Advanced Networked Agents for Security and Trust Assessment in CPS/IoT Architectures

D6.4

Final Technical integration and validation Report

This deliverable presents the final results of ANASTACIA Task 6.1 - Technical integration and validation. Revised version of D6.1 after intermediate evaluation. The updated technical specifications, development outcomes and testing results of the Integrated ANASTACIA Platform, are provided.

Distribution level	PU
Contractual date	30.09.2019 [M33]
Delivery date	11.12.2019 [M36]
WP / Task	WP6 / T6.1
WP Leader	UBITECH
Authors	Giannis Ledakis, Konstantinos Theodosiou (UBITECH), Jesús Villalobos, Ruben Trapero (ATOS), Federico Sismondi (MAND), Jorge Bernal, Alejandro Molina (UMU), Rafael Marín Pérez (ODINS), Piotr Sobonsky (UTRC), Diego Rivera (MONT), Miloud Bagaa (AALTO)
EC Project Officer	Carmen Ifrim carmen.ifrim@ec.europa.eu
Project Coordinator	Softeco Sismat SpA Stefano Bianchi Via De Marini 1, 16149 Genova – Italy +39 0106026368 stefano.bianchi@softeco.it
Project website	www.ANASTACIA-h2020.eu

ANASTACIA has received funding from the European Union's Horizon 2020 research and innovation programme under Grant Agreement N° 731558 and from the Swiss State Secretariat for Education, Research and Innovation.

This document only reflects the ANASTACIA Consortium's view.
The European Commission is not responsible for any use that may be made of the information it contains.

Table of contents

Index of figures

Index of tables

PUBLIC SUMMARY

This deliverable is the final reporting for the outcomes of task T.6.1. In the first reporting document, ANASTACIA Deliverable D6.1[1] , we provided the integration and technical testing plan of the ANASTACIA framework, along with the suggested development cycle and the tools that can be used to support both the collaborative development. The integration plan has been created using ANASTACIA Deliverable D1.3[2] (delivered at M9) as starting point and, by identifying and specifying the necessary integration points between components, also by constructing the integration approach that will be followed. As architecture has changed a lot since then, there has been a continuous sync with the advancements of task T1.3 and that results to be reported in D1.5 (at M36) were considered in this document.

This deliverable extends and updates D6.1 with the results about integration, deployment and validation.

1 Introduction

1.1 Aims of the Document

In this document we provided the output of the final efforts of executing the integration, testing and technical evaluation plan of ANASTACIA framework. For the creation of the integration plan technical partners collaborated in many different occasions, especially for the definition of the necessary integration points between components, but also for a common approach regarding testing and evaluation. The starting point for the integration plan was ANASTACIA Deliverable D1.3[2] and the work done in task T1.3: Architectural Design, continued with the specification and development of the mechanisms described in WP2, WP3, WP4 and WP5. Then, in WP6 we defined the integration plan of the discrete mechanisms and software components, with the agreement on identified interfaces, the usage of selected tools, and the setup of a development lifecycle scheme that includes source code management, continuous integration, source-code quality control, release management and ticketing. As this effort was presented in D6.1[1] in this document we provide mostly the results, but we also provide all the updates on the integration interfaces, deployment and integration testing of the ANASTACIA framework.

1.2 Applicable and Reference Documents

This document refers to the following documents:

- Grant Agreement N°731558 – Annex I (Part A) – Description of Action[3]
- D1.2 – User-centred Requirement Initial Analysis[4]
- D1.3 - Initial Architectural Design[2]
- D2.8 - Secure Software Development Guidelines Final Report[9]
- D1.4 - Final User-centred Requirements Analysis[5]
- D6.1 - Initial Technical integration and validation Report[1]

1.3 Revision History

Version	Date	Author	Description
0.1	14.08.2019	Giannis Ledakis (UBI)	Skeleton of expected contents
0.2	5.09.2018	Giannis Ledakis (UBI)	Draft content in sections 2, 3 and 5
0.3	22.09.2018	Giannis Ledakis (UBI)	Updates section 2
0.4	5.10.2018	Giannis Ledakis (UBI), ALL	Integration tests section 5.3, first draft
0.5	16.10.2018	Giannis Ledakis (UBI), ALL	Section 3 and Section 4 content integrated
0.6	29.10.2018	Giannis Ledakis (UBI), ALL	Section 5 added missing content

Version	Date	Author	Description
0.7	07.11.2019	Giannis Ledakis (UBI), ALL	Updating integration tests section 5.3 based on input from all partners, after meeting@Cork
0.8	26.11.2019	Federico (MI)	Updated DSPS interface information
0.9	27.11.2019	Miloud Bagaa (Aalto)	Draft content in sections 2, 3 and 4.
0.9.3	28.11.2019	Ruben Trapero (ATOS)	Updating section 2.1 and first review by ATOS
0.9.5	10.12.2019	Giannis Ledakis (UBI), ALL	Deliverable Finalized; Updating based on first review comments, updated component's information, updated section 4.1
1.0	11.12.2019	Stefano Bianchi	Final proofreading

1.4 Acronyms and Definitions

Acronym	Meaning
API	Application Programming Interface
CISO	Chief Information Security Officer
DSPS	Dynamic Security and Privacy Seal
HSPL	High Security Policy Language
IoT	Internet of Things
MANO	Management and Orchestration
MAS	Mitigation Action Service
MMT	Montimage Monitoring Tool
MSPL	Medium Security Policy Language
NFV	Network Function Virtualization
OS	Operating System
REST	REpresentational State Transfer
RPC	Remote Procedure Call

Acronym	Meaning
SAS	Security Alert Service
SDN	Software Defined Networking
SIEM	Security Information and Event Management
UI	User Interface
VDSS	Verdict and Decision Support System
VM	Virtual Machine

2 Platform Integration Overview

A modern software system like ANASTACIA is a combination of different subsystems cooperating so that the overall framework is able to deliver the needed functionalities. These subsystems need to be integrated in such a way that they can support common business processes and data sharing across whole framework.

As described in D6.1[1] , in ANASTACIA we tried to combine the desired characteristics of the different approaches and create an integration that is based on both direct communications between components (Star architecture) and also asynchronous, loosely-coupled integration through message broker usage. This approach helped us on achieving characteristics of event-driven architecture and enable ANASTACIA for the proper supporting of production, detection, consumption of, and reaction to events.

2.1 ANASTACIA Integrated Framework Architecture

2.1.1 The Envisioned Platform

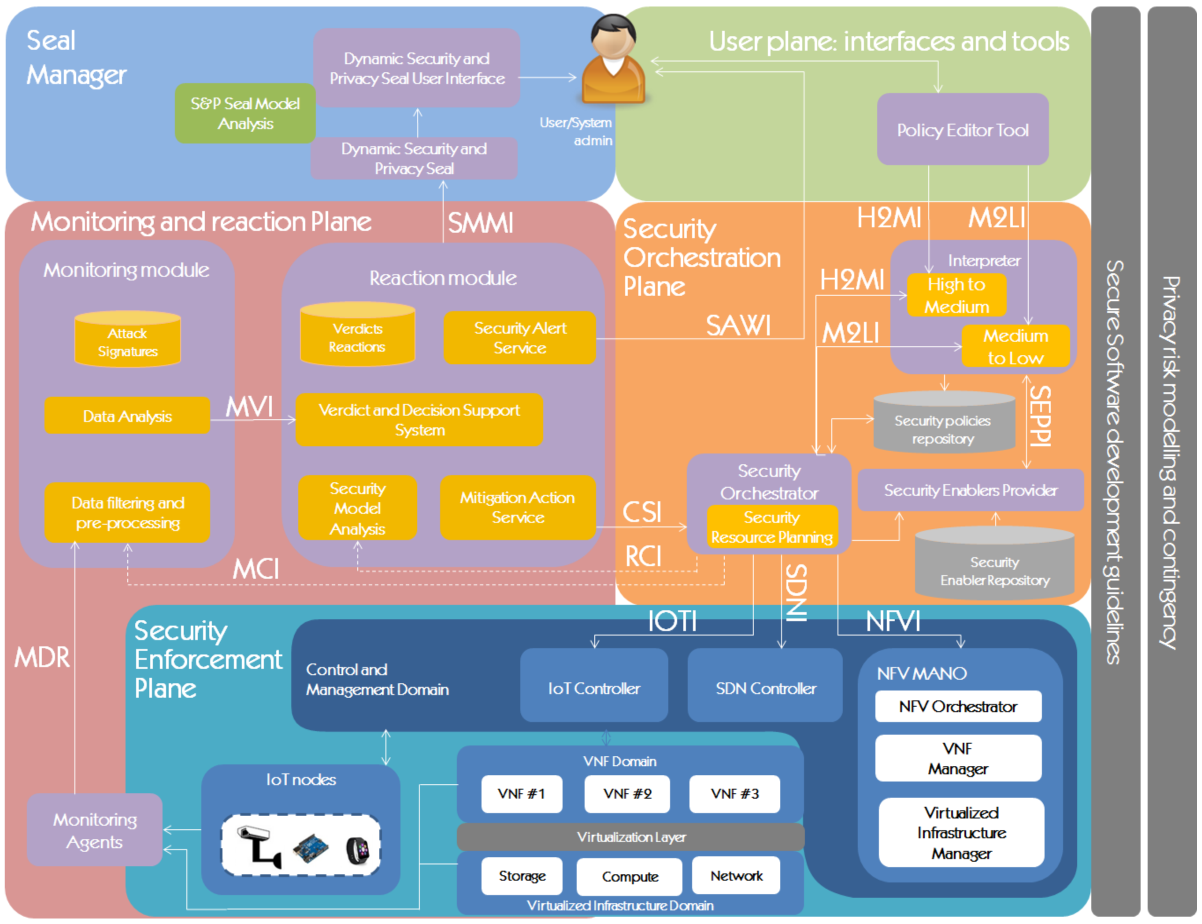
An important role for the decision regarding the architectural approach followed in ANASTACIA was the clarification of the platform vision regarding the way that the ANASTACIA as whole will be used. Based on the analysis of initial requirements and the use cases reported in Deliverable D1.2[4] , in Deliverable D1.3[2] five main activities to be supported by the platform were identified, with each of them utilizing specific components. For the integration planning it is important to clarify how each of these components interconnects to achieve these identified activities that are shortly presented below;

- **Security policy set-up activity**. This is the initial process triggered once a security policy has been defined by the user. In this process the policy has to be configured in the platform in order to be enforced. The interpretation of the security policy claims, the configurations required to monitor the security controls associated to a policy or the definition of thresholds to identify policy violations, are some activities carried out by this process.
- **Security policy orchestration activity**. Once the policy has been defined, it is necessary to enforce the controls specified within the policy. To orchestrate the selected IoT/SDN/NFV-based security enablers, appropriate interactions with the relevant management modules are required.
- **Security monitoring activity**. In this process the monitoring information is extracted from the devices through monitoring agents and according to the security controls interpreted from the security policy. In this activity, the monitoring data is filtered and aggregated in order to carry out its analysis and the detection of anomalies.
- **Security reaction activity**. In this process the detected anomalies are evaluated to design counter measures in order to mitigate the effects of attacks and potential threats.
- **Dynamic security and privacy seal creation activity**. In this process, relevant information about detected threats, monitored information is evaluated to create a seal that determine the level of security guaranteed/offered by an IoT platform.

The aforementioned activities, along with their sub-activities and resulting architecture of the platform are described with detail to the deliverable D1.3. In this deliverable the updated architecture is also presented in order to identify and confirm the interfaces and provide technical details about the needed integration points between the components that will allow the platform integration.

2.1.2 Integration Points

Figure 1 shows the ANASTACIA architecture which includes the interfaces between modules, as defined in D6.1, based on the architecture version delivered at M9 in D1.3.



Seal Manager
S&P Seal Model Analysis
Dynamic Security and Privacy Seal User Interface
Dynamic Security and Privacy Seal
User/System admin
User plane: interfaces and tools
Policy Editor Tool
Monitoring and reaction Plane
SMMI
Monitoring module
Attack Signatures
Data Analysis
Data filtering and pre-processing
Reaction module
Verdicts Reactions
Security Alert Service
Verdict and Decision Support System
Security Model Analysis
Mitigation Action Service
MVI
MCI
MDR
Security Orchestration Plane
H2MI
M2LI
SAWI
Interpreter
High to Medium
Medium to Low
SEPPI
Security policies repository
Security Orchestrator
Security Resource Planning
Security Enablers Provider
Security Enabler Repository
CSI
RCI
SDNI
IOTI
NFVI
Security Enforcement Plane
Control and Management Domain
IoT Controller
SDN Controller
NFV MANO
NFV Orchestrator
VNF Manager
Virtualized Infrastructure Manager
IoT nodes
Monitoring Agents
VNF Domain
VNF #1
VNF #2
VNF #3
Virtualization Layer
Storage
Compute
Network
Virtualized Infrastructure Domain
Secure Software development guidelines
Privacy risk modeling and contingency

Figure 1. ANASTACIA architecture – Interface View- initial version

During the development and integration of ANASTACIA framework, updates had been made in the architecture by adding new components and interfaces or even removing some unneeded interfaces. Figure 2 depicts an updated view of architecture of the ANASTACIA framework, which includes the latest changes. The final version of the architecture will be provided in ANASTACIA Deliverable D1.5 at the end of the project. The most important aspects that have evolved since then are summarized as follows:

- Monitoring Plane includes new components (**Incident Detector, Resource and QoS monitoring**)
- Reaction plane, where the Reaction Module is defined with more details, including components required by the Verdict and Decision Support System (VDSS) to fetch input from different parts of the ANASTACIA framework (**Assets Model, Mitigations Repository**).
- Security Orchestrator, has involved and is now detailed with the components **System Model Service,** the **Security Orchestrator Optimizer,** the **Performance Data Analytics**, and the additional components/**drivers** needed to interact with NFV, SDN and IoT functions.
- Seal Manager, where the components included have been **refined** (DSPS Agent, Seal Manager and DSPS Repository now available).
- User plane extended with more User Interfaces that have been decoupled by the backed services and are now included in this plane (**DSPS UI, Alerting and reaction dashboard, Policy Editor UI**).

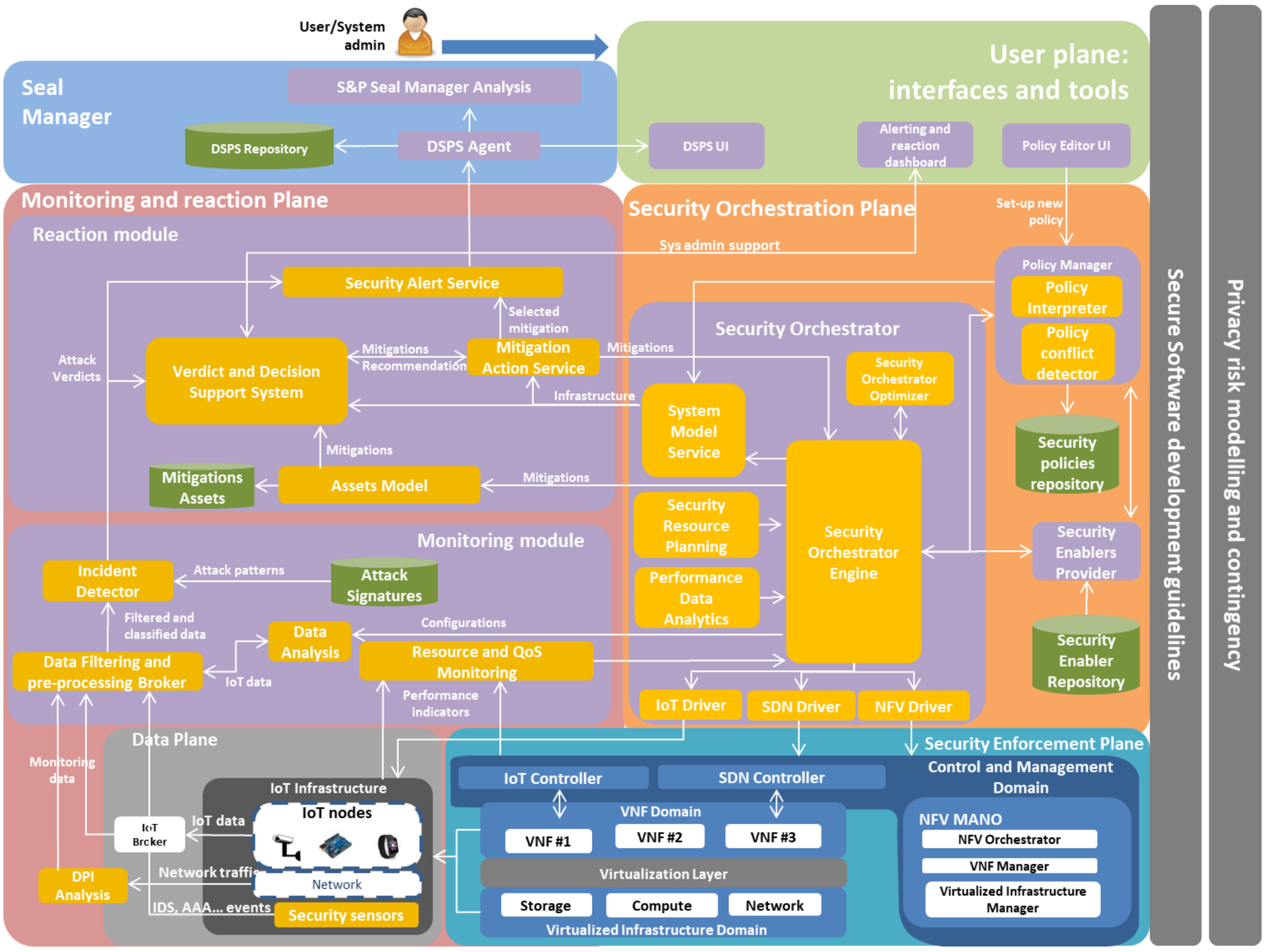


Figure 2. ANASTACIA final architecture design

Figure 3 shows the major interfaces of the platform. The colouring scheme that is used presents with white the unchanged interfaces, with yellow interfaces that were defined but changed and with green the newly added interfaces. The most important updates in the interfaces that are added are explained below.

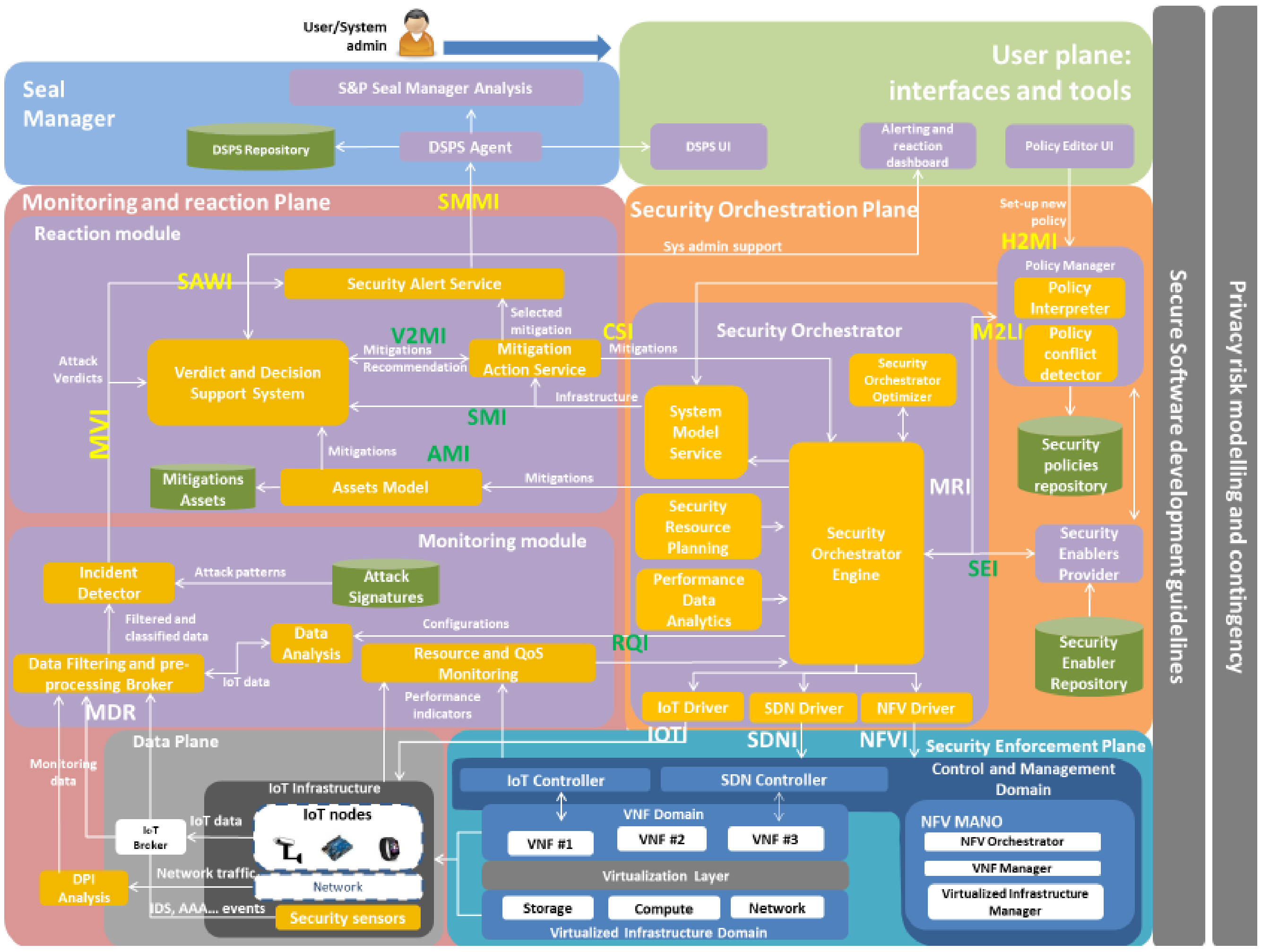


Figure 3. ANASTACIA final architecture – Interface View

More specifically, the interfaces introduced that were introduced in D1.3 and no or only minor updates were introduced are the following:

- **MSPL Reception Interface (MRI):** This interface is used by the policy interpreter to send the MSPL to the Security Orchestrator.
- **Security Enabler Provider Plugin Interface (SEPPI)**: Interface exposed by the Security Enablers Provider. It is used to get an appropriate enabler plugin during the lower policy refinement done at the Policy Interpreter, as well as providing the list of available security enablers.
- **Monitoring Data Receiver (MDR):** The interfaces that are used to allow the Monitoring Agents to provide their output to the Monitoring module. Actually, different Kafka topics are used for different monitoring agents.
- **Security Enforcement Plane Interface for IoT, SDN and NFV (IOTI, SDNI, NFVI)**: Set Interface between the IoT, SDN, NFV Drivers and the Control and Management Domain. This is a mostly a representation change as these interfaces are just updated in the sense that the connection the controllers is made through specific drivers.
    - **IoT-oriented Security Enforcement Plane Interface (IOTI):** This interface is used from the security orchestrator in order to configure the IoT controller.
    - **SDN-oriented Security Enforcement Plane Interface (SDNI):** Interfaces between the Security Orchestrator and the SDN controllers. It provides the connectivity required among the Network Virtual Functions, and some basic security reactions.

  - **NFV-oriented Security Enforcement Plane Interface (NFVI)**: This interface allows managing the security VNFs via the ETSI-oriented NFV MANO modules. The Security Orchestrator can request the enforcement of the security VNFs according to the configurations generated by the policy refinement process.

Some of the interfaces were updated based on the architectural updates and were more precisely defined:

- **High to Medium interface (H2MI):** Interface between the User Plane and the Orchestration Plane used for translating and refine policies. H2MI provide information at a high level of granularity. This interface is also used internally by the Security Orchestrator to get details about the capabilities that needs to be enforced within the IoT platform.
- **Medium to Lower interface (M2LI)**: Interface between the User Plane and the Orchestration Plane used for translating and refine policies. M2LI provides a lower level of granularity than the information provided by H2MI. This interface is also used internally by the Security Orchestrator to get details about the capabilities that needs to be enforced within the IoT platform.
- **Seal Manager Metadata Interface (SMMI)**: The interface provides the requested information to evaluate the security and the privacy in a real-time fashion. The security and privacy policies defined by the user are stored inside the policies repository and an interface is available to retrieve and set them from the seal manager. Based on the latest architecture this interface is now between the Security Alert Service (SAS) and the DPSS Agent of the Dynamic Security and Privacy Seal.
- **Security Alerts and Warnings Interface (SAWI)**: Interface between the Reaction module and the user plane which is used for the notification to the User/System admin about relevant information regarding alarms, countermeasures, etc.
- **Monitoring Verdicts Interface (MVI)**: Interface between the Monitoring module and the Reaction module used for exchanging information about detected incidents.
- **Also, the Countermeasures Suggestions Interface (CSI) has been renamed to Mitigations Enforcement Interface (acronym remained CSI)**: Interface between the Reaction module (Mitigation Action Service (MAS)) and the Security Orchestrator to exchange information about the countermeasures to be enforced in the IoT platform in order to react to certain incident.

