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

INSPIRE-5Gplus

INtelligent Security and PervasIve tRust for 5G and Beyond

D3.2: Security drivers and associated software-defined models

Version: v1.4

Deliverable type	R (Document, report)
Dissemination level	PU (Public)
Due date	31/10/2021
Submission date	05/11/2021
Lead editor	Edgardo Montes de Oca (Montimage)
Authors	Dhouha Ayed (TSG), Geoffroy Chollon (TSG), Nicolas Peiffer (TSG), Cyril Dangerville (TSG), Orestis Mavropoulos (CLS), Anastasios Kourtis (NCSRD), George Xilouris (NCSRD), Maria Christopoulou (NCSRD), Themistoklis Anagnostopoulos (NCSRD), Edgardo Montes de Oca (MI), Huu Nghia Nguyen (MI), Antonio Pastor (TID), , Juan Carlos Caja (TID), Vincent Lefebvre (TAGES), Chafika Benzaid (AALTO), Tarik Taleb (AALTO), Othmane Hireche (AALTO), Yongchao Dang (AALTO), Pol Alemany (CTTC), Charalampos Kalalas (CTTC), Ricard Vilalta (CTTC), Raul Muñoz (CTTC), Gürkan Gür (ZHAW), Wissem Soussi (ZHAW), Jordi Ortiz (UMU), Alejandro Molina (UMU), Rodrigo Asensio (UMU), Pawani Porambage (UOULU), Tharaka Hewa (UOULU), Aleksandra Podlasek (OPL), Rafał Artych (OPL)
Reviewers	Dhouha Ayed (TSG), Raul Muñoz (CTTC), Jean-Philippe Wary (Orange)
Work package, Task	WP3, T3.2
Keywords	Security policies, Security SLA, 5G security architecture, security management, security orchestration, secure network slices, service chaining, microservices, moving target defence, intrusion detection, intrusion prevention, monitoring, NFV, VNF, SDN, risk assessment

Abstract

This deliverable describes the results of Task “Software-defined models and tools to drive 5G Security” of the INSPIRE-5Gplus project that focused on identifying the list of governing models and enablers supporting software defined security required by 5G mobile networks. It provides the final version of the APIs that will be implemented by the tools developed or adapted by the different INSPIRE-5Gplus project partners.

Document revision history

Version	Date	Description of change	List of contributor(s)
v0.1 MS	13/01/21	First version of MS4 with ToC (**MS refers to Milestone**)	E. Montes de Oca (MI)
v0.2 MS	30/04/21	Contributions from partners	All authors
v1.0 MS	31/05/21	Final version of MS4	E. Montes de Oca (MI)
v0.2	15/07/21	Plans for D3.2	E. Montes de Oca (MI)
v1.0	26/09/21	New main contributions: 1 Introduction updated (MI) 2.1 Introduction added (UMU) 2.2 Modifications up to figure 4 (UMU) 3.2 Some modifications (UMU, TSG) 3.3.3 First paragraph added + several modifications in existing text (UMU) 3.4 Extended and several modifications (NCSRD, CTTC, UOULU) 3.5 Extended and several modifications (UMU) 4.2 Extended and several modifications (MI) 4.3 Extended and several modifications (UMU) 4.4 Added (ZHAW) 5.1 Extended and several modifications (MI) 5.2 Extended and several modifications (OLP) 6.1 Updated (UMU, TSG) 6.4 Added (NCSRD) 6.5 Updated (UOULU) 6.7 Updated (CLS) 6.8 Updated (UMU) 6.9 Updated (MI) 6.10 Added (OPL) 6.11/6.12 Added (TID) 6.13 Added (ZHAW) 7 Added Conclusion (MI) 8 Added Glossary (MI)	All authors
v1.0	26/10/21	Addressed comments and suggestions from internal reviewers	All authors
v1.1	28/10/21	Adapt to Deliverable template	E. Montes de Oca (MI)
v1.2	03/11/21	Final editing	A. Köhler (Eures)
v1.3	05/11/21	Final updates of Introduction and Conclusions section	Dhouha Ayed (TSG)
v1.4	22/02/23	Higher-quality versions of figures 18, 19, 27 and 28 included	Orestis Mavropoulos (CLS), Maria Christopoulou (NCSRD)

List of contributing partners, per section

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

Disclaimer

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 is a public deliverable that describes the results of the Task “Software-defined models and tools to drive 5G Security” carried out during the months starting from 5/2020 to 9/2021. The main objective is to define and develop advanced techniques for providing the enablers that support 5G Security. For this, the Task focuses on identifying, defining and developing the 5G security drivers (e.g., security policies and models), tools and techniques that include the development of modelling techniques for 5G Software Defined Networks to improve automation and customisation of security management.

Customisation allows the users and operators to specify the required security levels and the policies that need to be enforced in each network slice and for each vertical application. The specification of Security Service Level Agreements (SSLAs) corresponds to a formal way of expressing these requirements that allow automating their enforcement and assessment. Furthermore, the application of AI/ML techniques helps improve the automated management of the security breaches and prevention strategies, as well as the optimisation and coordination of the different security functions.

An important enabler to achieve this enforcement and assessment is the monitoring of the different events that occur in the network, the system and the applications. Monitoring allows, for instance, verifying the integrity status of a 5G node, its installed software, its configurations, perform real-time assessment of SSLA conformance, and provide awareness in a complex dynamic and customised network environment.

Another important aspect is the introduction of policy-driven enablers. Policy modelling is introduced for specifying high, mid and low-level security policies that contain the technical information needed to deploy the necessary elements, and, when combined with the sufficiently fine-grained monitoring capabilities, allow verify that the policies are respected and providing the enablers the enforce them through, for instance, the Security Orchestrators. Here we define and instrument a High-level Security Policy Language and Medium-level Security Policy Language that facilitate the orchestration and enforcement processes.

Security Orchestration is defined to make it possible to manage 5G environments in multi-party and multi-domain contexts driven by the user defined security policies. It needs to interact with the security assets through the SDN Controllers and/or the NFV Orchestrators to guarantee that the security policies are correctly implemented by the network and functions.

Finally, another aspect that is considered in this document are the optimization techniques that can be used to improve the performance of the security management processes. Here we present techniques for chaining of security services that can be used for considering different constraints and capabilities, e.g., enforcing SSLAs, guaranteeing a certain level of performance and scalability, providing more flexible maintenance, preventing or mitigate attacks and anomalies. Moving Target Defence is a technique that it also presented that allows preventing and countering security breaches by systematically, or when needed, to change the system to improve its resilience to attacks.

The outcomes of this task are the specification of the Software-defined Models, the tools which will drive 5G security and make it adaptive, as well as the specification and development of the security extensions that make use of the devised models. It also allowed determining the impact on the orchestrators, policy and Slice managers.

Table of Contents

List of Figures

List of Tables

Abbreviations

AAL	Authentication Assurance Level
AES	Advanced Encryption Standard
AI/ML	Artificial Intelligence / Machine Learning
API	Application Programming Interface
DC	Datacentres
DPI	Deep Packet Inspection
DTLS	Datagram Transport Layer Security
E2E	End to end
ECC	Elliptic Curve Cryptography
ECDSA	Elliptic Curve Digital Signature Algorithm
ETSI	European Telecommunications Standards Institute
GPU	Graphical Processing Unit
GSMA	Global System for Mobile Communications
GST	Generic Network Slice Template
HIDS	Host-based Intrusion Detection System
HLA	High Level Architecture
HSM	Hardware Security Module
HSPL-OP	High-level Security Policy Language – Orchestration Policy
I2NSF	Interface to Network Security Functions
IETF	Internet Engineering Task Force
IPSec	Internet Protocol Security
KPIs	Key Performance Indicators
MANO	Management and Orchestration
MEC	Multi-Access Edge Computing
MEC RA	Multi-Access Edge Computing Reference Architecture
MILP	Mixed-Integer Linear Programming
MS	Milestone
MSPL-OP	Medium-level Security Policy Language – Orchestration Policy
MVNO	Mobile Virtual Network Operator
NEST	NEtwork Slice Type
NFs	Networking Functions
NFV	Network Function Virtualization
NFVM	Network Function Virtualization Manager
NFVO	NFV Orchestrator
NIST	National Institute of Standards and Technology
NSB	Network Slice Brokering
NSSI	Network Slice Subnet Instance
OASIS	Organization for the Advancement of Structured Information Standards

OGF	Open Grid Forum
ONAP	Open Network Automation Platform
OSSEC	Open-Source Host-based Intrusion Detection System SECurity
OTT	Over-The-Top media service
PCD	Policy Conflict Detector
RAM	Read Access Memory
RAN	Radio Access Network
RT-SSLA	Real-time SSLA
RX/TX	Transmit and Receive
SAD	Security Association Database
SC	Service Consumer
SFC	Service Function Chaining
SDN	Software-Defined Networking
SFs	Security Functions
SFSBroker	Secure and Federated Network Slice Broker
SLA	Service Level Agreement
SLOs	Security Service Level Objectives
SMD	Security Management Domain
SP	Service Provider
SPD	Security Policy Database
SPECS	Secure Provisioning of Cloud Services based on SLA management
SSLA	Security Service Level Agreement
RT-SSLA	Runtime monitoring Security Service Level Agreement
TEE	Trusted Execution Environment
TOSCA	Topology and Orchestration Specification for Cloud Applications
vCPU	virtual Central Processing Unit
VIM	Virtualized Infrastructure Management
VNF	Virtualized Network Function
VSF	Virtual Security Function
WS-Agreement	Web Services Agreement Specification
XML	Extensible Markup Language
XSLT	Extensible Stylesheet Language Transformations
ZSM	Zero-touch network and Service Management

1 Introduction

Software-defined models and tools correspond to a mayor trend that is driving 5G Security. They are the basis for defining and developing advanced techniques for providing the enablers that support smart (i.e., intelligent, adaptive, flexible and automated) 5G Security. This deliverable focuses on identifying, defining and developing the 5G security drivers based on security policies and models, tools and techniques related to software-based networks and that include:

- Modelling techniques for 5G Software Defined Networks to improve automation and customisation of security management;
- Security policy modelling at various levels of abstraction and their usage to deploy the necessary elements to enforce the required security level at various domains and to verify that they are respected;
- Security Service Level Agreements (SSLAs) modelling and formalisation and customisation mechanisms for specific vertical slices based on security policies and SSLAs;
- Security orchestration methods and techniques to allow an automatic and autonomous management of 5G environments in multi-party and multi-domain contexts through security policies and to optimize the provisioning, the sharing (between different tenants) and the chaining of virtualised security functions, micro-services, and virtualised network functions;
- Definition of a monitoring framework to verify the integrity status of a 5G node, its installed software, its configurations, and real-time assessment of SSLA conformance and also to provide awareness comprising customised VNFs and dynamism;
- Usage of AI to improve the automated management of the security breaches and prevention strategies;
- Definition of policy enforcement functions that implement the way the policies are deployed and are integrated in the system as drivers that communicate with final security and network assets (e.g., SDN Controllers or NFV Orchestrators);
- Definition of optimization techniques for chaining of security services depending on various constraints (i.e., security and performance constraints);
- Support of dynamic adaptation of chains to prevent or mitigate attacks and anomalies.

By Software-defined models we mean models that enable the software-based configuration and control of the network functions (NF) and particularly of the Virtualised Network Functions (VNF).

Policy orchestration models allow high-level, mid-level and low-level specification of policies containing the technical information needed to deploy the necessary elements, enforce the policies, and verify that they are respected. The Security Orchestrator manages the policies to cover the different Security Management Domains. Policies can reach intermediate levels as SDN Controllers, or the NFV Orchestrators that are responsible for controlling and managing the NFV infrastructure, or they can arrive to final security asset as low-level policies (e.g., configuration).

In order to reach the listed objectives, the Task 3.2 related to this deliverable has focused on identifying the models and the enablers that build the security architecture defined in INSPIRE-5Gplus project and support the test cases that have been defined.

In particular, the following enablers are covered in this document (see WP3 enablers detailed in D3.1 [11]):

1. Security Orchestrator (SO): enabler that enforces security policies by deploying, configuring the security functions and network, and interacting with the VIM or MANO orchestrators, and SDN controllers.
2. Policy Framework (PF): specifies security policies that will regulate the slicing configurations, service chaining, security prevention and remediation strategies, etc. It allows the modelling

and management of the policy rules, policy conflict detection, policy storage, translation of policies to lower-level representations.

3. Security Service Level Agreement (SSLA) Manager: manages the security requirements defined by the SSLAs during the full life-cycle of a Slice.
4. Security Slice Manager (SSM): enables the creation of multiple virtual networks on top of a physical architecture, allowing operators to provide portions of their networks that fit with the requirements (particularly security) coming from different vertical industries and service providers (e.g., mobile virtual network operator, MVNO). This enabler encompasses 3 enablers that were considered as separate in D3.1 [11]: the SFS Broker providing secure slice brokering capabilities, the Secured Network Slice Manager for SSLAs providing slice management based on SSLAs, and the KATANA Slice manager that implements security policy enforcement capabilities based on slice management.
5. Threat Assessment (TA): A graphical enabler that analyses the security posture of a system
6. Security Monitoring Framework and RT-SLA (SMF): tools for capturing security related information from the network, system and applications, and correlating and analysing this information to detect security breaches or vulnerabilities by assessing, for instance Real Time SSLA.
7. Moving Target Defence (MTD): aims at dynamically modifying (parts of) the infrastructure or their fingerprint to make it hard for an attacker to exploit vulnerabilities, e.g., the network's topology to make eavesdropping on specific traffic more difficult.
8. Virtual Channel Protection (VCP): a policy enforcement function enabler that represents a (D)TLS proxy.
9. I2NSF-IPSec: a policy enforcement function enabler that represents an Interface to Network Security Functions - Internet Protocol Security.

The other enablers related to WP3 (mainly those providing AI/ML analysis and management capabilities) will be detailed in deliverables D3.3 and D3.4 [1].

Figure 1 shows the coverage of the enablers with respect to the High-Level Architecture (HLA) specified in D2.2 [2].

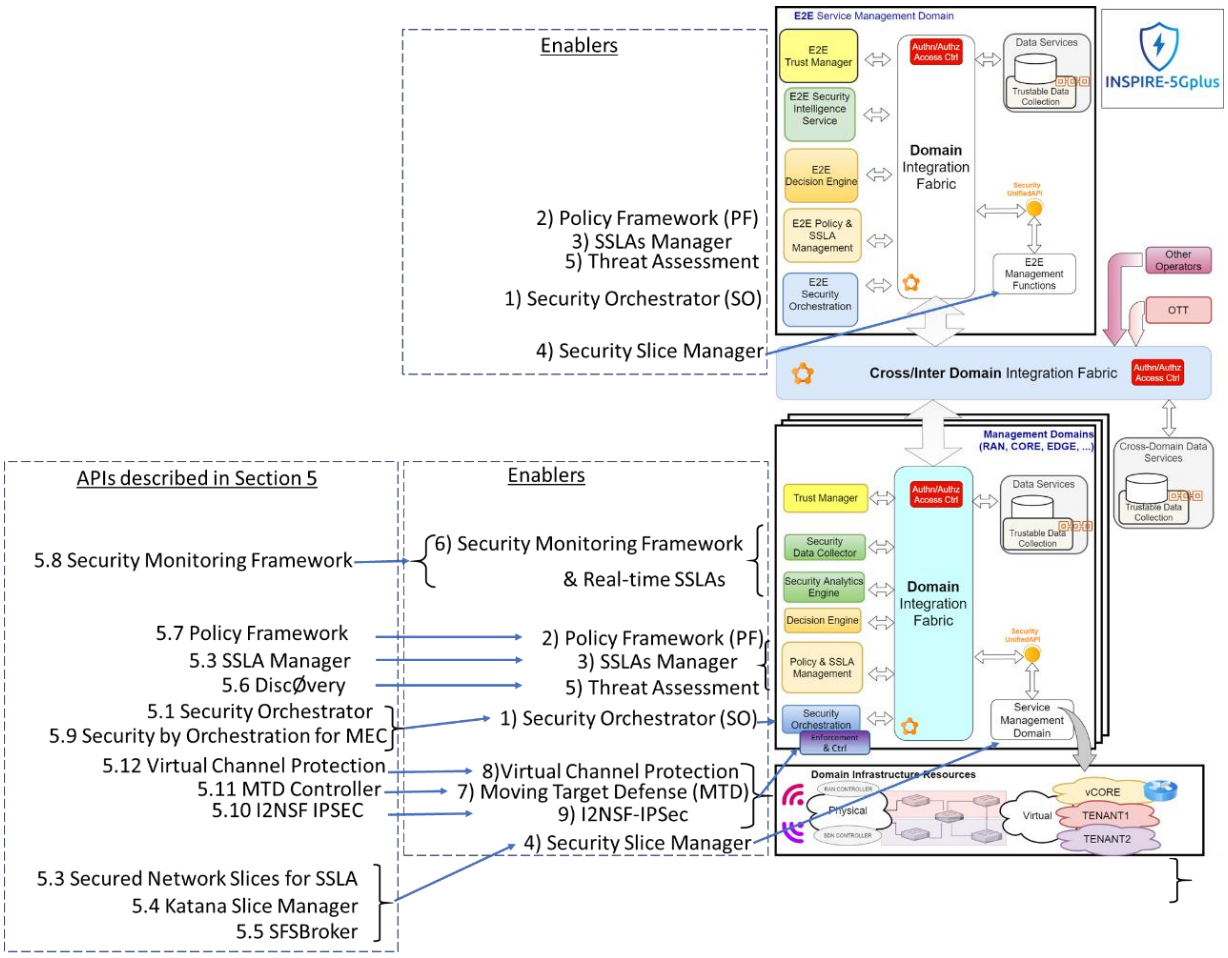

Figure 1: Mapping of enablers, APIs and HLA elements

The column in the middle lists the enablers described in this document and how they map to the HLA elements of one domain. Several enablers can correspond to one element since they cover different aspects of the elements' functions. This is notably the case for the Policy and SSLA management, and the orchestrator. An enabler can also cover more than one HLA element since it integrates several functions. This is the case for the Monitoring Framework that consists of probes that collect data and a centralised module that manages the probes and analyses the information they provide (e.g., RT-SSLA assessments).

In order to reach the listed objectives, the document is structured as follows: 1) the definition of the security policies and models that allow specifying user requirements, how these can be converted from high level requirements defined by users to low level machine actionable configurations and procedures; 2) the definition of the complete security management process that enables protect-detect-react closed loop automation based on security orchestration and SSLAs and security policies management in a Slice; 4) the definition of the models and techniques that enable automated detection, protection, and optimised security enforcement; 5) A specification of a first version of the APIs that need to be implemented so that the different tools can interact and take advantage of the security models and mechanisms to construct a fully automated end-to-end multi-tenant smart network and service security management framework across multi-domains.

2 Security policies and models

New 5G infrastructures are characterised by dynamic deployments of heterogeneous devices, functions and applications that must be constantly configured, secured and managed. In that sense, the traditional management approach, where a system administrator deals with every single configuration, is not sufficient. Policy-based approaches are able to well-define a common way to represent different kind of requirements, providing a level of abstraction that homogenizes the management and allows consistency verifications. Different policy designs and specifications focusing on different scopes have emerged and evolved along the years but only some of them already cover a wide set of security aspects. In order to ease security management, automation, deployment and configurations for different scopes, INSPIRE-5GPlus extends the multi-level abstraction approach and capabilities defined in ANASTACIA-H2020 [3]. Specifically, High-level Security Policy Language Orchestration Policy models (HSPL-OP) and Medium-level Security Policy Language Orchestration Policy models (MSPL-OP) are extended and adapted to be applied in the new INSPIRE-5GPlus orchestration and enforcement processes, which allow enforcing and managing HSPL-OP and MSPL-OP in multi-domain ZSM-aligned infrastructures.

2.1 High-level Security Policy Language for Orchestration Policies (HSPL-OP)

The High-level Security Policy Language for Orchestration Policies (HSPL-OP) [4] models high-level security requirements, priorities and dependencies in form of high-level orchestration security policies, which are disassociated from underlying technologies, e.g., “Bob is authorized to access Internet traffic once he has been authenticated”. The HSPL-OP scheme is used to codify security requirements in XML. This scheme provides different elements for modelling security policies based on different capabilities such as authorization, monitoring, channel protection and operation, among others. In this way, an HSPL Orchestration Policy is mainly composed of an action (e.g., do not authorize access) that must be performed for a specific subject (e.g., Bob) and for a specific object (e.g., Internet traffic), but it also allows the modelling of conditional fields to customize other parameters (e.g., time period) as well as the specification of priorities and dependencies between policies or dependencies between policies and the system. It can be specified by the following elements:

[subject] [action] [object] [extra_fields] [dependencies] [priority]

The *action* element is an enumeration associated with the subject and the *object* representing the type of action to be performed (e.g., authorise access). It provides multiple actions according to different security capabilities (e.g., authorise access, do not authorise access, protect confidentiality, configure monitoring). Also, the *object* element is an enumeration that represents a conceptual object or target (e.g., Internet traffic, authentication traffic, resource). The *extra fields* element allows customizing high-level policies by indicating values such as *time period*, *target*, *purpose* and *resource*. *Time period* defines the amount of time the security policy must be enforced by the system. *Target* specifies the policy target (e.g., Authentication Agent). And finally, *purpose* and *resource* fields allow specifying additional information about the purpose and the resource involved in the policy (e.g., update a specific resource). These fields also allow modelling multiple properties, consisting of sets of key/value pairs instead of a single string value. Regarding the orchestration features, the *priority* element provides a priority rank to be considered during the policy orchestration. In this way, the Security Orchestrator will consider those policies with the highest priority first. The *dependencies* element provides a list of dependencies that must be satisfied before processing the security policy. This field considers dependencies between security policies as well as dependencies between security policies and system events (Event-Condition-Action). Finally, a bi-directional attribute eases the auto-generation of multiple medium-level security policies during the refinement process, in case the same policy must be enforced in a bi-directional way.

2.2 Medium-level Security Policy Language for Orchestration Policies (MSPL-OP)

The Medium-level Security Policy Language for Orchestration Policies (MSPL-OP) [5] allows modelling more technical information in the form of medium-level orchestration security policies. These policies, although less abstract, are still independent of the underlying infrastructure. Thus, they allow representing information like IP addresses or protocols without specifying the final configurations for specific end-points. As for HSPL-OP, MSPL-OP specifies priorities and dependencies to enhance the orchestration features.

