framework shall support advanced techniques (e.g., ML) to classify and detect anomalies in encrypted traffic.
FC-REQ-12	INSPIRE-5Gplus framework shall support passive access to continuous up-to-date traffic in the network.
FC-REQ-13	INSPIRE-5Gplus framework shall support the capability to store telemetry data (or to steer their appropriate storage).
FC-REQ-14	INSPIRE-5Gplus framework shall support the capability to (pre-) process and filter the telemetry data, and to perform cross-domain data aggregation.

Functional Security Req. ID	Security Requirement Description
FC-REQ-15	INSPIRE-5Gplus framework shall support the capability to automatically deploy virtualized network functions software. (including virtualized security functions)
FC-REQ-16	INSPIRE-5Gplus framework shall support automatic configuration of virtualized network function parameters. (including virtualized security functions)
FC-REQ-17	INSPIRE-5Gplus framework shall support the capability of automatic verification of virtualized network functions normality after deployment.
FC-REQ-18	INSPIRE-5Gplus framework shall support the capability to specify security policies.
FC-REQ-19	INSPIRE-5Gplus framework shall support the capability to define the security policies in a technology independent policy definition language.
FC-REQ-20	INSPIRE-5Gplus framework shall support the capability to at least store, delete, activate, and deactivate security policies.
FC-REQ-21	INSPIRE-5Gplus framework shall support the capability to manage the defined security policies.
FC-REQ-22	INSPIRE-5Gplus framework shall support the capability to detect security policy conditions.
FC-REQ-23	INSPIRE-5Gplus framework shall have the capability to decide on security policy execution.
FC-REQ-24	INSPIRE-5Gplus framework shall support the capability to trigger the actions defined in the security policies.
FC-REQ-25	INSPIRE-5Gplus framework shall have the capability to detect conflicting security policies.
FC-REQ-26	INSPIRE-5Gplus framework shall support the interoperation with the NFV MANO APIs for management of NFV network services.
FC-REQ-27	INSPIRE-5Gplus framework shall support the collection of data on network slices status.
FC-REQ-28	INSPIRE-5Gplus framework shall support the capability to take action to mitigate performance degradation due to security issues.
FC-REQ-29	INSPIRE-5Gplus framework shall support automated management for compute, storage, and network resources, VNFs, slices and services for an automated MTD operation.
FC-REQ-30	INSPIRE-5Gplus framework shall support the detection of abnormal behaviours of the managed networks and services.
FC-REQ-31	INSPIRE-5Gplus framework shall support the capability to govern collected telemetry data.
FC-REQ-32	INSPIRE-5Gplus framework shall support the capability to common access to the collected up-to-date telemetry data.

Functional Security Req. ID	Security Requirement Description
FC-REQ-33	INSPIRE-5Gplus framework shall support stepwise introduction of ML-based management.
FC-REQ-34	INSPIRE-5Gplus framework shall support the capability to store historical data needed for the prediction and analytics.
FC-REQ-35	INSPIRE-5Gplus framework shall support the collection of data from ZSM managed entities to perform automated network and service security management based on AI.
FC-REQ-36	INSPIRE-5Gplus framework shall provide cyber security insights that would be valuable to a security analyst.
FC-REQ-37	INSPIRE-5Gplus framework shall support the automation of security assessment based on identified vulnerabilities.
FC-REQ-38	INSPIRE-5Gplus framework shall support the threat identification based on best practices, network configurations, user activities.
FC-REQ-39	INSPIRE-5Gplus framework shall support the use of network related information to elicit its components and other relevant information for security analysis.

Table 40: Functional Security Requirements

A.1.2 INSPIRE-5Gplus HLA's Services

Functional Module	Service
Security Data Collector (SDC)	Data Collection Service
Security Analytics Engine (SAE)	Anomaly Detection Service
	Root Cause Analysis Service
Decision Engine (DE)	Service Policy Proposal Service
	Security Asset Priority Service
Security Orchestrator (SO)	Security Policy Enforcement Service
Policy and SSLA Management (PSM)	HSPL Refinement Service
	MSPL/TOSCA Refinement Service
	Security Policy Storage Service
	Policy Conflict Detection Service
	SSLA Storage Service

Trust Management (TM)	Trust Reputation Manager
	Component Certification Service
	Slice Trustworthiness Service
	Ordered Proof of Transit Service
	Service Trust Manager Service
	Wrapper Service
	Hijacking Detection Service
	Deep Attestation Service
E2E Security Analytics Engine (E2E SAE)	Anomaly Detection Service
	Root Cause Analysis Service
E2E Decision Engine (E2E DE)	Security Policy Synchronization Service
	Security Policy Proposal service
	Security Asset Priority Service
E2E Security Orchestrator (E2E SO)	Security Policy Enforcement Service
E2E Policy and SSLA Management (E2E PSM)	Security SLA Refinement Service
	HSPL Refinement Service
	Policy Conflict Detection Service
	Security Policy Storage Service
E2E Trust Management (E2E TM)	Trust Reputation Manager Service
	Collaborative E2E Network Slice Management
Domain-Level & Cross-Domain Data Services	Data Access Service
Integration Fabric (IF)	Registration Service
	Discovery Service
	Invocation Service
	Communication service
Security Agent (SA)	Enforcement Point Service

	Network Monitoring and Telemetry Service
	Enforcement Point Service
Unified Security API	Network Service Actions List
Service Management Domain (SMD)	Network Slicing Management
	Network Digital Twin
E2E Service Management Domain (E2E SMD)	Network Slice Brokering

Table 41: INSPIRE-5Gplus HLA Services.

A.2 Integration Tests in Demonstrators

Demo	Enabler (Provider)	Enabler (Consumer)	Integration Test	Test Verdict (Passed/Failed)
Demo 1	WP3 - SSLA Manager (TSG)	WP3 - Secured Network Slice Manager for SSLAs (CTTC)	Demo1_SSLAMngr-SecSlice	Passed
Demo 1	WP3 - Secured Network Slice Manager for SSLAs (CTTC)	WP3 - Security orchestrator (UMU, TSG)	Demo1_SecSlice-SO	Passed
Demo 1	WP3 - SSLA Manager (TSG)	WP3 - SFSBroker (UOULU)	Demo1_SSLAMngr-SFSBroker	Passed
Demo 1	WP3 - SFSBroker (UOULU)	WP3 - Secured Network Slice Manager for SSLAs (CTTC)	Demo1_SFSBroker-SecSlice	Passed
Demo 1	WP3 - Decision Engine (TSG)	WP3 - Smart Traffic Analysis (TID)	Demo1_DE-STA	Passed
Demo 1	WP3 - DDoS Detection & Mitigation in Network Slicing (DDoS Detector) (AALTO)	WP3 - Security orchestrator (UMU, TSG)	Demo1_DDoSDetector-SO	Passed
Demo 1	WP3 - Lightweight and Space-efficient Authentication with Misbehavior Detection (CTTC)	WP3 - Security orchestrator (UMU, TSG)	Demo1_V2XMisDet-SO	Passed
Demo 1	WP4 - Trust Reputation Manager (UMU)	WP3 - Security orchestrator (UMU, TSG)	Demo1_SecOrch-TRM	Passed
Demo 1	WP3 - Security orchestrator (UMU, TSG)	WP3 - Security orchestrator (UMU, TSG)	Demo1_E2E_SO-SO	Passed
Demo 1	WP3 - Virtual Channel Protection (TSG)	WP3 - Security Monitoring Framework (MI)	Demo1_VCP-SMF	Passed
Demo 1	WP3 - Security Monitoring Framework (MI)	WP3 - Security orchestrator (UMU, TSG)	Demo1_SMF-SO	Passed
Demo 1	WP3 - Decision Engine (TSG)	WP3 - Security orchestrator (UMU, TSG)	Demo1_E2E_DE-E2E_SO	Passed
Demo 1	WP3 - Decision Engine (TSG)	WP3 - Decision Engine (TSG)	Demo1_DE-E2E_DE	Passed
Demo 1	WP3 - Security orchestrator (UMU, TSG)	WP3 - Decision Engine (TSG)	Demo1_DE-SO	Passed
Demo 1	WP3 - Security orchestrator (UMU, TSG)	WP3 - Lightweight and Space-efficient Authentication with Misbehavior Detection (CTTC)	Demo1_SO-V2XMisDet	Passed
Demo 1	WP3 - Smart Traffic Analysis (TID)	WP4 - Trust Reputation Manager (UMU)	Demo1_STA-TRM	Passed

Demo 1	WP4 - Trust Reputation Manager (UMU)	WP4 - E2E Trust Reputation Manager (UMU)	Demo1_TRM-E2E TRM	Passed
Demo 1	WP3 - Lightweight and Space-efficient Authentication with Misbehavior Detection (CTTC)	WP3 - Decision Engine (TSG)	Demo1_V2XMisDet-E2E_DE	Passed
Demo 1	WP3 - Virtual Channel Protection (TSG)	WP3 - Security orchestrator (UMU, TSG)	Demo1_VCP-SO	Passed
Demo 1	WP3 - Policy Framework (UMU)	WP3 - Security orchestrator (UMU, TSG)	Demo1_PolFram-SecOrch	Passed
Demo 1	WP3 - I2NSF IPSEC (TID)	WP3 - Security orchestrator (UMU, TSG)	Demo1_I2NSF-SO	Passed
Demo 1	WP4 - POT: Proof of Transit (TID)	WP4 - Trust Reputation Manager (UMU)	Demo1_PoT-TRM	Passed
Demo 1	WP3 - Smart Traffic Analysis (TID)	WP3 - Security orchestrator (UMU, TSG)	Demo1_STA-SO	Passed
Demo 1	WP4 - POT: Proof of Transit (TID)	WP3 - Security orchestrator (UMU, TSG)	Demo1_PoT-SO	Passed
Demo 1	WP3 - Security Monitoring Framework (MI)	WP3 - Security orchestrator (UMU, TSG)	Demo1_SMF-SO	Passed
Demo 2	WP4 - RCA : Root Cause Analysis (MI)	WP4 - Security by Orchestration K8S (ORA)	Demo2_RCA-SO	Passed
Demo 2	WP3_Security Monitoring Framework (MI)	WP4 - Security by Orchestration K8S (ORA)	Demo2_SMF-SOK	Passed
Demo 2	WP3_Security Monitoring Framework (MI)	WP4 - Security by Orchestration MEC (OLP)	Demo2_SMF-SOM	Passed
Demo 2	WP4 - Systemic VNF Wrapper (TAGES)	WP3_Security Monitoring Framework (MI)	Demo2 Wrapper-SMF	Passed
Demo 2	WP3_IOT Campus (MI)	WP4 - Security by Orchestration MEC (OLP)	Demo2_IOT-SOM	Passed
Demo 2	WP3_IOT Campus (MI)	WP4 - Security by Orchestration K8S (ORA)	Demo2_IOT-SOK	Passed
Demo 2	WP3_Security Monitoring Framework (MI)	WP4_Technician Command Center (ORA / OLP)	Demo2 SMF-TCC	Passed
Demo 2	WP4 - Systemic VNF Wrapper (TAGES)	WP4_Technician Command Center (ORA / OLP)	Demo2 Wrapper-TCC	Passed
Demo 2	WP4 - Deep Attestation Server (ORA)	WP4_Technician Command Center (ORA / OLP)	Demo2 DAS-TCC	Passed
Demo 2	WP4 - RCA : Root Cause Analysis (MI)	WP4_Technician Command Center (ORA / OLP)	Demo2 RCA-TCC	Passed
Demo 2	WP3_IOT Campus (MI)	WP4_Technician Command Center (ORA / OLP)	Demo2 IOT-TCC	Passed
Demo 2	WP4 - Discovery (CLS)	WP4_Technician Command Center (ORA / OLP)	Demo2_Disc_TCC	Passed

Demo 3	WP3 - Security Monitoring Framework (MI)	WP3 - OptSFC (ZHAW)	Demo3_SMF_OptSFC	**Passed**
Demo 3	WP3 - Security Analytics Framework (NCSRD)	WP3 - OptSFC (ZHAW)	Demo3_SAF-OptSFC	**Passed**
Demo 3	WP3 - OptSFC (ZHAW)	WP3 - Moving Target Defense Controller (ZHAW)	Demo3_OptSFC-MotDec	**Passed**
Demo 3	WP4 - Systemic VNF Wrapper (TAGES)	WP3 - OptSFC (ZHAW)	Demo3_Systemic_OptSFC	**Passed**

Table 42: Demo 1, 2 and 3 Integration Tests

A.2.1 Demo 1 Integration Tests

The deployment described in the Figure 5 is complex. Multiple testbeds are linked together and glued with an Integration Fabric. Before running the scenarios, the deployment was verified by running some integration tests. This section describes the tests put in place and the checks done to confirm the correct interaction between the components.