The newly added interfaces are the following:

- **System Model Interface (SMI):** Interface of the System Model of the Security Orchestrator that is used by the Reaction Module (MAS and VDSS)
- **Resources and QoS Interface (RQI):** Interface between the Resource and QoS Monitoring -> Security Orchestrator Engine
- **Resource Monitoring Interface (RPI):** Interface between the IoT infrastructure and the Resource and QoS Monitoring
- **Mitigation Recommendations Interface(V2MI):** Interface between the VDSS and MAS that sends verdicts and recommendations for the mitigations to be performed
- **Asset Model Interface (AMI)**: Interface between the Security Orchestrator and the Assets Model

  Finally, the interfaces of configuring the motoring (MCI and RCI) were not used and therefore been deprecated in this last version of the architecture.

In the following section, the detailed technical description of all the identified interfaces is presented based on bilateral and general discussions between the technical partners.

2.2 Detailed Description of the Interfaces

This section gathers information about the interfaces required for the implementation of the integrated solution of ANASTACIA by defining the communication between the components created in WP2-3-4-5.

The following subsections describe these interfaces (organized per activity) by detailing the following information:

- **Description**: describes the purpose of the interface
- **Component providing the interface**: describes the component that is offering the described interface.
- **Consumer components**: describes the components that are using the described interface.
- **Type of interface:** REST, XML-RPC, GUI, Java API etc.
- **Input data**: describes how data that is required by the described interface (e.g.: Methods or Endpoints, values and parameters of the interface)
- **Output data**: describes the data that is returned by the described interface (e.g.: the returned data of methods or REST call)
- **Constraints:** Any security or authentication related topics regarding this interface, specifically the need to use a secure transfer protocol. Also, any other constraints (e.g. specific prerequisites, data-types, encoding, transfer rates) which apply to the interface.
- **State:** Synchronous/Asynchronous, Stream
- **Responsibilities:** Partner that is responsible for the implementation and usage of the interface

In D6.1 we provided a first version of the identified interfaces while in this deliverable we describe the updated and final versions of the interfaces as they have been actually implemented.

2.2.1 Interfaces for Policy Set-up Activity

The following tables describe the interfaces involved in the set-up of a new policy, comprising the interpretation of a security policy set-up at the editor, involving the interfaces H2MI (Table 1), M2LI (Table 2), MDCTI (Table 3), RPI(Table 4), MRI (Table 5) and SEPI (Table 6). These tables extend and update the information gathered in D1.3 and D6.1.

2.2.1.1 High to Medium Interface

Table 1. Policy Editor Tool <-> Interpreter H2M (H2MI)

High to Medium interface (H2MI)	
Description	The interface allows requesting a policy refinement from a High-level Security Policy (HSPL) to a Medium level Security Policy (MSPL), as well as to request a policy enforcement from a HSPL (avoiding to manually request M2L and MRI interfaces).
Component providing the interface	Policy Interpreter
Consumer components	Policy Editor Tool

Type of Interface	REST		
State	Synchronous		
Input data / Output Data	**Methods or endpoints** of the interface	**Parameters** of the method	**Return Values** of the method
	h2mservice h2eservice	HSPL-OP codified in XML which contains different kind of HSPL policies.	MSPL-OP codified in XML which contains the policy refinements, including dependencies as well as a list of security enablers candidates for the policies enforcement.
Constraints	The values provided in the HSPL policies must be stored in the system model in order to perform the policy refinement successfully. This interface requires the Security Enabler Provider in order to get the security enablers candidates.		
Responsibilities	○ UMU		

2.2.1.2 Medium to Lower Interface

Table 2. Security Orchestrator <-> Interpreter M2L (M2LI)

Medium to Lower interface (M2LI)	
Description	The interface allows to request a policy translation from Medium Level Security Policy Language Orchestration Policies (MSPL-OP) to specific security enablers configurations/tasks
Component providing the interface	Policy Interpreter
Consumer components	Security Orchestrator
Type of Interface	REST
State	Synchronous

Input data / Output Data	**Methods or endpoints** of the interface	**Parameters** of the method	**Return Values** of the method
	m2lservice	MSPL-OP codified in XML which contains different kind of MSPL policies with the selected security enabler for each one of them, as well as intra orchestration policy dependencies.	JSON with a mapping which includes the MSPL, the security enabler, the translation and two lists with the detected MSPL conflicts and MSPL dependencies.
Constraints	M2LI uses the system model and the Security Enablers provider interface in order to obtain the enabler plugin and perform the M2L translation. The interface also uses the conflict detector interface in order to perform the conflicts and dependencies detection.		
Responsibilities	○ UMU		

2.2.1.3 Conflict and dependencies detection Interface

Table 3. Policy Interpreter <-> Conflict detector (MDCTI)

Conflict and dependencies detection interface (MCDTI)			
Description	The interface allows to request MSPL-OP conflicts and dependencies detection.		
Component providing the interface	Conflict Detector		
Consumer components	Policy Interpreter / Security Orchestrator		
Type of Interface	REST		
State	Synchronous		
Input data / Output Data	**Methods or endpoints** of the interface	**Parameters** of the method	**Return Values** of the method
	mcdtservice	MSPL-OP codified in XML which contains different kind of MSPL policies.	JSON with two different lists for conflicts and dependencies.

Constraints	This interface requires the system model as well as the policy repository in order to perform the conflict and dependencies detection taking into account the current status of the system.
Responsibilities	○ UMU

2.2.1.4 Policy Repository Interface

Table 4. Policy Interpreter <-> Policy Repository (PRI)

Policy Repository Interface (PRI)			
Description	The interface allows to store and retrieve security policy instances, templates and their status at different levels.		
Component providing the interface	Policy Repository		
Consumer components	Policy Interpreter / Security Orchestrator / Mitigation Action Service		
Type of Interface	REST		
State	Synchronous		
Input data / Output Data	**Methods or endpoints** of the interface	**Parameters** of the method	**Return Values** of the method
	capabilities	None	JSON with the current ANASTACIA supported capabilities
	hspl-instances	String for filtering the results	JSON with the HSPL-OP/HSPL instances created as part of a refinement
	mspl-instances	String for filtering the results	JSON with the MSPL-OP/MSPL instances created as part of a refinement/translation
	Lowlevel-instances	String for filtering the results	JSON with the low-level configuration instances created as part of a translation

	policy-refinements (GET & SET)	String for filtering the results JSON with the correspondence between HSPL-OP and MSPL-OP	JSON with the requested correspondence of HSPL-OP/MSPL-OP
	policy-translations (GET & SET)	String for filtering the results JSON with the correspondence between MSPL-OP and low-level configurations	JSON with the requested correspondence of MSPL-OP/Low-level configurations
	Policy-enforcement (GET & SET)	String for filtering the results JSON with the correspondence between MSPL ID and the status	JSON with the current status of the MSPL.
Constraints	This interface requires the policy repository database.		
Responsibilities	○ UMU		

2.2.1.5 MPSL Reception Interface

Table 5. Policy Interpreter<-> Security Orchestrator

MSPL Reception Interface (MRI)	
Description	Interface used by the policy interpreter to receive the MSPL Orchestration Policies (MSPL-OP) from the Security Orchestrator.
Component providing the interface	Security Orchestrator
Consumer components	Policy Interpreter
Type of Interface	REST
State	Synchronous

Input data / Output Data	**Methods or endpoints** of the interface	**Parameters** of the method	**Return Values** of the method
	Load_MSPL	MSPL-OP codified in XML which contains different kind of MSPL policies, as well as intra orchestration policy dependencies and the security enablers candidates	The same MSPL-OP but each MSPL now contains only one security enabler candidate.
Constraints	None		
Responsibilities	○ AALTO ○ UMU		

2.2.1.6 Security Enabler Provider Interface

Table 6. Policy Interpreter <-> Security Enabler Provider (SEPI)

Security Enabler Provider Interface (SEPI)			
Description	The interface allows requesting the available plugin catalogue as well as the plugin files which implements part of the translation process from MSPL to Enabler configurations.		
Component providing the interface	Security Enablers Provider		
Consumer components	Policy Interpreter		
Type of Interface	REST		
State	Synchronous		
Input data / Output Data	**Methods or endpoints** of the interface	**Parameters** of the method	**Return Values** of the method
	getplugins	The required capability	A list of security enablers which could be able to enforce the required capability.

	getplugin	The Security Enabler Identifier	The plugin file which implements the MSPL translation.
Constraints	The plugin as a piece of software must be a python file which implements the method getConfiguration().		
Responsibilities	○ UMU (getplugin) ○ THALES (getplugins)		

2.2.2 Interfaces for Policy Orchestration and Enforcement

The following interfaces are used for the enforcement of security policies in IoT devices. Three possible ways of orchestrating or enforcing a policy can be used depending on the interface used:

- Policy enforcement using SDN controllers through the SDNI (Table 7).
- Policy enforcement using NFV-MANO modules through the NFVI (Table 8).
- Policy enforcement using IoT controllers through the IOIT (Table 9).

2.2.2.1 SDN-oriented Security Enforcement Plane Interface

Table 7. Security Orchestrator <-> SDN controllers (SDNI)

SDN-oriented Security Enforcement Plane Interface (SDNI)			
Description	This interface allows managing the SDN networking configuration via the SDN controller(s). The Security Orchestrator requests the enforcement of the SDN traffic flow rules received as outcome of the policy refinement process. This interface is also used to retrieve the network information from the SDN controllers.		
Component providing the interface	SDN controller(s) : ONOS		
Consumer components	Security Orchestrator		
Type of Interface	REST		
State	Synchronous		
Input data / Output Data	**Methods or endpoints** of the interface	**Parameters** of the method	**Return Values** of the method

	Flow_dropping Flow_mirroring Flow_forwarding Flow_bandwidth_limitation	JSON with the list of parameters required to manage the flows	JSON with method execution results
Constraints	Regarding ONOS north-bound APIs, authentication based on user and password is required for issuing commands.		
Responsibilities	○ AALTO		

2.2.2.2 NFV-oriented Security Enforcement Plane Interface

Table 8. Security Orchestrator <-> NFV MANO modules (NFVI)

NFV-oriented Security Enforcement Plane Interface (NFVI)			
Description	This interface allows to manage the security VNFs via the ETSI-oriented NFV MANO modules. The Security Orchestrator can request the enforcement of the security VNFs according to the configurations generated by the policy refinement process.		
Component providing the interface	NFV MANO (Management and Orchestration) modules: OSM (under evaluation)		
Consumer components	Security Orchestrator		
Type of Interface	REST		
State	Synchronous		
Input data / Output Data	**Methods or endpoints** of the interface	**Parameters** of the method	**Return Values** of the method
	Onboard/export Virtual Network Function Descriptor (VNFD)/ Network Service Descriptor (NSD) Create/Delete Network Service (NS)	Data packages defining NSD/VNFD. Information about the NS to manage	Method execution results

	Execute configuration primitives on Network Services.		
Constraints	OSM authentication is based on user and password is required for issuing commands. Also, HTTPs is enabled. Additional security features can be considered.		
Responsibilities	○ AALTO ○ THALES		

2.2.2.3 IoT-oriented Security Enforcement Plane Interface

Table 9. Security Orchestrator <-> IoT controllers (IOTI)

IoT-oriented Security Enforcement Plane Interface (IOTI)			
Description	This interface allows managing the configuration of IoT nodes via specific IoT controllers, as well as register new IoT devices. The Security Orchestrator requests the enforcement of the security controls within the IoT nodes according to the configurations generated by the policy translation process.		
Component providing the interface	IoT controllers		
Consumer components	Security Orchestrator		
Type of Interface	REST		
State	Synchronous		
Input data / Output Data	**Methods or endpoints** of the interface	**Parameters** of the method	**Return Values** of the method
	register	JSON which includes the information of the new IoT device.	Registration notifications
	resources	JSON with the required resource request or modification	IoT resource values/result of the operation (turn off, disable radio, bootstrapping), in plain text

	honeynet	None	A honeynet model in order to build IoT honeynet policies.
Constraints	The system model is required in order to register the IoT devices properly.		
Responsibilities	○ UMU/OdinS		

2.2.3 Interfaces for Monitoring

The following tables describe the interfaces involved in the Monitoring processes. The collection of monitoring data is described in the Monitoring Data Receiver (MDR) set of interfaces (Table 10), while resources monitoring is part of RQI (Table 11) interface.

2.2.3.1 Monitoring Data Receiver

Table 10. Monitoring Agents <-> Monitoring Module (MDR)

Monitoring Data Receiver (MDR)			
Description	This integration point is needed in order to allow the Monitoring Agents to provide their output to the Monitoring Module through the Data Filtering Component. It is not a single interface but a collection of Kafka topics[1] used for the collection of the data from a diverse set of monitoring agents.		
Component providing the interface	Kafka Message Broker		
Consumer components	Data Filtering Component (Monitoring Service)		
Type of Interface	Type of Kafka Topic depends of the Monitoring Agent		
State	Asynchronous		
Input data / Output Data	**Methods or endpoints** of the interface	**Parameters** of the method	**Return Values** of the method
	One topic per each monitoring agent • AAA events - *IoTBroker*	Parameters agreed per agent, in json format	N/A

[1] https://kafka.apache.org/documentation/

	• Deep Packet Inspection scanning – *security.report* • Data Analysis - *UTRCVerdicts*		
Constraints	Each monitoring agent should be able to connect to the Kafka Broker		
Responsibilities	○ UBITECH		

2.2.3.2 Resources Monitoring and QoS Interface

Table 11. Resource and QoS Monitoring Module <-> Security Orchestrator (RQI)

Resources and QoS Interface (RQI)			
Description	This interface enables the security orchestrator to real-time monitoring of the resource utilization of the VNFs that includes the CPU usage percentage, memory details available, used and total.		
Component providing the interface	Resource and QoS Monitoring Module		
Consumer components	Security Orchestrator		
Type of Interface	REST		
State	Synchronous		
Input data / Output Data	**Methods or endpoints** of the interface	**Parameters** of the method	**Return Values** of the method
	Retrieve_Resources	JSON that includes the information of the VNF.	JSON with method execution results
Constraints	The VNF is up and running. Also, the security orchestrator should have right access to the different deployed VNFs.		
Responsibilities	○ AALTO		

2.2.4 Interfaces for Reaction Activity

The following interfaces are used for exchanging relevant data required for the fulfilment of a security policy within an IoT platform. This includes:

- The notification of detected incidents between the Monitoring and the Reaction modules through the MVI (Table 12).
- The notification of alerts and countermeasures from the Reaction module to the User/System admin through the SAWI (Table 13).
- The list of mitigations capable of mitigating an ongoing incident along with a suitability score through V2MI (Table 14).
- The computed countermeasures to the Security Orchestrator for its enforcement through CSI (Table 15).
- The information related with the System Model and the Security Enablers deployed in the network through SMI (Table 16).
- The latest information about mitigation strategies, capabilities and actions through AMI (Table 17).

2.2.4.1 Monitoring Verdicts Interface

Table 12. Monitoring -> Reaction definition (MVI)

Monitoring Verdicts Interface (MVI)			
Description	This interface provides the required monitoring information from the Monitoring to the Reaction Module. The transferred data is mainly composed of the verdicts of the security properties tested on the network.		
Component providing the interface	Incident Detector (Monitoring Module)		
Consumer components	Verdict and Decision Support System (Reaction Module)		
Type of Interface	RabbitMQ queue		
State	Asynchronous		
Input data / Output Data	**Methods or endpoints** of the interface	**Parameters** of the method	**Return Values** of the method
	RabbitMQBolt, listening to the eu.anastacia.siem_server_output queue	JSON including: alarmID, BacklogID, AlarmEvent. Alarm event containing related to the incident detected.	None
Constraints	None		

Responsibilities	○ MONT ○ ATOS

2.2.4.2 Security Alerts and Warnings Interface

Table 13. Reaction -> User/System Administrator definition (SAWI)

Security Alerts and Warnings Interface (SAWI)			
Description	This interface transfers the alerts and warnings from the Reaction Module to the end-user interfaces. It is the main communication channel between the Reaction Module and the ANASTACIA User Plane.		
Component providing the interface	Security Alert Service (Reaction Module)		
Consumer components	End-user interface		
Input data / Output Data	Database directly accessed from the user plane interfaces		
State	Asynchronous		
Input data / Output Data	**Methods or endpoints** of the interface	**Parameters** of the method	**Return Values** of the method
	Specific tables on the database containing the information	The set of detected security issues (for raiseAlert), and the applied countermeasures (for informReaction)	None
Constraints	None		
Responsibilities	○ ATOS ○ MONT ○ CNR ○ UTRC		

2.2.4.3 Mitigation Recommendations Interface

Table 14. Verdict and Decision Support System -> Mitigation Action Service (V2MI)

Mitigation Recommendations Interface (V2MI)

Description	This interface provides a list of mitigation recommendations capable of mitigating an ongoing incident. For every mitigation a suitability score is also provided. It is used by the MAS to trigger the mitigation suggested by the VDSS.		
Component providing the interface	VDSS		
Consumer components	MAS		
Type of Interface	RabbitMQ queue		
State	Asynchronous		
Input data / Output Data	**Methods or endpoints** of the interface	**Parameters** of the method	**Return Values** of the method
	Exchange queue "exchange.recommendations" attached to a queue "queue.recommendations.mas" used by the MAS to consume messages	JSON, which includes: - The complete alarm to mitigate - Name of the incident - List of mitigations, including: o Id of the mitigation o Suitability score of the mitigation o Whether the mitigation has been chosen by the Chief Information Security Officer (CISO)	None
Constraints	None		
Responsibilities	o MONT o ATOS		

2.2.4.4 Countermeasures Suggestions Interface

Table 15. Reaction -> Orchestrator definition (CSI)

Mitigations Enforcement Interface (CSI)			
Description	This interface allows the Mitigation Action Service to send the computed countermeasures to the Security Orchestrator for its enforcement. The countermeasures are sent in the MSPL format, using an HTTP interface.		
Component providing the interface	Security Orchestrator		
Consumer components	Mitigation Action Service (Reaction Module)		
Type of Interface	HTTP REST interface		
State	Asynchronous		
Input data / Output Data	**Methods or endpoints** of the interface	**Parameters** of the method	**Return Values** of the method
	HTTP endpoint for sending the MSPL countermeasure	An MSPL-compliant XML file that codifies the computed countermeasures.	ID
Constraints	Following the usage of open standards, this interface uses both the HTTP protocol as the main wrapper for the countermeasures. In addition, the mitigation actions are expressed in the MSPL language,		
Responsibilities	○ AALTO ○ MONT		

2.2.4.5 System Model Interface

Table 16. Verdict and Decision Support System (VDSS)<-> System Models Service (SMI)

System Models Interface (SMI)	
Description	This interface provides the ANASTACIA components with the information related with the System Model and the Security Enablers deployed in the network. This information includes the description of the devices deployed, their endpoints, among others.

Component providing the interface	Security Orchestrator		
Consumer components	Mitigation Action Service (Reaction Module) Assets Model (Reaction Module) Verdicts and Decision Support System (Reaction Module) Policy Interpreter Policy Conflict Detector IoT Controller (IoT Register)		
Type of Interface	HTTP Interface (Swagger-based)		
State	Synchronous		
Input data / Output Data	**Methods or endpoints** of the interface	**Parameters** of the method	**Return Values** of the method
	HTTP GET/POST/DELETE for retrieving or updating the information.	JSON-based object that could specify different needed parameters.	A JSON-based object with the requested information.
Constraints	None		
Responsibilities	○ AALTO		

2.2.4.6 Asset Model Interface

Table 17. VDSS -> Assets Model (AMI)

Asset Model Interface (AMI)	
Description	Asset Model component is responsible for storing and sharing latest information about mitigation strategies, capabilities and actions. This interface is used by components to access this information.
Component providing the interface	Assets Model
Consumer components	Verdict and Decision Support System

Type of Interface	REST API		
State	State is persevered at Assets Model level. This helps ANASTACIA framework keep latest information about strategies, capabilities and actions available in the framework.		
Input data / Output Data	**Methods or endpoints** of the interface	**Parameters** of the method	**Return Values** of the method
	<host_ip:port>/strategies/ [GET]	None	List of all strategies stored by AM component
	<host_ip:port>/strategies/<int:id> [GET, PUT]	Strategy ID or new strategy	One strategy with unique ID (GET) or adds new strategy (PUT)
	<host_ip:port>/strategies?available =<int:availability> [GET]	Availability flag	List of strategies with flag enabled (1) or disabled (0)
	<host_ip:port>/strategies/threat /<int:id> [GET]	Threat identifier	List of strategies with given threat ID or
	<host_ip:port>/strategies/threat/ <int:id>?available=<int:availability> [GET]	Threat identifier and availability flag	List of strategies with availability flag enabled (1) or disabled (0)
	<host_ip:port>/capabilities/ [GET]	None	List of all capabilities
	<host_ip:port>/capabilities /<int:id> [GET, PUT]	Capability identifier	One capability with unique ID (GET) or adds new capability (PUT)
	<host_ip:port>/capabilities? available=<int:availability> [GET]	Capability identifier and availability flag	List of capabilities with availability flag enabled (1) or disabled (0)
	<host_ip:port>/actions/ [GET]	None	List of all actions
	<host_ip:port>/actions/<int:id> [GET]	Action identifier	One action with unique ID
Constraints	N/A		
Responsibilities	○ THALES with UTRC support		

2.2.5 Interfaces for Seal Creation

2.2.5.1 Seal Manager Metadata Interface

The following interface SMMI (Table 18) is used for the exchange of the relevant data that the seal manager needs in order to create the Dynamic Security and Privacy Seal.

Table 18. Reaction -> Seal Manager definition (SMMI)

Seal Manager Metadata Interface (SMMI)			
Description	The interface is used to provide the alerts, mitigation actions and mitigation states to forge the dynamic security and privacy seal. Security and the privacy in a real-time fashion.		
Component providing the interface	Dynamic Security and Privacy Seal		
Consumer components	• Security Alert Service • DSPS agent		
Type of Interface	REST. Consumes json, following STIX/TAXII standard (Structured Threat Information Expression and Trusted Automated eXchange of Indicator Information).		
State	Synchronous		
Input data / Output Data	**Methods or endpoints** of the interface	**Parameters** of the method	**Return Values** of the method

	[POST]
	/api/v3/system_status
		STIX json object.
		It should include threat information, and may include mitigation information.
		e.g.:
		{
		 "name": "Man in the Middle on IoT data",
		 "type": "attack-pattern",
		 "id": "attack-pattern--8cc017e2-1df3-4163-ada2-b33d5faead19",
		 "created": "2019-05-20T13:52:34.748Z",
		 "modified": "2019-05-20T13:52:34.748Z",
		 "spec_version": "2.0",
		 "x_dsps_anastacia_alert": {
		  "request_id": "12345",
		  "status_complete": true,
		  "policy": null,
		  "AlarmEvent": {
		   "PRIORITY": 5,
		   "RELIABILITY": 10,
		   "ORGANIZATION": "ATOS",
		   "SID_NAME": "Man in the Middle on IoT data 2",
		   "USERDATA2": "Probing attack",
		   "RISK": 10,
		   "CATEGORY": "Attack",
		   "SUBCATEGORY": "compromise"
		  }
		 }
		}
			None

Constraints	N/A		

Responsibilities	○ MAND ○ CNR

2.3 Technical Integration Mechanisms And Process

For the technical integration in ANASTACIA we need many different components to be deployed and communicate using dedicated interfaces or through message broker usage. In the following table, we provide an overview of the mechanisms selected by the consortium to help the integration process. The reader can find more detailed information about the mechanisms in deliverable D6.1[1] .

Table 19. ANASTACIA integration mechanisms

Multiple Facets Integration	
At Deployment Level	**Using Docker whenever possible and configure components' deployment using ENV variables and Docker compose** **Dedicated container registries using Gitlab**
At Interfaces Level	**Documentation of Interfaces using Swagger**
At Code Level	**Dedicated code repositories using Gitlab**
At Knowledge Level	**Dedicated folder for collaboration on the shared repository of consortium**

2.3.1 Using Docker for Integration

Although providing components as Docker images and collection of components as Docker compose files was not a hard requirement it was a practice that was followed at the extend this was possible in order to ease the testing and deployment of ANASTACIA. Dockerizing an application is a very diverse procedure and multiple approaches can be followed, for this reason dedicated guidelines and examples have been shared among the technical partners. For hosting the needed docker images[2] Gitlab has been used.