Figure 2 shows a simplified representation of the main components of an MSPL Orchestration Policy. The *"IT Resource Orchestration"* element represents the MSPL-OP. It can be composed of one or multiple *"IT Resources"* (MSPL) elements. *"IT Resource"* elements contain a *"Configuration"* and one or more *"Dependencies"*. They can also indicate *"Enabler Candidates"*. Priorities between policies are specified as an attribute of the "*ITResources*". *"Rule Set Configuration"* extends a "*Configuration*" and represents multiple *"Capabilities"* as well as multiple *"Configuration Rules"*. On the one hand, "*Capability*" elements represent main security functionalities such as resource authorisation, filtering or channel protection. In this way, capabilities play a key role as the first step for deciding a suitable enforcement point. To enforce a specific security policy, the enforcement point must implement the required capability. For instance, *filtering capability* could be enforced by enforcement points that implement traffic filtering features, such as IPTABLES or SDN management. On the other hand, each *"ConfigurationRule"* models different *"ConfigurationRuleAction"* elements and *"ConfigurationRuleCondition"* elements. This approach extends the policy language by providing new capabilities, including new actions and conditions for them.

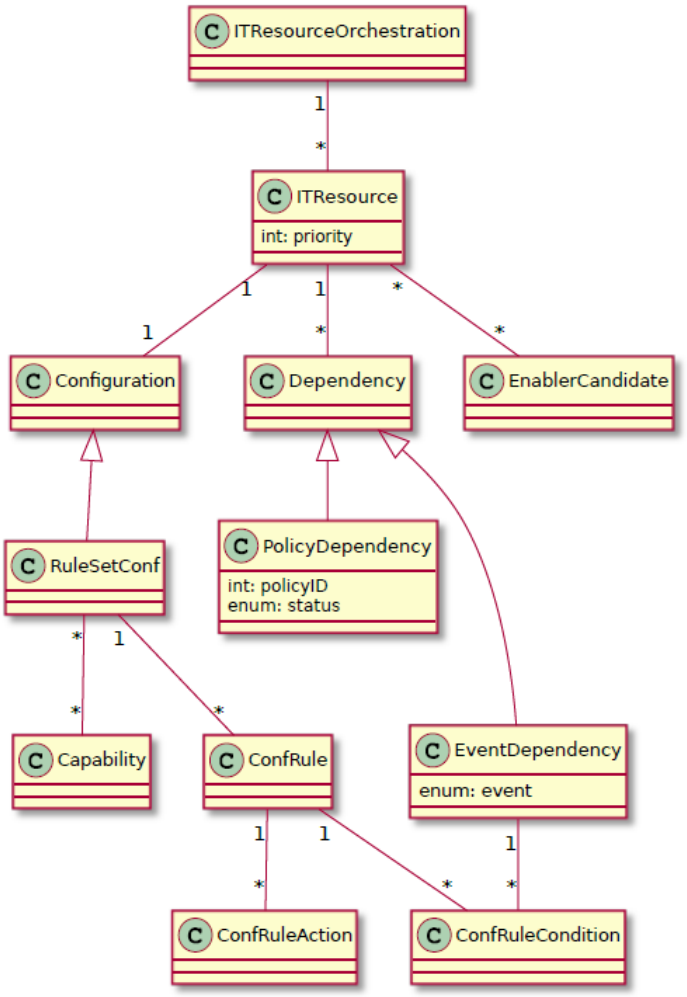

Figure 2: MSPL-OP main elements

Regarding dependencies, the *"Policy Dependency"* and *"Event Dependency"* elements extends the *"Dependency"* element, depending on the nature of the required dependency. *"Policy Dependency"* indicates that a security policy depends on a specific status of another security policy (e.g., a security policy requires that another security policy be enforced before). *"Event Dependency"* specifies that the security policy depends on a specific event triggered by the system (e.g., an authorisation security policy may depend on an authentication success event). Finally, each extended MSPL policy composing the MSPL Orchestration Policy may include an *"Enabler Candidate"* list that indicates the potential enforcement points candidates that could be considered for enforcing the security policy.

In order to allow enforcing SSLAS as well as new security capabilities across the INSPIRE5G-Plus HLA, previous models need to be extended. To this aim, new fields are being defined according to the SSLA requirements, for instance, monitoring policies have been extended to provide counting capabilities, so traffic can be measured and controlled. This is performed by providing count and threshold new elements to the monitoring configuration condition that allows specifying generic values such as the measures unit, the value and the range (e.g., 60 connections per hour, 60 seconds per day...). Besides, to ease the policy enforcement in the ZSM 5G infrastructures, models have been also extended to specify multi-tenant and multi-domain information that allows traceability during the policy orchestration and enforcement processes (e.g., Tenant and management domain id fields). Of course, beyond policies model extension, policy transformation processes also need to be improved to allow E2E policies enforcement since previous work only contemplates single domains.

2.3 Translation between models

While policy models provide a well-defined common way to represent different requirements, this approach requires additional processes to translate these policy models into final infrastructure configurations. Depending on the policy models abstract level, this policy transformation can be composed by multiple operations. Since INSPIRE-5GPlus extends a multi-level approach to provide different levels of abstraction, multiple policy transformation operations are required. Specifically, the extended security policies have 2 levels of abstraction, High-level Security Policy Language for Orchestration Policies (HSPL-OP) and Medium-level Security Policy Language for Orchestration Policies (MSPL-OP). The process that transforms HSPL-OP into MSPL-OP is denominated policy refinement, whereas the process that transforms MSPL-OP into final infrastructure configurations is denominated policy translation. *Figure 3* shows the mapping of these processes in the INSPIRE-5GPlus High-level architecture.

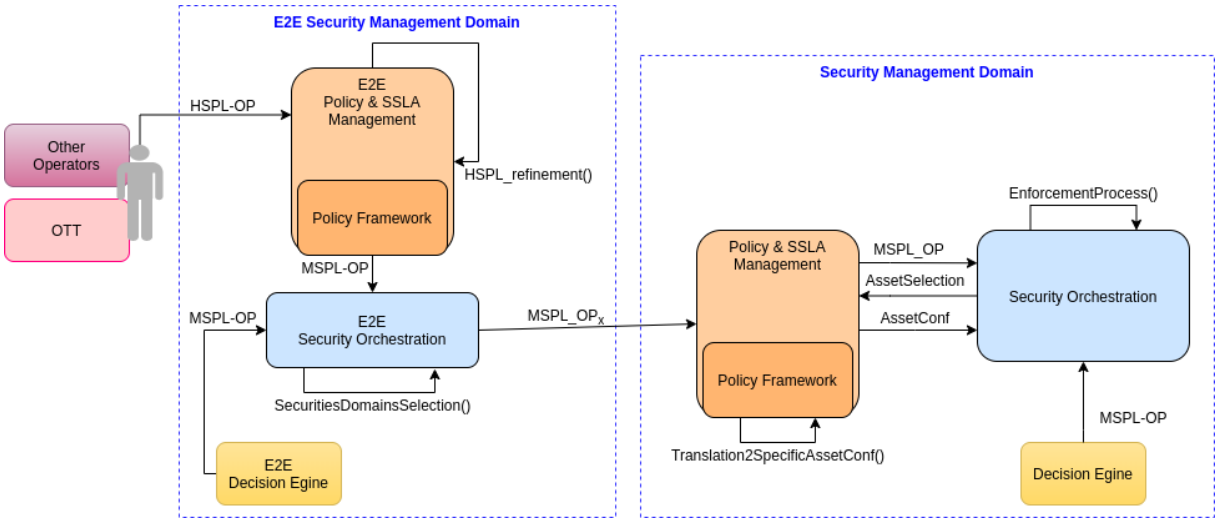

Figure 3: HSPL to MSPL translation

In INSPIRE-5GPlus, the refinement process is mainly performed at E2E level (it could also be done at the SMD level by SMD security administrators) whereas the policy translation is performed at SMD level. Since security policies must be enforced across multiple domains, HSPL-OP will be refined into different MSPL-OP that will be sent to each involved SMD. Then, each SMD will translate the MSPL-OP to final configurations in order to enforce them across the SMD infrastructure. Since the MSPL-OP provides more information (but still independent of the final configurations) it is also suitable to be generated by the Decision Engine to request the enforcement of reactive countermeasures. This is performed at both levels depending on the required scope of the reaction (intra domain or inter domain). Thus, E2E level is also able to receive MSPL-OP policies and orchestrate between the SMDs.

The policy-based orchestration processes that manages policy refinements and translations are explained in Section 3.3.3.

2.4 Policy Refinement

In INSPIRE-5GPlus, HSPL-OP are intended to be applied at E2E SMD level to ease the specification of security policies even for non-technical users through a GUI form with a reduced number of fields. This level of abstraction cannot be enforced directly due to the lack of information for the system; thus, a policy refinement process is required as part of the E2E Orchestration. When the E2E SMD Policy Framework receives a HSPL-OP, it identifies the capabilities needed to enforce each HSPL policy that composes the HSPL-OP. A well-defined capability matching between the HSPL-OP fields will provide the required capability.

Action	Object	Capability
no_authorise_access	*traffic	FILTERING
no_authorise_access	Resource	AUTHORISATION
prot_conf_integr	*traffic	CHANNEL_PROTECTION
config_monitoring	*traffic	TRAFFIC_ANALYSIS

Table 1: HSPL-OP - MSPL-OP Capability mapping example

Table 1 shows a simplified example of this matching. As it can be seen in this example, different combinations of the HSPL-OP fields will generate different capabilities. Once the capabilities have been identified, a capability-based refinement process starts. To this aim, the refiner implements a MSPL-OP skeleton generator as well as N refinement methods, where N represents the number of available capabilities. Each method implements the logic on how to generate the MSPL-OP parameters according to the identified capability and the HSPL-OP fields. To this aim, data services are essential to retrieve information about management domain infrastructure, as well as about the HSPL-OP fields since some methods will generate different MSPL parameters according to this information. For instance, a channel protection MSPL will have different parameters depending on the security level required, the network and the nature of the subject and the target (e.g., constrained devices could require DTLS whereas regular devices could use IPSec). When each MSPL has been generated, the MSPL-OP skeleton is filled, also including (if any) the HSPL-OP dependencies and priorities.

Figure 4 shows an example of channel protection HSPL-OP refinement. In this case, according to the HSPL-OP parameters, the capability matching identifies channel protection capability. To model the common MSPL E2E channel protection parameters, another matching is performed in order to find common channel protection properties among the channel protection information of each involved management domain available in the data services. In this example, the matching between both Security Management Domain channel protection information selects the IPSec protocol with a specific crypto-suite common to both Security Management Domains. If there is no common channel protection solution the process will notify it to the security administrator.

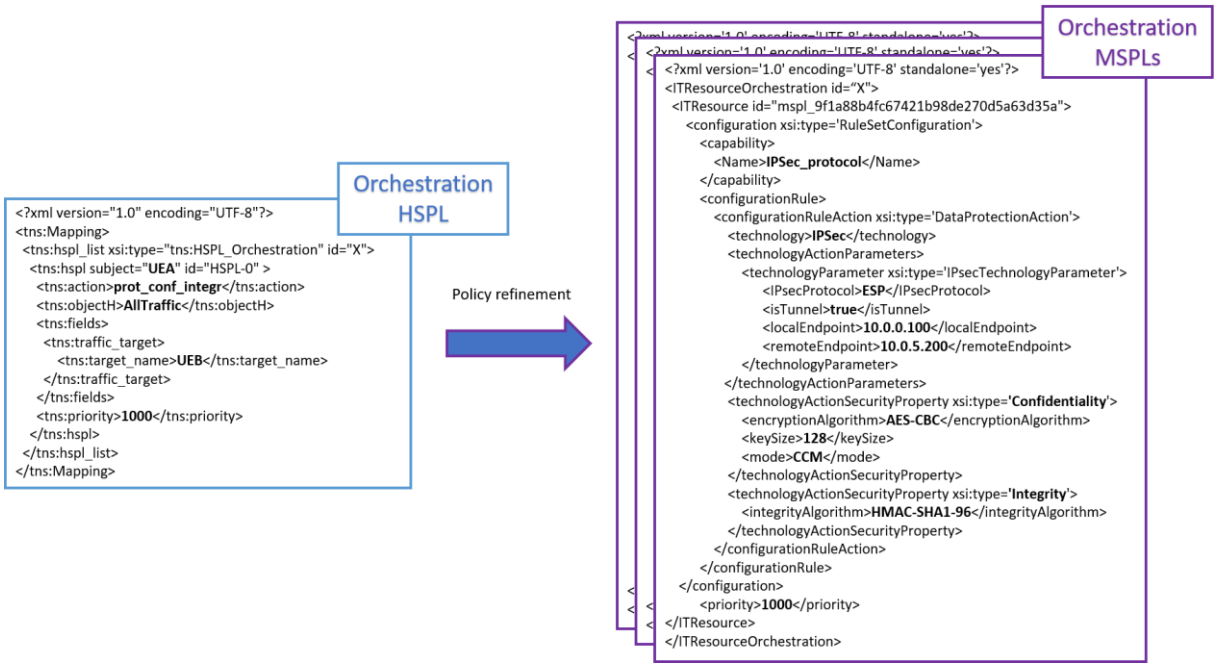

Figure 4: Policy Refinement example

The amount of MSPLs generated inside the MSPL-OP will vary depending on HSPL-OP refinement method implementation as well as on specific parameters (e.g., the "bidirectional" parameter will generate two MSPLs to cover both traffic directions). Besides, the E2E Security Orchestrator could request multiple refinements for the same MSPL-OP in order to generate multiple MSPL-OP to enforce them across the required Security Management Domains. This Refinement procedure is triggered by sending corresponding HSPL-OP to h2eservice at Policy Framework which will use h2mservice to perform the refinement.

2.5 Policy Translation

Unlike the refinement process that can be performed at the E2E SMD level and at the SMD level to refine high-level policies into medium level policies, the policy translation process is only performed at the SMD level to translate MSPL-OP into specific configurations of the infrastructure as part of the SMD orchestration. In fact, the process occurs once the SMD Security Orchestrator has selected suitable assets to enforce the MSPL-OP. Thus, the SMD Policy Framework receives a MSPL-OP and a list of the assets able to enforce each MSPL. The Policy Framework then starts the translation process which dynamically loads a translator plug-in for each tuple <MSPL, selected asset>, and executes it to get all the required low-level configurations. The plug-in approach provides a scalable and easy way to introduce new assets in the policy-based approach just by implementing a well-defined method in an isolated piece of code that will be loaded dynamically. In this way, each plug-in will be in charge of implementing the logic to translate the MSPL into the final configurations, and use the libraries provided by the policy framework such as the access management to the data services in order to retrieve the required information about the infrastructure to perform the translation. This Translation procedure is triggered by sending corresponding MSPL-OP to m2eservice at Policy Framework which will use m2lservice to perform the translation.

Figure 5 shows an example of a translation from MSPL-OP to I2NSF controller IPSec configurations. Details of the API for this enabler are available at Section 5.10. In this case, the plug-in builds a skeleton of the IETF IPSec model that is able to provide the channel protection information for the IPSec Security Association Database (SAD) and the Security Policy Database (SPD). The MSPL-OP is then processed in order to extract the required values. Any missing values (or default values) can be retrieved from the data services in order to complete the translation.

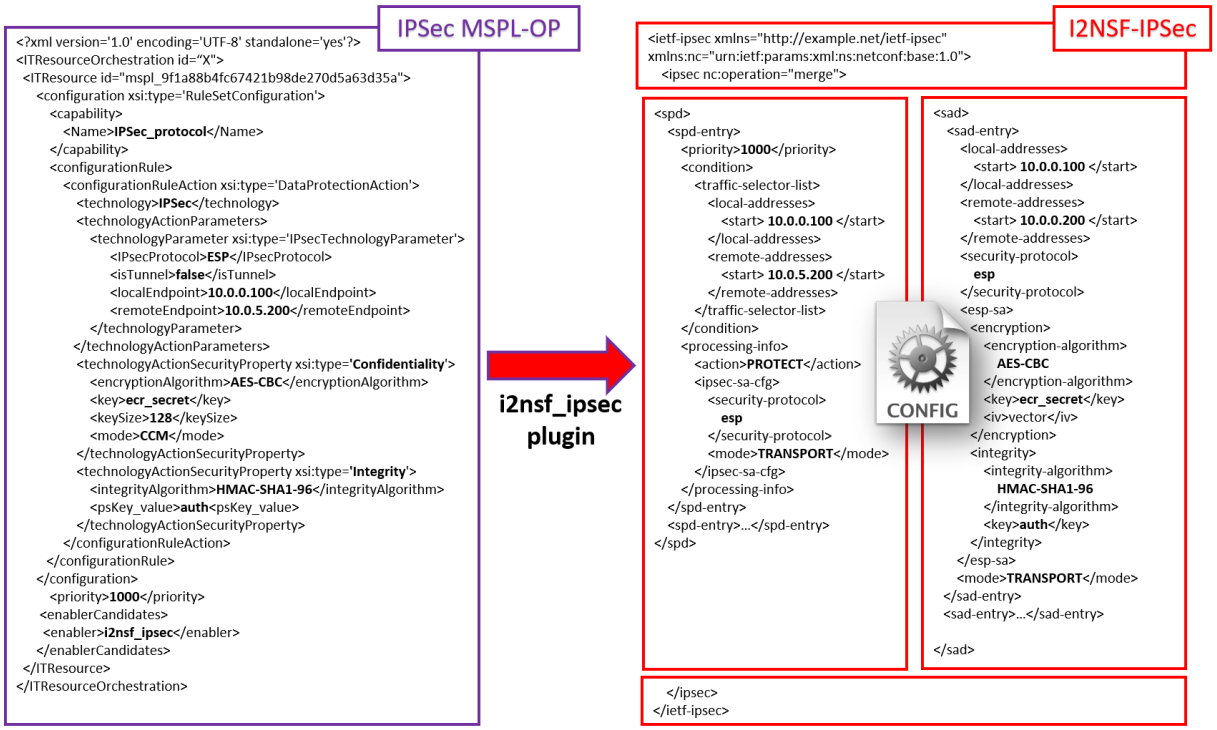

Figure 5: Policy Translation example

It must be noted that it is important to consider that the final configurations provided during the translation process are not only intended for configuring the final devices or security tools but also other platforms and controllers. For instance, in order to interoperate with open-source VIM orchestrators, the MSPL-OP can be translated to a language supported by them such as TOSCA (Topology and Orchestration Specification for Cloud Applications) supported by ONAP. TOSCA is a standard defined by OASIS [6] that allows specifying the topology of the network services and components, relationships between them, and the management processes.

The TOSCA meta-model is based on service topology templates that describe cloud workloads corresponding to a graph of node modelling components and of relationship modelling relations between them. The node and relationship types also define the life-cycle of operations to implement the behaviour a VIM orchestration engine will invoke when instantiating a service template. For instance, a *create* operation can indicate to the VIM orchestrator to create an instance of a component, a *start* or *stop* operation can serve to trigger these events. The operations are grouped as scripts that specify the actual behaviour required. The VIM orchestration will use these scripts to manage the components during runtime. The relationship between components serve to order the component instantiations.

The TOSCA basic profile defines node and relationship types that need to be supported, e.g., Compute node, Network node, and Database node types. Additional types can be defined for customizing and extending existing types. The policy type and trigger grammars are depicted in *Figure 6*.

```
<policy_type_name>:
  derived_from: <parent_policy_type_name>
  version: <version_number>
  metadata:
    <map of string>
  description: <policy_description>
  properties:
    <property_definitions>
  targets: [ <list_of_valid_target_types> ]
  triggers:
    <list_of_trigger_definitions>
```

```
<trigger_name>:
  description: <trigger_description>
  # TBD: need to separate "simple" and "full" grammar for event type name
  event: <event_type_name>
    type: <event_type_name>
  schedule: <time_interval_for_trigger>
  target_filter:
    <event_filter_definition>
  condition: <attribute_constraint_clause>
    constraint: <constraint_clause>
    period: <scalar-unit.time> # e.g., 60 sec
    evaluations: <integer> # e.g., 1
    method: <string> # e.g., average
  action:
    <operation_definition>
```

Figure 6: Policy type and trigger TOSCA grammars

TOSCA includes policy types related to placement, scaling, updating, and performance, and can be extended from the *tosca.policies.Root*, the default TOSCA policy type from which all other policy types are derived. Thus, the TOSCA model can be extended with security policies that can add security aspects, such as regulatory compliance, data retention, isolation, and security-focused placement. An example described in [10] is depicted in *Figure 7*.

TOSCA policy type in YAML of a security policy for resource placement

```
ares.5g_ns.policies.security.Placement:
  derived_from: tosca.policies.Placement
  description: >
    Restricts the placement of resources by
    location and required certification of the
    resource provider.
  properties:
    location:
      type: list
      entry_schema:
        type: string
    certification:
      type: list
      entry_schema:
        type: string
```

TOSCA policy template in YAML for our security-focused placement policy

```
certified_placement_in_germany_policy:
  type: ares.5g_ns.policies.security.Placement
  description: >
    Allows only placements where resources are
    located in Germany and the provider is
    certified for IT-Grundschutz and BSI C5.
  properties:
    location: [ Germany ]
    certification: [ IT-Grundschutz, BSI C5 ]
```

Figure 7: Security-focused placement example

Thus, the security policies can correspond to a TOSCA extension. They can be event-based but also related to access control, life cycle, etc. Translating MSPL to TOSCA provides a form understandable by the VIM orchestrator or tools able to perform the policy enforcement. The following simple example (*Figure 8*) illustrates how HSPL is instantiated to MSPL and translated to TOSCA.