Demo Int. Test Name		**Demo1_SSLAMngr-SecSlice**	
Test Purpose		Validate the integration between the Secured Network Slice (SNS) for SSLAs and the SSLA Manager enablers defined in WP3.	
Description		The test will be used to validate to get the SSLA information and extract the required data to associate to a Network Slice in order to generate a MSPL with all the information to request the deployment of a secured network slice.	
Scenario		separated document (Figure of integration test showing the testbeds involved)	
Test flow		separated document (ideally, think to use Robot Framework, but it's not mandatory)	
Test Sequence	Step	Description	Result
	1	Setting Up Environment	Prepare environment information to be used during the test.
	2	Reception and acceptance to deploy a network Slice with a specific SSLA.	Request is accepted.
	3	The SecSlice requests to the SSLA Manager the information of a specific SSLA ID.	The SSLA information is returned.
	4	The SecSlice received the SSLA information with the correct format to work with it.	An MSPL data object is generated for the SO.
Test verdict		If no error appears during the different steps defined and the NSI deployment begins, the test will be considered as successful.	

Demo Int. Test Name		**Demo1_SecSlice-SO**	
Test Purpose		Validate the integration between the Secured Network Slice (SNS) for SSLAs and the Security Orchestrator (SO) enablers defined in WP3.	
Description		The test will be used to validate if an MSPL file generated by the SNS is accepted and applied by the SO. IN this case, the objective is to pass an MSPL with the policy defining which Network Slice Template (NST) to deploy.	
Scenario		separated document (Figure of integration test showing the testbeds involved)	
Test flow		separated document (ideally, think to use Robot Framework, but it's not mandatory)	
Test Sequence	Step	Description	Result
	1	Setting Up Environment	Prepare environment information to be used during the test.
	2	Reception and acceptance to deploy a network Slice with a specific SSLA.	Request (SLA ID + Slice Info) is accepted and generation of the corresponding MSPL.
	3	NST deployment procedure	A new Network Slice Instance (NSI) is created with the SLA ID and the slice information required (NST, etc).
	4	The SecSlice sends the policy defined in an MSPL to the SO.	Mapping of the NSI into an MSPL data object for the SO.
	5	PMSPL object accepted by the SO	The SO informs back the SNSM about the the correct appliance of the policy, meaning the NSI is well created.

Test verdict		If no error appears during the different steps defined and the NSI is correctly instantiated, the test will be considered as successful.	

Demo Int. Test Name		**Demo1_SSLAMngr-SFSBroker**	
Test Purpose		Validate the integration between SSLA Manager and SFSBroker enablers defined in WP3	
Description		The test will be used to validate to get security requirements from the tenants to create the SSLA.	
Scenario		separated document (Figure of integration test showing the testbeds involved)	
Test flow		separated document (ideally, think to use Robot Framework, but it's not mandatory)	
Test Sequence	Step	Description	Result
	1	Setting up environment	Prepare environment information to be used during the test.
	2	Verify resource request and run resource selection algorithm.	Request is accepted
	3	Create request for SSLA that matches security requirements.	SSLA is created and ACK is received
	4	Invoke the slice manager(s) to create the (federated) slice(s) as an off-chain invocation	The slice is instantiated in the resource provider
	5	Log the slice ID(s) in the distributed ledger	The slice IDs are registered corresponding to the transaction
Test verdict		If no error appears during different steps defined and the slice information and SSLA ID are sent to slice manager.	

Demo Int. Test Name		**Demo1_SFSBroker-SecSlice**	
Test Purpose		Validate the integration between the SFSBroker and Secured Network Slice Manager for SSLAs enablers defined in WP3.	
Description		The test will be used to retrieve the request for secure slice deployment.	
Scenario		separated document (Figure of integration test showing the testbeds involved)	
Test flow		separated document (ideally, think to use Robot Framework, but it's not mandatory)	
Test Sequence	Step	Description	Result
	1	Setting up environment	Prepare environment information to be used during the test.
	2	Reception of the acknowledgement of SSLA ID.	Request for SSLA ID is accepted and mapped with the resource allocation of the Network Slice
	3	Send SSLA ID and slice information as a request for secure slice deployment.	Reception of SSLA and Slice information for initiating the slice deployment.
	4	Receive success response from the SSLA manager	SSLA manager sends the success response of the SSLA establishment to the SFSBroker
	5	Log the SSLA response in the ledger	Blockchain registers the SSLA response corresponding to the request ID
Test verdict		If no error appears during the different steps defined and the NSI is correctly received by the slice manager to start the initiation process, the test will be considered as successful.	

Demo Int. Test Name		**Demo1_DE-STA**	
Test Purpose		Validate the integration between the Decision Engine (DE) and the Smart Traffic Analyzer (STA)	
Description		The test will be used to validate that the attack detection analytic generated by STA is collected by DE	
Scenario		In a separated document (Figure of integration test showing the testbeds involved)	
Test flow		In a separated document (ideally, think to use Robot Framework, but it's not mandatory)	
Test Sequence	Step	Description	Result
	1	Setting Up Environment	Prepare environment information to be used during the test.
	2	Start the Decision engine (DE)	The Decision Engine is active and collect insights from STA.
	3	STA generate security events and send to DE	STA VM/docker runs locally some traffic captures and generate security prediction in JSON format
	2	DE receives a request to store the event	Request is accepted.
	3	DE process and store the information	Event is stored in the DE
Test verdict		If no error appears and DE is able to store the information, the test will be considered as successful Detection information is received by DE	

Demo Int. Test Name		**Demo1_DDoSDetector-SO**	
Test Purpose		Validate the integration between the DDoS Detector and the Security Orchestrator (SO) enablers defined in WP3.	
Description		The test will be used to validate if an MSPL policy generated by the DDoS Detector is accepted and applied by the SO. In this case, the objective is to pass an MSPL with a drop policy to block traffic from a source when DDoS attack is detected.	
Scenario		separated document (Figure of integration test showing the testbeds involved)	
Test flow		separated document (ideally, think to use Robot Framework, but it's not mandatory)	
Test Sequence	Step	Description	Result
	1	Setting Up Environment	Prepare environment information to be used during the test.
	2	DDoS attack is detected by DDoS Detector.	Generation of an MSPL policy to block traffic from the malicious source.
	3	DDoS Detector sends the generated MSPL policy to the SO	MSPL containing information on the detected attacker and action to enforce (i.e., traffic dropping)
	4	SO receives the MSPL and enforces using the appropriate security appliance (e.g., SDN controller)	MSPL is converted to low-level configuration and enforced. The DDoS traffic is blocked.
Test verdict		If no error appears during the different steps defined and the DDoS traffic is blocked, the test will be considered as successful.	

Test Case Name		**Demo1_V2XMisDet-SO**	
Test Purpose		Fusion of V2X network traces, analysis and application of RL algorithm, detection of misbehaviour, and remediation (closing loop in CTTC SMD) upon decision that DoS attack has been detected	
Description		Streaming vehicular data reports are processed and sequentially analysed through their mobility patterns (position, velocity, acceleration) to instruct an RL algorithm for the detection of misbehaviour attacks. Filtering of the malicious traffic is performed as remediation action	
Scenario		separated document (Figure of integration test showing the testbeds involved)	
Test flow		separated document (ideally, think to use Robot Framework, but it's not mandatory)	
Test Sequence	Step	Description	Result
	1	Setting up the environment	Prepare environment with the Security Orchestrator and the policy framework
	2	Start the misbehaviour detection framework	Misbehaviour detection engine is active and ready to receive V2X time-series data from vehicles
	3	Start the generation of V2X time-series data	VM/docker runs locally (or from the data plane) V2X data sources and generates time-series
	4	DoS attack is detected by the misbehaviour detector engine	Generation of a reactive MSPL policy to deny traffic from the malicious IP source
	5	Upon detection of misbehaviour events, detection framework informs SO to apply a given security policy based on the generated MSPL policy	MSPL containing information on the detected attacker and action to enforce (i.e., deny traffic)

	6	SO receives the MSPL and enforces the appropriate security measure (filtering of the malicious traffic)	MSPL is converted to low-level configuration and enforced. Misbehaving data source is isolated/dropped/blocked
	7	Malicious IP traffic source is denied	Malicious traffic filtering takes place at the application layer
Test verdict		If no error appears during the different steps and DoS traffic is denied, the test will be considered as successful.	

