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

D5.2: First 5G security testing infrastructure implementation and preliminary results

Version: v1.1

Deliverable type	R (Document, report)
Dissemination level	PU (Public)
Due date	31/10/2021
Submission date	29/10/2021
Lead editor	Charalampos Kalalas (CTTC)
Authors	Ricard Vilalta, Pol Alemany, Roshan Sedar, Charalampos Kalalas, Raul Muñoz (CTTC), Wissem Soussi (ZHAW), Antonio Pastor, Juan Carlos Caja (TID), Chafika Benzaid, Othmane Hireche, Tarik Taleb (AALTO), Rodrigo Asensio, Noelia Pérez, Alejandro Molina (UMU), Vincent Lefebvre, Gianni Santinelli (TAGES), Edgardo Montes de Oca, Huu Nghia Nguyen, Manh-Dung Nguyen (MI), Geoffroy Chollon (TSG), Orestis Mavropoulos, Sabina Sandia (CLS), Jean-Philippe Wary (Orange), Cyril Dangerville (TSG), Klaas-Pieter Vlieg (EURES)
Reviewers	Vincent Lefebvre (TAGES), Dhouha Ayed (TSG)
Work package, Task	WP5, T5.2, T5.3
Keywords	Security testing infrastructure; Integration and functional verification; Key performance indicators; Operational validation; Integration of security and trust/liability enablers.

Abstract

This deliverable describes the first implementation of the 5G security testing infrastructure environment and details the test sequences defined for the functional verification of INSPIRE-5Gplus test cases (TCs). Three new demonstrators are introduced towards a thorough validation of INSPIRE-5Gplus architectural components and evaluation against key performance indicators. This deliverable also reports preliminary results which demonstrate the progress towards the integration of enablers in the TCs.

Document revision history

Version	Date	Description of change	List of contributor(s)
v0.1	01/07/21	Table of Contents (ToC)	Charalampos Kalalas, Raul Muñoz (CTTC)
	30/07/21	Inputs from partners	All authors
	27/08/21	Inputs from partners	All authors
v0.2	10/09/21	Slight modifications in ToC structure; Inputs from partners	All authors
v0.3	17/09/21	Inputs from partners	All authors
	23/09/21	Editorial updates, text polishing, final inputs	All authors
v0.4	24/09/21	Version submitted for internal review	Vincent Lefebvre (TAGES), Dhouha Ayed (TSG)
v0.5	11/10/21	Address review comments	All authors
v0.6	19/10/21	Editorial corrections	Charalampos Kalalas (CTTC)
v0.7	20/10/21	Final editing	Charalampos Kalalas (CTTC), Anja Köhler (EURES)
v0.8	22/10/21	Version submitted for GA approval	U. Herzog (EURES)
v1.0	29/10/21	Submission	U. Herzog (EURES)
v1.1	22/03/23	Corrections to address final Project Review Recommendations	Charalampos Kalalas (CTTC)

List of contributing partners, per section

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

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 first implementation of the 5G security testing infrastructure environments developed in the context of INSPIRE-5Gplus project activities.

Specifically, the content of this deliverable includes:

- Updates on the specifications and operating principles of certain test cases (TCs) and definition of new operational environments, called demonstrators, which leverage on the advanced security components developed in the TCs to provide an extensive coverage of the INSPIRE-5Gplus High-Level Architecture (HLA) functionalities.
- An integration methodology framework acting as a representative 5G infrastructure which ensures the continuous integration and re-configuration of the developed INSPIRE-5Gplus enablers along with a detailed description of the functional verification tests performed for each TC.
- A preliminary implementation of the INSPIRE-5Gplus closed loop on top of the first instantiation of the HLA in a multi-site environment.
- A set of quantifiable key performance indicators (KPIs), stemming from the development of specific security and trust/liability INSPIRE-5Gplus enablers. The overall goal is to set up a baseline of assessment criteria which should be fulfilled by the enablers involved in the demonstrators for operational validation. Each KPI definition is accompanied by the evaluation methodology steps followed for its assessment.
- Preliminary results pertaining to the status of each TC towards the integration of relevant enablers and testbeds as well as the verification of security components against pre-defined tests for each TC.

Table of Contents

List of Figures

List of Tables

Abbreviations

5G-PPP	5G Infrastructure Public Private Partnership
ADF	Anomaly Detection Framework
AI	Artificial Intelligence
AOI	Average Outage time Interval
ASP	Attack Success Probability
ATF	Average Time interval between Failures
BSAF	Back Situation Awareness Function
CCAM	Cooperative, Connected and Automated Mobility
CPU	Central Processing Unit
DDoS	Distributed Denial-of-Service
DM	Data Migration
DoS	Denial-of-Service
DTLS	Datagram Transport Layer Security
E2E	End-to-End
EMS	Element Management System
ET	Enforcement Time
FM	Forwarded Message
FN	False Negatives
FP	False Positives
GDPR	General Data Protection Regulation
HLA	High-Level Architecture
HPA	Horizontal Pod Autoscaling
HSPL	High-level Security Policy Language
I2NSF	Interface to Network Security Functions
IPSec	Internet Protocol Security
IT	Initial Time
ITS	Intelligent Transportation System
KACD	Kubernetes Admission Controller Delegator
LM	Last Message
MDTA	Mean decision time for MTD action
ML	Machine Learning
MMFT	Montimage Monitoring Framework
MMT	Montimage Monitoring Tool
MNO	Mobile Network Operator

MSPL	Medium-level Security Policy Language
MOTDEC	Moving Target Defense Controller
MT	Migration Time
MTBF	Mean Time Between Failures
MTD	Moving Target Defense
MTDAC	MTD action cost
MTID	Mean Time to implement the MTD action
MTTC	Mean Time to Contain
MTTD	Mean Time to Detect
MTTR	Mean Time to Resolve
NBI	North Bound Interface
NF	Network Function
NFVI	Network Function Virtualization Infrastructure
NIS	Network and Information Security
NSI	Network Slice Instance
NST	Network Slice Template
OBU	On-board Unit
OptSFC	Optimizer of Security Functions
PDR	Packet Delivery Ratio
PGMTD	Protection gain of an MTD policy
PLR	Packet Loss Ratio
RAM	Random-access memory
QoS	Quality-of-Service
QoSO	Quality-of-Service gain/loss of the protected resources
RT-SSLA	Runtime Monitoring Security Service Level Agreement
RTT	Round Trip Time
SBA	Service Based Architecture
SDN	Software Defined Network
SECaaS	Security as a Service
SF	Security Functions
SFC	Service Function Chaining
SGX	Software Guard Extensions
SLA	Service Level Agreement
SMD	Security Management Domain
SNS	Secured Network Slice
SO	Security Orchestrator

SSB	Security Service Blockchain
SSLA	Security Service Level Agreement
SSLO	Security Service Level Objective
TC	Test Case
TEE	Trusted Execution Environment
TM	Trust Management
TN	True Negative
TP	True Positive
TRM	Trust Reputation Management
UPF	User Plane Function
UUID	Universally Unique Identifier
vAAA	virtual Authentication, Authorization and Accounting
vIDS	virtual Intrusion Detection System
VNF	Virtualized Network Function
VIM	Virtual Infrastructure Manager
vOBU	virtual On-board Unit
VSF	Virtual Network Security Functions
ZSM	Zero-touch network and Service Management

1 Introduction

D5.2 constitutes the second public deliverable of the INSPIRE-5Gplus project’s Work Package 5 (WP5) and reports the progress on the development of an integration and qualification environment for the functional verification of the defined test cases (TCs). The integration and verification environment allows to verify whether the enablers integration for each TC conforms to its specification before the actual deployment in the operational environments of the 5G testing facilities.

Aiming to provide an in-depth coverage and validation of the High-Level Architecture (HLA) functionalities proposed in WP2, we further introduce three operational environments, coined demonstrators, which build upon the advanced implementation of specific components of certain TCs. In the course of WP5 activities, our focus will be on the expansion of the demonstrators with additional TCs’ components in order to achieve: i) a holistic and in-depth applicability assessment of HLA features and ii) a rigorous feasibility evaluation of WP3/WP4 enablers in real-world scenarios.

This deliverable further presents a set of quantifiable key performance indicators (KPIs), stemming from the development of specific security and trust/liability WP3/WP4 enablers. The overall goal is to set up a baseline of assessment criteria which should be fulfilled by the involved enablers. Each KPI definition is accompanied by evaluation methodology steps followed for performance assessment. Finally, preliminary results pertaining to the functional verification and integration of WP3/WP4 enablers in the TCs are provided.

1.1 Scope

D5.2 reports on the consortium efforts towards the development of an integration and experimentation framework, with the objective of validating specific 5G security, trust, and liability TCs in INSPIRE-5Gplus. The specification of a common testing environment offers a common baseline for the functional verification of TCs. In addition, the advancements of security components in certain TCs pave the way for the definition of advanced operational environments, called demonstrators, which intend to showcase a holistic coverage of HLA components. D5.2 also aims to extend and corroborate the identified INSPIRE-5Gplus KPIs stemming from the development of specific security and trust/liability INSPIRE-5Gplus enablers reported in D5.1. Finally, preliminary results pertaining to the verification of security components against pre-defined tests are presented for each TC, with an objective to demonstrate the progress towards the integration of enablers in TCs.

1.2 Target Audience

The target audience of this deliverable are stakeholders, industry and academic working groups interested in security of 5G technologies, and infrastructure.

1.3 Structure

The rest of this deliverable is structured as follows. Section 2 summarizes the key updates brought on some security and trust TCs with respect to the reported status in D5.1. In addition, a concise definition of the three INSPIRE-5Gplus Demonstrators is provided along with a preliminary implementation of the INSPIRE-5Gplus closed loop. Section 3 elaborates on the integration and verification environment for the functional verification of the defined TCs. The scenario/workflow and the test sequences defined for each TC are detailed. Section 4 provides the definition and evaluation methodology of the KPIs which are expected to be monitored, measured and experimentally validated in the context of WP5. Finally, Section 5 reports preliminary results pertaining to the operational validation of each TC. Information related to the integration of certain TCs with ICT-17/18/19 platform scenarios is provided in the Appendix.

2 Updates of Security and Trust Test Cases

This section presents the key updates on certain security and trust TCs with respect to the reported status in D5.1. On top of the INSPIRE-5Gplus TCs, three new demonstrators are also introduced aiming to validate in-depth the HLA components and provide meaningful evaluation with respect to KPIs.

2.1 Update on Integration and Functional Verification Test Cases

2.1.1 Merge of TC3/TC4

TC3 is proposed to address problems around malicious traffic in control and data planes in 5G network, over a pervasive encryption environment such as 5G Core Service Based Architecture (SBA). To this end, it provides several enablers that can be deployed in the 5G infrastructure domain, acting as security agents following the INSPIRE-5Gplus HLA definitions. A key security agent is provided as Montimage Monitoring Tool (MMT) probe enabler, deployed over the network and exposed to possible introspection attacks. Systemic software security solution prevents such attacks, offering a shield leveraging Intel's Software Guard Extensions (SGX) enclave for the probe software and processed data integrity and confidentiality. An apparent shortcoming in TC3 is that it focuses on only one Security Management Domain (SMD) and the lack of reaction part and mitigation response to leverage the closed-loop management. TC4 provides end-to-end (E2E) cryptographic protection in services over 5G. It gives the capacity to work with more than one SMD, including access networks in 5G and increase security and trustworthiness. Also, TC4's ambition is to cover additional components in HLA architecture using several enablers, such as security orchestrator (SO), policy manager and trust reputation management. On the contrary, one shortcoming from TC4 is viewed as the difficulty to trigger multi-domain close-loop automation based on malicious activities and how they impact networks trustworthiness. Consequently, to increase the HLA demonstration, a merged effort from both TCs has been defined.

In turn, the combination into one TC3/TC4 will provide:

- Integrated testbed 5G infrastructure and service.
- Several SMD.
- Triggering conditions based on attacks or management decisions.
- Common closed-loop automation and management using several enablers.

Additionally, new network conditions will improve the demonstration of trust calculations over 5G infrastructure and services.

This TC3/TC4 test case's main objective is to deploy security mechanisms like Internet Protocol Security (IPSec) tunnelling and to detect/mitigate attacks such as non-legit Virtualized Network Function (VNF) creation and manipulation.

Figure 1 depicts the simplified procedure of the deployment of all the enablers that are the result of the merging process of TC3 and TC4. The enablers' main objective is to monitor and collect security information from different parts of the network and forward to the security analytics engine and Trust Reputation Manager (TRM) for further analysis and trust updates respectively. Thus, at E2E level, different monitoring and channel protection policies are requested to be enforced, so the E2E security orchestrator elaborates an SMD enforcement plan and orchestrates it across multiple SMDs (step 1.a,b,c). SMDs orchestrators receive the policies and they orchestrate and enforce them across the SMDs infrastructures to enforce monitoring policies in STA assets/MMT Probes and channel protection policies in Interface to Network Security Functions (I2NSF) agents (step 2.a,b,c). Finally, Systemic system will verify the new monitoring rules for the MMT Probe (3.a,b).

The reactive process of the test case is explained in Section 3.2.3.

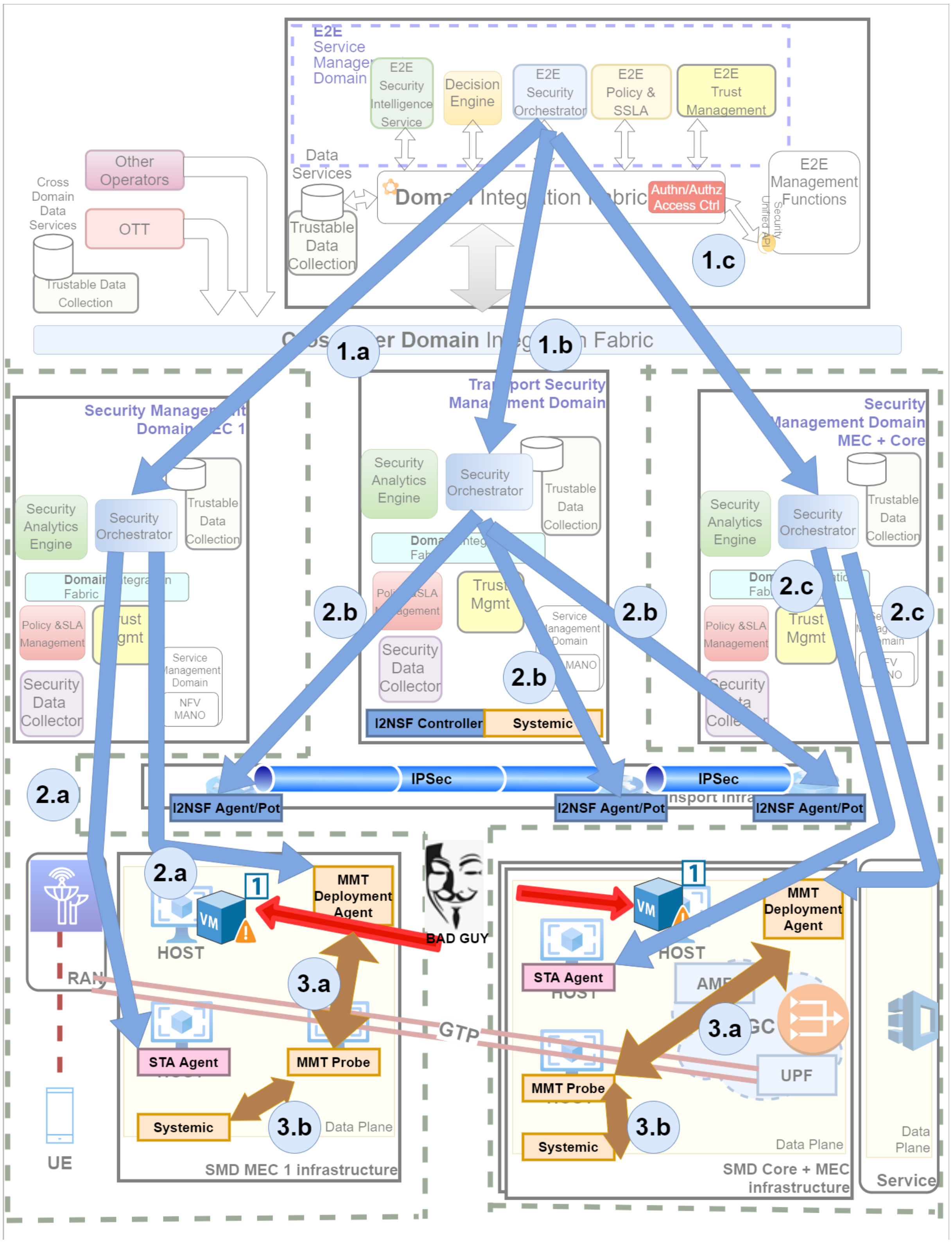


Figure 1: IPSec tunnelling deployment

2.1.2 Update on TC6

The main purpose of TC6 is to show the benefits of using INSPIRE-5Gplus architecture to deploy automatically a virtual device in which the vehicle is delegating the computational load of specific on-road calculations. In order to be compliant with security paradigms, TC6 is updated to grant security mechanisms by adding Datagram Transport Layer Security (DTLS) capabilities to the vehicle which will communicate with DTLS proxy deployed also automatically and located before the virtual On-board

Unit (vOBU). As the vehicle is a moving target, the TC6 is evolved to perform the migration procedure which reallocates both the vOBU and the vDTLS-proxy, and the associated cryptographic material.

When an OBU changes domain (0), a new vOBU must be deployed and for that, the gNB of the new domain performs a signalling process towards the 5G core (1) which will be managed at the Security Analytics Engine (2). The Security Analytics Engine, at the 5GC Management Domain level, forwards the gathered information to the E2E Security Management Domain (3) whose Security Orchestrator will first notify the E2E TRM which updates the computed value according to the mitigation outcome (4). If failed, the vOBU will no longer belong to that Domain. Then, the Security Orchestrator sends the notification to the Edge Virtual Domain (5.b) where the TRM calculates the reputation score of the specific vOBU that is going to be deployed (6), updating its score once the vOBU is ready, as well as in each successful/failed migration process. The Transport Domain which has also received the notification (5.a), instructs the re-routing of the current vOBU traffic to the new one (7, 8 and 9). Finally, the Edge Virtual Domain instantiates and released the new vOBU (10) and its corresponding OBU traffic is forwarded through it (11).

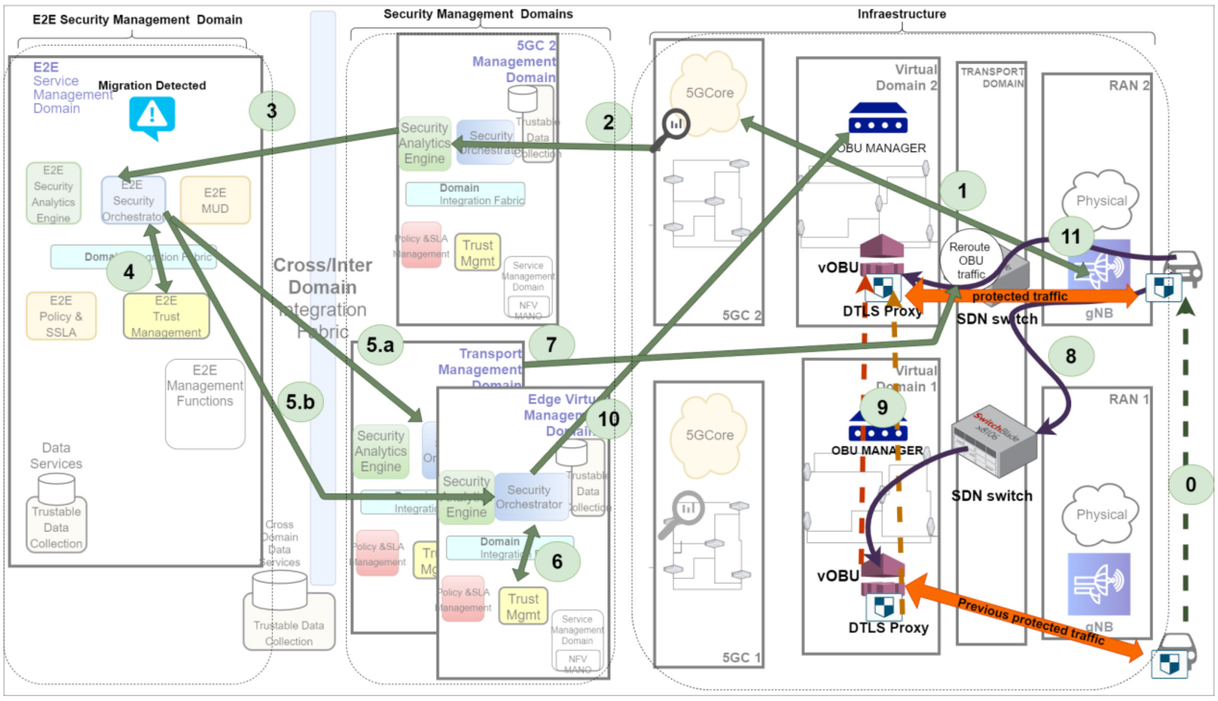

E2E Security Management Domain
Security Management Domains
Infraestructure
E2E Service Management Domain
Migration Detected
E2E Security Analytics Engine
E2E Security Orchestrator
E2E MUD
Domain Integration Fabric
E2E Policy & SSLA
E2E Trust Management
E2E Management Functions
Data Services
Trustable Data Collection
Cross/Inter Domain Integration Fabric
Cross Domain Data Services
5GC 2 Management Domain
Security Analytics Engine
Security Orchestrator
Trustable Data Collection
Policy &SLA Management
Trust Mgmt
Service Management Domain
NFV MANO
Transport Management Domain
Edge Virtual Management Domain
5GCore
5GC 2
5GC 1
Virtual Domain 2
Virtual Domain 1
OBU MANAGER
vOBU
DTLS Proxy
TRANSPORT DOMAIN
Reroute OBU traffic
protected traffic
SDN switch
Previous protected traffic
RAN 2
RAN 1
Physical
gNB
0
1
2
3
4
5.a
5.b
6
7
8
9
10
11

Figure 2: vOBU deployment

2.1.3 Update on TC7

The goal of TC7 is to provide a damage control mechanism to protect resources. TC7 supposes that the network layer is unable to detect an undergoing Distributed Denial-of-Service (DDoS) attack. As it goes unmitigated, such an attack may consume all available resources. Left alone, this situation will lead to a deprivation of resources.

The TC7 update was to consider a potential implementation inside a Kubernetes platform. In this case, a Virtualized Network Function (VNF) takes the form of a container inside Kubernetes' Pod. This VNF is using computing resources to handle its workload. As the DDoS attack goes on, the Kubernetes platform will automatically detect the Central Processing Unit (CPU) usage spike and trigger a scale up in order to increase the number of Pods. After a while, the Kubernetes platform will be full and unable to instantiate new services. To support this TC update, an auto-scaling delegator was developed specifically in the context of INSPIRE-5Gplus project. This new component Kubernetes Admission Controller Delegator (KACD) integrates with the Kubernetes API to intercept the scaling events and forward them to an external component for validation. In this case, the DDoS Mitigator enabler will perform this validation.

Eurescom started to participate in TC7 in Q3/2021 and will contribute to the testbed and focus on Denial-of-Service (DoS) detection and mitigation techniques and tools.

2.2 Update on Demonstrators for Validation and KPI Evaluation

This section offers a concise description of the three new INSPIRE-5Gplus demonstrators that will be used for validation of the HLA components and KPI evaluation. The three demonstrators aim to provide a holistic coverage of the entire spectrum of HLA functionalities as summarized in Table 1. In the course of INSPIRE-5Gplus activities, these demonstrators are expected to further develop and gradually integrate security components from several TCs.

Demonstrators	Category with respect to HLA													
	Secure Slice Management	SSLA/Policy Manager	Security Orchestrator	Security Analytics (using ML)	Decision Engine	Security Enforcement and Control	Policy Repository	Security Assessment	Security Data Collector	Security Agent	Integration Fabric	Trust & Liability Management	E2E ZSM Security Management	Trustable Data Services
Demo 1 – Security Management Closed Loop	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Demo 2 – Trust & Liability Management				X	X					X		X		X
Demo 3 – Moving Target Defense			X	X	X	X		X	X	X			X	

Table 1: Coverage of HLA functionalities by INSPIRE-5Gplus Demonstrators

2.2.1 Demonstrator 1: Security Management Closed Loop

The first demonstrator will extend and enrich the current status of the security management closed loop to validate multiple INSPIRE-5Gplus' security enablers in the Zero-touch network and Service Management (ZSM) multi-domain approach defined in the HLA. Specifically, the demonstration will showcase an example of providing proactive security to the 5G infrastructure at the E2E level and various SMDs by enforcing different Security Service Level Objectives (SSLOs) that are specified in a Security Service Level Agreement (SSLA), monitoring policies and channel protection policies. The different monitoring policies will pursue to configure different tools at different points of the 5G infrastructure focused on detecting different threats. Channel protection policies will pursue to provide channel protection properties to user traffic. The policies' orchestration and enforcement processed will be driven by trust metrics. Therefore, the selected security enablers or security agents that will enforce the security policies will depend on their trust scores. For the demonstration, channel protection policies will be enforced through the Software Defined Network (SDN)-based I2NSF IPSec solution. Monitoring policies will be enforced by MMT and STA agents. Proof of Transit agents will also provide metrics about the IPSec status. In addition to the proactive ZSM security orchestration and management automation, this demonstrator will also showcase the ZSM reactive behaviour. After the E2E and SMD policies enforcement, a malicious operator will deploy a compromised 5G component that will attack the 5G core. Both, Systemic Security as a Service (SECaaS) protected version of MMT Probe and STA will detect different aspects of the threat (according to the monitoring policies) and they will alert the Security Analytics Engine. This alert will unchain a new reactive policy enforcement at E2E and SMD levels. The enforcement again will consider the new trust scores that evolved according to the current status of the infrastructure and the received alerts from other architectural elements. Thus, the demonstration will finish by enforcing the reactive countermeasures that will close the Security Management closed loop at both levels, E2E and SMD. Demonstrator 1 will be led by UMU (academic) and TID (industry). Participating partners are: UMU, TID, TSG, CTTC, AALTO, MI, UOULU, NCSRD, EURES and TAGES.

2.2.2 Demonstrator 2: Trust and Liability Management

The aim of this proposal is to demonstrate the investigated concepts of trust and liability management on a virtualized infrastructure for a 5G type ecosystem.

The existing or planned legal and standard framework and vertical needs shows that 5G and B5G infrastructures will have to meet heterogeneous requirements (the EU Cybersecurity Act, NIS directive and regulations or standards related to 5G verticals like e-Health, Transport, Energy, Vehicular, Seveso industries), and be able to dynamically adapt (almost in a near real time). The strategy to implement the highest level of security may not be sustainable and scalable as some requirements may be incompatible and most use cases do not need the strongest security level. Another important point is verticals could be reluctant to pay for services that they do not need or use. More, maintaining such a security level for all network and service components is an overloading task that could increase the costs of some configurations in an inconsiderate manner.

A first objective of Demonstrator 2, centred on Trust and Liability related services, is to implement on-demand SSLA and deliver evidences of SSLA deployment and operation over the targeted infrastructure. Demonstrator 2 will illustrate this concept of vertical SSLA deployment through specific commitment of vertical service isolation over the proposed infrastructure. Critical services, subject to the Network and Information Security (NIS) directive, have to demonstrate their fulfilment to slice isolation (i.e., a legal obligation), and Demonstrator 2 may propose a way to dynamically manage and serve those constraints.

A second objective of Demonstrator 2 is to generate, through combination of KPI measurements, evaluation and attestation systems, evidence of KPI measure or SSLA operations on infrastructures specific components:

• On-demand-Probes: Allow third party vertical to evaluate or re-evaluate a committed SSLA/KPI on-the-fly. Demonstrator 2 will propose a short catalogue of authorized Probes / KPIs usable by Third Party as illustration of concepts.

• On-Demand-Evidence-Proof: Deliver evidences for localization, time and proof of origin of software measuring KPI and real KPI measured.

The second Demonstrator will be led by ZHAW (academic) and Orange (industry). Identified Demonstrator 2 partners: ZHAW, Orange, OPL, MI, TAGES, TSG and CLS in this first proposal. Demonstrator 2 could be further enhanced with potential other enablers.

2.2.3 Demonstrator 3: Moving Target Defense

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 will be the proactive change of the slice configuration 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 should 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 attacks.

An important aspect of this Demonstration 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.

The third Demonstrator will be led by NCSRD (academic) and MI (SME). The identified Demonstrator 3 partners are ZHAW, NCSRD and MI, and could be further enhanced with potential other enablers and partners if deemed necessary.

3 Integration and Verification Environment

This section first describes the methodology followed to verify whether the enablers' integration for each TC conforms to its specification before the actual deployment in the operation environments of the 5G testing facilities. The functional verification tests performed for each TC are further detailed along with a preliminary implementation of the INSPIRE-5Gplus closed-loop.

3.1 Integration Methodology

3.1.1 Platform description

One of the main activities carried out in the context of WP5 (since month 9) has been focusing on the specification of the appropriate testing environment for the integration and verification of the INSPIRE-5Gplus TCs. Each TC is composed of several WP3/WP4 enablers that are provided by different partners. The target integration and verification environment is a stripped-down multi-domain infrastructure exploited for continuous integration activities and functional verification tests of the developed WP3/WP4 enablers, containing the minimum required hardware and software components or mock-up versions of real ones.

The platform designed connects eight geographically distributed testbeds through a private VPN specifically created for the INSPIRE-5Gplus project. The different testbeds are provided by CTTC (Barcelona), NCSRD (Athens), UMU (Murcia), AALTO (Helsinki), OULU (Oulu), TID (Madrid), MI (Paris), and CLS (Eindhoven). As presented in Figure 3, all testbeds are connected to CTTC premises using open VPN tunnels as in the CTTC testbed there is a common set of services available for all the project partners. In addition to the eight testbeds, there are two partners (TSG and Eurescom) represented in a different way than the other partners. The reason for this is the fact that these two partners have access to the VPN but they do not bring physical resources, placing their WP3/WP4 INSPIRE-5Gplus enablers in one of the other testbeds. For example, TSG WP3/WP4 enablers will be placed in CTTC premises. Additionally, CTTC also provides an NFV infrastructure (NFVI) composed by an OpenStack node, a Kubernetes node, and an OSM controller, that can be used by any partner for each TC to integrate and verify the INSPIRE-5Gplus enablers developed in WP3 and WP4.

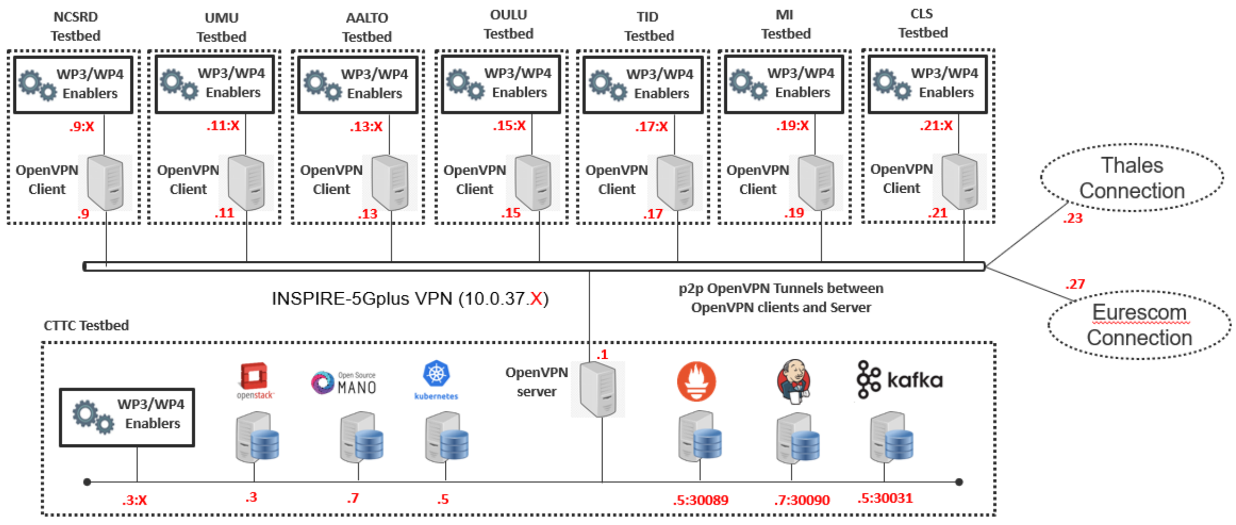


Figure 3: Platform architecture

In addition, three more services are offered: a Prometheus server to gather monitoring information, a Jenkins server to automate tests related with the WP3/WP4 INSPIRE-5Gplus enablers and, finally, a Kafka server as the selected technology to develop the INSPIRE-5Gplus Integration Fabric described in WP2 deliverables. Some partners use the Kafka server for the communication between enablers deployed in different testbeds. Kafka is an open-source event streaming platform. Kafka combines three key capabilities: to publish (write) and subscribe to (read) streams of events, including continuous import/export of data from other systems; to store streams of events durably and reliably for as long as you want; to process streams of events as they occur or retrospectively.

The demonstration performed during the mid-term review regarding the preliminary implementation of the INSPIRE-5Gplus closed loop was supported by this collaborative infrastructure by deploying different domains and planes in different testbeds of this platform design.

3.1.2 Test automation

In order to carry out this action, it was decided to make use of a well-known combination of two applications:

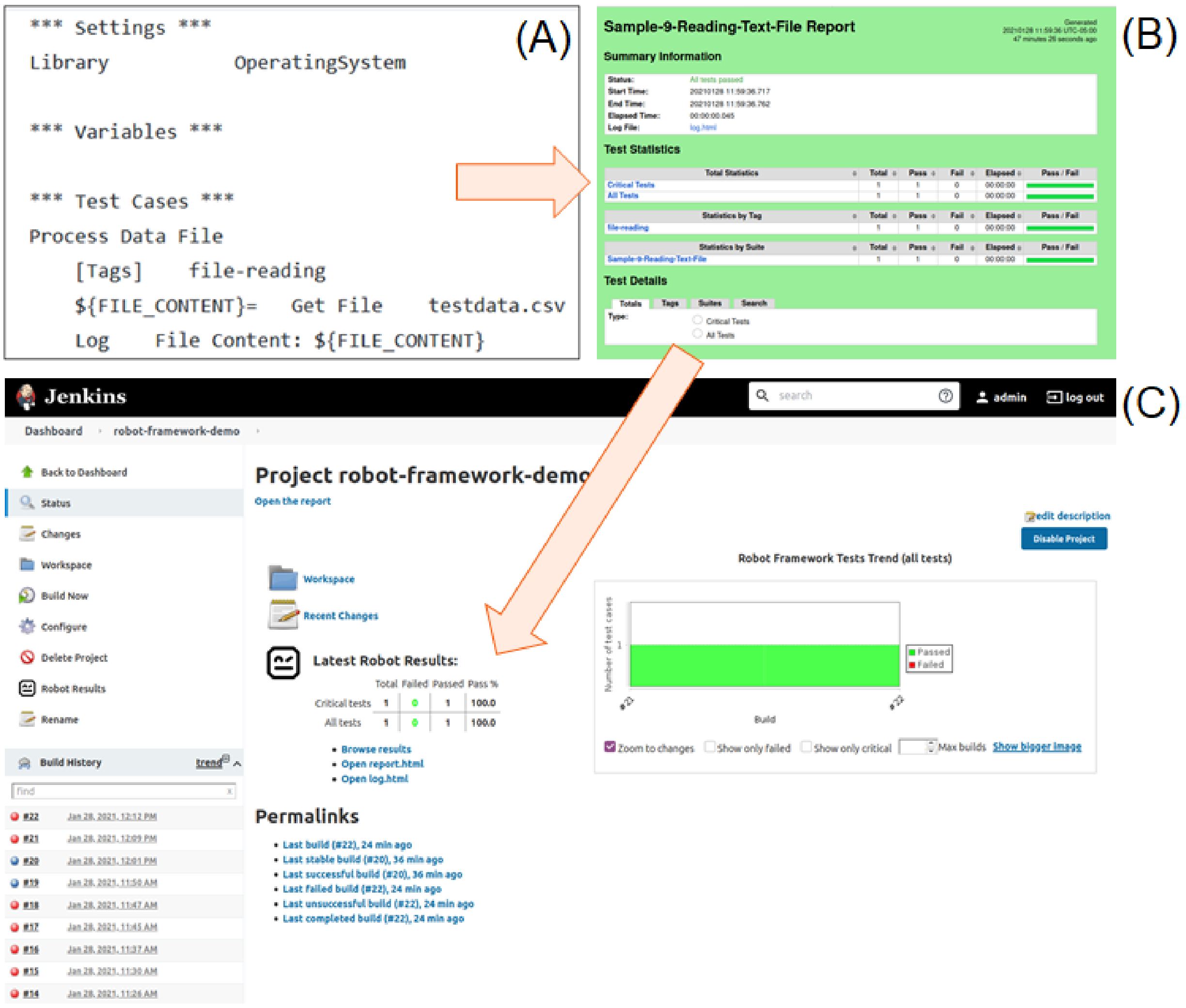
– Robot Framework[2]: It is a Python-based, extensible, keyword-driven testing automation framework. Among others, its main advantages are that its scripts are easy to write and read as they make use of a common set of human-readable keywords. Moreover, it has a lot of support libraries from both, Python and Java languages and it automatically generates a set of logs and reports with information related to each step in the functional verification tests in HTML format.

– Jenkins[3]: It is an open source automation server with multiple plug-ins allowing to create a stable environment to automate different actions of a project (i.e., building, deploying, etc.) by using the concept of "jobs". A pre-defined procedure that will be triggered and managed in automatic way. In the INSPIRE-5Gplus environment, it is used to define the functional verification tests involving the different WP3/WP4 INSPIRE-5Gplus enablers. Jenkins has a plug-in to work with Robot Framework, used to launch functional verification tests in the different testbeds and get their reports and results.

A resumed version of what Jenkins and Robot offer together is presented in Figure 4. It can be observed that there are three main actions: the Robot script defines the test actions (A), the HTML results generated by Robot (B) and the same results seen from Jenkins (C), in a more visual and easy format to interpret.

[2] https://robotframework.org/

[3] https://www.jenkins.io/



```
*** Settings ***
Library              OperatingSystem

*** Variables ***

*** Test Cases ***
Process Data File
    [Tags]    file-reading
    ${FILE_CONTENT}=   Get File    testdata.csv
    Log    File Content: ${FILE_CONTENT}
```

Sample-9-Reading-Text-File Report

Generated
20210128 11:59:36 UTC-05:00
47 minutes 26 seconds ago

Summary Information

Status:	All tests passed
Start Time:	20210128 11:59:36.717
End Time:	20210128 11:59:36.762
Elapsed Time:	00:00:00.045
Log File:	log.html

Test Statistics

Total Statistics	Total	Pass	Fail	Elapsed	Pass / Fail
Critical Tests	1	1	0	00:00:00	
All Tests	1	1	0	00:00:00	

Statistics by Tag	Total	Pass	Fail	Elapsed	Pass / Fail
file-reading	1	1	0	00:00:00	

Statistics by Suite	Total	Pass	Fail	Elapsed	Pass / Fail
Sample-9-Reading-Text-File	1	1	0	00:00:00	

Test Details

Totals | Tags | Suites | Search

Type: Critical Tests / All Tests

Jenkins — search — admin — log out

Dashboard › robot-framework-demo ›

Back to Dashboard
Status
Changes
Workspace
Build Now
Configure
Delete Project
Robot Results
Rename

Build History — trend

#22	Jan 28, 2021, 12:12 PM
#21	Jan 28, 2021, 12:09 PM
#20	Jan 28, 2021, 12:01 PM
#19	Jan 28, 2021, 11:50 AM
#18	Jan 28, 2021, 11:47 AM
#17	Jan 28, 2021, 11:45 AM
#16	Jan 28, 2021, 11:37 AM
#15	Jan 28, 2021, 11:30 AM
#14	Jan 28, 2021, 11:26 AM

Project robot-framework-demo

Open the report

edit description — Disable Project

Workspace
Recent Changes

Latest Robot Results:

	Total	Failed	Passed	Pass %
Critical tests	1	0	1	100.0
All tests	1	0	1	100.0

- Browse results
- Open report.html
- Open log.html

Robot Framework Tests Trend (all tests)

[✓ Zoom to changes] Show only failed · Show only critical · Max builds · Show bigger image

Permalinks

- Last build (#22), 24 min ago
- Last stable build (#20), 36 min ago
- Last successful build (#20), 36 min ago
- Last failed build (#22), 24 min ago
- Last unsuccessful build (#22), 24 min ago
- Last completed build (#22), 24 min ago

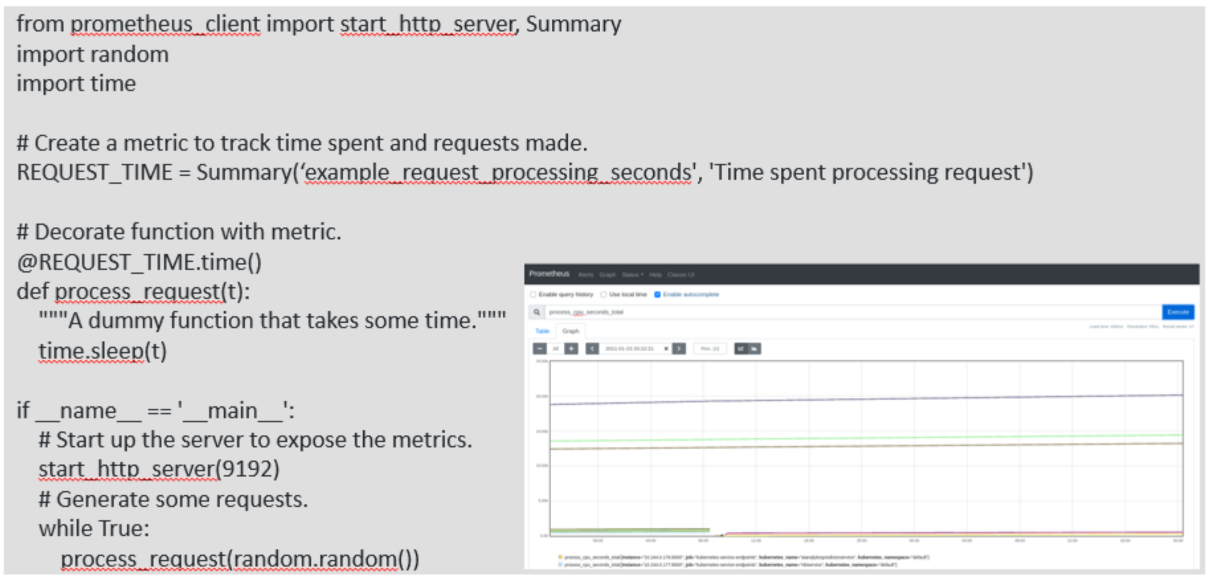
Figure 4: Simplified automation test procedure

3.1.3 KPI monitoring automation

Following the idea to have the biggest number of automated procedures possible, Prometheus[4] allows to automate the monitoring process and, thus, the actions to gather information and to generate alert events based on pre-defined requirements. Prometheus allows monitoring from different data sources, such as OpenStack or Kubernetes nodes but also dedicated exported metrics (KPIs). Precisely, this last case is the main objective to use Prometheus; the KPI monitoring. Using its client library, Prometheus extracts metrics from instrumented jobs (in our case, enablers and/or test cases). It stores all collected samples locally and runs rules over this data to either aggregate and record new time series from existing data or to generate alerts.