- HSPL:

  Alice is not authorized to access Internet
  (time period, {08:00-18:00 GMT+1})

- **MSPL**:

```
<ITResource ID="MSPL_f9b27422-15b3-4bb5-ad21-3e08af5b1a1c"...>
<configuration xsi:type="RuleSetConfiguration"...>
        <capability> <Name>Timing</Name> </capability>
        <capability> <Name>Filtering_L4</Name> </capability>
        <defaultAction xsi:type=\"FilteringAction\"> <FilteringActionType>ALLOW</FilteringActionType> </defaultAction>
        <configurationRule>
                <configurationRuleAction xsi:type=\"FilteringAction\">
                        <FilteringActionType>DENY</FilteringActionType>
                </configurationRuleAction>
                <configurationCondition  xsi:type=\"FilteringConfigurationCondition\">
                        <isCNF>false</isCNF>
                        <packetFilterCondition>
                                <SourceAddress>10.0.0.1,</SourceAddress>
                        </packetFilterCondition>
                        <timeCondition> <Time>08:00-18:00,</Time> </timeCondition>
                </configurationCondition>
        </configurationRule>
        <Name>MSPL_f9b27422-15b3-4bb5-ad21-3e08af5b1a1c</Name>
</configuration>
</ITResource>
```

- **Tosca**:

```
tosca.policies.mspl:
  derived_from: tosca.policies.Root
  version: 0.1
  description: conversation mspl in tosca policies
  properties:
    <property_definitions>
  targets:
      SourceAddress: 10.0.0.1
  triggers:
      block_internet
        condition:
          time: 08:00-18:00 GMT+1
        action:
          DENY
```

Figure 8: Example translation HSPL -> MSPL -> TOSCA

3 Security management

The formalisms presented in the previous Section 2 allow define the policies that can be acted on, but, for this, one needs to understand and specify how the security breach detections need to be managed as well as what strategies need to be performed for mitigating, countering or even preventing them. In this Section we first give an overall view of the management process, and then present the management of SSLAs, the role of the security orchestrators, and the management of network slices. We then present how to deal with conflicts between the platform and services, and the security functions. Finally, we present the threat assessment process.

3.1 Overall security management process

The complexity of the 5G architecture has introduced disruptive concepts and technologies for which the resulting risks are yet not fully known, e.g., softwarisation, virtualisation, and cloudification. These innovative technologies have positively impacted the flexibility and adaptive capabilities of networks. Nevertheless, security management requires a significant level of situation awareness and is sensitive to the increased complexity and dynamicity introduced by the novel concepts.

The integration of disruptive technologies requires, on the one hand, to secure these technologies. On the other hand, these technologies should be applied for security purposes in order to obtain benefit from them and reach consistency with system properties. In order to reach the required scalability and dynamism levels of security, security services should follow, as much as possible, the “as-a-service” model that includes virtualisation and software-defined control. Management and control of security should remain aligned to these innovative paradigms to obtain what can be called smart orchestration that considers chaining and AI as the enablers for providing an intelligent distribution of security functions across the systems and domains. Dynamic and intelligent orchestration is essential for implementing a “protect-detect-react” loop that ensures compliance with security policies and Security Service Level Agreements (SSLAs), optimizes the detection of anomalies and known attacks, and dynamically triggers the required mitigation actions.

The deliverable D2.2 [2] describes a fully automated end-to-end smart network and service security management framework across multi-domains by detailing the main functional blocks and their role in enabling intelligent closed-loop security operations. Here, we briefly describe the building blocks related to the security management that are essential for the automation of the closed loop. In particular, the enablers that provide dynamic orchestration of security and ensure that the provided security is compliant with the expected SSLAs, security policies and regulatory requirements. The AI security management enablers will be detailed in D3.3 and D3.4 [1].

The complete protect-detect-react closed loop automation can be seen as follows:

(1a) – The initial E2E SSLA / Security Policy can be defined by the operator’s security administrator or an external entity (e.g., OTT) requesting secure services from the operator.

Or

(1b) – The Security Policy can also be received from the E2E Decision Engine.

(2) After checking for potential conflicts and/or impossibility of fulfilment, the E2E Policy & SSLA Management module communicates the requested E2E SSLA/Security Policy to E2E Security Orchestrator for enforcement.

(3 – 4) The E2E Security Orchestrator relies on E2E Policy & SSLA Management services to refine the E2E SSLA /Security Policy, providing medium-level description of the E2E policy and its mapping to domain-level policies.

(5 – 6) Each domain receives its corresponding domain-level policy that will first be checked for potential conflicts and/or impossibility of fulfilment by the Policy & SSLA Management module before being transmitted to the Security Orchestrator for enforcement.

(7 – 8) – The Security Orchestrator relies on Policy & SSLA Management services to refine the domain-level policy into low-level actions that can be enforced on the domain infrastructure.

(9 – 10) Depending on the situation (e.g., the security awareness provided by monitoring functions), the security policies can be enforced directly on the resources (e.g., configuration of new rules on a deployed virtual Firewall, load balancer, traffic splitter, or monitoring probe) or via the Unified Security API offered by the network/service orchestration services (e.g., instantiation of a new security VNFs, service chaining, reconfiguration of a VNF or network Slice).

3.2 SSLA Management

SSLAs (Security Service Level Agreements) have been defined and are being used in two different approaches. One is to manage the security requirements in a Slice, and the other is to assess the security properties using monitoring techniques during runtime, called RT-SSLAs. The objective is to manage the security requirements, defined by the SSLAs, during the full life-cycle of a Slice by: a) gathering the verticals/end-users security requirements; b) deploying the necessary security controls to enforce the agreed SSLA by enriching or configuring the services of the Service Providers (SPs) services; c) real-time assessment of RT-SSLAs using monitoring techniques to detect that the security functions are working as expected and that there are no security breaches; d) detecting violations in security provisioning level based on an analytic engine and notifying both end-users and SPs; and e) enabling the automation of reaction strategies in real-time to adapt the provided level of security or to trigger proper countermeasures.

In order to automate the security life cycle of a Slice, a machine-readable SSLA format is adopted based on the SPECS [7] SSLA model that is extended to support slicing and security orchestration in the 5G context. This extension introduces security-related information allowing to specify the following Sections in a Slice term description:

- Slice resource providers that describe the available infrastructure of the resource providers (appliances, networks, etc.);
- Security capabilities required in a Slice. A capability is defined as a set of security controls. In our case, the NIST's Control Framework [8] is used to specify these security controls;
- Security metrics referenced in the Slice service properties and used to define Security Service Level Objectives (SLOs) in the guarantee terms Section. A metric specification includes information about it and also information to process the SLOs, such as the metric name and definition, its scale of measurement, and the expression used to compute its value. Section 4.2 provides more details on the SSLAs defined for real-time assessment.

The SSLA model is an extension of the WS-Agreement model standardized by the Open Grid Forum (XML based and formally defined by an XML schema). The data model is depicted by the following Figure 9.

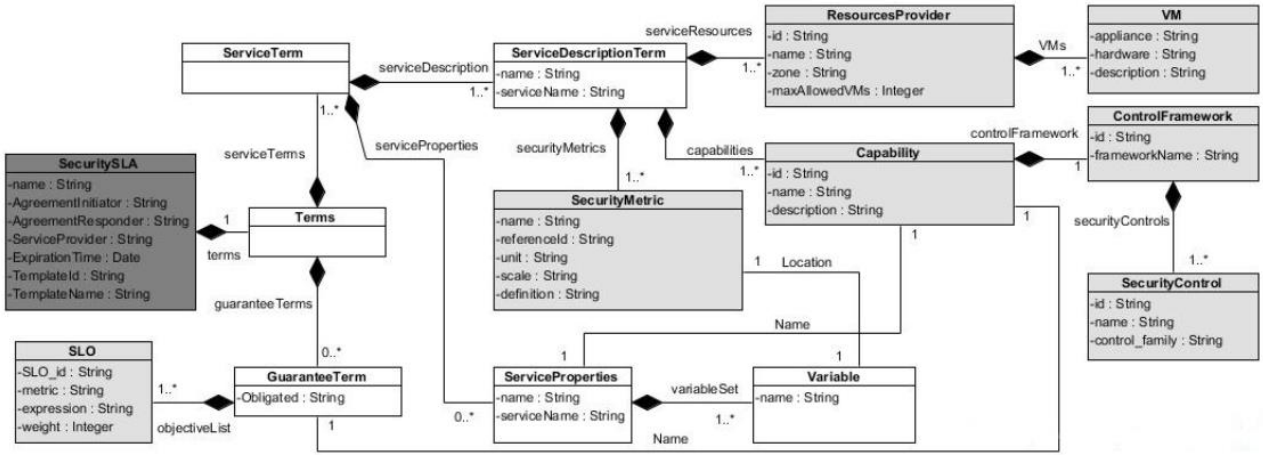

Figure 9: WS-Agreement model

For example, if we consider a web application server with DoS mitigation capability, we can consider the Security Controls from the NIST SP 800-53 for such capability shown in Figure 10.

Control Framework	Control Family	Security Control
NIST-800-53r4	CA - Security Assessment and Authorization	CA-7 Continuous Monitoring
	IR - Incident Response	IR-6 Incident Reporting
	SC - System and Communication Protection	SC-5 Denial of Service Protection SC-7 Boundary Protection
	SI - System and Information Integrity	SI-3 Malicious code protection SI-4 Information system monitoring SI-7 Software, firmware and information integrity

Figure 10: NIST SP 800-53 Security Controls

And the following Security Metrics shown in Figure 11.

Metric Name	Metric Description
Detection_latency	It represents the time interval between the first symptom of a (detected) attack and the generation of a message event
False_positives	It reports the number of detected false positives in a predefined time interval
Detected_attacks	It reports the number of attacks detected in a predefined time interval
Attack_Report_Max_Age	It represents the frequency of attack report generation

Figure 11: Security Metrics

An example of Security Mechanism may be a combination of the *OSSEC* open source HIDS solution for the DoS detection and a custom script for the actual mitigation measure that, for instance, shuts down attacked ports or blacklists source IP addresses.

In order to assess the security properties using monitoring techniques during runtime, low level security property rules can be specified and/or derived from the MSPLs previously discussed. This is presented in Section 4.

SSLAs may be managed by the SSLA manager presented in Section 62

3.2.1 SSLA Refinement

The goal of SSLA refinement is to tune the input policy (MSPL) to the Security Orchestrator (SO) - described in the next Section - according to the SSLA, so that the security system (or system of systems) deployed by the SO meets the SLOs (Service Level Objectives) defined in the SSLA, and especially that

the proper monitoring enablers are in place to detect any RT-SSLA violation at runtime in case a change of the state of the system occurs.
The refinement process assumes that there is a catalogue of available Services on which SSLAs can apply (referenced in the *ServiceDescriptionTerm* element); and it also assumes that for each such Service, there exists a corresponding MSPL template, pre-defined by the Service Provider (SP). Each *capability* defined in the SSLA should match a *capability* of one of the <*ITResource*>s in the MSPL policy. Indeed, some initial policy (MSPL in this case) information is necessary in addition to the SSLA as primary input. Because on the one hand, the SSLA defines agreements only on high-level service properties (functional or non-functional) that the SP exposes to the Service Consumer (SC), i.e., the ones that the SP is willing to negotiate with the SC. In fact, there will be very likely only a subset of all negotiable properties in a SSLA because the SC cares only about some of them usually. Also, the SSLA does not care about the *how*, i.e., *how* the SLOs will be met, implemented or monitored. On the other hand, the MSPL policy gives details about the *how* by defining the IT resource or the composition of IT resources (with dependencies, etc.) - security enablers - that the service is composed of, with an exhaustive set of all the configuration properties required to configure each security enabler, regardless of whether the properties are present in the SSLA or not.

At the abstract level, the SSLA refinement process has two effects on the MSPL policy:

1. Restrict the range of possible values of properties in the MSPL. For example, the MSPL DataProtection policy may allow ECDSA keys of any ECC curve and any length for authentication, but the SSLA requires specific NIST curves and key lengths of 256 bits and above to comply with a specific regulation. The SSLA may fix a single value in some cases, e.g., the AES key size must be 256 bits.
2. Add the necessary monitoring resources (<ITResource>s) to the MSPL in order to monitor the SSLA metrics with regards to the SLOs, especially for non-functional properties which do not map to any configuration property in the MSPL.

Since SSLAs come in XML form (based on WS-Agreement specification) the refinement could be supported by XSLT since this is the standard way of transforming XML data into another text form like MSPL (which is also defined in XML). When writing the MSPL template, the author must identify whether and how each SLO in the SSLA can be translated into one or more configuration properties in the MSPL *<ITResource>*s, and write the corresponding XSLT rule in the template, in place of the values of these configuration properties. However, the SSLA refinement process is not only about text transformation. To fill those MSPL fields that cannot be inferred directly from the SSLA (e.g., context specific values), the process will need to retrieve the required information from the data services.

The following example shows an **excerpt** (simplified) of the MSPL template (XSLT stylesheet) defining a one-to-one mapping between the SSLA-defined minimal AES key size for the pre-shared key, actually defined in an SLO *min_dtls_aes_key_size* (referring to a metric *aes_key_size*) associated to a *Security Gateway* Service with *DTLS* capability; and the MSPL-defined value of *a Confidentiality* property of the DTLS proxy (ITResource)'s *DataProtection* policy:

```
<?xml version="1.1" encoding="UTF-8"?>
<xsl:stylesheet version="3.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
        xmlns:mspl="http://modeliosoft/xsddesigner/a22bd60b-ee3d-425c-8618-
beb6a854051a/ITResource.xsd"
        xmlns:xslt="http://www.w3.org/1999/XSL/Transform"
        xpath-default-namespace="http://www.w3.org/2005/xpath-functions"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<xsl:template match="/">
...
<!— Only the part in red is a change to this MSPL template. —>
<ITResourceOrchestration ...>
        <ITResource ...>
```



```
            <configuration xsi:type='RuleSetConfiguration'>
                <capability>
                  <Name>DTLS_protocol</Name>
                </capability>
                    <configurationRule>
                            <configurationRuleAction xsi:type='DataProtectionAction'>
                                    <technology>DTLS</technology>
                                      <technologyActionSecurityProperty
                                xsi:type='Confidentiality>
                                                <encryptionAlgorithm>AES</encryptionAlgorithm>
                                      <!— Extract the Key size value from the input SSLA
                                document (XSLT input) here, more precisely the SLO with attribute
                                SLO_ID min_dtls_aes_key_size —>
                                      <keySize><xsl:value-of
                                      select=”//specs:SLO[@SLO_ID=’min_dtls_aes_key_size’]/spec
                                      s:SLOexpression/specs:oneOpExpression/specs:operand”
                                      /></keySize>
                                  ...
                                      </technologyActionSecurityProperty>
                                </configurationRuleAction>
                        ...
                        </configurationRule>
            ...
            </configuration>
        </ITResource>
...
</ITResourceOrchestration>
...
</xsl:template>
</xsl:stylesheet>
```

Figure 12: Example of an MSPL template defining the minimal AES key size

The part in red creates a value from the value of a node in the SSLA XML document using XPath expression (look for the SLO element with SLO_ID *min_dtls_aes_key_size*, get the SLOexpression/oneOpExpression/operand child element in it).

The following example (*Figure 13*) shows a more complex mapping between a SSLA-defined high-level security property *AAL* (*Authentication Assurance Level* as defined in NIST SP 800-63) and a lower-level configuration of the *Authentication* part of the *DataProtection* policy in the MSPL, depending on the value of the *AAL* (showing only the relevant part of the MSPL template), i.e., if AAL is lower than 3, we use simple pre-shared key for authentication, else we use public keys (or certificates) with HSM-generated/owned and PIN-protected private key.

```
<configurationRuleAction xsi:type='DataProtectionAction'>
<technology>DTLS</technology>
<xsl:variable name="aal "
select="//specs:SLO[@SLO_ID=’dtls_aal’]/specs:SLOexpression/specs:oneOpExpression/specs:operan
d”" />
<technologyActionParameters>
...
        <authenticationParameters>
```

```
                <xsl:choose>
                 <xsl:when test="aal &lt; 3">
                  <psKey_value>mypk</psKey_value>
                  ...
                 </xsl:when>
                 <xsl:otherwise>
                  <keyId>dtls_auth_key</keyId>
                  <hsmProtection>true</hsmProtection>
                  <!— 4-digit PIN —>
                        <pin><xsl:value-of select=”ceiling(1000 + (8999 * random-number-
                        generator()?number))” /></pin>
                 </xsl:otherwise>
                </xsl:choose>
        </authenticationParameters>
</technologyActionParameters>
...
<technologyActionSecurityProperty xsi:type='Authentication'>
                <xsl:choose>
                 <xsl:when test="aal &lt; 3">
        <peerAuthenticationMechanism>preshared_key</peerAuthenticationMechanism>
                 </xsl:when>
                 <xsl:otherwise>
        <peerAuthenticationMechanism>raw_public_key</peerAuthenticationMechanism>
                 </xsl:otherwise>
                </xsl:choose>
</technologyActionSecurityProperty>
</configurationRuleAction>
```

Figure 13: Example of a mapping between a high-level security property AAL and a lower-level configuration of the Authentication part of the DataProtection policy in the MSPL

Although XSLT is recommended here as template language and requires an XSLT engine, other non-XSLT-based yet XML-friendly template engines might be used for simple use cases (Freemarker, Velocity, etc.).

Certain service properties in the SSLA - metrics - are non-functional and may not correspond to specific configuration properties of the security (enforcement) enablers (ITResources in the MSPL), in which case the MSPL template should also include *<ITResource>s* for monitoring enablers that are capable of measuring these metrics regularly (frequency to be defined) and detect a deviation from the SLO. For example, in a SSLA that refers to a *Security Gateway Service* with *DTLS* capability with support for PSK (pre-shared key) and certificate authentication, such a metric could be the *mean time to react to (remediate) a key compromise*; in more concrete terms, how much time does it take to notify/alter whomever should be informed, and change (remove, re-distribute, etc.) pre-shared keys or private keys (and revoke certificates) and the DTLS communication to be re-established with the new keys, after a key compromise is detected.

3.3 Security orchestration

The overall 5G architecture relies on the composition of modular systems and services that must be mission-aware but also security-aware. For this, the security orchestration is needed to meet user/client expectations, policies and regulations.

Security policy orchestration is raising numerous challenges. One challenge is related to the definition of the policies and SSLAs themselves. Particularly, it is important determining how to declare those policies and SSLAs, how to compute compliant resources and services respecting the policy constraints. Another challenge is related to the deployment of the security enforcement strategies, i.e., determining how to schedule and guarantee the availability of the security functions and associated resources. Furthermore, interoperability and combination of the various authority perimeters is required to ensure global security. On the other hand, smart protection deployment, smart detection deployment, smart remediation strategies will greatly benefit from the flexibility offered by orchestration.

3.3.1 Security orchestration capabilities

As detailed in Section 5.1 of D3.1 [11] various orchestration solutions are provided by management platforms for clouds and networks, but there is a need for a dedicated Security Orchestrator for enabling support of a fine-grained and end-to-end security deployment and configuration. The capabilities of a such orchestrator are the following:

- The ability to provide a holistic view on end-to-end security at a vertical level (for example, security deployment and configuration at the network l, IT or application levels).
- Full automation of the deployment control and configuration of all security functions in a highly dynamic environment.
- The ability to interact with each level of orchestration according to the security orchestration needs.
- The ability to align the security policies in an automated way inside of a domain and inter-domain context.
- The consideration of the specified SSLAs to orchestrate security according security policies and requirements.
- Full automation of the selection of security services/VSFs to be orchestrated based on an automated catalogue.

3.3.2 Security orchestration based on SSLAs

The SSLA selection process (as shown in Figure 14) consists of retrieving, for each capability described in the SSLA file, a list of enablers from the catalogue supporting all the metrics marked with a "HIGH" priority. Indeed, since the metrics can be associated with three different priority levels (“HIGH”, “MEDIUM” or “LOW”), it seems logical to select only the enablers supporting all the metrics with the highest priority level. Nevertheless, we have chosen to classify the enablers according to the other supported metrics, favouring the greatest number of metrics with “MEDIUM” priority implemented, then the greatest number of metrics with “LOW” priority. The Figure shows the result of the selection before the enablers are sorted.

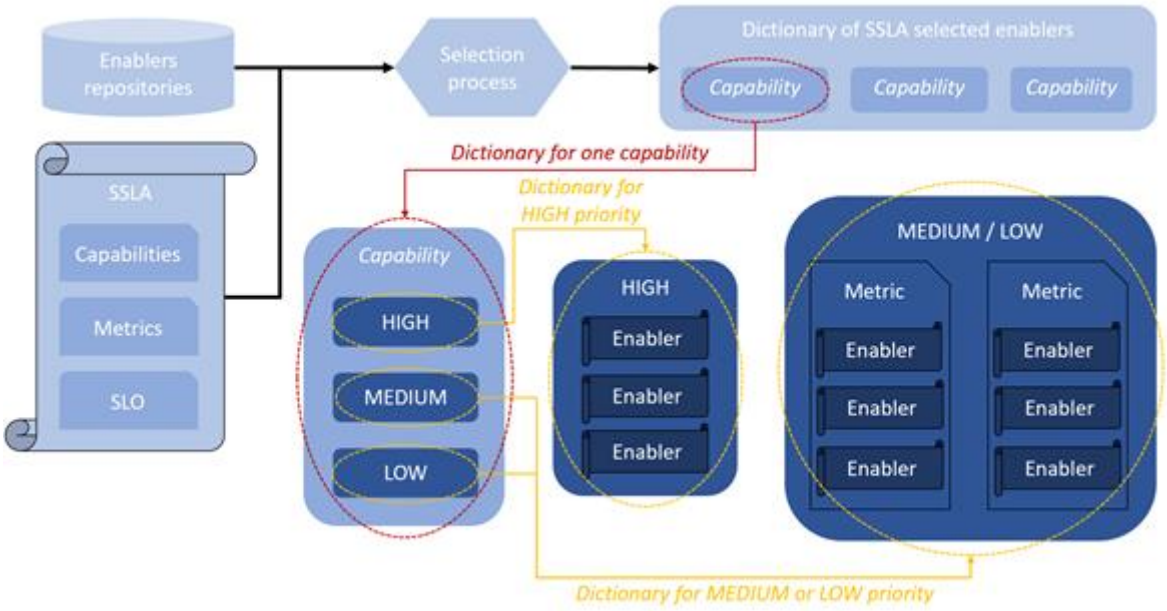

Figure 14: SSLA-based enabler selection process

Thus, this first selection remains effective, in the sense that only the enablers that comply correctly with the critical points are chosen without becoming too drastic and filtering only the enablers which implement all the metrics correctly, in which case it would not be obvious to always find it. Indeed, we want to avoid cases where the result of the selection would be an empty set of enablers.