Demo Name		**Demo1_SecOrch-TRM**	
Test Purpose		Validate the integration between Security Orchestrator (UMU) and Trust Reputation Manager (UMU)	
Description		The test will be used to validate the communication between both enablers	
Scenario		In a separate document (Figure of integration test showing the testbeds involved)	
Test flow		In a separate document (eventually using the Robot Framework, but this is not mandatory)	
Test Sequence	Step	Description	Result
	1	Setting Up Environment	Prepare environment information to be used during the test
	2	Request a policy orchestration	Policy orchestration process starts
	3	Security Orchestrator requests trust metrics to TRM	Security orchestrator retrieves trust metrics

	4	Security Orchestrator uses trust metrics during the policy orchestration process	Policies are orchestrated
Test verdict		Security Orchestrator have trust metrics to select assets among the possibilities	

Demo Name		**Demo1_E2E_SecOrch-Security_Orchestrator**	
Test Purpose		Validate the integration between E2E_Security Orchestrator (UMU) and Security Orchestrator (UMU) and Security Orchestrator (TSG)	
Description		The test will be used to validate the communication between both enablers	
Scenario		In a separate document (Figure of integration test showing the testbeds involved)	
Test flow		In a separate document (eventually using the Robot Framework, but this is not mandatory)	
Test Sequence	Step	Description	Result
	1	Setting Up Environment	Prepare environment information to be used during the test
	2	Request a policy orchestration at SMD level	Policy orchestration process starts
	3	E2E Security Orchestrator requests policy orchestration to SMD Security Orchestrator	Security orchestrator receives the MSPL-OP from E2E SO
	4	Security Orchestrator generates and starts the orchestration plan	Policy orchestration plan starts
Test verdict		Each SMD SO has received the corresponding MSPL-OP and starts the orchestration plan	Each SO has its corresponding MSPL-OP

Demo Int. Test Name	**Demo1_VCP-SMF**	
Test Purpose	Validate the ability of the Security Monitoring Framework (MI) to detect anomalous change to VCP enabler (DTLS Proxy) configuration	

Description		The test will be used to validate the detection of changes in DTLS configuration of the DTLS proxy(ies) in one of the SMD (MI/EURESCOM), that violate - are not compliant with - the SSLA.	
Scenario		The SMD SOs receive a security policy for deploying the VCP enabler instances in both SMDs, deploy them with the proper configuration. By analysing the network traffic the MI probe detects some unexpected DTLS configuration change (e.g. DTLS/ciphersuite downgrade), and raises a security alert to the Decision Engine or else. More details on the figure on the right, SMD (MI) part.	
Test flow		separated document (ideally, think to use Robot Framework, but it's not mandatory)	
Test Sequence	Step	Description	Result
	1	Each SMD SO receives a channel protection policy for its SMD, then each SMD SO deploys its own VCP enabler instance (DTLS proxy) on K8s; and finally the two DTLS proxies are able to establish a secure channel with each other from one SMD to the other.	Passed DTLS handshakes captured
	2	One of the DTLS proxy (e.g. the one in MI SMD) configuration is changed manually by an attacker (with access to the container shell or K8s API) or privileged admin (insider) to a non-compliant DTLS configuration (not compliant with the SSLA).	Passed Change in DTLS version (3.0 to 2.1 downgrade) or cipher suite
	3	By analysing the network traffic (or testing the TLS handshake itself), the MI probe (SMF Security Data Collector) detects some unexpected DTLS configuration change in the DTLS handshakes (e.g. DTLS/ciphersuite downgrade), which triggers a security alert to the SMF Security Analytics Engine.	Passed Security alert received by DE

Test verdict		The SMF Security Analytics Engine receives a security alert 'SSLA violation' from the SMF Security Data Collector (MI).	

Demo Int. Test Name		**Demo1_SMF-SO**
Test Purpose		Enforce monitoring policies in MI SMD for SSLA 2
Description		MI SMD's SO deploys and configures Security Monitoring Framework's MMT Probe on MI SMD according to SSLA 2
Scenario		separated document (Figure of integration test showing the testbeds involved)
Test flow		separated document (ideally, think to use Robot Framework, but it's not mandatory)
Test Sequence	Step	Description
	1	Security orchestrator receives the MSPL-OP
	2	Security Orchestrator calls the Policy Framework to translate MSPL-OP to MMTProbe configurations.
	3	SO deploys and configures the MMT Probe
Test verdict		MMT Probe deployed

Demo Name	**Demo1_E2E_DE-E2E_SO**	
Test Purpose	Validate the integration between E2E Decision Engine (TSG) and E2E Security Orchestrator (UMU)	

Description		The test will be used to validate the communication between both enablers	
Scenario		In a separate document (Figure of integration test showing the testbeds involved)	
Test flow		In a separate document (eventually using the Robot Framework, but this is not mandatory)	
Test Sequence	Step	Description	Result
	1	Setting Up Environment with the E2E DE and the E2E SO and a fake component to verify the SO output	Prepare environment information to be used during the test
	2	Trigger an alter to the E2E DE	The Decision Engine process start and create a MLPS-OP
	3	The E2E DE send the generated MSPL-OP to the E2E SO	E2E Security Orchestrator receive the MPL-OP from the E2E DE
	4	The E2E Security Orchestrator starts the orchestration plan	The E2E SO emits a custom MSPL-OP to a underlying SMD SO
Test verdict		The MSPL-OP for the SMD SO is captured	A MSPL-OP is sent to the a fake SMD SO

Demo Name		**Demo1_DE-E2E_DE**	
Test Purpose		Validate the integration between a SMD DE (TSG) and E2E Decision Engine (TSG)	
Description		The test will be used to validate the communication between both enablers	
Scenario		In a separate document (Figure of integration test showing the testbeds involved)	
Test flow		In a separate document (eventually using the Robot Framework, but this is not mandatory)	

Test Sequence	Step	Description	Result
	1	Setting Up Environment	Prepare environment information to be used during the test
	2	Trigger an alert to the SMD DE	The SMD DE generated an updated a MSPL-OP
	3	The SMD DE send the notification to the E2E DE	The E2E Decision Engine receive a notification with the locally updated MSPL-OP
	4	The E2E Decision Engine is able to detect a potential double notification	The E2E DE follows with the E2E SO
Test verdict		The E2E is able to emit an MSPL-OP from a SMD notification	

Demo Name		**Demo1_DE-SO**	
Test Purpose		Validate the integration between the SMD Decision Engine (TSG) and SMD Security Orchestrator (UMU), in particular the DE feature that consists to sends a new MSPL-OP to the SO as remediation for a security incident (attack, SSLA violation, etc.). In this context, the DE consumes SO's HTTP API to update the orchestration policy.	
Description		The test will be used to validate the communication between both enablers	
Scenario		In a separate document (Figure of integration test showing the testbeds involved)	
Test flow		In a separate document (eventually using the Robot Framework, but this is not mandatory)	
Test Sequence	Step	Description	Result
	1	Setting Up Environment with the SMD DE and the SMD SO and a fake component to verify the SO output	Passed Prepare environment information to be used during the test

	2	Trigger an alter to the SMD DE	Passed The Decision Engine process start and create a MSPL-OP
	3	The SMD DE send the generated MSPL-OP to the SO	Passed The SMD Security Orchestrator receives the MPL-OP from the SMD DE
	4	The SMD Security Orchestrator starts the orchestration plan	Passed The SMD SO orchestrates the underlying VIM
Test verdict		The output from the SMD SO is captured	A MSPL-OP is sent to the SMD SO

Demo Int. Test Name		**Demo1_DDoSDetector-SO**	
Test Purpose		Validate the integration between the DDoS Detector and the Security Orchestrator (SO) enablers defined in WP3.	
Description		The test will be used to validate if an MSPL policy generated by the DDoS Detector is accepted and applied by the SO. In this case, the objective is to pass an MSPL with a drop policy to block traffic from a source when DDoS attack is detected.	
Scenario		separated document (Figure of integration test showing the testbeds involved)	
Test flow		separated document (ideally, think to use Robot Framework, but it's not mandatory)	
Test Sequence	Step	Description	Result

	1	Setting Up Environment	Prepare environment information to be used during the test.
	2	DDoS attack is detected by DDoS Detector.	Generation of an MSPL policy to block traffic from the malicious source.
	3	DDoS Detector sends the generated MSPL policy to the SO	MSPL containing information on the detected attacker and action to enforce (i.e., traffic dropping)
	4	SO receives the MSPL and enforces using the appropriate security appliance (e.g., SDN controller)	MSPL is converted to low-level configuration and enforced. The DDoS traffic is blocked.
Test verdict		If no error appears during the different steps defined and the DDoS traffic is blocked, the test will be considered as successful.	

Test Case Name		**Demo1_SO-V2XMisDet**	
Test Purpose		Proactive configuration of the V2X misbehaviour detector from the SO to detect DDoS traffic. This test is part of the DoS detection enforcement workflow.	
Description		The SO interacts with the enabler for the applied policy which is represented by the inter-arrival time threshold of basic safety messages in V2X	
Scenario		separated document (Figure of integration test showing the testbeds involved)	
Test flow		separated document (ideally, think to use Robot Framework, but it's not mandatory)	
Test Sequence	Step	Description	Result

	1	Setting up the environment	Prepare environment with the SO and the policy framework
	2	Start the misbehaviour detection framework	Misbehaviour detection engine is active and ready to receive V2X time-series data from vehicles
	3	SO receives V2X DDoS misbehaviour detector configuration	A detection threshold value is specified for the detection of DDoS traffic
	4	V2X DDoS misbehaviour detector configuration by the SO	SO configures the V2X detector with a detection threshold value, related to inter-arrival time of basic safety messages
Test verdict		If no error appears during the different steps, the test will be considered as successful.	

Demo Int. Test Name		**Demo1_STA-TRM**	
Test Purpose		Validate the integration between Smart Traffic Analysis STA (TID) and Trust Reputation Manager (UMU)	
Description		The test will be used to validate the communication between both enablers	
Scenario		In a separate document (Figure of integration test showing the testbeds involved)	
Test flow		In a separate document (eventually using the Robot Framework, but this is not mandatory)	
Test Sequence	Step	Description	Result
	1	Setting Up Environment	Prepare environment information to be used during the test

	2	Smart Traffic Analysis STA generate metrics about the (enforced) traffic	Metrics generated
	3	Smart Traffic Analysis STA sends those metrics to TRM	TRM receives metrics
	4	TRM computes the corresponding trust score based on received metrics and previously stored information and then stores the metrics and the computed trust score inside DLT	Metrics are stored inside DLT
Test verdict		If valid metrics are stored in DLT, the test will be considered as successful.	

Demo Int. Test Name		**Demo1_TRM-E2E TRM**	
Test Purpose		Validate the integration between Trust Reputation Manager (UMU) and E2E Trust Reputation Manager (UMU)	
Description		The test will be used to validate the communication between both enablers	
Scenario		In a separate document (Figure of integration test showing the testbeds involved)	
Test flow		In a separate document (eventually using the Robot Framework, but this is not mandatory)	
Test Sequence	Step	Description	Result

	1	Setting Up Environment	Prepare environment information to be used during the test
	2	TRM computes the trust score value fora given	Value computed
	3	TRM sends the trust score value of the enabler, the enabler id and the domain where the enabler is located to the E2E TRM	E2E TRM receives the information (json format)
	4	E2E TRM computes the corresponding DOMAIN trust score based on received values and previous stored information about the given SMD and then stores the received information and the new computed domain trust score inside DLT	Values are stored inside DLT
Test verdict		If valid metrics are stored in DLT, the test will be considered as successful.	