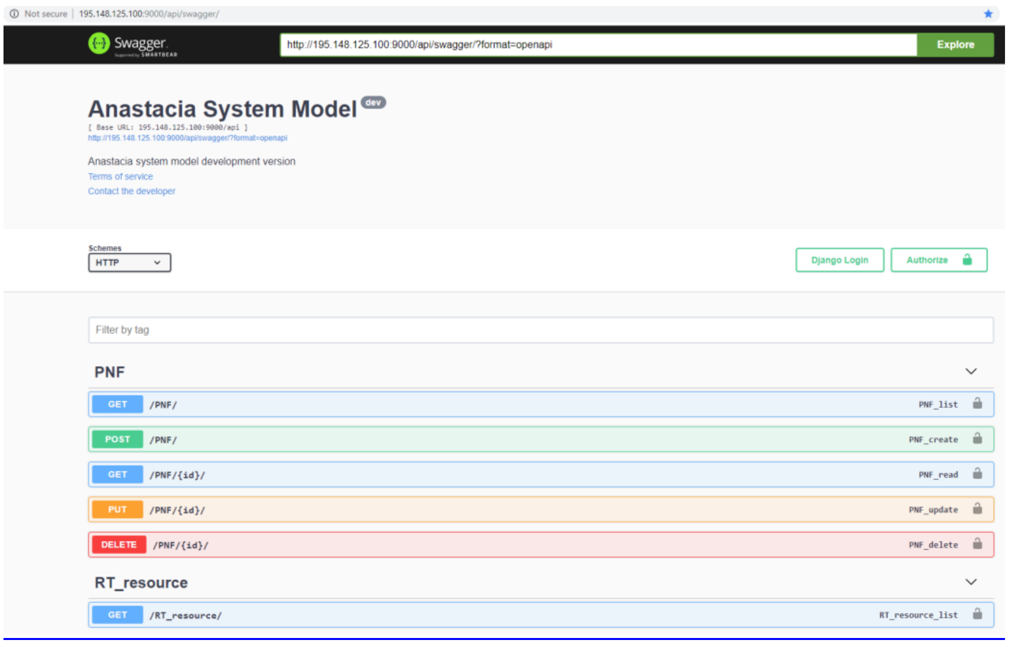
2.3.2 Integration at Interface Level

For the better coordination of the development and integration through the interfaces we suggested the usage of Swagger[3]. Also, we have used message queues for providing support for Asynchronous operations.

2.3.2.1 Swagger Usage

At some of the components, especially ones with extended APIs, Swagger has been used in order to ease the integration and testing. In Figure 4 below a screenshot of the Anastacia System Model component is provided.

[2] registry.gitlab.com/anastacia-project/framework
[3] https://swagger.io/



Not secure | 195.148.125.100:9000/api/swagger/

http://195.148.125.100:9000/api/swagger/?format=openapi

Anastacia System Model dev

```
[ Base URL: 195.148.125.100:9000/api ]
```

http://195.148.125.100:9000/api/swagger/?format=openapi

Anastacia system model development version

Terms of service

Contact the developer

Schemes: HTTP

PNF

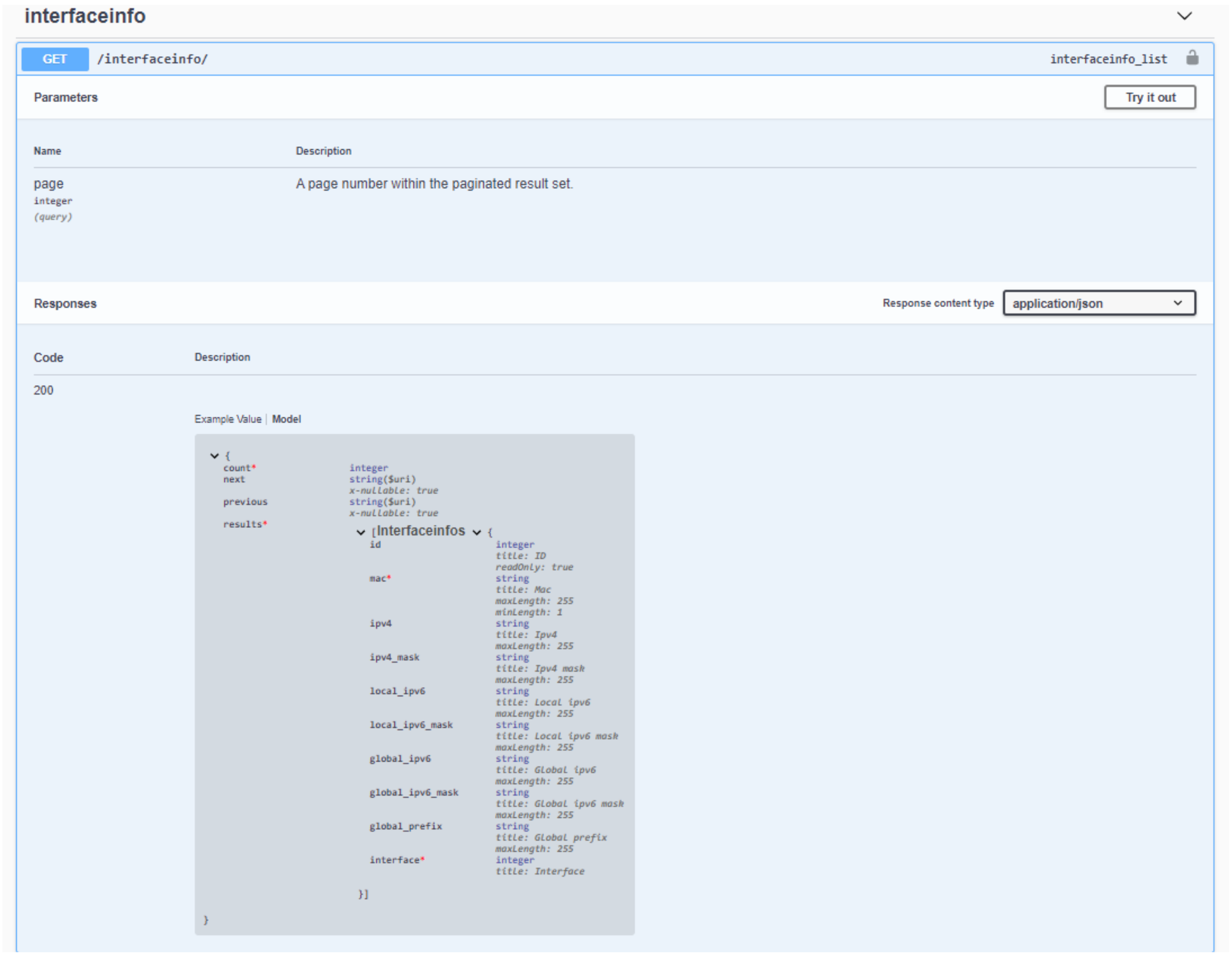
```
GET     /PNF/          PNF_list
POST    /PNF/          PNF_create
GET     /PNF/{id}/     PNF_read
PUT     /PNF/{id}/     PNF_update
DELETE  /PNF/{id}/     PNF_delete
```

RT_resource

```
GET     /RT_resource/  RT_resource_list
```

Figure 4. Swagger usage in ANASTACIA System Model

The usage of Swagger allowed having always up-to-date information not only about the available functions but also about the way that these functions should be called.



interfaceinfo

```
GET  /interfaceinfo/     interfaceinfo_list
```

Parameters

Name	Description
page integer (query)	A page number within the paginated result set.

Responses — Response content type: application/json

Code	Description
200	Example Value | Model

```
{
  count*                    integer
  next                      string($uri)
                            x-nullable: true
  previous                  string($uri)
                            x-nullable: true
  results*                  [Interfaceinfos {
      id                        integer
                                title: ID
                                readOnly: true
      mac*                      string
                                title: Mac
                                maxLength: 255
                                minLength: 1
      ipv4                      string
                                title: Ipv4
                                maxLength: 255
      ipv4_mask                 string
                                title: Ipv4 mask
                                maxLength: 255
      local_ipv6                string
                                title: Local ipv6
                                maxLength: 255
      local_ipv6_mask           string
                                title: Local ipv6 mask
                                maxLength: 255
      global_ipv6               string
                                title: Global ipv6
                                maxLength: 255
      global_ipv6_mask          string
                                title: Global ipv6 mask
                                maxLength: 255
      global_prefix             string
                                title: Global prefix
                                maxLength: 255
      interface*                integer
                                title: Interface
  }]
}
```

Figure 5. Swagger usage in ANASTACIA System Model

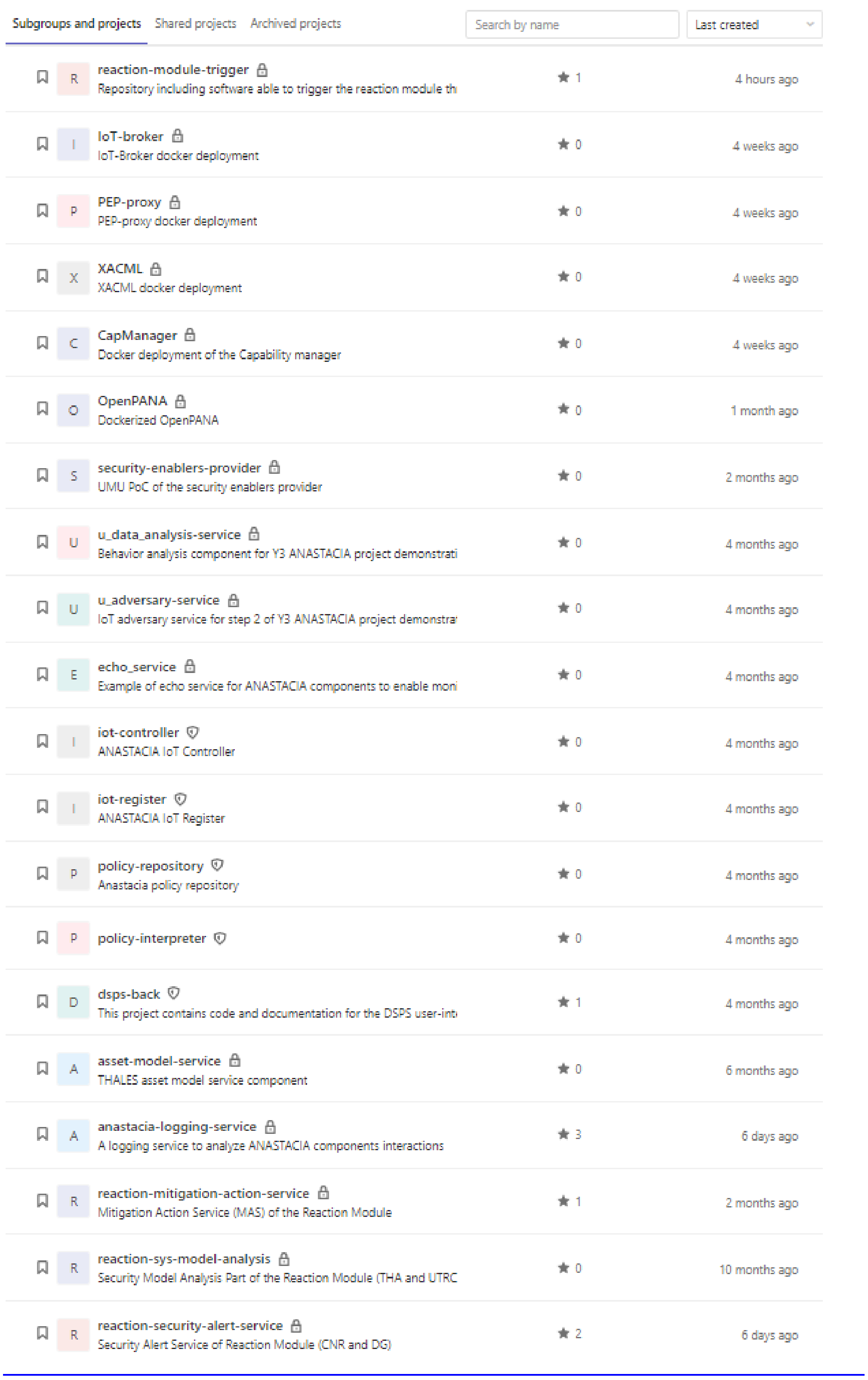
2.3.2.2 Asynchronous Operations

For interfaces that needed asynchronous operation mode for their communication, message brokers have been used. The publish/subscribe (pub/sub) messaging pattern is realized using destinations known as topics. Publishers send messages to the topic and subscribers register to receive messages from the topic. Any messages sent to the topic are automatically delivered to all subscribers. In ANASTACIA we are using Apache Kafka[4] as message broker for the integration of monitoring agents' feedback, and by other components needing asynchronous communication, such as the Logging Server. RabbitMQ that is part of the ATOS's VDSS component has also been used.

2.3.3 Code Level Integration - Working on the same components

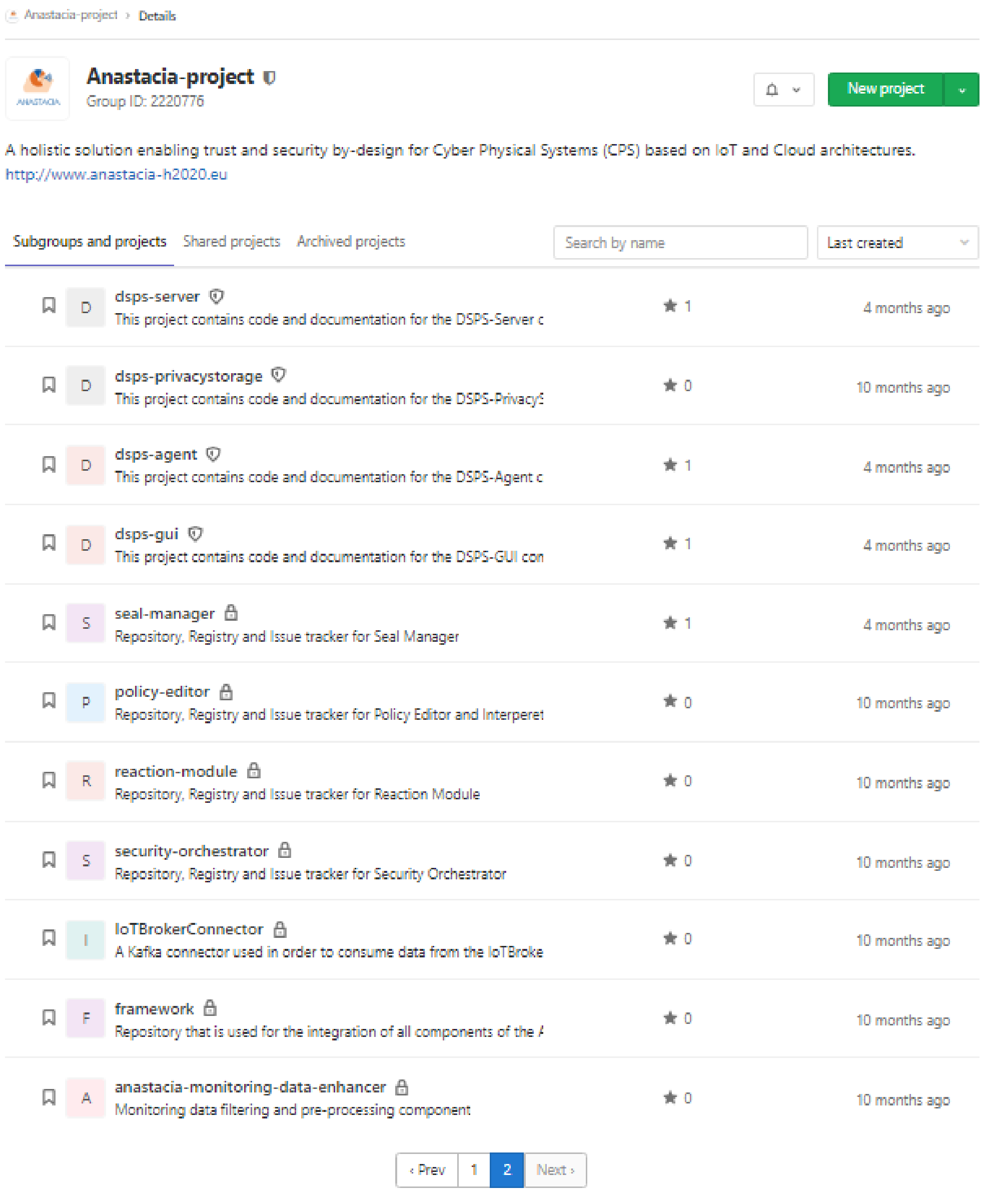
In order to support the cases that multiple partners need to work on the same components, code level integration was used. Code repositories were used by all partners not only to work together but also to store their component's code safety. The source code repositories are available at: https://gitlab.com/ANASTACIA-project. For the actual components of the platform 22 different repositories were created. One additional repo/project was created for hosting common files for the Anastacia framework creation; 8 additional repositories have been also added for storing and sharing additional tools, attacker emulators etc.

[4] https://kafka.apache.org/



R	reaction-module-trigger	Repository including software able to trigger the reaction module th	★ 1	4 hours ago
I	IoT-broker	IoT-Broker docker deployment	★ 0	4 weeks ago
P	PEP-proxy	PEP-proxy docker deployment	★ 0	4 weeks ago
X	XACML	XACML docker deployment	★ 0	4 weeks ago
C	CapManager	Docker deployment of the Capability manager	★ 0	4 weeks ago
O	OpenPANA	Dockerized OpenPANA	★ 0	1 month ago
S	security-enablers-provider	UMU PoC of the security enablers provider	★ 0	2 months ago
U	u_data_analysis-service	Behavior analysis component for Y3 ANASTACIA project demonstrati	★ 0	4 months ago
U	u_adversary-service	IoT adversary service for step 2 of Y3 ANASTACIA project demonstra	★ 0	4 months ago
E	echo_service	Example of echo service for ANASTACIA components to enable moni	★ 0	4 months ago
I	iot-controller	ANASTACIA IoT Controller	★ 0	4 months ago
I	iot-register	ANASTACIA IoT Register	★ 0	4 months ago
P	policy-repository	Anastacia policy repository	★ 0	4 months ago
P	policy-interpreter		★ 0	4 months ago
D	dsps-back	This project contains code and documentation for the DSPS user-int	★ 1	4 months ago
A	asset-model-service	THALES asset model service component	★ 0	6 months ago
A	anastacia-logging-service	A logging service to analyze ANASTACIA components interactions	★ 3	6 days ago
R	reaction-mitigation-action-service	Mitigation Action Service (MAS) of the Reaction Module	★ 1	2 months ago
R	reaction-sys-model-analysis	Security Model Analysis Part of the Reaction Module (THA and UTRC	★ 0	10 months ago
R	reaction-security-alert-service	Security Alert Service of Reaction Module (CNR and DG)	★ 2	6 days ago

Figure 6. ANASTACIA project group in GitLab (1/2)



Anastacia-project › Details

Anastacia-project
Group ID: 2220776

New project

A holistic solution enabling trust and security by-design for Cyber Physical Systems (CPS) based on IoT and Cloud architectures.
http://www.anastacia-h2020.eu

Subgroups and projects | Shared projects | Archived projects | Search by name | Last created

D	dsps-server This project contains code and documentation for the DSPS-Server c	★ 1	4 months ago
D	dsps-privacystorage This project contains code and documentation for the DSPS-PrivacyS	★ 0	10 months ago
D	dsps-agent This project contains code and documentation for the DSPS-Agent c	★ 1	4 months ago
D	dsps-gui This project contains code and documentation for the DSPS-GUI con	★ 1	4 months ago
S	seal-manager Repository, Registry and Issue tracker for Seal Manager	★ 1	4 months ago
P	policy-editor Repository, Registry and Issue tracker for Policy Editor and Interperet	★ 0	10 months ago
R	reaction-module Repository, Registry and Issue tracker for Reaction Module	★ 0	10 months ago
S	security-orchestrator Repository, Registry and Issue tracker for Security Orchestrator	★ 0	10 months ago
I	IoTBrokerConnector A Kafka connector used in order to consume data from the IoTBroke	★ 0	10 months ago
F	framework Repository that is used for the integration of all components of the A	★ 0	10 months ago
A	anastacia-monitoring-data-enhancer Monitoring data filtering and pre-processing component	★ 0	10 months ago

‹ Prev 1 2 Next ›

Figure 7. ANASTACIA project group in GitLab (2/2)

3 Platform Implementation and Integration Planning

In ANASTACIA have been performing two cycles of technical and user evaluations during the project period, while the development of the platform and its evaluation was be performed in parallel. The final version of ANASTACIA Framework will be delivered at the end of the project, as part of milestone “MS32 - Second iterative cycle of development completed and validated”. This version is the fully integrated framework that has undergone technical validation, and in Table 20 we provide the actual status of the components towards this final release. As seen all components have been finalized; minor changes are only performed based on the final setup and tests on the demonstration environment for the final review demo.

Table 20. ANASTACIA components’ status for final release (M36)

Component	Responsible Partner (subcomponent)	Status for Final Release
Policy Editor Tool	UMU	Final implementation covering the main identified use cases, allowing to model HSPL-OP policies through the GUI as well as request their translation and enforcement.
Interpreter	UMU	Final implementation comprising the H2M refinement and M2L translation processes for the main identified use cases.
Policy Conflict Detector	UMU	Final implementation comprising the MSPL-OP conflict and dependencies detection for the main identified use cases.
Policy Repository	UMU	Final implementation comprising the storage of HSPL-OP, MSPL-OP and Low level configuration instances as well as their status.
Security Enablers Provider	UMU	Final implementation comprising the plugins catalogue as well as the plugin files.
Security Enablers Plugins	OdinS (DTLS proxy)	Final integration of DTLS proxy to enable the dynamic deployment by NVF orchestrator.
	OdinS (AAA Architecture)	Final integration of all AAA security mechanisms to enable the dynamic deployment by NVF orchestrator.
	OdinS (Network Authenticator)	Final integration of Network Authenticator to enable the dynamic deployment by NVF orchestrator.
	UMU (IPTABLES)	Final development of filtering capabilities using the IPTABLES enabler in order to cope with the main identified use cases.

	UMU (IoT Controller)	Final implementation of power management, bootstrapping and IoT honeynet capabilities, as well as IoT devices registration.
	UMU (SDN ONOS NB & SDN ODL NB)	Final implementation of required networking capabilities by the use cases.
	UMU (Cooja)	Final implementation of IoT honeynet translation and deployment as VNF, from different IoT architecture topologies.
	UBITECH (Kippo)	Final implementation of the ssh honeynet enabler.
Security orchestrator Engine	AALTO	Final implementation of the Security Orchestrator as a virtual instance covering the defined use cases.
Security Orchestrator Optimizer	AALTO	Final development of optimization Algorithm that gives the possibility to ensure an efficient life cycle management of different VNFs.
IoT Controllers and Drivers	OdinS / UMU	Final development and integration with the Orchestrator component for enabling all security actuations over IoT network. Final implementation and integration of the IoT controller northbound/southbound endpoints in order to cope with the main identified use cases.
NVF Orchestrators and Drivers	AALTO	Final version of Open Source Mano with the relevant VNF and NS descriptors and the final version of the OSM driver.
SDN Controllers and Drivers	AALTO	Final version of the ONOS controller with the relevant underlying architecture and the final version of the ONOS driver.
Monitoring Agents	OdinS	Final development and integration of all attacks notifications provided by IoT broker and AAA architecture when detect malicious behaviours according to use cases proposed in ANASTACIA project.
	MONT	Integration of the adapted version of the MMT-Probe to support all the IoT protocols involved in the ANASTACIA use cases.

	ATOS	Implementation of new plugins for the incorporation of new sources of events, including new virtual security services and 5G related incidents.
Data Filtering and pre-processing Component	UBITECH	Finalization of appropriate models used for filtering and pre-processing of data, taking into account 5g related incidents.
Data Analysis Component	UTRC	Final implementation of adversary emulator for APT that generates MiTM attacks in Y3 scenario.
Verdict and Decision Support System	ATOS	Creation of a Decision Support System based on alerts detected, devices affected and available mitigations to provide with suitability scores for the available mitigations, which are used to recommend about mitigations. Integration of the VDSS with the Assets Model and with the System Model of the Orchestrator.
Mitigation Action Service	MONT	Implementation of the standard language (MSPL) to communicate the set of computed countermeasures to the Security Orchestrator. Integration with the other components of the ANASTACIA platform.
	ATOS	Integration of the VDSS output with the MAS for submitting mitigations recommendations.
Security Alert Service	ATOS, CNR	Incorporation of alerts, configuration of the decision support service, visualization of reports, interfaces for the configuration of monitoring, alerting and reaction components.
	ATOS	Integration of the VDSS and Incident Detector output for submitting information about incidents detected and mitigations recommendations.
Dynamic Security and Privacy Seal Agent	MAND, DG, AS	Dynamic Security and Privacy Seal implementation completed.
Security and Privacy Manager Analysis	MAND, DG, AS	Implementation completed and integrated with other components.
Dynamic Security and Privacy Seal User Interface	MAND, DG, AS	GUI of Dynamic Security and Privacy Seal implementation completed.
System Model Service	AALTO	Final version of System Model Service that confirms with open API specification.