Grafana (or other API consumers) can be used to visualize the collected data. Each of the enablers and/or test cases needs to add instrumentation to their code via one of the Prometheus client libraries. These implement the Prometheus metric types. Figure 5 presents an example of a script to collect the selected metric and an API showing the historical record of the metric in a more visual way.

[4] https://prometheus.io/



```
from prometheus_client import start_http_server, Summary
import random
import time

# Create a metric to track time spent and requests made.
REQUEST_TIME = Summary(‘example_request_processing_seconds', 'Time spent processing request')

# Decorate function with metric.
@REQUEST_TIME.time()
def process_request(t):
    """A dummy function that takes some time."""
    time.sleep(t)

if __name__ == '__main__':
    # Start up the server to expose the metrics.
    start_http_server(9192)
    # Generate some requests.
    while True:
        process_request(random.random())
```

Figure 5: Prometheus script and monitoring example

3.2 Functional Verification Tests

3.2.1 Test Case 1

Table 2 summarizes the list of WP3/WP4 enablers used for the functional verification of TC1. As previously presented in D5.1, TC1 defines two different scenarios related to the EU 5GCroCo project. For the first scenario two test sequences are presented while for the second scenario an only one test sequence is detailed. We provide the details in the following subsection.

WP3/WP4 Enablers	Owner
Secured Network Slices for SSLAs	CTTC
Trusted Blockchain-based Network Slices	CTTC
Security Orchestrator	UMU
Policy Framework	UMU
SSLA Manager	TSG
Component Certification Tool	TSG

Table 2: WP3/WP4 enablers and partners developing each enabler for the functional verification of TC1

3.2.1.1 Scenario and workflow

The architecture for scenario 1 is presented in Figure 6, where all WP3 enablers involved and their internal relationships are illustrated.

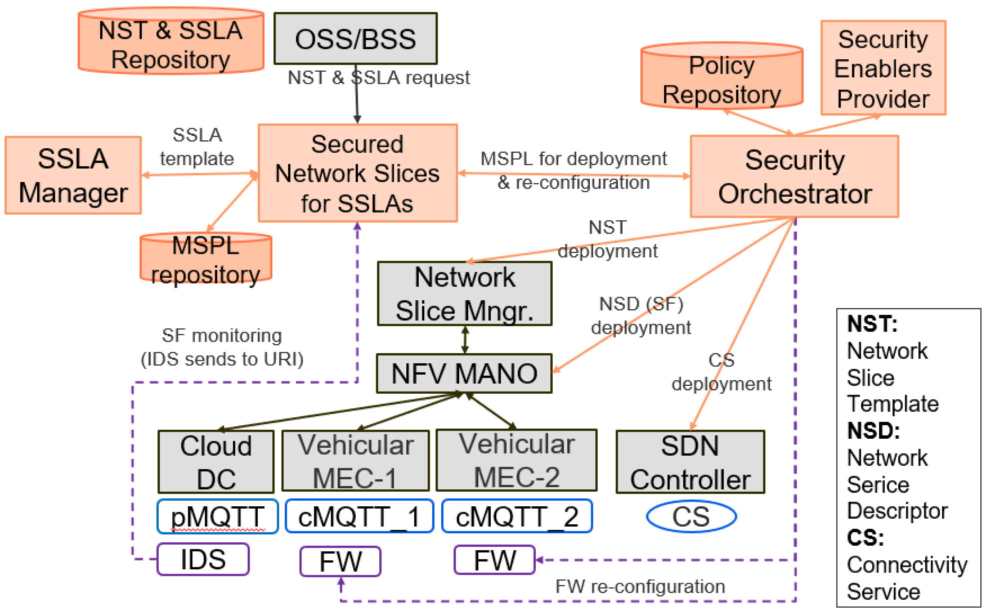


Figure 6: TC1 scenario 1 architecture and relationship with other enablers

Based on this architecture, the main actions (as illustrated in Figures 7-9) to present in the TC1 scenario 1 are the deployment of a Network Slice Template (NST) with an associated SSLA to add the correct security level to the deployed NSI and the KPI monitoring to validate that the selected Security Functions (SF) deployed next to the NSI are working properly. The following two workflows present the different steps to do during the NST deployment and the KPI monitoring workflows.

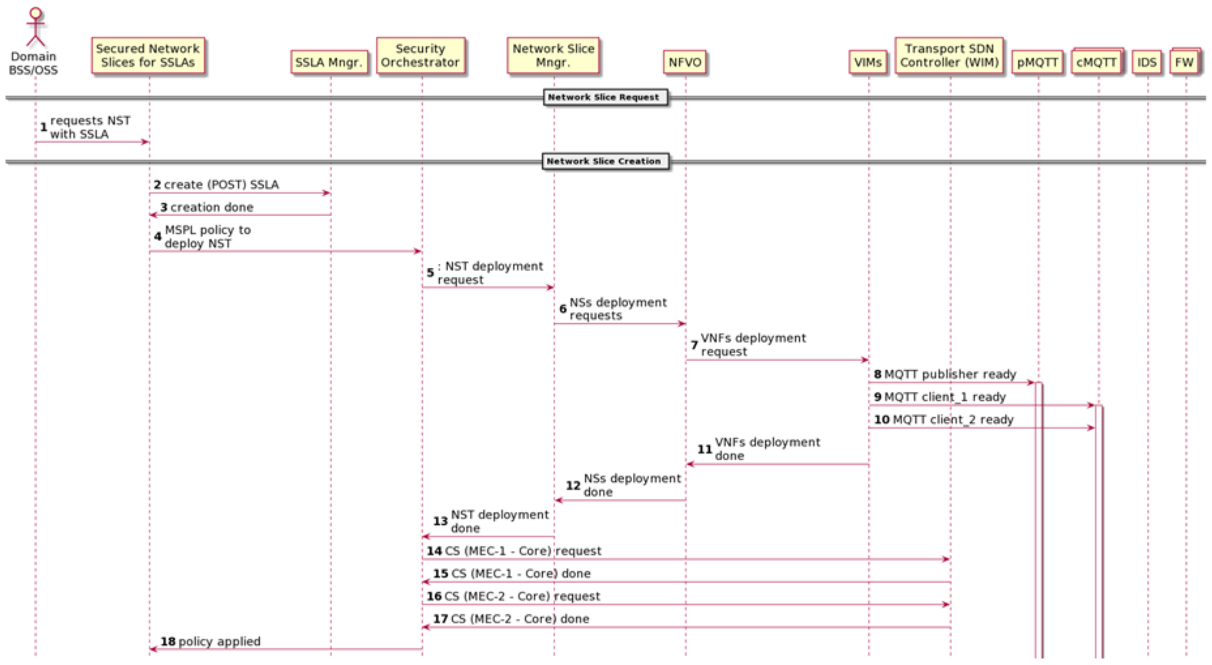


Figure 7: Network slice deployment (part 1)

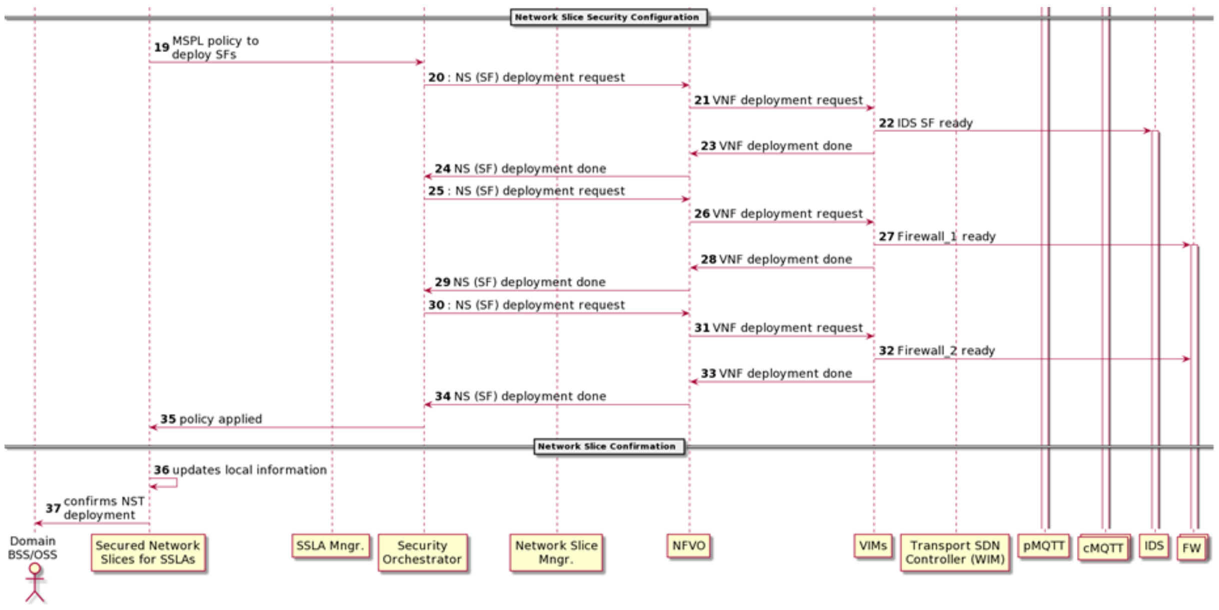

Network Slice Security Configuration
19 MSPL policy to deploy SFs
20 : NS (SF) deployment request
21 VNF deployment request
22 IDS SF ready
23 VNF deployment done
24 NS (SF) deployment done
25 : NS (SF) deployment request
26 VNF deployment request
27 Firewall_1 ready
28 VNF deployment done
29 NS (SF) deployment done
30 : NS (SF) deployment request
31 VNF deployment request
32 Firewall_2 ready
33 VNF deployment done
34 NS (SF) deployment done
35 policy applied
Network Slice Confirmation
36 updates local information
37 confirms NST deployment
Domain BSS/OSS
Secured Network Slices for SSLAs
SSLA Mngr.
Security Orchestrator
Network Slice Mngr.
NFVO
VIMs
Transport SDN Controller (WIM)
pMQTT
cMQTT
IDS
FW

Figure 8: Network slice deployment (part 2)

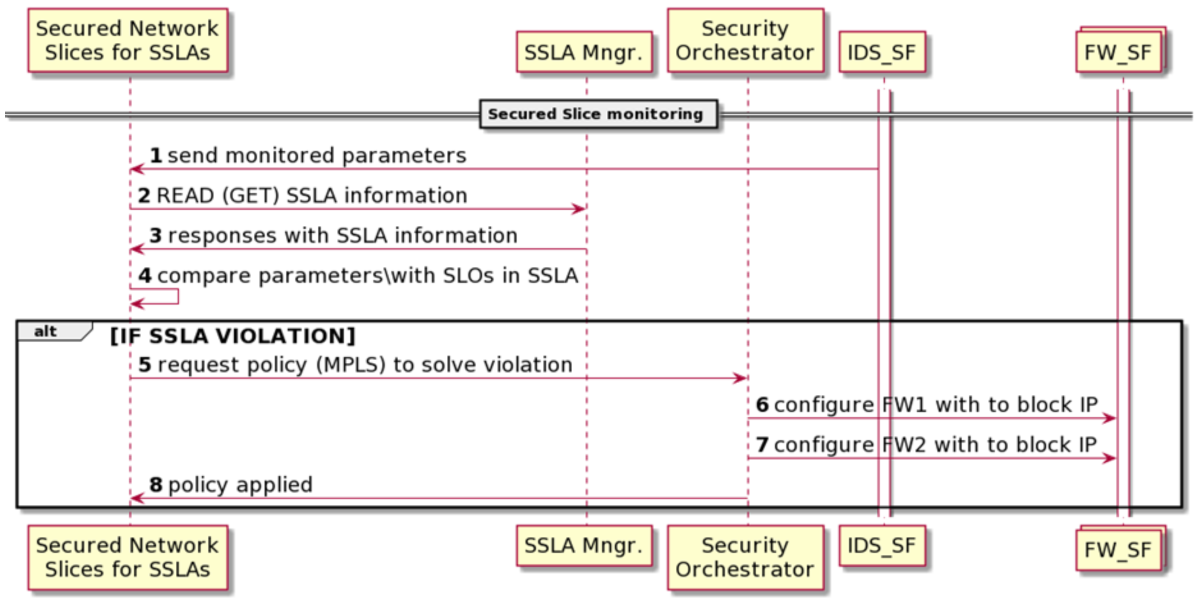

Secured Network Slices for SSLAs
SSLA Mngr.
Security Orchestrator
IDS_SF
FW_SF
Secured Slice monitoring
1 send monitored parameters
2 READ (GET) SSLA information
3 responses with SSLA information
4 compare parameters\with SLOs in SSLA
alt
[IF SSLA VIOLATION]
5 request policy (MPLS) to solve violation
6 configure FW1 with to block IP
7 configure FW2 with to block IP
8 policy applied

Figure 9: KPI monitoring workflow

The enablers used in the TC1 scenario 2 are presented in Figure 10 with the relationship between the CTTC enabler (i.e., Trusted Blockchain-based Network Slices (TBNS)) and the WP4 enabler Component Certification Tool (CCT) deployed by TSG. As it can be observed, the internal architecture of the TBNS enabler has two main elements: the element focused on transport networks and the element focused on network slicing actions. Our focus in INSPIRE-5Gplus is the second element as its security aspects are the main focus of our work.

Based on this architecture, two workflows are presented in Figures 11-13. The first workflow (Figure 11) is related to the validation process to check if a NST is certified or not to do what is expected to do. The second workflow (Figures 12-13) is the deployment of an E2E network slice across different operator domains using Blockchain as the tool to bring trust between the multi-domain operators.

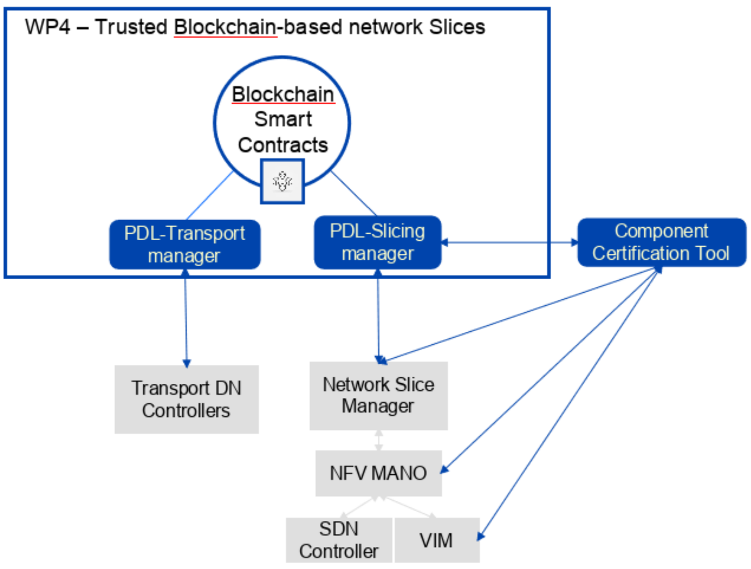


Figure 10: TC1 scenario 2 enablers (blue boxes) architecture

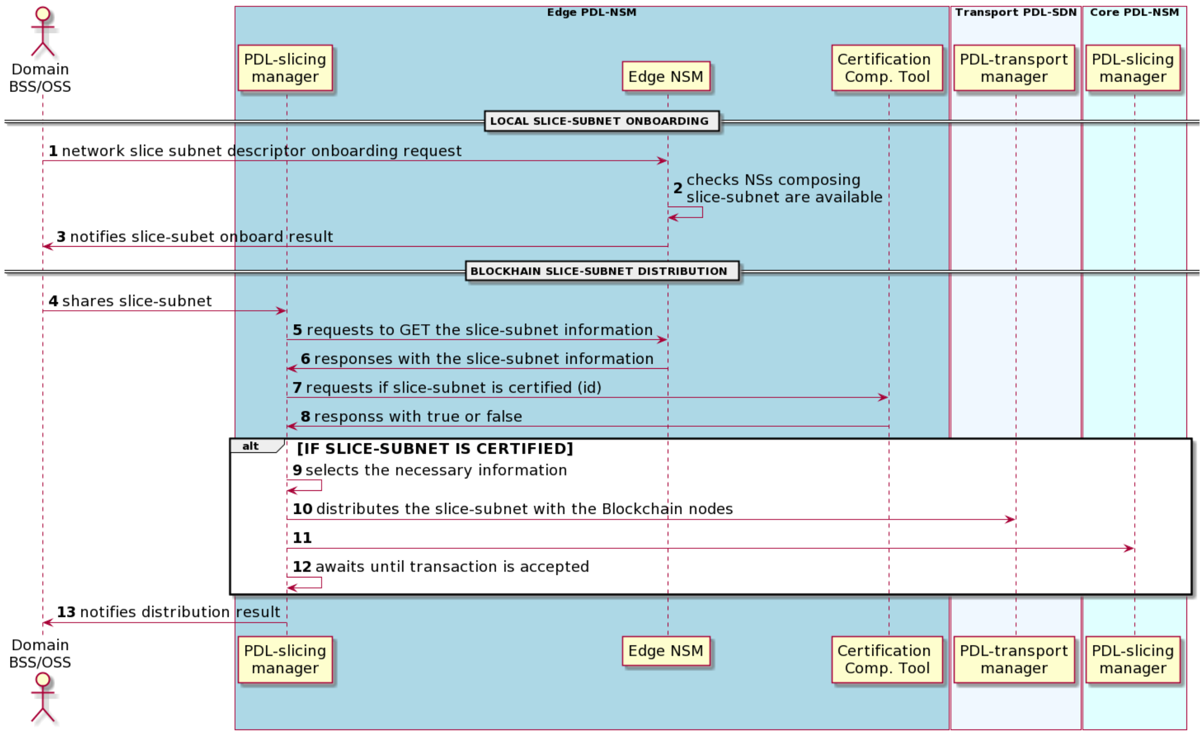

Edge PDL-NSM
Transport PDL-SDN
Core PDL-NSM
Domain BSS/OSS
PDL-slicing manager
Edge NSM
Certification Comp. Tool
PDL-transport manager
PDL-slicing manager
LOCAL SLICE-SUBNET ONBOARDING
1 network slice subnet descriptor onboarding request
2 checks NSs composing slice-subnet are available
3 notifies slice-subet onboard result
BLOCKHAIN SLICE-SUBNET DISTRIBUTION
4 shares slice-subnet
5 requests to GET the slice-subnet information
6 responses with the slice-subnet information
7 requests if slice-subnet is certified (id)
8 responss with true or false
alt
[IF SLICE-SUBNET IS CERTIFIED]
9 selects the necessary information
10 distributes the slice-subnet with the Blockchain nodes
11
12 awaits until transaction is accepted
13 notifies distribution result

Figure 11: Certificate verification of requested NST

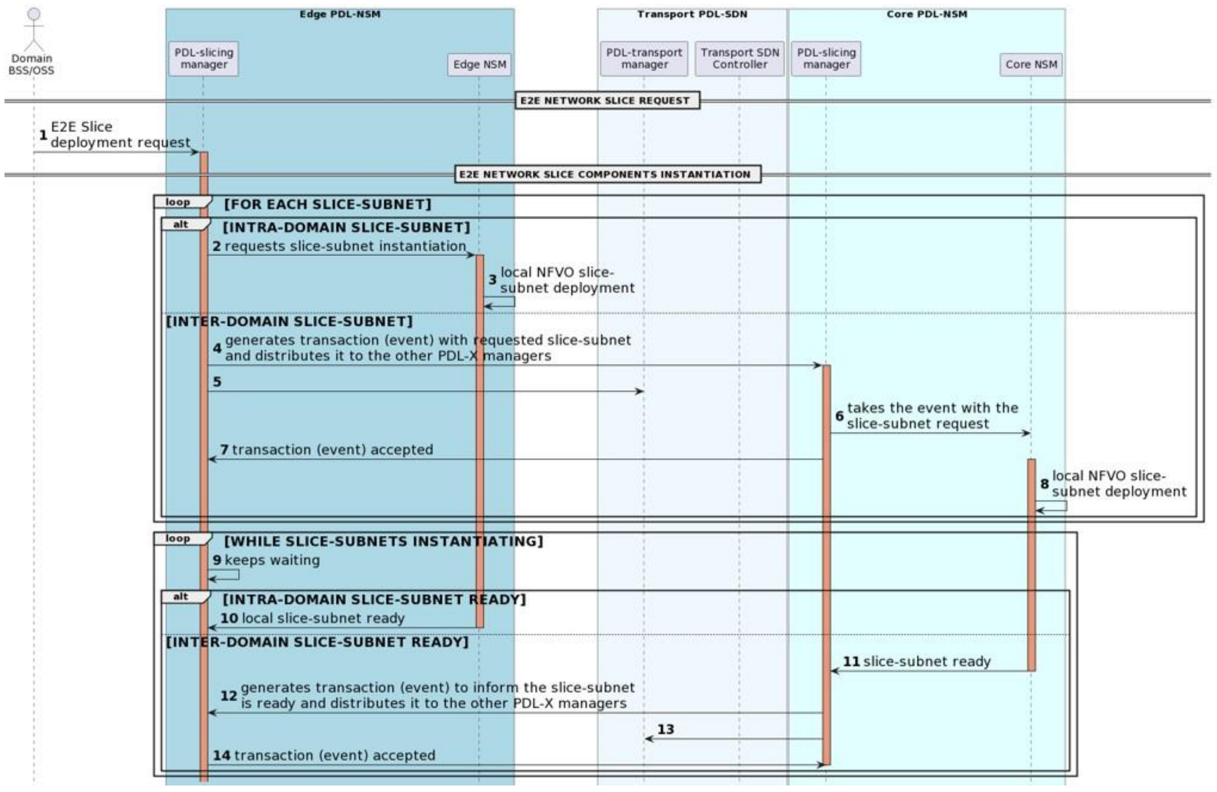

Domain BSS/OSS
Edge PDL-NSM
PDL-slicing manager
Edge NSM
Transport PDL-SDN
PDL-transport manager
Transport SDN Controller
Core PDL-NSM
PDL-slicing manager
Core NSM
E2E NETWORK SLICE REQUEST
1 E2E Slice deployment request
E2E NETWORK SLICE COMPONENTS INSTANTIATION
loop [FOR EACH SLICE-SUBNET]
alt [INTRA-DOMAIN SLICE-SUBNET]
2 requests slice-subnet instantiation
3 local NFVO slice-subnet deployment
[INTER-DOMAIN SLICE-SUBNET]
4 generates transaction (event) with requested slice-subnet and distributes it to the other PDL-X managers
5
6 takes the event with the slice-subnet request
7 transaction (event) accepted
8 local NFVO slice-subnet deployment
loop [WHILE SLICE-SUBNETS INSTANTIATING]
9 keeps waiting
alt [INTRA-DOMAIN SLICE-SUBNET READY]
10 local slice-subnet ready
[INTER-DOMAIN SLICE-SUBNET READY]
11 slice-subnet ready
12 generates transaction (event) to inform the slice-subnet is ready and distributes it to the other PDL-X managers
13
14 transaction (event) accepted

Figure 12: Blockchain-based network slice deployment (part 1)

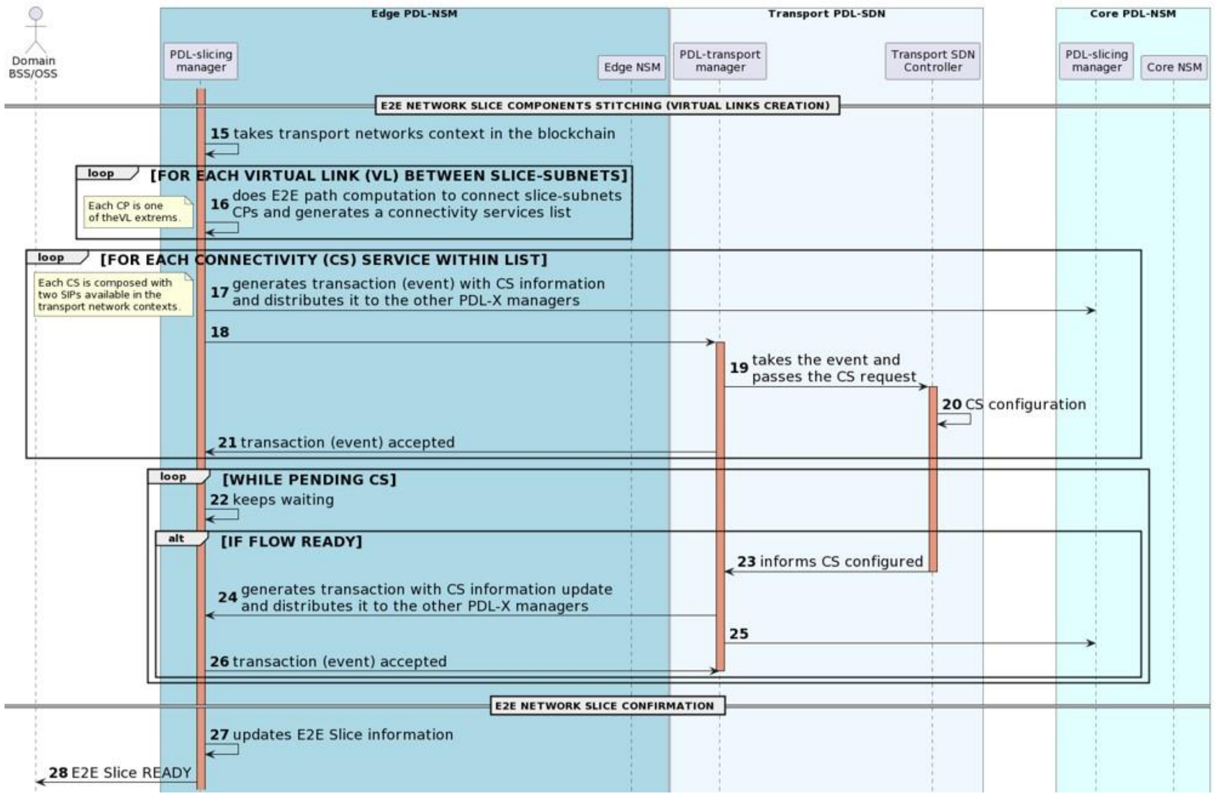

Domain BSS/OSS
Edge PDL-NSM
PDL-slicing manager
Edge NSM
Transport PDL-SDN
PDL-transport manager
Transport SDN Controller
Core PDL-NSM
PDL-slicing manager
Core NSM
E2E NETWORK SLICE COMPONENTS STITCHING (VIRTUAL LINKS CREATION)
15 takes transport networks context in the blockchain
loop [FOR EACH VIRTUAL LINK (VL) BETWEEN SLICE-SUBNETS]
Each CP is one of theVL extrems.
16 does E2E path computation to connect slice-subnets CPs and generates a connectivity services list
loop [FOR EACH CONNECTIVITY (CS) SERVICE WITHIN LIST]
Each CS is composed with two SIPs available in the transport network contexts.
17 generates transaction (event) with CS information and distributes it to the other PDL-X managers
18
19 takes the event and passes the CS request
20 CS configuration
21 transaction (event) accepted
loop [WHILE PENDING CS]
22 keeps waiting
alt [IF FLOW READY]
23 informs CS configured
24 generates transaction with CS information update and distributes it to the other PDL-X managers
25
26 transaction (event) accepted
E2E NETWORK SLICE CONFIRMATION
27 updates E2E Slice information
28 E2E Slice READY

Figure 13: Blockchain-based network slice deployment (part 2)

3.2.1.2 Definition of the test sequence

The first two test sequences belong to the TC1 scenario 1 in which the objective is the use of SSLAs on network slices with a deployed automation service to exchange road information. The first test verifies the integration between the Secured Network Slice Manager developed by CTTC and the SSLA Manager developed by TSG, while the second test verifies the integration with the security orchestrator developed by UMU. The third test presented belongs to the TC1 scenario 2, in this case the objective is a trusted collaborative management of the automotive service deployed using different domain operators through a Blockchain network.

Figure 14 presents the conceptual idea of both TC1 scenarios. Figure 14-A presents a situation in which an attacker sends messages simulating a fake car accident is placed in the middle of the road (TC 1 scenario 1), while Figure 14-B presents a set of different domains working in a trustworthy and collaborative way using the Blockchain (TC 1 scenario 2).

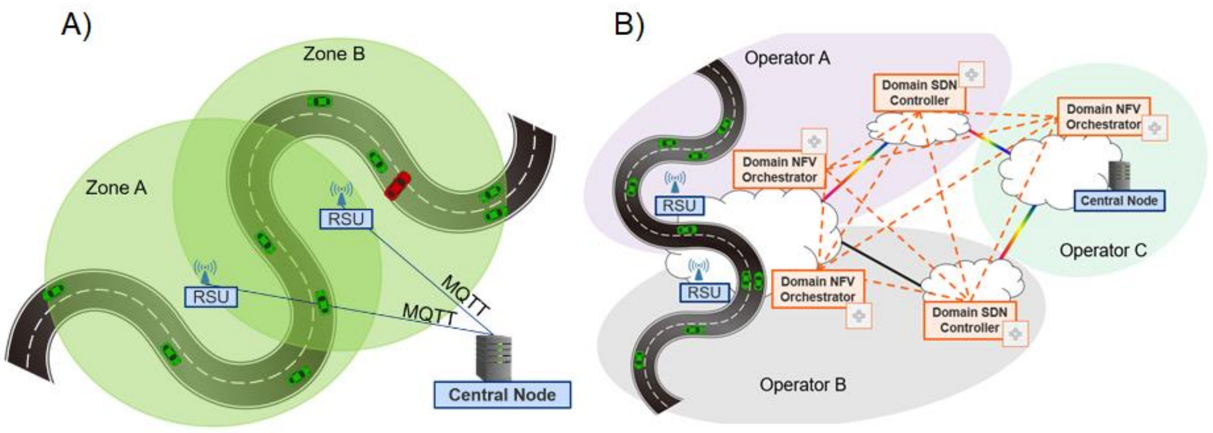


Figure 14: TC1 scenarios 1 and 2

Test Case Name		TC1_SecSlice-SSLA	
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 Medium-level Security Policy Language (MSPL) with all the information to request the deployment of a secured network slice.	
Scenario		Presented in Figure 14	
Test flow		Presented in Figure 15	
Test sequence	Steps	Description	Result
	1	Setting Up Environment	Prepare environment information to be used during the test.
	2	Reception of a request from the Domain BSS/OSS and its acceptance.	Request is accepted.
	3	The SNS requests to the SSLA Manager the SSLA information	The SSLA information is returned.

		based on the incoming information.	
	4	The SNS joints the Network Slice Descriptor and the SSLA information to generate the MSPL for the Security Orchestrator (SO).	An MSPL file following the expected data model.
Test verdict	If no error appears during the different steps defined and the NSI deployment begins, the test will be considered as successful.		

Table 3: Test sequence for TC1_SecSlice-SSLA

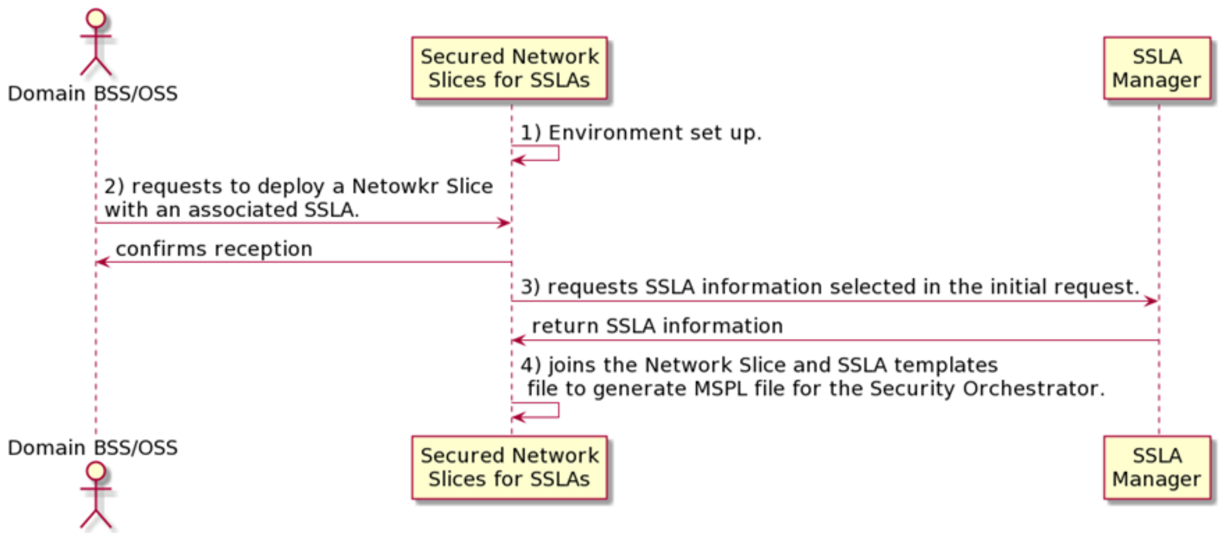

Domain BSS/OSS
Secured Network Slices for SSLAs
SSLA Manager
1) Environment set up.
2) requests to deploy a Netowkr Slice with an associated SSLA.
confirms reception
3) requests SSLA information selected in the initial request.
return SSLA information
4) joins the Network Slice and SSLA templates file to generate MSPL file for the Security Orchestrator.
Domain BSS/OSS
Secured Network Slices for SSLAs
SSLA Manager

Figure 15: Test flow for TC1_SecSlice-SSLA

Test Case Name	TC1_SecSlice-Orch		
Test Purpose	Validate the integration between the 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 NST to deploy.		
Scenario	Presented in Figure 14		
Test flow	Presented in Figure 16		
Test sequence	Steps	Description	Result
	1	Setting Up Environment	Prepare environment information to be used during the test.
	2	Reception of a request from the Domain BSS/OSS and its acceptance. (At this point the previous integration test is	Request is accepted and generation of the corresponding MSPL.

Test Case Name		TC1_SecSlice-Orch	
		applied as this is an evolution from it).	
	3	The SNSM sends the defined MSPL file to the SO.	Acceptance of the request and the MSPL.
	4	NST deployment procedure	The NST is deployed, and a new Network Slice Instance (NSI) created.
	5	Policy applied	The SO informs back the SNSM about 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 deployment begins, the test will be considered as successful.	

Table 4: Test sequence for TC1_SecSlice-Orch

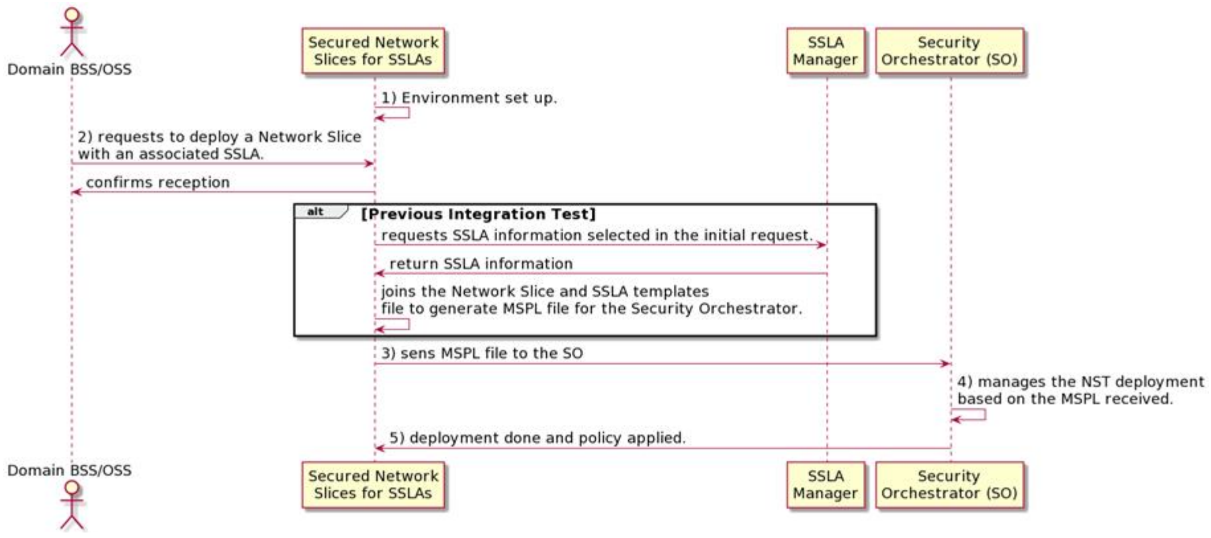

Domain BSS/OSS
Secured Network Slices for SSLAs
SSLA Manager
Security Orchestrator (SO)
1) Environment set up.
2) requests to deploy a Network Slice with an associated SSLA.
confirms reception
alt
[Previous Integration Test]
requests SSLA information selected in the initial request.
return SSLA information
joins the Network Slice and SSLA templates file to generate MSPL file for the Security Orchestrator.
3) sens MSPL file to the SO
4) manages the NST deployment based on the MSPL received.
5) deployment done and policy applied.
Domain BSS/OSS
Secured Network Slices for SSLAs
SSLA Manager
Security Orchestrator (SO)

Figure 16: Test flow for TC1_SecSlice-Orch

Test Case Name	TC1_TBNS-CCT
Test Purpose	Validate the integration between the Trusted Blockchain-based Network Slices (TBNS) and the CCT enablers defined in WP4.
Description	The test will check if the interactions between the two enablers are well implemented. To do so, the TBNS will request if a descriptor has been certified (e.g., it does what it says it does) or not.
Scenario	Presented in Figure 14
Test flow	Presented in Figure 17

Test Case Name		TC1_TBNS-CCT	
Test sequence	Steps	Description	Result
	1	Setting Up Environment	Prepare environment information to be used during the test.
	2	Reception of a request to deploy a Network Slice from the Domain BSS/OSS and its acceptance.	Request is accepted.
	3	The TBNS requests if the selected descriptor is in its database and has been certified.	CCT accepts the request.
	4	The CCT answers back with the information about the descriptor requested.	A confirmation or negation about the certification is given.
	5	The TBNS adds the necessary information in the Network Slice Instance (NSI) descriptor defining whether the descriptor is certified or not.	The network slice deployment begins, and in the NSI descriptor there is the conclusion about the certification of the NSI.
Test verdict		If no error appears during the different steps defined and the NSI is correctly instantiated, the test will be considered as successful.	

Table 5: Test sequence for TC1_BLSlice-CCT

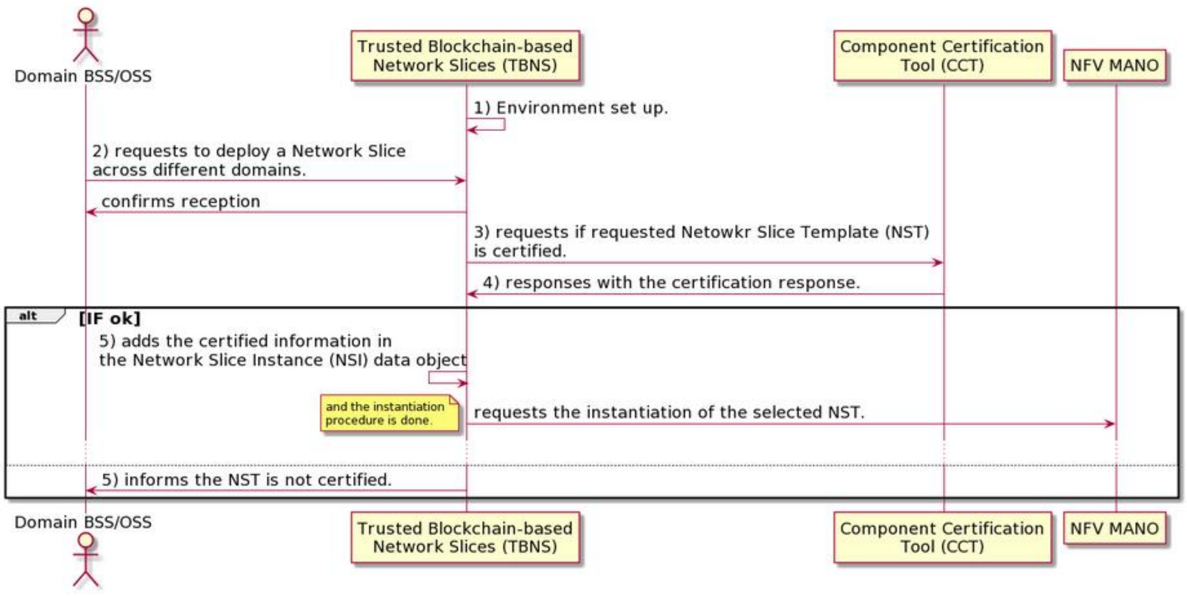

Domain BSS/OSS
Trusted Blockchain-based Network Slices (TBNS)
Component Certification Tool (CCT)
NFV MANO
1) Environment set up.
2) requests to deploy a Network Slice across different domains.
confirms reception
3) requests if requested Netowkr Slice Template (NST) is certified.
4) responses with the certification response.
alt
[IF ok]
5) adds the certified information in the Network Slice Instance (NSI) data object
and the instantiation procedure is done.
requests the instantiation of the selected NST.
5) informs the NST is not certified.