Test Case Name		**Demo1_V2XMisDet-E2E_DE**	
Test Purpose		The DE of the V2X misbehaviour detector informs the decision about DoS detection in CTTC SMD to the E2E DE	
Description		Decision about DoS detection is communicated to the E2E DE	
Scenario		separated document (Figure of integration test showing the testbeds involved)	
Test flow		separated document (ideally, think to use Robot Framework, but it's not mandatory)	

Test Sequence	Step	Description	Result
	1	Setting up the environment	Prepare environment with the E2E DE
	2	Alert triggered in the SMD DE	The SMD DE generates an updated MSPL-OP
	3	SMD DE sends the notification to the E2E DE	The E2E DE receives a notification with the locally updated MSPL-OP
	4	The E2E DE is able to receive communication from the SMD DE	The E2E DE follows with the E2E SO
Test verdict		The E2E DE is able to emit an MSPL-OP from the SMD DE notification	

Demo Int. Test Name	**Demo1_VCP-SO**
Test Purpose	Validate the integration between SMD Security Orchestrator (TSG) and Virtual Channel Protection enabler (TSG)
Description	The test will be used to validate the process by which the SMD SO deploys the VCP enabler and verifies/manages the deployment progress.
Scenario	The SO receives a security policy that requires to deploy the VCP enabler, deploys the VCP enabler with the proper configuration and checks the progress until the policy enforcement is active. More details on the figure on the right (SMD part).
Test flow	separated document (ideally, think to use Robot Framework, but it's not mandatory)

Test Sequence	Step	Description
	1	SMD SO receives a channel protection policy and deploys the VCP enabler as CNF (container) in Kubernetes (VIM).
	2	SMD SO checks with K8s API that the VCP enabler (DTLS proxy) is successfully deployed and running
	3	
	4	
Test verdict		A DTLS client from the other SMD with a valid certificate is able to establish a secure DTLS channel with the deployed VCP enabler using expected DTLS settings.

Demo Name		**Demo1_Security_Orchestrator-I2NSF-Controller**	
Test Purpose		Validate the integration between E2E_Security Orchestrator (UMU) and I2NSF Controller (TID)	
Description		The test will be used to validate the communication between both enablers	
Scenario		In a separate document (Figure of integration test showing the testbeds involved)	
Test flow		In a separate document (eventually using the Robot Framework, but this is not mandatory)	
Test	St	Description	Result

Sequence	ep		
	1	Setting Up Environment	Prepare environment information to be used during the test
	2	Request the deployment of I2NSF Agent and Controller	SO receives final asset configuration for I2NSF Agent and Controller
	3	SO deploy required infrastructure to host IN2SNF Agent and Controller	VMs deployed to host services
	4	Security Orchestrator configures the I2NSF Controller	I2NSF Controller will communicate with specified I2NSF Agents and deploy IPsec tunnel between them
Test verdict		IPSec tunnel is deployed	traffic is encrypted between endpoints

Demo Int. Test Name		**Demo1_POT-TRM**	
Test Purpose		Validate the integration between Proof of Transit (TID) and Trust Reputation Manager (UMU)	
Description		The test will be used to validate the communication between both enablers	
Scenario		In a separate document (Figure of integration test showing the testbeds involved)	
Test flow		In a separate document (eventually using the Robot Framework, but this is not mandatory)	
Test Sequence	Step	Description	Result
	1	Setting Up Environment	Prepare environment information to be used during the test
	2	PoT generate metrics about the (enforced) traffic	Metrics generated
	3	PoT sends those metrics to TRM	TRM receives metrics

	4	TRM computes the corresponding trust score based on received metrics and previous stored information and then stores the metrics and the computed trust score inside DLT	Metrics are stored inside DLT
Test verdict		If valid metrics are stored in DLT, the test will be considered as successful.	

Demo Int. Test Name		**Demo1_STA-SO**	
Test Purpose		Validate the integration between Security orchestrator (UMU) and STA (TID)	
Description		The test will be used to validate the activation of the STA monitoring.	
Scenario		In a separate document (Figure of integration test showing the testbeds involved)	
Test flow		In a separate document (eventually using the Robot Framework, but this is not mandatory)	
Test Sequence	Step	Description	Result
	1	Setting Up Environment	Prepare environment with the Security Orchestrator, policy framework and I2NSF IPSEC enabler
	2	Security Orchestrator receives MSPL-OP	MSPL-OP
	3	Security Orchestrator prepares an orchestration plan according to the orchestration policies	Orchestration plan
	4	Security Orchestrator requests MSPL-OP translation to Policy Framework for STA enabler	Final asset configurations
Test verdict		Security Orchestrator requests conf enforcement to STA enabler	STA deployed

Demo Int. Test Name		Demo1_PoT_Controller-SO	
Test Purpose		Validate the integration between Security orchestrator (UMU) and PoT_Controller (TID)	
Description		The test will be used to validate the activation of the PoT validation.	

Scenario		In a separate document (Figure of integration test showing the testbeds involved)	
Test flow		In a separate document (eventually using the Robot Framework, but this is not mandatory)	
Test Sequence	Step	Description	Result
	1	Setting Up Environment	Prepare environment with the Security Orchestrator, policy framework and IPoT enabler
	2	Security Orchestrator receives MSPL-OP	MSPL-OP
	3	Security Orchestrator prepares an orchestration plan according to the orchestration policies	Orchestration plan
	4	Security Orchestrator requests MSPL-OP translation to Policy Framework for PoT enabler	Final asset configurations
Test verdict		Security Orchestrator requests conf enforcement to PoT enabler	PoT validations is executed

Demo Int. Test Name		**Demo1_SMF-SO**
Test Purpose		Enforce monitoring policies in MI SMD for SSLA 2
Description		MI SMD's SO deploys and configures Security Monitoring Framework's MMT Probe on MI SMD according to SSLA 2
Scenario		separated document (Figure of integration test showing the testbeds involved)
Test flow		separated document (ideally, think to use Robot Framework, but it's not mandatory)
Test Sequence	Step	Description

	1	Security orchestrator receives the MSPL-OP
	2	Security Orchestrator calls the Policy Framework to translate MSPL-OP to MMTProbe configurations.
	3	SO deploys and configures the MMT Probe
Test verdict		MMT Probe deployed

A.2.2 Demo 2 Integration Tests

Demo Int. Test Name		**Demo2_RCA-SO**	
Test Purpose		Validate the integration between the Root Cause Analysis (MI) enabler and the Security Orchestrator (SO) enabler(s) defined in WP4.	
Description		The test covers the anomaly detection and root-cause determination of MI's RCA enabler in monitoring an industrial campus and the integration with the Security Orchestrator under normal and critical state (e.g., fire)	
Scenario		separated document (Figure of integration test showing the testbeds involved)	
Test flow		separated document (ideally, think to use Robot Framework, but it's not mandatory)	
Test Sequence	Step	Description	Expected Result
	1	MI's RCA monitors the industrial campus under normal state	Monitoring data are properly collected and analysed (network traffic, hardware-related indicators). Monitoring interface displays "GREEN" state.
	2	RCA detects an anomaly at network level and suggests that it might be due to a physical incident (e.g., fire)	SO is informed and the monitoring interface displays "RED" state.
	3	SO requests to enable the video streaming service to verify the suggested root-cause	Video streaming service is enabled

	4		
Test verdict		If the monitoring interface displays correctly the "GREEN"/ "RED" state, the incident is precisely detected and the video streaming is enabled once requested, the test will be considered successful.	

Demo Int. Test Name		**Demo2 DAS-TCC**	
Test Purpose		Check that the attestation server acts correctly	
Description		The test covers the following properties (i) the integration of the attestation server and the attestation agents and (ii) the attestation server can answer a request via the provided API.	
Scenario		separated document (Figure of integration test showing the testbeds involved)	
Test flow		separated document (ideally, think to use Robot Framework, but it's not mandatory)	
Test Sequence	Step	Description	Result
	1	The attestation server is launched.	The API is operational and the attestation server can receive commands.
	2	The attestation agents are launched.	The attestation agents listen on a given port.

	3	Register a target / agent via the API	The target/ agent is added to the attestation server database.
	4	Ask for an attestation via the API	The attestation server returns a quote.
Test verdict		There is no error then the test is correct	

Demo Int. Test Name		**Demo2_SMF-SOK**	
Test Purpose		Validate the integration between the Security Monitoring Framework (MI) enabler and the Security by Orchestration K8S (ORA) enabler(s) defined in WP4.	
Description		The test covers the integration and communication of Security Monitoring Framework (MI) enabler and the Security Orchestrator K8S under normal and critical state (e.g., fire)	
Scenario		separated document (Figure of integration test showing the testbeds involved)	
Test flow		separated document (ideally, think to use Robot Framework, but it's not mandatory)	
Test Sequence	Step	Description	Expected Result

	1	SMF is deployed to monitor the industrial campus. It sends "statistics" about the IoT campus to SO in real-time, using MQTT	MQTT messages are properly sent/received with no loss.
	2	SMF detects an anomaly at network level due to a physical incident (e.g., fire)	The dashboard shows the anomaly reflected in figures/ alerts
	3		
	4		
Test verdict		If there are no errors, then the test is successful.	

Demo Int. Test Name	**Demo2_SMF-SOM**	
Test Purpose	Validate the integration between the Security Monitoring Framework (MI) enabler and the Security by Orchestration MEC (OLP) enabler(s) defined in WP4.	
Description	The test covers the integration and communication of Security Monitoring Framework (MI) enabler and the Security Orchestrator MEC under normal and critical state (e.g., fire)	
Scenario	separated document (Figure of integration test showing the testbeds involved)	

Test flow		separated document (ideally, think to use Robot Framework, but it's not mandatory)	
Test Sequence	Step	Description	Expected Result
	1	SMF is deployed to monitor the industrial campus. It sends "statistics" about the IoT campus to SO in real-time, using MQTT	MQTT messages are properly sent/received with no loss.
	2	SMF detects an anomaly at network level due to a physical incident (e.g., fire)	The dashboard shows the anomaly reflected in figures/ alerts
Test verdict		If there are no errors, then the test is successful.	

Demo Int. Test Name	**Demo2_IOT-SOM**	
Test Purpose	Validate the integration between the IoT Campus (MI) and the Security by Orchestration MEC (OLP) enabler(s) defined in WP4.	
Description	The test covers the integration and communication of IoT Campus (MI) enabler and the Security Orchestrator MEC under normal and critical state (e.g., fire)	
Scenario	separated document (Figure of integration test showing the testbeds involved)	

Test flow		separated document (ideally, think to use Robot Framework, but it's not mandatory)	
Test Sequence	Step	Description	Expected Result
	1	IoT Campus is deployed and the sensors send MQTT messages to SO in real-time	MQTT messages are properly sent/received with no loss.
	2	A physical incident (e.g., fire) occurs and is reflected on the SO's dashboards.	The dashboard shows the anomaly reflected in figures/ alerts
Test verdict		If there are no errors, then the test is successful.	

Demo Int. Test Name	**Demo2_IOT-SOK**	
Test Purpose	Validate the integration between the IoT Campus (MI) and the Security by Orchestration K8S (ORA) enabler(s) defined in WP4.	
Description	The test covers the integration and communication of IoT Campus (MI) enabler and the Security Orchestrator K8S under normal and critical state (e.g., fire)	
Scenario	separated document (Figure of integration test showing the testbeds involved)	

Test flow		separated document (ideally, think to use Robot Framework, but it's not mandatory)	
Test Sequence	Step	Description	Expected Result
	1	IoT Campus is deployed and the sensors send MQTT messages to SO in real-time	MQTT messages are properly sent/received with no loss.
	2	A physical incident (e.g., fire) occurs and is reflected on the SO's dashboards.	The dashboard shows the anomaly reflected in figures/ alerts
	3		
	4		
Test verdict		If there are no errors, then the test is successful.	