Assets Model	THALES	Component development and integration completed.
Security Resource Planning	THALES	Component development and integration completed.
Incident Detector	ATOS, MONT	Incorporation of cross correlated alarms that integrate different events from different sources for the generation of more accurate alarms. Incorporation of support for additional alerts related to 5G incidents. Incorporation of IoT device related information (such as location, type of device) to provide with risk calculations based on the characteristics of the device affected.
Security Sensors	MONT (MMT-Probe)	Inclusion of the detection algorithms for untrusted communications and SlowDoS attacks.

4 Platform Deployment

The deployment of ANASTACIA as platform is currently executed the premises of UMU where the actual demonstrator of the project takes place. Prior to this, for development and demonstration purposes a test installation of the core platform components has been provided where components were hosted in various locations on the premises of technical partners. In the table below we provide the basic requirements of the components, in terms of memory, storage, processing needs, and the OS used.

Table 21. ANASTACIA components' requirements

Component(s)	Memory	Storage	Processor	OS
Policy Editor Tool	1GB	1GB	1vCPU	Ubuntu
Policy Interpreter	1GB	1GB	1vCPU	Ubuntu
Policy Conflict Detector	1GB	1GB	1vCPU	Ubuntu
Policy Repository	1GB	1GB	1vCPU	Ubuntu
Security Enablers Provider	1GB	1GB	1vCPU	Ubuntu
Security orchestrator Engine	2GB	10GB	1vCPU	Ubuntu
Security Orchestrator Optimizer	2GB	10GB	1vCPU	Ubuntu
Data Filtering and pre-processing Component	16Gb	100Gb	4vCPUs	Ubuntu
Data Analysis Component	1Gb	1GB	1vCPU	Ubuntu
Verdict and Decision Support System	2Gb	10Gb	2vCPU	Ubuntu
Mitigation Action Service	2Gb	20Gb	1vCPU	Ubuntu
Security Alert Service	2Gb	20Gb	1vCPU	Ubuntu
Dynamic Security and Privacy Seal Agent	4Gb	50Gb	4VCPU	Ubuntu
Security and Privacy Manager Analysis*	4Gb	50Gb	8VPCU	Ubuntu
Dynamic Security and Privacy Seal User Interface	8GB	50GB	8vCPU	Ubuntu

System Model Service	2GB	10GB	1vCPU	Ubuntu
Assets Model	100MB	100MB	1vCPU	Ubuntu
Incident Detector	16GB	80GB	8vCPU	Ubuntu

*10 nodes required.

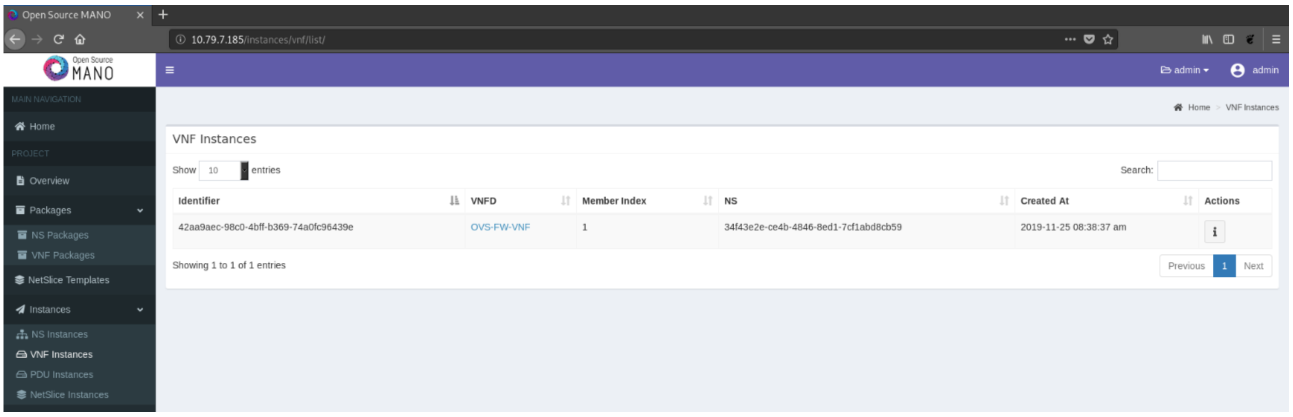
The final setup of the platform is performed in the demonstration environment in order to execute the tests that are documented in D6.5[7] , as outcome of task T6.2. In addition to the components of the platform the demonstration environment includes all the needed infrastructure as IoT devices and IoT broker, PANA agents and also simulated attackers.

4.1 Deployed Components Overview

This chapter provides information about the most important ANASTACIA modules that have been deployed in UMU premises. As Anastacia is an integrated framework consisting of many different components, this section also aims to assist on the understanding of the platform.

4.1.1 Security Orchestrator & Supporting Components

Security Orchestrator is the central part of ANASTACIA, as it executes and orchestrates all actions instructed by the other components. Figure 8 is a screenshot from OSM[5] interface in the VNF instances section showing the running OVS-FW VNF instantiated by the Security Orchestrator. Figure 9 shows the OVS-FW instance on OpenStack and its IPs and ONOS network represented in Figure 10.



VNF Instances

Show 10 entries Search:

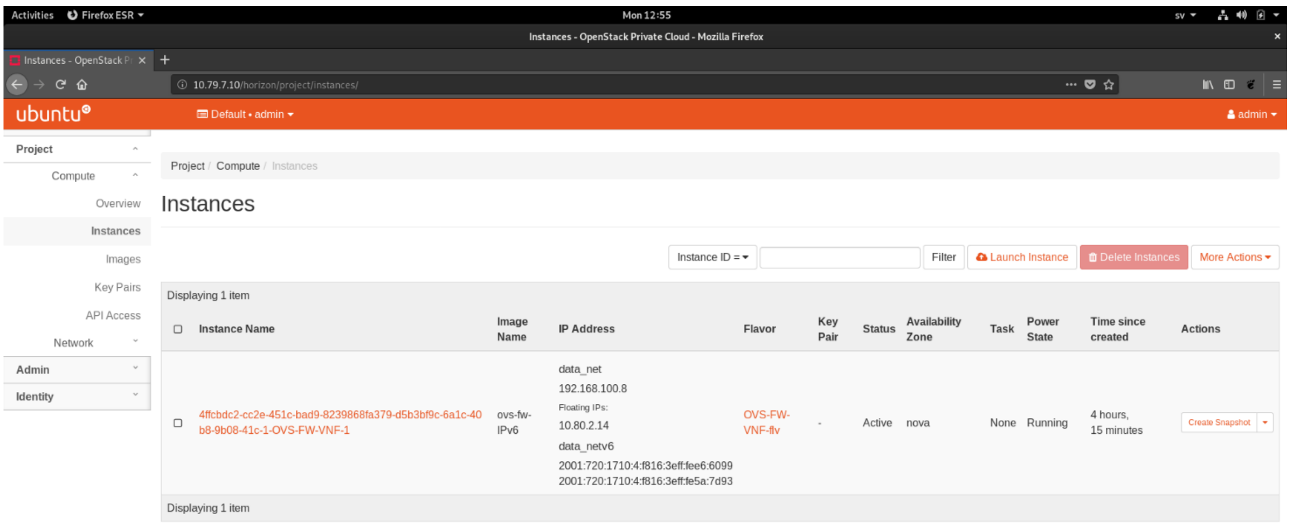
Identifier	VNFD	Member Index	NS	Created At	Actions
42aa9aec-98c0-4bff-b369-74a0fc96439e	OVS-FW-VNF	1	34f43e2e-ce4b-4846-8ed1-7cf1abd8cb59	2019-11-25 08:38:37 am	i

Showing 1 to 1 of 1 entries

Previous 1 Next

Figure 8. Security Orchestrator Instantiates OVS-FW on OSM

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

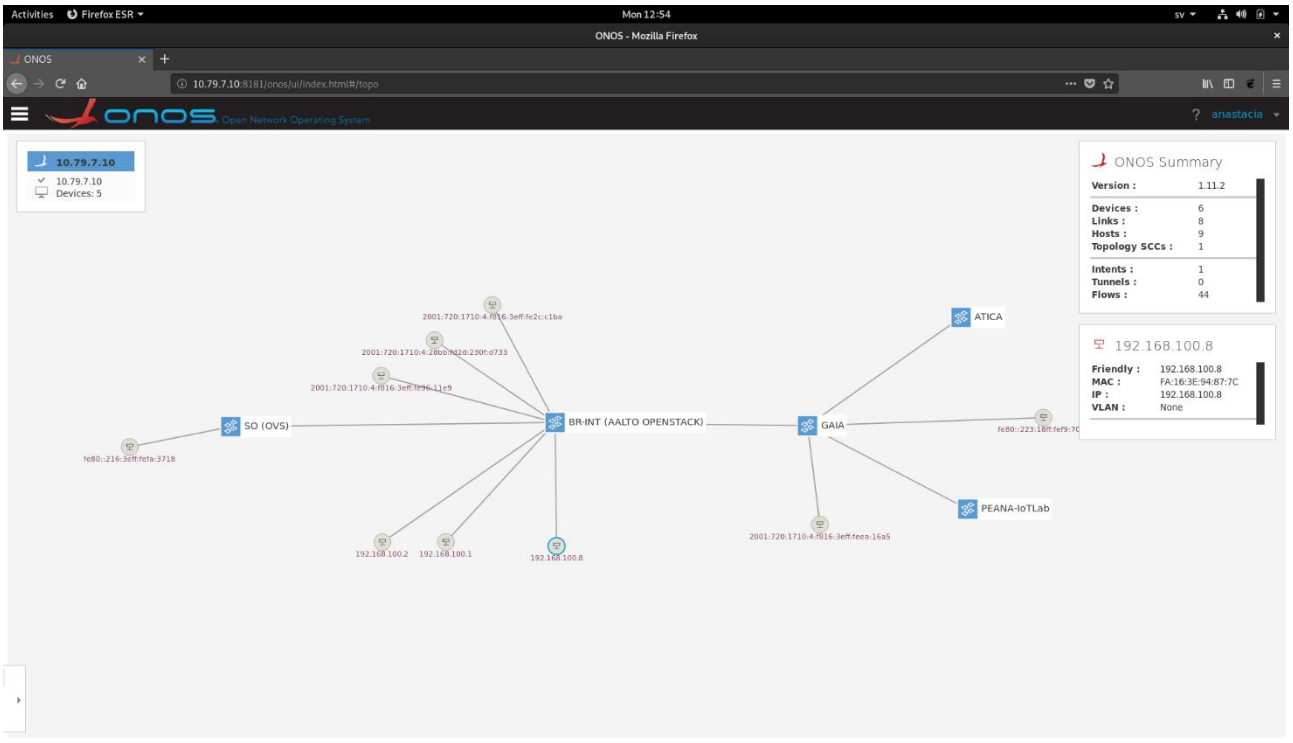

Instances

Displaying 1 item

	Instance Name	Image Name	IP Address	Flavor	Key Pair	Status	Availability Zone	Task	Power State	Time since created	Actions
☐	4ffcbdc2-cc2e-451c-bad9-8239868fa379-d5b3bf9c-6a1c-40b8-9b08-41c-1-OVS-FW-VNF-1	ovs-fw-IPv6	data_net 192.168.100.8 Floating IPs: 10.80.2.14 data_netv6 2001:720:1710:4:f816:3eff:fee6:6099 2001:720:1710:4:f816:3eff:fe5a:7d93	OVS-FW-VNF-flv	-	Active	nova	None	Running	4 hours, 15 minutes	Create Snapshot

Displaying 1 item

Figure 9. OVS-FW instance on OpenStack with management and floating IPs and IPV6 interfaces



ONOS Summary

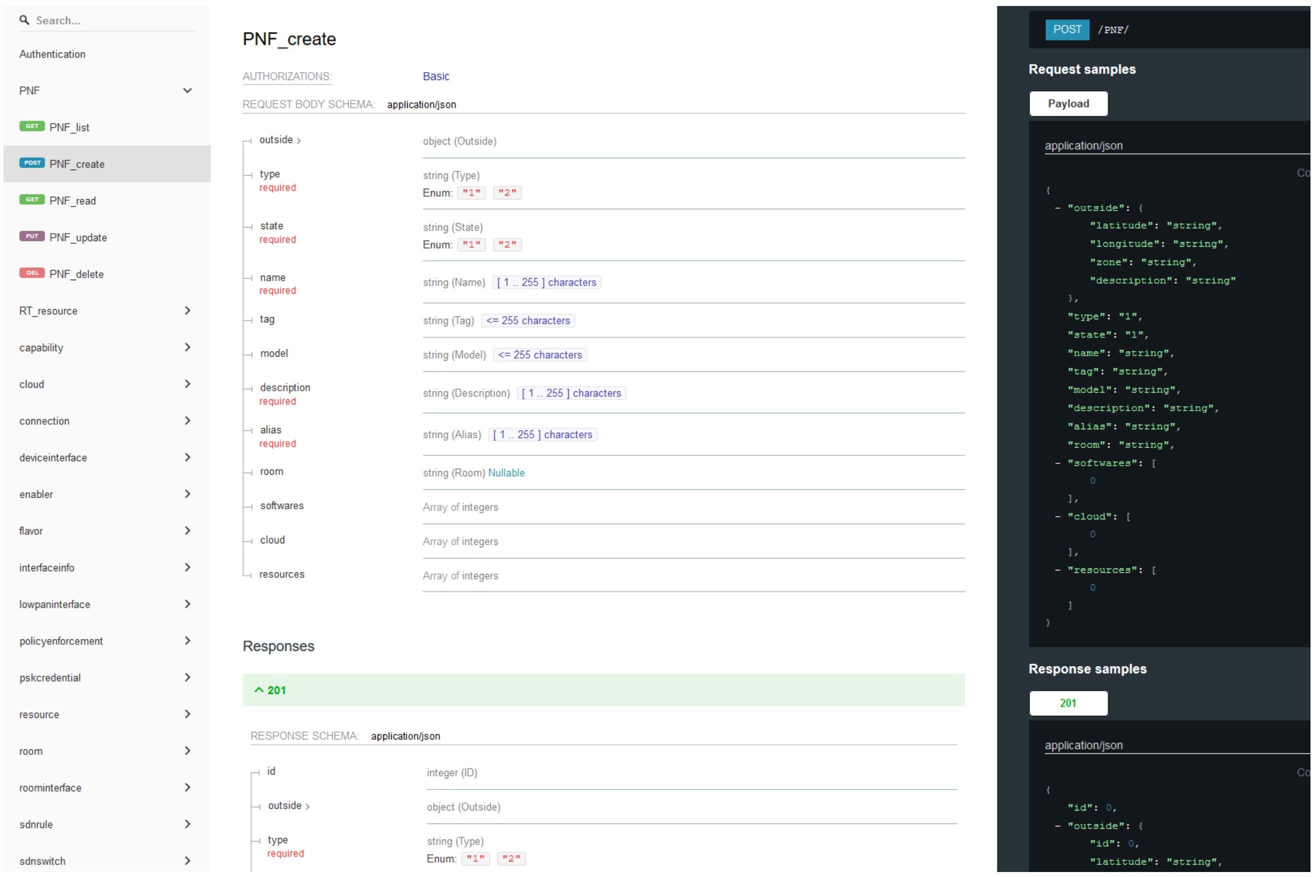
Version :	1.11.2
Devices :	6
Links :	8
Hosts :	9
Topology SCCs :	1
Intents :	1
Tunnels :	0
Flows :	44

192.168.100.8

Friendly :	192.168.100.8
MAC :	FA:16:3E:94:87:7C
IP :	192.168.100.8
VLAN :	None

Figure 10. OVS-FW on ONOS GUI

One of the main supporting components of the Security Orchestrator and one of the components used by many other components of the platform is the System Model Service. The API of System Model Service is depicted below.



PNF_create

AUTHORIZATIONS: Basic

REQUEST BODY SCHEMA: application/json

outside ›	object (Outside)
type required	string (Type) Enum: "1" "2"
state required	string (State) Enum: "1" "2"
name required	string (Name) [1 .. 255] characters
tag	string (Tag) <= 255 characters
model	string (Model) <= 255 characters
description required	string (Description) [1 .. 255] characters
alias required	string (Alias) [1 .. 255] characters
room	string (Room) Nullable
softwares	Array of integers
cloud	Array of integers
resources	Array of integers

Responses

201

RESPONSE SCHEMA: application/json

id	integer (ID)
outside ›	object (Outside)
type required	string (Type) Enum: "1" "2"

POST /PNF/

Request samples

Payload

application/json

```
{
 - "outside": {
       "latitude": "string",
       "longitude": "string",
       "zone": "string",
       "description": "string"
   },
   "type": "1",
   "state": "1",
   "name": "string",
   "tag": "string",
   "model": "string",
   "description": "string",
   "alias": "string",
   "room": "string",
 - "softwares": [
       0
   ],
 - "cloud": [
       0
   ],
 - "resources": [
       0
   ]
}
```

Response samples

201

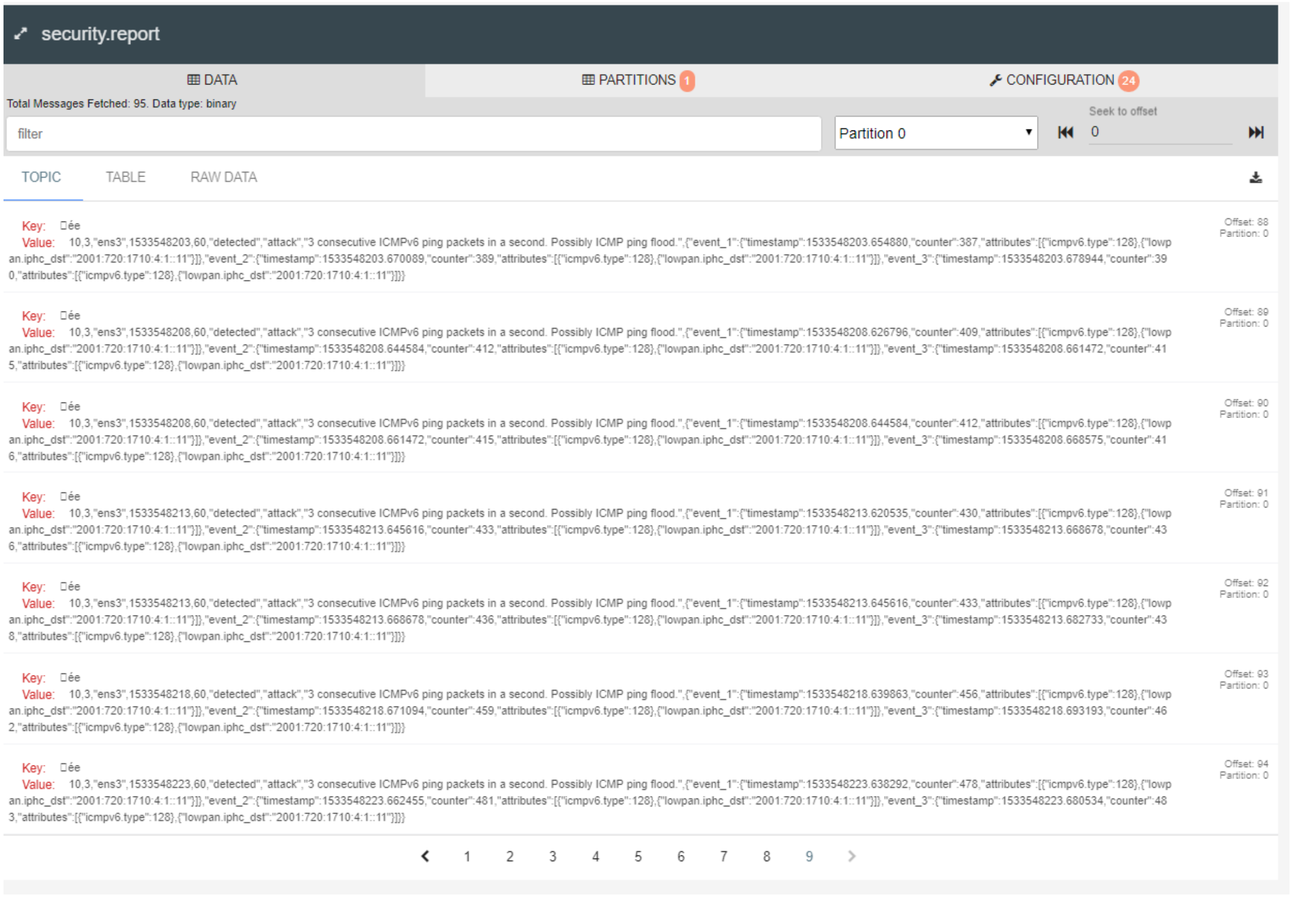
application/json

```
{
   "id": 0,
 - "outside": {
       "id": 0,
       "latitude": "string",
```

Figure 11. System model swagger documentation

4.1.2 Data Filtering and Pre-processing Component / Kafka Broker

Another component that is used by other ANASTACIA components is the Kafka broker that is part of the Data Filtering and pre-processing Component and is depicted below Figure 12.