Figure 17: Test-flow for TC1_TBNS-CCT

3.2.2 Test Case 2

TC2, which defines a scenario showcasing and testing the SSLAs, involves the integration between the following enablers: the Policy Framework, the Security Orchestrator, the Decision Engine, the Security Agent (e.g., the MMT-Probe) and the Security Analytics Engine. The main objective of this test case is to formally define runtime monitoring SSLAs (RT-SSLAs) rules for real-time assessment and testing to verify the specified purposes and requirements in the 5G context. Firstly, the security functions of existing enablers are correctly implemented and can provide the necessary security capabilities, such as monitoring, filtering, encryption, etc., according to the user-defined SSLAs. Secondly, the security properties extracted from traffic and traces are verified to determine that the SSLAs rules are not violated in real time. Finally, the violations will automatically trigger self-healing and self-protection strategies in case of errors, for example, to redeploy a new instance of a service or update the firewall rules to block banned traffic.

Table 6 summarizes the list of WP3/WP4 enablers used for the functional verification of TC2.

WP3/WP4 Enablers	Owner
Policy Manager	UMU
Security Orchestrator	UMU
Decision Engine	TGS
Security Monitoring Framework	MI
Security Analytics Engine	MI

Table 6: WP3/WP4 enablers and partners developing each enabler for the functional verification of TC2.

3.2.2.1 Scenario and workflow

Figure 18 depicts the functional architecture of the TC2. User requirements are defined in a High-level Security Policy Language (HSPL) (e.g., in XML format like the Web Services Agreement Specifications) for specifying abstract security policies regardless of the underlying technology. This key feature of the framework allows multiple implementations and enforcement points for the same high-level policy. This level of abstraction also provides other important features such as allowing non-technical end users to specify general actions or protection requirements without possessing deep knowledge of the lower technical layers of the system.

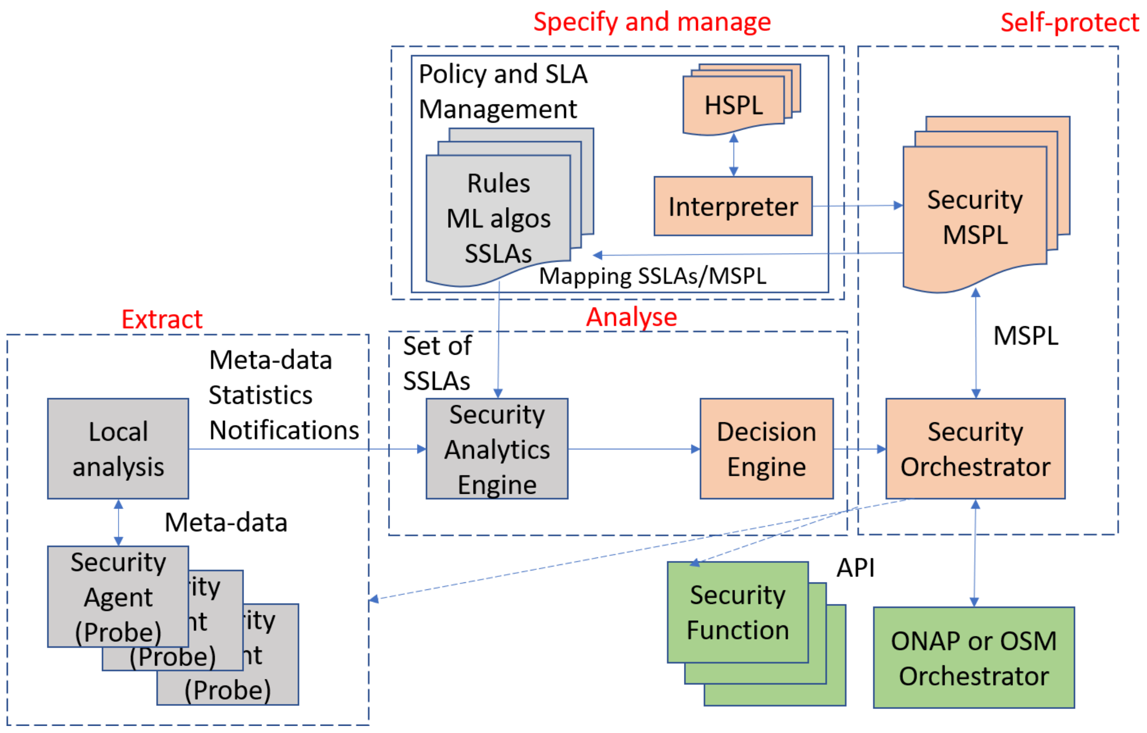


Figure 18: TC2 architecture and relationship with other enablers

3.2.2.2 Definition of the test sequence

Test Case Name		TC2_Assessment and enforcement of SSLAs	
Test Purpose		Test that the SSLAs defining the security rules and functions are respected during operation	
Description		The TC will be used to validate that the SSLA rules are correctly interpreted from the specified HSPL and MSPL. It will also ensure that the assessment and enforcement process works correctly and interacts with other components as expected.	
Scenario		Presented in Figure 18	
Test flow		Presented in Figure 19	
Test sequence	Steps	Description	Result
	1	Setting Up Environment	Prepare environment information to be used during the test
	2	Definition of HSPLs and SSLAs	HSPLs and SSLAs are defined, created and stored in a database
	3	The HSPLs are parsed, analysed, and translated into Security MSPL that is sent to an Orchestrator to deploy the required NFs and Security Agents (e.g., MMT-Probes). Eventually, the MSPL are converted to TOSCA or other formalisms that are required by	The SSLAs have the correct format and the key information is extracted from them

Test Case Name		TC2_Assessment and enforcement of SSLAs	
		the orchestrator (e.g., ONAP, OSM). - The MSPLs are parsed to determine what SSLAs rules need to be assessed and enforced. The set of rules are provided to the Policy and SSLA management and Security Analytics Engine functions (e.g., implemented by the Policy Manager and/or the MMT-Operator).	
	4	The Security Orchestrator uses MSPL to deploy/configure the MMT-Probes to collect statistics and metadata, and the Security NFs to enforce the security policies	The monitoring tool can collect metadata in real time, and the security functions needed to enforce the security policies are deployed
	5	The Security Analytics Engine notifies the Decision Engine with an alert whenever: - the security properties/rules are violated - the status of deployed security NFs (e.g., working as expected or incorrectly deployed/configured)	Security policies are assessed, and the reactions are performed, if needed
Test verdict		If no error appears during the different steps, the test will be considered successful.	

Table 7: Test sequence for TC2_Assessment and enforcement of SSLAs

Figure 19 shows the sequence diagram of TC2. The security requirements are defined by a user as HSPL. The SSLAs are rules that are specified and stored in a database to perform the real time assessment. The HSPLs are translated to security MSPLs that will be used by the Security Orchestrator[5], e.g., to deploy the probes to collect metadata in real time or to perform the remediation actions. MSPLs are also used to determine what SSLA rules should be used. The selected set of SSLA rules are: managed by the policy and SSLA management function; used by the probes to determine what metadata is needed to assess the SSLAs and eventually perform some local pre-analytics; and, used by the Security Analytics Engine function to analyse the extracted data, detect any violations of the SSLAs and inform the Decision Engine of any violations for further actions.

The probes provide the real time metrics and threshold values required by the SSLAs and perform local analytics. The metrics, statistics and results of local analytics provided by one or more probes are fed to the Security Analytics Engine to perform the final global assessment of the SSLAs. The Decision Engine will then trigger the corrective actions that could involve interacting with the Security Orchestrator or directly with the Security Functions and Controllers.

[5] In the case where other orchestrators need to intervene, the MSPL specification can be translated to the formalism supported by it (e.g., ONAP or OSM that use Tosca).

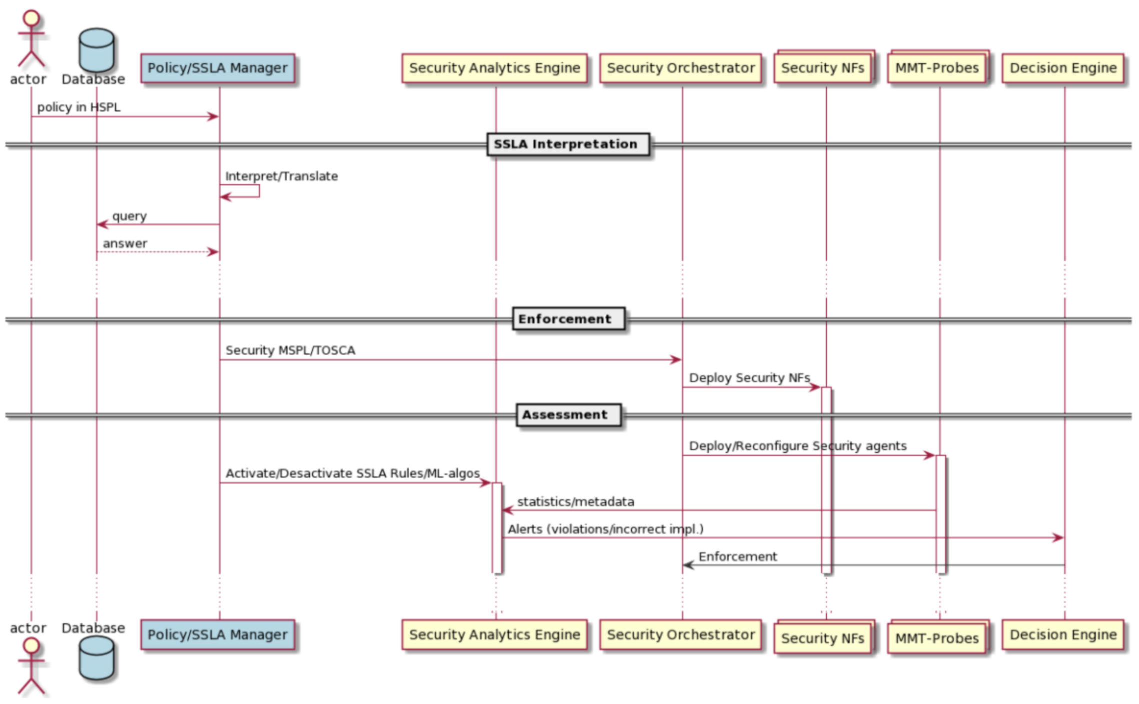


Figure 19: TC2 workflow

3.2.3 Test Case 3/Test Case 4

This test case merges TC3 and TC4 to provide a rich test case that encompasses multiple INSPIRE-5Gplus features by integrating different security enablers. Specifically, this improved test case enforces proactively channel protection and monitoring policies that will be enforced through a SDN based channel protection tool (I2NSF) and different monitoring tools such as MMT, STA and Proof of Transit nodes. Trust is also a key part of this test case. Trust Reputation Manager retrieves information of the underlying infrastructure that is used to calculate trust measures. These measures are used during the orchestration process to elaborate the orchestration plan to decide the best security enablers where security policies must be enforced. Besides, to validate the enforced policies, an attack will be produced from inside the 5G network. This attack will be detected by the different monitoring tools which will alert at SMD level. Apart from the possible reactions performed at SMD level, alerts will be also propagated to the E2E level (if required). New policies will then be enforced to mitigate the detected threats.

Table 8 summarizes the list of WP3/WP4 enablers considered in the definition of the TC3/TC4.

WP3/WP4 Enablers	Owner
Security Monitoring Framework (SMF)	MI
Systemic	TAGES
Smart Traffic Analyzer (STA)	TID
Security Orchestrator (SO)	UMU
Proof of Transit (PoT)	TID
Policy Framework	UMU
I2NSF IPsec	TID
Trust Reputation Manager (TRM)	UMU

Table 8: WP3/WP4 enablers and partners developing each enabler for the functional verification of TC3/TC4

3.2.3.1 Scenario and workflow

The main scenario is divided in two main stages. The proactive stage was introduced in Section 2.1.1 and it shows the security enablers and security assets deployment that are automatically deployed/configured as part of the proactive policy enforcement. Specifically, a security administrator will request the enforcement of high-level monitoring and channel protection policies from the E2E SMD. Those policies will be refined and sent to the involved SMDs. These are, those domains that require configurations or new deployments to accomplish the policy requirements. When each SMD receives the security policies, it performs a policy orchestration process that will select the most suitable (and trustable) security enabler or security agent on which the policy or policies should be enforced to. After this selection, security policies are translated to specific configurations that are enforced in the selected security asset. For this test case, monitoring policies will be enforced over STA agent and MMT Probe (according to trust metrics), whereas E2E channel protection policy will be enforced in two different SMDs by using the SDN-based I2NSF agents. Once the proactive security policies have been enforced, the proactive stage retires.

To unchain the reactive stage, malicious VNFs are deployed in the 5G infrastructure. Figure 20 shows the reactive scenario unchained after the malicious VNFs start attacking the network. Monitoring assets detect the issue (according to the proactive policies) and they send the information collected to the Security Data Collector (1.a/b) which will gather all the security information from different sources and forward it to the Security Analytics Engine (2.a/b). After the analytic procedure have been performed, the anomaly is detected, the attack is recognized, and an alert is sent to the E2E Security Analytics Engine (7.a/b) which shows the information to the system administrator. In parallel, The Trust Reputation Manager (TRM) also receives the alert data from the Security Data Collector (3.a/b) and updates the trust score of the monitored components/domains, lowering its values according to the alerts. These updates are really essential because they will prevent the deployment of the compromised VNFs in the future. Finally, when the system administrator receives the alert, she reacts by requesting the enforcement of new security policies (8). In this case, the countermeasure will isolate the compromised VNFs by enforcing filtering/forwarding policies through a high-rated trust score security enabler (9,10,11).

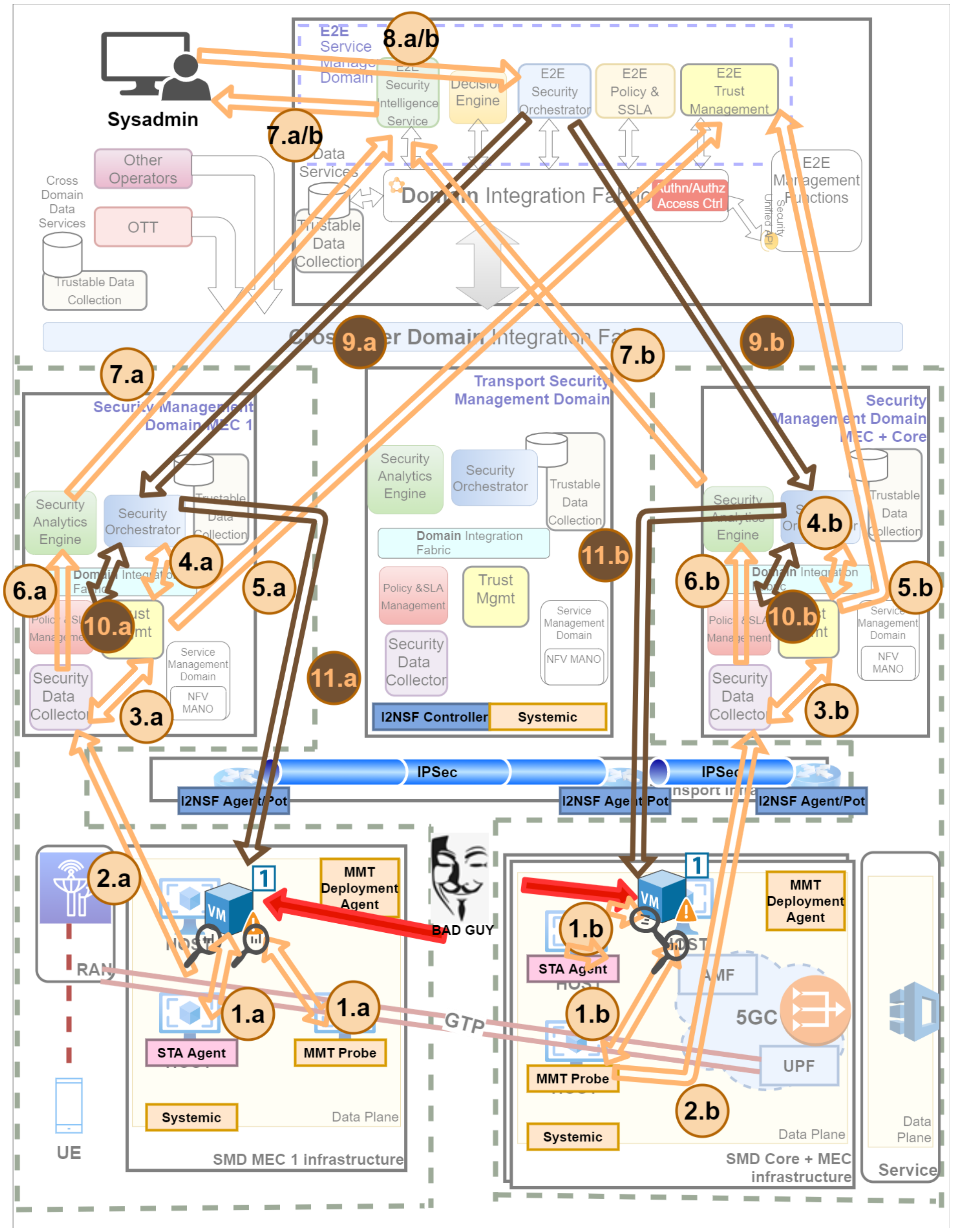


Figure 20: TC3/TC4 reactive workflow

3.2.3.2 Definition of the test sequence

To validate the TC, different test sequences between different security enablers have been defined. Thus, it is possible to validate each security enabler interaction before validating the whole use case scenario. Following tables detail the test sequences for the envisaged interactions. Besides, diagrams focused on each test sequence are also provided.

Test Case Name		TC3-4_SMF-Systemic	
Test Purpose		- Demonstrate how Systemic binary wrapping tool can be used to protect Montimage SMF code (and to be specific MMT probe code). - Demonstrate that the resulting protected version is protected against the threats given as attack in integrity and attack in confidentiality of both MMT code and its dependencies (MMT-probe) generated continuously on the fly during network operation.	
Description		The test objectives are threefold: i) demonstrate the set-up workflow, ii) demonstrate how the protected variant leverages automatically Intel SGX or alternatively Solidshield proprietary software-based secure environment pending SGX availability; iii) demonstrate how the protected version of MMT code is safeguarded against the integrity and confidentiality attacks on MMT code and as well as on rule tampering or rogue rule injection.	
Scenario		Presented in Figure 20	
Test flow		Presented in Figure 21	
Test sequence	Steps	Description	Result
	1	Setting Up Environment (install a SECaaS server, deliver user credential, check SGX enablement on the MMT targeted running platform.	Installed SECaaS server, MMT probe and main code are uploaded on the server, checked SGX enablement
	2	Protect both MMT code (main and probe) by use of User Interface. Input and access of required cryptographic elements (keys for authentication and encryption).	The code is protected. Keys are provisioned both ways (from and to the SECaaS).
	3	Deploy the protected variants of MMT probe and main code. Check MMT main code launch (as it depends on SGX-embedded or Solidshield-enabled Systemic routine). MMT main launch signifies that the code is self-authenticated, decrypted, launches and interacts with SGX-embedded (or alternatively Solidshield-embedded) systemic routine.	Protected SGX-enabled MMT main code launches
	2	Transmission by MI of protected MMT probe code	Protected MMT probe is authenticated before being called by MMT main

Test Case Name		TC3-4_SMF-Systemic	
	3	Attack in integrity by introspection on the running platform Attack in confidentiality by introspection on the running platform Attack by code injection (rogue dependency)	The three attacks are halted or significantly obstructed.
	4	Extraction of the protection project metadata	The encrypted metadata (appended to the MMT main protected version) is accessed and reflects the protection level of the code

Table 9: SMF-Systemic test sequence

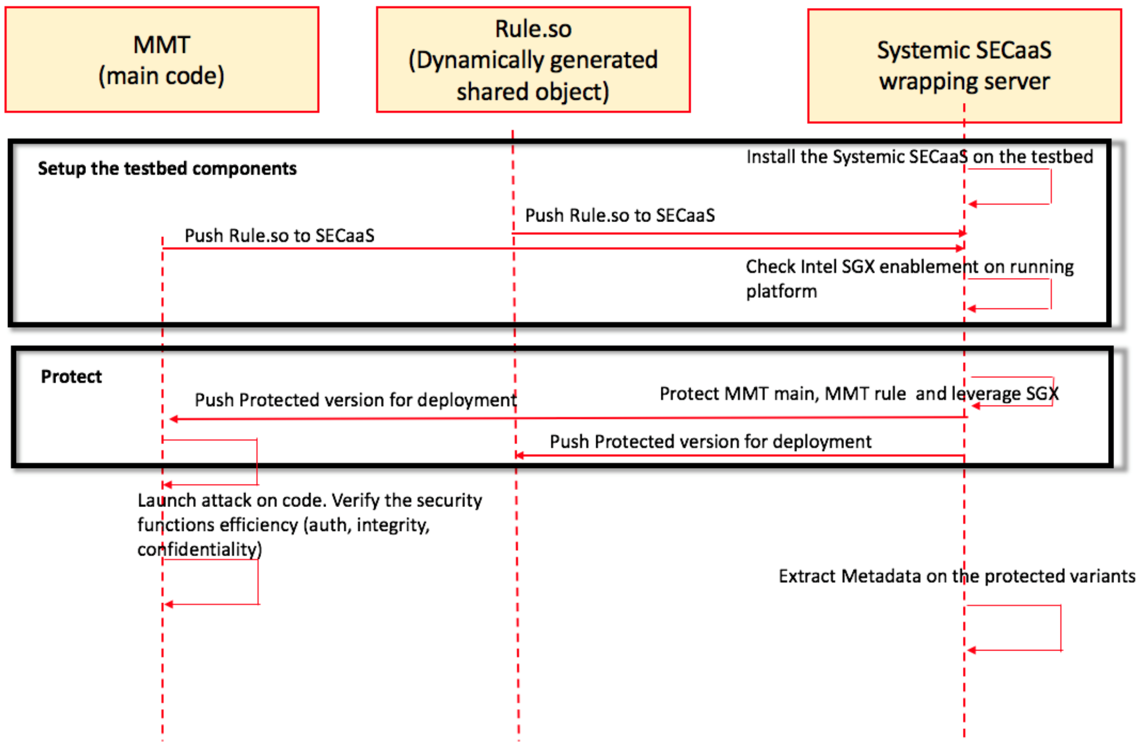

MMT (main code)
Rule.so (Dynamically generated shared object)
Systemic SECaaS wrapping server
Setup the testbed components
Install the Systemic SECaaS on the testbed
Push Rule.so to SECaaS
Push Rule.so to SECaaS
Check Intel SGX enablement on running platform
Protect
Protect MMT main, MMT rule and leverage SGX
Push Protected version for deployment
Push Protected version for deployment
Launch attack on code. Verify the security functions efficiency (auth, integrity, confidentiality)
Extract Metadata on the protected variants

Figure 21: SMF-Systemic test workflow

Test Case Name	TC3-4_SMF-STA
Test Purpose	Validate the integration between the Security Monitoring Framework (SMF) and the Smart Traffic Analyzer (STA)
Description	The test will be used to validate that the data events generated by STA is collected by SMF
Scenario	Presented in Figure 20

Test Case Name		TC3-4_SMF-STA	
Test sequence	Steps	Description	Result
	1	Setting Up Environment	Prepare environment information to be used during the test.
	2	Start the Security Monitoring Framework (SMF)	The Security Monitoring Framework is active and collect data from the environment.
	3	STA generate security events and send to SMF	STA VM/docker runs locally some traffic captures and generate security events in JSON format
	2	SMF receives a request to store the event	Request is accepted.
	3	SMF process and store the information	Event is stored in the STM
	4	SMF visualize the alert	Information contained in the JSON (network flow is shown)
Test verdict		If no error appears and SMF is able to show the information, the test will be considered as successful	

Table 10: SMF-STA test sequence

Test Case Name		TC3-4_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		Presented in Figure 20	
Test flow		Presented in Figure 22	
Test sequence	Steps	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	IPSec enforcement

Table 11: STA-SO test sequence

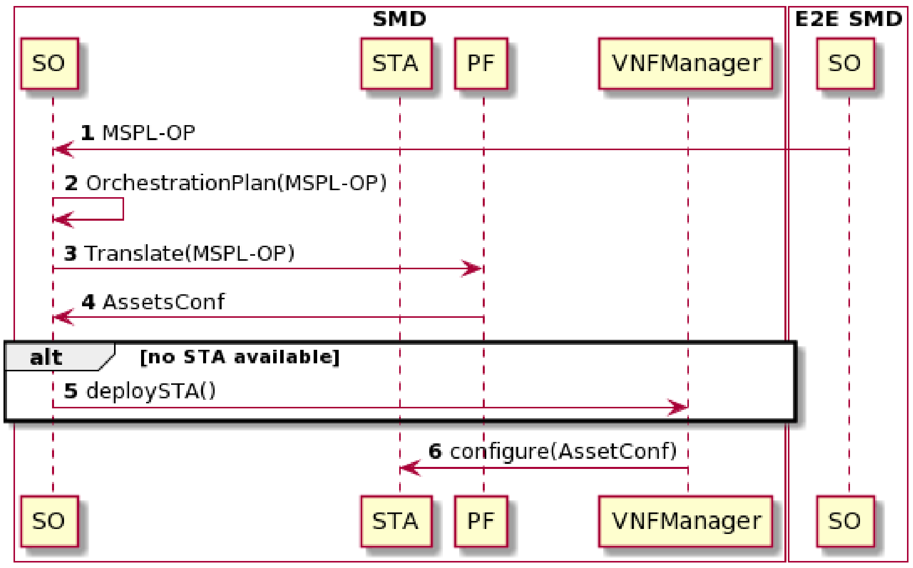


Figure 22: STA-SO test sequence

Test Case Name		**TC3-4_PoT-SO**	
Test Purpose		Validate the integration between Security orchestrator (UMU) and PoT (TID)	
Description		The test will be used to validate the activation of the PoT validation.	
Scenario		Presented in Figure 20	
Test flow		Presented in Figure 23	
Test sequence	Steps	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

Table 12: PoT-SO test sequence

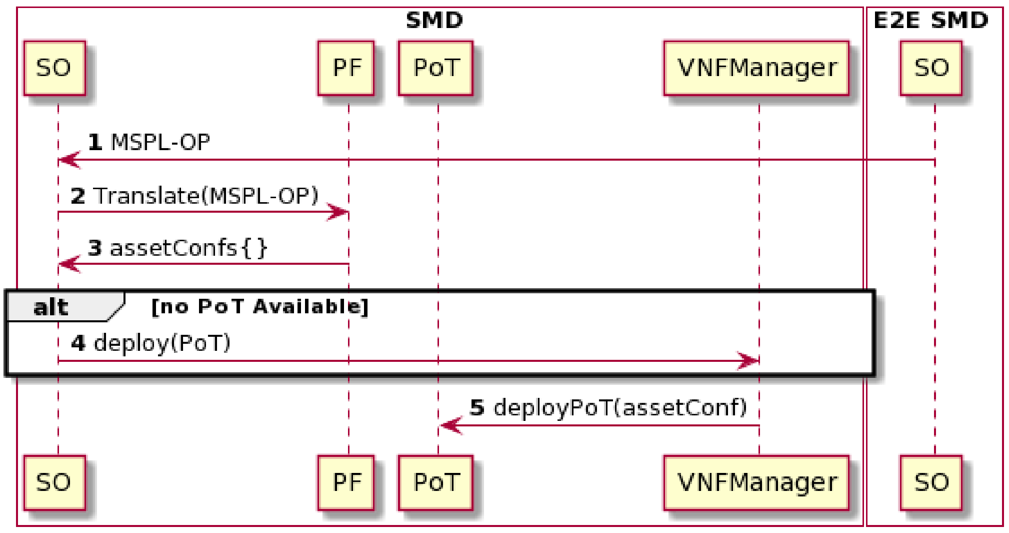


Figure 23: PoT-SO test sequence

Test Case Name		TC3-4_SMF-SO	
Test Purpose		Validate the integration between Security orchestrator (UMU) and SMF (MI)	
Description		The test will be used to validate the activation of the MMT Probe in SMF	
Scenario		Presented in Figure 20	
Test flow		Presented in Figure 24	
Test sequence	Steps	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	Final asset configurations

		Framework for MMT Probe enabler	
	5	MMT Probe Interacts with Systemic to perform protection rules	MMT protected rules
Test verdict		Security Orchestrator requests conf enforcement to MMT Probe enabler	MMT-Probe protected rules are in place

Table 13: SMF-SO test sequence

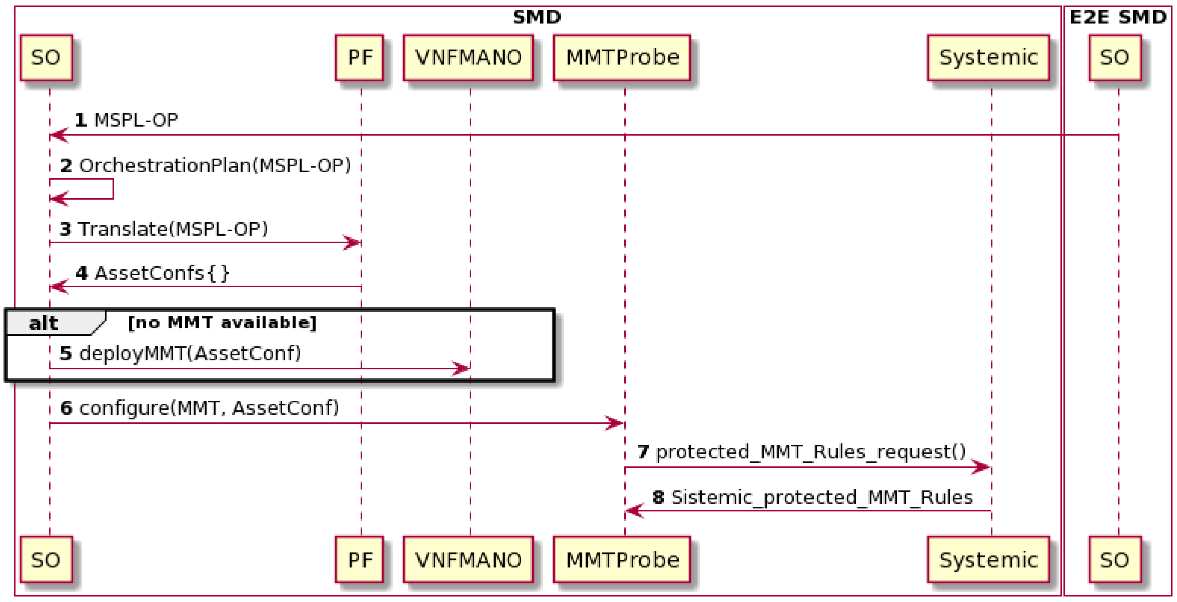


Figure 24: SMF-SO test sequence

Test Case Name		TC3-4-PolFram-SecOrch	
Test Purpose		Validate the integration between the Policy Framework (UMU PF) and the Security Orchestrator (UMU SO) enablers defined in WP3.	
Description		The test will be used to validate the policy operations required to perform policy-based orchestration operations.	
Scenario		Presented in Figure 20	
Test flow		Presented in Figure 25	
Test sequence	Steps	Description	Result
	1	Setting Up Environment	Prepare environment with the Security Orchestrator and the policy framework
	2	E2E Security Orchestrator receives a HSPL-OP policy	HSPL-OP

Test Case Name		TC3-4-PolFram-SecOrch	
	3	E2E Security Orchestrator identifies the involved management domains	List of management domains
	4	E2E Security Orchestrator prepares an orchestration plan according to the orchestration policies	E2E Orchestration plan
	5	E2E Security Orchestrator requests for HSPL-OP refinement to Policy Framework	MSPL-OPs
	6	E2E Security Orchestrator requests MSPL-OP enforcement to each involved domain	management domain enforcements
	5	Security Orchestrator receives MSPL-OP	MSPL-OP
	6	Security Orchestrator prepares an orchestration plan according to the orchestration policies	Orchestration plan
	7	Security Orchestrator requests MSPL-OP translation to Policy Framework	Final asset configurations
Test verdict	If no error appears during the different steps, the test will be considered as successful.		

Table 14: PolFram-SecOrch test sequence

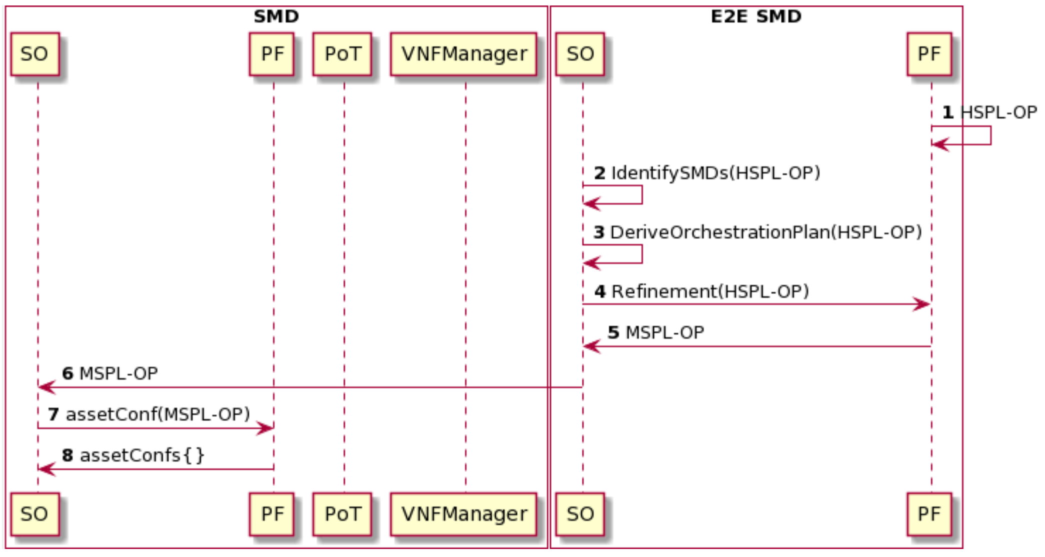


Figure 25: PolFram-SecOrch test sequence

Test Case Name		**TC3-4_I2NSF-SO**	
Test Purpose		Validate the integration between the Security Orchestrator (UMU SO) and the I2NSF IPSEC (TID-UMU) enablers defined in WP3	
Description		The test will be used to validate the channel protection policy enforcement over the I2NSF IPSEC enabler	
Scenario		Presented in Figure 20	
Test flow		Presented in Figure 26	
Test sequence	Steps	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 I2NSF IPSEC enabler	Final asset configurations
	5	Security Orchestrator requests conf enforcement to I2NSF IPSEC enabler	IPSec enforcement
Test verdict		If no error appears during the different steps, the test will be considered as successful.	

Table 15: I2NSF-SO test sequence

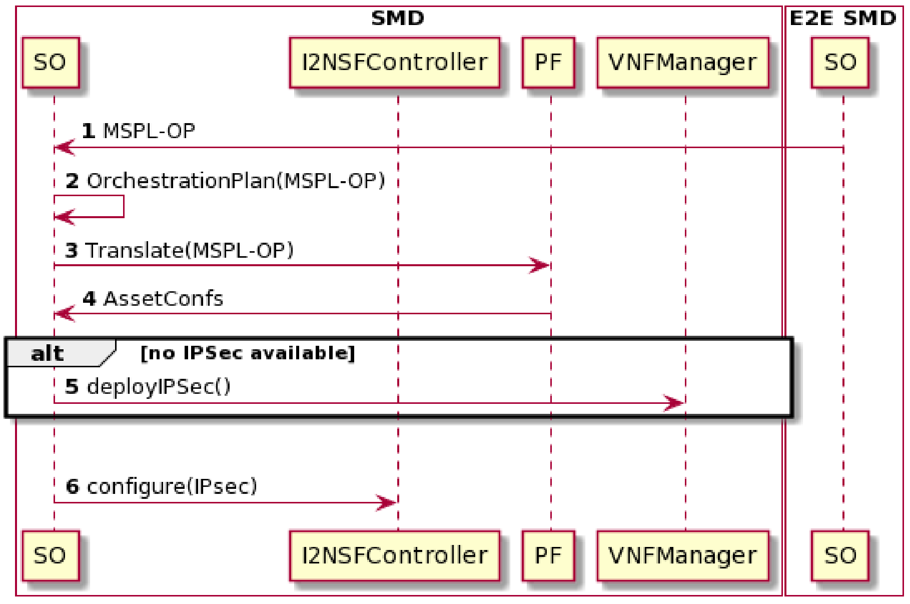


Figure 26: I2NSF-SO test sequence

Test Case Name		TC3-4_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		Presented in Figure 20	
Test flow		Presented in Figure 27	
Test sequence	Steps	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 stores the metrics inside DLT	Metrics are stored inside DLT
Test verdict		If valid metrics are stored in DLT, the test will be considered as successful.	

Table 16: POT-TRM test sequence

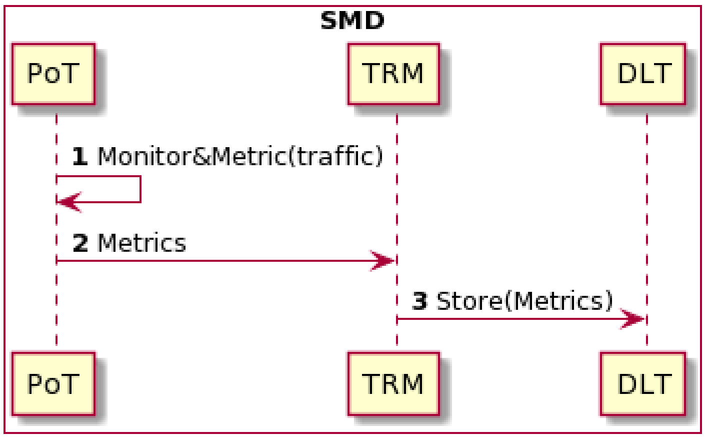


Figure 27: PoT-TRM test sequence

3.2.4 Test Case 5

TC5 defines a scenario showcasing the security of network slices by means of the enablers as summarized in Table 17.

WP3/WP4 Enablers	Owner
Moving Target Defense Controller (MOTDEC)	ZHAW
Optimizer of Security functions (OptSFC)	ZHAW
Katana Slice Manager	NCSRD
Anomaly Detection Framework (ADF)	NCSRD
Montimage Monitoring Framework (MMT)	MI

Table 17: WP3/WP4 enablers and partners developing each enabler for the functional verification of TC5

MOTDEC operates with the Katana Network Slice Manager and is automated using an ML pre-trained enabler, the Optimizer of Security Functions (OptSFC), which ingests the monitoring and anomaly detection data from MMT and ADF.

3.2.4.1 Scenario and workflow

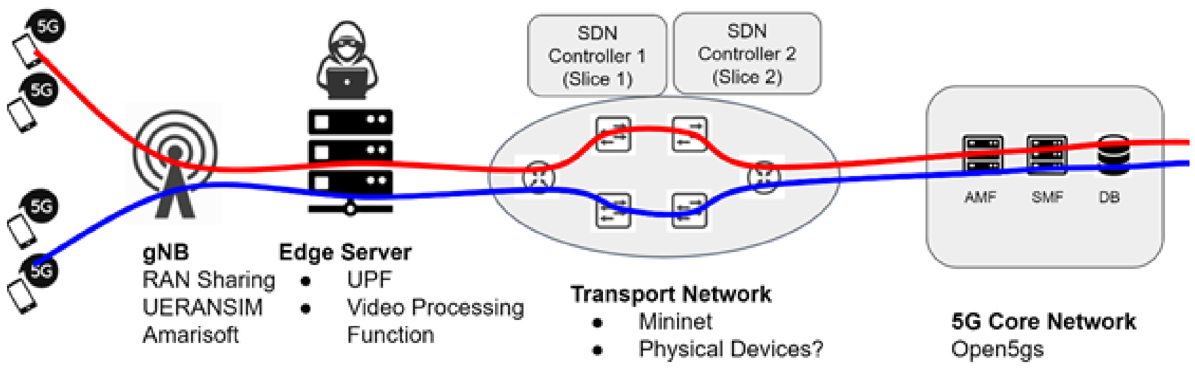


Figure 28: TC5 scenario

The presented scenario in Figure 28 displays a 5G infrastructure using edge computing and hosting two different network slices, a public one (i.e., owned by the network operator) and a private one (e.g., allocated for an enterprise intra-network or service). Simulated attacks will be targeting the network slice components (VNFs or NSs) at the edge server. The INSPIRE-5Gplus enablers will be providing the required mitigation and proactive security using two test sequences, each test sequence presenting the interaction between one enabler and the other.

3.2.4.2 Definition of the test sequence

Test Case Name		TC5_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 Manager that should be properly accepted and instantiated.	
Scenario		Presented in Figure 28	
Test flow		Presented in Figure 29-30	
Test sequence	Steps	Description	Result
	1	Setting Up Environment	Define the testing environment.
	2	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
	3	MOTDEC sends a request to the Slice Manager to get VIM status	The Slice Manager returns the VIM status for that instant *t*: current usage of CPU, RAM, disk and bandwidth
	4	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.
	5	MOTDEC checks the status of the slice.	The Slice Manager returns the slice status for that instant *t*: is it running, is it operational, what are its IPv4 and IPv6 addresses, in which VIM is it deployed and bandwidth used
	6	MOTDEC sends a request to modify the network slice.	The request is accepted by the Slice Manager and properly modifies the network slice
Test verdict		If there are no errors, the test is successful.	