Demo Int. Test Name	**xDemo2 SBOMEC-TCC**	
Test Purpose	Validate the MEC application on-boarding and placement optimization using Security by Orchestration for MEC Enabler	
Description	The test will be used to validate interaction between Technician Command Center and MEC Orchestrator in order to onboard MEC Application (taking into account respective SLA elements including isolation requirement)	

Scenario		separated document (Figure of integration test showing the testbeds involved)	
Test flow		separated document (ideally, think to use Robot Framework, but it's not mandatory)	
Test Sequence	Step	Description	Result
	1	Setting Up Environment	Prepare environment information (including MEC infrastructure and hosted applications) to be used during the test
	2	New MEC application is added	MEC application image is added to the repository, MEC application data is added to MEC Orchestrator
	3	Placement procedure is executed	The optimal placement for each application instance is defined.
	4	Applications deployment is executed	MEC application instances are started in defined locations
	5	The report of running application instances is invoked	The report of running applications is available to verify the isolation constraints
Test verdict		If no error appears during the different steps defined and MEC applications are deployed with defined constraints, the test will be considered as successful.	

Demo Int. Test Name	**Demo2_SMF-TCC**	
Test Purpose	Validate the integration between the Security Monitoring Framework (MI) and the Technician Command Center (ORA / OLP) enabler(s) defined in WP4.	

Description		The test covers the integration and communication of the Security Monitoring Framework (MI) enabler and the Technician Command Center (ORA / OLP) under normal and critical state (e.g., fire)	
Scenario		separated document (Figure of integration test showing the testbeds involved)	
Test flow		separated document (ideally, think to use Robot Framework, but it's not mandatory)	
Test Sequence	Step	Description	Result
	1	SMF monitors the IoT Campus under normal conditions	The GUI stays "GREEN" at TCC
	2	SMF raises an alert indicating an anomaly at network level	The GUI becomes "RED" at TCC. A button appears to enable CRITICAL mode and the video streaming service.
Test verdict		If there are no errors, then the test is successful.	

Demo Int. Test Name	**Demo2 Wrapper-TCC**	Test prerequisites and Results
Test Purpose	This test covers the reception, authentication, content parsing and presentation of signed heartbeats attesting the correct functioning state of MMT protected software.	All actions relevant to the sub-test referred as Wrapper-SMF (i.e., MMT-probe is duly protected by Systemic-SGX) are all pre-requisite conditions for this sub-test. These steps 1 and 2 are recalled here with the yellow

			background.
Description		The heartbeats integrate the relevant information for identifying the unique deployed instance, the confirmation that it truly executes, at the correct position (i.e., a AES key provisioned machine) and is integrated. The heartbeats are signed by systemic routine. The heartbeats contain the associate pub key used to produce the attestation verification.	
Scenario		Generation of heartbeats on MMT-Probe software	
Test flow		Protection of MMT-Probe, launch of protected version, generation of heartbeats and transmission to TCC, exploitation at TCC as described above (auth verification, parsing, presentation).	
Test Sequence	Step	The protected MMT-Probe generates its signed heartbeats, transmits them safely to the TCC specific Systemic app which produces the auth verification, parsing and presentation of the content.	Presentation of the functional status of the deployed code. (security properties are validated, security properties are breached).
Protect MMT-Probe	1	Protect (without failure code)	No failure message during protection process
Install MMT-Probe	2	Load and launch the protected variant of MMT-Probe	No abnormal behaviour at program start
Generation of signed heartbeats by Systemic routine	3	Verification of the periodic heartbeat generation at MMT-probe level (memory introspection?)	Secure communication link is established
Verification of communication link to TCC	4	Check the secure communication channel between the Terminal and Systemic appended security routine	Received order at Systemic

Heartbeats reception, auth verification, parsing, presentation	5	Verification of the good functioning of the heartbeat exploitation app by TCC.	Receipt of the heartbeats.
			The content of the heartbeat payload is a binary blob, encrypted using AES-128-GCM.
			Once decrypted using the key pre-shared at protection time, the blob contains a JSON as follows:
			{
			"integrity": true,
			"pid": 123,
			"systemic_id": "id-515123534",
			"created_at": "1994-08-25T00:18:58Z",
			"request_id": "id-794312464",
			}
			The request headers will contain two custom headers:
			* x-content-digest: The request body digest, hashing the request body with SHA-256 and then encoding the result in base64
			* x-signature: The signature of the content digest, using RSA-3072 and encode the result in base64.
			
			At the server side at least 3 validations should be performed:
			1. Check for the existence of the custom headers.
			2. Digest the request body and check if it is equal to the value of the "x-content-digest" header.
			3. Verify the signature of the "x-signature" header
			4. Decrypt the binary blob using AES-128-GCM pre-shared key
			
			All those validations must succeed to accept the request. In the case heartbeats are not timely received while the service is expected to be up and running, launch an alert.

Test verdict		If heartbeats are received and validated, test is	

		successful.	

Demo Int. Test Name	**Demo2 Wrapper-SMF**	Other information
Test Purpose	Check the good execution of SYSTEMIC-SGX wrapping of MMT-Probe. As a result of the test, MMT-Probe shall be protected and be running on a SGX-enabled platform, taking advantage of the SGX security for the security of the appended Systemic routine.	Important: This test is followed by the test Wrapper-TCC which covers the good functioning of the heartbeat generation, the completeness of their content for three different security properties (execution, at the right location, in integrated form). In other words, this test only covers the good execution of the wrapping process by Systemic-SGX while the second test checks the good functioning of the security properties verification offered by the heartbeats monitoring.
Description	Protect SMF with Systemic-SGX. Install the protected variant and launch it. Verify MMT-Probe launches successfully.	

Scenario		No testbed integration of Systemic SECaaS is considered. The wrapping of Montimage's SMF (ie, MMT-Probe) is done on TAGES own server, by use of Montimage credentials to access the service. Both Systemic SECaaS and the Systemic routine shall be viewed as inside Montimage Security Domain. The test verifies that MMT-Probe is protected by Systemic-SGX and its good functioning in its protected form.	
Test flow		Protect and install the Systemic-SGX protected MMT-Probe into MI's SMT on one SGX enabled platform. Verification of its correct functioning state on this platform.	Tampering is detected and reported by Systemic (system print)
Test Sequence	Step		Result
protect	1	Protect on TAGES own infrastructure of the MMT-Probe (with SGX security flavor)	Check the good operation of the SECaaS wrapper (no failure during protection process)
Install	2	Install the protected variant and launch the code on MI's SMT on one SGX enabled platform	For testing purposes decryption key is self-contained in protected binary.
Verify launch sequence	3	Verify that the code passes its self-authentication and decryption steps and launches	Successful launch verifies that self-authentication and decryption stages pass and that original decrypted code executes well.
Test verdict		If SMF process launches correctly and MMT-Probe process is brought up, test is successful	

Demo Int. Test Name		**Demo2_RCA-TCC**	
Test Purpose		Validate the integration between the Root Cause Analysis (MI) enabler and the Technician Command Center (ORA / OLP) enabler(s) defined in WP4.	
Description		The test covers the integration and communication of Root Cause Analysis (MI) enabler and the Technician Command Center (ORA / OLP) under normal and critical state (e.g., fire)	
Scenario		separated document (Figure of integration test showing the testbeds involved)	
Test flow		separated document (ideally, think to use Robot Framework, but it's not mandatory)	
Test Sequence	Step	Description	Result
	1	RCA monitors the IoT Campus under normal conditions	The GUI stays "GREEN" at TCC
	2	RCA raises an alert indicating that a fire might be the root cause of the anomaly	The GUI becomes "RED" at TCC. A button appears to enable CRITICAL mode and the video streaming service.
	3		
	4		
Test verdict		If there are no errors, then the test is successful.	

Demo Int. Test Name	**Demo2_IOT-TCC**	

Test Purpose		Validate the integration between the IOT Campus (MI) enabler and the Technician Command Center (ORA / OLP) enabler(s) defined in WP4.	
Description		The test covers the integration and communication of IOT Campus (MI) enabler and the Technician Command Center (ORA / OLP) under normal and critical state (e.g., fire)	
Scenario		separated document (Figure of integration test showing the testbeds involved)	
Test flow		separated document (ideally, think to use Robot Framework, but it's not mandatory)	
Test Sequence	Step	Description	Result
	1	IoT Campus works under normal conditions	The GUI stays "GREEN" at TCC
	2	A fire (real or virtual) occurs at IoT Campus	The GUI becomes "RED" at TCC. A button appears to enable CRITICAL mode and the video streaming service.
Test verdict		If there are no errors, then the test is successful.	

Demo Int. Test Name	**Demo2_Disc_TCC**	
Test Purpose	Validate the integration between DiscØvery and the Technician Command Center.	
Description	The test validates the message exchange between the DiscØvery enabler and the Technician Command Center.	
Scenario	separated document (Figure of integration test showing the testbeds involved)	

Test flow		separated document (ideally, think to use Robot Framework, but it's not mandatory)	
Test Sequence	Step	Description	Result
	1	DiscØvery detects a security event and generates an alarm.	DiscØvery generates an alarm
	2	The alarm is sent to the Technician Command Center in a JSON format via a REST API.	Alarm is sent
	3	The alarm is received by the Technician Command Center.	Alarm is received
	4	Receipt of the alarm	Receipt of the alarm is sent to DiscØvery
Test verdict		Step 4 is validated	

A.2.3 Demo 3 Integration Tests

Test Case Name		**Demo3_MotDec-SliceM**	
Test Purpose		Validate the interaction between MTD Controller MOTDEC (ZHAW) and the Slice Manager (NCSRD).	
Description		MOTDEC sends create/modify/delete requests to the Slice MManager that should be properly accepted and instantiated.	
Scenario		separated document (Figure of integration test showing the testbeds involved)	
Test flow		separated document (ideally, think to use Robot Framework, but it's not mandatory)	
Test Sequence	Step	Description	**Result**

	1	Setting Up Environment	Define the testing environment.
	2	MOTDEC sends a request to the Slice Manager to create a slice.	The Slice Manager returns the UUID of the slice.
	3	MOTDEC sends a request to the Slice Manager to get running slices.	The Slice Manager returns a list with all the running slices, including the UUID of each one.
	4	MOTDEC checks the status of the slice.	Slice Manager will return information regarding the new slice, including the status (Init/Placement/Provisioning/Activation/Running), what are its IPv4 and IPv6 addresses, in which VIM is it deployed and bandwidth used
	5	MOTDEC sends a request to the Slice Manager to get available VIMs.	The Slice Manager returns the UUID and full description of the VIMs: environment (Openstack, VMware, etc.), max / current usage CPU capacity, max. RAM capacity and max. disk capacity.
	6	MOTDEC sends a request to reinitiate the network slice or sub-VNF.	The Slice Manager reinitiates the network slice or sub-VNF
	7	MOTDEC sends a request to migrate the network slice or sub-VNF.	The Slice Manager migrated the network slice or sub--VNF
Test verdict		If there are no errors, then the test is successful.	