In addition, we can notice here that only the capabilities (as defined by NIST) are considered. A more exhaustive work taking into account the security controls (as also defined by NIST) will have to be developed at the future using a new model.

3.3.3 Security orchestration based on Security Policies

Policy-based security orchestration allows orchestrating security requirements from well-defined security policies. This approach allows managing security on complex and heterogeneous 5G infrastructures by homogenizing final configurations in security policies that provide multiple level of abstractions as the ones provided in Section 2. Besides, a well-defined policy-based approach significantly improves the consistency of the system through different conflict detection and dependencies processes. Since the INSPIRE-5GPlus architecture extends the ZSM approach, it must be considered not only with respect to intra domain policy-based orchestration, but also with respect to E2E policy-based security orchestration.

Figure 15 shows the main workflow for E2E Orchestration, which includes inter/intra domain orchestration. When the E2E orchestrator receives a HSPL Orchestration Policy (HSPL-OP) (1), a preliminary high-level conflict detection process is performed to verify if there is any issue at E2E level (2). If the high-level policies composing the HSPL-OP policy do not generate conflicts, the E2E orchestrator identifies the required capabilities with the support of the policy framework (3), as well as the involved domains with the support of the System Model which is mapped to data services (4) that contain the required information like the SMD where the affected devices/services are connected to. Once the involved domains have been identified, the E2E orchestrator retrieves the endpoint information for each one of them (5). If the capability requires some common parameters (e.g., channel protection capability requires that both end points implement the same channel protection technology), the E2E orchestrator requests the available parameters for each involved domain (6) and looks for a common solution (7). If there is a common solution, the E2E Orchestrator requests a customized HSPL-OP refinement to the Policy Framework (8). This refinement process generates MSPL Orchestration policies from HSPL Orchestration (9) policies as it has been explained on Section 2.3.

Depending on the implementation, some steps of the E2E policy refinement, part of the E2E Orchestration process, could be performed by the policy refiner (inside of the Policy Framework) to lighten the load of the E2E Security Orchestrator. Once the E2E Security Orchestrator receives the MSPL-OP, it orchestrates an enforcement plan. To this aim, it identifies the enforcement domain for each security policy since the enforcement domain could be different from the targeted domain (e.g., a filtering policy for a specific subject could be enforced in the transport domain instead of in the management domain where the subject belongs to). Then, it sorts the plan according to the security policies' priorities, managing the dependencies if any (10). Those policies that contain unsolved dependencies are queued until the proper event triggers the enforcement, whereas those policies that are ready to be enforced are introduced into the enforcement plan ordered according to the defined priorities. Finally, the E2E Security Orchestrator requests the policies' enforcement to the different Management Domains according to the enforcement plan.

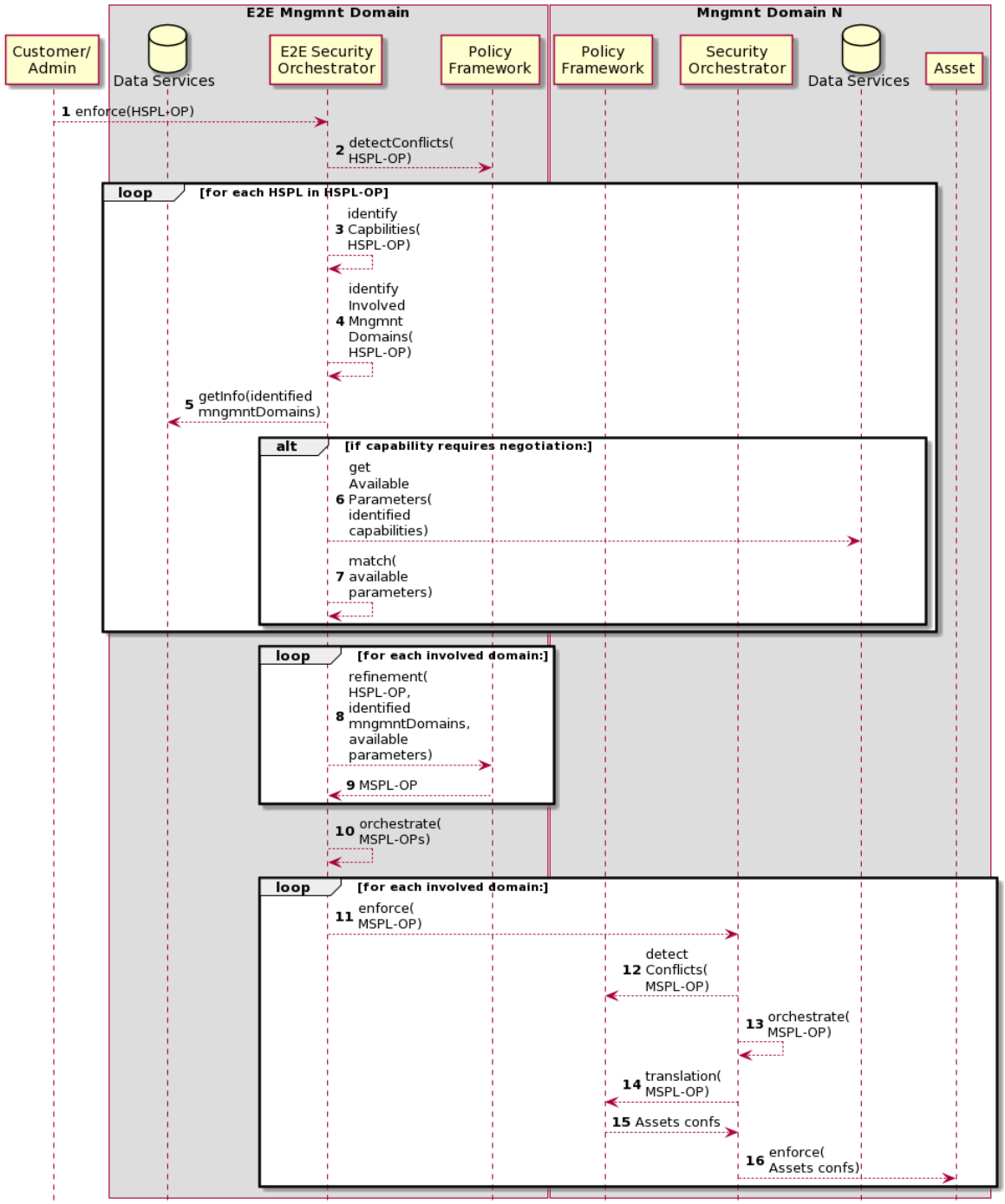

Figure 15: E2E orchestration workflow

When a Management´s Domain Security Orchestrator receives a MSPL-OP enforcement request (11), it verifies that the new policies will not generate any conflict according to the current domain policies and infrastructure (12). In that case, the Security Orchestrator orchestrate the enforcement of the MPLS-OP (13), this is, it decides the best asset to perform the enforcement. To this aim, the orchestration algorithm retrieves the available candidate assets from the domain data services, and starts an allocation optimization process which decides a suitable policy enforcement point according to the available asset plug-ins (that contains the logic to translate MSPL-OP to specific configurations for the specific asset), the security policy information, data services information, and the allocation algorithm. The latter is also considered since different allocation algorithms can be provided to cover different approaches. For instance, if the infrastructure does not have NFV-MANO features, or new deployments are restricted an algorithm which only looks for enforcing policies in the most suitable allocation place among the already deployed/existing assets can be selected, otherwise, if the infrastructure allows dynamic deployments, advanced optimization algorithms such as weight-based, scored, greedy, Mixed-Integer Linear Programming, ant colony-based or deep-learning-assisted can be applied. It is important to highlight that during the orchestration process, multiple conflict verifications can be performed for ensuring that selected security enablers and allocation resources are fully compliant with the security policy requirements More information regarding intra domain policy orchestration and enforcement can be found in [5]. After the orchestration process, translation process (explained in Section 2) and enforcement process (explained in Section 4.3) are performed. It is important to highlight that part of the process will not only be executed proactively but also reactively. For instance, as part of a countermeasure or when an event solves any policy dependency that triggers a reactive policy enforcement. Among this Software Defined process, the APIs described in Section 5.7 are used for communicating the enablers. In particular, the E2E SO receives the HSPL-OP at **e2eheservice** and it uses **h2eservice** at the E2E Policy Framework to perform the refinement. Once MSPL-OPs are ready, the E2E SO uses the **meservice** to perform the enforcement at the SO SMD. The SMD SO uses the **mcdtservice** to detect conflict and dependencies and then the **m2eservice,** both at the Policy Framework, to perform the translation process to final-asset configuration.

3.4 Secure Slice Management

Network Slicing have been studied during the last years as it is seen as a key concept for network management. While there are multiple works done on the design, development and management of Network Slices, the security on Network Slicing is a path to be researched with many possibilities. INSPIRE-5Gplus is investigating two aspects related to the security and the management of network Slices: Security SLA and slice brokering.

3.4.1 Security SLA

The use of SSLAs allows defining a set of objectives to be monitored that should lead to the detection of attacks targeting the deployed network Slices and rise alarms to notify what is happening and, if necessary, apply the appropriate actions to counteract to the current attack and prevent similar ones or at least reduce their negative effects.

Based on some of the research done [12] in the context of the INSPIRE-5Gplus project, this Section presents an initial draft of how the deployment relationship of Network Slices and SSLA should be. And once deployed, the steps to perform for the SSLA monitoring. In order to apply these two actions (deployment and monitoring), two data objects are necessary:

1. NEtwork Slice Type (NEST): describes how a set of Network Services are interconnected among them to create the logical network for to offer the requested service.
   The NEST complements the slice creation request that a user sends to the Slice Manager to deploy a new slice. In this context, GSMA has published the NG.116 – Network Slice Template v2.0 specifications [9] that define the Generic Network Slice Template (GST), a set of attributes

that characterize a network slice type. GST is generic and is not tied to any specific network deployment. In effect, the NEST is a GST filled with values in order to fulfil a given set of requirements derived from a network slice use case. The Slice Mapping process then parses the NST, which define the Network Slice Subnet Instances (NSSIs) that shall be deployed as part of the slice.

2. SSLA descriptor: defines the set of Key Performance Indicators (KPIs) and the Service Level objectives (SLOs) that define the QoS for the deployed Network Slice.
   Having these two data objects, the process to deploy a Network Slice with a set of monitored security KPIs should follow the steps described in the next procedure:

   1. To request to the NFV Orchestrator (NFVO) the instantiation of the services using virtual elements over a set of computing resources
   2. To request to the SDN controllers the creation of the connectivity services across networking resources.
   3. To request the SSLA configuration, this action has a set of sub-steps:
      a. The deployment of its associated Security Functions (SFs) to add the security expected.
      b. The configuration of the SFs to deliver the necessary information to be monitored.
      c. The configuration of the monitoring system to receive the data from the SFs, analyse it and decide if an SSLA is being respected.

   With the Network Slice deployed, the next phase on its life-cycle is the monitoring of those KPIs defined in the SSLA. This procedure follows the next steps:

   1. Based on their configuration, the SFs gather the information necessary for the monitoring system.
   2. The gathered information is compared with the SLO and metrics (i.e., they are the threshold references). Here two options are possible:
      a. The SSLA is NOT violated, so no control and management action is required.
      b. The SSLA is violated, it is necessary to apply a solution to bring the monitoring KPIs within the accepted threshold values defines in the SSLA. At this point different options are available to solve an SSLA violation; re-configuring the SFs, apply a scale-in/out or scale-up/down of the resources, etc. The final solution depends on the set of policies available and associated to solve each SSLA.
   3. Finally, once the SSLA is solved, the Network Slice monitoring should be back at a normal status until its next violation or the end of its life-cycle.

As a result, the minimum set of operations that need to be supported by the Slice Manager are the following:

- Network Slice Instance Creation
- Network Slice Subnet Instance Creation
- Network Slice Instance Termination
- Network Slice Subnet Instance Termination
- Network Slice Instance Modification
- Network Slice Subnet Modification

Based on the above, the Slice Manager should provide the following features:

- **Slice Mapping:**
  This process selects the infrastructure resources for a new slice based on the NEST. It should also enable NSSIs to be shared among slices that run concurrently.

- **Slice Monitoring:**
  This process monitors the health and status of the deployed network slices and sends status messages to the MANO components.

- **Interface with MANO Components:**
  The Slice Manager should support essential connectivity with the MANO components.

In the context of the INSPIRE-5Gplus project, a new enabler called “Secured Network Slices for SSLA” is designed and being developed in order to associate SSLAs at a Network Slice level and so, to apply certain security requirements to the deployed service. This enabler is placed in the E2E Security Management Domain within the INSPIRE-5Gplus HLA.

This enabler aims to take care of the following functions: a) to enforce the association of Network Slices and SSLAs by offering the available SSLAs in the SSLA manger together with the NSTs in the DataBase (DB), b) to manage the deployment and termination of Network Slice Instances (NSIs) based on the available NSTs by requesting the corresponding slice-subnets with SSLA to the Security orchestrator (SO) through MSPL policies.

As illustrated in Figure 16, this enabler is composed by seven modules:

- **Main:** contains the API and the configuration files to set up the enabler. A more detailed description of the API can be found in Section 62.
- **Secured Slices Manager:** manages the deployment of the Secured NST with the associated SSLA.
- **Slice_2_mspl:** takes care to convert the requests NST+SSLA data object to an MSPL object for the SO.
- **Secured NST:** Database (DB) with the available Network Slice Templates (NST) to be deployed.
- **Secured NSI:** DB with the Network Slice instances (NSI) deployed/terminated.
- **SSLAMngrmapper:** in charge to pass any associated SSLA request to the SSLA Mngr.
- **SOMapper:** in charge to pass any associated MSPL request to the SO.

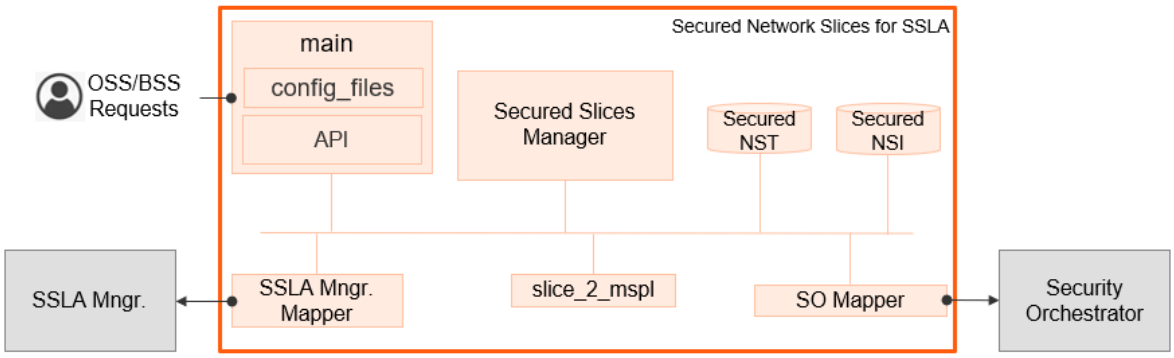

Figure 16: "Secured Network Slices for SSLA" internal architecture.

3.4.2 Slice brokering

In 5G, network Slice brokering (NSB) is introduced as a new business model for dynamic network sharing wherein a logically centralized entity named as Slice broker governs the secure and privacy preserved resource trading between network operators/service providers at one end, and multiple network tenants at the other end. Apart from facilitating on-demand resource allocation, the Slice broker performs admission control based on traffic monitoring and forecasting, and mobility management based on a global network view.

The business model of NSB was initially proposed for on-demand multi-tenant network architecture. NSB is proposed as a centralized mediator and a controlling entity to provide admission control for the incoming resource requests. As the initial business scenarios, it has identified few main use cases such

as sharing a common RAN by multiple core networks, enhancing coverage by operator collaboration, sharing network coverage and spectrum, and sharing a common core network by multiple RANs. In addition to given scenarios, there are many other possibilities that require the sharing of cloud resources for storage and computational processes. Establishing network slices need to be performed dynamically and suitable to the service requirements. With respect to the 3GPP specifications, NSB is responsible for three main tasks: On demand resource allocation; Based on the traffic monitoring and traffic forecasting, controlling network admission; Allocate resources to the network tenants. In order to provide an uninterrupted service, NSB needs to maintain the synchronization with slice orchestration, slice managers as well as the service level agreement (SLA) manager.

When the consumer is eligible to acquire network and computational resources from multiple resource providers, the respective network slice should be formulated as federated slice. In such a scenario, the role of NSB is significant to act as a mediator and a controlling entity between the resource requester and multiple network operators or cloud service providers.

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

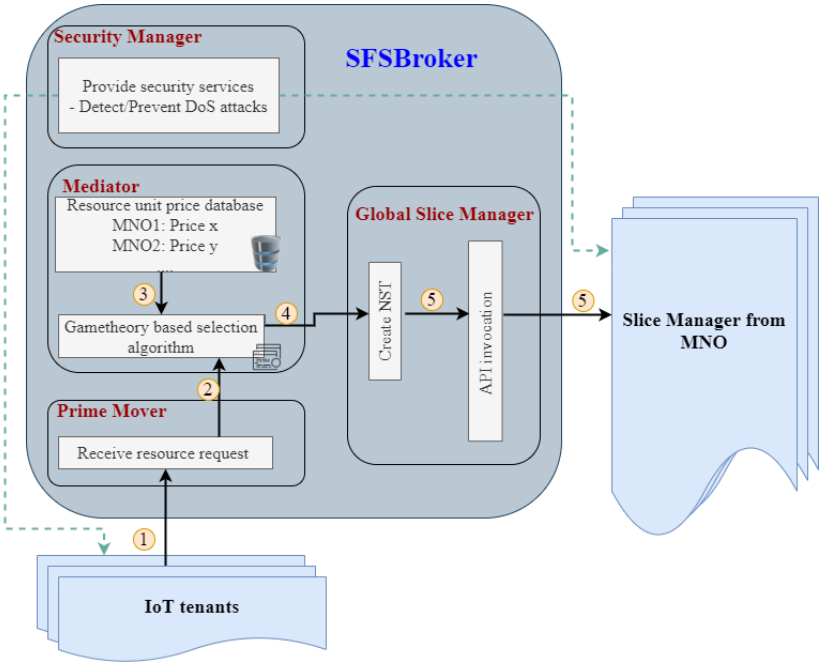

Figure 17: SFSBroker architecture

As shown in *Figure 17*, SFSBroker is running as a blockchain service and consisting of four modules (i.e., prime mover, mediator, global slice manager, security manager) which are implemented as separate smart contracts and integrated later. When a tenant resource request is sent (Step 1) to SFSBroker, it is accepted and processed to a network slice blue print by Prime Mover module of SFSBroker. Then the NS blue print is sent (Step 2) to Mediator which runs the slice selection algorithm based on the resource requirements, resource availability and unit price available in the database (Step 3).The data base is updated as an independent process from the resource requests, where the regular statistics regarding the resource availability and the prices are retrieved from there source providers or mobile network operators (MNOs).Once the Mediator decides the formation of federated network slice template (NST), it is created by the Global Slice Manager with the participation of the respective slice

managers of MNOs (Step 5). Security manager is responsible for identifying and mitigating the security attacks that may occur on SFSBroker.

SFSBroker blockchain service is implemented in hyperledger fabric and the selection algorithm is encoded in Java-based smart contracts. IoT tenants and MNOs are connected with SFSBroker using MQ Telemetry Transport (MQTT). Security Service Blockchain (SSB) [13] is integrated with Security Manager to verify the resource requests and MNO offers. APIs are described in Section 5.5.

3.5 Conflict and Dependency detection

This Section considers the risks of potential conflicts between inconsistent platform and service provisioning and security functions. Nowadays networks have become incredibly complex and diverse. In 5G, multiple of thousands of devices are interconnected belonging to different technologies, where an automatic and optimized response to events and the adaptation to system and device capabilities is required to ensure the proper behaviour of authorized devices. In this context, Inspire5G presents a policy-based framework to exploit the maximum performance of Software Defined Network and Network Function Virtualization. Fully aware of the context, the 5G infrastructure by using a well-defined policy procedure, is capable of self-healing, and self-repairing; detecting and mitigating possible inconsistencies in the system. Policies abstracts complexity from underlying layers, and allow the definition of rules that must be accomplished within and for the system to ensure the intended behaviour.

Prior to enforcing new security policies in the system, it is fundamental to detect any potential incompatibility, contradiction or dependency with the current status of the infrastructure. To this aim, the INSPIRE-5GPlus Policy Framework security enabler provides the Policy Conflict Detector module by extending previous approaches [15] to deal with conflicts at E2E Management Domain level as well as Management Domain level. This module implements a rule engine that processes the requested security policies against a set of rules and a base of knowledge (facts). Starting from a well-known set of rules, organisations can extend and customize them according to the conflicts and dependencies they want to detect. Besides, new rules could be added dynamically as part of reactive processes.

The following snippet shows a rule example (duties conflict). The logic rules are formed by antecedent, that represents rule conditions, and the consequent that will be performed when the antecedent is fulfilled. Thus, only when the conditions are met, the consequent is triggered. The rules make reference security policies statements as well as elements provided by data services such as information about the infrastructure. Besides, when a consequent is triggered, it is also considered as a new fact. This allows adding additional facts when the certain contextual and policy conditions are met. By following this approach, a set of semantic rules are defined to detect in the antecedents conflicting behaviours referring to facts in security policies and data services information. Thus, the rule-based reasoning process allows detecting automatically inconsistencies as well as detecting semantic conflicts in the managed system. For instance, the rule shown in the snippet verifies if a duties conflict is present. In this basic example, duties conflict will be triggered if channel protection and deep packet inspection want to be configured for the same subject and target.