Table 18: Test sequence for MotDec-SliceM

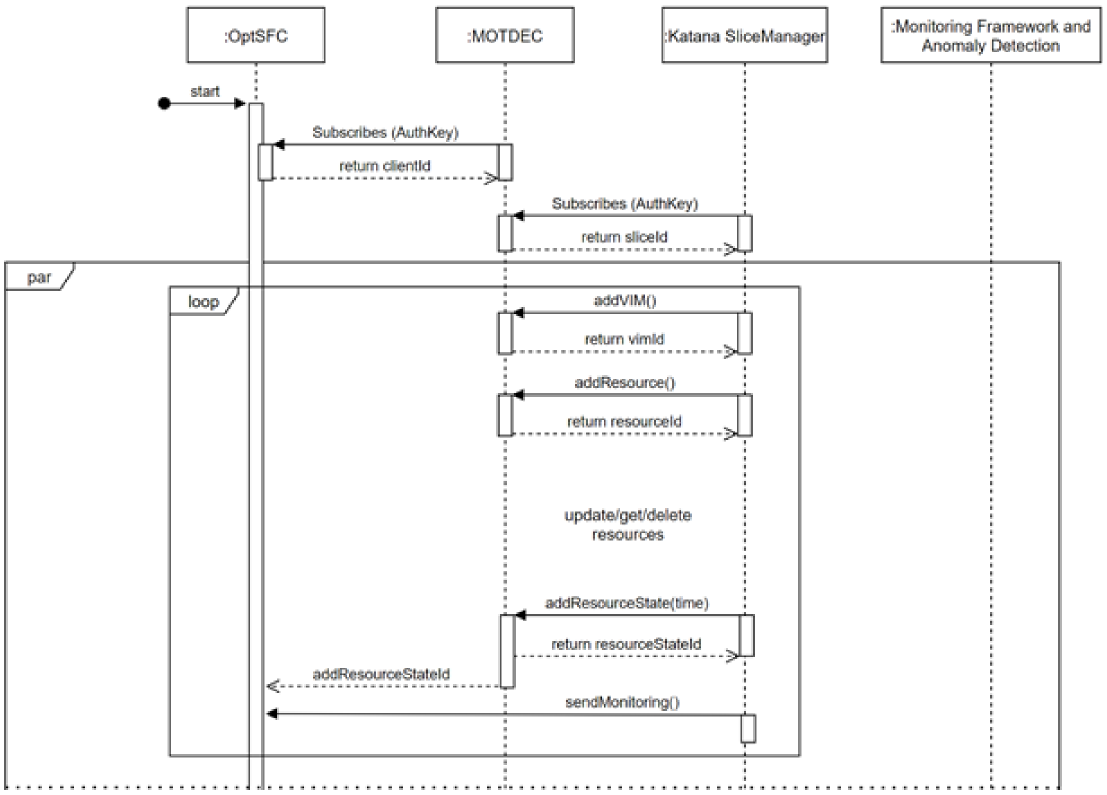


Figure 29: MOTDEC-SliceM test flow - part 1

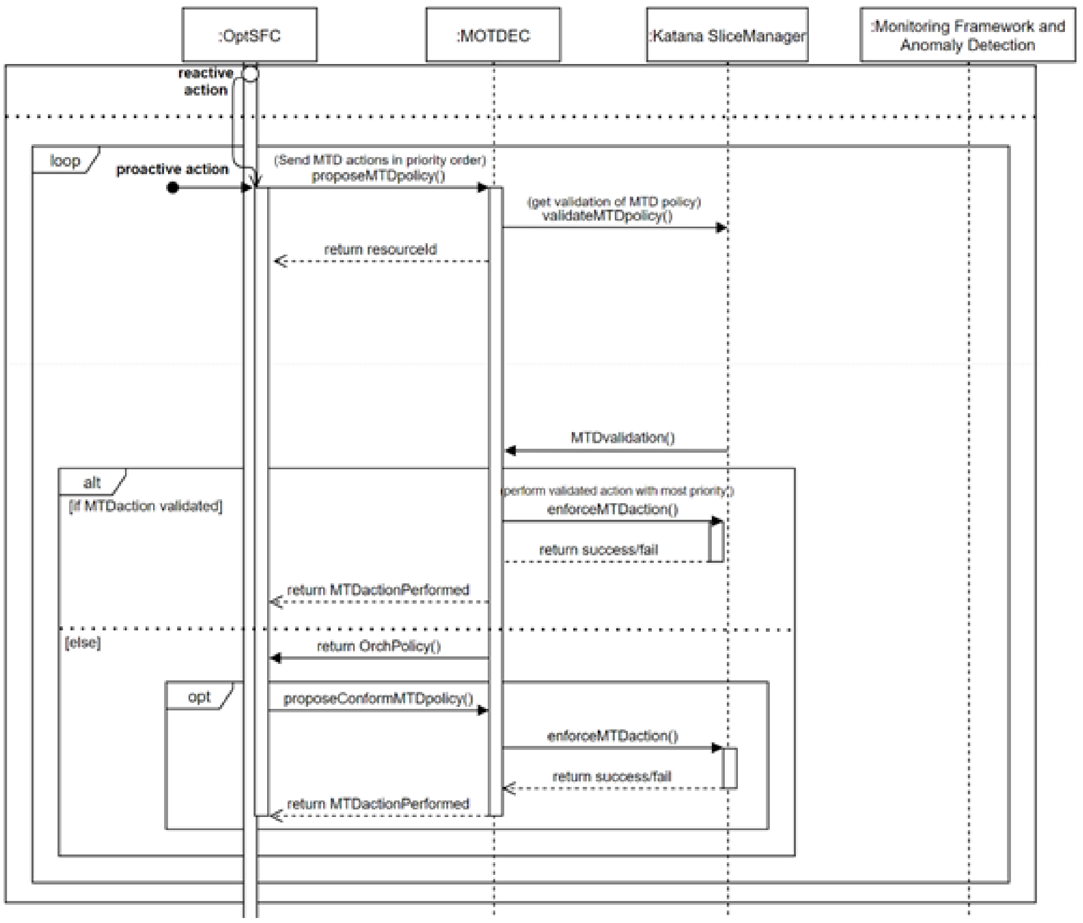


Figure 30: MOTDEC-SliceM test flow - part 2

Test Case Name		TC5_OptSFC-MMTF/ADF	
Test Purpose		Validate the interaction between OptSFC (ZHAW) and MMT (MI) and ADF (NCSRD).	
Description		OptSFC receives monitoring data and anomaly detection alerts from MMT and ADF.	
Scenario		Presented in Figure 28	
Test flow		Presented in Figure 31	
Test sequence	Steps	Description	Result
	1	Setting Up Environment	Prepare the testing environment.
	2	OptSFC indicates to MMT and ADF which networks to monitor	MMT activates probes of the networks required by OptSFC
	3	MMT and ADF send anomaly and attack detection to OptSFC	MMT sends the targeted IP address, the "attacker"'s IP address, and the type of attack: anomaly detection or DoS (sent via Kafka?)
	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, the test is successful.	

Table 19: Test sequence for OptSFC-MMT/ADF

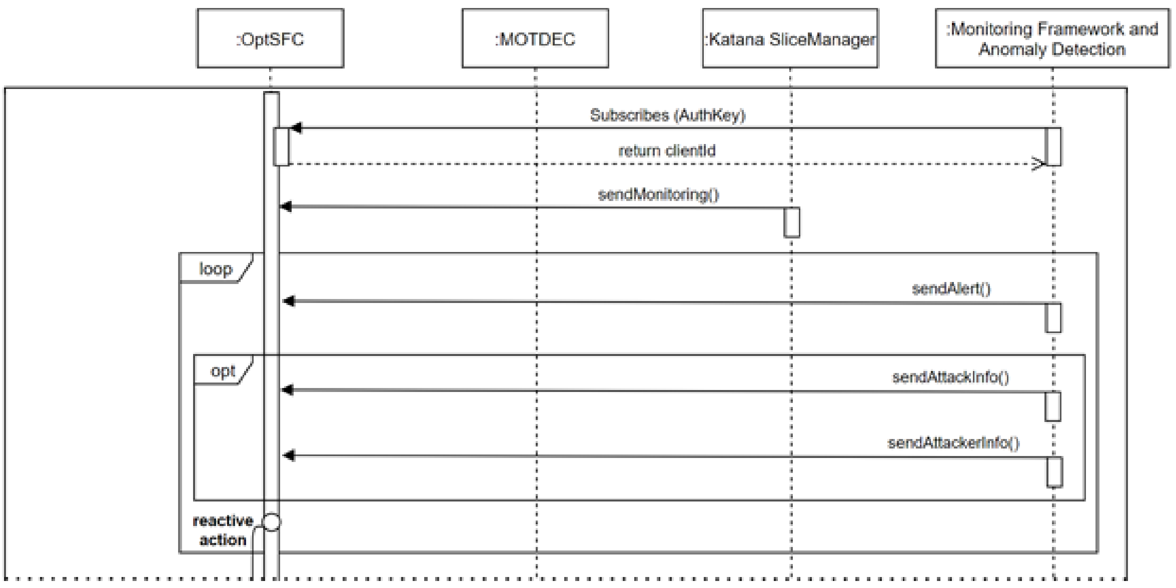


Figure 31: OptSFC-MMT/ADF test flow

3.2.5 Test Case 6

Table 20 summarizes the list of WP3/WP4 enablers used for the functional verification of TC6.

WP3/WP4 Enablers	Owner
Security Orchestrator	UMU
Policy Framework	UMU
Trust Reputation Manager	UMU
Security Analytic Engine	MI
Virtual Channel Protection (deployed as DTLS proxy in context of TC6)	TSG

Table 20: WP3/WP4 enablers and partners developing each enabler for the functional verification of TC6

3.2.5.1 Scenario and workflow

In the scenario considered for TC6, the vehicle triggers the handover procedure when crossing inter-domain borders (0), this handover is performed by 5G system (1) and it is detected by monitoring asset that informs to the Security Analytics Engine (2) and this to the E2E Security Analytics Engine (3) which then starts the migration procedure for vOBU and DTLS, as a reaction to this change in location (4) the Security Orchestrator then creates the orchestration plan (5, 6.a, 6.b). The migration procedure is aided by the OBU Manager asset (12), which retrieves the required information to deploy the same vOBU on required domain (11). Before the migration occurs, the traffic is redirected to the old vOBU (10) via SDN switching (9), and the migration procedure retires when the vOBU is deployed (10) and the traffic is redirected to the new vOBU (13). Within this process, in parallel, the VCP (Virtual Channel Protection enabler)'s DTLS proxy deployed on the edge is also migrated in an efficient and secure way, avoiding compute-intensive generation of new asymmetric security keys for cryptographic algorithms.

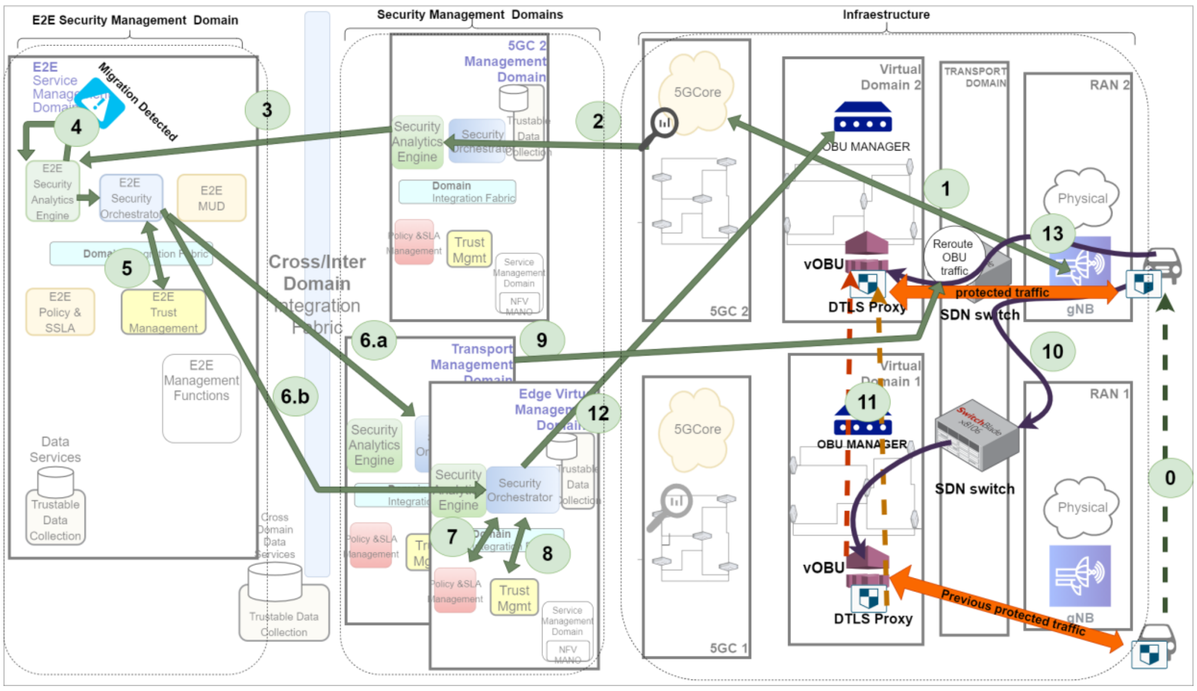

E2E Security Management Domain
Security Management Domains
Infraestructure
E2E Service Management Domain
Migration Detected
E2E Security Analytics Engine
E2E Security Orchestrator
E2E MUD
E2E Policy & SSLA
E2E Trust Management
E2E Management Functions
Data Services
Trustable Data Collection
Cross/Inter Domain Integration Fabric
Cross Domain Data Services
5GC 2 Management Domain
Security Analytics Engine
Security Orchestrator
Trustable Data Collection
Domain Integration Fabric
Policy &SLA Management
Trust Mgmt
Service Management Domain
NFV MANO
Transport Management Domain
Edge Virtual Management Domain
5GCore
5GC 2
5GC 1
Virtual Domain 2
Virtual Domain 1
OBU MANAGER
vOBU
DTLS Proxy
TRANSPORT DOMAIN
Reroute OBU traffic
protected traffic
Previous protected traffic
SDN switch
RAN 2
RAN 1
Physical
gNB
0
1
2
3
4
5
6.a
6.b
7
8
9
10
11
12
13

Figure 32: Scenario for TC6

3.2.5.2 Definition of the test sequence

Test Case Name		TC6_SMF_TRM	
Test Purpose		Validate the integration between the Security Monitoring Framework (SMF) and the Trust Reputation Manager (TRM)	
Description		The test will be used to validate that the data obtained by SMF goes to TRM	
Scenario		Presented in Figure 32	
Test flow		Presented in Figure 33	
Test sequence	Steps	Description	Result
	1	Setting Up Environment	Prepare environment information to be used during the test.
	2	TRM receives a request of trust calculation	Request is accepted.
	3	TRM asks SMF about last events concerning the entity which trust need to calculate	Petition to SMF
	4	SMF informs about the values	TRM receives the values
	5	TRM calculates trust using that input	Trust value
Test verdict		If no error appears and TRM is able to calculate a trust value, the test will be considered as successful	

Table 21: Test sequence for SMF_TRM

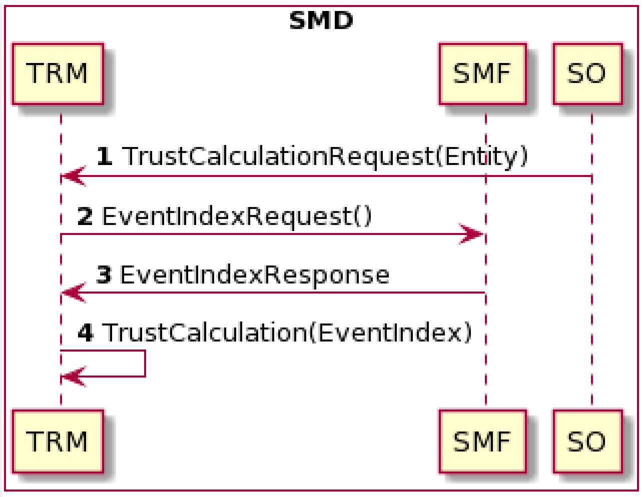

SMD
TRM
SMF
SO
1 TrustCalculationRequest(Entity)
2 EventIndexRequest()
3 EventIndexResponse
4 TrustCalculation(EventIndex)
TRM
SMF
SO

Figure 33: Test flow for SMF_TRM

Test Case Name		TC6-PolFram-SecOrch	
Test Purpose		Validate the integration between the Policy Framework (UMU PF) and the Security Orchestrator (UMU SO) enablers defined in WP3.	
Description		The test will be used to validate the policy operations required to perform policy-based orchestration operations.	
Scenario		Presented in Figure 32	
Test flow		Presented in Figure 34	
Test sequence	Steps	Description	Result
	1	Setting Up Environment	Prepare environment with the Security Orchestrator and the policy framework
	2	E2E Security Orchestrator receives a HSPL-OP policy	HSPL-OP
	3	E2E Security Orchestrator identifies the involved management domains	List of management domains
	4	E2E Security Orchestrator prepares an orchestration plan according to the orchestration policies	E2E Orchestration plan
	5	E2E Security Orchestrator requests for HSPL-OP refinement to Policy Framework	MSPL-OPs
	6	E2E Security Orchestrator requests MSPL-OP enforcement to each involved domain	Management domain enforcements
	5	Security Orchestrator receives MSPL-OP	MSPL-OP
	6	Security Orchestrator prepares an orchestration plan according to the orchestration policies	Orchestration plan
	7	Security Orchestrator requests MSPL-OP translation to Policy Framework	Final asset configurations
Test verdict		If no error appears during the different steps, the test will be considered as successful.	

Table 22: Test sequence for PolFram-SecOrch

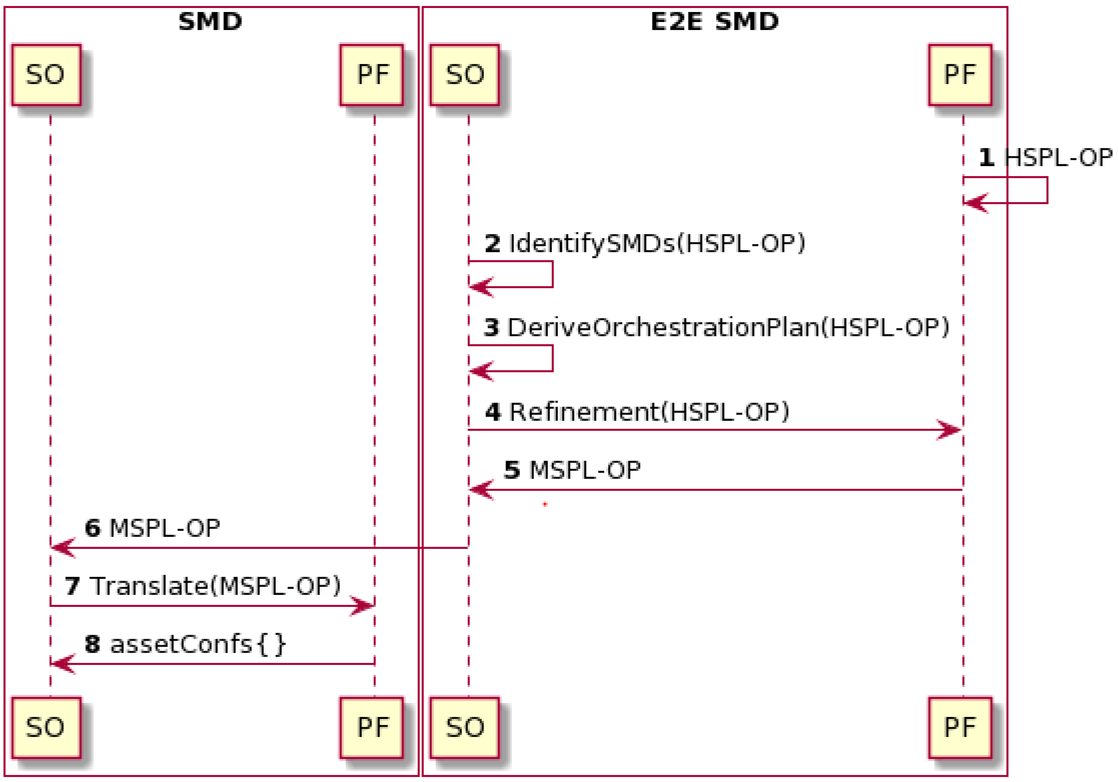


Figure 34: Test flow for PolFram-SecOrch

3.2.6 Test Case 7

Table 23 summarizes the list of WP3/WP4 enablers used for the functional verification of TC7.

WP3/WP4 Enablers	Owner
Security Monitoring Framework (MMT Probe)	MI
Auto-Scaling Module (Admission Controller Delegator)	TSG
Damage Controller (DDoS Mitigator)	AALTO
Decision Engine (Optional)	TSG

Table 23: WP3/WP4 enablers and partners developing each enabler for the functional verification of TC7

The main objective of TC7 is to leverage ML to detect and prevent malicious auto-scaling operations due to workload caused by a DDoS attack. Figure 35 illustrates a potential attack scenario where uncontrolled scaling up/scaling out of Slice A's resources may lead to exhausting physical resources shared with Slice B.

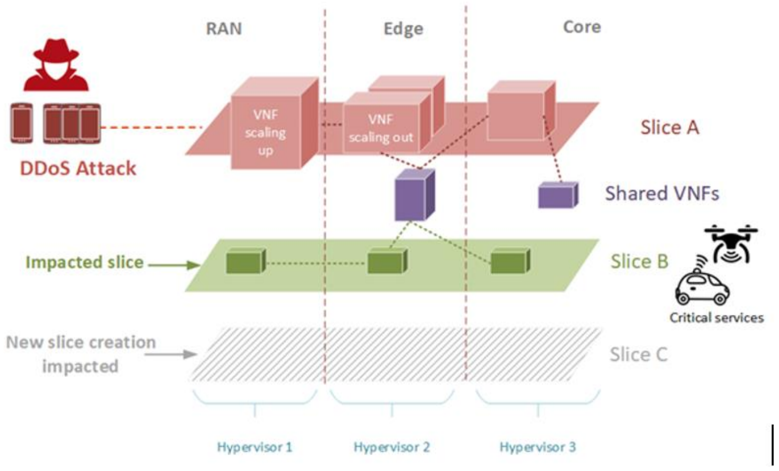


Figure 35: TC7 DDoS attack against shared resources scenario

3.2.6.1 Scenario and workflow

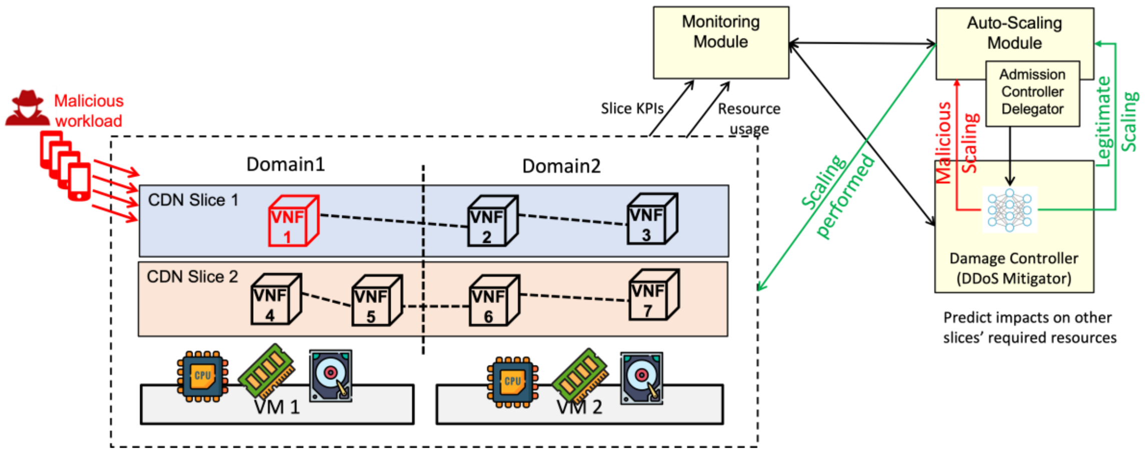


Figure 36: TC7 test architecture and relationship between involved enablers

Figure 36 depicts the architecture for the test scenario for TC7, showing the interactions between the different WP3 enablers involved in TC7, namely: (i) the ***Monitoring Framework***, which is in charge of collecting resource usage and performance metrics from the slices via deployed probes; (ii) the ***Admission Controller Delegator***, which is responsible for intercepting the auto-scaling request triggered by the auto-scaling module and delegate the scaling decision to the Damage Controller for validation; and (iii) the ***Damage Controller***, which runs a ***DDoS Mitigator*** model that uses ML to detect whether the scaling is due to legitimate workload or rather malicious workload caused by an application-layer DDoS attack. If the workload is malicious, the scaling operation is refused, and the Decision Engine can (optionally) be informed to take further measures to mitigate the attack. This could include analysing network traffic to identify the origin of the attack and block the attackers, and/or retraining the intrusion detection system on the new malicious network traffic to improve its capabilities in detecting the attack in the future. Figure 37 illustrates the proposed workflow of scenario considered in TC7. It is worth mentioning that the ***Decision Engine*** is not planned to be demonstrated in TC7.

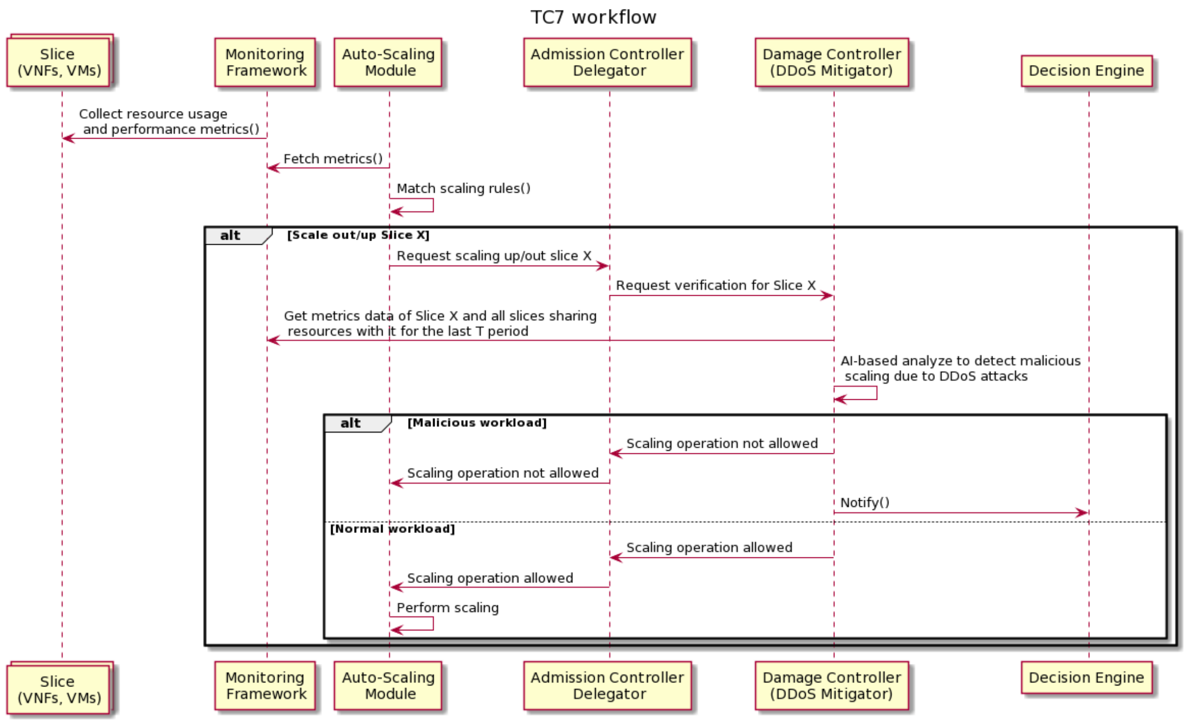

TC7 workflow
Slice (VNFs, VMs)
Monitoring Framework
Auto-Scaling Module
Admission Controller Delegator
Damage Controller (DDoS Mitigator)
Decision Engine
Collect resource usage and performance metrics()
Fetch metrics()
Match scaling rules()
alt
[Scale out/up Slice X]
Request scaling up/out slice X
Request verification for Slice X
Get metrics data of Slice X and all slices sharing resources with it for the last T period
AI-based analyze to detect malicious scaling due to DDoS attacks
alt
[Malicious workload]
Scaling operation not allowed
Scaling operation not allowed
Notify()
[Normal workload]
Scaling operation allowed
Scaling operation allowed
Perform scaling

Figure 37: TC7 workflow

3.2.6.2 Definition of the test sequence

TC7 defines two test sequences to validate the integration between the involved WP3 enablers. The first test sequence validates the integration between the Security Monitoring Framework (MI's MMT) and the Damage Controller (DDoS Mitigator) enablers. The first test sequence is described in Table 24 and its flow is illustrated in Figure 38. The second test sequence validates the integration between the Admission Controller Delegator and the Damage Controller (DDoS Mitigator) enablers. The second test sequence is described in Table 25 and its flow is depicted in Figure 39.

Test Case Name		TC7_SMF-DDoSMitigator	
Test Purpose		Validate the integration between the Security Monitoring Framework (MI's MMT) and the DDoS Detection & Mitigation in Network Slicing (DDoS Mitigator AALTO) enablers defined in WP3.	
Description		The test will be used to validate to get data on resource usage and performance from a CDN slice using the Security Monitoring Framework and how the DDoS Mitigator can decide if the autoscaling request is caused by a normal workload or due to a DDoS attack.	
Scenario		Presented in Figure 36	
Test flow		Presented in Figure 38	
Test sequence	Steps	Description	Result
	1	Setting Up Environment	Prepare environment with 2 CDN slices using a set of shared resources (VM) and deploy the probes.

Test Case Name		TC7_SMF-DDoSMitigator	
	2	Start the Security Monitoring Framework	The Security Monitoring Framework is active and collect data from the environment.
	3	The DDoS Mitigator requests monitoring data from the Security Monitoring Framework	The DDoS Mitigator can read data from the Security Monitoring Framework
	4	The DDoS Mitigator test if an autoscaling is allowed or not	Depending on the data Collected from the Security Monitoring Framework the DDoS Mitigator can decide to allow the auto scaling or not
Test verdict		If no error appears during the different steps, the test will be considered as successful.	

Table 24: Integration test sequence between Monitoring Framework and Damage Controller (DDoS Mitigator) in TC7

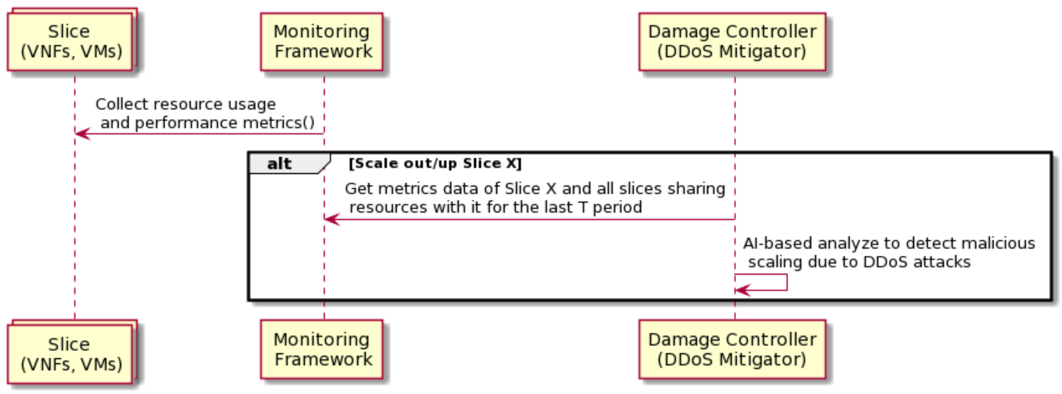

Slice (VNFs, VMs)
Monitoring Framework
Damage Controller (DDoS Mitigator)
Collect resource usage and performance metrics()
alt
[Scale out/up Slice X]
Get metrics data of Slice X and all slices sharing resources with it for the last T period
AI-based analyze to detect malicious scaling due to DDoS attacks
Slice (VNFs, VMs)
Monitoring Framework
Damage Controller (DDoS Mitigator)

Figure 38: Test flow for integration between Monitoring Framework and Damage Controller (DDoS Mitigator) enablers in TC7

Test Case Name		TC7_ACD-DDoSMitigator	
Test Purpose		Validate the integration between the Admission Controller Delegator (TSG's ACD) and the DDoS Detection & Mitigation in Network Slicing (AALTO's DDoS Mitigator) enablers defined in WP3.	
Description		The test will be used to validate that the auto-scaling request can be intercepted by the ACD and redirected to the DDoS Mitigator to decide if the autoscaling request is caused by a normal workload or due to a DDoS attack.	
Scenario		Presented in Figure 36	
Test flow		Presented in Figure 39	
Test sequence	Steps	Description	Result

Test Case Name		TC7_ACD-DDoSMitigator	
	1	Setting Up Environment and starting the Security Monitoring Framework	See results of steps 1 and 2 of the previous test (i.e., TC7_SMF-DDoSMitigator)
	2	Auto-Scaling Module gather resource usage and performance metrics of a slice's VNFs	The Auto-Scaling Module can read data from the Security Monitoring Framework
	3	Auto-Scaling Module trigger a scale up operation	The Auto-Scaling Module requests the system to create new instances of a VNF to handle the increase in workload
	4	The ACD intercepts the scaling request	The scaling request triggered by the Auto-Scaling Module is intercepted by ACD and redirected to the DDoS Mitigator for verification
	5	The DDoS Mitigator tests if the autoscaling is allowed or not (At this stage, the previous integration test is applied).	See results of steps 3 and 4 of the previous test (i.e., TC7_SMF-DDoSMitigator) The decision (i.e., scaling event legitimate or malicious) is forwarded to the ACD.
	6	The ACD forwards back the auto-scaling decision made by DDoS Mitigator decision	Depending on the decision taken by the DDoS Mitigator, the scaling request will be allowed (if legitimate) or ignored (if malicious)
Test verdict		If no error appears during the different steps, the test will be considered as successful.	

Table 25: Integration test sequence between Admission Controller Delegator (ACD) and Damage Controller (DDoS Mitigator) in TC7

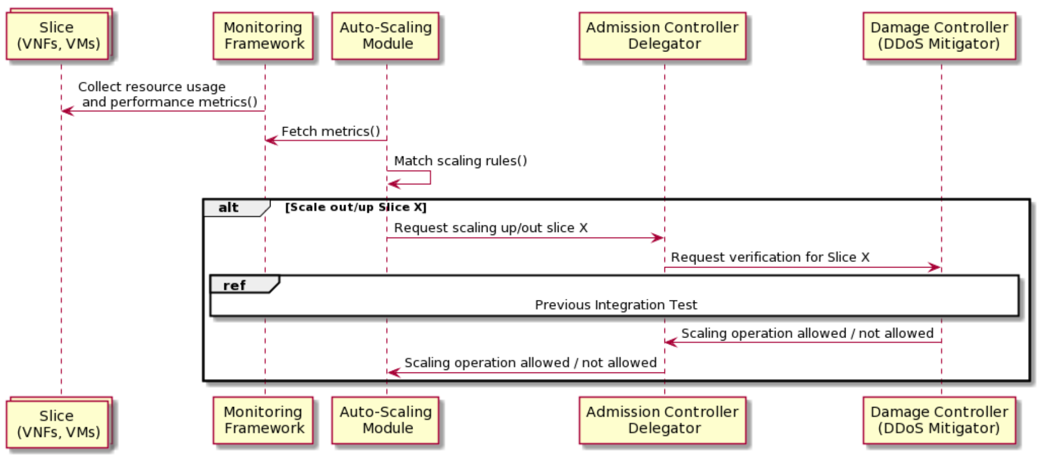

Slice (VNFs, VMs)
Monitoring Framework
Auto-Scaling Module
Admission Controller Delegator
Damage Controller (DDoS Mitigator)
Collect resource usage and performance metrics()
Fetch metrics()
Match scaling rules()
alt [Scale out/up Slice X]
Request scaling up/out slice X
Request verification for Slice X
ref
Previous Integration Test
Scaling operation allowed / not allowed
Scaling operation allowed / not allowed
Slice (VNFs, VMs)
Monitoring Framework
Auto-Scaling Module
Admission Controller Delegator
Damage Controller (DDoS Mitigator)

Figure 39: Test flow for integration between Admission Controller Delegator and Damage Controller (DDoS Mitigator) enablers in TC7

As we are planning to demonstrate TC7 in a Kubernetes environment, we describe in what follows the integration test sequence between the three enablers involved in the scenario of TC7 for this environment. The test sequence is described in Table 26 and its flow is illustrated in Figure 40.

Test Case Name		**TC7_SMF-ACD-DDoSMitigator (applied to Kubernetes)**	
Test Purpose		Validate the integration between the Security Monitoring Framework (MI's MMT) and the DDoS Detection & Mitigation in Network Slicing (AALTO's DDoS Mitigator) and the Kubernetes Admission Controller Delegator (TSG's KACD) enablers defined in WP3.	
Description		The test will be used to validate the collection of data (i.e., resource usage and performance metrics) about VNFs to be fed into an auto-scaling module inside a Kubernetes environment. The test will then validate that the auto-scaling can be intercepted by the KACD and redirected to the DDoS Mitigator.	
Scenario		Presented in Figure 36	
Test flow		Presented in Figure 40	
Test sequence	Steps	Description	Result
	1	Monitoring VNFs metrics	A metrics server surveys the VNFs' resources usage and performance metrics.
	2	Aggregating metrics	The metrics server aggregates the metrics before exposing them.
	3	Gathering resource usage and performance metrics	The Horizontal Pod Autoscaler (HPA) gathers the metrics related to the managed Pod.
	4	Trigger a scale up	The HPA decides to scale the desired Pod number to handle the increase in workload.
	5	Intercepting the scale up	The I5G+ KACD was configured to install webhooks inside the K8S api. These webhooks are called when a change in the manifest related to a Pod is submitted. Thus, the control flow is moved from Kubernetes to KACD.
	6	Delegating the scaling decision	KACD delegates the scaling decision to the Damage Controller (DDoS Mitigator) for validation. It passes the Pod json.
	7	Computing the mitigation	The DDoS Mitigator employs AI-based algorithms to classify the scaling event.

Test Case Name		TC7_SMF-ACD-DDoSMitigator (applied to Kubernetes)	
	8	Refusing the scaling up	The DDoS Mitigator returns a false response to specify that the scaling event is not legitimate.
	9	Notifying the situation	The DDoS Mitigator can optionally forward its action to alert the Decision Engine.
	10	Refusing the scale up	KACD returns a 'not allowed' response to the Kubernetes API, which will ignore the initial resources modification. The Pod are left unchanged.
	11	Validating the scaling up	The DDoS Mitigator returns a True response signalling that the scaling event is validated.
	12	Accepting the scale up	KACD return an "allowed" response to the Kubernetes API, which will mutate the various manifests inside its DB.
	13	Updating the number of replicas	Inside Kubernetes, the ReplicaSet controller will see the manifests change and modify the number to pod inside a service.
	14	A new Pod is created	The Kubelet service will apply the new manifest and spawn a new pod accordingly.
Test verdict		If the Pod creation is prevented in the case of attack, then the test is considered successful.	

Table 26: Integration test sequence between the three enablers involved in TC7

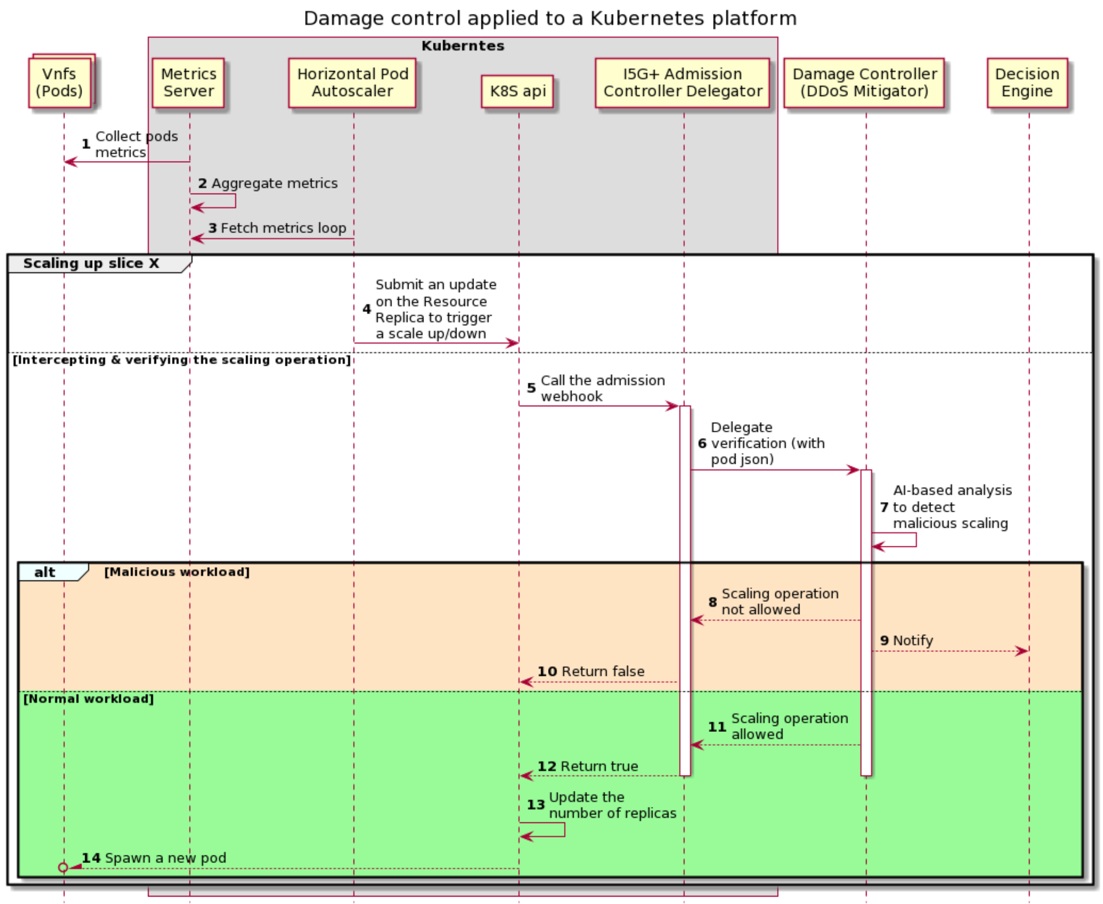

Damage control applied to a Kubernetes platform
Kuberntes
Vnfs (Pods)
Metrics Server
Horizontal Pod Autoscaler
K8S api
I5G+ Admission Controller Delegator
Damage Controller (DDoS Mitigator)
Decision Engine
1 Collect pods metrics
2 Aggregate metrics
3 Fetch metrics loop
Scaling up slice X
4 Submit an update on the Resource Replica to trigger a scale up/down
[Intercepting & verifying the scaling operation]
5 Call the admission webhook
6 Delegate verification (with pod json)
7 AI-based analysis to detect malicious scaling
alt [Malicious workload]
8 Scaling operation not allowed
9 Notify
10 Return false
[Normal workload]
11 Scaling operation allowed
12 Return true
13 Update the number of replicas
14 Spawn a new pod

Figure 40: Test flow for integration between the three enablers involved in TC7 in a Kubernetes environment

3.2.7 Test Case 8

Table 27 summarizes the list of WP3/WP4 enablers used for the functional verification of TC8.