security.report

DATA | PARTITIONS 1 | CONFIGURATION 24

Total Messages Fetched: 95. Data type: binary

filter | Partition 0 | Seek to offset 0

TOPIC | TABLE | RAW DATA

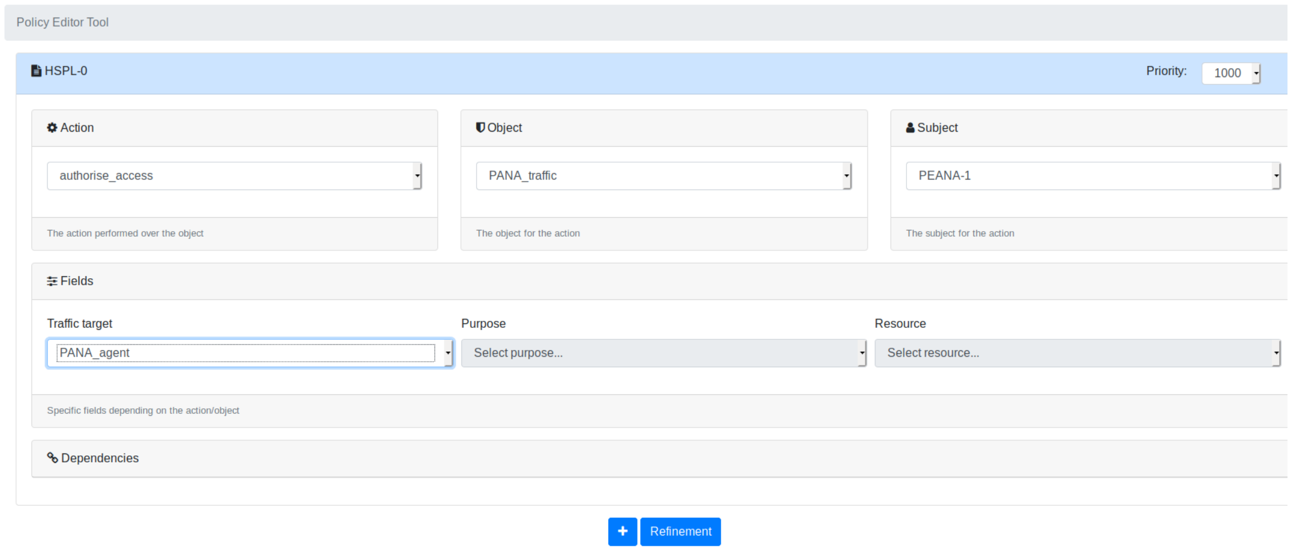
```
Key:   □ée                                                                                                                                                                                          Offset: 88
Value:   10,3,"ens3",1533548203,60,"detected","attack","3 consecutive ICMPv6 ping packets in a second. Possibly ICMP ping flood.",{"event_1":{"timestamp":1533548203.654880,"counter":387,"attributes":[{"icmpv6.type":128},{"lowp    Partition: 0
an.iphc_dst":"2001:720:1710:4:1::11"}]},"event_2":{"timestamp":1533548203.670089,"counter":389,"attributes":[{"icmpv6.type":128},{"lowpan.iphc_dst":"2001:720:1710:4:1::11"}]},"event_3":{"timestamp":1533548203.678944,"counter":39
0,"attributes":[{"icmpv6.type":128},{"lowpan.iphc_dst":"2001:720:1710:4:1::11"}]}}

Key:   □ée                                                                                                                                                                                          Offset: 89
Value:   10,3,"ens3",1533548208,60,"detected","attack","3 consecutive ICMPv6 ping packets in a second. Possibly ICMP ping flood.",{"event_1":{"timestamp":1533548208.626796,"counter":409,"attributes":[{"icmpv6.type":128},{"lowp    Partition: 0
an.iphc_dst":"2001:720:1710:4:1::11"}]},"event_2":{"timestamp":1533548208.644584,"counter":412,"attributes":[{"icmpv6.type":128},{"lowpan.iphc_dst":"2001:720:1710:4:1::11"}]},"event_3":{"timestamp":1533548208.661472,"counter":41
5,"attributes":[{"icmpv6.type":128},{"lowpan.iphc_dst":"2001:720:1710:4:1::11"}]}}

Key:   □ée                                                                                                                                                                                          Offset: 90
Value:   10,3,"ens3",1533548208,60,"detected","attack","3 consecutive ICMPv6 ping packets in a second. Possibly ICMP ping flood.",{"event_1":{"timestamp":1533548208.644584,"counter":412,"attributes":[{"icmpv6.type":128},{"lowp    Partition: 0
an.iphc_dst":"2001:720:1710:4:1::11"}]},"event_2":{"timestamp":1533548208.661472,"counter":415,"attributes":[{"icmpv6.type":128},{"lowpan.iphc_dst":"2001:720:1710:4:1::11"}]},"event_3":{"timestamp":1533548208.668575,"counter":41
6,"attributes":[{"icmpv6.type":128},{"lowpan.iphc_dst":"2001:720:1710:4:1::11"}]}}

Key:   □ée                                                                                                                                                                                          Offset: 91
Value:   10,3,"ens3",1533548213,60,"detected","attack","3 consecutive ICMPv6 ping packets in a second. Possibly ICMP ping flood.",{"event_1":{"timestamp":1533548213.620535,"counter":430,"attributes":[{"icmpv6.type":128},{"lowp    Partition: 0
an.iphc_dst":"2001:720:1710:4:1::11"}]},"event_2":{"timestamp":1533548213.645616,"counter":433,"attributes":[{"icmpv6.type":128},{"lowpan.iphc_dst":"2001:720:1710:4:1::11"}]},"event_3":{"timestamp":1533548213.668678,"counter":43
6,"attributes":[{"icmpv6.type":128},{"lowpan.iphc_dst":"2001:720:1710:4:1::11"}]}}

Key:   □ée                                                                                                                                                                                          Offset: 92
Value:   10,3,"ens3",1533548213,60,"detected","attack","3 consecutive ICMPv6 ping packets in a second. Possibly ICMP ping flood.",{"event_1":{"timestamp":1533548213.645616,"counter":433,"attributes":[{"icmpv6.type":128},{"lowp    Partition: 0
an.iphc_dst":"2001:720:1710:4:1::11"}]},"event_2":{"timestamp":1533548213.668678,"counter":436,"attributes":[{"icmpv6.type":128},{"lowpan.iphc_dst":"2001:720:1710:4:1::11"}]},"event_3":{"timestamp":1533548213.682733,"counter":43
8,"attributes":[{"icmpv6.type":128},{"lowpan.iphc_dst":"2001:720:1710:4:1::11"}]}}

Key:   □ée                                                                                                                                                                                          Offset: 93
Value:   10,3,"ens3",1533548218,60,"detected","attack","3 consecutive ICMPv6 ping packets in a second. Possibly ICMP ping flood.",{"event_1":{"timestamp":1533548218.639863,"counter":456,"attributes":[{"icmpv6.type":128},{"lowp    Partition: 0
an.iphc_dst":"2001:720:1710:4:1::11"}]},"event_2":{"timestamp":1533548218.671094,"counter":459,"attributes":[{"icmpv6.type":128},{"lowpan.iphc_dst":"2001:720:1710:4:1::11"}]},"event_3":{"timestamp":1533548218.693193,"counter":46
2,"attributes":[{"icmpv6.type":128},{"lowpan.iphc_dst":"2001:720:1710:4:1::11"}]}}

Key:   □ée                                                                                                                                                                                          Offset: 94
Value:   10,3,"ens3",1533548223,60,"detected","attack","3 consecutive ICMPv6 ping packets in a second. Possibly ICMP ping flood.",{"event_1":{"timestamp":1533548223.638292,"counter":478,"attributes":[{"icmpv6.type":128},{"lowp    Partition: 0
an.iphc_dst":"2001:720:1710:4:1::11"}]},"event_2":{"timestamp":1533548223.662455,"counter":481,"attributes":[{"icmpv6.type":128},{"lowpan.iphc_dst":"2001:720:1710:4:1::11"}]},"event_3":{"timestamp":1533548223.680534,"counter":48
3,"attributes":[{"icmpv6.type":128},{"lowpan.iphc_dst":"2001:720:1710:4:1::11"}]}}
```

Figure 12. Topics view in Kafka broker

Data Filtering and pre-processing Component is subscribing to the topics that are collecting data from sensors, does filtering and pre-processing using Apache Storm and then sends the cleaned data to Incident Detector.

4.1.3 Policy Editor Tool & Supporting Components

With Security Orchestrator in place the Policy Editor Tool can be deployed and used, as shown in Figure 13.



Policy Editor Tool

HSPL-0 — Priority: 1000

Action: authorise_access — The action performed over the object

Object: PANA_traffic — The object for the action

Subject: PEANA-1 — The subject for the action

Fields

Traffic target: PANA_agent

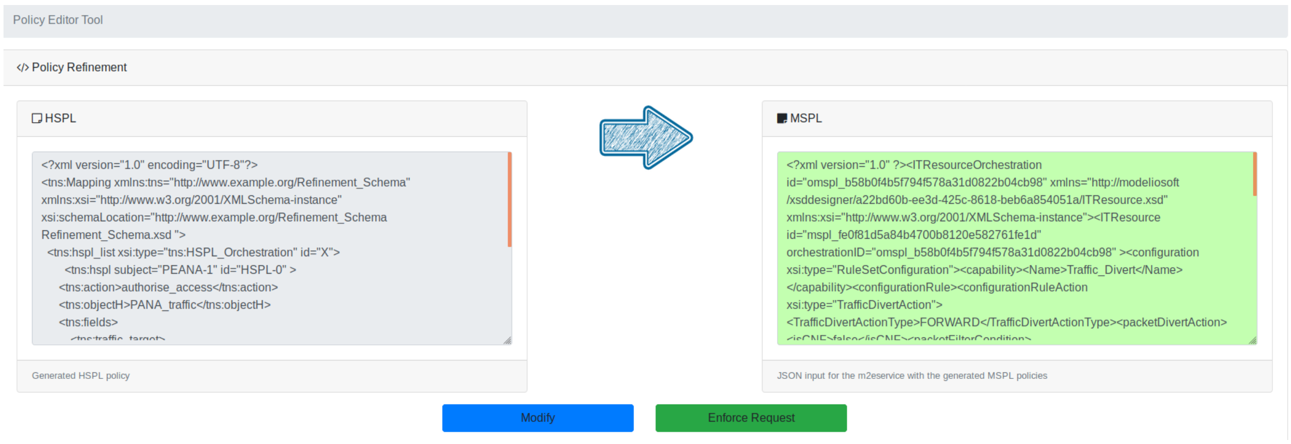
Purpose: Select purpose...

Resource: Select resource...

Specific fields depending on the action/object

Figure 13. Example of Policy Editor Tool screenshot

The policies are then refined from the HSPL level to MSPL level, as shown in Figure 14.



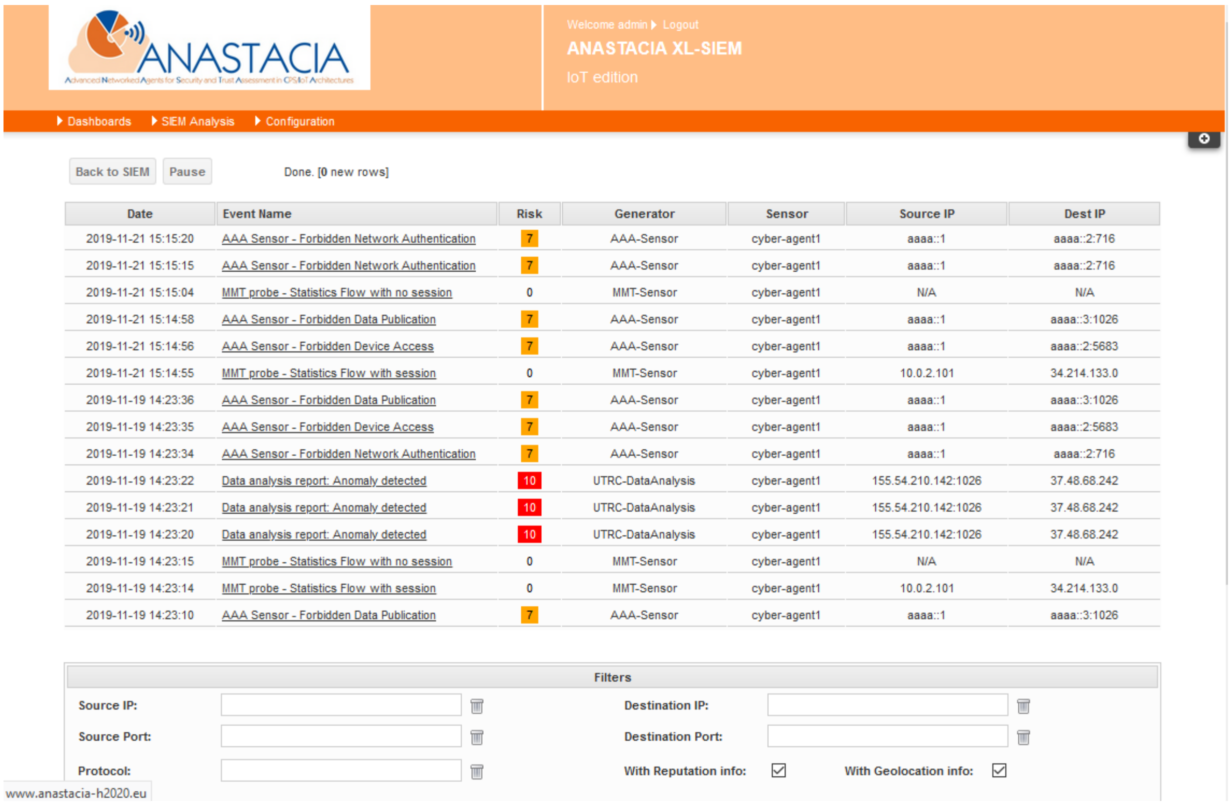
```
<?xml version="1.0" encoding="UTF-8"?>
<tns:Mapping xmlns:tns="http://www.example.org/Refinement_Schema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.example.org/Refinement_Schema
Refinement_Schema.xsd ">
 <tns:hspl_list xsi:type="tns:HSPL_Orchestration" id="X">
      <tns:hspl subject="PEANA-1" id="HSPL-0" >
    <tns:action>authorise_access</tns:action>
    <tns:objectH>PANA_traffic</tns:objectH>
    <tns:fields>
```

```
<?xml version="1.0" ?><ITResourceOrchestration
id="omspl_b58b0f4b5f794f578a31d0822b04cb98" xmlns="http://modeliosoft
/xsddesigner/a22bd60b-ee3d-425c-8618-beb6a854051a/ITResource.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"><ITResource
id="mspl_fe0f81d5a84b4700b8120e582761fe1d"
orchestrationID="omspl_b58b0f4b5f794f578a31d0822b04cb98" ><configuration
xsi:type="RuleSetConfiguration"><capability><Name>Traffic_Divert</Name>
</capability><configurationRule><configurationRuleAction
xsi:type="TrafficDivertAction">
<TrafficDivertActionType>FORWARD</TrafficDivertActionType><packetDivertAction>
```

Figure 14. Example of Policy Editor Tool screenshot

4.1.4 Incident Detector UI

Figure 15 represents the list of events received by the Incident detector after its normalization by the Atos agent and received from security probes through the Data Filtering component. Figure 16 represents the list of alerts triggered by the Incident detector after correlating the events represented in Figure 15.

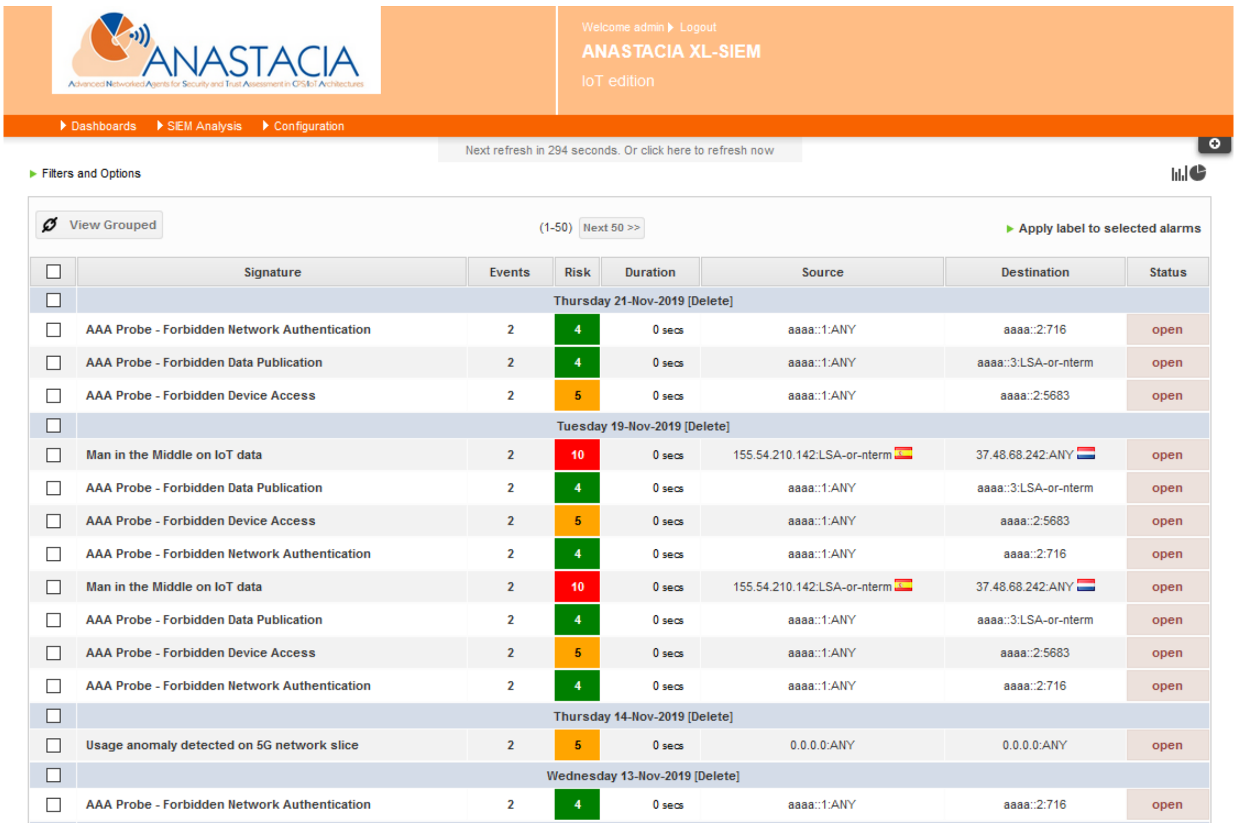


Date	Event Name	Risk	Generator	Sensor	Source IP	Dest IP
2019-11-21 15:15:20	AAA Sensor - Forbidden Network Authentication	7	AAA-Sensor	cyber-agent1	aaaa::1	aaaa::2:716
2019-11-21 15:15:15	AAA Sensor - Forbidden Network Authentication	7	AAA-Sensor	cyber-agent1	aaaa::1	aaaa::2:716
2019-11-21 15:15:04	MMT probe - Statistics Flow with no session	0	MMT-Sensor	cyber-agent1	N/A	N/A
2019-11-21 15:14:58	AAA Sensor - Forbidden Data Publication	7	AAA-Sensor	cyber-agent1	aaaa::1	aaaa::3:1026
2019-11-21 15:14:56	AAA Sensor - Forbidden Device Access	7	AAA-Sensor	cyber-agent1	aaaa::1	aaaa::2:5683
2019-11-21 15:14:55	MMT probe - Statistics Flow with session	0	MMT-Sensor	cyber-agent1	10.0.2.101	34.214.133.0
2019-11-19 14:23:36	AAA Sensor - Forbidden Data Publication	7	AAA-Sensor	cyber-agent1	aaaa::1	aaaa::3:1026
2019-11-19 14:23:35	AAA Sensor - Forbidden Device Access	7	AAA-Sensor	cyber-agent1	aaaa::1	aaaa::2:5683
2019-11-19 14:23:34	AAA Sensor - Forbidden Network Authentication	7	AAA-Sensor	cyber-agent1	aaaa::1	aaaa::2:716
2019-11-19 14:23:22	Data analysis report: Anomaly detected	10	UTRC-DataAnalysis	cyber-agent1	155.54.210.142:1026	37.48.68.242
2019-11-19 14:23:21	Data analysis report: Anomaly detected	10	UTRC-DataAnalysis	cyber-agent1	155.54.210.142:1026	37.48.68.242
2019-11-19 14:23:20	Data analysis report: Anomaly detected	10	UTRC-DataAnalysis	cyber-agent1	155.54.210.142:1026	37.48.68.242
2019-11-19 14:23:15	MMT probe - Statistics Flow with no session	0	MMT-Sensor	cyber-agent1	N/A	N/A
2019-11-19 14:23:14	MMT probe - Statistics Flow with session	0	MMT-Sensor	cyber-agent1	10.0.2.101	34.214.133.0
2019-11-19 14:23:10	AAA Sensor - Forbidden Data Publication	7	AAA-Sensor	cyber-agent1	aaaa::1	aaaa::3:1026

Filters

Source IP:		Destination IP:	
Source Port:		Destination Port:	
Protocol:		With Reputation info: ✓	With Geolocation info: ✓

Figure 15. Events panel at the Incident Detector GUI

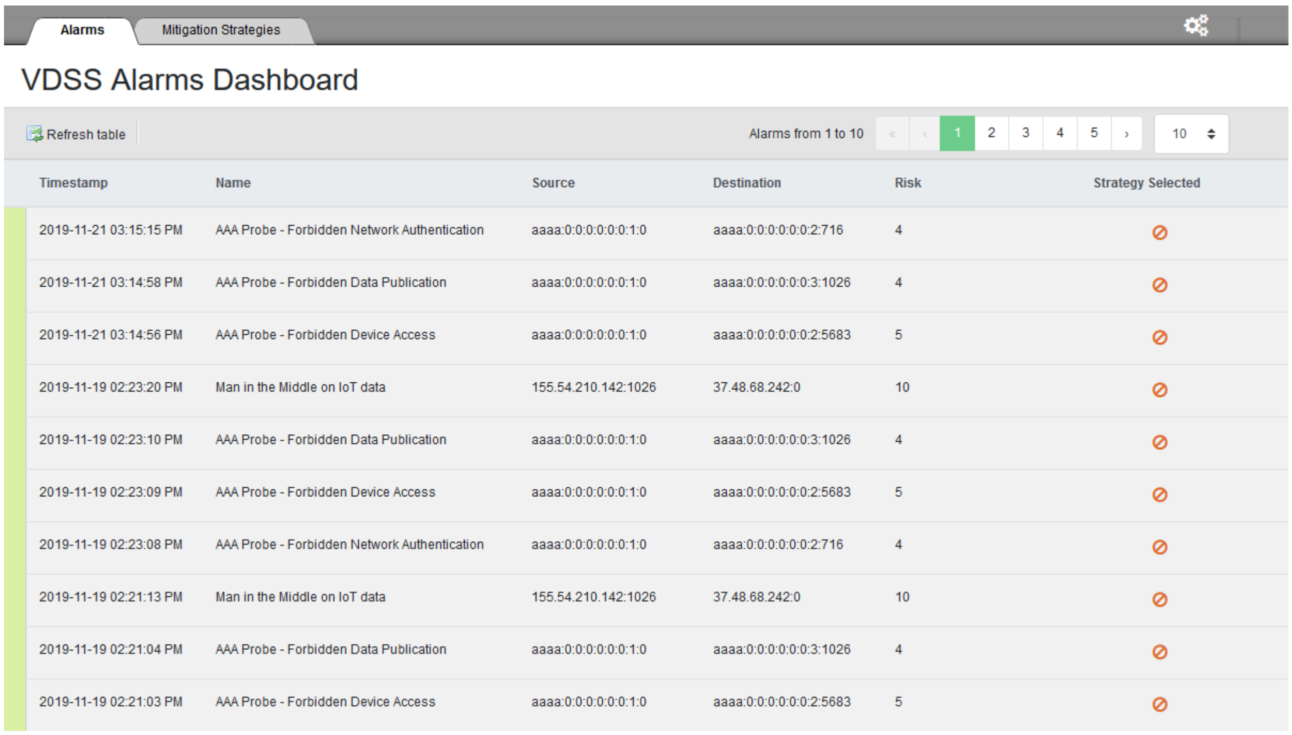


☐	Signature	Events	Risk	Duration	Source	Destination	Status
☐	Thursday 21-Nov-2019 [Delete]						
☐	AAA Probe - Forbidden Network Authentication	2	4	0 secs	aaaa::1:ANY	aaaa::2:716	open
☐	AAA Probe - Forbidden Data Publication	2	4	0 secs	aaaa::1:ANY	aaaa::3:LSA-or-nterm	open
☐	AAA Probe - Forbidden Device Access	2	5	0 secs	aaaa::1:ANY	aaaa::2:5683	open
☐	Tuesday 19-Nov-2019 [Delete]						
☐	Man in the Middle on IoT data	2	10	0 secs	155.54.210.142:LSA-or-nterm	37.48.68.242:ANY	open
☐	AAA Probe - Forbidden Data Publication	2	4	0 secs	aaaa::1:ANY	aaaa::3:LSA-or-nterm	open
☐	AAA Probe - Forbidden Device Access	2	5	0 secs	aaaa::1:ANY	aaaa::2:5683	open
☐	AAA Probe - Forbidden Network Authentication	2	4	0 secs	aaaa::1:ANY	aaaa::2:716	open
☐	Man in the Middle on IoT data	2	10	0 secs	155.54.210.142:LSA-or-nterm	37.48.68.242:ANY	open
☐	AAA Probe - Forbidden Data Publication	2	4	0 secs	aaaa::1:ANY	aaaa::3:LSA-or-nterm	open
☐	AAA Probe - Forbidden Device Access	2	5	0 secs	aaaa::1:ANY	aaaa::2:5683	open
☐	AAA Probe - Forbidden Network Authentication	2	4	0 secs	aaaa::1:ANY	aaaa::2:716	open
☐	Thursday 14-Nov-2019 [Delete]						
☐	Usage anomaly detected on 5G network slice	2	5	0 secs	0.0.0.0:ANY	0.0.0.0:ANY	open
☐	Wednesday 13-Nov-2019 [Delete]						
☐	AAA Probe - Forbidden Network Authentication	2	4	0 secs	aaaa::1:ANY	aaaa::2:716	open

Figure 16. Alerts panel at the Incident Detector GUI

4.1.5 VDSS UI

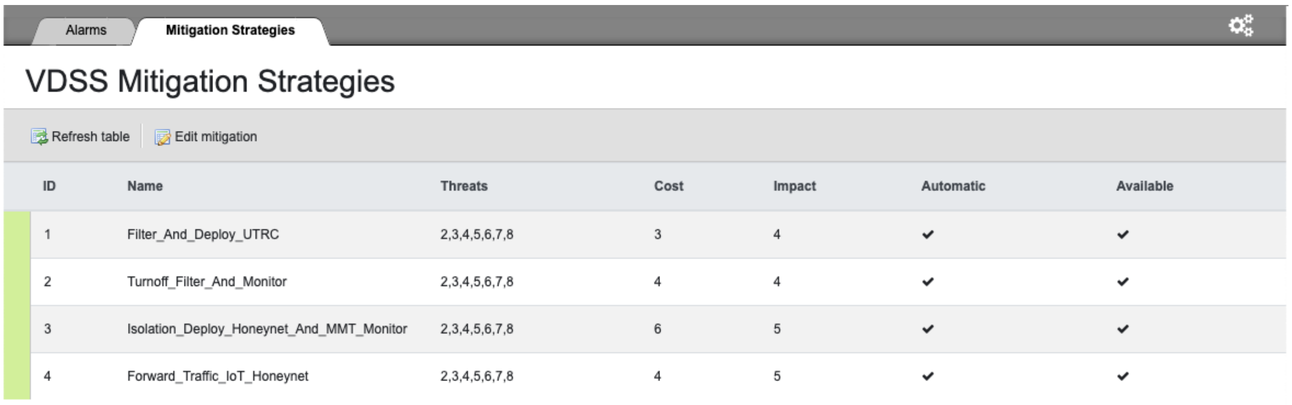
Figure 17 represents the view of the alarms received by the VDSS from the Incident Detector and information about the associated risk and whether there has been enforced any mitigation for it.