MSPL(?m1) ^ MSPL(?m2) ^ m1.capability! = m2.capability ^ (1)

m1.capability = ”DPI” ^ m2.capability = ”DTLS” ^ (2)

m1.srcAddr = m2.srcAddr ^ m1.dstAddr = m2.dstAddr (3)

-> ”DutiesConflict(?m1, ?m2)”

Depending on the main goal of the conflict detection, this approach classifies the rules as semantic conflict detection or context-based conflict detection. From the one hand, semantic conflicts are

focused on semantic inconsistencies that can occurs between policies of the same orchestration policy (intra-orchestration) or between orchestration policies (inter-orchestration). From the other hand, context-based conflicts also consider the current context of the infrastructure. Following some conflict examples applicable for both E2E Security Management Domain and Security Management Domain are provided:

Semantic conflict examples:

- Redundancy Conflict: detection of 2 security policies with the same identifier or with the same behaviour. For instance, the same security policy is being enforced twice or two different security policies models the same security properties.
- Priority Dependency Conflict: detection of the emplacement of lower priority policy before of higher priority policy and detection of dependency between 2 policies with different priorities. For instance, a policy depends on another one with less priority.
- Duties Conflict: detection of possible harm from a desired policy enforcement to an already deployed policy or to another desired policy deployment. For instance, different policies with incompatible capabilities are trying to be enforced over the same target.
- Event and Policy Dependencies: requirement for the deployment completion of one or a set of previous policies. It identifies event and policy dependencies.
- Managers Conflict: detection of possible conflict between system administrators. For instance, different administrators try to enforce incompatible actions (e.g., Allow vs Deny).
- Override Conflict: detection of overriding behaviour of previous deployed policies. For instance, a new filtering policy is more permissive than other previously enforced.

Context-based conflict examples:

- Capability Missing Conflict: Detection of lack of capability by an asset to enforce the required policy. For instance, the required target or subject specified in the security policy is not able to provide the requested capability.
- Insufficient Resources Conflicts: Detection of lack or unavailability of resources to enforce the required policy. For instance, at Management Domain level this conflict will be generated when there are not enough resources to enforce the security policy (e.g., for new VNF deployment), whereas at E2E level it could be generated when common parameters are required at E2E level to configure common resources (e.g., channel protection tunnel), but there are no common parameters available.

Regarding the knowledge base, it needs to be constantly updated (reactively and proactively), generating facts from different sources such as monitoring tools and crawlers which provide different kinds of information about the infrastructure. Information about security policies and their status is also tracked for identifying successfully deployed policies and analysing events based on context, thus maintaining the system consistency. In this way, when a new security policy is received, the rule engine performs a rule matching against the current facts, to avoid conflicts and dependencies between the policies or between the policy and the status of the infrastructure. This process will return a list of tuples that specify the type of object involved and the type of issue (conflicts or dependencies). From the one hand, the Policy Conflict Detection procedure is coordinated with the rest of the system by sending the result back to the Security Orchestrator who verifies it, and notifies the conflicts to the entity who requested the policy enforcement. From the other hand, dependencies are analysed and queued until they are satisfied. Once these dependencies have been solved, security policies are ready to continue with the orchestration and enforcement processes. Conflict and Dependency detection procedure are performed at Policy Framework API, more precisely at mcdtservice point.

3.6 Threat Assessment

The threat assessment process is used to analyse the security posture of a system under analysis. In the context of INSPIRE-5Gplus, the threat assessment process is based on system analysis and representation with the use of a domain-specific language for 5G networks. The domain-specific language is used to express networks in a manner that facilitates reasoning about their security posture. A security engineer can define assets of the network that protect, identify threats and vulnerabilities, get security insights on how to improve security and privacy, in a software aided analysis. As described in the D3.1 [11], the aims of the software aided security analysis are to 1) augment the expertise of a security analyst; 2) detect network and system threats in complex distributed environments; 3) remotely and automatically identifying hardware, software and even policy-related vulnerabilities; 4) provision of tailored reports (DiscØvery's cyber-insights), which are suggestions based on the unique characteristics of a network; 5) holistic visualization of the complete threat landscape, including the people, the systems, the networks and the associated policies.
During the course of the Task concerning this deliverable, we have used the outputs of from the INSPIRE-5Gplus project Task that defined the high-level security reference architecture (HLA), the work done on the current status and future trends, and defined security Use Cases, Enablers and Mechanisms for Liability-aware Trustable Smart 5G Security to improve the cyber-insights offered by DiscØvery. The deliverable D2.1 [14] and D2.2 [2] offered valuable information on the type and nature of 5G specific threats. One of the features of DiscØvery is the ability to elicit threats based on the network's configuration. The threat landscape of D2.1 [14] offers a clear mapping between threats and targeted 5G assets that was used to define a dataset of threats for consumption by DiscØvery. The dataset has been structured with the following schema:

```
threatsList = {
        ThreatNumber: {
                concept: “concept of the metamodel”,
                attribute: “attribute of the concept”,
                attributeValue: “value of the attribute”,
                threat: 'description of the threat',
        }
}
```

Based on that schema, DiscØvery can identify which network assets can be targeted by which threats and why. Additionally, using the same information DiscØvery can provide a list of suggestions on how to best mitigate the threat and minimize the attack surface of the network. An example of the threat identification is shown in Figure 18 and Figure 19, showing a 5G network with several edge components, such as MEC platforms and orchestrators, as well as several vehicles that use 5G resources for assisted-driving applications.

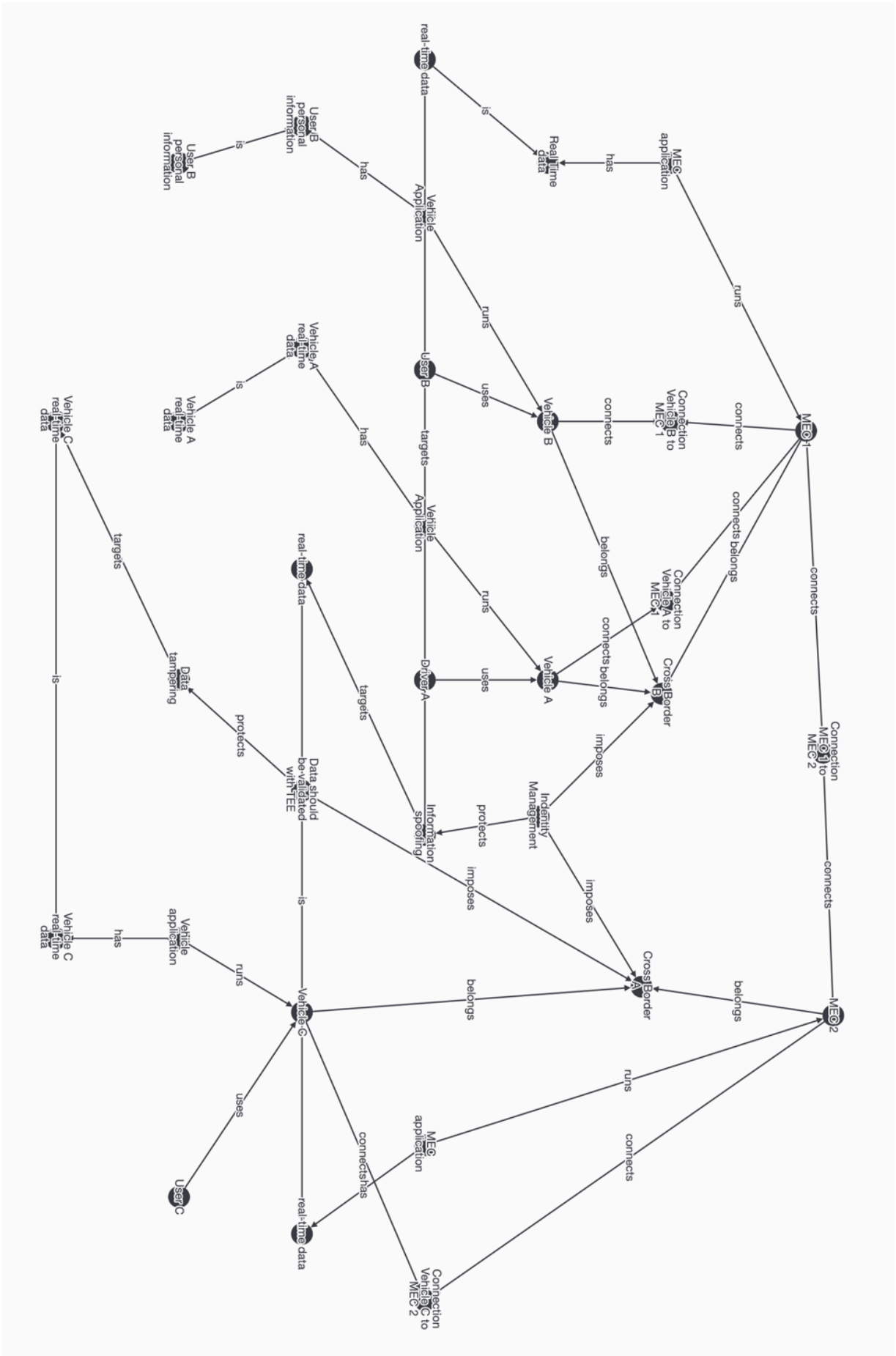

Figure 18: Threat assessment model using DiscØvery

In Figure 19, we show an example of the elicited threats that can target the 5G network components. For example, DiscØvery detected that the network connections between the MECs and the vehicles use insecure network protocols and suggested the use of encrypted alternatives. Furthermore, it detected that most components of the networks do not have policies for firmware and software updates. As a result, such components are prone to be exploited by malicious actors if not updated by an administrator. The enabler suggests to define an update policy for the network assets.

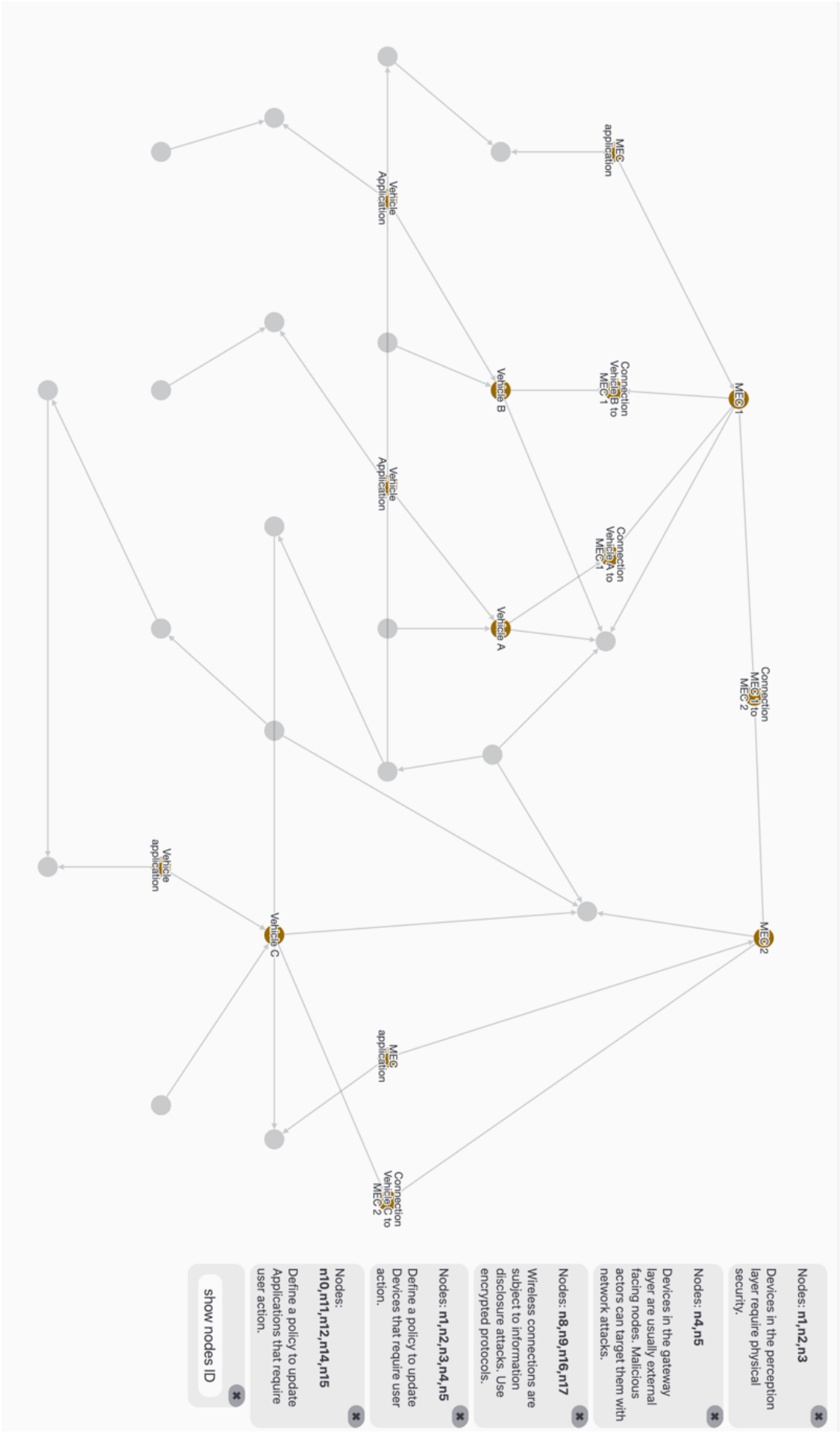

Figure 19: Elicited threats using DiscØvery's dataset

Our next steps for this work are to improve the threat and lists of DiscØvery to better reflect the evolving thread landscape. Our goal is open source the dataset as part of the DiscØvery project so it can be incorporated to other similar tools and improved and adopted by other stakeholders. Additionally, we aim to refine the cyber-insights dataset with more actionable information and learning resources to better assist the security engineer during the threat assessment process. The results of this work will be presented in D3.3 and D3.4 [1]. The APIs of the enabler are described in Section 5.6.

4 Automated detection and enforcement

4.1 Model-driven data management for security monitoring and detection

The application of closed-loop control is the most relevant functionality needed to achieve zero-touch management. Security closed-loop control has as critical requirement a timely and trustworthy flow of data about the entity being managed, focusing on any kind of information that can affect the security of the infrastructure itself or the related services. The heterogeneity, pervasiveness and network topology dependency challenge the availability and utility of these data flows.

The closed-loop operates on input data (i.e., measurements and observations captured in a continuous or punctual collection phase). The producers of this data usually correspond to different sources, the number and heterogeneity of which depend on the use case. In any sufficiently complex system, the number of potential sources of data and security events and their characteristics becomes extremely high, creating a combinatorial explosion that makes control infeasible, especially if no human operation is pursued in the loop.

Some data sources in data management rely on pull-based methods (e.g., Syslog, IPFIX, SNMP), while others depend on push-based methods (e.g., streaming telemetry). The first are based on requests at predefined intervals, and the second are based on a subscription that allows the data consumers (i.e., subscriber) to subscribe to data models and data sources (i.e., publisher). Data availability is another capability where in many cases, timing constraints on both the data and their processing have to be considered. The usability of data for security depends on avoiding too much time delays, and the continuity of the data flow guarantee accurate detections and valid responses.

This high variety of data sources, capabilities and formats make it difficult to manage the information to be used by the network management functions. There is a strong effort in the networks domain to adopt Model-driven data management based on the idea of applying modelling languages to formally describe data sources, defining their semantics, syntax, structure and constraints on the objects they are associated to. Model-driven data management considers data models, transport protocols and data coding languages as independent layers, easing the aggregation of any new protocol and/or coding that follows the same principles of model-driven data management. YANG [17] is the reference in data model language for network management data modelling. YANG allows creating a data model, define the data organization in that model, as well as the constraints. Although model-driven data management started with the use of the NETCONF [18] protocol and XML [19] coding, there are different solutions that make possible their implementation.

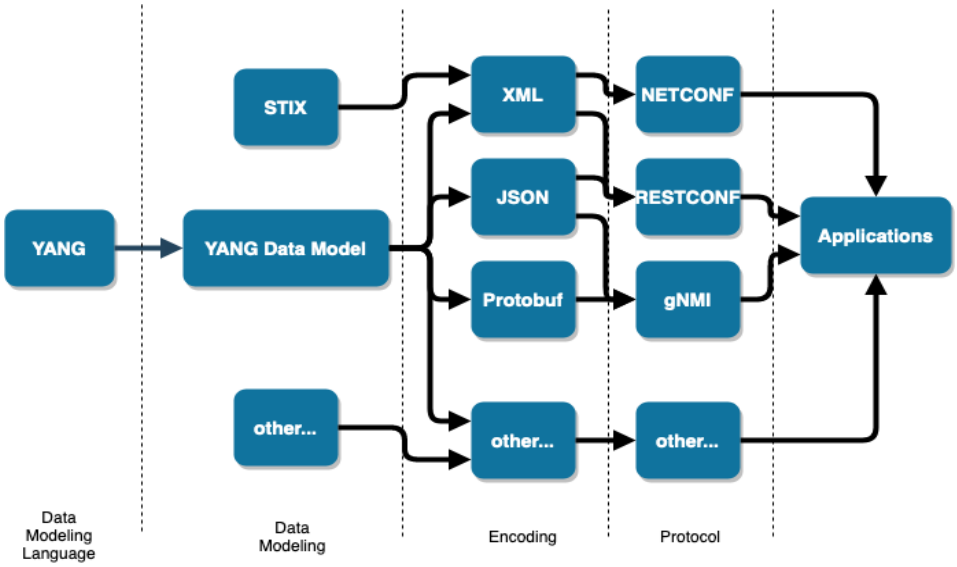

Figure 20: Model-driven data management components

Figure 20 depicts the different components that are available for implementing model-driven data management solutions. In the specific area of security monitoring and data exchange, there are also additional models, encodings and transport protocols [16] that can be adopted. Once the data models are defined that describe the data sources provided by the servers (e.g., network devices, network probes, security agents, network elements), a client (i.e., the security data collector) can select the most appropriate encoding (XML, JSON [21], Protobufs [24], etc.) and the most appropriate transport protocol (NETCONF, RESTCONF [25] or gRPC [26] / gNMI [27], etc.) to collect and store the data.

The ETSI ISG CIM (cross-cutting Context Information Management [28]) approach is based on the context information concept where any relevant information about entities, their properties, and their relationships with other entities are described. Context information is exchanged amongst applications, context producers, and context brokers. CIM considers that the exchange of data and metadata allows better information collection from different origins create derivative information or decisions. The application of the CIM framework has been focused on IoT applications so far, but given its support for binding context (data) sources and consumers, it is possible to apply to network telemetry YANG models, and others security monitoring models. Some potential examples are SNMP MIBs, time series databases (Prometheus), Netflow/IPFIX data flows, or STIX data model.

The INSPIRE-5GPlus security data collector adopts the CIM context and data management principles. It should act as a server for the data consumers attached to it. Different data models are used by the consumers (e.g., the data analytic component) for accessing the data mediated by the component. The use of STIX/TAXII [22] for exchanging threat data is particularly appropriate since it is “the preferred mechanism for the information exchange between CSIRT and LE communities” and its use by CSIRTs/CERTs is recommended by ENISA [23]. STIX/TAXII is to be used in the Cyber Threat Intelligence exchanges in INSPIRE-5Gplus.

4.2 Rule and RT-SSLA conformance

Run-Time-SSLA (RT-SLA) conformance or assessment is based on metrics that can be measured or captured using automated monitoring techniques. They are not to be confused with the SSLAs that allow specifying the security properties of slices and verticals. RT-SSLAs can be derived from these “static” SSLAs but also need to be complemented with more fine-grained rules based on security metrics. These metrics are, for instance, those that allow determining that the security functions are working correctly (e.g., mean time to incident recovery, frequency of security controls, version of security protocols being used) and those that allow detecting security breaches (e.g., using signature

and behaviour-based security rules, change point detection or machine learning algorithms). The RT-SSLA formalism has been initially defined for cloud services during the SPECS [7] project, and extended in the MUSA [29] project where a catalogue [30] was created. INSPIRE-5Gplus adapts these models and formalisms so that they can be used for the security monitoring in Slice/SDN/NFV-based 5G and B5G mobile networks.

The RT-SSLAs are rules based on the captured metrics and allow performing threshold analysis, complex event processing, and deep packet and flow inspection. They also rely on algorithms implementing machine learning techniques and statistical analysis for identifying behaviour anomalies.

Real-time SSLA assessment is done by deriving rules from more abstract models (e.g., specified using HSPL/MSPL proposed in Section 2) that can be monitored by security agents or probes managed by the Security Monitoring Framework. We can predefine security rules using the capabilities provided by security agents, then explicitly extract security rules or threshold values from MSPL. An example of simple RT-SSLA to detect the isolation level between slices is given in the following *Figure 22*. The rule uses the XML syntax to specify the metrics and the thresholds that determine if the observed metrics are considered normal or should provoke a countermeasure.

```
<Metric name="Isolation Accesss from other Slices">
…
<definition>
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)
</definition>
…
<ParameterDefinition name="N" referenceId="">
<def>N: n° flows going to or from given IP ranges of other Slices</def>
</ParameterDefinition>
…
<ParameterDefinition name="T" referenceId="">
<def>T: total number of IP flows of the application</def>
</ParameterDefinition>
```

Figure 21: An example of an RT-SSLA rule to detect the isolation level between slices

In the above example, one can define what is considered a normal threshold that can be zero (complete isolation) or a certain percentage (limiting the network flows going to or from the slice) . The IP ranges allocated to a slice are retrieved from the configuration repository. The following *Figure 22* shows a complete XML specification.

```
<?xml version="1.0" encoding="UTF-8"?>
<wsag:AgreementOffer xmlns:wsag="ws-agreement" xmlns:specs="SLAtemplate" xmlns:xsi="XMLSchema-instance">

 <specs:Metric name="Isolation Accesss from other Slices" referenceId="ISOLATION_ACCESS">
  <specs:MetricDefinition>
   <specs:unit name="%">
    <specs:intervalUnit>
     <specs:intervalItemsType>integer</specs:intervalItemsType>
     <specs:intervalItemStart>0</specs:intervalItemStart>
     <specs:intervalItemStop>100</specs:intervalItemStop>
     <specs:intervalItemStep>1</specs:intervalItemStep>
    </specs:intervalUnit>
```

```
    </specs:unit>
    <specs:scale>
     <specs:Quantitative>Ratio</specs:Quantitative>
    </specs:scale>
    <specs:expression>(N/T) *100</specs:expression>
    <specs:definition>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).
    </specs:definition>
    <specs:note />
   </specs:MetricDefinition>
   <specs:AbstractMetricRuleDefinition>
    <specs:RuleDefinition name="" referenceId="">
     <specs:definition />
     <specs:note />
    </specs:RuleDefinition>
   </specs:AbstractMetricRuleDefinition>
   <specs:AbstractMetricParameterDefinition>
    <specs:ParameterDefinition name="N" referenceId="">
     <specs:definition>N: n°  flows going to or from given IP ranges of other slices
     </specs:definition>
     <specs:parameterType />
     <specs:note />
    </specs:ParameterDefinition>
    <specs:ParameterDefinition name="T" referenceId="">
     <specs:definition>T: total number of IP flows of the application
     </specs:definition>
     <specs:parameterType>integer</specs:parameterType>
     <specs:note />
    </specs:ParameterDefinition>
   </specs:AbstractMetricParameterDefinition>
   <specs:MetricRules>
    <specs:MetricRule>
     <specs:ruleDefinitionId />
     <specs:value />
     <specs:note />
    </specs:MetricRule>
   </specs:MetricRules>
   <specs:MetricParameters>
    <specs:MetricParameter>
     <specs:parameterDefinitionId>Result</specs:parameterDefinitionId>
     <specs:value>0</specs:value>
     <specs:note>It is the percent parameter.</specs:note>
    </specs:MetricParameter>
   </specs:MetricParameters>
   <specs:note />
  </specs:Metric>
</wsag:AgreementOffer>
```

Figure 22: An example of a complete XML specification

The RT-SSLA rules will be used and deployed by the Security Monitoring Framework (SMF) for their assessment during operation. Thus, the SMF needs to interact with other enablers as defined in Section

5.8: with the Policy Management to determine what rules need to be deployed, and with the Security Orchestrator to notify their violation and to enforce them.