WP3/WP4 Enablers	Owner
DiscØvery	CLS
Security Analytics Framework	NCSRD

Table 27: WP3/WP4 enablers and partners developing each enabler for the functional verification of TC8

3.2.7.1 Scenario and workflow

The testbed can be configured to generate 5G network traffic based on the use cases and architecture of 5G-CARMEN. 5G-CARMEN makes use of OpenMano software that is deployed on the MEC infrastructure, and connected vehicles make use of Linux-based operating hardware components for connectivity and processing. The software and hardware components of the TC8 have been virtualized in TC8 to generate 5G network traffic based on different cybersecurity scenarios, such as denial of service attacks, and impersonations attacks, that would be difficult to test in real-life scenarios. Once the different network traffic datasets are generated, they will be used as an input to the Security Analytics Framework and the DiscØvery enabler to provide a security analysis report based on the security posture of the network. The security report will provide a list of suggestions to improve security and assist a security analyst with the decision support process.

3.2.7.2 Definition of the test sequence

The test sequence is the following:

- Initial Condition: Connected vehicles A, B, C and the Emergency Vehicle are moving on a highway.
  - Vehicles B, C are on the right lane at moderate speed (90-100km/h) with some distance between them (e.g., 100m)
  - Vehicle A approached on the left lane (10 -20 seconds away) moving a bit faster (110 - 130 km/h, eventually overtake)
  - Emergency Vehicle is about 20 - 30 seconds away from Vehicle A at 130 km/h
- Event: Emergency Vehicle turns its emergency state on (electronically); DENM notification are sent periodically
  - This triggers an emergency vehicle warning with the Estimated Time of Arrival (ETA)
- Reaction: The overtaking lane needs to be cleared by the cooperative vehicles, therefore
  - Vehicle A needs to shift lane and the slowdown to a moderate speed
  - Depending on the ETA and speed differences:
    - ETA much bigger than overtaking time: Vehicle A ends the overtake
    - ETA much smaller than overtaking time: Vehicle A shifts lane and queues behind Vehicles B, C
    - ETA in between: Vehicles B, C keep on the right lane, and do a cooperative lane merge with Vehicle A

Conclusion: Emergency Vehicle passes undisturbed on the cleared overtaking lane.

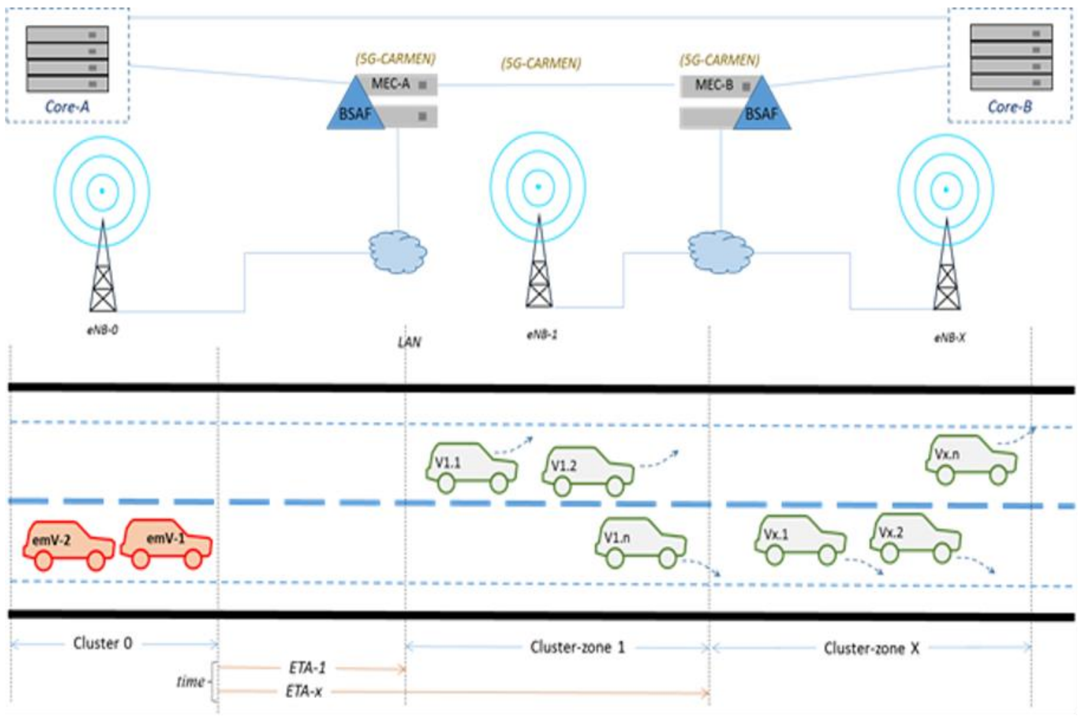


Figure 41: TC8 initial state scenario

Test Case Name		TC8_SecAn-CLS	
Test Purpose		Validate the integration between the Security Analytics Framework (NCSRD) and the DiscØvery enabler (CLS).	
Description		The test will evaluate if the connection for data transfer between the two enablers is active.	
Scenario		Presented in Figure 41	
Test flow		Presented in Figure 42	
Test sequence	Steps	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 DiscØvery and configure it to read the output of the Security Analytics Framework.	DiscØvery is active and reads data from the Security Analytics Framework.
	4	Use DiscØvery to generate a system model based on the network traffic outputted by the Security Analytics Framework.	A system model is generated.
Test verdict		If the network connection is live and no errors appears during the different steps defined, the test will be considered as successful.	

Table 28: Test sequence for SecAn-CLS

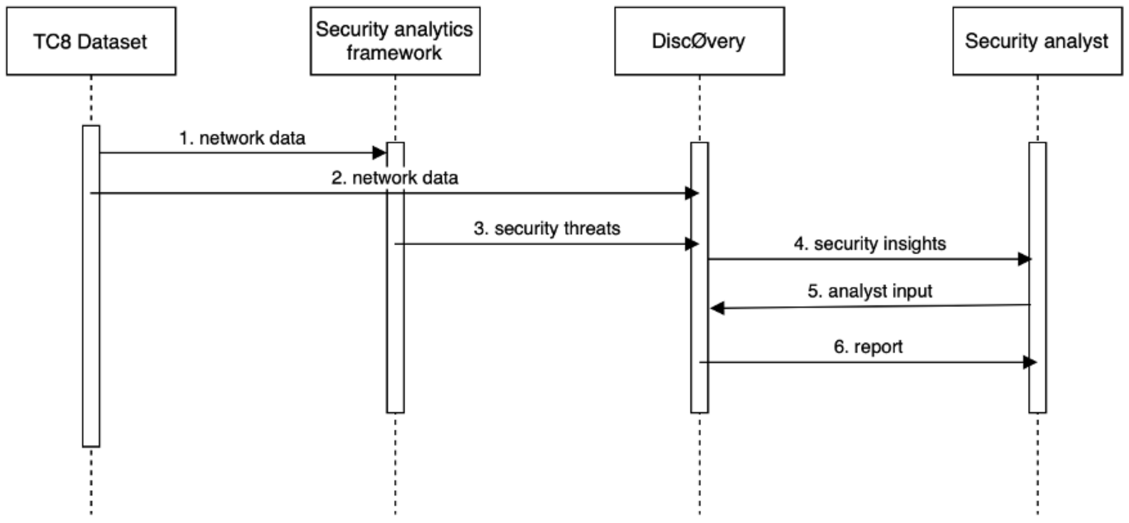


Figure 42: TC8 test workflow

3.2.8 Test Case 9

Table 29 summarizes the list of WP3/WP4 enablers used for the functional verification of TC9.

WP3/WP4 Enablers	Owner
SFSBroker: Secure and Federated Network Slice Broker	UOULU
Katana Network Slice Manager	NCSRD

Table 29: WP3/WP4 enablers and partners developing each enabler for the functional verification of TC9

3.2.8.1 Scenario and workflow

The deployment scenario of TC9 in the 5G architecture is presented in Figure 43 and summarized under this section. IoT nodes and Mobile Network Operators (MNOs) are the main stakeholders connected to the SFSBroker. Consumer ends (or IoT nodes) are allowed to send network slice requests to SFSBroker via fog nodes. In our scenario, both the legitimate and malicious requests are considered. Meanwhile, we are under the assumption that Katana slice manager, OpenMano and OpenStack softwares are installed in MNOs for the purpose of management of network slices, management and orchestration, and provision of infrastructure respectively. SSLA is deployed in another separate blockchain on top of SFSBroker to ensure end-to-end security of network slices. SFSBroker is built on the Hyperledger-Fabric blockchain platform. It comprises four main entities and their main responsibilities are listed below:

- Prime mover - a smart contract programmed to create the network slice blueprint based on customer requirements and then passes to the mediator.
- Resource unit price database - saves all the resources advertised by MNOs and their respective unit prices. MNOs are allowed to add or update the resource information whenever it is necessary.
- Mediator - a smart contract programmed to run the Stackelberg game-theory based selection algorithm to discover optimal MNOs to create slices. Unit prices of the resources are required to perform the slice selection operation. Thus, pricing data is retrieved from the database.
- Global slice manager - a smart contract programmed to invoke North Bound Interface (NBI) of Katana REST API to formulate federated network slices.

One of the main security biased components in our proposed architecture is the Security Service Blockchain (SSB), which is responsible for preventing DoS attacks on SFSBroker. These attacks might be launched by malicious IoT tenants or from the compromised MNOs.

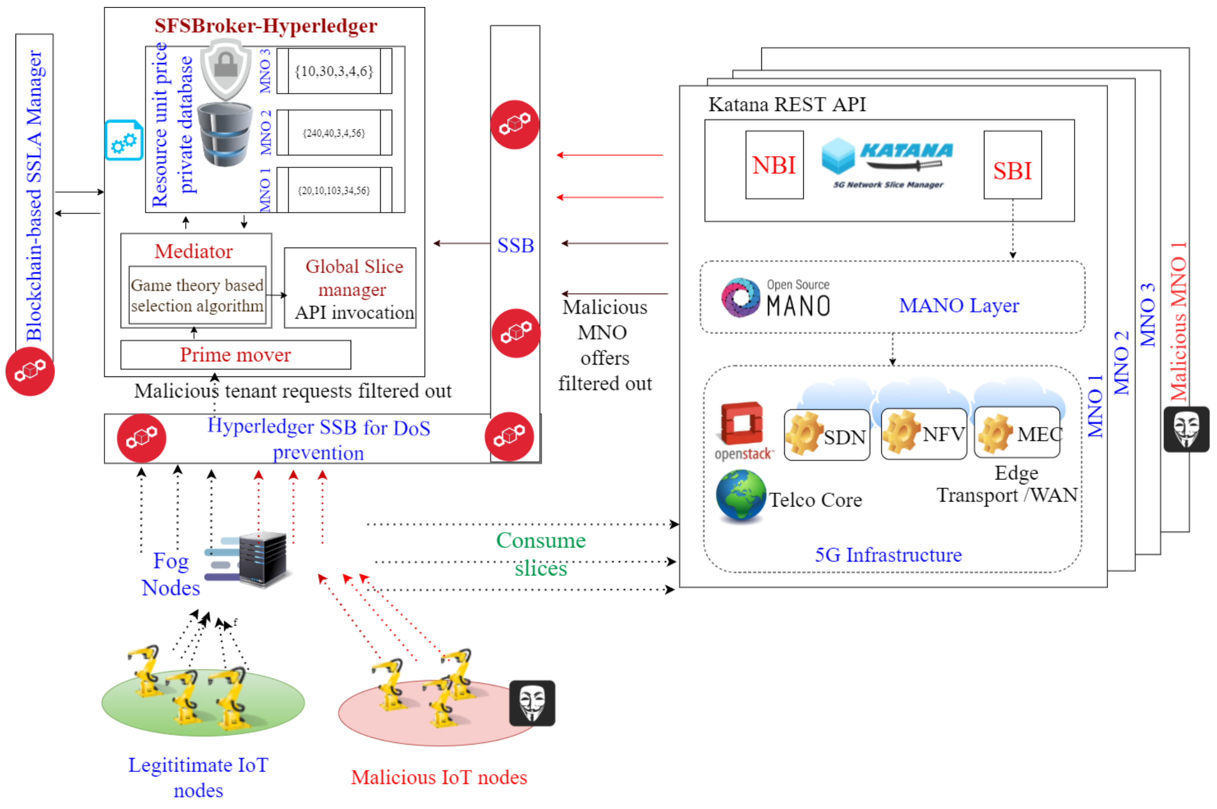


Figure 43: Integration setup for TC9

The interactions between the aforementioned components of the proposed architecture are illustrated and described below.

With reference to the Figure 44, the whole process initiates when an IoT tenant submits a request for a network slice along with their preferred security requirements (1). The Security Service Blockchain (SSB) is responsible to defend incoming slice requests to the proposed architecture, against DoS attacks (2). The verified slice request will be sent to the SFSBroker (3). In the subsequent stage, SFSBroker runs the slice selection algorithm, where optimal resource providers are chosen and allocated for a network slice aligned with tenant requirements (4). Afterwards, SFSBroker notifies all the slice managers relevant to the newly formed slice (5). Next, the slice manager sends the network slice blueprint to the tenant (6). Meanwhile, SFSBroker notifies blockchain-based SSLA managers that slice has been instantiated and forwards the SSLA information (7). Thereafter, Blockchain-based SSLA manager initiates slice monitoring process, where it thoroughly checks whether the MNO service offerings are in compliance with the SSLA rules (8). The slice termination triggers if the agreement expires or if any SSLA breaches are discovered.

In our case, MNOs are allowed to add or modify their network resources and their unit prices. The slice managers of each MNOs are responsible for updating this information.

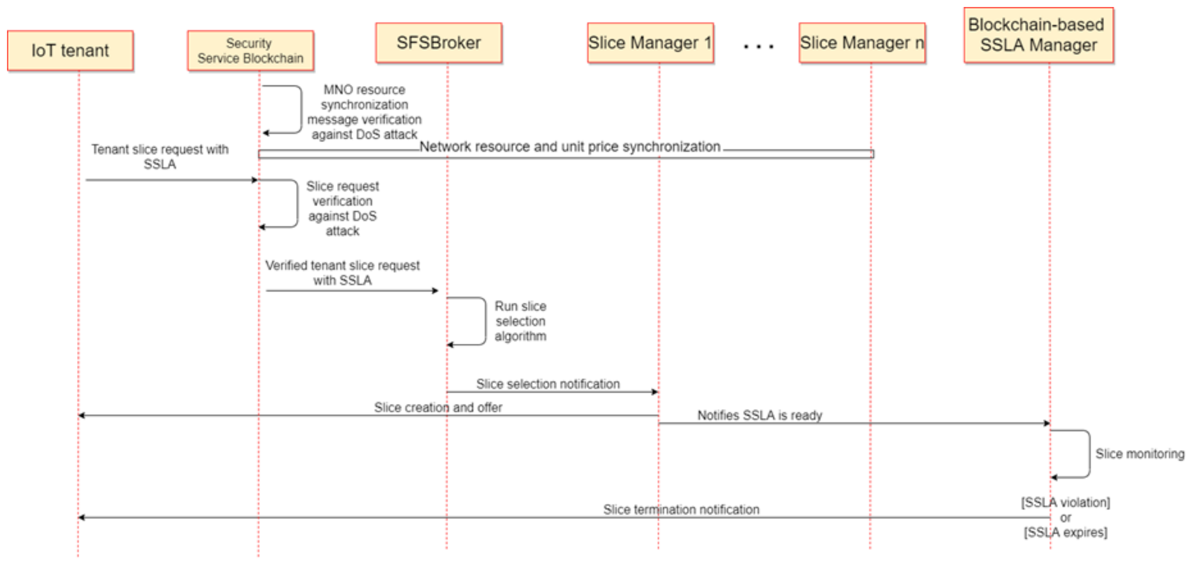

IoT tenant
Security Service Blockchain
SFSBroker
Slice Manager 1
. . .
Slice Manager n
Blockchain-based SSLA Manager
MNO resource synchronization message verification against DoS attack
Network resource and unit price synchronization
Tenant slice request with SSLA
Slice request verification against DoS attack
Verified tenant slice request with SSLA
Run slice selection algorithm
Slice selection notification
Slice creation and offer
Notifies SSLA is ready
Slice monitoring
[SSLA violation] or [SSLA expires]
Slice termination notification

Figure 44: The workflow diagram of TC9

3.2.8.2 Definition of the test sequence

This TC considers the secure and privacy enabled resource allocation for network slicing in a multi-operator multi-tenant platform. As shown in Figure 45, the scenario in the test case demonstrates the use of blockchain based solutions for network slice brokering and SSLAs for local 5G operators and infrastructure providers running on a common platform. The first phase of the test case use is to integrate two security enablers including SFSBroker from UOULU and Katana Slice Manager from NCRSD.

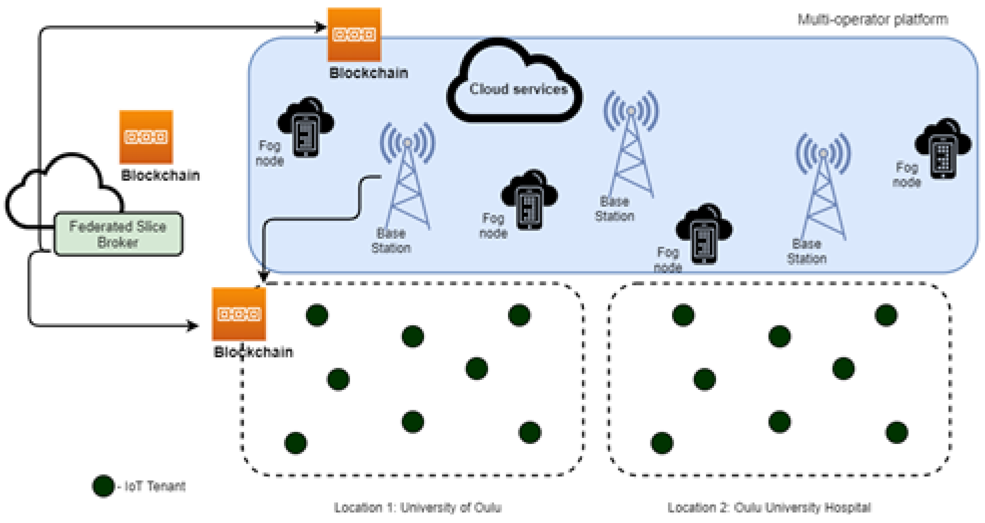


Figure 45: Operating scenario for TC9

Test Case Name	TC9_SliceM_SliceB
Test Purpose	Validate the integration between the Secured Federated Slice Broker and MNO's slice manager
Description	The test will be used to validate the services for the retrieval of MNO's resources and instantiate the slice from template
Scenario	Presented in Figure 45

Test Case Name		TC9_SliceM_SliceB	
Test flow		Presented in Figure 46	
Test sequence	Steps	Description	Result
	1	Setting Up Environment	Define the environment information, including services and integration points to be used during the test.
	2	Retrieval of available resources (VIM) from each MNOs and store in the ledger	Retrieve the available resources and their unit prices from the individual MNOs
	3	Receive the resource request from tenant	Acceptance of the resource request for the slice from fog nodes which represent the consumer end
	4	Compute the best offer based on the stored data	The game theory-based algorithm running on SFSBroker's smart contracts to select the optimal slice based on the information
	5	Instantiate the slice	The SFSBroker invokes the MNO's corresponding service to instantiate the slice
Test verdict		If there is no error, the corresponding slice is correctly instantiated	

Table 30: Test sequence for SliceM_SliceB

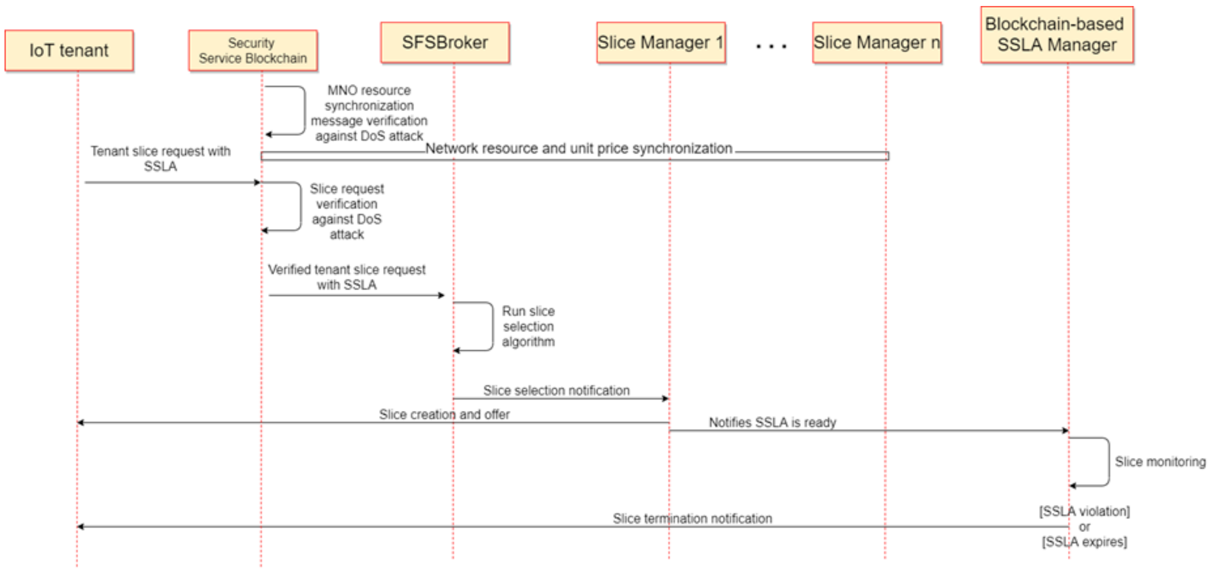

IoT tenant
Security Service Blockchain
SFSBroker
Slice Manager 1
. . .
Slice Manager n
Blockchain-based SSLA Manager
MNO resource synchronization message verification against DoS attack
Network resource and unit price synchronization
Tenant slice request with SSLA
Slice request verification against DoS attack
Verified tenant slice request with SSLA
Run slice selection algorithm
Slice selection notification
Slice creation and offer
Notifies SSLA is ready
Slice monitoring
[SSLA violation] or [SSLA expires]
Slice termination notification

Figure 46: Test flow for SliceM_SliceB

3.3 Preliminary implementation of the INSPIRE-5Gplus closed loop

A preliminary implementation of the INSPIRE-5Gplus closed loop was developed and deployed for showcasing the INSPIRE-5Gplus' closed-loop on top of the first instantiation of the HLA in a multi-site environment. The purpose of this implementation is to demonstrate the interactions among the main functional elements of the HLA irrespective of the mitigation of various security threats/attacks performed in the TCs (as described in Section 3.2).

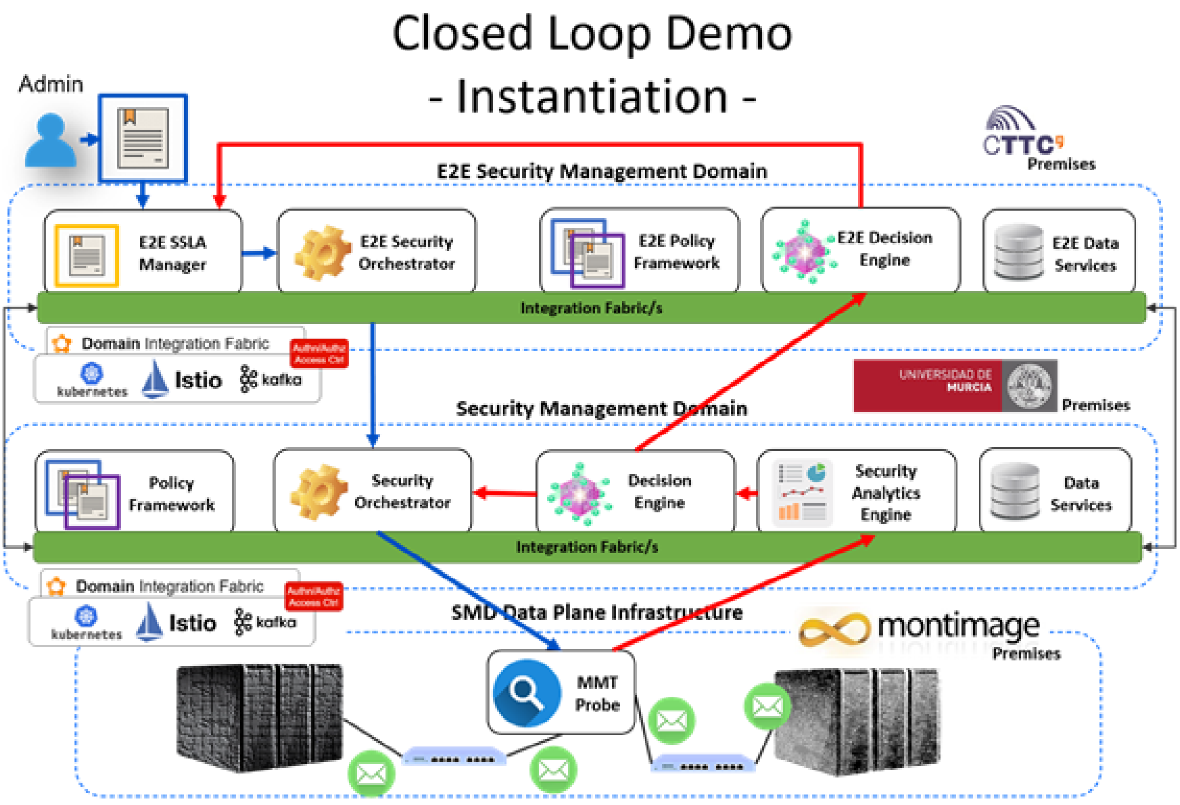


Figure 47: Preliminary implementation of the INSPIRE-5Gplus closed loop

Figure 47 shows the closed loop instantiation for the preliminary implementation. Blue arrows represent the proactive part of the closed-loop whereas red arrows represent the reactive part of the closed loop. As a proactive measure, SSLAs enforcement can be requested at the E2E Security Management Domain level to limit the protocols allowed. SSLAs are then refined in per-domain policies and sent to the involved Security Management Domains (SMD), according to the SSLA requirements. Once each SMD receives the security policies, these are translated into specific actions and enforced according to the SMD orchestration process at SMD level. For instance, in the previous figure, security policies are translated into monitoring configuration of a specific monitoring tool available in the SMD to detect the limits established by the SSLAs.

Regarding the reactive part of the closed loop, when the limits established are overtaken (with a simulated event) by unlawful traffic being injected, the unlawful traffic is detected by the already configured monitoring tool, which sends an alert to the Security Analytics Engine to be further analyzed. As a result of the analysis, a new countermeasure in form of a new SMD security policy is generated dynamically. This new security policy specifies that the offending traffic must be forwarded to other monitoring tools to be further analysed, therefore closing the loop for the Security Management Domain.

In parallel to the Security Management Domain closed-loop closure, a notification towards E2E Security Management Domain is also sent that in turn creates an E2E reaction, closing the loop for the E2E Security Management Domain level as it is shown in Figure 48.

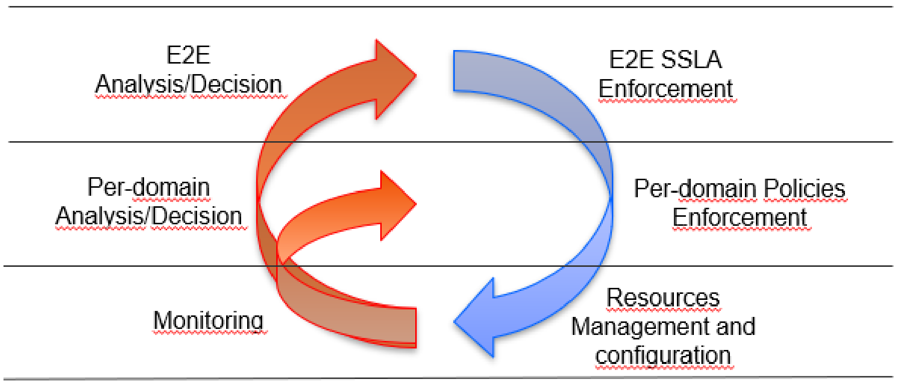

E2E Analysis/Decision
E2E SSLA Enforcement
Per-domain Analysis/Decision
Per-domain Policies Enforcement
Monitoring
Resources Management and configuration

Figure 48: SMD/E2E SMD closed-loop

It is important to highlight that the communication between the different elements developed as result of the preliminary implementation is done through an instantiation of the multi-domain Integration Fabric proposed in D2.2.

To validate this preliminary implementation, the following demonstration was performed:

A. The instantiation of SSLA (SSLA Manager) at E2E Security Management Domain level deployed on CTTC premises and the refinement into Security Policies (E2E PF) were showcased. The SSLA defined a SLA (metrics and Service Level Objectives) on a protocol capabilities limitation Detection and Prevention service; this SSLA was then translated into HSPL-OP (it can be translated directly to MSPL-OP).

B. The enforcement of the E2E policies onto SMD (E2E SO) was showed. Therefore HSPL-OP refinement to MSPL-OP was performed and sent to the affected Security Management Domain. In this case, the one deployed over UMU and MontImage premises.

C. The translation (SO and PF) of the Security Policies at SMD level into enforcement actions over the infrastructure was showed. This means that the Monitoring tool was configured to control that the protocols limits established were not exceeded on the data plane and in such case, notify the architecture.

D. The acquisition of notifications (SAE) from the infrastructure and the corresponding reaction (DE) was showed. The monitoring tool detected offending traffic and sent the notification to be analyzed. After being analyzed, a dynamic reaction was decided to allow further research of the traffic and therefore issuing a forwarding policy to be enforced at the SMD level.

E. Enforcement of the reaction (SO), closing the loop at SMD level, while notification to E2E SMD is triggered (DE) was performed. The forwarding policy generated was then enforced and notified towards the E2E SMD, so that possible reactions taking into account the multi-domain constitution of the solution and potentially producing new reactions that would affect neighbouring SMDs.

Notification was received at E2E level (E2E DE), analyzed and reacts, in this case closing the loop at E2E level (SSLA Manager). For the demo the notification of no further action needed to the SSLA closes the loop. Even if no action is performed, this notification is necessary for further decisions and liability analysis.

4 KPIs for Operational Validation

This section elaborates the definition and evaluation methodology of KPIs which are applicable to INSPIRE-5Gplus TCs. This set of KPIs represents a horizontal security/trust assessment framework covering multiple testing environments for the evaluation of the respective integrated security and trust/liability enablers. We provide the details in the following subsections.

4.1 Mean Time to Detect

4.1.1 Definition

Mean Time to Detect (MTTD) is defined as the average length of time between the start of adversary acts and their discovery[6]. This KPI indicates the quality of security incidents detection. MTTD assumes a consistent method for identifying when an incident begins and when an incident is discovered. The vertical industry specifies the required MTTD values in SSLAs. In general, lower MTTD values are better.

4.1.2 Evaluation methodology

MTTD is the amount of time that elapsed between the Date of Occurrence and the Date of Discovery for a given set of incidents, divided by the number of incidents[7]:

$$MTTD = \frac{\sum_i (Date\ of\ Discovery_i - Date\ of\ Occurence_i)}{Number\ of\ incidents}$$

The measurement unit can be expressed in [time]/incident, where [time] can be hours, minutes or seconds in order to provide readability of the result. The calculation can be grouped per types of incidents or incident severity. It is recommended[8] to use the following incident categories:

- Denial of Service
- Malicious Code
- Unauthorized access
- Inappropriate usage

MTTD depends on several elements, including the processing power of computers and servers, configuration of monitoring systems (real-time or batch processing), delays in the underlying infrastructure, QoS configuration and distance between the infrastructures and the monitoring systems[9].

[6] Deb Bodeau, Rich Graubart, Len LaPadula, Peter Kertzner, Arnie Rosenthal, Jay Brennan, Cyber Resiliency Metrics Version 1.0 Rev.1, April 2012. Available online: https://register.mitre.org/sr/12_2226.pdf, Accessed: 09/2021.

[7] ENISA, Resilience Metrics and Measurements: Technical Report, February 2011, Available online: https://www.enisa.europa.eu/publications/metrics-tech-report, Accessed: 09/2021

[8] Paul Cichonski, Tom Millar, Tim Grance, Karen Scarfone, Computer Security Incident Handling Guide Rev. 2, Recommendations of the National Institute of Standards and Technology (NIST), August 2012, http://dx.doi.org/10.6028/NIST.SP.800-61r2

[9] C. Onwubiko and K. Ouazzane, "SOTER: A Playbook for Cybersecurity Incident Management," in IEEE Transactions on Engineering Management, doi: 10.1109/TEM.2020.2979832.

4.2 Mean Time to Contain

4.2.1 Definition

Mean Time to Contain (MTTC) is defined as the average time duration it takes for an incident-response system to: i) detect an incident, ii) acknowledge it, and iii) effectively prevent the propagation of the security incident over the network components. In particular, the first step involves the identification that an event has occurred that may, potentially, represent a security incident; the second step determines that the event is in fact a security incident; the third step aims to limit the resulting impairment by the incident and prevent the attacker from causing further harm. It is to be noted that typically the third step does not imply full remediation of the attack which often requires re-evaluating the specific security policy. A similar set of incident categories as in MTTD can also be considered for the definition of MTTC.

4.2.2 Evaluation methodology

MTTC is calculated as the total time to conduct all the aforementioned steps, i.e., the time it takes to detect, understand, and contain, for a given set of security incidents, averaged across all incidents which have occurred in the system. The mathematical expression of MTTC can be defined as follows:

$$MTTC = \frac{\sum_i Total\ time\ to\ detect, understand\ and\ contain\ a\ security\ incident}{Number\ of\ incidents}$$

The evaluation of the median MTTC can be also used as a quantitative security performance metric, i.e., the lower the median MTTC, the more effective a security monitoring framework can be considered.

4.3 Mean Time to Resolve

4.3.1 Definition

As a security KPI, Mean Time to Resolve (MTTR) can be defined as the mean/average time to resolve a security incident such as an attack detection, mitigation, running a protection mechanism, or creating an SSLA. MTTR measures how long it takes the system to resolve potential security incidents within the network environment. Typically, MTTR is an explicit indicator that describes the availability and reliability of a networking system against security threats. Availability indicates the probability that the system is operational at any specific instantaneous point in time. Reliability indicates the probability that a service will remain operational over its life-cycle. The definition of MTTR can also be expanded to quantify the time needed for a system to regain normal operation performance. In this case, MTTR incorporates not only the time spent detecting the incident, diagnosing the problem, and resolving the issue, but also the time spent ensuring that the security incident will not occur again at platform level.

MTTR security KPI can be related to the general 5G-PPP performance KPIs in terms of enhancing capacity, reducing service time, etc.

4.3.2 Evaluation methodology

In principle, MTTR is the average time duration to offer a given security service (e.g., identify a security threat, mitigate an attack, etc.). The shorter the MTTR values, the higher the reliability and availability of the service against security threats. The mathematical expression of MTTR can be defined as follows:

$$MTTR = \frac{Total\ duration\ of\ operational\ time\ for\ security\ service}{Total\ number\ of\ service\ requests\ or\ security\ incidents}$$

4.4 Transaction speed

4.4.1 Definition

The KPI transaction speed is used to measure the number of transactions per second that can be performed in a given service (e.g., a blockchain). Transaction speed is the rate at which a service is transferred from one entity to another. The faster a transaction is confirmed, the better the transaction speed is said to be. For instance, transaction speed of a blockchain is one of the prime parameters through which viability of a blockchain is gauged. The block processing time within a given unit time can be considered. Transaction speed in turn hinges from numerous other factors like block size, block time, traffic on the network, or transaction fees. It can be also defined as the number of total security services or transactions performed in a time unit.

Transaction speed security KPI can be related to the general 5G-PPP performance KPIs in terms of enhancing capacity, reducing average service time, facilitating dense deployments, etc.

4.4.2 Evaluation methodology

The evaluation methodology is defined as below:

Transaction speed is evaluated by computing the total number of individual security services/transactions performed in a unit time. This means the total number of transactions per second (tps) the network can handle.

The mathematical expression of transaction speed can be defined as follows:

$$Transaction\ speed\ = \frac{Total\ number\ of\ transactions\ executed\ for\ the\ test}{Total\ duration\ of\ the\ test}$$

4.5 Packet loss ratio

4.5.1 Definition

Packet Loss Ratio (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. Packet losses could usually occur due to channel errors or network congestion. This metric is typically associated with Quality of Service (QoS) considerations and the amount of tolerable packet losses (e.g., 1% or 5%-10%) depends on the type of data being sent. In the context of network security, packet-dropping or blackhole is a type of DoS attack where a network node drops (i.e., erase) packets that it should not have. A DoS attack can happen at different layers, e.g., application layer or network layer. If a node is repeatedly dropping packets, that is an indication of potential malicious behaviour which could lead to communication unavailability for benign users.

4.5.2 Evaluation methodology

PLR is computed as a percentage of packet drops with respect to packets forwarded over a specific time period. The mathematical expression of PLR can be defined as follows:

$$PLR = \frac{Total\ number\ of\ dropped/lost\ packets}{Total\ number\ of\ packets}$$

The impact of a packet-dropping attack can be evaluated by computing PLR under the presence of the attack and without the attack. The lower the PLR, the higher the reliability and availability of the service against security threats. Alternatively, PLR can be measured by computing the packet delivery ratio (PDR) which represents the ratio of total packets received to the number of packets that have actually been forwarded by a network node. The higher the PDR, the higher the reliability and availability.

4.6 False positives

4.6.1 Definition

False Positive (FP) is considered as a type of error for a binary classification. In the context of security and specifically in the machine learning field is commonly associated with the concept of false alarm. The idea behind this concept is that from a dataset used by a model, a sample fits in the condition that the model is searching, when in fact it is not. Typical examples can be a valid email classified "positively" as spam, or a network flow associated with malware activity. Depending on the application area, avoiding false positives can be critical (e.g., benign software classified as malware and stopped). Automatic close-loop process without human in the loop, will require very low false positives to be reliable. On the other hand, systems outputs with high rate of false positive need to be monitored by an expert to discard them. True positive (TP), is an opposite concept, where the binary classifier correctly identifies what is searching for.

4.6.2 Evaluation methodology

To make the evaluation a "validation dataset" is needed. It will include a "label" for each entry with the real information if the condition is met or not. This dataset will not be used to train ML models.

Same dataset without the "label" will be delivered to the classifier inference engine. Each calculated prediction done by the classifier is compared with the real value from the validation dataset. For each wrong positive prediction, a counter is increased. The final count is the False positive (FP) number.

In addition, multiple statistical values can be derived from the combination of the TP, FP, False Negative (FN) and True Negative (TN). Accuracy, precision, recall or F1-score, are commonly used. For example, precision can be calculated as the percentage of TP of the total positives in the validation dataset:

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

4.7 False negatives

4.7.1 Definition

False Negative (FN) is considered the complementary error measurement to False Positive (FP). Commonly, this KPI is associated to the idea of falling "under the radar" in security context. In this case, it refers to the samples in a dataset that are not detected by the classifier despite to be what is searching for (accomplish the condition). Using the spam filter example, a spam email is treated as a rightful email. True negative (TN), is the opposite concept, where the binary classifier correctly identifies what is not searching for.

4.7.2 Evaluation methodology

The evaluation methodology follows a similar approach to FP. The same labelled validation dataset is needed. Each calculated prediction done by the classifier is compared with the real value in label from the validation dataset. For each wrong false prediction (i.e., despite to accomplish the condition, the classifier fails), a counter is increased. The final count is the FN number. As mentioned in the previous FP KPI, it can be combined to obtain more statistical metrics.

4.8 Initial time

4.8.1 Definition

Initial Time (IT) is defined as the elapsed time until messages can be processed by the network, a domain or a platform after an event. This event can be the deployment of a new component, a change on the topology/network, or even a recovery from an attack. Depending on the event, the measure will differ. However, the most extended use of this KPI is to measure the time a new component (for example, a countermeasure VNF or a security component as an IPSec tunnel) is ready and working completely.