Timestamp	Name	Source	Destination	Risk	Strategy Selected
2019-11-21 03:15:15 PM	AAA Probe - Forbidden Network Authentication	aaaa:0:0:0:0:0:0:1:0	aaaa:0:0:0:0:0:0:2:716	4	⊘
2019-11-21 03:14:58 PM	AAA Probe - Forbidden Data Publication	aaaa:0:0:0:0:0:0:1:0	aaaa:0:0:0:0:0:0:3:1026	4	⊘
2019-11-21 03:14:56 PM	AAA Probe - Forbidden Device Access	aaaa:0:0:0:0:0:0:1:0	aaaa:0:0:0:0:0:0:2:5683	5	⊘
2019-11-19 02:23:20 PM	Man in the Middle on IoT data	155.54.210.142:1026	37.48.68.242:0	10	⊘
2019-11-19 02:23:10 PM	AAA Probe - Forbidden Data Publication	aaaa:0:0:0:0:0:0:1:0	aaaa:0:0:0:0:0:0:3:1026	4	⊘
2019-11-19 02:23:09 PM	AAA Probe - Forbidden Device Access	aaaa:0:0:0:0:0:0:1:0	aaaa:0:0:0:0:0:0:2:5683	5	⊘
2019-11-19 02:23:08 PM	AAA Probe - Forbidden Network Authentication	aaaa:0:0:0:0:0:0:1:0	aaaa:0:0:0:0:0:0:2:716	4	⊘
2019-11-19 02:21:13 PM	Man in the Middle on IoT data	155.54.210.142:1026	37.48.68.242:0	10	⊘
2019-11-19 02:21:04 PM	AAA Probe - Forbidden Data Publication	aaaa:0:0:0:0:0:0:1:0	aaaa:0:0:0:0:0:0:3:1026	4	⊘
2019-11-19 02:21:03 PM	AAA Probe - Forbidden Device Access	aaaa:0:0:0:0:0:0:1:0	aaaa:0:0:0:0:0:0:2:5683	5	⊘

Figure 17. Example ATOS VDSS UI screenshot showing Alarms and their mitigation status

Figure 18 depicts the list of mitigation strategies available at the VDSS for a given execution time. It contains information of the threats that they are capable of mitigating (identified by their alert id), details about the cost of and impact of deploying such mitigation and whether the mitigation strategy is to be deployed automatically, without CISO intervention, whether it is available.



Alarms | Mitigation Strategies

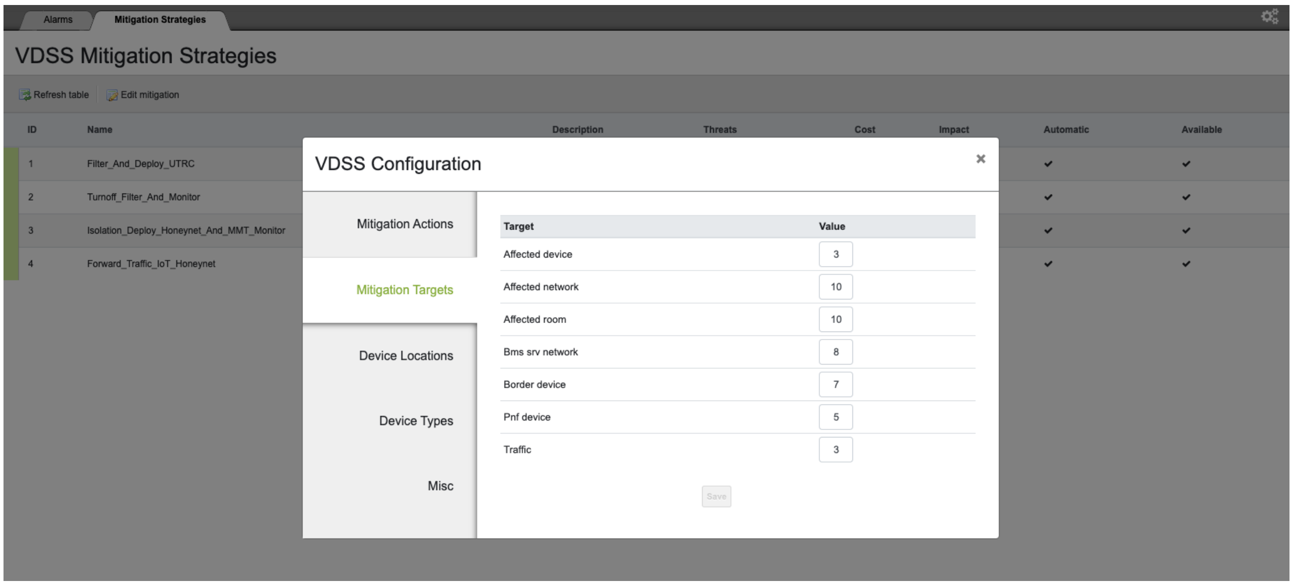
VDSS Mitigation Strategies

Refresh table | Edit mitigation

ID	Name	Threats	Cost	Impact	Automatic	Available
1	Filter_And_Deploy_UTRC	2,3,4,5,6,7,8	3	4	✓	✓
2	Turnoff_Filter_And_Monitor	2,3,4,5,6,7,8	4	4	✓	✓
3	Isolation_Deploy_Honeynet_And_MMT_Monitor	2,3,4,5,6,7,8	6	5	✓	✓
4	Forward_Traffic_IoT_Honeynet	2,3,4,5,6,7,8	4	5	✓	✓

Figure 18. View of the mitigation strategies used by the VDSS at a given moment

These cost and impact values can be configured at the VDSS by the CISO, as it can be seen in screenshots represented in Figure 19 (for configuring Mitigation Targets) and Figure 20 (for configuring Mitigation Actions).



Alarms | Mitigation Strategies

VDSS Mitigation Strategies

Refresh table | Edit mitigation

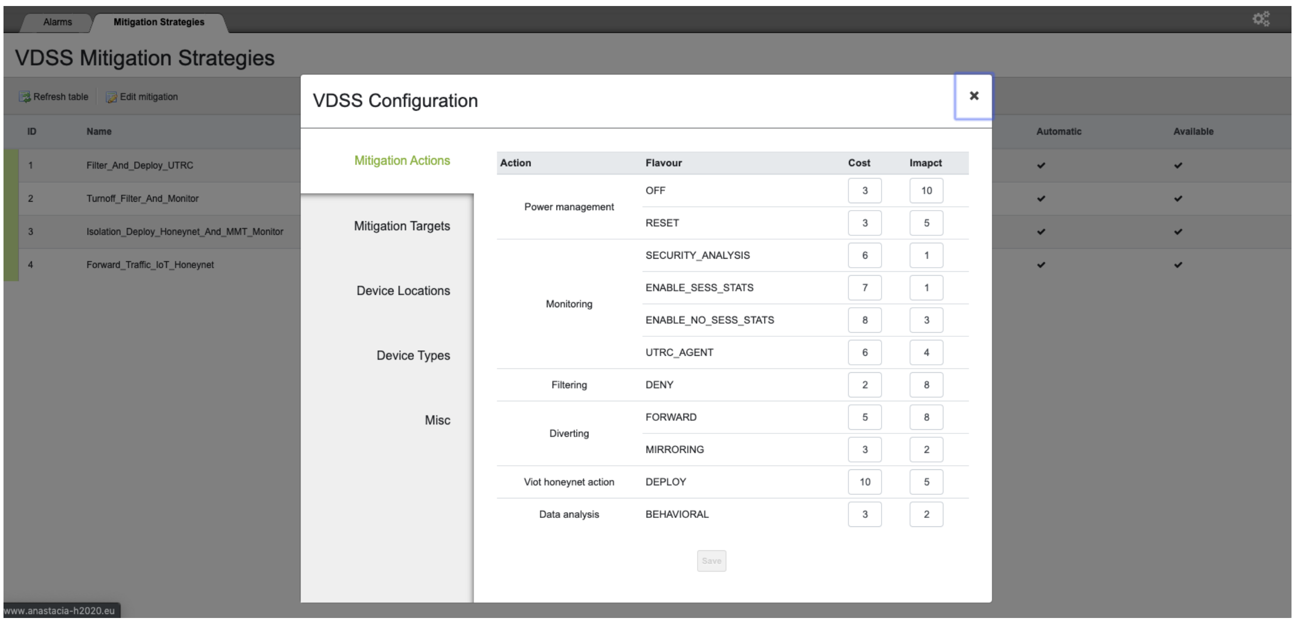
VDSS Configuration ×

Mitigation Actions
Mitigation Targets
Device Locations
Device Types
Misc

Target	Value
Affected device	3
Affected network	10
Affected room	10
Bms srv network	8
Border device	7
Pnf device	5
Traffic	3

Save

Figure 19. Mitigation Targets configuration screen at the VDSS



ID	Name	Automatic	Available
1	Filter_And_Deploy_UTRC	✓	✓
2	Turnoff_Filter_And_Monitor	✓	✓
3	Isolation_Deploy_Honeynet_And_MMT_Monitor	✓	✓
4	Forward_Traffic_IoT_Honeynet	✓	✓

Action	Flavour	Cost	Imapct
Power management	OFF	3	10
	RESET	3	5
Monitoring	SECURITY_ANALYSIS	6	1
	ENABLE_SESS_STATS	7	1
	ENABLE_NO_SESS_STATS	8	3
	UTRC_AGENT	6	4
Filtering	DENY	2	8
Diverting	FORWARD	5	8
	MIRRORING	3	2
Viot honeynet action	DEPLOY	10	5
Data analysis	BEHAVIORAL	3	2

Figure 20. Screen for configuring Mitigation Actions properties at the VDSS

4.1.6 Security Alert Service

The Dockerfile for the Security Alert Service is used for setup is presented below. As SAS is a backend service no UI is provided but the service is available on ports 8080 and 5671.

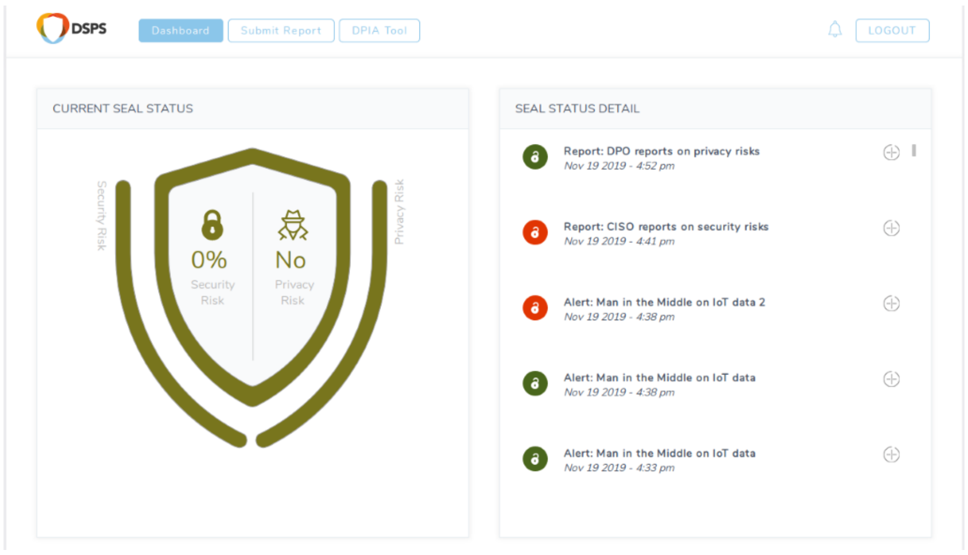
```
FROM ubuntu:16.04
MAINTAINER CNR-IEIIT (security@ge.ieiit.cnr.it)
ENV HOME /root
ENV DEBIAN_FRONTEND noninteractive
ENV TERM xterm
ADD . /app
WORKDIR /tmp
RUN apt-get update
RUN apt-get upgrade -y
RUN apt-get install -y apt-utils
RUN apt-get install -y default-jre
RUN apt-get install -y wget
RUN apt-get install -y build-essential
RUN apt-get update -y
RUN apt-get install -y locales
RUN locale-gen en_US en_US.UTF-8

# installation of erlang
RUN apt-get -y install autoconf
RUN apt-get -y install m4
RUN apt-get -y install libncurses5-dev
RUN apt-get -y install libgl1-mesa-dev libglu1-mesa-dev libpng3
RUN apt-get -y install libssh-dev
RUN apt-get -y install unixodbc-dev
RUN wget https://packages.erlang-solutions.com/erlang/debian/pool/esl-
erlang_21.3.1-1~ubuntu~xenial_amd64.deb
RUN apt install -y ./esl-erlang_21.3.1-1~ubuntu~xenial_amd64.deb
```

```
# installation of rabbitmq server
RUN wget https://github.com/rabbitmq/rabbitmq-
server/releases/download/v3.7.13/rabbitmq-server_3.7.13-1_all.deb
RUN apt install -y ./rabbitmq-server_3.7.13-1_all.deb

# configuration
WORKDIR /app
RUN cp -r rabbitmq/* /etc/rabbitmq/
RUN chown -R rabbitmq:rabbitmq /etc/rabbitmq/rabbitmq.conf
RUN chown -R rabbitmq:rabbitmq /etc/rabbitmq/certs/
RUN rabbitmq-plugins enable rabbitmq_auth_mechanism_ssl
RUN rabbitmq-plugins enable rabbitmq_management

# installation of the sas software
RUN apt-get install -y python-minimal python-pip
RUN pip install --upgrade pip
RUN pip install -r requirements.txt

EXPOSE 8080 5671

COPY ./start.sh /
CMD ["sh", "/start.sh"]
```

Figure 21. Example of security alert service component screenshot

4.1.7 Security and Privacy Seal

The following figure shows the main DSPS GUI view:



Report: DPO reports on privacy risks
Nov 19 2019 - 4:52 pm

Report: CISO reports on security risks
Nov 19 2019 - 4:41 pm

Alert: Man in the Middle on IoT data 2
Nov 19 2019 - 4:38 pm

Alert: Man in the Middle on IoT data
Nov 19 2019 - 4:38 pm

Alert: Man in the Middle on IoT data
Nov 19 2019 - 4:33 pm

Figure 22. Example of Security and Privacy Seal with Seal status details

The security and privacy risk gouge as well as the health shield represents the live security risk at the precise moment of a change in the risk evaluation thus providing a direct user feedback of the overall risk.

In the following figure you can see the seal degradation evolution from healthy to compromised seal.

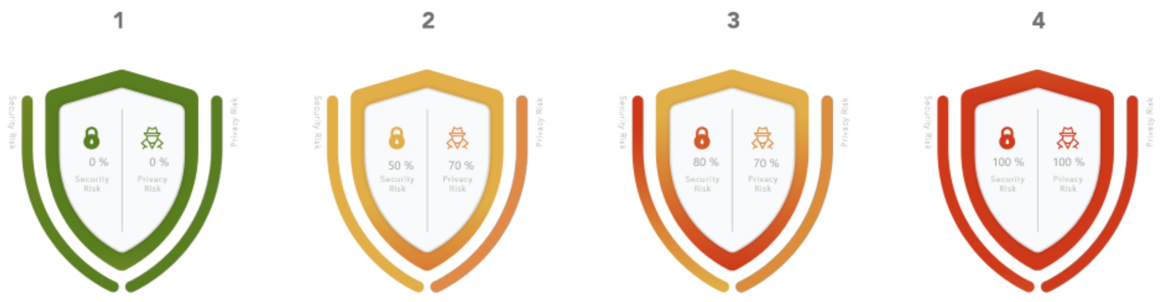
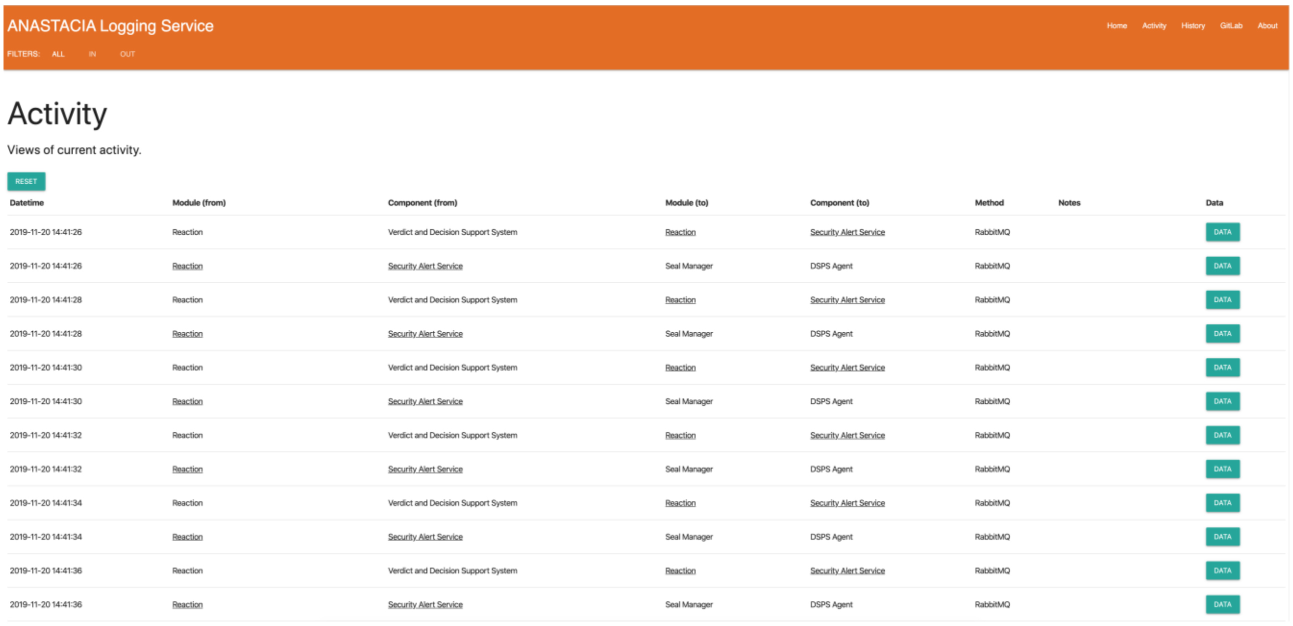


Figure 23. Seal degradation evolution

For more details on this please see deliverable D5.3[8] .

4.1.8 Logging Service UI

For the purpose of better overview of the activities and for making easier the measurement during testing, and additional component has been developed. All components of the platform are connecting to this component in order to send logs of the activities, therefore this component is component is not included to the architecture in order to keep it easier to read and understand. Screenshots of the Logging Service are provided below.

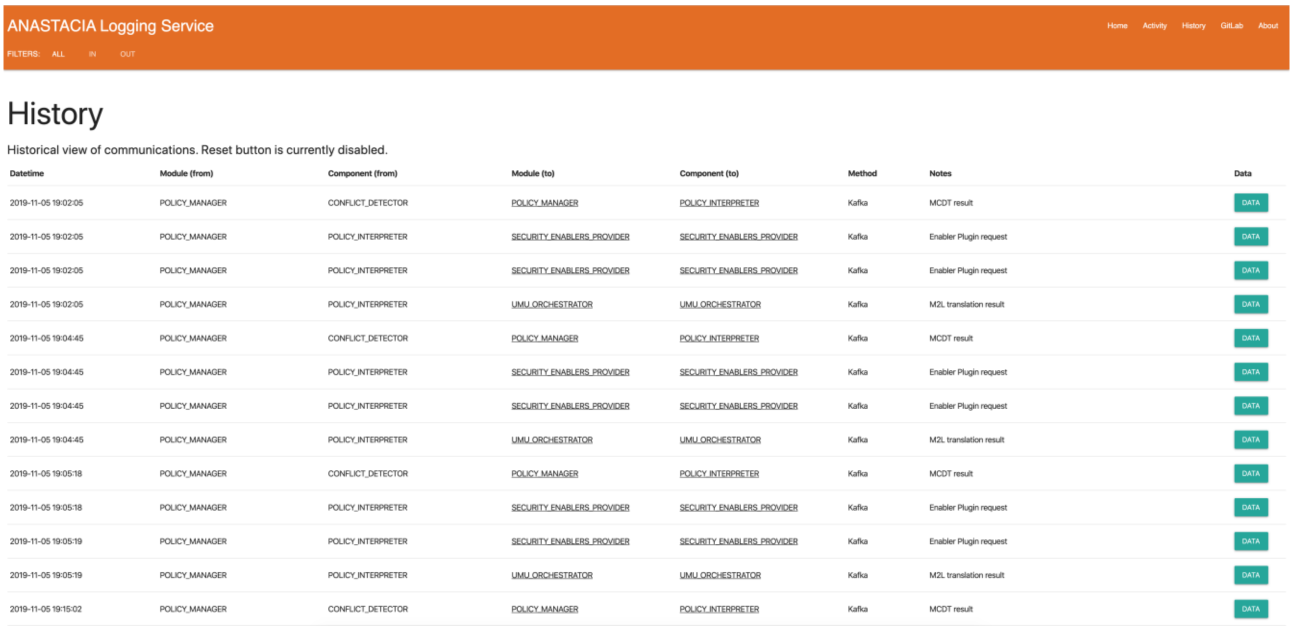


Activity

Views of current activity.

Datetime	Module (from)	Component (from)	Module (to)	Component (to)	Method	Notes	Data
2019-11-20 14:41:26	Reaction	Verdict and Decision Support System	Reaction	Security Alert Service	RabbitMQ		DATA
2019-11-20 14:41:26	Reaction	Security Alert Service	Seal Manager	DSPS Agent	RabbitMQ		DATA
2019-11-20 14:41:28	Reaction	Verdict and Decision Support System	Reaction	Security Alert Service	RabbitMQ		DATA
2019-11-20 14:41:28	Reaction	Security Alert Service	Seal Manager	DSPS Agent	RabbitMQ		DATA
2019-11-20 14:41:30	Reaction	Verdict and Decision Support System	Reaction	Security Alert Service	RabbitMQ		DATA
2019-11-20 14:41:30	Reaction	Security Alert Service	Seal Manager	DSPS Agent	RabbitMQ		DATA
2019-11-20 14:41:32	Reaction	Verdict and Decision Support System	Reaction	Security Alert Service	RabbitMQ		DATA
2019-11-20 14:41:32	Reaction	Security Alert Service	Seal Manager	DSPS Agent	RabbitMQ		DATA
2019-11-20 14:41:34	Reaction	Verdict and Decision Support System	Reaction	Security Alert Service	RabbitMQ		DATA
2019-11-20 14:41:34	Reaction	Security Alert Service	Seal Manager	DSPS Agent	RabbitMQ		DATA
2019-11-20 14:41:36	Reaction	Verdict and Decision Support System	Reaction	Security Alert Service	RabbitMQ		DATA
2019-11-20 14:41:36	Reaction	Security Alert Service	Seal Manager	DSPS Agent	RabbitMQ		DATA

Figure 24. Example of CNR logging service UI used in ANASTACIA framework testing (Activity).



History

Historical view of communications. Reset button is currently disabled.

Datetime	Module (from)	Component (from)	Module (to)	Component (to)	Method	Notes	Data
2019-11-05 19:02:05	POLICY_MANAGER	CONFLICT_DETECTOR	POLICY_MANAGER	POLICY_INTERPRETER	Kafka	MCDT result	DATA
2019-11-05 19:02:05	POLICY_MANAGER	POLICY_INTERPRETER	SECURITY_ENABLERS_PROVIDER	SECURITY_ENABLERS_PROVIDER	Kafka	Enabler Plugin request	DATA
2019-11-05 19:02:05	POLICY_MANAGER	POLICY_INTERPRETER	SECURITY_ENABLERS_PROVIDER	SECURITY_ENABLERS_PROVIDER	Kafka	Enabler Plugin request	DATA
2019-11-05 19:02:05	POLICY_MANAGER	POLICY_INTERPRETER	UMU_ORCHESTRATOR	UMU_ORCHESTRATOR	Kafka	M2L translation result	DATA
2019-11-05 19:04:45	POLICY_MANAGER	CONFLICT_DETECTOR	POLICY_MANAGER	POLICY_INTERPRETER	Kafka	MCDT result	DATA
2019-11-05 19:04:45	POLICY_MANAGER	POLICY_INTERPRETER	SECURITY_ENABLERS_PROVIDER	SECURITY_ENABLERS_PROVIDER	Kafka	Enabler Plugin request	DATA
2019-11-05 19:04:45	POLICY_MANAGER	POLICY_INTERPRETER	SECURITY_ENABLERS_PROVIDER	SECURITY_ENABLERS_PROVIDER	Kafka	Enabler Plugin request	DATA
2019-11-05 19:04:45	POLICY_MANAGER	POLICY_INTERPRETER	UMU_ORCHESTRATOR	UMU_ORCHESTRATOR	Kafka	M2L translation result	DATA
2019-11-05 19:05:18	POLICY_MANAGER	CONFLICT_DETECTOR	POLICY_MANAGER	POLICY_INTERPRETER	Kafka	MCDT result	DATA
2019-11-05 19:05:18	POLICY_MANAGER	POLICY_INTERPRETER	SECURITY_ENABLERS_PROVIDER	SECURITY_ENABLERS_PROVIDER	Kafka	Enabler Plugin request	DATA
2019-11-05 19:05:19	POLICY_MANAGER	POLICY_INTERPRETER	SECURITY_ENABLERS_PROVIDER	SECURITY_ENABLERS_PROVIDER	Kafka	Enabler Plugin request	DATA
2019-11-05 19:05:19	POLICY_MANAGER	POLICY_INTERPRETER	UMU_ORCHESTRATOR	UMU_ORCHESTRATOR	Kafka	M2L translation result	DATA
2019-11-05 19:15:02	POLICY_MANAGER	CONFLICT_DETECTOR	POLICY_MANAGER	POLICY_INTERPRETER	Kafka	MCDT result	DATA

Figure 25. Example of CNR logging service UI used in ANASTACIA framework testing (History).