Test Case Name		**Demo3_SMF_OptSFC**	
Test Purpose		Validate the interaction between OptSFC (ZHAW) and MMT (MI).	
Description		OptSFC receives attack alerts from MI	
Scenario		separated document (Figure of integration test showing the testbeds involved)	
Test flow		separated document (ideally, think to use Robot Framework, but it's not mandatory)	

Test Sequence	Step	Description	**Result**
	1	Setting Up Environment	Define the testing environment.
	2	OptSFC indicates to MMT which networks to monitor	MMT activate probes of the networks required by OptSFC
	3	MMT sends anomaly and attack detection to OptSFC	MMT sends the targeted IP addr, the "attacker" 's IP addr, and the type of attack: anomaly detection or DDoS.
	4	MMT sends analysis of technical KPIs to OptSFC	MMT-QoS/QoE library of the MMT probe/s collects KPIs and QoS metrics for each slice and service: latency, jitter, packet loss rate, retransmission rate.
Test verdict		If there are no errors, then the test is successful.	

Test Case Name		**Demo3_SAF-OptSFC**	
Test Purpose		Validate the integration between the Security Analytics Framework (NCSRD) and the OptSFC enabler (ZHAW).	
Description		The test will evaluate if the connection for data transfer between the two enablers is active.	
Scenario		separated document (Figure of integration test showing the testbeds involved)	
Test flow		separated document (ideally, think to use Robot Framework, but it's not mandatory)	
Test Sequence	Step	Description	Result
	1	Setup the Virtual Environment	Prepare the virtual environment and the parameters of the test.
	2	Start the Security Analytics Framework to generate network traffic.	Security Analytics Framework is active and generates network traffic.

	3	Start OptSFC and configure it to read the output of the Security Analytics Framework.	OptSFC is active and reads data from the Security Analytics Framework.
Test verdict		If the network connection is live and no errors appears during the different steps defined, the test will be considered as successful.	

Test Case Name		**Demo3_Systemic-OptSFC**	Technical prerequisite and Results
Test Purpose		Validate the integration between Systemic (TAGES) and OptSFC (ZHAW) enablers.	Show the interaction of Systemic and OptSFC enablers.

Description		The test will evaluate if Systemic integrity alerts are delivered to OptSFC and if the connection for data transfer between the two enablers is active.	Demo 3 takes place inside NCRSD infra with VM management. This has turned TAGES and ZHAW to **remove the SGX requirement** as it overcomplexifies the demo with SGX-through VM issues). This demo is aimed at demonstrating the benefit of the inter enabler interaction, not the efficience of Systemic (ability to protect of VNF). Therefore, Systemic will use a **VideoProcessingVNF** made for the purpose. The protected binary will packaged using OpenStack VM and delivered to NCSRD for installation on their OSM managed VM infrastructure. The protected VNF performs periodict integrity checks and upon failure, sends an alert to OptSFC endpoint. Systemic will exchange with OptSFC-API through a **POST/Sytemic Alert JSON-formatted message** containing the following fields:
			{
			"type": "tampering",
			"description": "Integrity check fault detected",
			"systemic_id": "id-515123534",
			"created_at": "2018-11-20T15:58:44.767594",
			"pid": 123,
			"tid": 321,
			"signature": "AABBCCDDEEFF100122334455667788 99"
			}
			
			OptSFC gets the **geolocation of the tampering** using the IP address of

			the API connection. OptSFC shall ideally verify the **authentication of the JSON file** using a pre-provisionned public key before processing the information.
Scenario		The test will evaluate if the connection for data transfer between the two enablers is active.	OptSFC receives a integrity alert on a VNF and integrates it in its MTD strategy
Test flow		separated document (ideally, think to use Robot Framework, but it's not mandatory)	Systemic protects its HelloWorld VNF and packs the binary into an OpenStack VM. NCSRD installs the VM in its infrastructure. The protected VNF transmits a tampering alert to OptSFC.
Test Sequence	Step	Description	**Result**
	1	Setup the Virtual Environment	Prepare the virtual environment and the parameters of the test. protect the helloworld VNF and pack it as an OpenStack VM delivered to NCSRD
	2	Start the VNF wrapped with Systemic	Systemic tampering detection system activated on the VNF.
	3	Start OptSFC and configure it to read the output of Systemic	OptSFC is active and has an API POST request to receive alerts from Systemic.
	4	Tamper the memory of protected VNF	A GET request to the /attack/1 endpoint of wrapped VNF tampers the code of the VNF in memory. This results in a modification of the content in the page receveid with a GET request on the root path: fomr "Hello **NORMAL** World" to "Hello **HACKED** World".
	5	Systemic detects a tampering attack and alerts OptSFC	Systemic successfully sends an alert to OptSFC via a POST request.
Test verdict		If the network connection is live and no errors appears during the different steps defined, the test will be considered as successful.	

Test Case Name		**Demo3_OptSFC-MOTDEC**	
Test Purpose		Validate the integration between the OptSFC and MOTDEC (ZHAW).	
Description		The test will evaluate if MOTDEC receives the OptSFC decisions	
Scenario		separated document (Figure of integration test showing the testbeds involved)	
Test flow		separated document (ideally, think to use Robot Framework, but it's not mandatory)	
Test Sequence	Step	Description	Result
	1	MOTDEC shares the real-time data on the network state to Opt-SFC	Opt-SFC models a Markov Decision Process
	2	OptSFC's RL agent propose an MTD action	MOTDEC receives the REST API instruction from OptSFC to enforce the MTD action
	3	If MOTDEC accepts the MTD action (after verification with a global orchestrator), the action is enforced	OptSFC receives back whether the action was accepted or was rejected
	4	Go to the next iteration (Step 1)	After *t* seconds, the process is iterated
Test verdict		If the network connection is live and no errors appears during the different steps defined, the test will be considered as successful.	

A.3 Port Forwarding - HAProxy

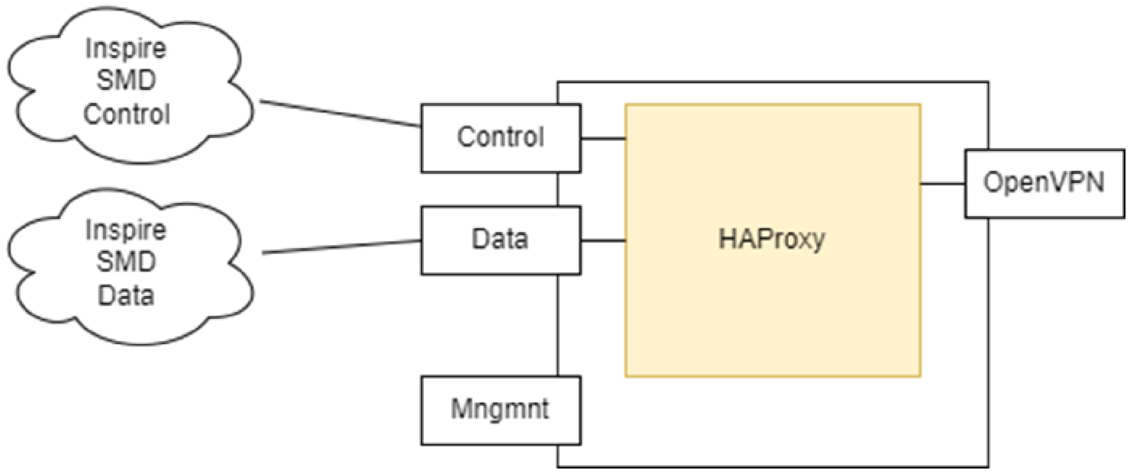


Figure 50: HAProxy

The VM has 2 internal networks that are related to the control and data plane, as well as one interface facing external networks via the openVPN connection. Configure the NICs to use your internal VLANs and networks.

Configuration
Under /etc/haproxy/haproxy.cfg.

There are two main usages:

- **Frontend** exposes the ports that will be translated into an internal IP via **backend**
- **Frontend** translate internal IP:port into VPNdest:port destination to communicate with certain enabler via **backend**.

Backend will be server declared while frontend will bind an IP:port to a backend.

Some examples:

```
frontend cttcsideFrontPortA

                bind 10.0.37.11:8080

        default_backend umusideBackPortA

backend umusideBackPortA

        server id1 10.208.7.46:80
```

From outside to inside, translating bind address into server address:

```
frontend umusideFrontPortA

        bind 10.208.88.30:2222

        default_backend cttcsideBackPortA

backend cttcsideBackPortA

        server cttck8s 10.0.37.5:22
```

From inside to outside, translating internal address into vpn cttc address.

The following command verify if configuration is valid:

```
        $ haproxy -c -f /etc/haproxy/haproxy.cfg
```

and for launch it:

```
        $ haproxy -f /etc/haproxy/haproxy.cfg
```

To test that everything is good, open a terminal and make a tcpdump over tun interface and telnet connection to some of translated IPs:Port (eg. in our example we made telnet 10.208.88.30:2222and it was redirected to 10.0.37.5:22)

A.4 Port Forwarding - Nginx

NGINX as TCP/UDP proxy - UMU PoC

This appendix provides a configuration example to use Nginx as TCP/UDP proxy.

Prerequisites

Current configuration was tested in Ubuntu 20.04 server.

Setup

```
sudo apt-get install nginx
```

Configuration

Disable default server configuration:

```
sudo rm /etc/nginx/sites-enabled/defaul
```

Add stream directive in /etc/nginx/nginx.conf **between "include" and "event" directives**.

```
stream {
        upstream mybackend_ssh {
              server 10.0.37.5:22;
       }
        upstream mybackend_dns {
              server 10.0.37.4:53;
       }
       server {
              listen 10.204.4.69:2222;
              proxy_pass mybackend_ssh;
       }
        server {
              listen 10.204.4.69:53 udp;
              proxy_pass mybackend_dns;
       }
}
```

In previous example, nginx server IP is 10.204.4.69 and it has also an interface attached to 10.0.37.0 network. The flow can be read as following:

- Traffic comming from 10.204.4.69:2222 will be sent to 10.0.37.5:22
- Traffic comming from 10.204.4.69:53 will be sent to 10.0.37.4:53

Apply the changes

```
sudo service nginx reload
```

A.5 Port Forwarding - iptables

# iptables as TCP/UDP proxy (port forwarding) - EURES PoC

This appendix provides a configuration example to use iptables TCP/UDP proxy.

### Scenario

node1 (MI 10.0.37.19) and node2 (EURES 10.0.37.27 on tun1 interface) are interconnected via CTTC VPN via DLTS Proxy. Node1 must be able to connect to ports 5684, 8080 and 8443 of services on node2 which have an IP address from MetalLB IP range (for example 10.204.4.5). This can be done by remap of relevant ports with iptables NAT and masquerade traffic from VPN node2 to the MetalLB network (10.204.4.0/24).