For the enforcement, one needs to indicate in the policy or RT-SSLA what needs to be done. This then needs to be translated to a format that can be understood and implemented by the VIM Orchestrator or the Security Orchestrator. Formalisms such as TOSCA are enforced through a VIM orchestration (e.g., OpenStack HEAT). The UMU Security Orchestrator, on the other hand, only receives MSPLs. The UMU Security Orchestrator could request an MSPL to TOSCA translation and then request the TOSCA enforcement using a VIM orchestrator as enforcement point.

The following *Figure 23* gives an example in TOSCA/YAML for deploying a WordPress content management system monitored by MMT.

```
tosca_definitions_version: tosca_simple_yaml_1_0

description: >
 TOSCA deployment template for deploying WordPress being monitored by Montimage Monitoring Tool

imports:
 - toscaparseretso/custom_types/k8s-container-type.yaml

topology_template:
 inputs:
  k8s_namespace:
   type: string
   default: "sendate-demo"

 node_templates:

  ###################### MMT ########################
  mmt-database-volume:
   type: tosca.nodes.ETSO.K8s.V1Volume

  mmt-database:
   type: tosca.nodes.ETSO.K8s.V1Container
   properties:
    image: mongo:xenial
    args:
    - "--dbpath=/mongo-database"
    ports:
    - container_port: 27017
     name: mmt-database
    readiness_probe:
     tcp_socket:
      port: 27017
    volume_mounts:
    - name: mmt-database-volume
     mount_path: /mongo-database

  mmt-operator:
   type: tosca.nodes.ETSO.K8s.V1Container
   properties:
    image: montimage/mmt:operator
```

```
    args:
    - "-Xdatabase_server.host=mmt-database"
      "-Xinput_mode=socket"
    ports:
    - container_port: 8080
      name: mmt-operator
    readiness_probe:
      tcp_socket:
        port: 8080

mmt-probe:
  type: tosca.datatypes.MMT.K8s.V1Montior
  properties:
    image: montimage/mmt:probe
    visualisation: mmt-operator
    args:
      -Xinput.source=eth0
      -Xfile-output.enable=false
      -Xsocket-output.enable=true
      -Xsocket-output.hostname=mmt-operator
      -Xsession-report.output-channel=socket
      -Xsecurity.enable=true
      -Xsecurity.output-channel=socket
#####################################################
database:
  type: tosca.nodes.ETSO.K8s.V1Container
  properties:
    name: mysql
    image: mysql:5.6
    env:
    - name: MYSQL_ROOT_PASSWORD
      value: password
    ports:
    - container_port: 3306
      name: mysql
    readiness_probe:
      tcp_socket:
        port: 3306
    volume_mounts:
    - name: mysql-volume
      mount_path: /var/lib/mysql
    monitor:
    - id : 1
      name: mmt-probe

mysql-volume:
  type: tosca.nodes.ETSO.K8s.V1Volume
  properties:
    empty_dir: {}

wordpress-mysql:
  type: tosca.nodes.ETSO.K8s.V1Service
  properties:
    metadata:
```

```
      name: "wordpress-mysql"
      labels:
        internal: "wordpress"
    spec:
      ports:
        - port: 3306
          target_port: 3306
      selector:
        database: mysql

 frontend:
  type: tosca.nodes.ETSO.K8s.V1Container
  properties:
    name: wordpress
    image: wordpress
    env:
    - name: WORDPRESS_DB_HOST
      value: wordpress-mysql
    - name: WORDPRESS_DB_PASSWORD
      value: password
    ports:
    - container_port: 80
      name: wordpress
    readiness_probe:
      tcp_socket:
        port: 80
    monitor:
    - id : 2
      name: mmt-probe

 wordpress:
  type: tosca.nodes.ETSO.K8s.V1Service
  properties:
    metadata:
      name: "wordpress"
      labels:
        app: "wordpress"
    spec:
      type: "NodePort"
      ports:
        - port: 80
          target_port: 80
          node_port: 31780
      selector:
        frontend: wordpress

policies:
 - deployment_order:
    type: tosca.policies.ETSO.DeploymentOrdering
    properties:
      deployment_order:
        - [database]
        - [frontend]
```

Figure 23: TOSCA/YAML for deploying a WordPress content management system monitored by MMT

4.3 Policy enforcement function

During the Security Orchestration procedures, the policy enforcement process which is depicted in Figure 24, is responsible for enforcing MSPL Orchestration Policies throughout the infrastructure. As the system is constantly monitored and analysed, the policy enforcement process may be triggered proactively as the result of security measures provided by SMD administrators (from the Policy Framework) (1.a in the *Figure*) as well as inputs from the E2E Security Management Domain, or reactively as dynamic countermeasures from the Decision Engine (1.b) within the Security Management Domain as well as from E2E Security Management domain as part of an E2E countermeasure.

After the orchestration process described in Section 3.3.3, and once the SMD Security Orchestrator has decided the most suitable security assets and their location by gathering information of the infrastructure from the data services for each MSPL policy (2,3), it requests the policy translation procedure to the Policy Framework (4) that performs the translation process as explained in Section 2.2. When the SMD Security Orchestrator receives the final configurations, it starts enforcing the configurations through the different controllers, managers and security assets that will act as security enforcement points of the infrastructure. To this aim, a driver-based approach allows introducing new connection logics as new enforcement technologies are available. Thus, these enforcement points use specific driver implementations to enforce the requested configurations. Depending on the nature of the enforcement point, the enforcement on the final configurations could be delegated to specific managers or controllers (5.a) like NFV MANO management for new/dynamic asset deployment, or they could be directly enforced into the required security asset (5.b) by reconfiguring certain security asset (e.g., by modifying a monitoring agent behaviour or by updating channel protection cryptosuite). It is important to highlight that the enforcement process can be also triggered dynamically to enforce security policies whose dependencies have been solved. The policy enforcement is triggered at the **meservice** located in Security Orchestrator's API.

Typical examples of policy enforcement points are the security enablers illustrated on the Figure 24, such as the Virtual Channel Protection enabler presented in Section 5.12, or the I2NSF IPSEC enabler presented in Section 5.10

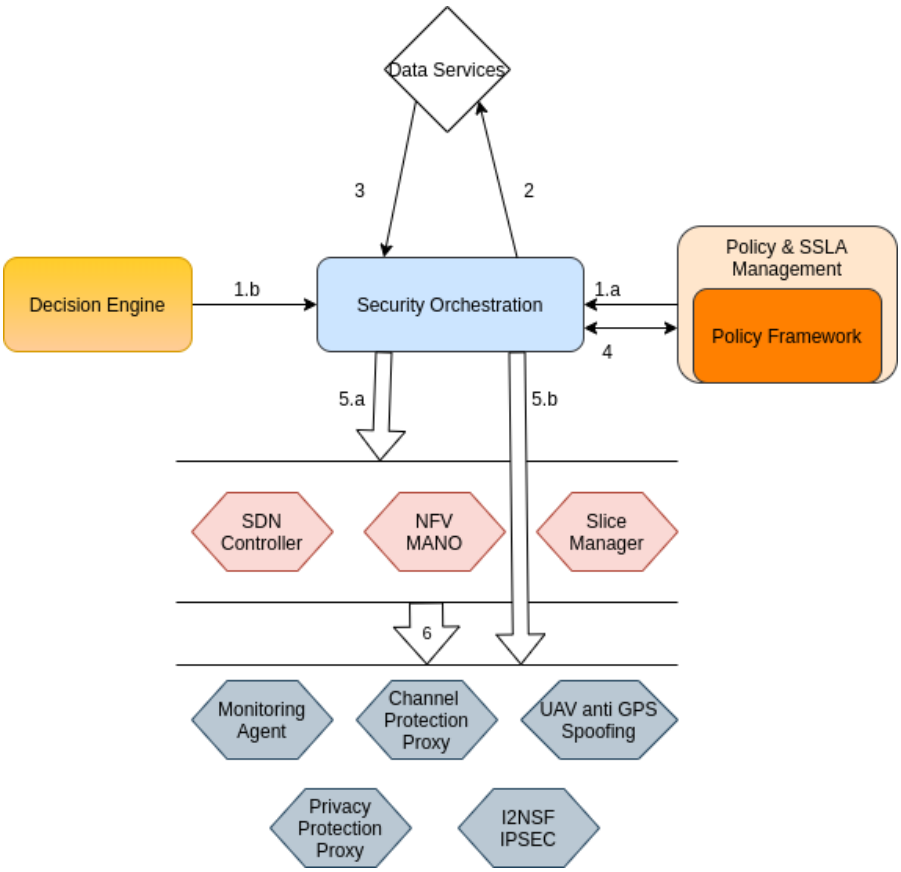

Figure 24: Policy enforcement concept

4.4 Security orchestration with MTD policy enforcement

An additional security layer at the Service Management Domain is the Moving Target Defence (MTD) policy. MTD operations concerns the strategic placement of the different verticals' resources *proactively*, increasing the complexity for malicious users to perform reconnaissance attacks and use the gathered intelligence in time for effective and tailored attacks, and *reactively*, mitigating current attacks detected by the Security Analytics Engine. The enforcer gets directives from the Decision Engine, enabled with an MTD policy that can indicate the MTD actions to perform based on the network's state.

MTD operations are enforced on different abstraction layers of the infrastructure (see Figure 25 above):

1. At the virtualisation layer, to secure NFV assets such as network slices, network services and virtual network functions by operating with the service and vertical orchestrator and the slice manager. These operations are re-initiation of a vertical's component, migration of a service to a different Virtual Infrastructure Manager (VIM), and dynamic deployment of additional security network functions;
2. At the transport network layer, to control the traffic using SDN controllers, and to change the network's topology by shuffling the IP address space and by reconfiguring network interfaces. These operations can be performed both at the internal and external views of the network: in the former MOTDEC prevent intruders to further explore the internal network, while the latter may be used for reactive placements against suspicious users or traffics;

MTD strategies can also dynamically deploy deceptive networks and honeypots to deceive the intruder. This set of MTD actions maximizes the unpredictability of network's movement for both external and internal malicious users.

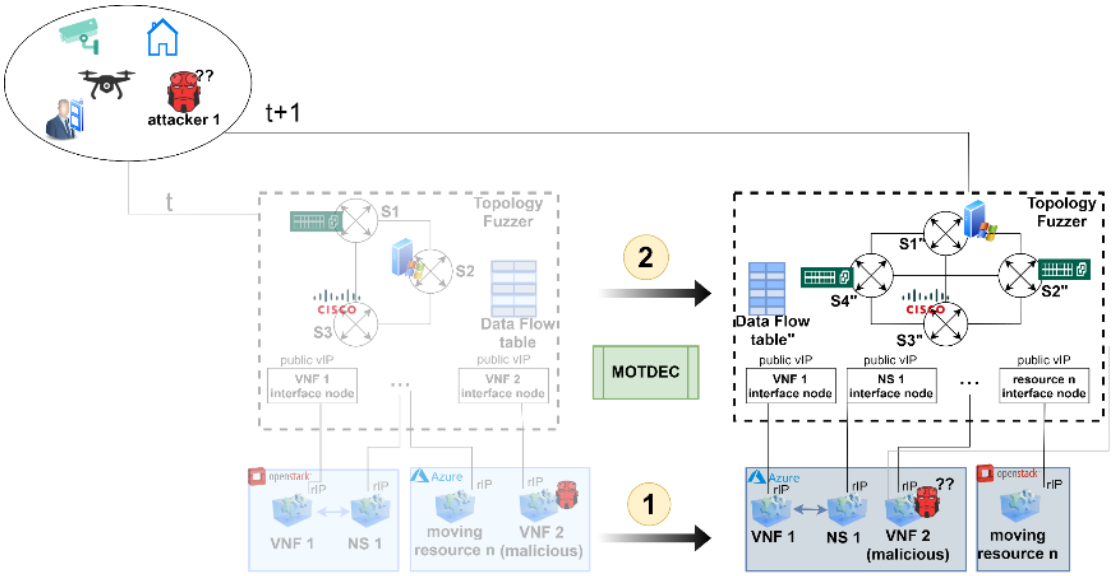

Figure 25: MTD actions at different layers of the infrastructure

In the context of the INSPIRE-5Gplus project, a new security enabler, namely the MTD controller (MOTDEC), is being developed in order to enforce certain MTD operations on NFV assets such as VNFs, VLDs, NSs and network services with a main focus on the protection of the latter, which includes the coordinated usage of the former ones. MOTDEC is developed as a security orchestration component within the INSPIRE-5Gplus HLA, defined in the deliverable D2.2 [2].

The enabler's current implementation performs MTD actions at the virtualisation layer. Notably, it can re-instantiate components of a network service to mitigate intrusion attacks. It can also move the components from one physical server or MEC platform to a different one, solving host-related issues that can affect the QoS of the protected slice network.

MTD actions vary in resource consumption and have various overheads on the performance of the protected services. Thus, the MTD policy needs to be optimized for an optimum trade-off between security requirements (i.e., SSLAs) and functional requirements (i.e., SLA, QoS, etc). For this reason, MOTDEC is using another INSPIRE-5Gplus enabler, the optimizer of security functions (OptSFC), an offline tool used in testbeds to pre-train ML models that maximize the efficiency of a set of security functions (in this case the MTD actions). MOTDEC is also interfaced with other enablers such as the network slice manager and monitoring frameworks to keep track of the running network slices, sub-components, and their health status. The REST API interface defined to this scope is described in Section 5.11.

4.5 Katana Slice Manager

Network Slicing is a procedure that creates multiple virtual networks on top of a common physical infrastructure. Based on NFV and SDN, operators can provide portions of their networks to different vertical industries, tailored to their specific requirements. Katana Slice Manager is central software component that is responsible for creating, modifying, monitoring and deleting network slices. It interacts with higher layer entities through its SBI, such as Policy and Decision Engines, mobile operators and interacts with the underlying infrastructure through its NBI, that communicates with the the components of the Management and Orchestration Layer (MANO), namely the NFV Orchestrator (NFVO), the Virtual Infrastructure Manager (VIM), the Element Management System (EMS) and the WAN Infrastructure Management (WIM), in order to manage the functions in the network and perform CRUD operations on End-to-End network slices.

The REST API of the Katana Slice Manager can be found in the following link in OPENAPI format.

- https://github.com/5genesis/katana-slice_manager/blob/master/katana-swagger/swagger.json

The complete documentation that also includes the APIs definition can be found in the project's Github page:

- https://github.com/5genesis/katana-slice_manager

In the context of INSPIRE-5Gplus, Katana is being extended to support MTD policy actions for the purposes of Demonstrator 3. The points below highlight these extensions and the following subsections describe them in detail.

- Generalization of NEST/GST 3GPP template in order to introduce additional technical specifications
- Slice Mapping and Scheduling components of KATANA extension to support Moving Target Defence
- Monitoring mechanisms for network slices that are being shared among different tenants/service
- Support of Open5GS core network implementation for Demonstrator 3

4.5.1 Generalization of NEST/GST 3GPP template

The Network Slice Template that describes the parameters of network slices in the context of 5GENESIS follows the specifications of *NG.116 – Network Slice Template v2.0* [20], defined by the GSMA. This document aims to assist network slice providers in mapping network slices' use cases into generic attributes. For this purpose, GSMA defines the Generic Network Slice Template (GST), a set of attributes that characterize a network slice type. GST is generic and is not tied to any specific network deployment.

The NEtwork Slice Template (NEST) is a GST filled with values. The defined attributes and their values are assigned to fulfil a given set of requirements derived from a network slice use case. This process is depicted in the diagram of Figure 26, where a Network Slice Customer (NSC) provides the requirements for a specific use case to the Network Slice Provider (NSP). The requirements are mapped into the attributes of the GST with appropriate values generating a NEST.

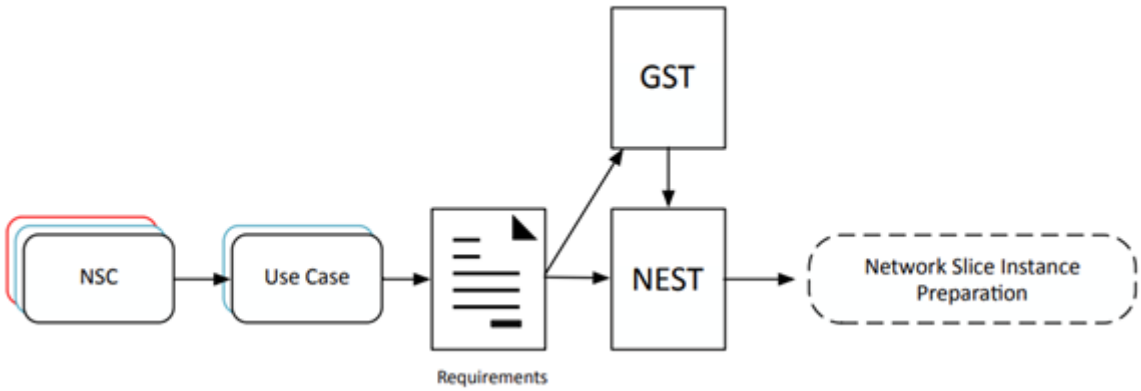

Figure 26: Network slice creation request

The NEST is an input to the network slice (instance) preparation performed by the Slice Manager. The user sends a NEST to Katana Slice Manager along with a slice creation request. The Slice Mapping

process then parses the NEST, which, combined with the supported network functions supported by the underlying infrastructure, defines the NSSIs that shall be deployed as part of the slice.

The NEST can be divided into three main sections:

- **Base Slice Descriptor:** This section includes the main parameters that the Slice Manager utilizes for the Slice Mapping process, such as location, bandwidth requirements, isolation level, QoS, etc. This process selects the appropriate NSSIs that will be used for constructing the new slice.
- **Vertical Services Descriptor:** This section describes network services that are not part of the slice's core service but are defined by the Network Slice Customer, such as firewalls, caches, proxies, etc. These services can be either physical or virtual and must be supported by the underlying platform infrastructure.
- **Test Descriptor:** This section specifies a series of tests that the Slice Manager must run against the newly deployed slice. These tests can validate various aspects of the slice, such as the service's performance and reliability.

While the Base Slice Descriptor is a mandatory section for a NEST used to deploy a new slice, the other two sections are optional and might be omitted. Slice Manager allows the administrator to on-board descriptors for each section before the slice creation phase. Moreover, it creates an endpoint that returns a list with all the on-boarded descriptors.

4.5.2 Slice Mapping and Scheduling components of KATANA extension to support MTD

The Slice Mapping process runs during the slice creation phase. It is responsible for optimally selecting the infrastructure resources to be used for a new slice, based on the slice requirements, as described in the NEST and the available resources of the infrastructure layer. When Slice Manager receives a request for the creation of a new slice, based on the definition of the available NSSIs (whether they can be used by multiple slices at the same time or not) and the requirements of the new slice, this module decides if the new slice can also use an NSSI that is already part of another running slice.

In order to support the Moving Target Defence functionalities, the Slice Mapping process will introduce some additional operations that will allow specific Day-2 configuration actions on a running Slice. These operations will include the re-instantiation of a Virtual Network Function and re-establishment of the Transport Network paths, complying with some specific constraints provided by the MTD component. These Day-2 operations will enhance the security of a running slice and mitigate the risk of permitting unauthorized actions by malicious parties.

4.5.3 Monitoring mechanisms for network slices that are shared by different tenants/service

The Slice Monitoring module in Release B enables monitoring, visualization, and alerting services responsible for tracking the status of components and services part of the instantiated slices and the Slice Manager itself. It comprises a Prometheus server and a Grafana web application running in Docker containers as part of the overall Slice Manager software stack. This process provides insights on the performance while ensures continuous uptime and good health of the services realized by every network slice. Platform administrators can utilize the slice monitoring feature to quickly detect networking failures and determine in real-time whether the platform is running optimally. Release B of the 5GENESIS Slice Manager capitalizes on Prometheus and Grafana to create a toolkit that offers monitoring, visualization, and alerting capabilities. The tools are packaged in Docker containers and delivered as part of the Slice Manager microservices architecture.

Slice Manager exploits Prometheus as a time-series database. It collects, organizes, and stores metrics from targets by scraping HTTP endpoints. These targets are either physical or virtual components and services of the underlying infrastructure that have been instantiated and configured to be part of a

Figure 27: Grafana dashboard for a deployed slice by Katana

deployed slice. Prometheus offers a file-based service discovery mechanism, which allows adding new targets dynamically in a JSON file, along with metadata about those data. Every component of the MANO layer that the platform administrator adds to the Slice Manager is configured to be a new Prometheus target if the component supports it. This feature enables Slice Manager to collect monitoring information from various domains that are part of each slice.

Alerting rules is another valuable feature of Prometheus that the Slice Manager utilizes. This feature allows the platform administrator to define alert conditions based on Prometheus expression language. Whenever the alerting criteria are met for one or more elements at a given point in time, Prometheus sends notifications about the firing alerts to external services.