4.1.9 Data Analysis Dockerfile Used for Component Wrapping

Finally, in the Listing below shows up example of Docker file used to wrap UTRC made components into Docker image that can be easily deployed on any Docker supported environment including UMU deployment.

```
# Import Ubuntu
FROM ubuntu:19.04

ARG APP_ID=utrc_agent

# Change main developer accordingly to application assignment
LABEL maintainer=sobonsp@utrc.utc.com company='UTRC Ireland' project=
${APP_ID} link=https://www.anastacia-h2020.eu/

# Update OS
RUN apt-get update -y
RUN apt-get autoremove -y
RUN apt-get clean

# Default working Agent container directory
WORKDIR /${APP_ID}

# Environmental variables
ARG KAFKA_IP
ARG KAFKA_PORT

# Prepare directory structure
RUN mkdir /${APP_ID}/conf/
RUN mkdir /${APP_ID}/log/
RUN mkdir /${APP_ID}/data/
RUN mkdir /${APP_ID}/kpi/
```

```
# Copy Data Analysis Agent content
                             COPY ${APP_ID} /${APP_ID}

RUN chmod +x /${APP_ID}/${APP_ID}

# Expose REST API ports
EXPOSE $UTRC_ECHO_PORT

RUN cd /${APP_ID}

# Run Agent service
CMD ["./" + ${APP_ID}, "-c", "conf/config.yml"]
```
Figure 26. The docker file used for UTRC componentsDeployment Status Model

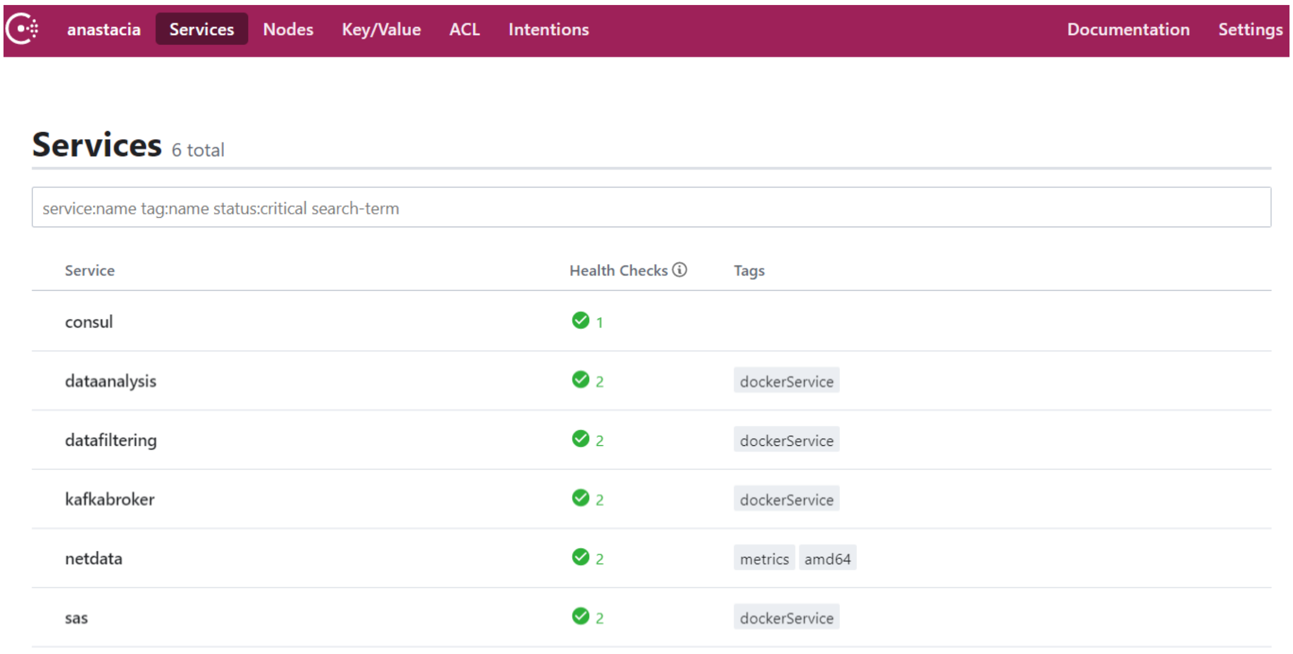
The deployment status of the components is provided in the table below.

Table 22. ANASTACIA deployment status for final demonstrator

Component	Partner	Deployment Status
Policy Manager	UMU	Deployed in UMU
Policy Interpreter (Policy Manager)	UMU	Deployed in UMU
Policy Conflict Detector (Policy Manager)	UMU	Deployed in UMU
Policy Editor Tool (UI)	UMU	Deployed in UMU
Policy Repository	UMU	Deployed in UMU
Security orchestrator	AALTO	Deployed in UMU
System Model Service (Security Orchestrator)	AALTO	Deployed in UMU
NVF orchestrators	AALTO	Deployed in UMU
SDN controllers	AALTO	Deployed in UMU
Security Enabler Provider	UMU	Deployed in UMU
Data Filtering and pre-processing Component	UBITECH	Deployed in UMU
DPI Analysis	MONT	Deployed in UMU
Data Analysis Component	UTRC	Deployed in UMU
Verdict and Decision Support System (VDSS)	ATOS	Deployed in UMU
XL-SIEM Server	ATOS	Deployed in UMU
XL-SIEM Agent	ATOS	Deployed in UMU
Assets Model	THALES/UTRC	Deployed in UMU

Alerting and reaction dashboard (Incident Detector and VDSS UIs)	ATOS	Deployed in UMU
Mitigation Action Service	MONT	Deployed in UMU
Security Alert Service	CNR	Deployed in UMU
Logging Service (No part of architecture)	CNR	Deployed in UMU
Dynamic Security and Privacy Seal server	MAND, DG, AS	Deployed in UMU
Dynamic Security and Privacy Seal storage (blockchain)	MAND, DG, AS	Deployed in UMU
Dynamic Security and Privacy Seal User Interface	MAND, DG, AS	Deployed in UMU
Dynamic Security and Privacy Seal Agent	MAND, DG, AS	Deployed in UMU

Due to the fact that ANASTACIA is composed by many different components we need a way to easily check the status of all components. For this we adopted the usage of Consul[6], an "API-driven" service discovery solution, by installing a Consul Server (in specific in the same VM with the Kafka broker), and used Consul agents in each VM in order to provide health check for each component. This means that each component the status should be able to be checked. For components providing REST endpoint checking is straightforward and to assist the usage by more components an assistive service has been developed[7]. It is however possible to use Consul with command line arguments as well, and this approach has been also used for some services.



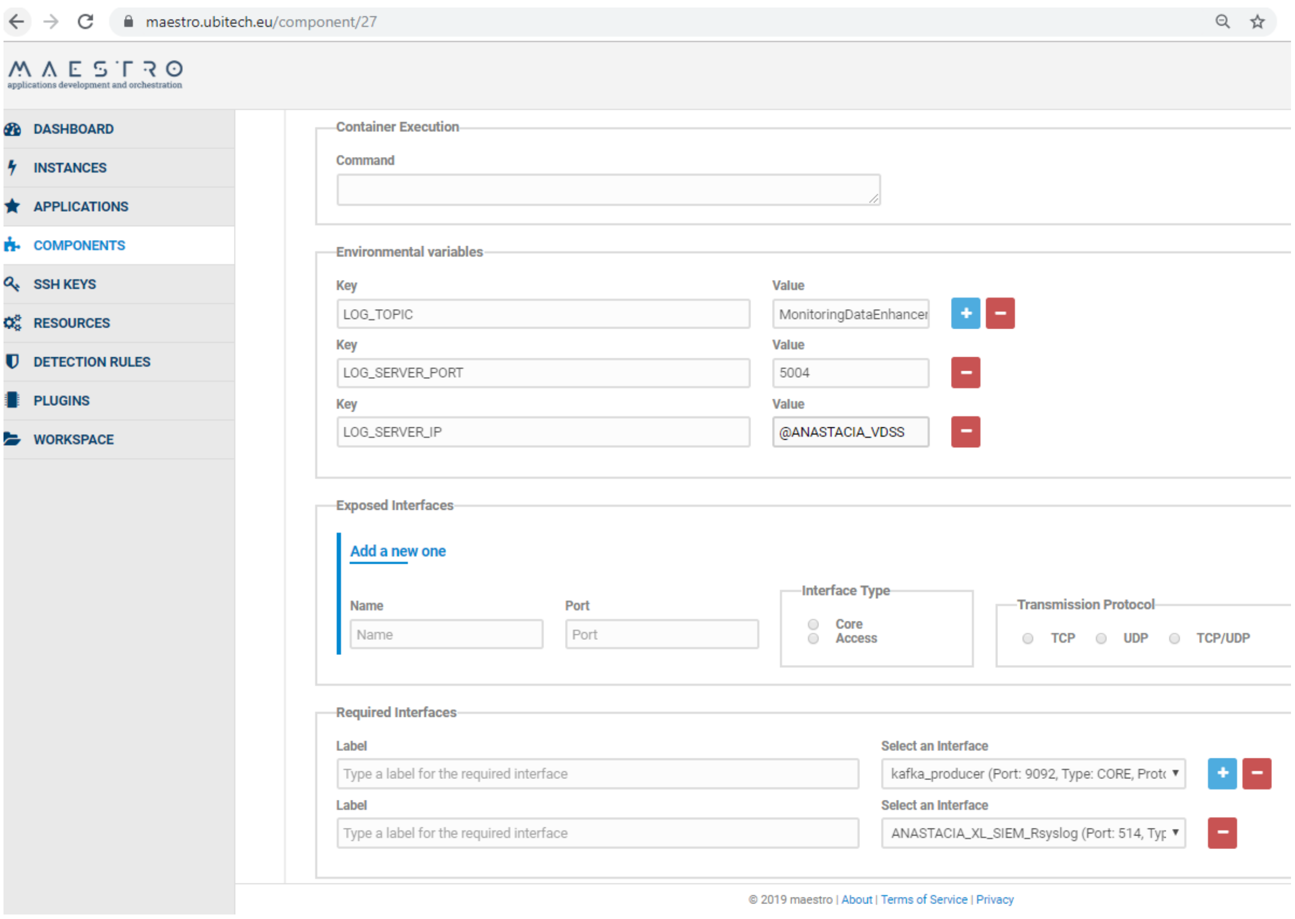
Service	Health Checks	Tags
consul	✓ 1	
dataanalysis	✓ 2	dockerService
datafiltering	✓ 2	dockerService
kafkabroker	✓ 2	dockerService
netdata	✓ 2	metrics amd64
sas	✓ 2	dockerService

Figure 27. Consul Server UI used for the monitoring of the applications

[6] https://www.consul.io/
[7] https://gitlab.com/anastacia-project/echo_service

4.2 Deployment Automation with MAESTRO

It has to be mentioned that in parallel with the deployment of ANASTACIA in the demonstration infrastructure we have been working to make ANASTACIA an application that can be easily deployed with MAESTRO[8]. MAESTRO is a multi-cloud orchestration platform developed by UBITECH that allows the easy deployment and management of microservices and cloud-native applications built with Docker. It provides out of the box embedded monitoring, elasticity and security benefits. This work is in progress and is mostly guided by the exploitation interest of UBITECH towards the ANASTACIA framework. In MAESTRO we define each component that is bundled in a different docker image through the dedicated UI pages. A snapshot of an ANASTACIA component is depicted in Figure 28 below.



maestro.ubitech.eu/component/27
MAESTRO
applications development and orchestration
DASHBOARD
INSTANCES
APPLICATIONS
COMPONENTS
SSH KEYS
RESOURCES
DETECTION RULES
PLUGINS
WORKSPACE
Add a new one
© 2019 maestro | About | Terms of Service | Privacy

Container Execution

Command:

Environmental variables

Key	Value
LOG_TOPIC	MonitoringDataEnhancer
LOG_SERVER_PORT	5004
LOG_SERVER_IP	@ANASTACIA_VDSS

Exposed Interfaces

Name: Name | Port: Port | Interface Type: Core / Access | Transmission Protocol: TCP / UDP / TCP/UDP

Required Interfaces

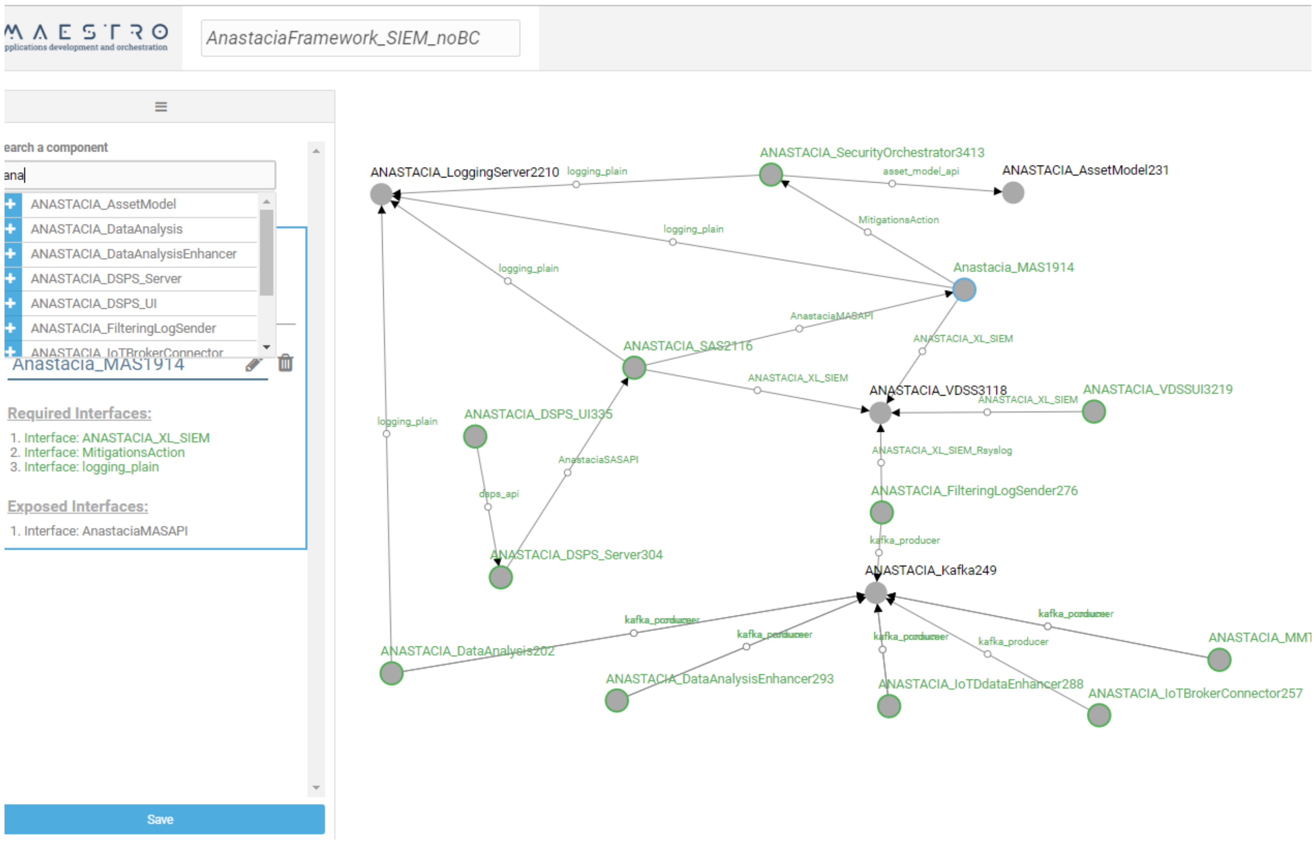
Label	Select an Interface
Type a label for the required interface	kafka_producer (Port: 9092, Type: CORE, Prot
Type a label for the required interface	ANASTACIA_XL_SIEM_Rsyslog (Port: 514, Typ

Figure 28. A snapshot of ANASTACIA component defined in MAESTRO

With MAESTRO we are able not only to define the components, but also the interfaces between them in order to construct the integrated application. Furthermore, with the usage of environmental variables in the ANASTACIA components it is possible to configure the application based on the actual deployment needs and even dynamically using the IP address of other ANASTACIA components.

Then with all ANASTACIA components in place the application template of ANASTACIA framework can be created by adding components in the application canvas and then with drag and drop connect the interfaces of the components, as depicted in Figure 29. Actually, different versions of the framework with inclusion or exclusion of specific components, is possible and easily defined.

[8] https://themaestro.ubitech.eu/



Anastacia_MAS1914

Required Interfaces:
1. Interface: ANASTACIA_XL_SIEM
2. Interface: MitigationsAction
3. Interface: logging_plain

Exposed Interfaces:
1. Interface: AnastaciaMASAPI

Figure 29. Creating ANASTACIA application using MAESTRO

With the Anastacia application template created, a user with appropriate rights can deploy ANASTACIA with a just few clicks to resources registered in the MAESTRO platform. The configuration of components prior to deployment is also possible. The application that is configured and ready to be deployed is depicted in Figure 30. Once the application is correctly deployed the user can see the component status and monitoring metrics of the applications through the UI. At the current state, some of the components have not been fully containerized (such as the Security Orchestrator) so additional, manual steps for their deployment are also needed. Finally, as MAESTRO can support elasticity (http and grpc based scaling at the moment but support for more frameworks such as Apache Storm, Apache Spark is under implementation), scaling of components can also take place if appropriate scaling rules and supported component are used.

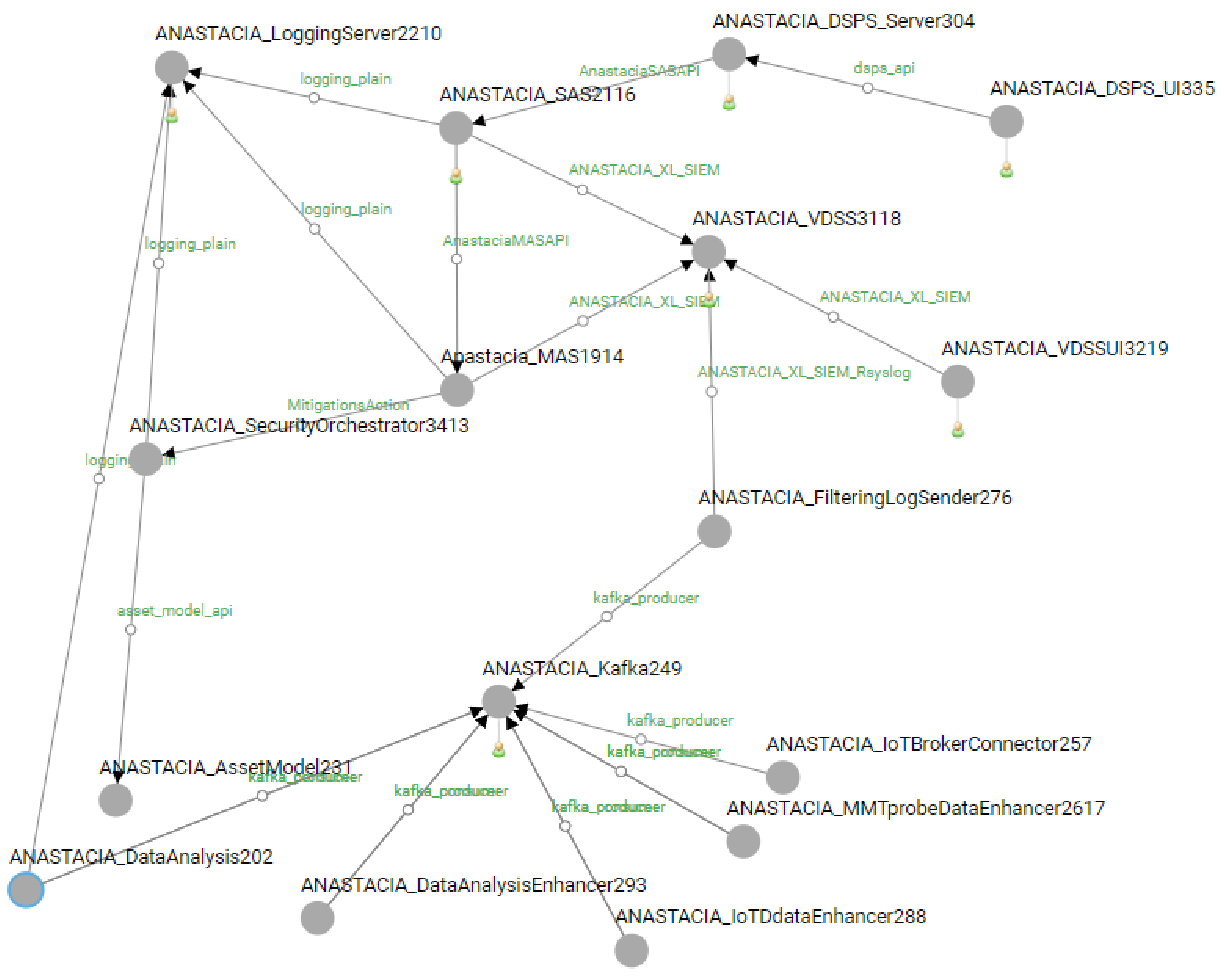


Figure 30. Deployment of ANASTACIA application using MAESTRO

5 Platform Testing and Validation

This section presents an overview of the platform testing and validation as part of the overall evaluation strategy in the context of ANASTACIA, as it was described in D6.1. In ANASTACIA we defined the following facets of testing:

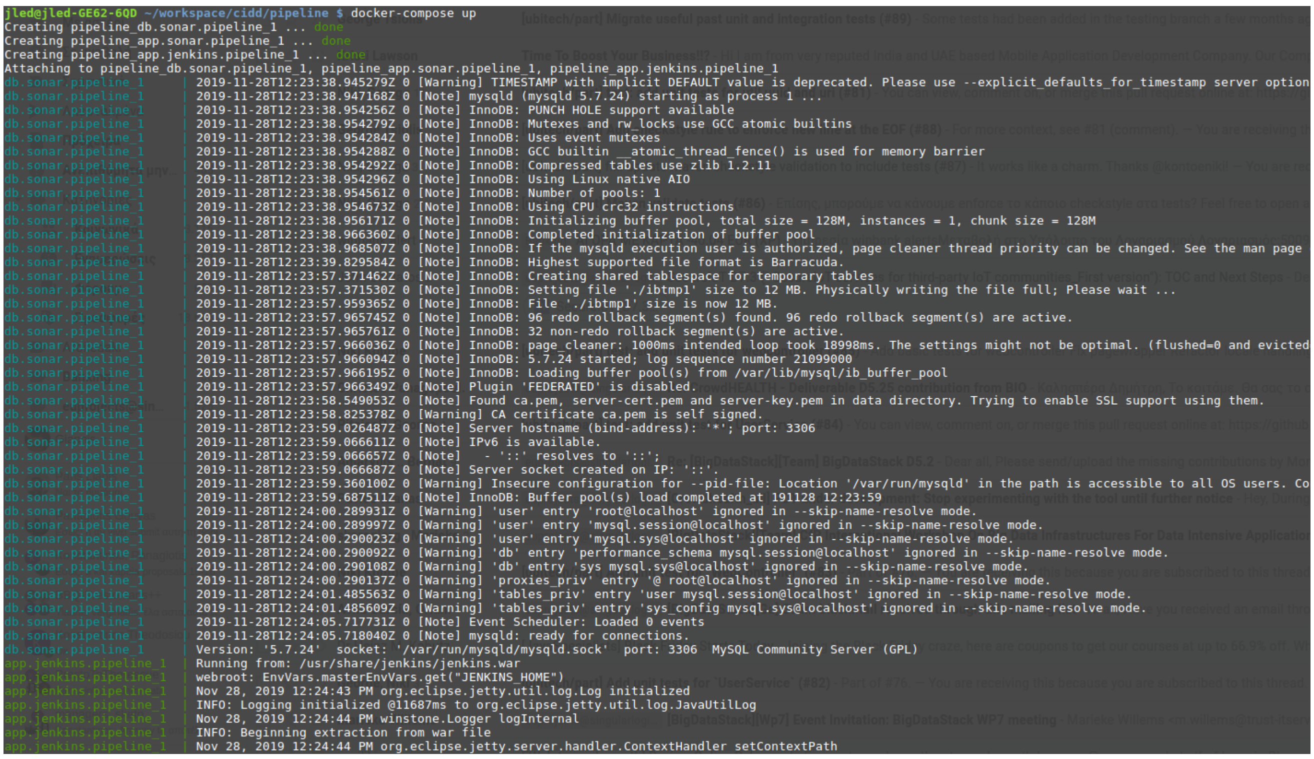
- Source code quality, including unit testing that can be performed by the separate development teams when new functionalities are developed. Results presented in section 5.1.
- Integration testing performed by the development teams in order to test the smooth co-operation between the various layers and components, in accordance also to the CI scheme defined in D6.1. Results presented in section 5.3.
- Stress testing can be performed for the benchmarking of the system or specific components in order to understand the limits and the bottlenecks of the system. Towards this we made an analysis of the scalability aspects of each component, presented in section 5.2 in order to identify possible bottlenecks. Such a component is the Data Filtering and pre-processing Component that receives all the load of incoming data, so stress test can be a supplementary action that can be performed for this component.
- Validation based on the requirements' coverage (functional completeness and correctness). This is covered in deliverable D2.8[9] .
- User acceptance testing will ensure the usability of the system performed during the use case evaluation and documented in deliverables D6.3[10] and the upcoming deliverable D6.6.