### Prerequisites

Configuration was tested in Ubuntu 20.04 server and connectivity was proven to work fine between MI and EURES nodes. It is necessary to enable forwarding in the kernel. The steps for doing that are:

- edit /etc/sysctl.conf and add -> net.ipv4.ip_forward=1

- sudo sysctl -p

- sudo sysctl -system

## Setup

```

Add following iptables rules:

masquerade traffic from VPN node2 to the MetalLB network => NOT NEEDED

```
sudo iptables -t nat -A POSTROUTING -s 10.204.4.0/24 -o tun1 -m policy --dir out --pol none -j MASQUERADE

sudo iptables -t nat -A POSTROUTING -s 192.168.198.0/24 -o tun1 -m policy --dir out --pol none -j MASQUERADE
```

simpler alternative:

```
sudo iptables -t nat -I POSTROUTING 1 -o tun1 -j MASQUERADE
```

Method for debugging your rules is to add an identical rule to the one you're interested in, but set the action to being:
```

-j LOG --log-prefix "rule description"

```
sudo iptables -t nat -I POSTROUTING 2 -o tun1 -j LOG --log-prefix "tun1 MASQUERADE"



List MASQUERADE rules:

sudo iptables -t nat -v -L POSTROUTING -n --line-number
```

Delete MASQUERADE rules:

```
sudo iptables -t nat -D POSTROUTING 5
```

remap of relevant ports with iptables NAT (DNAT rules). One rule for tcp and one for udp

MetalLB destinations (10.204.4.3 - 10.204.4.12) don't work

Calico Pod addresses work (192.168.198.0/24) work and can be found with kubectl get pods --all-namespaces -o wide

```
sudo iptables -t nat -A OUTPUT -d 10.0.37.27/32 -p tcp -m tcp --sport 1:65535 --dport 5684 -j DNAT --to-destination 10.204.4.6:5684

sudo iptables -t nat -A OUTPUT -d 10.0.37.27/32 -p udp -m udp --sport 1:65535 --dport 5684 -j DNAT --to-destination 10.204.4.6:5684

sudo iptables -t nat -A OUTPUT -d 10.0.37.27/32 -p tcp -m tcp --sport 1:65535 --dport 8080 -j DNAT --to-destination 10.204.4.6:8080

sudo iptables -t nat -A OUTPUT -d 10.0.37.27/32 -p udp -m udp --sport 1:65535 --dport 8080 -j DNAT --to-destination 10.204.4.6:8080

sudo iptables -t nat -A OUTPUT -d 10.0.37.27/32 -p tcp -m tcp --sport 1:65535 --dport 8443 -j DNAT --to-destination 10.204.4.6:8443

sudo iptables -t nat -A OUTPUT -d 10.0.37.27/32 -p udp -m udp --sport 1:65535 --dport 8443 -j DNAT --to-destination 10.204.4.6:8443



sudo iptables -t nat -I OUTPUT 1 -d 10.0.37.27/32 -p tcp -m tcp --sport 1:65535 --dport 389 -j DNAT --to-destination 10.204.4.5:389

sudo iptables -t nat -A OUTPUT -d 10.0.37.27/32 -p udp -m udp --sport 1:65535 --dport 389 -j DNAT --to-destination 10.204.4.5:389



sudo iptables -t nat -A OUTPUT -d 10.0.37.27/32 -p tcp -m tcp --sport 1:65535 --dport 389 -j DNAT --to-destination 192.168.198.13:389
```

ldap:

```
sudo iptables -t nat -A PREROUTING -d 10.0.37.27/32 -p tcp -m tcp --sport 1:65535 --dport 389 -j DNAT --to-destination 192.168.198.13:389 => works (DNAT on server, no MASQUERADE required)

sudo iptables -t nat -A PREROUTING -d 10.0.37.27/32 -p tcp -m tcp --sport 1:65535 --dport 389 -j DNAT --to-destination 10.204.4.5:389 => doesn't work (no reply received)

sudo iptables -t nat -A PREROUTING -d 10.0.37.27/32 -p tcp -m tcp --sport 1:65535 --dport 389 -j LOG --log-prefix "MetalLB DNAT port 389"
```

dtls:

```
sudo iptables -t nat -A PREROUTING -d 10.0.37.27/32 -p udp -m udp --sport 1:65535 --dport 5684 -j DNAT --to-destination 192.168.198.25:5684 => works (DNAT on server, no MASQUERADE required)



sudo iptables -t nat -A PREROUTING -d 10.0.37.27/32 -p tcp -m tcp --sport 1:65535 --dport 5684 -j DNAT --to-destination 192.168.198.25:5684

sudo iptables -t nat -A PREROUTING -d 10.0.37.27/32 -p udp -m udp --sport 1:65535 --dport 5684 -j DNAT --to-destination 192.168.198.25:5684

sudo iptables -t nat -A PREROUTING -d 10.0.37.27/32 -p tcp -m tcp --sport 1:65535 --dport 8080 -j DNAT --to-destination 192.168.198.25:8080

sudo iptables -t nat -A PREROUTING -d 10.0.37.27/32 -p udp -m udp --sport 1:65535 --dport 8080 -j DNAT --to-destination 192.168.198.25:8080

sudo iptables -t nat -A PREROUTING -d 10.0.37.27/32 -p tcp -m tcp --sport 1:65535 --dport 8443 -j DNAT --to-destination 192.168.198.25:8443

sudo iptables -t nat -A PREROUTING -d 10.0.37.27/32 -p udp -m udp --sport 1:65535 --dport 8443 -j DNAT --to-destination 192.168.198.25:8443



sudo iptables -t nat -A PREROUTING -d 10.0.37.27/32 -p tcp -m tcp --sport 1:65535 --dport 5684 -j DNAT --to-destination 10.204.4.6:5684

sudo iptables -t nat -A PREROUTING -d 10.0.37.27/32 -p udp -m udp --sport 1:65535 --dport 5684 -j DNAT --to-destination 10.204.4.6:5684 => doesn't work (no reply received)

sudo iptables -t nat -A PREROUTING -d 10.0.37.27/32 -p tcp -m tcp --sport 1:65535 --dport 8080 -j DNAT --to-destination 10.204.4.6:8080

sudo iptables -t nat -A PREROUTING -d 10.0.37.27/32 -p udp -m udp --sport 1:65535 --dport 8080 -j DNAT --to-destination 10.204.4.6:8080

sudo iptables -t nat -A PREROUTING -d 10.0.37.27/32 -p tcp -m tcp --sport 1:65535 --dport 8443 -j DNAT --to-destination 10.204.4.6:8443

sudo iptables -t nat -A PREROUTING -d 10.0.37.27/32 -p udp -m udp --sport 1:65535 --dport 8443 -j DNAT --to-destination 10.204.4.6:8443
```

List PREROUTING rules:

```
sudo iptables -t nat -v -L PREROUTING -n --line-number
```

Delete PREROUTING rules:

```
sudo iptables -t nat -D PREROUTING 4
```

Rules can be listed with:

List MASQUERADE rules with:

```
sudo iptables -t nat -v -L POSTROUTING -n --line-number
```

```
Chain POSTROUTING (policy ACCEPT 7811 packets, 475K bytes)
num   pkts bytes target     prot opt in     out     source               destination
1      364K    21M cali-POSTROUTING  all  --  *      *       0.0.0.0/0            0.0.0.0/0            /* cali:O3lYWMrLQYEMJtB5 */
2     280K   17M KUBE-POSTROUTING  all  --  *      *       0.0.0.0/0            0.0.0.0/0            /* kubernetes postrouting rules */
3        0     0 MASQUERADE  all  --  *      !docker0  172.17.0.0/16        0.0.0.0/0
4        0     0 MASQUERADE  all  --  *      !br-ffa946f6f5dc  192.168.49.0/24      0.0.0.0/0
5        0     0 MASQUERADE  all  --  *      tun0    10.204.4.0/24        0.0.0.0/0            policy match dir out pol none
```

MASQUERADE rules can be deleted with:

```
sudo iptables -t nat -D POSTROUTING 5
```

List DNAT rules with:

```
sudo iptables -t nat -v -L OUTPUT -n --line-number
```

```
Chain OUTPUT (policy ACCEPT 7573 packets, 455K bytes)
num   pkts bytes target     prot opt in     out     source               destination
1     273K   16M cali-OUTPUT  all  --  *      *       0.0.0.0/0            0.0.0.0/0            /* cali:tVnHkvAo15HuiPy0 */
```

```
2    273K   16M KUBE-SERVICES  all  --  *      *       0.0.0.0/0            0.0.0.0/0            /* kubernetes service portals */

3    69316 4159K DOCKER     all  --  *      *       0.0.0.0/0           !127.0.0.0/8          ADDRTYPE match dst-type LOCAL

4        1    60 DNAT       tcp  --  *      *       0.0.0.0/0            10.0.37.27           tcp spts:1:65535 dpt:15021 to:10.204.4.3:15021

5        2   120 DNAT       tcp  --  *      *       0.0.0.0/0            10.0.37.27           tcp spts:1:65535 dpt:1337 to:10.204.4.3:1337

6        0     0 DNAT       tcp  --  *      *       0.0.0.0/0            10.0.37.27           tcp spts:1:65535 dpt:1337 to:10.204.4.3:1337

7        2    68 DNAT       udp  --  *      *       0.0.0.0/0            10.0.37.27           udp spts:1:65535 dpt:1337 to:10.204.4.3:1337

8        0     0 DNAT       tcp  --  *      *       0.0.0.0/0            10.0.37.27           tcp spts:1:65535 dpt:1337 to:10.204.4.2:1337

9        0     0 DNAT       udp  --  *      *       0.0.0.0/0            10.0.37.27           udp spts:1:65535 dpt:1337 to:10.204.4.2:1337
```

DNAT rules can be deleted with:

```
sudo iptables -t nat -D OUTPUT 4
```

Rules can be saved with:

```
sudo service iptables-persistent save
```

### Recommendation

It is recommended to configure an additional systemd service to ensure the rules can be easily modified and ensure that they are reloaded at boot time. To do this:

1. create /etc/systemd/system/iptables-load-rules.service

```
[Unit]

After=kubelet.service


[Service]

ExecStart=/usr/local/bin/iptables-load-rules.sh
```

```
[Install]

WantedBy=default.target
```

2. create /usr/local/bin/iptables-load-rules.sh

```
sudo chmod 744 /usr/local/bin/iptables-load-rules.sh
```

#!/bin/bash

```
sudo iptables -t nat -A POSTROUTING -s 10.204.4.0/24 -o tun1 -m policy --dir out --pol none -j MASQUERADE

sudo iptables -t nat -A OUTPUT -d 10.0.37.27/32 -p tcp -m tcp --sport 1:65535 --dport 5684 -j DNAT --to-destination 10.204.4.5:5684

sudo iptables -t nat -A OUTPUT -d 10.0.37.27/32 -p udp -m udp --sport 1:65535 --dport 5684 -j DNAT --to-destination 10.204.4.5:5684

sudo iptables -t nat -A OUTPUT -d 10.0.37.27/32 -p tcp -m tcp --sport 1:65535 --dport 8080 -j DNAT --to-destination 10.204.4.5:8080

sudo iptables -t nat -A OUTPUT -d 10.0.37.27/32 -p udp -m udp --sport 1:65535 --dport 8080 -j DNAT --to-destination 10.204.4.5:8080

sudo iptables -t nat -A OUTPUT -d 10.0.37.27/32 -p tcp -m tcp --sport 1:65535 --dport 8443 -j DNAT --to-destination 10.204.4.5:8443

sudo iptables -t nat -A OUTPUT -d 10.0.37.27/32 -p udp -m udp --sport 1:65535 --dport 8443 -j DNAT --to-destination 10.204.4.5:8443

sudo netfilter-persistent save
```

```
3. sudo systemctl daemon-reload


4. sudo systemctl enable iptables-load-rules.service


5. reboot
```

### Check and Tests

To check the rules are set correctly, use following commands:

```
sudo iptables -t nat -v -L POSTROUTING -n --line-number  => should show the masquerade rule
sudo iptables -t nat -v -L OUTPUT -n --line-number       => should show the DNAT rules
```

Testing can be done with "nc"

first open listening port on node2:

TCP:

```
    nc -l -p 5684 (8080 8443)
    check if listening with: ss -nat|grep 5684 (8080 8443)
```

then start client on node1:

```
    nc 10.0.37.27 5684 (8080 8443)
    root@zork:~# nc 10.0.37.27 5684
    hello
    ^C
```

UDP:

```
    nc -u -l -p 5684 (8080 8443)
    check if listening with: ss -u -nat|grep 5684 (8080 8443)
```

then start client on node1:

```
    nc -u 10.0.37.27 5684 (8080 8443)
    root@zork:~# nc -u 10.0.37.27 5684
    hello
    ^C
```

A.6 INSPIRE5GPlus SMD Control Fabric from scratch

This Appendix provides detailed information about how to instantiate the UMU Integration fabric from scratch to ease its adoption by the rest of the partners.