In addition to the process of scrapping the data, Slice Manager utilizes Grafana to visualize the collected metrics stored in the Prometheus database. Grafana allows the creation of multiple dynamic and reusable dashboards that include custom visualizations based on predefined templates. Slice Manager exploits this capability to create a dedicated dashboard for every deployed slice, concentrating and visualizing the metrics related to each one. This approach allows the Slice Manager administrator to efficiently keep track of the services and components that are instantiated and configured as part of a slice. When a network slice is terminated, Slice Manager dynamically deletes the respective Grafana dashboard. Figure 27 provides a snapshot of a Grafana dashboard that the Slice Manager created for a deployed slice. In this example, Grafana visualizes metrics collected from i) the NFVO related to the health and status of the instantiated VNFs, ii) the WIM related to the network traffic in the slice, and iii) the VIM related to the performance of the created VMs.

Figure 28: Grafana home dashboard

Furthermore, Grafana creates a Dashboard that provides an overview of the deployed slices and their status. This Dashboard is based on a static configuration and is used as the Grafana home page dashboard. Figure 28 depicts a snapshot of the home dashboard.

4.6 Optimisation based on chaining and microservices

4.6.1 Chaining of virtualised security functions, micro-services, and VNFs

The 5G network aims to converge mobile and fixed networks for supporting E2E applications and services in a more secure and resilient manner. To improve the flexibility and adaptability of the Virtualised Networking Functions (VNFs), one technique is to introduce the composition of network functions based on micro-services at the control and data-planes. NFs enable a wide variety of applications to be implemented as in-network software functions that range from lightweight software firewalls, high performance proxy to complex Deep Packet Inspection (DPI) applications. This is referred to as Service Function Chaining (SFC) to build more flexible and complex security services that involve the monitoring and enforcement. For example, network slicing in 5G could be achieved thanks to SFC when data processing will become a sequence of services managed by different stakeholders. Furthermore, deploying a monitoring service that monitors the network traffic is crucial for detecting anomalies in NFV-based architectures, as the monitoring service allows to provide detailed metadata and security reports concerning the network traffic in real-time. One of the challenges is to find a way to efficiently transmit packets through a processing chain. In this perspective, the objective is to identify the security NFs that can be composed from micro-services, define multi-layer orchestration, and optimize micro-services' interactions.

One potential solution that has been investigated is OpenNetVM [31]. This tool allows managing and orchestrating micro-services. It provides a virtualisation-based high-speed packet delivery platform that obtains low resource overhead and low latency. The key idea is to eliminate the overhead of Operation System kernel packet processing, allowing zero-copy packet delivery between NFs, requiring no interrupts using DPDK [32] poll-mode driver, and allowing dynamically managing and reconfiguring the NFs using its NF Manager. It relies on a shared memory across NFs and NF Manager that eliminates overhead introduced when copying network packets or traversing network interfaces. It also allows defining security domains, i.e., security boundaries between trusted and untrusted NFs.

The OpenNetVM architecture is depicted in Figure 29 where we have a shared memory that can be accessed by the different container-based NFs, and the NF Manager that allows tracking the active NFs and directing the network flows from one to the other.

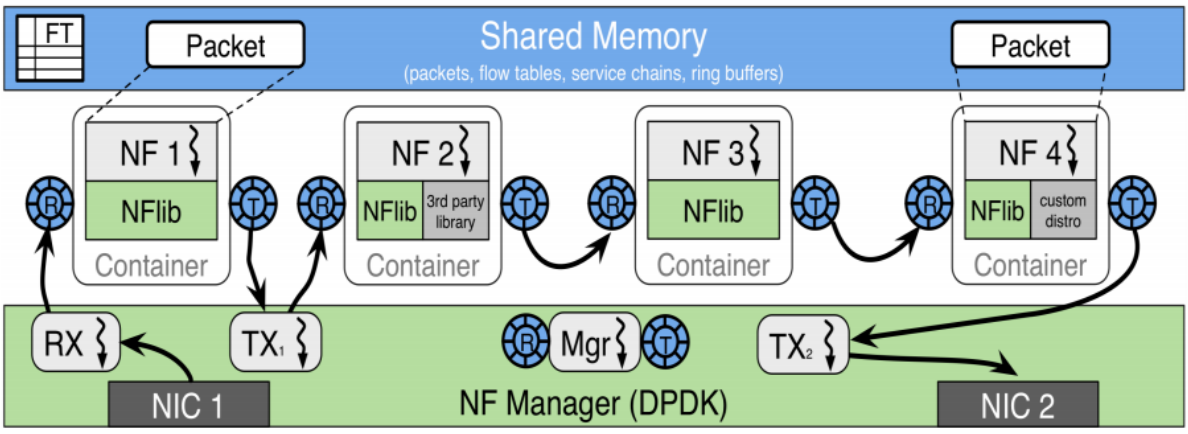

Figure 29: OpenNetVM architecture

The NFs are run inside Docker containers and use the NFlib API that allows each NF to interface with the NF Manager and other NFs, for instance:

- onvm_nflib_init function initializes all the data structures and memory regions that the NF needs to run and communicate with the manager.

- onvm_nflib_run is the communication protocol between the NF and the manager, where the NF provides a pointer to a packet handler function to the manager. The manager uses this pointer to pass packets to other NFs and in this way route the traffic. This function continuously loops and give the addresses of the packets one-by-one to the destination NF in the chain.

The shared memory handles huge pages for making the packets processing efficient. The RX threads directly receive the packets from NICs (Network Interface Cards) using DPDK, and the TX threads make the packet pointers available between NFs. Each NF is identified by its ServiceID and its InstanceID, several NFs can share the same ServiceID but the InstanceIDs are unique. For a given ServiceID, the manager will decide which instantiated NF will process the incoming packet, improving flexibility and fault tolerance. In case of a sudden freeze of an NF, the manager can simply route the packets to another one using the same ServiceID.

A simple example of a service chain is given in Figure 30 that shows one NF acting as a firewall to filter the packets that do not need to be processed by the security solution, and the disaggregation of the packets to optimise the processing by separating it into specialised processing and/or parallelising it. To increase the throughput speed and avoid the packet loss problem, we can develop a scaled version of complex services, such as the monitoring NF Probe, running on multiple cores on the OpenNetVM framework. Technically, the scaling NF will spawn multiple children NFs. and each child NF which has a unique InstanceID runs on a specific core. OpenNetVM uses the Receive Side Scaling (RSS) hash value to provide flow consistency and support load balancing among children NFs with the same ServiceID. All spawned NFs manage their own RX/TX ring buffers to receive and send packets to the destination NF in the service chain.

Furthermore, OpenNetVM facilitates the deployment and placement of NFs to ensure E2E flow performance. It is flexible to change the placement or redeploy new instances of services controlled by the Security Orchestrator in Section 3.3 according to the predefined SSLAs in Section 3.2 in the event of an accident.

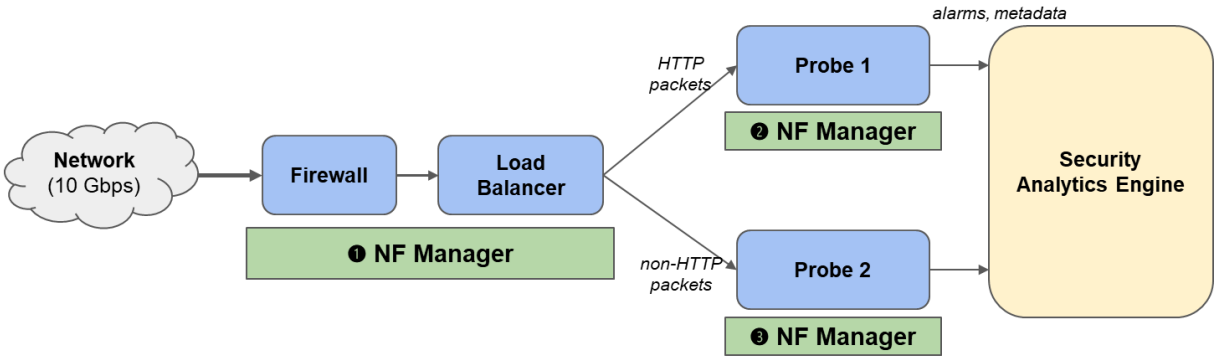

Figure 30: Example of a service chain

4.6.2 Security by Orchestration for optimizing the placement of Vertical applications

Multi-access Edge Computing (MEC) enables the 5G network operator to offer cloud computing capabilities and an environment dedicated to the delivery of IT services at the network edge. Using edge infrastructure distributed over the network, external applications can be hosted meeting specific requirements of vertical industries (e.g., low latency, high bandwidth, data sovereignty and privacy protection). The most widely used MEC reference architecture (MEC RA) is defined by ETSI [33]. It also describes process of orchestration and management of MEC applications.

As indicated in 5G-PPP white paper on Edge computing [34], the MEC system must be secured according to the best practices of server security, virtualisation infrastructure security and address

additional MEC specific threats. In addition to using various protection technologies like system hardening techniques, system-level authentication and access control, physical controls, communications security, software integrity protection, TEE, HSM, etc.; isolating sensitive workloads from non-sensitive ones is recommended in multi-tenant virtualised environments to which MEC belongs. The optimization of placement of Vertical applications with isolation constraint can be the solution to meet this recommendation within limited resources of edge infrastructure.

MEC infrastructure model

The edge infrastructure for 5G networks and beyond is often structured as 3-layer hierarchy (*Figure 31*) of edge datacentres (DC) reflecting available physical locations and underlying transport network topology consisting of:

- Regional Datacentres – providing new geographically distributed hosting capabilities for services based on NFV and function chaining;
- Local Datacentres – often reusing the infrastructure developed for fixed network (Central Office) with placement usually correlated with density of population;
- Cell Site Datacentres for extreme edge use cases – located close to RAN infrastructure.

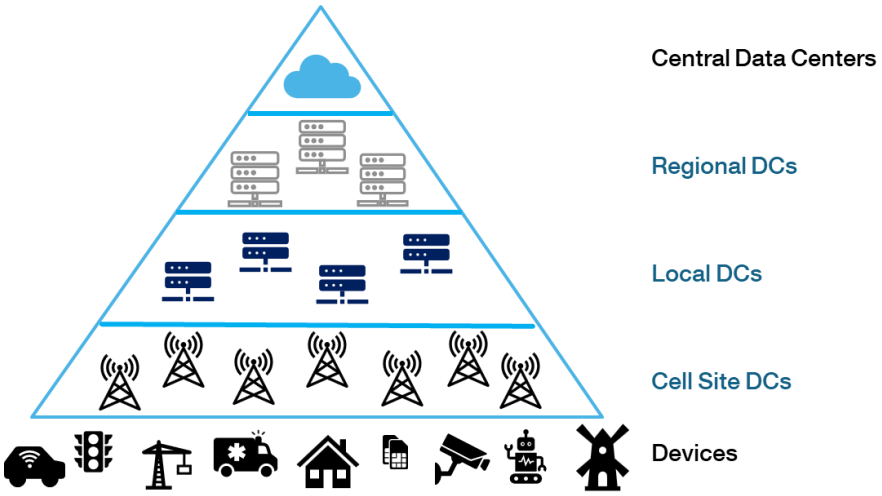

Figure 31: Model of 5G and beyond computing infrastructure

The quantity of possible Cell Site edge locations is much higher than Local or Regional datacentres. Going up in the presented datacentre hierarchy increases latency, but improves datacentre physical protection and available processing power. Going down the hierarchy brings higher costs (due to limited capacity of Cell Site edge datacentres) with better QoS parameters. In further considerations fully-fledged edge computing continuum with 3-level hierarchy of datacentres is assumed.

Security by Orchestration

According to ETSI MEC RA, the placement of MEC applications in Edge Datacentres (MEC hosts) is performed by MEC Orchestrator taking into account required performance and latency. In the proposed approach (Security by Orchestration) security constraints are taken into consideration in the orchestration process.

Initially for the isolation constraint the following hypothesis is used: *MEC application instances can be placed on the same physical node if and only if each of them has security level equal or higher than the maximum of requested isolation levels by each of them*.
It assumes that security level (*sec_lvl)* is assessed for each MEC application and application owner may request isolation_level (*iso_lvl*) not higher than *sec_lvl*:

$$iso\_lvl \leq sec\_lvl$$

Security level of MEC application image is determined in security assessment process using criteria such as image vulnerability scanning results; security assessment of base OS image used for MEC application image; security of development process (programming language, libraries, open-source components, etc.); security assessment of application testing. On the other hand, isolation requirement for each application instance is defined by application owner using application security level and its criticality for the whole service. Resulting affinity rule prevents placement of sensitive applications on the same resources with non-sensitive and potentially more vulnerable applications that can be taken over in order to perform attack on isolation.

In order to include these constraints in MEC application placement process, the placement optimization algorithm needs to be defined. Initially it is proposed the algorithm that calculates the optimal placement of all requested instances of all MEC applications over available edge infrastructure resources. Therefore, its input data should include details about edge infrastructure and MEC application instances to be placed in it. Edge infrastructure data model consists of detailed topology of edge datacentres (*Figure 32*) with information about available servers and their resources, i.e., vCPU, RAM, storage, GPU, HSM, power consumption, cost of vCPU.

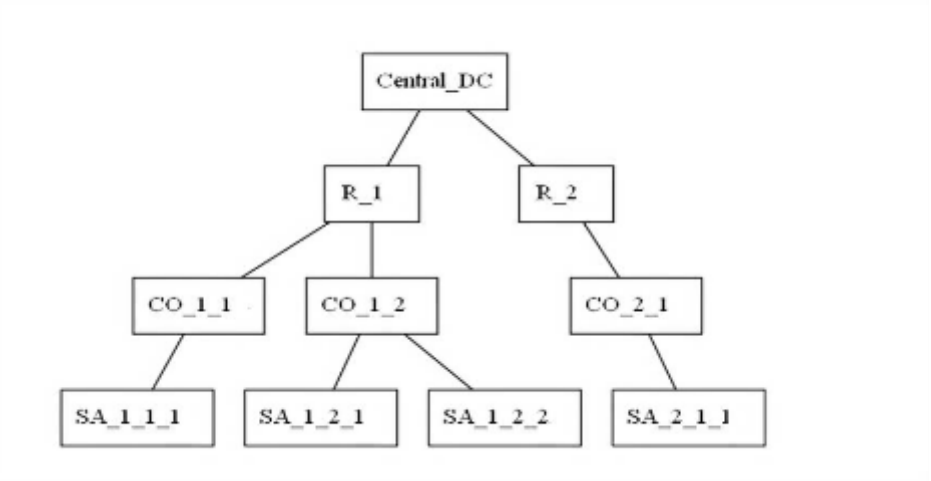

Figure 32: Simple example of MEC infrastructure topology

On the other hand, MEC application instance data model includes instance parameters, e.g., vCPU, RAM, storage, latency, preferred location, additional constraints (availability of GPU, HSM) and mentioned above security constraints: security level and isolation level. By contrast, the output data of placement algorithm gives optimised placement of application instances preserving all presented constraints (including isolation requirement).
The problem can be solved with different optimization techniques (like linear optimization) or using heuristics to define sub-optimal placement of applications. The optimisation criteria should cover cost of placement on specific hardware, overall latency imposed by location of application instances and possibly other needs as energy effectiveness.

Integration into ETSI MEC RA and INSPIRE-5Gplus architecture

The placement algorithm can be exposed via API (described in Section 5.9) forming Security by Orchestration for MEC enabler to be used by MEC Orchestrator. In order to get the optimal placement MEC Orchestrator needs to aggregate input data for the enabler. The data models for *MEC Application Package*, *MEC Application Descriptor*, *MEC Application Instance* defined in ETSI MEC specifications [33][35] can be extended with additional needed parameters.

Security by Orchestration for MEC enabler interacts directly with Service Orchestrator (in case of 5G MEC it is MEC Orchestrator). Therefore, its functionality is not related to security orchestration. In Security Management Edge Domain, the isolation requirements implementation and monitoring can be controlled within Policy & SSLA Management module. As depicted in Figure 33, MEC customers (Verticals) provide needed information to instantiate applications in MEC, security related information is then processed by Security Management framework and relevant data are transferred to Service Orchestrator where optimal placement of applications with security constraints is performed.

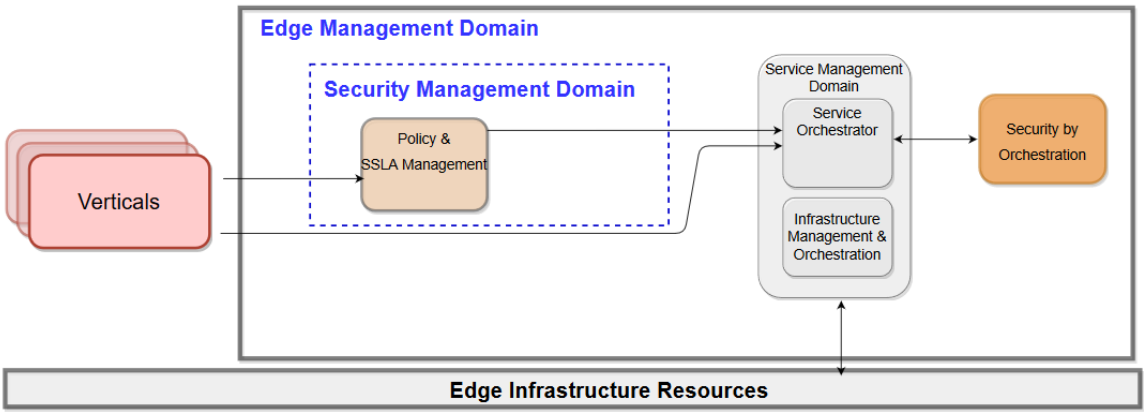

Figure 33: Security by Orchestration and Inspire-5Gplus architecture

Security by Orchestration for MEC enabler provides the optimization of placement of Vertical applications with isolation constraint for MEC applications deployed as virtual machines. Initial placement algorithm calculates locations for static set of applications and instances. Further evolution directions include the dynamic changes of instances to be placed, other virtualisation payloads (like containers) and additional constraints to be considered in the optimization.

5 Specification of the enablers' APIs

In the following we provide the links to the API specified for each enabler that drive the 5G security, making it adaptive and more automated. We give here the OpenAPI or Kafka messages specifications with a brief description.

5.1 Security Orchestrator (TSG, UMU)

The Security Orchestrator's REST API is specified in OpenAPI format on the Swagger website:

- https://app.swaggerhub.com/apis/INSPIRE-5Gplus/Security-Orchestrator/1.0.0

Sections 3.3.3 and 4.3 describe the security orchestration and enforcement features that will be provided by the Security Orchestrator enabler. In this regard, E2E SMD Security Orchestrator and SMD Security Orchestrator expose REST APIs to orchestrate and enforce HSPL-OP/MSPL-OP according to the description provided by those Sections. This is, they implement REST APIs to orchestrate and enforce security orchestration policies across multiple domains (E2E) and inter-domain.

The OpenAPI definition that allows requesting SMD E2E policy orchestrations and enforcements can be found at:

https://github.com/INSPIRE-5Gplus/i5p-hla-api/blob/main/WP3_SecurityOrchestrator_UMU/e2e_security_orchestrator.jsonBriefly summarised, the SMD E2E Policy Orchestrator APIs are the following:

- **e2eseservice**: point at which the SSLA is received, the SSLA orchestration manager is instantiated and refinement to MSPL-OP is performed for the subsequent creation of the e2e orchestration plan.
- **e2eheservice**: point at which the HSPL-OP is received, the HSPL-OP orchestration manager is instantiated and the refinement to MPSL-OP is performed for the subsequent creation of e2e orchestration plan.
- **e2emeservice**: point at which the MSPL-OP is received (from SMD Decision Engine), the MSPL-OP orchestration manager is instantiated and the orchestration process finalizes with the e2e orchestration plan (Calculated SMD involved domains, conflicts & dependency detected).

The OpenAPI definition that allows requesting SMD policy orchestrations and enforcements can be found at:

- https://github.com/INSPIRE-5Gplus/i5p-hla-api/blob/main/WP3_SecurityOrchestrator_UMU/security_orchestrator.json

Briefly summarised, the SMD E2E Policy Orchestrator APIs are the following:

- **meservice**: point at which the MSPL-OP is received (from E2E Security Orchesrtator), the MSPL-OP orchestration manager is instantiated and the orchestration process finalizes with the orchestration plan (Calculated Security Assets, conflicts & dependency detected ans Translation procedure).

Besides, logging information for request traceability can be provided through a message queue (e.g., Kafka broker). Current version uses the following JSON format:

{"timestamp": "", "from_module": "", "from_component": "", "to_module": "", "to_component": "", "incoming": "", "method": "", "data": "", "notes": ""}

5.2 SSLA Manager (TSG)

The SSLA Manager's REST API is specified in OpenAPI format on the Swagger website:

- https://app.swaggerhub.com/apis/INSPIRE-5Gplus/api-sla_management/2.0#free

The API provides SSLA lifecyle management operations (creation, update, removal) and access to the managed SSLA content. The main managed objects are actually SSLA templates as defined by the SLATemplate XML schema here:

https://bitbucket.org/specs-team/specs-utility-xml-sla-framework/src/master/schemas/SLAtemplate/

This schema is based on the WS-Addressing standard and work from the SPECS project.

5.3 Secured Network Slices for SSLA (CTTC)

The REST API for the Secured Network Slices for SSLA enabler is specified in OpenAPI format. The API is not in it its final version as integration actions can still be done. The current API version is accessible on the INSPIRE-5Gplus Github repository:

- https://github.com/INSPIRE-5Gplus/i5p-hla-api/tree/main/WP3_SecureNetworkSliceSSLA

Briefly summarised, the Secure Network Slice for SSLA API is structured in the following three main keywords:

- /nst: this root-path gathers the set of actions to manage the Network Slice Templates (NSTs) available to be deployed.
- /ssla: this root-path defines any action related to the SSLA information management, essentially to get the available SSLAs to be used with the NSTs.
- /sec_nsi: this root-path gather the multiple actions to manage actions related to the Secured Network Slice Instances (SecNSI) such as to get the existing ones from the Database or to trigger the deployment and termination of SecNSIs.

5.4 Katana Slice Manager (NCSRD)

The REST API of the Katana Slice Manager can be found in the following link in OPENAPI format.