4.8.2 Evaluation methodology

Initial Time (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, i.e.,

$$IT = PT - ET$$

where ET is the enforcement time measured on the device in which the asset is deployed and PT is the time in which the asset is ready). PT can be measured by means of LOG (i.e., recorded) messages or by monitoring operating system resources such as opened network ports. When the asset is distributed or follows a Service Function Chaining (SFC) approach, ET makes reference to the request arrival to the first element of the chain while PT needs to measure the entry point for the chain. Nevertheless, PT cannot be measured until each of the element of the chain is initialized, therefore assistance from the components is needed unless the readiness of each part of the chain can be measured via external events such as opened port, socket files, etc.

4.9 Migration time

4.9.1 Definition

Migration Time (MT) is defined as the time required to migrate assets, (i.e., Network Functions NFs), or scale computing/network resources since the moment the last message is processed in the initial state till the first message is processed to the migrated state. This time is crucial since in situations such as attacks or failure of a node/part of the infrastructure, being able to migrate the components in the shortest time is essential to provide the highest QoS. A different situation where MT can be relevant is when a migration or a rescale is required to improve the QoS (i.e., a new closer node is available, more resources are needed, or the client has moved and it is now farther from its original position). In these situations, MT is less critical since the service can still be provided during the process, but its QoS is significantly affected.

4.9.2 Evaluation methodology

MT is calculated as the time elapsed between the last message received in the original location, until the first message is received at the new location. At that point the new instance is fully operational and exchanges messages with the corresponding device. Therefore,

$$MT = FM - LM$$

where *LM (Last Message)* is the exact moment in which the initial component receives its last message, and FM (Forwarded Message) the moment when the new component (or network) provides the migrated function. If any data is required not only for providing the function but actually for maintaining the binding established in previous locations, that time is considered part of FM - LM and can be measured as Data Migration (DM), as a sub-KPI even if the MT is not affected since it is included in the already provided calculation.

4.10 Service response time

4.10.1 Definition

The service response time is defined as the time elapsed between request issuance and the corresponding response reception by an end user. It is an essential metric to assess the quality and availability of a service when an attack is ongoing. A high response time is an indication of the performance degradation of services that could be caused by a malicious behaviour such as a DDoS attack.

4.10.2 Evaluation methodology

The service response time is calculated as the elapsed time between when the request is sent by a legitimate user and when the corresponding response is fully received. To assess the effectiveness of a mitigation solution, this time is compared to a reference time where the mitigation solution is not used, and the improvement ratio is calculated.

4.11 Service downtime

4.11.1 Definition

Service downtime (SDT) measures the percentage of time a service (e.g., access to the network) is not working or is considered unavailable by the users (i.e., human or machines). In telecommunications, carrier-grade network services must be available continuously reaching even 99.999% uptime, even though this is more of a theoretical wish that a reality. The required uptime can be defined by the SLAs with explicit penalties if not satisfied. With respect to security, this KPI is closely related to the MTTR or the Mean Time Between Failures (MTBF) but also includes the downtime that can be caused by maintenance actions (e.g., updating a module, patching to eliminate a vulnerability in the code). Therefore, it constitutes a KPI that measures the availability of services as perceived by the user of the services.

As seen from the vertical applications' perspective the uptime for critical safety applications needs to be much higher that, for instance, consumer applications. Hence service downtime can also be closely linked to the latency KPI. Latency-critical applications could be considered as not satisfactory if a certain latency level is not respected. Thus, service downtime requirements also depend on the nature of supported applications.

4.11.2 Evaluation methodology

Service downtime can be measured for each service (vertical application or network function). Service availability or uptime is the percentage of time the service is operational or without degradation. SDT can be expressed as the percent of downtime over a given time period (day, week, month, year...), as the average outage time interval (AOI), or as the average time interval between failures (ATF). The number of failures or interruptions, or the average time to recovery when a failure occurs are metrics that can be useful to determine average service downtime.

*SDT = (Time not available or with degradation) / (Time in operation) * 100*

AOI = (Time not available or with degradation) / (Number of incidents)

ATF = (Time between failures or degradations) / (Number of incidents)

In INSPIRE-5Gplus, the service downtime, in the case of security breaches or the application of security measures, means that a service should only be degraded during the shortest transitory time slot due to the introduction of the security controls and mitigation strategies.

4.12 Security service level agreement enforcement

4.12.1 Definition

SSLA enforcement relates to a set of security requirements and rules - on a service - that have been decided between two constituents, and that need to be satisfied. It is an extension of the SLA concept used to define the level of service a customer expects from a vendor and it allows to define the metrics by which a service level is measured, as well as remedies or penalties applied if the agreed-upon service levels are not achieved.

To that aim, SSLAs add the possibility of agreeing on the expected level of security as defined by the security functions and controls that are implemented in a vertical, service, network or network slice; and, the security breaches that are detected, prevented and mitigated. SSLAs can be related to other security KPIs covered before (e.g., FP, FN and MTTR) since they can be measured using these KPIs, or can specify the allowed thresholds for these KPIs that must be respected by other security functions.

4.12.2 Evaluation methodology

In INSPIRE-5Gplus, this KPI can be measured by determining (at all times) that any violation of a specified SSLA is detected, and that the measures to enforce it are applied independently from any event in the network, such as maintenance actions, triggering of mitigation strategies (e.g., moving target defence, allocating a new slice), cyber-attacks (e.g., DDoS/DoS) or performance problems, etc.

4.13 Blocked adversarial examples rate

4.13.1 Definition

The blocked adversarial examples rate is defined as the percentage of adversarial examples generated to fool an ML model and successfully detected by the ML-assisted detection and mitigation system. In fact, adversarial attacks against an ML-based detection/mitigation solution are detrimental if the ML model is not able to resist them. An adversary may craft inputs to fool the ML model into making wrong decisions potentially resulting in endangering SLA fulfilment and security guarantees. For instance, an adversary may generate crafted traffic samples that result in misclassifying a DDoS traffic as normal traffic or identifying a malicious scaling-up operation as a legitimate operation.

4.13.2 Evaluation methodology

The metric is measured by calculating the percentage of adversarial examples that the ML model can resist compared to the total number of adversarial examples generated by the attacker. In particular:

$$Blocked\ Adversarial\ Examples\ Rate\ = \frac{Adversarial\ examples\ successfully\ detected}{Total\ number\ of\ adversarial\ examples}$$

4.14 Ratio of allowed malicious scale-up

4.14.1 Definition

The ratio of allowed malicious scale-up is defined as the percentage of malicious scale-up requests that have been triggered by a malicious workload and not correctly detected by the mitigation solution. If auto-scaling allows to deal with the load at service and/or infrastructure resources by automatically resizing (i.e., provisioning resources) the network slice to meet the SLA requirements, it can result in resource starvation and/or undesirable costs under (D)DoS attack. Thus, it is important that the mitigation solution could reduce as much as possible the execution of malicious scale-up requests.

4.14.2 Evaluation methodology

This metric is measured by calculating the percentage of allowed scale-up operations triggered by malicious behaviour with respect to the total number of malicious scale-up requests generated.

$$Ratio\ of\ allowed\ malicious\ scale-up = \frac{Number\ of\ malicious\ scale-up\ operations\ allowed}{Total\ number\ of\ malicious\ scale-up\ requests\ generated}$$

4.15 Automated vulnerability assessment

4.15.1 Definition

The automated vulnerability assessment is defined as the percentage of the vulnerabilities that can be identified from the information encoded on a model that can be used to exploit that actual system. The vulnerability identification is a process that uses the values from the attributes of certain concepts to identify vulnerabilities of the system. The quality of the output is affected by the detail of the input. Generic attribute values in the model will generate a large number of vulnerabilities, while more specific values will generate a smaller but more relevant number of vulnerabilities.

4.15.2 Evaluation methodology

The evaluation methodology is defined as below:

- Step 1: Design and implement a system with known vulnerabilities
- Step 2: Document the known vulnerabilities of the system
- Step 3: Model the system using the software tool DiscØvery[10]
- Step 4: Perform the automated vulnerability function of the tool
- Step 5: Compare the results of the automated analysis with the ones documented on Step 2

4.16 Automated model generation

4.16.1 Definition

The automated model generation is defined as the process that can model in a graphical manner the network components of a system with information derived from network traffic. This KPI is measured as a percentage of the actual components that are part of the system when compared to the components that have been modelled through algorithmic functions. For example, based on network traffic information we can identify network components such as machines, and applications with network access (web servers, internet browsers, update daemons, etc.).

4.16.2 Evaluation methodology

The KPI is measured as defined below:

- Step 1: We design and implement a system
- Step 2: We document the network components of system (devices, network applications, etc.)
- Step 3: We start capturing the network traffic of the system until is performs its full range of functions

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

- Step 4: We input the network capture file to algorithm process that automates the model generation
- Step 5: We measure the number of actual components that the algorithm outputted based on the documented network components from Step 2

4.17 Threat assessment

4.17.1 Definition

The threat assessment is defined as the percentage of the threats that can affect the system once it is implemented. Similarly, to automated vulnerability assessment, this process makes use of the values encoded on a system model to derive which threat can target the system. This process will generate a list of threats that can impact the modelled system based on its attributes and configuration. The threat list can include high-level threats that impact to system policy as well as low-level threats that impact the deployment or the configuration of the system.

4.17.2 Evaluation methodology

The evaluation methodology is defined as below:

- Step 1: Design and implement a system with known threats
- Step 2: Document the threats of the system
- Step 3: Model the system using the software tool DiscØvery
- Step 4: Perform the threat identification function of the tool
- Step 5: Compare the results of the analysis with the threats that were documented on Step 2

4.18 Cyber-security insights assessment

4.18.1 Definition

The cyber-security insights assessment is defined as the percentage of relevant processes, such as security mechanisms, policies, etc., that can be used to improve the security posture of the system. Those insights can provide the security analyst with additional information about the security posture of the system by highlighting possible security issues of the system’s configuration. For example, a system could have a connection that supports the TELNET protocol, which lacks encryption during data transmission. An insight could be to “use a secure transmission channel for wireless protocols that lack encryption”.

4.18.2 Evaluation methodology

The evaluation methodology is defined as below:

- Step 1: Model a system using the modelling language supported by the DiscØvery tool
- Step 2: Perform the cyber-insights function of the tool
- Step 3: Measure the percentage of the proposed cyber-insights that can be used to improve the security posture of the system

4.19 Latency

4.19.1 Definition

Latency is defined as the time it takes for a data packet to be transferred from its source to the destination. Depending on the layer under consideration, e.g., medium access control, network, transport, application, the exact definition of latency may vary. In this context, latency can provide an indication of i) the distance between source and destination, ii) the quality of the transmission medium, iii) the number of intermediate network hops, iv) the traffic congestion levels, and v) the server response time. In principle, latency drives the responsiveness of the network and can be used to explore the trade-offs between security and actual network performance in low-latency network services.

4.19.2 Evaluation methodology

Latency can be either evaluated considering the amount of packet delivery time for the point-to-point (or one way) communication link or by calculating the total round trip time (RTT). The mathematical expressions of latency can thus take the following forms:

$$Latency\ (one-way) = Timestamp_{dest} - Timestamp_{src}$$

$$Latency\ (RTT)\ = Timestamp_{src-new} - Timestamp_{src-old}$$

A common approach is to measure the latency performance by computing the difference when a security asset is enabled and without its presence. In this context, latency measurements evaluate whether security comes at the expense of degraded network performance for mission-critical services.

4.20 Mean Time to implement the MTD action

4.20.1 Definition

Mean Time to implement the MTD action (MTID) is defined as the average time required for an MTD action to be successfully executed, after the OptSFC enabler decides the MTD action to perform and the MOTDEC enabler implements it. MTD actions can be grouped into two categories: Hard MTD actions, corresponding to MTD actions with high MTID, like restarting a service or relocating it to a different NFVI, and Soft MTD actions, with smaller MTID like SDN operations or dynamic micro-settings. Hard MTD actions can be performed while running the original instance of the service, and switch to the new instance only after its completion. This avoids QoS disruption. Soft MTD actions, when directly executed on the running instance, should not exceed 5 seconds.

4.20.2 Evaluation methodology

To evaluate the MTID we need to consider several factors: *Factor 1*, The mean time for OptSFC to communicate the MTD action to enforce to MOTDEC; *Factor 2*, The mean time for MOTDEC to consult the SSLA and Policy Manager to validate the MTD action at the High-Level Architecture (HLA) level; and *Factor 3*, The mean time for MOTDEC to enforce the MTD action.

MTID = AVERAGE(Factor1 + Factor2 + Factor3)

The measurement unit is [time], which can be expressed in seconds, minutes or hours depending on the best readability.

4.21 MTD action cost

4.21.1 Definition

MTD action cost (MTDAC) is a comparative value showing the overhead of MTD actions on: CPU load, random-access memory (RAM) load, and response time for the protected service. We set the ideal overhead not to exceed a 20% increase in the mean case, and 50% increase in the worst case.

4.21.2 Evaluation methodology

MTDAC is evaluated by monitoring the change in the resource consumption, namely: *CPU*, *RAM*, *traffic throughput* and *latency*. The measurement unit is percentage (%).

4.22 Protection gain of an MTD policy

4.22.1 Definition

Protection gain of an MTD policy (PGMTD) is a comparative value showing the gain in protection terms of performed MTD strategies/policies. The PGMTD is different for every distinct attack category, as an MTD policy can be more effective on certain attacks rather than others. The following categorization is used[11]: Denial of Service, Malicious Code, Unauthorized access, Inappropriate usage, and Multiple component incidents.

4.22.2 Evaluation methodology

The PGMTD is evaluated by simulating an attack on two different instances of the same service, one with MTD protection and one without it. Then we measure the difference in the Attack Success Probability (ASP) between the two instances. The measurement unit is percentage (%). The greater the difference, the better it is. The ASP is calculated by repeating an attack simulation and statistically measure its probability to succeed. Ideally, we would like to have a mean PGMTD of 10% or greater, and have worst case scenarios with PGMTD not lower than 5%.

4.23 Mean decision time for MTD action

4.23.1 Definition

Mean decision time for MTD action (MDTA) is defined as the mean time needed for the OptSFC enabler to decide which action to take after an alert has been triggered by a security agent, like the anomaly detection framework. The measurement unit is [time], expressed in milliseconds, as the MDTA should be highly responsive, targeting 500ms as the slower response.

4.23.2 Evaluation methodology

MDTA is evaluated empirically by measuring the mean time needed for the AI/ML algorithm to decide the MTD action to perform. Such time heavily depends on the type of ML algorithm and on the number of parameters in input.

[11] ENISA, Resilience Metrics and Measurements: Technical Report, February 2011, Available online: https://www.enisa.europa.eu/publications/metrics-tech-report, Accessed: 09/2021

4.24 QoS gain/loss of the protected resources

4.24.1 Definition

QoS gain/loss of the protected resources (QoSO) is defined as the overhead of the MTD framework on the QoS of the protected resources. The QoS might even get better if the MTD actions performed also consider distance, data and resource consumption (e.g., move a service from a busy NFVI to a free one).

4.24.2 Evaluation methodology

The QoS to evaluate may have different natures, based on the service type, i.e., eMBB (enhanced Mobile Broadband), URLLC (Ultra Reliable Low Latency Communications), or mMTC (massive Machine Type Communications). When security prevails over the QoS, (i.e., in the case of attack detection, a loss on the QoS is envisaged), it should bring a degradation beyond 10%. In the average, where the MTD framework is running MTD actions for proactive and preventive defenses, resource consumption and server relative position can be important factors of the decisions, and a light gain in the QoS can be considered (e.g., up to 5%).

4.25 Summary

This section extended and elaborated the identified INSPIRE-5Gplus KPIs stemming from the development of specific security and trust/liability INSPIRE-5Gplus enablers reported in D5.1. Leveraging on the comprehensive description of the different TCs and the undergoing evolution of the integrated security and trust/liability enablers, we have provided the definition and evaluation methodology of relevant KPIs. The contents of this section are expected to act as a reference point for the assessment of experimental activities to be carried out in the context of WP5.

Table 31 summarizes the mapping between the INSPIRE-5Gplus KPIs with the WP5 TCs. Several KPIs cover multiple TCs, aiming to provide a coherent evaluation environment among different testing environments, while other KPIs capture the peculiar characteristics of certain TCs and corresponding integrated enablers (Legend: X= mandatory, O= optional).

KPIs	TC1	TC2	TC3/TC4	TC5	TC6	TC7	TC8	TC9
Mean Time to Detect	X	X	X	X				
Mean Time to Contain	X		X					
Mean Time to Resolve	X		X	X				X
Transaction speed	X				O			X
Packet Loss Ratio	X	X			O			
Number of False positives		X	X				X	
Number of False negatives		X	X				X	

KPIs	TC1	TC2	TC3/TC 4	TC5	TC6	TC7	TC8	TC9
Initial time			X		X			
Migration time			X		X			
Service response time	O		O		O	X		
Service downtime	O		O		O	X		
SSLA enforcement	O	O				X		
Blocked adversarial examples rate						X		
Ratio of allowed malicious scale-up						X		
Automated vulnerability assessment							X	
Automated model generation							X	
Threat assessment							X	
Cyber-security insights assessment							X	
Latency	X							
Mean Time to implement the MTD action				X				
MTD action cost				X				
Protection gain of an MTD policy				X				
Mean decision time for MTD action				X				
QoS gain/loss of the protected resources				X				

Table 31: Mapping between the INSPIRE-5Gplus KPIs with the WP5 TCs

Finally, Table 32 illustrates the mapping of KPIs with the enablement categories for 5G advancement identified through the gap analysis performed in D2.1.

KPIs	AI & ML	E2E ZSM security management	Security enforcement & control	SSLA assessment and enforcement	Policy and SLA Management	Security Analytics	Secure Data Collection	Trustworthiness
Mean Time To Detect	+	+	+	+		+		+
Mean Time to Contain		+	+					+
Mean Time to Resolve	+	+	+			+		+
Transaction speed	+	+	+	+			+	+
Packet Loss Ratio	+	+	+	+			+	+
Number of False positives	+	+	+	+	+	+		+
Number of False negatives	+	+	+	+	+	+		+
Initial time		+	+	+			+	+
Migration time		+	+	+				+
Service response time	+	+	+	+			+	+
Service downtime	+	+	+	+		+	+	+
SSLA enforcement	+	+	+	+				+
Blocked adversarial examples rate	+		+					
Ratio of allowed malicious scale-up	+		+					
Automated vulnerability assessment					+	+		

KPIs	AI & ML	E2E ZSM security management	Security enforcement & control	SSLA assessment and enforcement	Policy and SLA Management	Security Analytics	Secure Data Collection	Trustworthiness
Automated model generation					+	+		
Threat assessment					+	+		
Cyber-security insights assessment					+	+		
Latency		+						+
Mean Time to implement the MTD action	+	+	+			+		
MTD action cost	+	+	+			+		
Protection gain of an MTD policy	+	+	+			+		
Mean decision time for MTD action	+	+	+			+		
QoS gain/loss of the protected resources	+	+	+			+		

Table 32: KPIs' mapping to enablement categories

5 Preliminary Results

This section aims to report the status of each TC towards the integration of relevant enablers and testbeds as well as preliminary results pertaining to the verification of security components against pre-defined tests for each TC.

5.1 Progress towards integration of enablers in TC1

5.1.1 Enablers' status and availability

As already defined, this TC contains two scenarios and, thus, two different set of enablers to be tested. For each TC1 scenario, the current status and availability of the involved WP3/WP4 is presented in the Table 33.

WP3/WP4 Enablers	TC1 (scenario 1)	TC1 (scenario 2)	Status	Availability
Secure Network Slice Manager for SSLAs	X		Being developed by CTTC.	Final Review Period.
Security Orchestrator	X		Developed by UMU	Currently available in UMU premises on demand.
SSLA Manager			Being developed by TSG.	Final Review Period.
Trusted Blockchain-based Network Slices (TBNS)		X	Testing phase.	Final Review Period.
Component Certification Tool		X	Being developed by TSG.	Final Review Period.

Table 33: Status and availability of the involved WP3/WP4 enablers for TC1

5.1.2 Testbeds' status and functionality

The testbeds and functionality regarding each TC1 scenario are in a different shape at the time of producing this deliverable. While the scenario 1 was started at the beginning of the INSPIRE-5Gplus and two conference articles were presented[12][13] to describe the main idea, its development was paused in order to focus on the second TC1 scenario. This one is far more advanced as some preliminary results were obtained in order to validate the workflows presented in the next subsection. At this stage of the Project progress, our current step is the integration of the Trusted Blockchain-based Network Slices enabler with the Component Certification Tool enabler from TSG.

[12] P. Alemany et al., "Transport Network Slices with Security Service Level Agreements," 2020 22nd International Conference on Transparent Optical Networks (ICTON), 2020, pp. 1-4, doi: 10.1109/ICTON51198.2020.9248696.

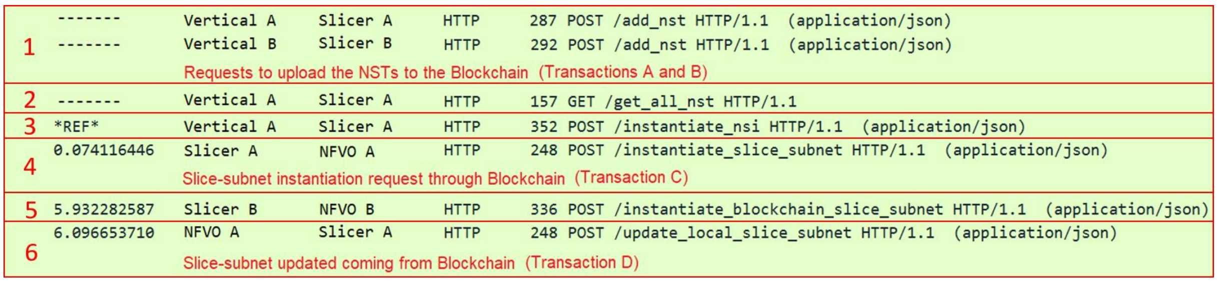
[13] R. Vilalta et al., "Applying Security Service Level Agreements in V2X Network Slices," 2020 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), 2020, pp. 114-115, doi: 10.1109/NFV-SDN50289.2020.9289861.

Finally, regarding the TC1 scenario 1, we have planned to mainly focus on its final development and integration during the last year of the project, where the different enablers involved will be far more developed and ready to be integrated among them.

5.1.3 Preliminary results

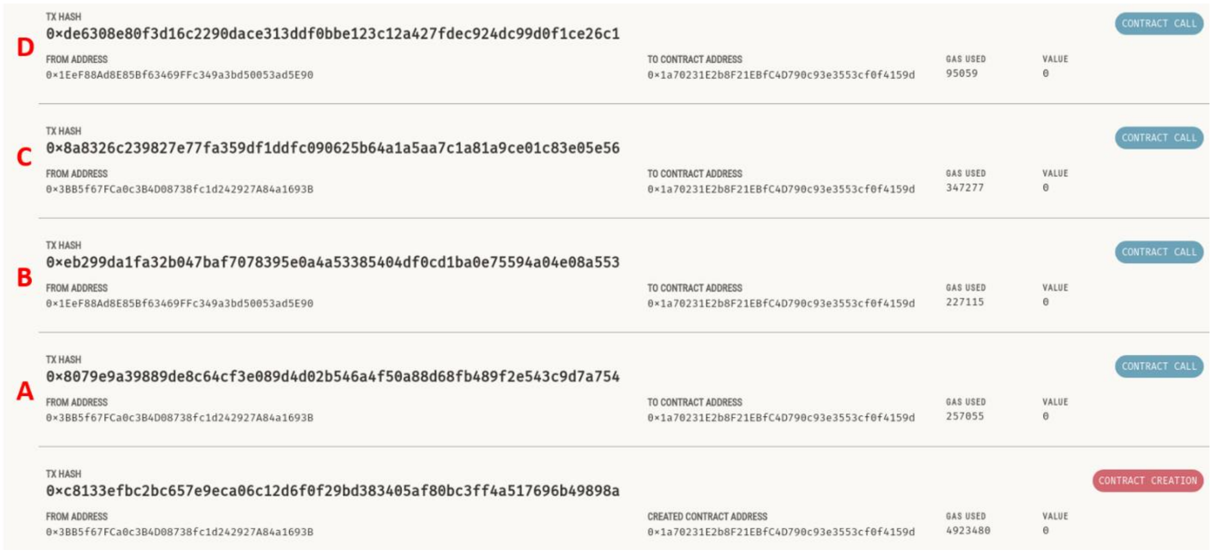
Our preliminary results are focused on the second scenario of TC1 as we intend to focus on scenario 1 during the last period of the INSPIRE-5Gplus project. Figures 49 and 50 present the HTTP traffic and the Ethereum Blockchain transactions respectively, which allows to demonstrate the instantiation procedure presented in Figures 12 and 13.

To achieve these preliminary results, we use an E2E NST composed by two domain network slices (from now on called slice-subnets). So, first all slice-subnets must be added into the local Data-Base (DB) (Figure 49 step 1) of each Slicer and then they are uploaded in the Blockchain (Figure 50 A and B). In that moment, all the domain BSS/OSS have all the NSTs available (Figure 49 step 2). When one of the 5G verticals requests the deployment of an E2E Slice (Figure 49 step 3) to its PDL-Slicing manager (Slicer A in Figure 49), the Slicer A creates the NSI object with its slice-subnets -i.e., selected NSTs- and requests their deployment to corresponding PDL-Slicing domain. It is important to remark that its own slice-subnets are requested (Figure 49 step 4) to its domain NFVO (NFVO A), while the external slice-subnets requests are sent to the corresponding PDL-Slicing manager (i.e., Slicer B) through the Blockchain (Figure 50 C). When Slicer B receives the request to deploy its slice-subnet, it creates an NSI to keep the local track of the computing resources used and requests (Figure 49 step 5) to its local NFVO (NFVO B) the local lice-subnet deployment. Once all the slice-subnets composing the E2E Network Slice are deployed, the E2E Network Slice owner -i.e., Slicer A- is informed directly by its local NFVO A (Figure 49 step 6), or through the Blockchain (Figure 50 D) about the slice-subnets deployments done in other domains.



```
   -------      Vertical A   Slicer A   HTTP   287 POST /add_nst HTTP/1.1  (application/json)
1  -------      Vertical B   Slicer B   HTTP   292 POST /add_nst HTTP/1.1  (application/json)
                Requests to upload the NSTs to the Blockchain  (Transactions A and B)
2  -------      Vertical A   Slicer A   HTTP   157 GET /get_all_nst HTTP/1.1
3  *REF*        Vertical A   Slicer A   HTTP   352 POST /instantiate_nsi HTTP/1.1  (application/json)
   0.074116446  Slicer A     NFVO A     HTTP   248 POST /instantiate_slice_subnet HTTP/1.1  (application/json)
4               Slice-subnet instantiation request through Blockchain  (Transaction C)
5  5.932282587  Slicer B     NFVO B     HTTP   336 POST /instantiate_blockchain_slice_subnet HTTP/1.1  (application/json)
   6.096653710  NFVO A       Slicer A   HTTP   248 POST /update_local_slice_subnet HTTP/1.1  (application/json)
6               Slice-subnet updated coming from Blockchain  (Transaction D)
```

Figure 49: Wireshark deployment steps



```
  TX HASH                                                                                                                  CONTRACT CALL
D 0×de6308e80f3d16c2290dace313ddf0bbe123c12a427fdec924dc99d0f1ce26c1
  FROM ADDRESS                                  TO CONTRACT ADDRESS                           GAS USED   VALUE
  0×1EeF88Ad8E85Bf63469FFc349a3bd50053ad5E90    0×1a70231E2b8F21EBfC4D790c93e3553cf0f4159d    95059      0

  TX HASH                                                                                                                  CONTRACT CALL
C 0×8a8326c239827e77fa359df1ddfc090625b64a1a5aa7c1a81a9ce01c83e05e56
  FROM ADDRESS                                  TO CONTRACT ADDRESS                           GAS USED   VALUE
  0×3BB5f67FCa0c3B4D08738fc1d242927A84a1693B    0×1a70231E2b8F21EBfC4D790c93e3553cf0f4159d    347277     0

  TX HASH                                                                                                                  CONTRACT CALL
B 0×eb299da1fa32b047baf7078395e0a4a53385404df0cd1ba0e75594a04e08a553
  FROM ADDRESS                                  TO CONTRACT ADDRESS                           GAS USED   VALUE
  0×1EeF88Ad8E85Bf63469FFc349a3bd50053ad5E90    0×1a70231E2b8F21EBfC4D790c93e3553cf0f4159d    227115     0

  TX HASH                                                                                                                  CONTRACT CALL
A 0×8079e9a39889de8c64cf3e089d4d02b546a4f50a88d68fb489f2e543c9d7a754
  FROM ADDRESS                                  TO CONTRACT ADDRESS                           GAS USED   VALUE
  0×3BB5f67FCa0c3B4D08738fc1d242927A84a1693B    0×1a70231E2b8F21EBfC4D790c93e3553cf0f4159d    257055     0

  TX HASH                                                                                                                  CONTRACT CREATION
  0×c8133efbc2bc657e9eca06c12d6f0f29bd383405af80bc3ff4a517696b49898a
  FROM ADDRESS                                  CREATED CONTRACT ADDRESS                      GAS USED   VALUE
  0×3BB5f67FCa0c3B4D08738fc1d242927A84a1693B    0×1a70231E2b8F21EBfC4D790c93e3553cf0f4159d    4923480    0
```

Figure 50: Blockchain logs with the information distribution (transactions)

5.2 Progress towards integration of enablers in TC2

5.2.1 Enablers' status and availability

Table 34 shows the current status and availability of the enablers involved in the TC2.

Enabler	Status	Availability
Policy Framework	Including and validating new plugins to translate MSPL policies to INSPIRE-5Gplus security enablers.	Currently it is available in UMU premises on demand
Security Orchestrator	Developed by UMU	Currently it is available in UMU premises on demand
Security Agent	Developed by MI and adapted to be run on different testbeds	Currently it is available in MI premises on demand
HSPL to MSPL converter	Developed by UMU, could add/extend fields in MSPL	Currently it is available in UMU premises on demand
Security Analytic Engine	Under development to determine, from the MSPLs, what RT-SSLA rules need to be used by the probes and analytics engine	Currently it is available in MI premises on demand
Decision Engine	Under development	Not available yet

Table 34: Status and availability of the involved WP3/WP4 enablers for TC2

5.2.2 Testbeds' status and functionality

Some enablers involved in the TC2 have been developed and tested in UMU and MI premises. The interactions between those principal components, focusing on proactive and reactive traffic monitoring in 5G, have been deployed and tested in the CTTC premises to show the closed-loop during the mid-term review. The integration of some other functionalities in the TC2 are still under development to perform more complex scenarios.

5.2.3 Preliminary results

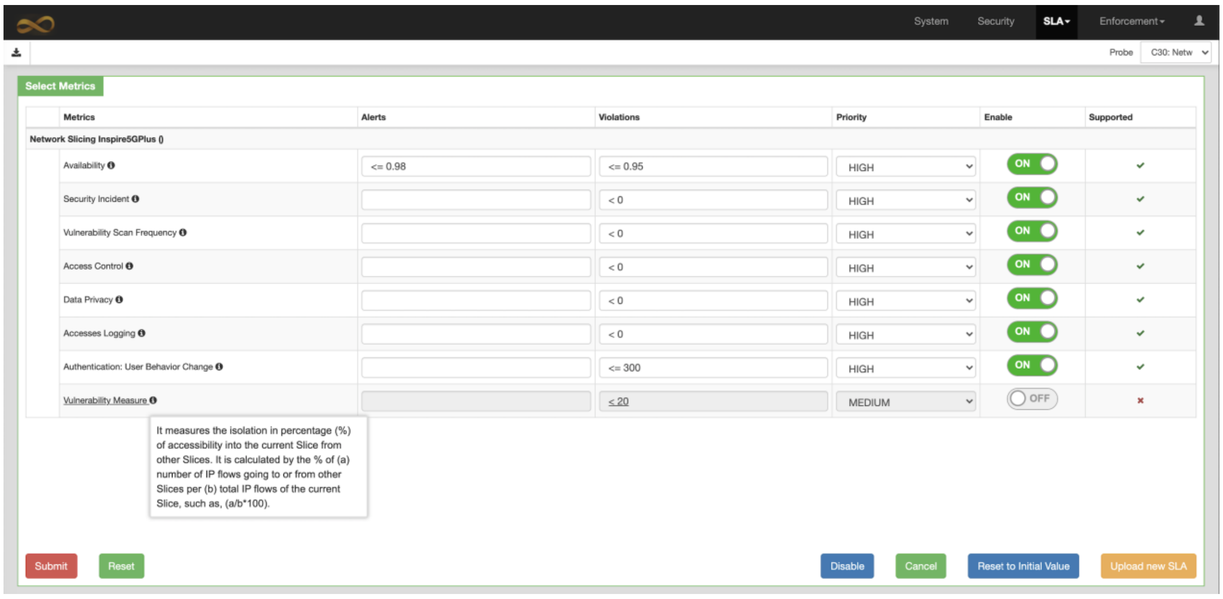
Our preliminary results are a collection of SSLAs on network slicing, defined in XML format. The SSLAs contain key information, such as metrics, security rules, parameters and threshold values. The MSPLs can serve to determine what SSLAs rules need to be used in the assessment and enforcement processes.

We predefined RT-SSLA rules template regarding the type of policy, then automatically generate rules by using key information extracted from MSPL. For example, given a filtering policy, the Boolean expression of the rule template could be defined as "((PROTOCOL.packet_count > 0) && ((ip.src != ip.dst) && (ethernet.src != ethernet.dst)))", where PROTOCOL value (e.g., Bittorrent, Facebook) could be extracted from the following MSPL.

```
...
<monitoringConfigurationCondition>
```

```
  <isCNF>false</isCNF>
  <packetFilterCondition>
    <DestinationPort>9999</DestinationPort>
    <ProtocolType>Bittorrent</ProtocolType>
  </packetFilterCondition>
</monitoringConfigurationCondition>
…
```

Figure 51 shows the dashboard of the Security Analytics Engine (i.e., the MMT-Operator). We can enable or disable security properties in the list so that the MMT-Probes will monitor the corresponding components to obtain the necessary metadata. When an alert is raised due to a threshold violation, the Security Analytics Engine will send this alert to the Decision Engine so that it can decide and perform any appropriate reactive actions.



Metrics	Alerts	Violations	Priority	Enable	Supported
Network Slicing Inspire5GPlus ()					
Availability	<= 0.98	<= 0.95	HIGH	ON	✓
Security Incident		< 0	HIGH	ON	✓
Vulnerability Scan Frequency		< 0	HIGH	ON	✓
Access Control		< 0	HIGH	ON	✓
Data Privacy		< 0	HIGH	ON	✓
Accesses Logging		< 0	HIGH	ON	✓
Authentication: User Behavior Change		<= 300	HIGH	ON	✓
Vulnerability Measure		≤ 20	MEDIUM	OFF	✗

It measures the isolation in percentage (%) of accessibility into the current Slice from other Slices. It is calculated by the % of (a) number of IP flows going to or from other Slices per (b) total IP flows of the current Slice, such as, (a/b*100).

Figure 51: Security properties extracted from SSLAs

5.3 Progress towards integration of enablers in TC3/TC4

5.3.1 Enablers' status and availability

Table 35 shows the current status and availability of the enablers involved in the TC3/TC4 merge.

Enabler	Status	Availability
E2E Security Orchestrator	Developed from scratch. Current implementation allows enforcing high-level policies in different management domains.	Currently it is available in UMU premises on demand. It will be released by the final review date.
Security Orchestrator	Current implementation is focused on extending orchestration features to consider trust.	Currently it is available in UMU premises on demand. It will be released by the final review date.

Enabler	Status	Availability
Policy Framework	Including and validating new plugins to translate MSPL policies to INSPIRE-5Gplus security enablers.	Currently it is available in UMU premises on demand. It will be released by the final review date.
Security Analytics Engine		Currently it is available in MI premises on demand
MMT Probe		Currently it is available in MI premises on demand
Systemic binary wrapper	Was pre-existing the Project but has been improved on several axis. The main progress brought are Intel SGX automatic leverage, a new advanced code confidentiality mode called Control Flow Shadowing and two SECaaS deployment flavors (container based and single binary)	SYSTEMIC-SGX is available, the current work is the completion of the user's documentation.
Trust Reputation Manager	Developed from scratch. Integration end points for the previously defined APIs have been developed as well as publication/subscription modules to ease asynchronous communications.	Currently it is available in UMU premises on demand. It will be released by the final review date.
PoT Controller	Developed version optimized for INSPIRE-5Gplus. The current version includes support for integration REST API and Kafka bus.	Currently it is available in TID premises on demand
I2NSF Controller	Enabler available and integrated with agent.	Currently it is available in TID premises on demand
Smart Traffic Analyser	Developed, operational virtualised software embedded in a virtual machine image.	Currently it is available in TID premises on demand
PoT/I2NSF agent	Developed version from scratch. The current version provides an isolated agent solution for PoT or I2NSF. Pending integration on shared agent	Currently it is available in TID premises on demand

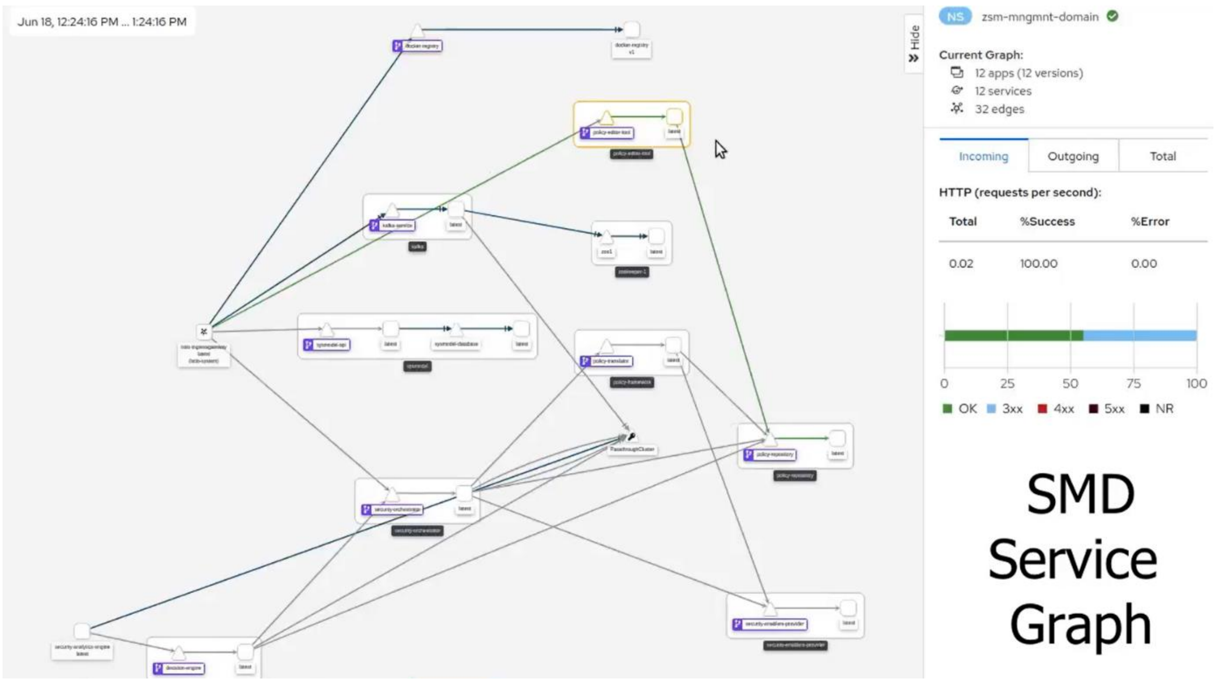
Table 35: Status and availability of the involved WP3/WP4 enablers for TC3/TC4

5.3.2 Testbeds' status and functionality

The tests have been performed in premises as well as in the collaborative platform provided as part of the WP5 activities. In fact, the majority of the components that compose TC3/TC4 have been deployed in different domains and they have been also tested not only for the test case but also during the mid-term review to show the INSPIRE-5Gplus closed-loop. Thus, the testbed is ready to manage a subset of proactive and reactive policies enforcement. In the current status, the testbed is able to receive high-level policies at E2E SMD and enforce them in the required SMD. Current tests have been focused on proactive 5G traffic monitoring and reactive traffic management features such as filtering and traffic divert operations.

5.3.3 Preliminary results

As preliminary results, the first implementation of the INSPIRE-5Gplus security enablers involved in this test case were deployed as part of the first HLA deployment, in the collaborative testbed defined in Section 2.1.1. To validate the security enablers in the multi-domain scenario, an E2E Security Management Domain and a Security Management Domain were instantiated. In fact, the SMD was deployed in two physical locations, UMU and MI whereas the E2E-SMD was deployed in CTTC. Figure 52 shows the SDM service graph deployed between UMU and MI premises (control plane in UMU and data plate in MI). The SMD deployment contains the security orchestrator, policy framework, security analytics engine, decision engine API and data services. The connectivity is provided by the integration fabric that is implemented by using Kafka and istio over Kubernetes. Regarding the E2E-SMD, it was instantiated with an E2E-Security Orchestrator, e2e data services, e2e policy framework and an e2e decision engine API. Figure 53 shows the E2E SMD instantiation in CTTC premises. By using this first implementation and deployment, E2E High-level security policies were enforced in order to configure monitoring tools to detect specific traffic profiles. The E2E Security Orchestrator managed to identify the domains where the monitoring policies must be enforced to. Then, medium-level security policies were automatically generated and enforced in the required SMD. The SMD enforced the received security policies by configuring a monitoring tool to accomplish with the new security policies. The new security policy enforced in the system can be seen in Figure 55 as "Network_traffic_analysis".