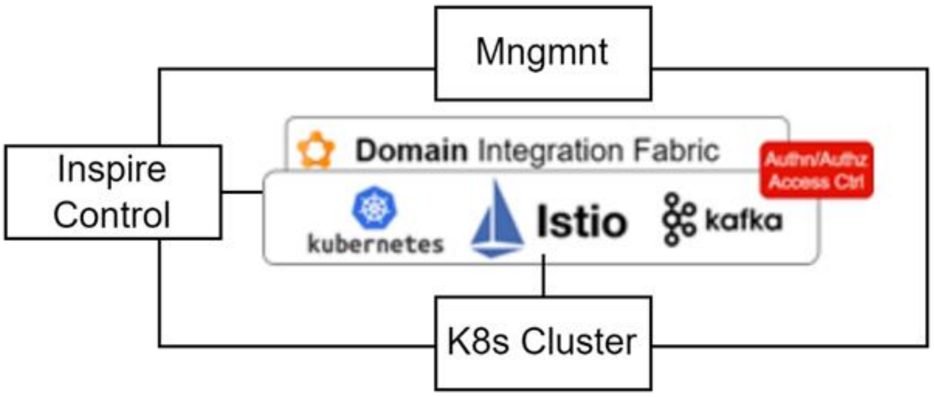


Figure 51: Interfaces

Network configuration:

- Management interface: To manage the VM.
- Inspire Control: Connected to the control plane.
- K8s Cluster: To communicate with other cluster nodes (if any). As master, k8s will be listening in that interface for join requests.

Update the dns:

Update the DNS address with the DNS you will use to resolve your queries.

```
sudo nano /etc/systemd/resolver.conf

DNS=X.X.X.X
```

K8S

Kubeadm install

- Install docker
  - $ curl -fsSL https://get.docker.com -o get-docker.sh
  - $ sudo sh get-docker.sh
  - sudo usermod -aG docker $USER
  - relogin for applying the new group
- Install kubernetes by using the kubernetes script (k8s website) or execute:

```
sudo apt-get update && sudo apt-get install -y apt-transport-https curl

curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add -

cat <<EOF | sudo tee /etc/apt/sources.list.d/kubernetes.list

deb https://apt.kubernetes.io/ kubernetes-xenial main

EOF
```

```
sudo apt-get update

sudo apt-get install -y kubelet kubeadm kubectl

sudo apt-mark hold kubelet kubeadm kubectl
```

- Disable swap before executing init or join
  - sudo swapoff -a
- To do it permanent (required) create rc.local and put the command before exit 0
  - sudo nano /etc/rc.local

```
#!/bin/sh -e

#

# rc.local

#

# This script is executed at the end of each multiuser runlevel.

# Make sure that the script will "exit 0" on success or any other

# value on error.

#

# In order to enable or disable this script just change the execution

# bits.

#

# By default this script does nothing.

swapoff -a

exit 0
```

- sudo chmod 744 /etc/rc.local

Kubeadm init

- sudo kubeadm init --apiserver-advertise-address=172.16.0.1 --pod-network-cidr=192.168.0.0/16
  - apiserver-advertise-address: Nodes will use it for connecting to the master
  - pod-network-cidr: This range is for Calico network plugin

```
Your Kubernetes control-plane has initialized successfully!

 To start using your cluster, you need to run the following as a regular user:

   mkdir -p $HOME/.kube

  sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config

  sudo chown $(id -u):$(id -g) $HOME/.kube/config

 Alternatively, if you are the root user, you can run:

   export KUBECONFIG=/etc/kubernetes/admin.conf
```

```
You should now deploy a pod network to the cluster.

Run "kubectl apply -f [podnetwork].yaml" with one of the options listed at:

 https://kubernetes.io/docs/concepts/cluster-administration/addons/

 Then you can join any number of worker nodes by running the following on each as root:

kubeadm join 172.16.0.1:6443 --token sxisyc.8nzqereu49var2af \

            --discovery-token-ca-cert-hash
sha256:83ca86d88147dd20050ca9b7094801b45d706263c133ed64d2108ccb59837981
```

- mkdir -p $HOME/.kube
- sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
- sudo chown $(id -u):$(id -g) $HOME/.kube/config

<u>Kubeadm test</u>

- kubectl get nodes
    - It will appear as not ready since we need to install the network plugin

<u>Adding the network plugin</u>

- curl https://docs.projectcalico.org/manifests/calico.yaml -O
- kubectl apply -f calico.yaml
- kubectl get nodes
- Now you should see master as ready

<u>Important:</u> If you do not add more nodes you will need to allow deployments on the control plane:

```
kubectl taint nodes --all node-role.kubernetes.io/master-
```

<u>Adding the GUI</u>

Create admin user:

```
kubectl create serviceaccount dashboard-admin-sa

kubectl create clusterrolebinding dashboard-admin-sa --clusterrole=cluster-admin --
serviceaccount=default:dashboard-admin-sa

kubectl get secrets

kubectl describe secret dashboard-admin-sa-token-xxxxx
```

- Copy the token

Exec the server from the master:

```
kubectl proxy
```

Create a tunnel between the master and your desktop machine:

```
ssh -L 9001 :127.0.0.1:8001 -N -f -l user IP
```

Open the browser:

```
http://localhost:9001                    /api/v1/namespaces/kubernetes-dashboard/services/https:kubernetes-
dashboard:/proxy/
```

- Paste the token in the login page

Adding ISTIO

Install istio

- curl -L https://istio.io/downloadIstio | sh -
- export PATH="$PATH:/home/ants/istio-1.10.2/bin"
- istioctl x precheck
- istioctl install --set profile=demo

Create namespace with istio

- kubectl create namespace inspire-smd
- kubectl label namespace inspire-smd istio-injection=enabled

Istio dashboard

- Install kiali (+prometheus+grafana+jeager):
- cd istio folder
- kubectl apply -f samples/addons
- kubectl rollout status deployment/kiali -n istio-system
- Exec kiali dashboard
  - istioctl dashboard kiali
- Redirect port for kiali
  - ssh -L 9002 :127.0.0.1:20001 -N -f -l user IP
  - ssh -L 9001:127.0.0.1:8001 -L 9002:127.0.0.1:20001 -N -f -l user IP (if you want to maintain also for kubernetes admin dashboard)
- Enter from the browser in the desktop machine where the redirect port was performed
  - http://localhost:9002/kiali

More info at:https://istio.io/latest/docs/setup/getting-started/

MetalLB

```
# see what changes would be made, returns nonzero returncode if different

kubectl get configmap kube-proxy -n kube-system -o yaml | \

sed -e "s/strictARP: false/strictARP: true/" | \

kubectl diff -f - -n kube-system
```

```
# actually apply the changes, returns nonzero return code on errors only

kubectl get configmap kube-proxy -n kube-system -o yaml | \

sed -e "s/strictARP: false/strictARP: true/" | \

kubectl apply -f - -n kube-system
```

Install metalLB

- kubectl apply -f https://raw.githubusercontent.com/metallb/metallb/v0.9.5/manifests/namespace.yaml
- kubectl apply -f https://raw.githubusercontent.com/metallb/metallb/v0.9.5/manifests/metallb.yaml
- # On first install only
  - kubectl create secret generic -n metallb-system memberlist --from-literal=secretkey="$(openssl rand -base64 128)"

Put the virtual IP in config.yaml

```
apiVersion: v1

kind: ConfigMap

metadata:

 namespace: metallb-system

 name: config

data:

 config: |

  address-pools:

  - name: default

   protocol: layer2

   addresses:

   - 10.204.4.3-10.204.4.3
```

- kubectl apply -f config.yaml

Verify now have the istio-ingress gw with external-ip

- kubectl get svc istio-ingressgateway -n istio-system

More info at: https://metallb.org/

DNS Entry

Request a wildcard entry for your domain in your DNS provider

```
*         IN   A   10.204.4.3
```

UMU lab example for k8s.gaialab zone:

```
nslookup. kafka.k8s.gaialab
```

Examples and tests

INSPIRE-5GPlus Integration fabric conf files and examples – UMU PoC repository includes examples to use k8s + istio + metalLB + kafka as ZSM integration fabric instantiation inside the INSPIRE-5GPlus H2020 European Project. It also provides examples for deploying control plane services as part of the control plane of the Security Management Domain (SMD) inside the k8s that will use integration fabric capabilities.

Repository URL:

```
https://ants-gitlab.inf.um.es/inspire-5gplus-release/integration-fabric-umu-basics
```

If UMU CA is not recognized in your device, you can use “git -c http.sslVerify=false clone …”

Important!!: The yamls provided for each service must be updated with your own DNS values (e.g., kafka.k8s.gaialab => kafka.YOURDOMAIN) according to your DNS records**.**

Deployment:

```
cd integration-fabric-umu-basics
 ./deploy.sh
```

It will deploy a kafka service, two instances of nginx (with load balancing example properties 90/10), an external service configuration and an internal service to perform different kind of tests. In general, a service definition is composed of three yamls:

- **Service.yaml**: The definition of the service itself.
- **Routing.yaml**: The definition of the routing. How the service will be accessed through the istio mesh.
- **Deployment.yaml**: The definition of the deployment. How many containers, versions etc.

Running the tests:

Test folder contains different tests to verify internal/external messaging, APIs and security features of the fabric.

- cd integration-fabric-umu-basics/test
    - kafka folder:
        - ./external-kafka-consumer-test.sh: Start a kafka consumer from the external metalLB IP
        - ./external-kafka-producer-test.sh: Start a kafka producer from the external metalLB IP
        - ./internal-kafka-consumer-test.sh: Start a kafka consumer from the internal service, and using the internal name (kafka-service)

    - ./internal-kafka-producer-test.sh: Start a kafka producer from the internal service, and using the internal name (kafka-service)

- smd-service folder:

    - ./external-smd-service-test.sh: Test the nginx service from external metalLB IP
    - ./internal-smd-service-test.sh: Test the nginx service from the internal service, and using the internal name
    - ./show_log.sh [v1/v2]: show the log for the different nginx instances

- external-service folder:

    - ./external-service-test.sh: HTTP request against a service placed outside of the SMD

- security folder:

    - Different examples for applying istio security policies

Clean-up:

```
cd integration-fabric-umu-basics ./clean-up.sh
```

[end of document]