- https://github.com/5genesis/katana-slice_manager/blob/master/katana-swagger/swagger.json

The complete documentation that also includes the APIs definition can be found in the project's Github page:

- https://github.com/5genesis/katana-slice_manager

5.5 SFSBroker (UOULU)

The REST API for SFSBroker for the tenant resource requests is to be specified in OpenAPI format

The complete documentation that also includes the APIs definition can be found in the project's Github page:

The REST API for the SFSBroker is specified in OpenAPI format and placed in the INSPIRE-5Gplus Github repository: https://github.com/INSPIRE-5Gplus/i5p-hla-api/tree/main/WP3_SFSBroker

For instance, the following APIs correspond to the monitoring and the enforcement of the SFSBroker Framework:

- "POST /resourceRequest" initiates the resource request from the consumer side
- "POST /updateMNOs" updates the available resources from MNOs
- Metadata message - /resourceRequest in JSON form

{"tenantIdentifier": "e86dfa01c678a31185338acca50535f0", "demandQuantity": "[1,3,5,6,7,8]", "timestamp":"19-12-2021 10:34:45"}

- Metadata message - /updateMNOs JSON form

{"mnoIdentifier": "e844544243feaa56acd3345ae2a2eae", "availaibleQuantity": "[2,3,56,6,77,4,5]", "timestamp":"19-12-2021 10:34:45"}

5.6 DiscØvery (CLS)

For the development of the DiscØvery API we aim to develop a REST API service. The REST API service will be used to upload and store data that are relevant to the security functions of the enabler, such as network capture files to generate models, and vulnerabilities datasets. The REST API schema is specified in OpenAPI. The current API is still under specification and development, and not yet finalized, but can be viewed on the open-source repository of DiscØvery.

The source code of the DiscØvery can be found on the following link:

- https://github.com/CyberLens/Discovery

The current version of the API of DiscØvery is specified using the OpenAPI format can be viewed on the following link: https://github.com/CyberLens/Discovery/API.md

The following APIs are currently supported by the enabler:

- “POST /model” accepts networks to generate models for security analysis
- “POST /vulnerabilities” updates the vulnerability database connected to DiscØvery
- “POST /insights” updates the cyber-insights repository of the enabler

5.7 Policy Framework (UMU)

Policy framework provides REST APIS to refine HSPL-OP into MSPL-OP, to translate MSPL-OP into final configurations of the underlying infrastructure (asset´s configurations) as well as to detect policy conflicts. Besides, it also provides end-points to store and retrieve policy instances, templates, refinement and translation results as well as policies enforcement status. Templates can be used by other components to build new security policies whereas the rest of the information eases traceability and consistency. These APIs are used during different processes like the ones described on Sections 2.2 and 3.5.

The OpenAPI definition to perform refinement operations and policy-based enforcement operations related to HSPL-OP can be found at:

https://github.com/INSPIRE-5Gplus/i5p-hla-api/blob/main/WP3_PolicyFramework_UMU/h2m_service.json

Briefly summarised, the E2E Policy Framework APIs are the following:

- **h2mservice:** It is the Refinement Procedure, transform HSPL-OP into MSPL-OP
- **h2eservice:** It is the Enforcement Service for HSPL-OP policies, it will use above service for its purpose.

The OpenAPI definition to perform translation operations and policy-based enforcement operations related to MSPL-OP can be found at:

- https://github.com/INSPIRE-5Gplus/i5p-hla-api/blob/main/WP3_PolicyFramework_UMU/m2l_service.json

The OpenAPI definition to perform conflict detection operations can be found at:

- https://github.com/INSPIRE-5Gplus/i5p-hla-api/blob/main/WP3_PolicyFramework_UMU/cdt_service.json

Briefly summarised, the Policy Framework APIs are the following:

- **m2lservice:** It is the Translation Procedure, it receives an MSPL-OP and it is translated to final security asset configurations.
- **m2eservice:** It is the Enforcement Service for MSPL-OP policies, it will use above service for its purpose.
- **mcdtservice:** It is the service in charge of detecting and managing conflicts and dependencies at MSPL level.

Besides, logging information for request traceability can be provided through a message queue (e.g., Kafka broker). Current version uses the following JSON format:

{"timestamp": "", "from_module": "", "from_component": "", "to_module": "", "to_component": "", "incoming": "", "method": "", "data": "", "notes": ""}

5.8 Security Monitoring Framework (MI)

The REST API for the Security Monitoring Framework enabler is specified in OpenAPI format and placed in the INSPIRE-5Gplus Github repository:

- https://github.com/INSPIRE-5Gplus/i5p-hla-api/tree/main/WP3_SecurityMonitoringFramework_MI

The OpenAPI definition for the internal communication between two components MMT-Probe and MMT-Operator can be found at:

- https://github.com/INSPIRE-5Gplus/i5p-hla-api/blob/main/WP3_SecurityMonitoringFramework_MI/MMT-1.6.13-resolved.yaml
- https://app.swaggerhub.com/apis/strongcourage/MMT/1.6.13

The OpenAPI definition for the Security Monitoring Framework interacting with other enablers, e.g., Security Orchestrator, Policy Management are as follows. The APIs are currently under development to be compatible with other enablers.

- https://github.com/INSPIRE-5Gplus/i5p-hla-api/blob/main/WP3_SecurityMonitoringFramework_MI/MMT-Enablers-1.0.0-resolved.yaml
- https://app.swaggerhub.com/apis/strongcourage/MMT-Enablers/1.0.0

For instance, the following APIs correspond to the monitoring and the enforcement of the Security Monitoring Framework:

- "POST /monitoring" starts a MMT-Probe without any security analysis
- "PUT /monitoring" starts (if not run yet) or updates MMT-Probe with security analysis (e.g., to activate a security rule to detect whether there exists BitTorrent in the network traffic)
- "DELETE /monitoring" stops MMT-Probe
- "GET /monitoring" gets the current status of MMT-Probe
- "POST /reaction/forwarding" enforces a reaction
- "GET /mspl" obtains a MSPL from Policy Manager and produces a RT-SSLA
- "GET /rt-ssla/metrics" extracts metrics from a RT-SSLA
- "POST /alerts" sends alerts to Decision Engine

The Security Monitoring Framework also supports the use of KAFKA message bus servers so that the MMT-Operator can receive metadata from the MMT-Probe via this type of publish/subscribe system. The KAFKA messages in JSON format provide the same data structures as specified in the OpenAPI specification. For example:

- Metadata message:

{"probe_id": "", "timestamp": "", "data_volume": "", "total_packet_count": "", "uplink_data_volume": "", "uplink_packet_count": "", "downlink_data_volume": "", "downlink_packet_count": "", "mac_source": ""}

- *Security message:*

{"probe_id": "", "timestamp": "", "property": "", "verdict": "", "type": "", "description": "", "count": ""}

5.9 Security by Orchestration for MEC (OPL)

The REST API of Security by Orchestration for MEC is specified in OpenAPI format. The REST API is currently in the development phase and further actions are envisaged. The current version is available at:

- https://github.com/INSPIRE-5Gplus/i5p-hla-api/blob/main/WP3_Security_by_Orchestration_for_MEC_OPL/openapi.yaml

Section 4.6. describe Security by Orchestration for MEC approach that will be provided by the Security by Orchestration for MEC enabler. The API for this enabler is as follows:

- “POST /placement” calculates the placement of MEC applications and its instances on the hierarchical edge infrastructure

Current version uses the following JSON format:

{"topology":{"regions":[{"id":"","latency":0,"edge_servers":[{"id":"","edge_server_id":""}],"central_office":[{"id":"","latency":0,"edge_servers":[{"id":"","edge_server_id":""}],"service_area":[{"id":"","latency":0,"edge_servers":[{"id":"","edge_server_id":""}]}]}]}],"edge_servers_specification":[{"edge_server_id":"","name":"","vCPU":0,"cost_vCPU":0,"max_OCPU":0,"RAM":0,"max_ORAM":0,"storage":0,"GPU":0,"HSM":0,"power_consumption":0,"backups":0}]},"mec_applications":[{"index":"","security_level":"","latency":0,"isolation_level":0,"qos":0,"migration_cost":0,"instances":[{"index":"","preferred_location":"","vCPU":0,"RAM":0,"storage":0,"HSM":0,"GPU":0}]}]}

5.10 I2NSF IPSEC (TID)

Based on the concept introduced in Section 4.1 for model-driven data management, this enabler uses a YANG model encoded in XML format and transported with a NETCONF interface for management of the agents on the network. This approach allows easy integration of the I2NSF controller with existing SDN controllers and orchestrator in network domains.

The specification of the YANG model used by the enabler is a reduced version of the RFC9061. The following example (*Figure 34*) shows a NETCONF based XML model to add a IPSec entry into the security agent.

```
<config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<ietf-ipsec xmlns="http://example.net/ietf-ipsec" xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
<ipsec>
<sad>
 <sad-entry>
      <spi>SPI_IN</spi> <!-- SPIa1=SPI1 in A-->
      <anti-replay-window>32</anti-replay-window>
      <rule-number>0</rule-number>
      <local-addresses>
           <start>REMOTE_INTERNAL</start>
           <end>REMOTE_INTERNAL</end>
      </local-addresses>
      <remote-addresses>
           <start>LOCAL_INTERNAL</start>
           <end>LOCAL_INTERNAL</end>
      </remote-addresses>
      <next-layer-protocol>TCP</next-layer-protocol>
      <local-ports>
           <start>0</start>
           <end>0</end>
      </local-ports>
      <remote-ports>
           <start>0</start>
           <end>0</end>
      </remote-ports>
      <security-protocol>esp</security-protocol>
      <esp-sa>
           <encryption>
```

```
            <encryption-algorithm>des</encryption-algorithm>
            <key>ENC_KEY</key>
            <iv>VECTOR</iv>
        </encryption>
    </esp-sa>
    <mode>TUNNEL</mode>
    <tunnel>
    <local>REMOTE_ADDRESS</local>
    <remote>LOCAL_ADDRESS</remote>
    </tunnel>
    <sad-lifetime-soft>
        <bytes>100000</bytes>
        <packets>1000</packets>
        <added>120</added>
        <used>111</used>
    </sad-lifetime-soft>
    <sad-lifetime-hard>
        <bytes>200000</bytes>
        <packets>2000</packets>
        <added>140</added>
        <used>121</used>
    </sad-lifetime-hard>
 </sad-entry>
</sad>
</ipsec>
</ietf-ipsec>
</config>
```

Figure 34: NETCONF based XML model to add a IPSec entry into the security agent

5.11 MTD Controller (ZHAW)

The MTD controller (MOTDEC) API is defined following the OpenAPI format. The API definition can be opened with the Swagger editor and is available in the INSPIRE-5Gplus Github repository:

https://github.com/INSPIRE-5Gplus/i5p-hla-api/blob/main/WP3_MOTDEC/WP3_motdec.yamlThe API is interfaced to a service subscriber, generally a slice manager or an orchestrator, providing the needed information on the resources to be protected. Such resources can be object descriptions of network slices, network services, or VNFs. These are mapped to infrastructure-related objects, such as VIMs or switches, providing their resource availability, resource usage, and dataflow monitoring. The monitoring data is collected through periodic API calls and added to a list of objects defined as the *resource history*. The structured data is then fed to the optimizer of security functions (OptSFC), and provides the different MTD actions that can be performed. Based on this, OptSFC will use the API to provide its decisions on the actions to take. This is used both during the pre-training phase of OptSFC, to find an optimal MTD security policy, and at runtime for the deployment of the security policy on the 5G network.

5.12 Virtual Channel Protection (TSG)

The Virtual Channel Protection enabler is deployed (or updated) by the SMD Security Orchestrator as a policy enforcement function - cf. 4.3 - in the data plane. In the context of INSPIRE-5Gplus, it is implemented as a (D)TLS proxy that can interact with a remote KMS (Key Management Server).

Although it does not expose an API for remote interactions on its own, it is designed to be deployed and managed through Kubernetes (K8s) API as a container, using a Kubernetes CustomResourceDefinition (CRD) as API resource description specification. For a later version, we are also considering the use of the TOSCA VNF standard as defined by ETSI GS NFV-SOL 001 [36], more precisely using a K8s CRD encapsulated in a TOSCA Containerized VNF format.

The K8s CRD defines two *kinds* - Ingress and Egress, depending on whether the proxy secures communications going out of a K8s cluster (*Egress* kind) or going in *(Ingress* kind). The CRD defines the following properties:

- Listening_address (string): UDP/TCP listening address in the form *<host>:<port>*
- Protocol (string): (D)TLS protocol/version. E.g., *DTLS_1.2* (DTLS implies UDP, TLS implies TCP)
- Keystore (associative array): private key store; the private key and certificate are used for authentication to (D)TLS peers if *kms* undefined, else to the KMS.
    - Provider (string, optional): keystore provider ID, i.e., default PEM provider or one of the registered PKCS#11 providers if any
    - Parameters (associative array): parameters depending on the keystore provider
        - If provider is the default (PEM),
            - Private_key (string): private key in PEM private key format. A Kubernetes-ready KMS provider or secure secret management solution like Hashicorp Vault should be enabled on the K8s cluster used as deployment target, in order to protect the key content properly.
            - Certificate (string): certificate (with CA chain) in PEM format
        - Else the parameters depend on the PKCS#11 provider (e.g., HSM vendor)
- Trusted_cas (string): Trusted CAs (Certificate Authorities) for authenticating (D)TLS peers if *kms* undefined, else the KMS.
- cipher_suites (array of strings): enabled (D)TLS cipher suites, e.g., TLS_ECDHE_PSK_WITH_AES_128_CBC_SHA256,TLS_DHE_PSK_WITH_AES_128_GCM_SHA256
- Kms (optional associative array): KMS-specific settings, if attribute-based encryption is required
    - Kms_url (URL): KMS endpoint address, must be a *https://* URL
    - Labeling_rules (array, optional): confidentiality labelling rules applied to data sent, required only for *Egress* kind; each labelling rule is an associative array of:
        - Destination (string): destination address (*<host>:<port>*) to which the rule applies
        - Label (string): confidentiality label applied to data sent to the destination above. The label is used as parameter in cryptographic key requests to the KMS. For example, in a publish-subscribe protocol, you can use the topic name as label in order to encrypt each topic under a different key, and therefore enforce a cryptographic access control.

6 Conclusions

This deliverable presents the drivers and models needed for managing end-to-end and multi-domain security defined by security policies and SSLAs. It considers the management of the security requirements of slices, and the monitoring and assessment of security policies and SSLAs during operation.

The formalisms used to define security policies are described, as well as how they are translated and used by the Security orchestration and monitoring functions. The HSPL models high-level security requirements, priorities and dependencies in form of high-level orchestration security policies, independent from the underlying technologies. To be able to automate the security management process, this high-level specification needs to be translated to a formalism (i.e., MSPL) less-abstract model that remains independent of the underlying infrastructure. Then, it can be translated to the low-level configurations using translator plug-ins for each asset that needs to be managed. The low-level infrastructure information needs to be obtained from the data services or provided by the human operators. An example is given that shows how the translation is done from MSPL-OP to I2NSF controller IPSec configurations.

The Security Orchestrator can also request the translation to other formalisms recognised by the VIM, VNFM and NFV orchestrators so that the different resources and functions can be managed to ensure the respect of the specified security policies and agreements.

Orchestration covers different tasks at various levels:

- Virtualised Infrastructure Manager (VIM): controlling and managing the NFVI compute, storage and network resources
- Virtualised Network Function Manager (VNFM): VNF instances life-cycle management
- Network Function Virtualisation (NFV) Orchestrator: Network Service (NS) life-cycle management and policy management for NS instances
- Security Orchestrator (SO): support the fine-grained, end-to-end and multiple-domain security deployment and configuration for enforcing and verifying that the security policies and SSLAs defined by the verticals are respected.
- at E2E level:
    - Identification of Involved Security Management Domains for enforcement
    - Conflict and Dependency detection
    - Refinement from high-level policy to mid-level policy
    - Creation of E2E enforcement plan
    - Delivery of respective enforcement plan to each involved SMD
- at SMD level:
    - Identification of Security Assets capable of deploying the policy
    - Intelligent selection of best fitted asset based on context and requirements
    - Conflict and Dependency detection
    - Translation to low-level policy (e.g., TOSCA, final security asset configuration)
    - Enforce the orchestration policies.

The SSLAs have been defined to manage the security agreements and automate the security life-cycle of network Slices provided by the operators to the users in the different vertical domains and applications. For this, a refinement process is defined that allows tuning the input policy specified in MSPL so that the Service Level Objectives are met.

To complement this, RT-SSLAs are defined to allow the monitoring function to assess the SSLAs compliance during operation. The RT-SSLAs correspond to rules and algorithms that are used to inspect the network exchanges, and system and application traces, to detect any deviations or violations involving both security breaches and non-operating security functions.

Optimisation models and mechanisms are also presented based on chaining of virtualised security functions and micro services, and Multi-access Edge Computing (MEC). These mechanisms, based on distributed processing, MEC and micro-service chaining, are important for improving the performance of the network management actions, but also for improving the flexibility that facilitates the maintenance, upgrading and multi-party provision of different parts of the system and deployed applications.

AI/ML techniques are used or can be used by many of the enablers presented in this document. This is especially true for the detection of anomalies and the optimisation of the management of the network and its functions. This is not discussed in this document and will be covered in the future deliverables D3.3 and D3.4 [1]. Here we focus more on data models and tool functionality.

Finally, each of the enabler's API specifications are described and a link is provided to obtain their full specification. These APIs allow the interoperation between the different enablers and can serve to construct the demonstrators that show how the security management can be automated to obtain what can be called *closed-loop security management* operation.

The enabler providers in the INSPIRE-5Gplus project are implementing and testing their enablers. These providers are also working together to integrate the enablers to demonstrate how they interoperate and allow building close loop Zero-touch Service and network Management solutions (ZSM). There will be three main demonstrators covering 1) end to end policy-based security management, 2) trust and liability-based management, and 3) Moving Target Defence (MTD) management. The demonstrators 1 and 3 practically involve all the demonstrators and APIs described.

References

[1] The deliverables D3.3 (5G security new breed of enablers) and D3.4 (Smart 5G Security) will be made available in the web page: https://www.inspire-5gplus.eu/public-deliverables/

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

[3] http://www.anastacia-h2020.eu/

[4] http://hdl.handle.net/10201/101661

[5] Zarca, A. M., Bernabé, J. B., Ortíz, J., & Skarmeta, A. Policy-based Definition and Policy for Orchestration Final Report. Anastacia H2020 European project deliverable D2.5. (online: http://www.anastacia-h2020.eu/deliverables/ANASTACIA-WP2-T2.1-UMU-D2.5-PolicyBasedDefinitionAndPolicyForOrchestrationFinalReport-v1.0.pdf).

[6] https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=tosca

[7] FP7 Project: https://cordis.europa.eu/project/id/610795

[8] https://www.nist.gov/cyberframework

[9] https://www.gsma.com/newsroom/all-documents/generic-network-slice-template-v2-0/

[10] Wendland, F., & Banse, C. (2018, August). Enhancing NFV orchestration with security policies. In Proceedings of the 13th international conference on availability, reliability and security (pp. 1-6).

[11] https://www.inspire-5gplus.eu/wp-content/uploads/2021/05/i5-d3.1_5g-security-assets-baseline-and-advancements_v0.7.pdf

[12] P.Alemany, D.Ayed, R.Vilalta, R.Muñoz, P.Bisson, R.Casellas, R.martínez, "Transport Network Slices with Security Service Level Agreements," 22nd International Conference on Transparent Optical Networks (ICTON), Bari, Italy, 2020.

[13] Tharaka Hewa et. al.,"How DoS attacks can be mounted on Network Slice Broker and can they be mitigated using blockchain?" , 2021 IEEE 32nd Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)

[14] https://doi.org/10.5281/zenodo.4569519

[15] Molina Zarca, A., Bagaa, M., Bernal Bernabe, J., Taleb, T. and Skarmeta, A., 2020. "Semantic-Aware Security Orchestration in SDN/NFV-Enabled IoT Systems. *Sensors*", 20(13), p.3622.

[16] https://www.enisa.europa.eu/publications/standards-and-tools-for-exchange-and-processing-of-actionable-information

[17] https://datatracker.ietf.org/doc/html/rfc6020

[18] https://datatracker.ietf.org/doc/html/rfc4741

[19] https://www.w3.org/TR/1998/REC-xml-19980210.html

[20] https://www.gsma.com/newsroom/wp-content/uploads//NG.116-v2.0.pdf

[21] https://www.iso.org/standard/71616.html

[22] https://oasis-open.github.io/cti-documentation/

[23] https://www.enisa.europa.eu/publications/information-sharing-and-common-taxonomies-between-csirts-and-law-enforcement/

[24] https://github.com/protocolbuffers/protobuf/releases/tag/v3.17.0

[25] https://datatracker.ietf.org/doc/html/rfc8040

[26] https://datatracker.ietf.org/meeting/98/materials/slides-98-rtgwg-gnmi-intro-draft-

openconfig-rtgwg-gnmi-spec-00

[27] https://datatracker.ietf.org/meeting/101/materials/slides-101-netconf-grpc-network-management-interface-gnmi-00

[28] https://www.etsi.org/deliver/etsi_gs/CIM/001_099/004/01.01.01_60/gs_CIM004v010101p.pdf

[29] http://www.musa-project.eu/

[30] https://www.musa-project.eu/repositories

[31] Wei Zhang, Guyue Liu, Wenhui Zhang, Neel Shah, Phil Lopreiato, Grégoire Todeschi, K. K. Ramakrishnan, Timothy Wood: OpenNetVM: A Platform for High Performance Network Service Chains. HotMiddlebox@SIGCOMM 2016: 26-31

[32] https://www.dpdk.org/

[33] ETSI GS MEC 003: "MEC - Framework and Reference Architecture" https://www.etsi.org/deliver/etsi_gs/MEC/001_099/003/02.01.01_60/gs_MEC003v020101p.pdf

[34] Edge Computing for 5G Networks, February 2021.URL: https://bscw.5g-ppp.eu/pub/bscw.cgi/d397473/EdgeComputingFor5GNetworks.pdf

[35] ETSI GS MEC 010-2 V2.1.1: "MEC - MEC Management; Part 2: Application lifecycle, rules and requirements management". URL: https://www.etsi.org/deliver/etsi_gs/MEC/001_099/01002/02.01.01_60/gs_MEC01002v020101p.pdf

[36] ETSI GS NFV-SOL 001 - Network Functions Virtualisation (NFV) Release 2; Protocols and Data Models; NFV descriptors based on TOSCA specification, 2018