Total	%Success	%Error
0.02	100.00	0.00

Figure 52: SMD instantiation

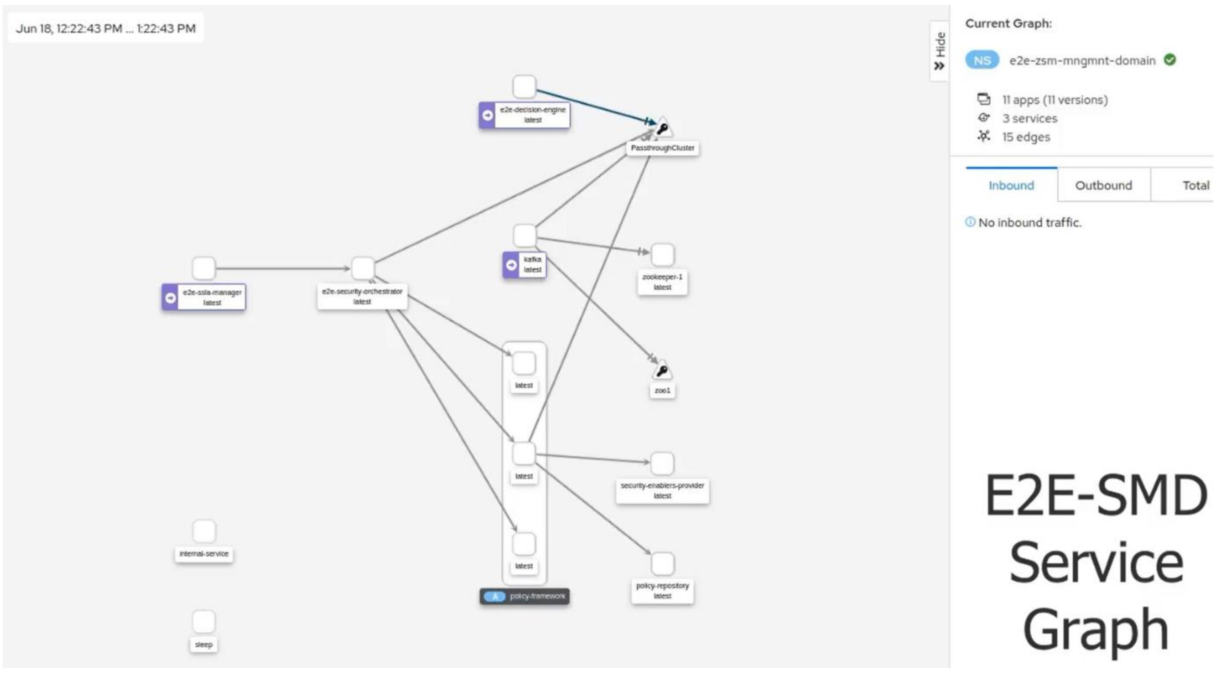
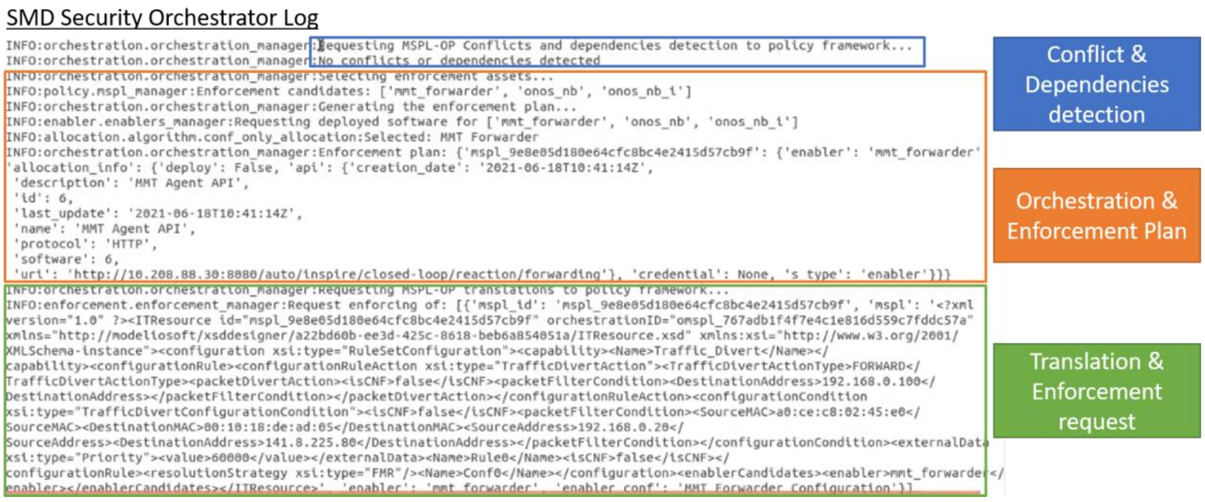


Figure 53: E2E SMD instantiation

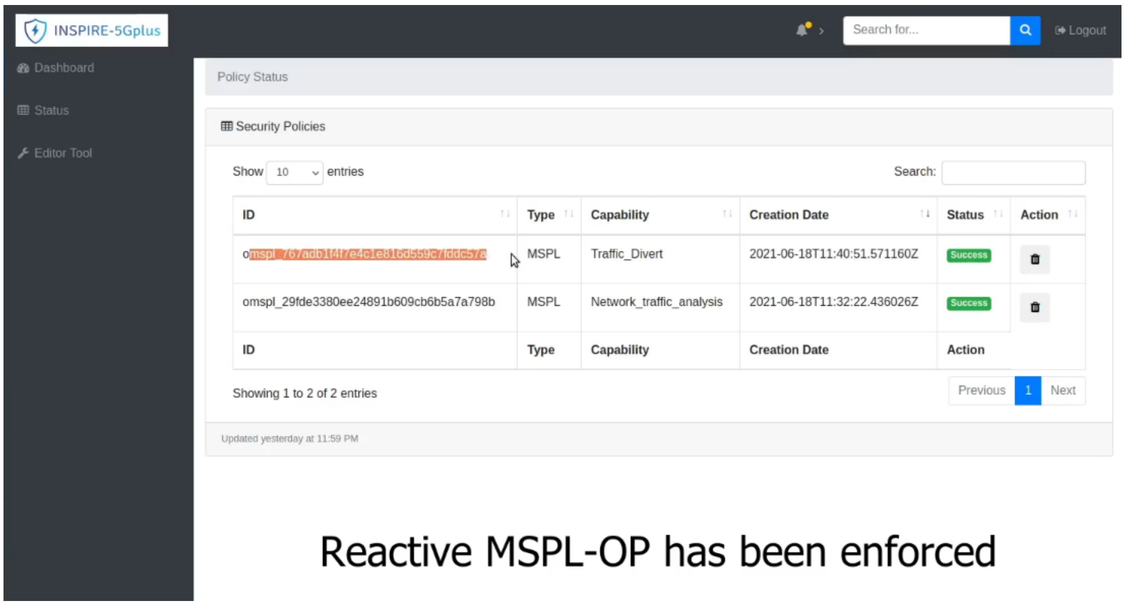
To validate the new expected behaviour, undesired traffic was injected in the 5G network. Then it was successfully detected by the previously configured monitoring tool according to the proactive policies. In fact, the scenario was validated for two different monitoring tools. Snort for simply monitoring policies and MMT for more complex behaviours that feed the security analytics engine. Then, a first version of the decision engine received an alert from the later. Then, the decision engine generated a countermeasure in form of a forwarding medium-level security policy. Finally, it requested the reactive SMD policy enforcement while at the same time it notified to the E2E SMD. Figure 54 shows the SMD Security Orchestrator log once it received the reactive security policy. The figure shows the different stages of the process that included conflicts and dependencies detection (no conflicts and dependencies were detected in this case), orchestration and enforcement plan (that for this test the MMT_forwarder was selected to manage the forwarding task) and finally the translation and enforcement request. The security policy was translated into MMT_forwarder configurations that were enforced into the MMT_forwarder tool. Once the reactive security policy was enforced, the new status of the enforced policies can be verified in the policy GUI. In this case, Figure 55 also shows the new reactive "Traffic_Divert" policy marked as enforced.


SMD Security Orchestrator Log

```
INFO:orchestration.orchestration_manager:Requesting MSPL-OP Conflicts and dependencies detection to policy framework...
INFO:orchestration.orchestration_manager:No conflicts or dependencies detected
INFO:orchestration.orchestration_manager:Selecting enforcement assets...
INFO:policy.mspl_manager:Enforcement candidates: ['mmt_forwarder', 'onos_nb', 'onos_nb_i']
INFO:orchestration.orchestration_manager:Generating the enforcement plan...
INFO:enabler.enablers_manager:Requesting deployed software for ['mmt_forwarder', 'onos_nb', 'onos_nb_i']
INFO:allocation.algorithm.conf_only_allocation:Selected: MMT Forwarder
INFO:orchestration.orchestration_manager:Enforcement plan: {'mspl_9e8e05d180e64cfc8bc4e2415d57cb9f': {'enabler': 'mmt_forwarder'
'allocation_info': {'deploy': False, 'api': {'creation_date': '2021-06-18T10:41:14Z',
 'description': 'MMT Agent API',
 'id': 6,
 'last_update': '2021-06-18T10:41:14Z',
 'name': 'MMT Agent API',
 'protocol': 'HTTP',
 'software': 6,
 'uri': 'http://10.208.88.30:8080/auto/inspire/closed-loop/reaction/forwarding'}, 'credential': None, 's_type': 'enabler'}}}
INFO:orchestration.orchestration_manager:Requesting MSPL-OP translations to policy framework...
INFO:enforcement.enforcement_manager:Request enforcing of: [{'mspl_id': 'mspl_9e8e05d180e64cfc8bc4e2415d57cb9f', 'mspl': '<?xml
version="1.0" ?><ITResource id="mspl_9e8e05d180e64cfc8bc4e2415d57cb9f" orchestrationID="omspl_767adb1f4f7e4c1e816d559c7fddc57a"
xmlns="http://modeliosoft/xsddesigner/a22bd60b-ee3d-425c-8618-beb6a854051a/ITResource.xsd" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance"><configuration xsi:type="RuleSetConfiguration"><capability><Name>Traffic_Divert</Name></
capability><configurationRule><configurationRuleAction xsi:type="TrafficDivertAction"><TrafficDivertActionType>FORWARD</
TrafficDivertActionType><packetDivertAction><isCNF>false</isCNF><packetFilterCondition><DestinationAddress>192.168.0.100</
DestinationAddress></packetFilterCondition></packetDivertAction></configurationRuleAction><configurationCondition
xsi:type="TrafficDivertConfigurationCondition"><isCNF>false</isCNF><packetFilterCondition><SourceMAC>a0:ce:c8:02:45:e0</
SourceMAC><DestinationMAC>00:10:18:de:ad:05</DestinationMAC><SourceAddress>192.168.0.20</
SourceAddress><DestinationAddress>141.8.225.80</DestinationAddress></packetFilterCondition></configurationCondition><externalData
xsi:type="Priority"><value>60000</value></externalData><Name>Rule0</Name><isCNF>false</isCNF></
configurationRule><resolutionStrategy xsi:type="FMR"/><Name>Conf0</Name></configuration><enablerCandidates><enabler>mmt_forwarder</
enabler></enablerCandidates></ITResource>', 'enabler': 'mmt_forwarder', 'enabler_conf': 'MMT Forwarder Configuration'}]
```

Conflict & Dependencies detection
Orchestration & Enforcement Plan
Translation & Enforcement request

Figure 54: SMD Security Orchestrator process



ID	Type	Capability	Creation Date	Status	Action
omspl_767adb1f4f7e4c1e816d559c7fddc57a	MSPL	Traffic_Divert	2021-06-18T11:40:51.571160Z	Success	🗑
omspl_29fde3380ee24891b609cb6b5a7a798b	MSPL	Network_traffic_analysis	2021-06-18T11:32:22.436026Z	Success	🗑
ID	Type	Capability	Creation Date	Action	

Showing 1 to 2 of 2 entries

Updated yesterday at 11:59 PM

Reactive MSPL-OP has been enforced

Figure 55: INSPIRE-5Gplus policy GUI

5.4 Progress towards integration of enablers in TC5

5.4.1 Enablers' status and availability

Table 36 shows the current status and availability of the enablers involved in the TC5.

Enabler	Status	Availability
Moving Target Defense Controller (MOTDEC)	Under development at ZHAW premises, a first version is currently being integrated with the slice manager to enforce MTD actions.	Not available yet.
Optimization of Security Functions (OptSFC)	Under development at ZHAW premises	Not available yet.
Katana Slice Manager	Developed from scratch by NCSRD in the 5GENESIS project. It is currently extended during INSPIRE-5Gplus to support dynamic slice reconfiguration MTD actions forwarded by MOTDEC	Available. Extensions are under development.
Anomaly Detection Framework (ADF)	Developed by NCSRD	Under development
Montimage Monitoring Framework (MMT)	Developed by MI	Available in NCSRD premises.

Table 36: Status and availability of the involved WP3/WP4 enablers for TC5

5.4.2 Testbeds' status and functionality

The testbed is hosted on the NCSRD premises on top of the 5GENESIS 5G infrastructure. The testbed features a variety of 5G SA and NSA deployment setups along with an extensive NFV infrastructure which are already in place. For the needs of the TC5 scenario, that features emulated attacks on the network edge (e.g., edge UPF), it was decided to deploy Open5GS[14] as well, in order to support UPF distributed placement, which was not possible with existing products.

In addition, since the TC is highly dependent on extensive data collection, the limited number of physical 5G COTS UEs may impose an issue on investigating the capabilities of the underlying ML algorithms featuring in the participating enablers. Thus, as an intermediary step before a final demonstration on the actual 5G infrastructure, it was decided to integrate virtualized gNBs and COTS UEs based on the UERANSIM[15] solution, in order to test a variety of scenarios that is not possible to implement with real infrastructure (e.g., extensive number of connected COTS UEs). So, at the moment we have installed both solutions on the infrastructure and we are currently extending the Katana Slice Manager and the Element Management System (EMS) to support these, as well. Of course, we are also planning a small-scale demonstration on the actual infrastructure after the intermediary step is completed.

We have also deployed the Montimage Monitoring Framework (MMFT) on our infrastructure, and we are able to collect data from multiple points of interest, providing integrated monitoring capabilities in our scenario. We are also working on developing the ADF, where our main focus was the underlying ML algorithm for cost effective anomaly detection and classification. At the moment, the integration step along with the other enablers (MMT and OptSFC) is under development. In addition, we have provided a dedicated environment for integrating the MOTDEC solution provided by ZHAW, in order to test and validate its interactions with the Katana Slice Manager.

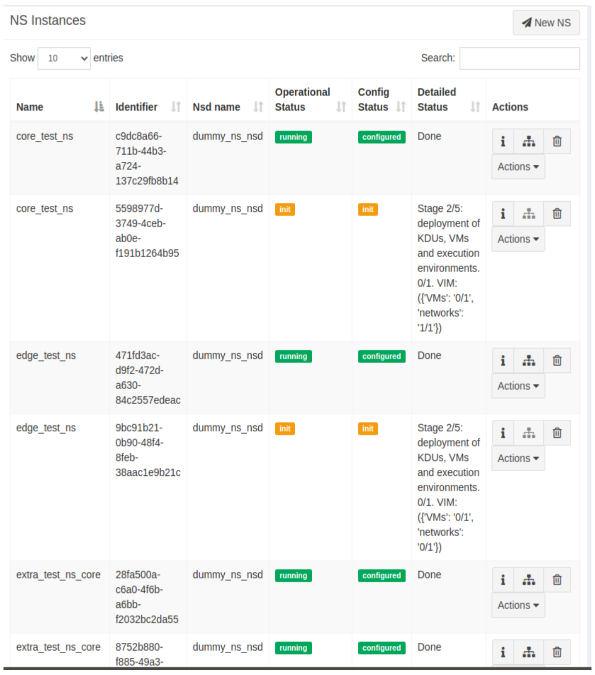
5.4.3 Preliminary results

The first MTD action implemented in MOTDEC allow to re-initiate the services composing a network slice. The operation is performed in two phases, depicted in Figures 56 and 57: in phase 1 new instances of the NSs are initiated following the original slice template, while the old instances stay employed. Phase 2 starts when the new instances are operational, then the old ones get decommissioned. The operation mitigates attacks that involves the compromise of a virtual computational unit of the service. In such scenarios, attackers have a wide range of choices such as eavesdropping, ransomware encryption, DoS and C&C.

The MTD restart action associated costs are the allocation of additional resources on the Virtual Infrastructure Manager (VIM) during the entire operation. However, the overhead on the QoE of end-users is limited only by the SDN-based traffic redirection after phase 1 is completed. The second MTD action implemented is the migration of specific services of the slice in a different infrastructure. The PoC (depicted in Figure 58) demonstrates the migration of a service in the edge node set back to the core infrastructure. The relocation of the service follows the same substitution technique used on the 'restart' MTD action, to preclude to a lengthy disruption the connection of the UEs. This operation is ideal when the infrastructure hosting the services get compromised, and restarting the service, e.g., the user plane function (UPF, a 5G core service) with the first MTD action does not fix the issue.

[14] https://open5gs.org/

[15] https://github.com/aligungr/UERANSIM



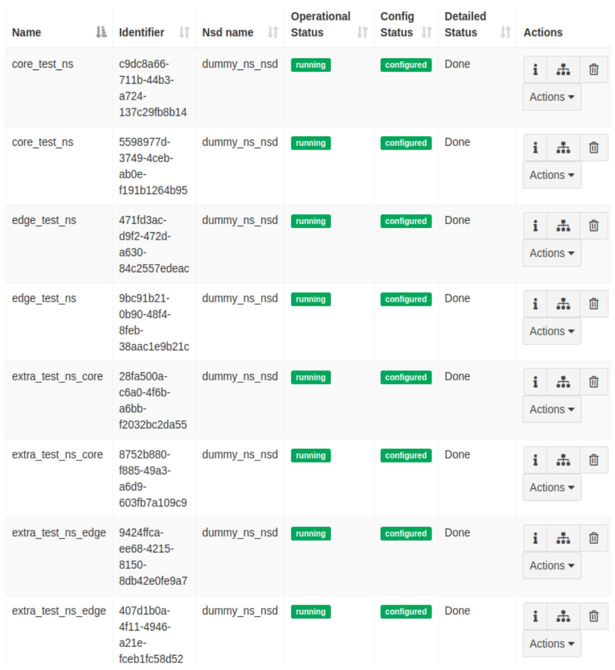
NS Instances

New NS

Show 10 entries

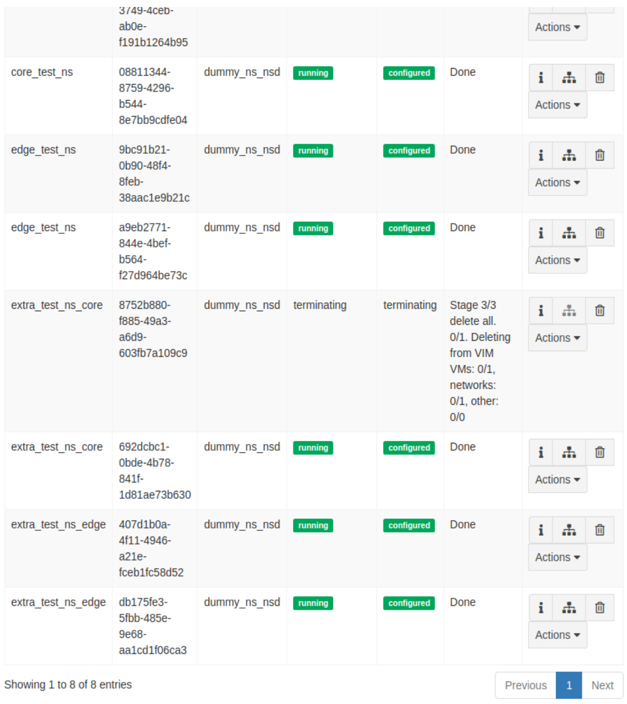
Search:

Name	Identifier	Nsd name	Operational Status	Config Status	Detailed Status	Actions
core_test_ns	c9dc8a66-711b-44b3-a724-137c29fb8b14	dummy_ns_nsd	running	configured	Done	Actions
core_test_ns	5598977d-3749-4ceb-ab0e-f191b1264b95	dummy_ns_nsd	init	init	Stage 2/5: deployment of KDUs, VMs and execution environments. 0/1. VIM: ({'VMs': '0/1', 'networks': '1/1'})	Actions
edge_test_ns	471fd3ac-d9f2-472d-a630-84c2557edeac	dummy_ns_nsd	running	configured	Done	Actions
edge_test_ns	9bc91b21-0b90-48f4-8feb-38aac1e9b21c	dummy_ns_nsd	init	init	Stage 2/5: deployment of KDUs, VMs and execution environments. 0/1. VIM: ({'VMs': '0/1', 'networks': '0/1'})	Actions
extra_test_ns_core	28fa500a-c6a0-4f6b-a6bb-f2032bc2da55	dummy_ns_nsd	running	configured	Done	Actions
extra_test_ns_core	8752b880-f885-49a3-	dummy_ns_nsd	running	configured	Done	



Name	Identifier	Nsd name	Operational Status	Config Status	Detailed Status	Actions
core_test_ns	c9dc8a66-711b-44b3-a724-137c29fb8b14	dummy_ns_nsd	running	configured	Done	Actions
core_test_ns	5598977d-3749-4ceb-ab0e-f191b1264b95	dummy_ns_nsd	running	configured	Done	Actions
edge_test_ns	471fd3ac-d9f2-472d-a630-84c2557edeac	dummy_ns_nsd	running	configured	Done	Actions
edge_test_ns	9bc91b21-0b90-48f4-8feb-38aac1e9b21c	dummy_ns_nsd	running	configured	Done	Actions
extra_test_ns_core	28fa500a-c6a0-4f6b-a6bb-f2032bc2da55	dummy_ns_nsd	running	configured	Done	Actions
extra_test_ns_core	8752b880-f885-49a3-a6d9-603fb7a109c9	dummy_ns_nsd	running	configured	Done	Actions
extra_test_ns_edge	9424ffca-ee68-4215-8150-8db42e0fe9a7	dummy_ns_nsd	running	configured	Done	Actions
extra_test_ns_edge	407d1b0a-4f11-4946-a21e-fceb1fc58d52	dummy_ns_nsd	running	configured	Done	Actions

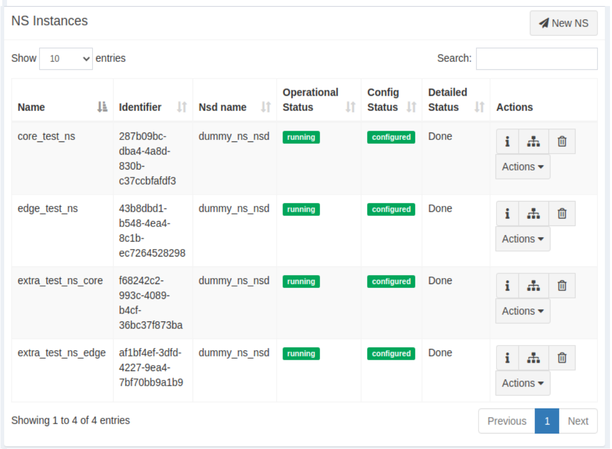
Figure 56: MTD slice re-instantiation phase 1



	3749-4ceb-ab0e-f191b1264b95					Actions
core_test_ns	08811344-8759-4296-b544-8e7bb9cdfe04	dummy_ns_nsd	running	configured	Done	Actions
edge_test_ns	9bc91b21-0b90-48f4-8feb-38aac1e9b21c	dummy_ns_nsd	running	configured	Done	Actions
edge_test_ns	a9eb2771-844e-4bef-b564-f27d964be73c	dummy_ns_nsd	running	configured	Done	Actions
extra_test_ns_core	8752b880-f885-49a3-a6d9-603fb7a109c9	dummy_ns_nsd	terminating	terminating	Stage 3/3 delete all. 0/1. Deleting from VIM VMs: 0/1, networks: 0/1, other: 0/0	Actions
extra_test_ns_core	692dcbc1-0bde-4b78-841f-1d81ae73b630	dummy_ns_nsd	running	configured	Done	Actions
extra_test_ns_edge	407d1b0a-4f11-4946-a21e-fceb1fc58d52	dummy_ns_nsd	running	configured	Done	Actions
extra_test_ns_edge	db175fe3-5fbb-485e-9e68-aa1cd1f06ca3	dummy_ns_nsd	running	configured	Done	Actions

Showing 1 to 8 of 8 entries

Previous 1 Next



NS Instances

New NS

Show 10 entries

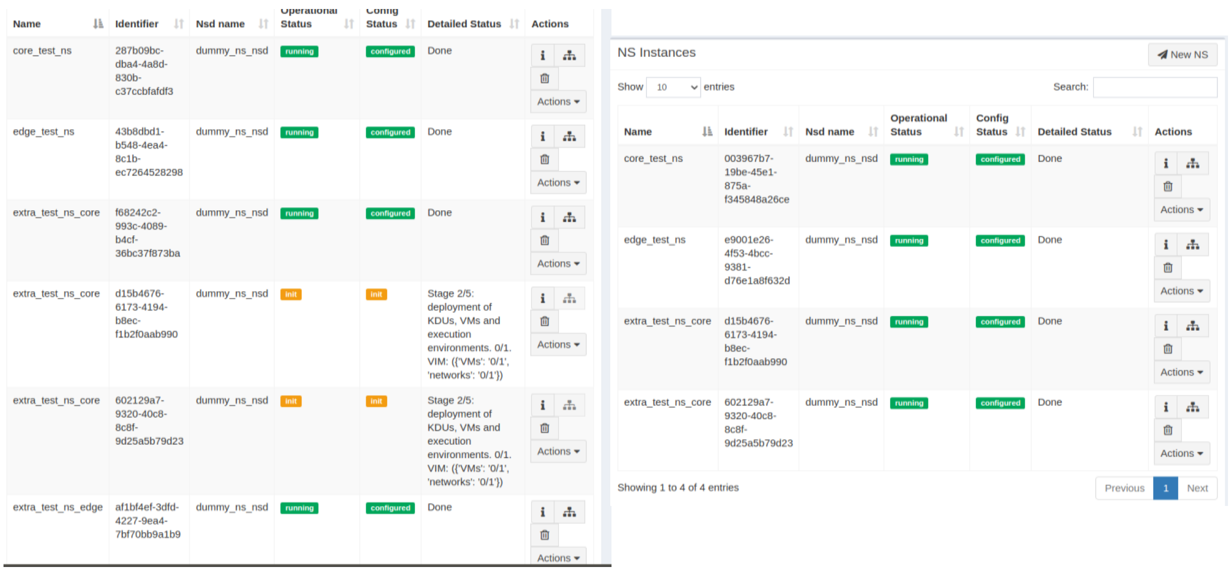
Search:

Name	Identifier	Nsd name	Operational Status	Config Status	Detailed Status	Actions
core_test_ns	287b09bc-dba4-4a8d-830b-c37ccbfafdf3	dummy_ns_nsd	running	configured	Done	Actions
edge_test_ns	43b8dbd1-b548-4ea4-8c1b-ec7264528298	dummy_ns_nsd	running	configured	Done	Actions
extra_test_ns_core	f68242c2-993c-4089-b4cf-36bc37f873ba	dummy_ns_nsd	running	configured	Done	Actions
extra_test_ns_edge	af1bf4ef-3dfd-4227-9ea4-7bf70bb9a1b9	dummy_ns_nsd	running	configured	Done	Actions

Showing 1 to 4 of 4 entries

Previous 1 Next

Figure 57: MTD slice re-instantiation phase 2



Name	Identifier	Nsd name	Operational Status	Config Status	Detailed Status	Actions
core_test_ns	287b09bc-dba4-4a8d-830b-c37ccbfafdf3	dummy_ns_nsd	running	configured	Done	Actions
edge_test_ns	43b8dbd1-b548-4ea4-8c1b-ec7264528298	dummy_ns_nsd	running	configured	Done	Actions
extra_test_ns_core	f68242c2-993c-4089-b4cf-36bc37f873ba	dummy_ns_nsd	running	configured	Done	Actions
extra_test_ns_core	d15b4676-6173-4194-b8ec-f1b2f0aab990	dummy_ns_nsd	init	init	Stage 2/5: deployment of KDUs, VMs and execution environments. 0/1. VIM: ({'VMs': '0/1', 'networks': '0/1'})	Actions
extra_test_ns_core	602129a7-9320-40c8-8c8f-9d25a5b79d23	dummy_ns_nsd	init	init	Stage 2/5: deployment of KDUs, VMs and execution environments. 0/1. VIM: ({'VMs': '0/1', 'networks': '0/1'})	Actions
extra_test_ns_edge	af1bf4ef-3dfd-4227-9ea4-7bf70bb9a1b9	dummy_ns_nsd	running	configured	Done	Actions

NS Instances

New NS

Show 10 entries Search:

Name	Identifier	Nsd name	Operational Status	Config Status	Detailed Status	Actions
core_test_ns	003967b7-19be-45e1-875a-f345848a26ce	dummy_ns_nsd	running	configured	Done	Actions
edge_test_ns	e9001e26-4f53-4bcc-9381-d76e1a8f632d	dummy_ns_nsd	running	configured	Done	Actions
extra_test_ns_core	d15b4676-6173-4194-b8ec-f1b2f0aab990	dummy_ns_nsd	running	configured	Done	Actions
extra_test_ns_core	602129a7-9320-40c8-8c8f-9d25a5b79d23	dummy_ns_nsd	running	configured	Done	Actions

Showing 1 to 4 of 4 entries

Previous 1 Next

Figure 58: MTD sub-service migration

5.5 Progress towards integration of enablers in TC6

5.5.1 Enablers' status and availability

Table 37 shows the current status and availability of the enablers involved in the TC6.

Enabler	Status	Availability
E2E Security Orchestrator	Developed from scratch. Current implementation allows enforcing high-level policies in different management domains.	Currently it is available in UMU premises on demand.
Security Orchestrator	Current implementation is focused on extending orchestration features to consider trust.	Currently it is available in UMU premises on demand.
Policy Framework	Currently it is available in UMU premises on demand.	
E2E Security Analytics Engine	Under development	
E2E MUD	Under development	
E2E Trust Reputation Manager	Under development	
E2E PF	Currently it is available in UMU premises on demand.	
Security Analytics Engine	Currently it is available in MI premises on demand	
Trust Reputation Manager	Under development	
OBU Manager	Under development	
Virtual Channel Protection (DTLS Proxy)	Under development	

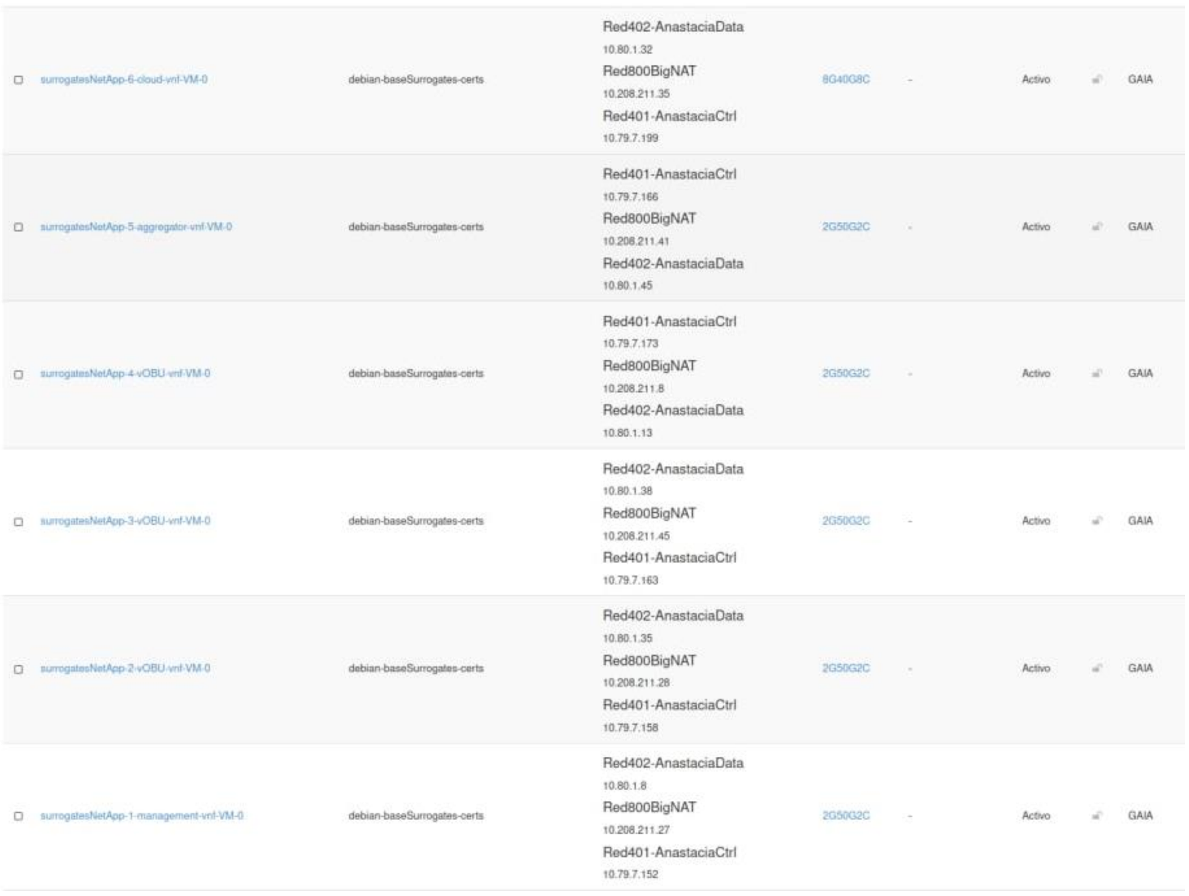
Table 37: Status and availability of the involved WP3/WP4 enablers for TC6

5.5.2 Testbeds' status and functionality

Some of the components that compose TC6 have been deployed in different domains and they have been tested during the mid-term review to show INSPIRE-5Gplus closed-loop. In the current testbed, the enablers that form part of the initial closed-loop are functional and capable of receiving HSPLs at E2E Domain and enforce the security policies in the required SMDs, while MUD and TRM integration is still in a preliminary state and current efforts relay on the policy translations into specific Security Asset configurations (SDN Controller and OBU Manager).

5.5.3 Preliminary results

Part of the scenario is deployed on OpenStack where vOBUs are deployed and registered in the OBU Manager. When the OBU requests a vOBU the Manager assign one vOBU of the pool to the physical OBU. Then, the assigned vOBU receives data from the OBU and it sends it to the DB Aggregator which stores the information. Currently, there are three vOBUs deployed in the scenario and the OBU Manager which is already deployed is in charge of assigning the required vOBU to the OBU, and managing the registration process of the vOBUs.



surrogatesNetApp-6-cloud-vnf-VM-0	debian-baseSurrogates-certs	Red402-AnastaciaData 10.80.1.32 Red800BigNAT 10.208.211.35 Red401-AnastaciaCtrl 10.79.7.199	8G40G8C	-	Activo		GAIA
surrogatesNetApp-5-aggregator-vnf-VM-0	debian-baseSurrogates-certs	Red401-AnastaciaCtrl 10.79.7.166 Red800BigNAT 10.208.211.41 Red402-AnastaciaData 10.80.1.45	2G50G2C	-	Activo		GAIA
surrogatesNetApp-4-vOBU-vnf-VM-0	debian-baseSurrogates-certs	Red401-AnastaciaCtrl 10.79.7.173 Red800BigNAT 10.208.211.8 Red402-AnastaciaData 10.80.1.13	2G50G2C	-	Activo		GAIA
surrogatesNetApp-3-vOBU-vnf-VM-0	debian-baseSurrogates-certs	Red402-AnastaciaData 10.80.1.38 Red800BigNAT 10.208.211.45 Red401-AnastaciaCtrl 10.79.7.163	2G50G2C	-	Activo		GAIA
surrogatesNetApp-2-vOBU-vnf-VM-0	debian-baseSurrogates-certs	Red402-AnastaciaData 10.80.1.35 Red800BigNAT 10.208.211.28 Red401-AnastaciaCtrl 10.79.7.158	2G50G2C	-	Activo		GAIA
surrogatesNetApp-1-management-vnf-VM-0	debian-baseSurrogates-certs	Red402-AnastaciaData 10.80.1.8 Red800BigNAT 10.208.211.27 Red401-AnastaciaCtrl 10.79.7.152	2G50G2C	-	Activo		GAIA

Figure 59: Scenario deployment for TC6

In Figure 59, we can observe the three vOBUs, the OBU Manager and the aggregator DB service running on OpenStack.

5.6 Progress towards integration of enablers in TC7

5.6.1 Enablers' status and availability

Table 38 shows the current status and availability of the enablers involved in the TC7.

Enabler	Status	Availability
Security Monitoring Framework (MMT probe)	Developed MMT-Probe as Prometheus exporter to provide network statistics	Currently it is available in MI premises on demand
Auto-scaling Module (Admission Controller Delegator)	Developed	Currently available
Damage Controller (DDoS Mitigator)	Under development. Current implementation aims at building an anomaly detection model using Recurrent Neural Networks (RNNs). The model can detect DDoS attack based on anomalies in resource usage and performance metrics of the VNFs composing a slice.	Final review period
Decision Engine (Optional)	Under development.	Not available

Table 38: Status and availability of the involved WP3/WP4 enablers for TC7

5.6.2 Testbeds' status and functionality

Currently, we are preparing a demonstration testbed based on Kubernetes environment in the premises of Aalto University. A Kubernetes cluster is created with one Master node and three Worker nodes. The monitoring system is enabled where resource usage and performance metrics are collected from the different probes using Prometheus. For better visualization, a Grafana service is installed and connected to Prometheus. The Horizontal Pod Autoscaling (HPA) functionality is enabled and can be triggered with different scaling rules based on observed per-pod metrics (e.g., CPU, RAM) or external metrics provided by Prometheus (e.g., number of HTTP requests or response time).

We are also envisaging to duplicate the test environment once all required functionalities are included in Eurescom's Kubernetes testbed for performing advanced large-scale tests.

5.6.3 Preliminary results

The current achieved results include:

- The creation of two CDN slices, the configuration of the Monitoring Module that can collect metrics from different probes, and the complete development of a dataset generator to generate the dataset for training the ML model. Figure 60 illustrates the current deployed setup. The generator extracts resource usage and performance metrics for the VNFs and hosting worker nodes involved in Slice 1 and Slice 2 for a specified period. The metrics to extract are formulated using PromQL queries submitted to Prometheus. The generator generates a csv file that contains the time series of the extracted metrics. An initial small dataset is collected and now used to train the DDoS Mitigator model. We are still working on improving the dataset size and identifying the appropriate metrics that should be used to enhance the model performance.

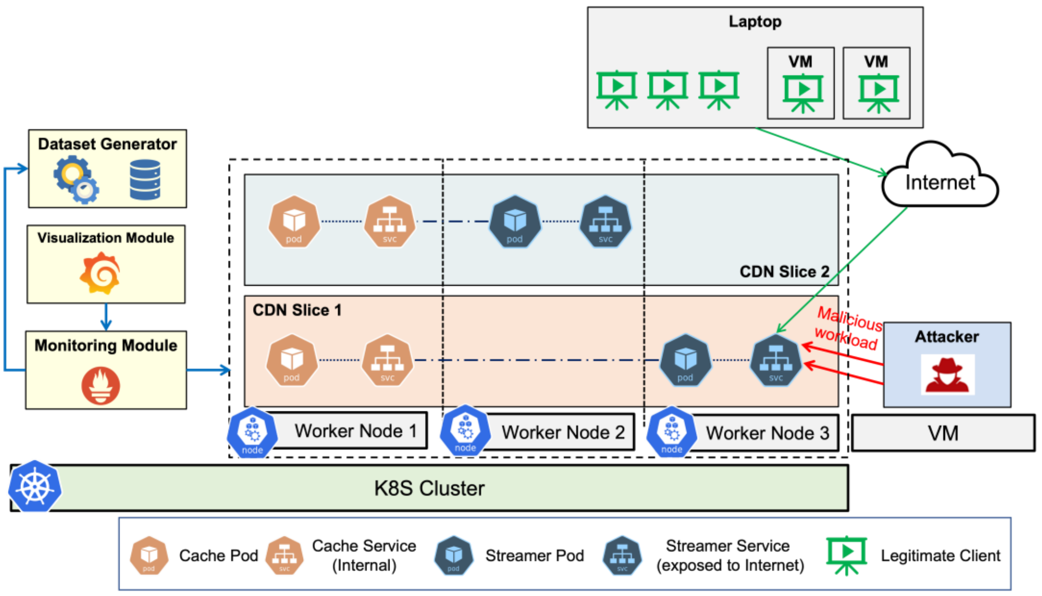


Figure 60: Kubernetes testbed for TC7

- Launching App-layer DDoS attack (e.g., Hulk) against the streamer of a slice and visualizing the attack's effect on resource usage. Figure 61 depicts the metrics extracted from streamer of Slice1 for normal behaviour and during Hulk attack.

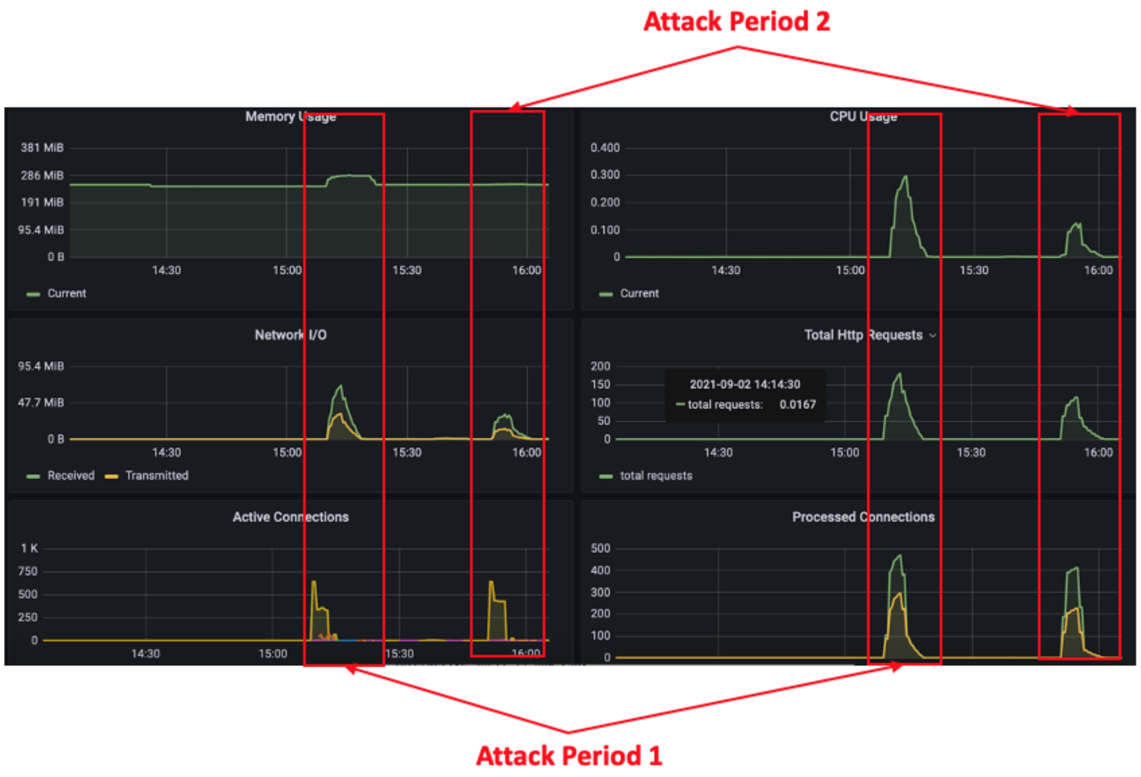
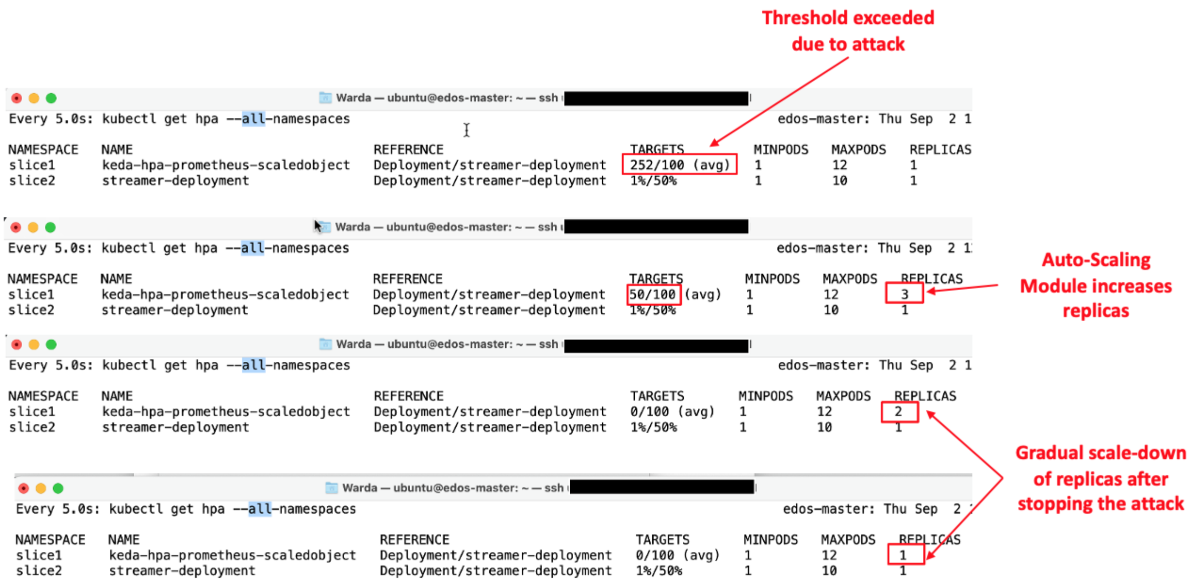


Figure 61: Grafana dashboard showing metrics extracted from streamer of Slice1

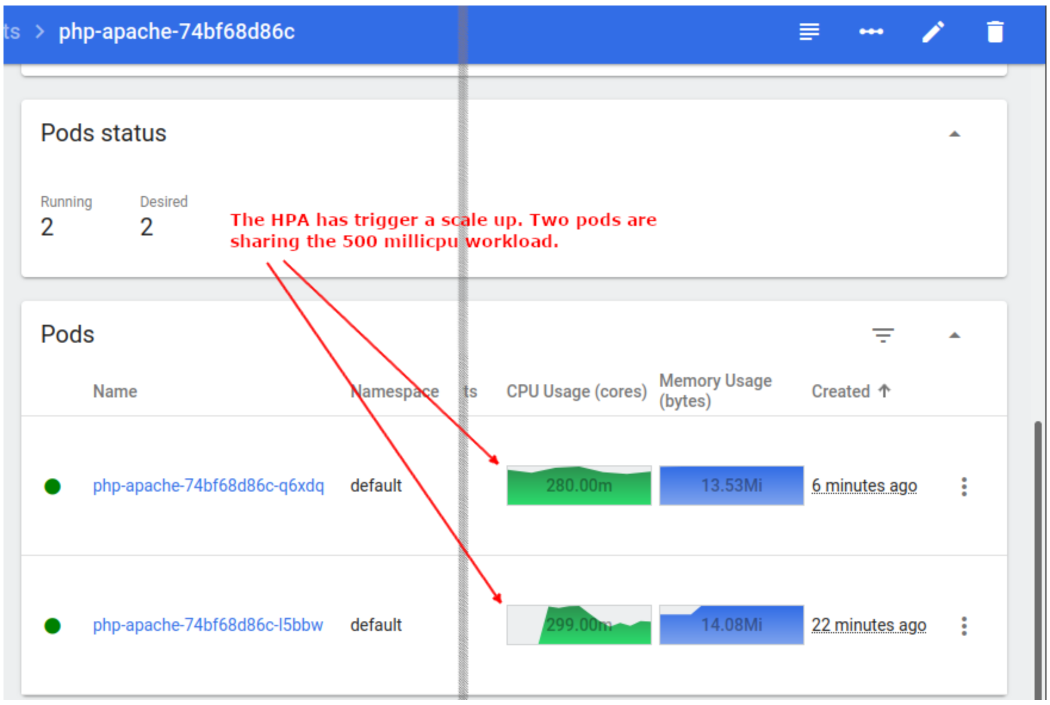
- Testing the impact of the attack on scaling with different horizontal scaling rules. The example in Figure 62 shows two horizontal scaling rules. The first scaling rule is based on the total http requests received by the streamer in a given time interval. Here a scaling threshold is set to 100 requests. Note that the "total http requests" metric is extracted from Prometheus. The second scaling rule is based on the CPU utilization by a streamer pod. We set a scaling threshold to 50%; that is, a new streamer pod is deployed if the current CPU utilization exceeds 50%. As illustrated in Figure 62, once the scaling rule is met for streamer of Slice1, the number of replicas of streamer pod is incremented to handle the workload. After stopping the attack, the HPA gradually scale-down the number of replicas.


Threshold exceeded due to attack
Auto-Scaling Module increases replicas
Gradual scale-down of replicas after stopping the attack

```
Warda — ubuntu@edos-master: ~ — ssh [redacted]
Every 5.0s: kubectl get hpa --all-namespaces                                          edos-master: Thu Sep  2 1

NAMESPACE   NAME                               REFERENCE                          TARGETS          MINPODS   MAXPODS   REPLICAS
slice1      keda-hpa-prometheus-scaledobject   Deployment/streamer-deployment     252/100 (avg)    1         12        1
slice2      streamer-deployment                Deployment/streamer-deployment     1%/50%           1         10        1

Warda — ubuntu@edos-master: ~ — ssh [redacted]
Every 5.0s: kubectl get hpa --all-namespaces                                          edos-master: Thu Sep  2 1

NAMESPACE   NAME                               REFERENCE                          TARGETS        MINPODS   MAXPODS   REPLICAS
slice1      keda-hpa-prometheus-scaledobject   Deployment/streamer-deployment     50/100 (avg)   1         12        3
slice2      streamer-deployment                Deployment/streamer-deployment     1%/50%         1         10        1

Warda — ubuntu@edos-master: ~ — ssh [redacted]
Every 5.0s: kubectl get hpa --all-namespaces                                          edos-master: Thu Sep  2 1

NAMESPACE   NAME                               REFERENCE                          TARGETS       MINPODS   MAXPODS   REPLICAS
slice1      keda-hpa-prometheus-scaledobject   Deployment/streamer-deployment     0/100 (avg)   1         12        2
slice2      streamer-deployment                Deployment/streamer-deployment     1%/50%        1         10        1

Warda — ubuntu@edos-master: ~ — ssh [redacted]
Every 5.0s: kubectl get hpa --all-namespaces                                          edos-master: Thu Sep  2

NAMESPACE   NAME                               REFERENCE                          TARGETS       MINPODS   MAXPODS   REPLICAS
slice1      keda-hpa-prometheus-scaledobject   Deployment/streamer-deployment     0/100 (avg)   1         12        1
slice2      streamer-deployment                Deployment/streamer-deployment     1%/50%        1         10        1
```

Figure 62: Horizontal auto-scaling reaction to DDoS attack

- The integral development (from design to test) of the auto-scaling module KACD (Kubernetes Admission Controller Delegator) and the testing of the KACD component inside a generic Kubernetes environment. Figure 63 displays an unmitigated scale up. The Horizontal Pod Autoscaler from Kubernetes has scaled up a Pod to meet the workload increase. The ongoing load of 500 millicpu has been split up between two Pods.



Pods status

Running 2 Desired 2

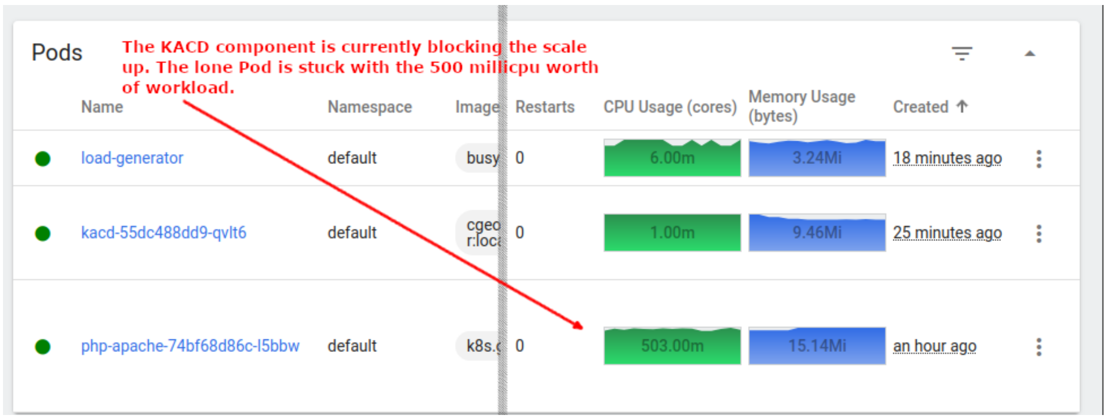
The HPA has trigger a scale up. Two pods are sharing the 500 millicpu workload.

Pods

Name	Namespace	ts	CPU Usage (cores)	Memory Usage (bytes)	Created ↑
● php-apache-74bf68d86c-q6xdq	default		280.00m	13.53Mi	6 minutes ago
● php-apache-74bf68d86c-l5bbw	default		299.00m	14.08Mi	22 minutes ago

Figure 63: A pod scale up in Kubernetes

When the KACD component is used to intercept and validate a scaling event then the Kubernetes platform will not apply the event. As shown in Figure 64, the overloaded Pod cannot scale up and has to handle the 500 millicpu worth of workload. Whereas in normal condition, the Pod would have been duplicated and each Pod would split the workload to only handle ~250 millicpu of work.



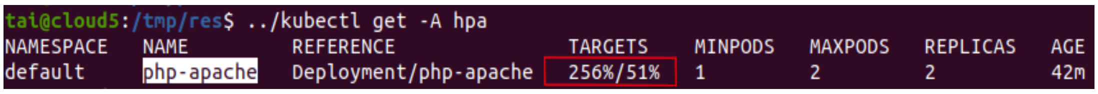
Pods

The KACD component is currently blocking the scale up. The lone Pod is stuck with the 500 millicpu worth of workload.

	Name	Namespace	Image	Restarts	CPU Usage (cores)	Memory Usage (bytes)	Created ↑
●	load-generator	default	busy	0	6.00m	3.24Mi	18 minutes ago
●	kacd-55dc488dd9-qvlt6	default	cgeo r:loca	0	1.00m	9.46Mi	25 minutes ago
●	php-apache-74bf68d86c-l5bbw	default	k8s.(	0	503.00m	15.14Mi	an hour ago

Figure 64: A blocked auto-scale up in Kubernetes using KACD

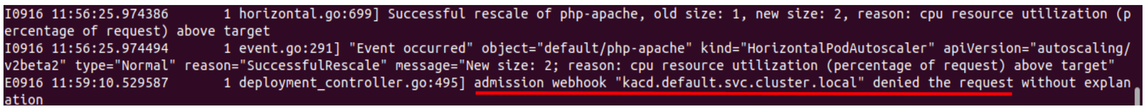
Figure 65 shows that the HPA has detected that his target has reached an overloaded state of 256% and it wants to set the number of desired replicas to 2.



```
tai@cloud5:/tmp/res$ ../kubectl get -A hpa
NAMESPACE   NAME         REFERENCE                TARGETS    MINPODS   MAXPODS   REPLICAS   AGE
default     php-apache   Deployment/php-apache    256%/51%   1         2         2          42m
```

Figure 65: HPA description

The KACD is blocking all requests made by the HPA to modify the number of Pods by submitting a new ReplicaSet resource to the Kubernetes API. The Fig shows the Kubernetes API logs where the HPA request is denied by the KACD component. In this version KACD delegates the decision to an external shell script, but it can be modified to almost anything, like making a curl request to the Damage Mitigator component. This behaviour is shown in Figure 66.



```
I0916 11:56:25.974386       1 horizontal.go:699] Successful rescale of php-apache, old size: 1, new size: 2, reason: cpu resource utilization (p
ercentage of request) above target
I0916 11:56:25.974494       1 event.go:291] "Event occurred" object="default/php-apache" kind="HorizontalPodAutoscaler" apiVersion="autoscaling/
v2beta2" type="Normal" reason="SuccessfulRescale" message="New size: 2; reason: cpu resource utilization (percentage of request) above target"
E0916 11:59:10.529587       1 deployment_controller.go:495] admission webhook "kacd.default.svc.cluster.local" denied the request without explan
ation
```

Figure 66: K8S log with KACD blocking a scale up request

5.7 Progress towards integration of enablers in TC8

5.7.1 Enablers' status and availability

TC8 involves two security enablers developed in WP3 as shown in Table 39. The DiscØvery enabler provided by CLS, and the Security Analytics Framework provided by NCSRD.

Enabler	Status	Availability
DiscØvery	Developed	Currently available
Security Analytics Framework	Developed	Currently available

Table 39: Status and availability of the involved WP3/WP4 enablers for TC8

5.7.2 Testbeds' status and functionality

The testbed of TC8 is developed and deployed on CLS's infrastructure. The testbed is fully operational. The testbed has been designed to virtualise the test environment that was available for the 5G-CARMEN use cases. The testbed can generate 5G network traffic from simulated connected vehicles and infrastructure. The testbed has been integrated with the CTTC's infrastructure to further test the integration of enablers.

5.7.3 Preliminary results

The test case infrastructure scenario is composed of two MECs, one MEC in each border, three vehicles, and one emergency vehicle. At the instance of the model, each vehicle is connected to the MEC closest to it. The two MEC and the vehicles host a software application that processes real-time data. The real-time data are necessary for connected vehicle applications, such as the emergency vehicle, or collaborative manoeuvring. The model of the system is presented in Figure 67.

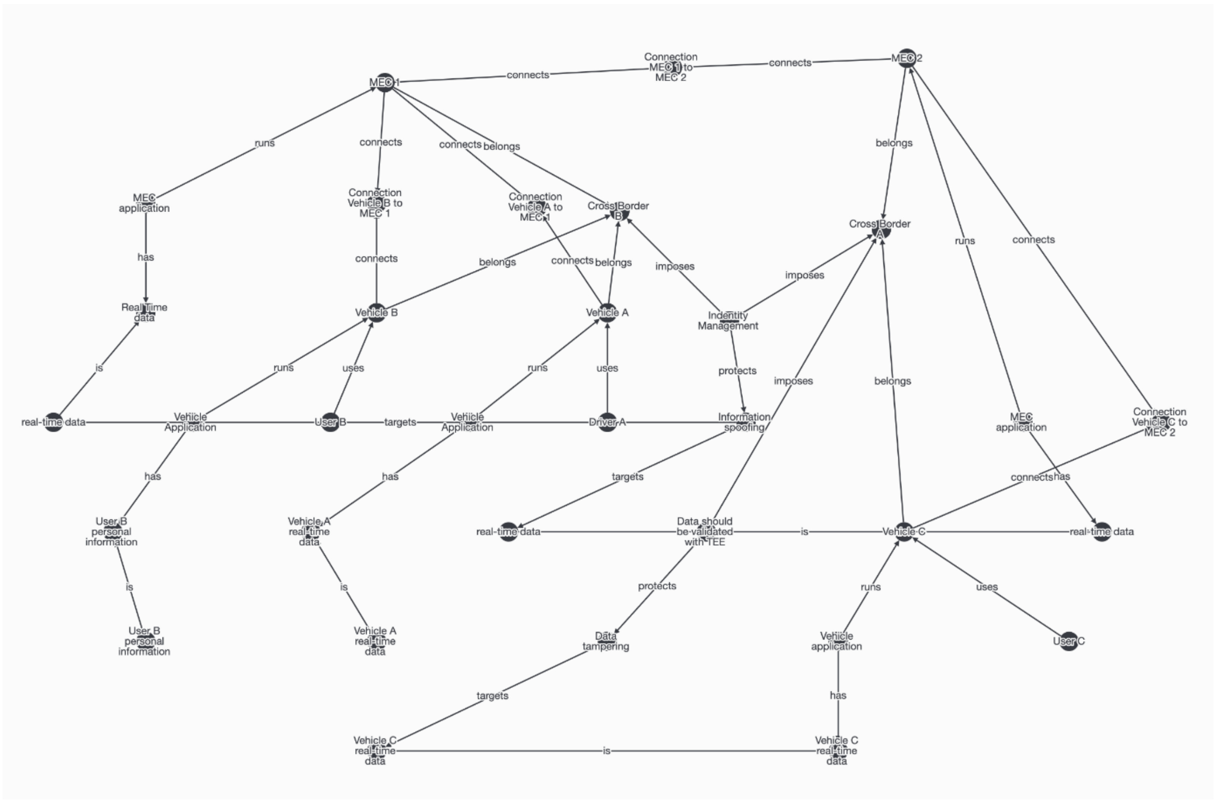
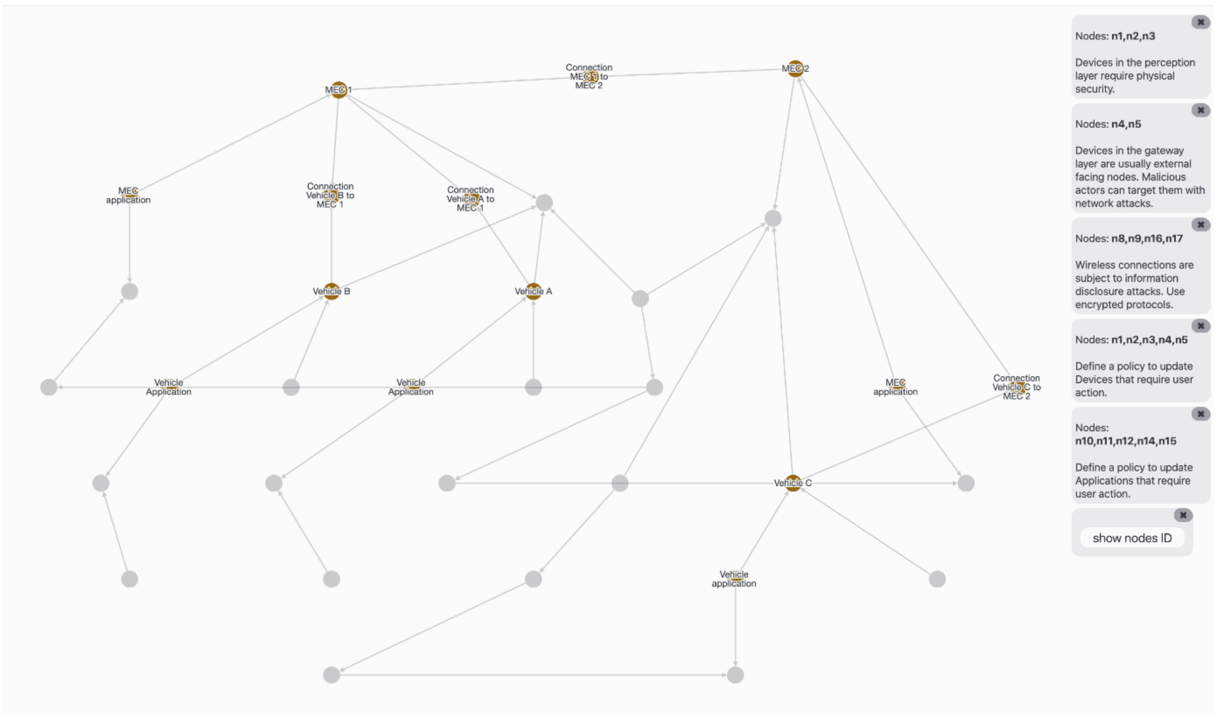


Figure 67: System model of TC8

As shown in the system model of TC8, the attributes of the model's components can be used to better understand the security posture of the system. This information can be leveraged to infer suggestions to improve the security of the components of the system or detect threats that may impact it. The preliminary results model represents only the basic components of the TC8. It does not include additional components that can be used to improve the process of threat assessment, such as mitigation mechanisms, vulnerabilities, or policies. However, the configuration of the system can reveal security issues. For example, the MEC deployment strategy of the test case shows that the resources of a MEC can be accessed by malicious actors in close proximity to the MEC server. A similar proximity-based security threat exists in the vehicles as well. DiscØvery can detect such security threats based on the information encoded in the model. The results can be presented in the form of high-level security suggestions to the security analyst.

In the TC model information, we did not include any type of policy, such as an update cycle policy. Update cycles of software applications can prevent the exploitation of assets by malicious actors. Software applications that are not regularly or automatically updated can remain in software versions that contain vulnerabilities. Malicious attackers can leverage such information and target outdated components of the system. DiscØvery can detect that the present model does not contain such policies. The analysis output of DiscØvery provides a security insight to the security analyst to incorporate and update policy. Another security issue that is inferred from the model is the network protocols that are used between the components. Certain components use unencrypted protocols, such as HTTP, which do not use TLS or other encryption layers. Unencrypted protocols can result in information disclosure attacks. The enabler can detect which network connection nodes do not use unencrypted protocols. It can then suggest to the security analyst to upgrading them into encrypted protocols. The preliminary security analysis is shown in Figure 68.



Nodes: **n1,n2,n3**

Devices in the perception layer require physical security.

Nodes: **n4,n5**

Devices in the gateway layer are usually external facing nodes. Malicious actors can target them with network attacks.

Nodes: **n8,n9,n16,n17**

Wireless connections are subject to information disclosure attacks. Use encrypted protocols.

Nodes: **n1,n2,n3,n4,n5**

Define a policy to update Devices that require user action.

Nodes: **n10,n11,n12,n14,n15**

Define a policy to update Applications that require user action.

Figure 68: Preliminary results of the security analysis of TC8

5.8 Progress towards integration of enablers in TC9

5.8.1 Enablers' status and availability

As defined in the previous section, this TC used two security enablers from WP3 as presented in Table 40 below.

Enabler	Status	Availability
Secure and federated network slice broker (SFSBroker)	Under development	Final Review Period
Katana slice manager	Developed	Currently available

Table 40: Status and availability of the involved WP3/WP4 enablers for TC9

5.8.2 Testbeds' status and functionality

The testbed of TC9 was started at the beginning of the INSPIRE-5Gplus and two conference papers were published[16][17] to present the main idea. Currently the testbed is locally implemented at Oulu premises with the local deployments of security enablers. Further development of the test bed will proceed with the improvements of the security enablers involved. The testing of TC9 in the integration platform is underway.

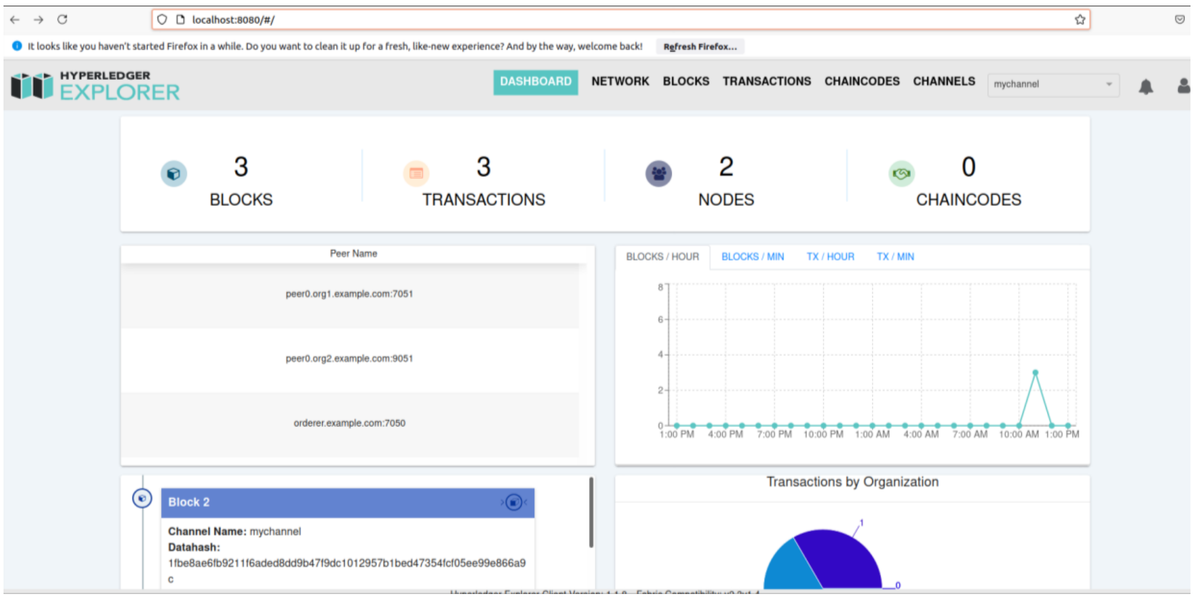
5.8.3 Preliminary results

The preliminary results included from the experiments performed in order to investigate different aspects of the SFSBroker. The experiments mainly included the end-to-end latency measurement on variable concurrent transaction count, end to end latency measurement on variable parameter and MNO counts, and the responsiveness of the SFSBroker for DoS attacks upon integration of the Security Service Blockchain (SSB) to prevent DoS attacks.

In the current experimental setup of SFSBroker, MNO services have been simulated in the local environment for the experimental evaluation. The SFSBroker has been implemented using Hyperledger Fabric blockchain network with Java-based smart contracts. Hyperledger Fabric has been deployed with RAFT consensus configuration. The integration of SFSBroker with the consumer end (IoT tenants) has been performed using MQTT brokering service.

The MNO services include infrastructure, management and orchestration service and slice manager. The infrastructure setup has been simulated using developer version of the OpenStack, which is known as DevStack. Management and orchestration implemented using OpenMANO and the slice manager has been implemented by deploying Katana slice manager. Katana slice manager integrated with the SFSBroker in REST APIs.

Figure 69 reflects the Hyperledger Explorer view on the SFSBroker blockchain. Figure 70 reflects the blocks which include the transactions committed by SFSBroker and Figure 71 reflects the Katana services deployed to simulate MNO integration in the local environment.


localhost:8080/#/

It looks like you haven't started Firefox in a while. Do you want to clean it up for a fresh, like-new experience? And by the way, welcome back! Refresh Firefox...

3 BLOCKS

3 TRANSACTIONS

2 NODES

0 CHAINCODES

Peer Name
peer0.org1.example.com:7051
peer0.org2.example.com:9051
orderer.example.com:7050

Block 2

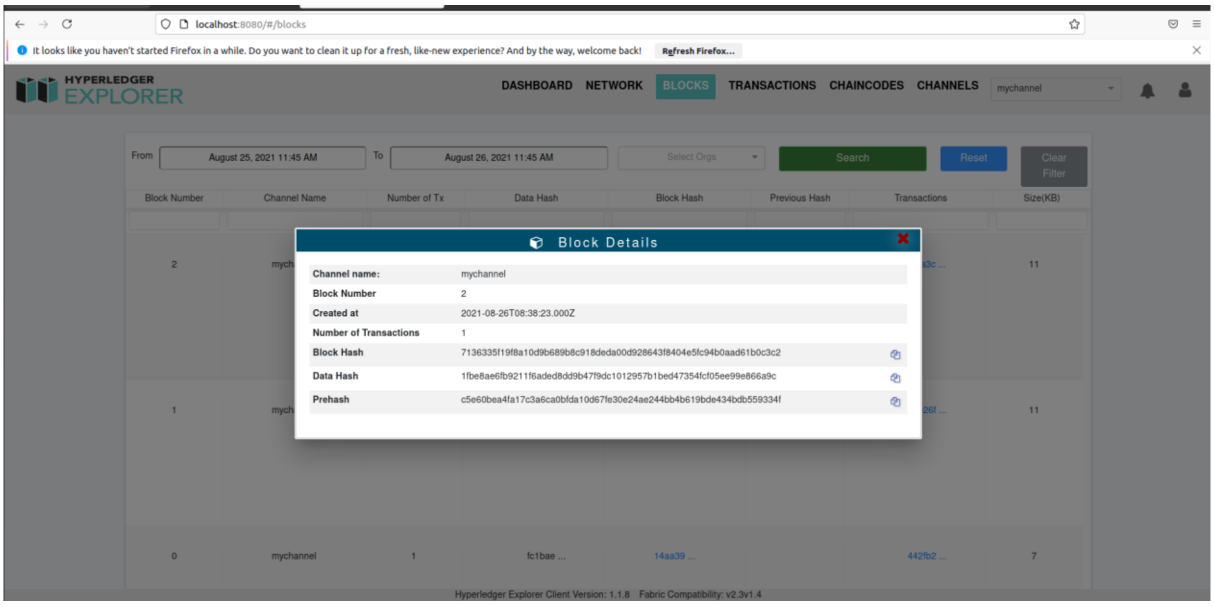
Channel Name: mychannel

Datahash: 1fbe8ae6fb9211f6aded8dd9b47f9dc1012957b1bed47354fcf05ee99e866a9c

Figure 69: Hyperledger Explorer view on the SFSBroker blockchain

[16] Hewa, T., Kalla, A., Porambage, P., Liyanage, M. and Ylianttila, M., 2021. How DoS attacks can be mounted on Network Slice Broker and can they be mitigated using blockchain? In Proc. IEEE 32nd Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC).

[17] Tharaka Hewa, T., Weerasinghe, N., Porambage, P., Kalla, A., Georgios, X., Christopoulou, M., Liyanage, M., Ylianttila, M., 2021. SFSBroker: Secure and Federated Network Slice Broker for 5G and Beyond. In Proc. IEEE Joint European Conference on Networks and Communications (EuCNC) 6G Summit.


localhost:8080/#/blocks
It looks like you haven't started Firefox in a while. Do you want to clean it up for a fresh, like-new experience? And by the way, welcome back!
Refresh Firefox...
HYPERLEDGER EXPLORER
DASHBOARD
NETWORK
BLOCKS
TRANSACTIONS
CHAINCODES
CHANNELS
mychannel
From
To
Select Orgs
Search
Reset
Clear Filter
Hyperledger Explorer Client Version: 1.1.8 Fabric Compatibility: v2.3v1.4

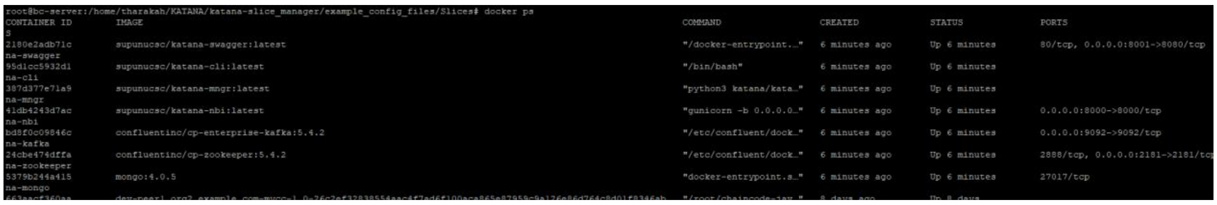
From: August 25, 2021 11:45 AM To: August 26, 2021 11:45 AM

Block Number	Channel Name	Number of Tx	Data Hash	Block Hash	Previous Hash	Transactions	Size(KB)
2	mych					3c ...	11
1	mych					26f ...	11
0	mychannel	1	fc1bae ...	14aa39 ...		442fb2 ...	7

Block Details

Channel name:	mychannel
Block Number	2
Created at	2021-08-26T08:38:23.000Z
Number of Transactions	1
Block Hash	7136335f19f8a10d9b689b8c918deda00d928643f8404e5fc94b0aad61b0c3c2
Data Hash	1fbe8ae6fb9211f6aded8dd9b47f9dc1012957b1bed47354fcf05ee99e866a9c
Prehash	c5e60bea4fa17c3a6ca0bfda10d67fe30e24ae244bb4b619bde434bdb559334f

Figure 70: The blocks which include the transactions committed by SFSBroker



```
root@bc-server:/home/tharakah/KATANA/katana-slice_manager/example_config_files/Slices# docker ps
CONTAINER ID        IMAGE                                                                    COMMAND                  CREATED          STATUS          PORTS
S
2180e2adb71c        supunucsc/katana-swagger:latest                                          "/docker-entrypoint.…"   6 minutes ago    Up 6 minutes    80/tcp, 0.0.0.0:8001->8080/tcp
na-swagger
95d1cc5932d1        supunucsc/katana-cli:latest                                              "/bin/bash"              6 minutes ago    Up 6 minutes
na-cli
387d377e71a9        supunucsc/katana-mngr:latest                                             "python3 katana/kata…"   6 minutes ago    Up 6 minutes
na-mngr
41db4243d7ac        supunucsc/katana-nbi:latest                                              "gunicorn -b 0.0.0.0…"   6 minutes ago    Up 6 minutes    0.0.0.0:8000->8000/tcp
na-nbi
bd8f0c09846c        confluentinc/cp-enterprise-kafka:5.4.2                                   "/etc/confluent/dock…"   6 minutes ago    Up 6 minutes    0.0.0.0:9092->9092/tcp
na-kafka
24cbe474dffa        confluentinc/cp-zookeeper:5.4.2                                          "/etc/confluent/dock…"   6 minutes ago    Up 6 minutes    2888/tcp, 0.0.0.0:2181->2181/tcp
na-zookeeper
5379b244a415        mongo:4.0.5                                                              "docker-entrypoint.s…"   6 minutes ago    Up 6 minutes    27017/tcp
na-mongo
663aacf360aa        dev-peer1.org2.example.com-mycc-1.0-26c2ef32838554aac4f7ad6f100aca865e87959c9a126e86d764c8d01f8346ab   "/root/chaincode-jav…"   8 days ago       Up 8 days
```

Figure 71: Katana services deployed to simulate MNO integration in the local environment

6 Conclusions

This deliverable provided the first implementation of the 5G security testing infrastructure environments developed in the context of INSPIRE-5Gplus project activities. To ensure consistency with prior outputs of WP5, the modifications/amendments carried out in certain TCs are reported and the definition of three new demonstrators is introduced. The demonstrators leverage on the progress of advanced security components in INSPIRE-5Gplus TCs to provide an extensive coverage of the HLA functionalities and evaluation against KPIs. Ongoing and future WP5 activities will aim to consolidate the operational principles of the three demonstrators.

In addition, an integration methodology framework was presented to ensure the seamless integration and re-configuration of the developed INSPIRE-5Gplus enablers in the TCs along with a detailed description of the functional verification tests performed for each TC. We further extend and corroborate the identified INSPIRE-5Gplus KPIs stemming from the development of specific security and trust/liability INSPIRE-5Gplus enablers reported in D5.1. Leveraging on the comprehensive description of the different TCs and the undergoing evolution of the integrated security and trust/liability enablers, we provided an evaluation methodology and a baseline of assessment criteria. Finally, preliminary results pertaining to the status of each TC towards the integration of relevant enablers and testbeds are reported in an effort to verify security components against pre-defined tests for each TC.

Besides the elaboration of the building blocks of the three demonstrators, the next steps in WP5 will continue to be performed in close interaction and cooperation with WP2, WP3 and WP4 towards the operational validation of the demonstrators and the evaluation of KPIs. The derived insights are expected to verify whether the security requirements can be satisfied using the developed 5G security testing infrastructure.

Appendix A Integration with ICT-17/18/19 projects

The Appendix provides concise updates related to the integration of certain TCs with ICT-17/18/19 platform scenarios.

A.1 Test Case 1

This whole TC is based on the use cases defined in the EU 5GCroCo project, where the communications between vehicles is researched and tested. The two scenarios presented are based on the architecture is presented in Figure 72 with a central node and wireless nodes as access points for the vehicles attached in the network.

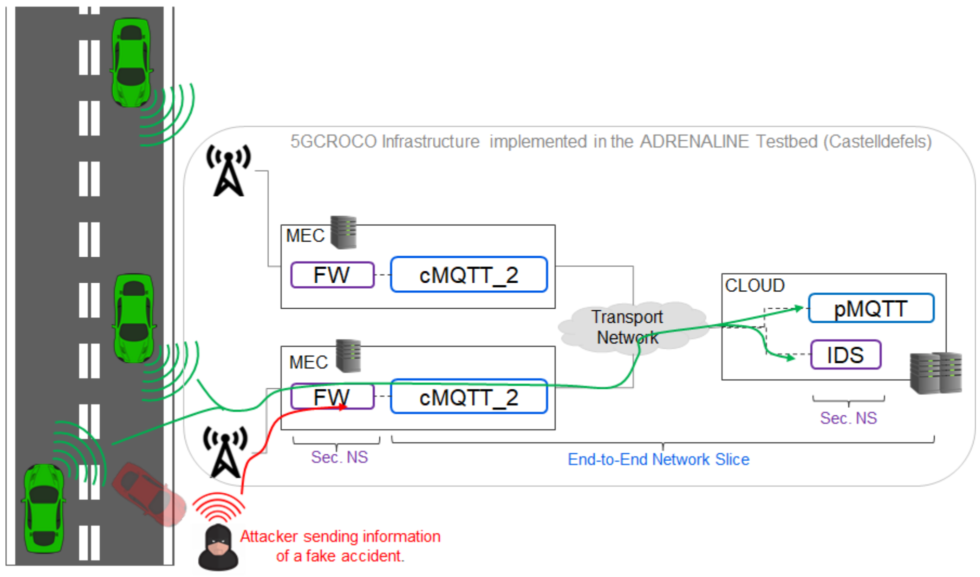

5GCROCO Infrastructure implemented in the ADRENALINE Testbed (Castelldefels)
MEC
FW
cMQTT_2
MEC
FW
cMQTT_2
Transport Network
CLOUD
pMQTT
IDS
Sec. NS
Sec. NS
End-to-End Network Slice
Attacker sending information of a fake accident.

Figure 72: 5GCroCo scenario for the INSPIRE-5Gplus TC1

A.2 Test Case 3/Test Case 4

The 5G infrastructure defined for this TC3/TC4 involves the Murcia testbed and 5TONIC testbed to demonstrate a multi-domain scenario. Currently, 5TONIC is a crucial infrastructure part of the infrastructure projects in the 5G PPP phase 3. It shares 5G resources and management (MEC, access and NFVI) within 5GVINNI and 5GEVE projects. It has also been involved in the applicability for 5G verticals, done in 5GROWTH, where it has also provided resources. As a result, the site already has a deployed network infrastructure for supporting pre-5G trials and several use-cases. The objective in TC3/TC4 is to enhance the 5GVINNI infrastructure (5TONIC site) connectivity through an interconnection with the Murcia site and provide the management and automation framework from INSPIRE-5Gplus architecture for the E2E encryption setup.

A.3 Test Case 5

The 5G infrastructure of TC5 is based on the Athens 5GENESIS testbed that will host the TC5 scenario. The Athens testbed features 5G NSA and SA deployments and an extensive NFV infrastructure, which is operated by the OSM, while the lifecycle of slices is managed by the Katana Slice Manager. The testbed allows experimenters to deploy experiments for validating network performance KPIs and provides a variety of configurations for supporting diverse vertical industries. The objective of TC5 is to highlight proactive and reactive security solutions and extend the testbed's security domain.

A.4 Test Case 6

The 5G infrastructure defined for this TC6 involves the Spain-Portugal cross-border corridor that connects the cities of Vigo and Porto. This corridor covers the complete value chain including car manufactures, telecom companies, public administrations and research institutions. The main goal of 5G-MOBIX is to set up the basis for the deployment of 5G cooperative, connected and automated mobility (CCAM) services and applications and give strong impulses in both countries towards the development of opportunities around 5G in the intelligent transportation system (ITS) sector. In this context, TC6 makes use of this infrastructure to achieve the migration of security procedures in handover scenarios through INSPIRE-5Gplus architecture

A.5 Test Case 8

The TC8 is based on the Back Situation Awareness Function (BSAF) use case of 5G-CARMEN, which is an ICT-18 project which aims to leverage 5G advances to provide safer, greener, and more intelligent transportation focusing on the Bologna-Munich corridor. The TC makes uses of the available security models and analysis that are made publicly available by the 5G-CARMEN consortium.