5.1 Source Code Quality Control

A parameter that can be used for the validation of the developed software is the quality measurement of source code. Sonar[9] is an open source software quality platform that uses various static code analysis tools such as CheckStyle[10], to extract software metrics, which then can be used to improve software quality.

A pipeline that uses Jenkins and SonarQube pipeline has been setup and used for the purpose of the project in order to assist the integration and testing aspects. For this we have created a docker-compose file[11] that installs both tools in the same machine (as seen in Figure 31).

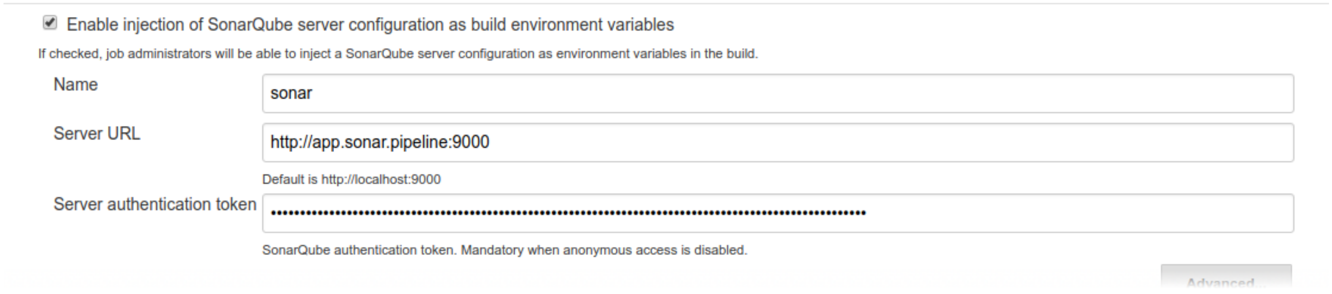
[9] http://www.sonarqube.org
[10] http://cruisecontrol.sourceforge.net
[11] https://gitlab.com/anastacia-project/framework/tree/master/pipeline



```
jled@jled-GE62-6QD ~/workspace/cidd/pipeline $ docker-compose up
Creating pipeline_db.sonar.pipeline_1 ... done
Creating pipeline_app.sonar.pipeline_1 ... done
Creating pipeline_app.jenkins.pipeline_1 ... done
Attaching to pipeline_db.sonar.pipeline_1, pipeline_app.sonar.pipeline_1, pipeline_app.jenkins.pipeline_1
db.sonar.pipeline_1      | 2019-11-28T12:23:38.945279Z 0 [Warning] TIMESTAMP with implicit DEFAULT value is deprecated. Please use --explicit_defaults_for_timestamp server option
db.sonar.pipeline_1      | 2019-11-28T12:23:38.947168Z 0 [Note] mysqld (mysqld 5.7.24) starting as process 1 ...
db.sonar.pipeline_1      | 2019-11-28T12:23:38.954256Z 0 [Note] InnoDB: PUNCH HOLE support available
db.sonar.pipeline_1      | 2019-11-28T12:23:38.954279Z 0 [Note] InnoDB: Mutexes and rw_locks use GCC atomic builtins
db.sonar.pipeline_1      | 2019-11-28T12:23:38.954284Z 0 [Note] InnoDB: Uses event mutexes
db.sonar.pipeline_1      | 2019-11-28T12:23:38.954288Z 0 [Note] InnoDB: GCC builtin __atomic_thread_fence() is used for memory barrier
db.sonar.pipeline_1      | 2019-11-28T12:23:38.954292Z 0 [Note] InnoDB: Compressed tables use zlib 1.2.11
db.sonar.pipeline_1      | 2019-11-28T12:23:38.954296Z 0 [Note] InnoDB: Using Linux native AIO
db.sonar.pipeline_1      | 2019-11-28T12:23:38.954561Z 0 [Note] InnoDB: Number of pools: 1
db.sonar.pipeline_1      | 2019-11-28T12:23:38.954673Z 0 [Note] InnoDB: Using CPU crc32 instructions
db.sonar.pipeline_1      | 2019-11-28T12:23:38.956171Z 0 [Note] InnoDB: Initializing buffer pool, total size = 128M, instances = 1, chunk size = 128M
db.sonar.pipeline_1      | 2019-11-28T12:23:38.966360Z 0 [Note] InnoDB: Completed initialization of buffer pool
db.sonar.pipeline_1      | 2019-11-28T12:23:38.968507Z 0 [Note] InnoDB: If the mysqld execution user is authorized, page cleaner thread priority can be changed. See the man page
db.sonar.pipeline_1      | 2019-11-28T12:23:39.829584Z 0 [Note] InnoDB: Highest supported file format is Barracuda.
db.sonar.pipeline_1      | 2019-11-28T12:23:57.371462Z 0 [Note] InnoDB: Creating shared tablespace for temporary tables
db.sonar.pipeline_1      | 2019-11-28T12:23:57.371530Z 0 [Note] InnoDB: Setting file './ibtmp1' size to 12 MB. Physically writing the file full; Please wait ...
db.sonar.pipeline_1      | 2019-11-28T12:23:57.959365Z 0 [Note] InnoDB: File './ibtmp1' size is now 12 MB.
db.sonar.pipeline_1      | 2019-11-28T12:23:57.965745Z 0 [Note] InnoDB: 96 redo rollback segment(s) found. 96 redo rollback segment(s) are active.
db.sonar.pipeline_1      | 2019-11-28T12:23:57.965761Z 0 [Note] InnoDB: 32 non-redo rollback segment(s) are active.
db.sonar.pipeline_1      | 2019-11-28T12:23:57.966036Z 0 [Note] InnoDB: page_cleaner: 1000ms intended loop took 18998ms. The settings might not be optimal. (flushed=0 and evicted
db.sonar.pipeline_1      | 2019-11-28T12:23:57.966094Z 0 [Note] InnoDB: 5.7.24 started; log sequence number 21099000
db.sonar.pipeline_1      | 2019-11-28T12:23:57.966195Z 0 [Note] InnoDB: Loading buffer pool(s) from /var/lib/mysql/ib_buffer_pool
db.sonar.pipeline_1      | 2019-11-28T12:23:57.966349Z 0 [Note] Plugin 'FEDERATED' is disabled.
db.sonar.pipeline_1      | 2019-11-28T12:23:58.549053Z 0 [Note] Found ca.pem, server-cert.pem and server-key.pem in data directory. Trying to enable SSL support using them.
db.sonar.pipeline_1      | 2019-11-28T12:23:58.825378Z 0 [Warning] CA certificate ca.pem is self signed.
db.sonar.pipeline_1      | 2019-11-28T12:23:59.026487Z 0 [Note] Server hostname (bind-address): '*'; port: 3306
db.sonar.pipeline_1      | 2019-11-28T12:23:59.066611Z 0 [Note] IPv6 is available.
db.sonar.pipeline_1      | 2019-11-28T12:23:59.066657Z 0 [Note]   - '::' resolves to '::';
db.sonar.pipeline_1      | 2019-11-28T12:23:59.066687Z 0 [Note] Server socket created on IP: '::'.
db.sonar.pipeline_1      | 2019-11-28T12:23:59.360100Z 0 [Warning] Insecure configuration for --pid-file: Location '/var/run/mysqld' in the path is accessible to all OS users. Co
db.sonar.pipeline_1      | 2019-11-28T12:23:59.687511Z 0 [Note] InnoDB: Buffer pool(s) load completed at 191128 12:23:59
db.sonar.pipeline_1      | 2019-11-28T12:24:00.289931Z 0 [Warning] 'user' entry 'root@localhost' ignored in --skip-name-resolve mode.
db.sonar.pipeline_1      | 2019-11-28T12:24:00.289997Z 0 [Warning] 'user' entry 'mysql.session@localhost' ignored in --skip-name-resolve mode.
db.sonar.pipeline_1      | 2019-11-28T12:24:00.290023Z 0 [Warning] 'user' entry 'mysql.sys@localhost' ignored in --skip-name-resolve mode.
db.sonar.pipeline_1      | 2019-11-28T12:24:00.290092Z 0 [Warning] 'db' entry 'performance_schema mysql.session@localhost' ignored in --skip-name-resolve mode.
db.sonar.pipeline_1      | 2019-11-28T12:24:00.290108Z 0 [Warning] 'db' entry 'sys mysql.sys@localhost' ignored in --skip-name-resolve mode.
db.sonar.pipeline_1      | 2019-11-28T12:24:00.290137Z 0 [Warning] 'proxies_priv' entry '@ root@localhost' ignored in --skip-name-resolve mode.
db.sonar.pipeline_1      | 2019-11-28T12:24:01.485563Z 0 [Warning] 'tables_priv' entry 'user mysql.session@localhost' ignored in --skip-name-resolve mode.
db.sonar.pipeline_1      | 2019-11-28T12:24:01.485609Z 0 [Warning] 'tables_priv' entry 'sys_config mysql.sys@localhost' ignored in --skip-name-resolve mode.
db.sonar.pipeline_1      | 2019-11-28T12:24:05.717731Z 0 [Note] Event Scheduler: Loaded 0 events
db.sonar.pipeline_1      | 2019-11-28T12:24:05.718040Z 0 [Note] mysqld: ready for connections.
db.sonar.pipeline_1      | Version: '5.7.24'  socket: '/var/run/mysqld/mysqld.sock'  port: 3306  MySQL Community Server (GPL)
app.jenkins.pipeline_1   | Running from: /usr/share/jenkins/jenkins.war
app.jenkins.pipeline_1   | webroot: EnvVars.masterEnvVars.get("JENKINS_HOME")
app.jenkins.pipeline_1   | Nov 28, 2019 12:24:43 PM org.eclipse.jetty.util.log.Log initialized
app.jenkins.pipeline_1   | INFO: Logging initialized @11687ms to org.eclipse.jetty.util.log.JavaUtilLog
app.jenkins.pipeline_1   | Nov 28, 2019 12:24:44 PM winstone.Logger logInternal
app.jenkins.pipeline_1   | INFO: Beginning extraction from war file
app.jenkins.pipeline_1   | Nov 28, 2019 12:24:44 PM org.eclipse.jetty.server.handler.ContextHandler setContextPath
```

Figure 31. Setup of Jenkins & SonarQube pipeline

Then we also configured Jenkins to connect with SonarQube, as depicted in Figure 32.



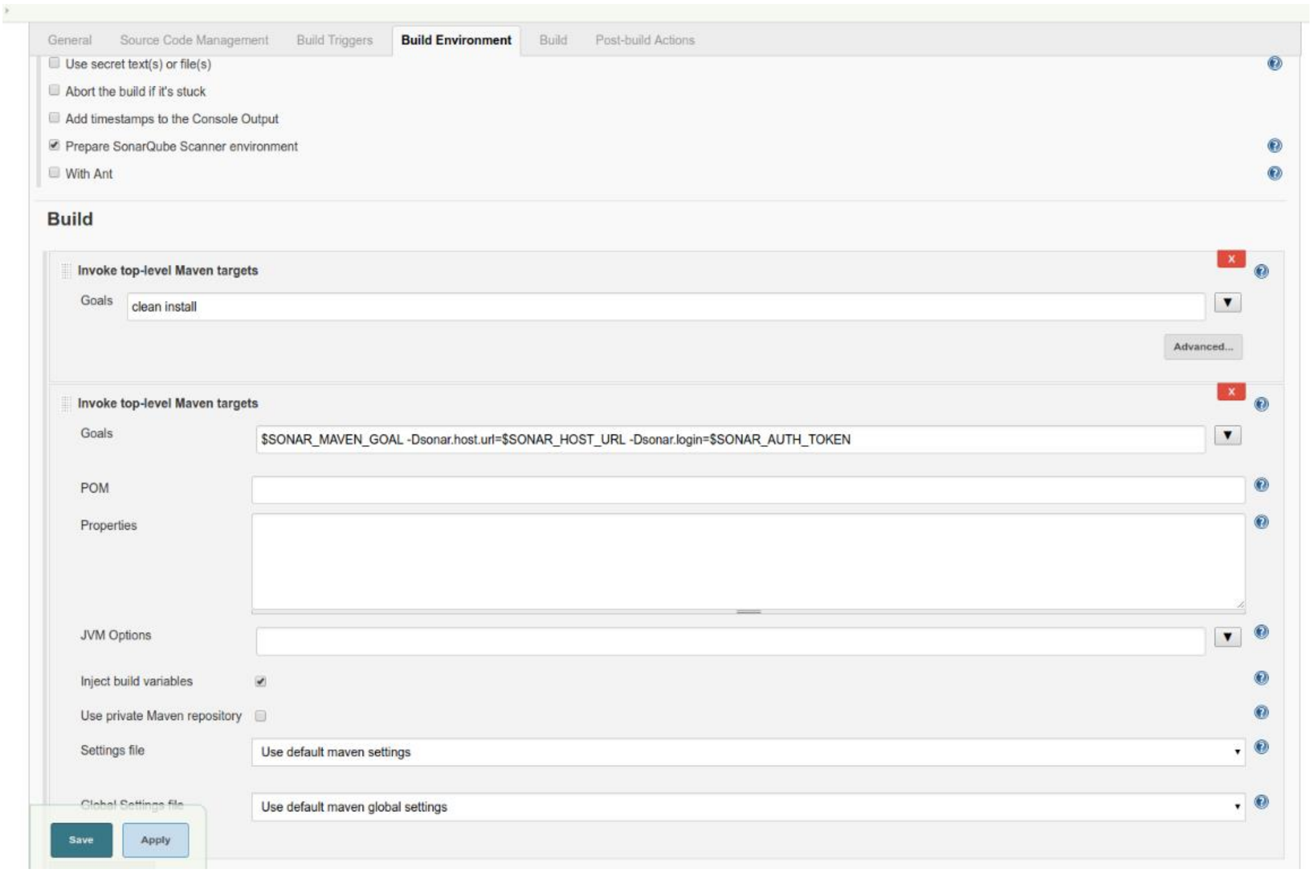
☑ Enable injection of SonarQube server configuration as build environment variables

If checked, job administrators will be able to inject a SonarQube server configuration as environment variables in the build.

Name	sonar
Server URL	http://app.sonar.pipeline:9000
	Default is http://localhost:9000
Server authentication token	••
	SonarQube authentication token. Mandatory when anonymous access is disabled.

Figure 32. Configuring SonarQube on the Jenkins pipeline

Jenkins easily allows to integrate Maven based Java projects to SonarQube, by adding goals, as depicted in Figure 33.



- [ ] Use secret text(s) or file(s)
- [ ] Abort the build if it's stuck
- [ ] Add timestamps to the Console Output
- [x] Prepare SonarQube Scanner environment
- [ ] With Ant

Build

Invoke top-level Maven targets

Goals: clean install

Invoke top-level Maven targets

Goals: $SONAR_MAVEN_GOAL -Dsonar.host.url=$SONAR_HOST_URL -Dsonar.login=$SONAR_AUTH_TOKEN

POM:

Properties:

JVM Options:

Inject build variables: ✓

Use private Maven repository:

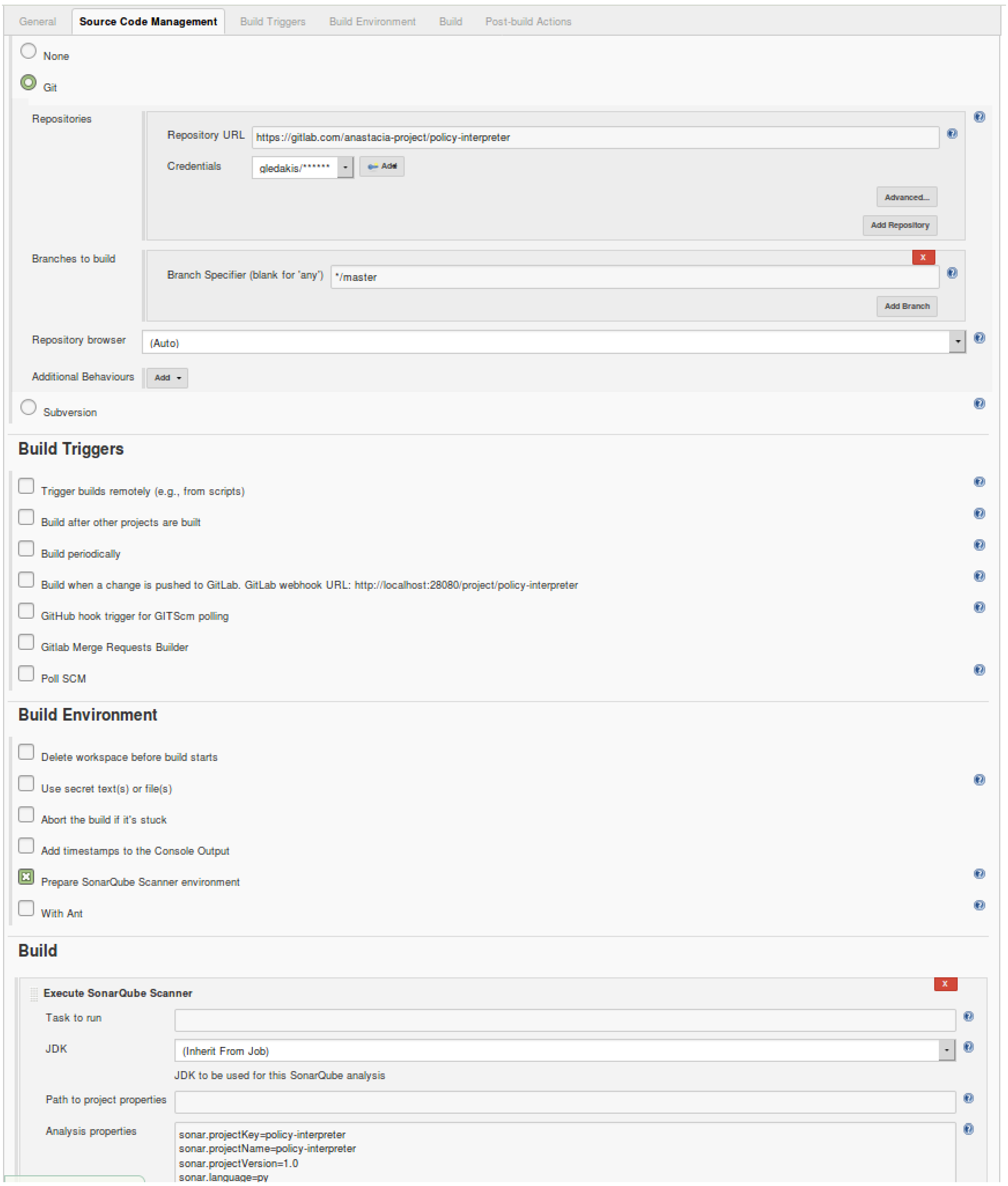
Settings file: Use default maven settings

Global Settings file: Use default maven global settings

Figure 33. Setup of Jenkins & SonarQube pipeline for Maven project

Although Jenkins is ideal for building Maven based Java project as part of the pipeline, ANASTACIA contains many components that have been developed in Python. Python based projects doesn't need to a build process in order to work, however Jenkins and is used in order to configure the SonarQube Analysis. For these two additional tools have been used; SonarScanner[12] and the SonarPython Plugin[13]. It has to be stated that for python components the analysis of SonarQube doesn't include unit testing.

[12] https://docs.sonarqube.org/latest/analysis/scan/sonarscanner/
[13] https://www.sonarsource.com/products/codeanalyzers/sonarpython.html


General | **Source Code Management** | Build Triggers | Build Environment | Build | Post-build Actions

○ None
[✓ ● Git]

Repositories
- Repository URL: https://gitlab.com/anastacia-project/policy-interpreter
- Credentials: gledakis/****** | Add
- Advanced... | Add Repository

Branches to build
- Branch Specifier (blank for 'any'): */master
- Add Branch

Repository browser: (Auto)

Additional Behaviours: Add

○ Subversion

Build Triggers

- ☐ Trigger builds remotely (e.g., from scripts)
- ☐ Build after other projects are built
- ☐ Build periodically
- ☐ Build when a change is pushed to GitLab. GitLab webhook URL: http://localhost:28080/project/policy-interpreter
- ☐ GitHub hook trigger for GITScm polling
- ☐ Gitlab Merge Requests Builder
- ☐ Poll SCM

Build Environment

- ☐ Delete workspace before build starts
- ☐ Use secret text(s) or file(s)
- ☐ Abort the build if it's stuck
- ☐ Add timestamps to the Console Output
- ☒ Prepare SonarQube Scanner environment
- ☐ With Ant

Build

Execute SonarQube Scanner

- Task to run:
- JDK: (Inherit From Job)
  JDK to be used for this SonarQube analysis
- Path to project properties:
- Analysis properties:

```
sonar.projectKey=policy-interpreter
sonar.projectName=policy-interpreter
sonar.projectVersion=1.0
sonar.language=py
```

Figure 34. Configuring Jenkins &SonarQube for Python

With this setup in place we were able to monitor all measures provided by SonarQube with the following being the most important for project development and integration;

- Number of violations against coding guidelines (critical, high, medium…) per component
  - Critical (Bugs)
  - High/Medium (Vulnerabilities)
  - Code Smells

- Technical Dept
- Unit tests success and coverage
- Density of comment lines (Comment lines / (Lines of code + Comment lines) * 100)
- Density of duplication (Duplicated lines / Lines * 100 < 10%)

A sample project analysis of a component is presented in Figure 35 below.

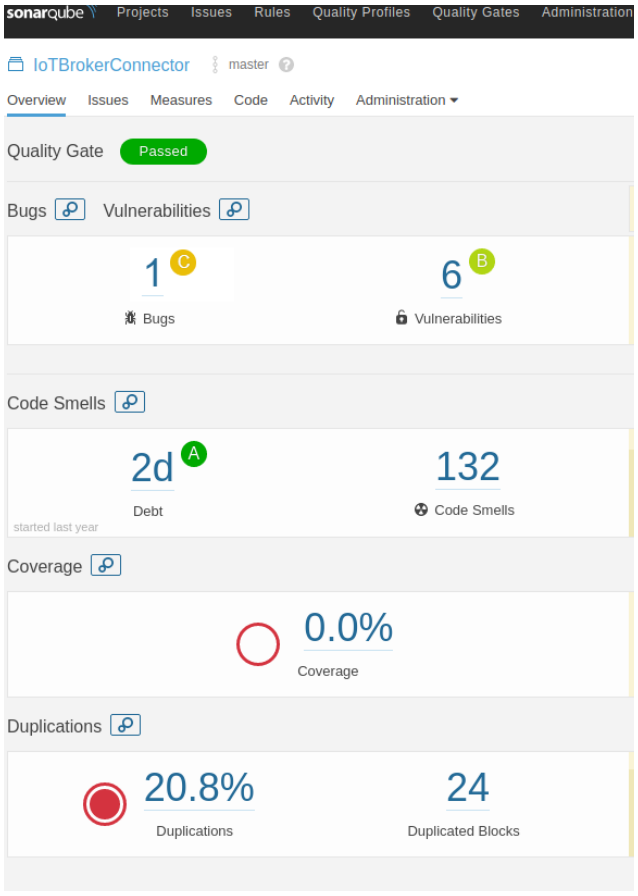


Figure 35. SonarQube Metrics

Due to privacy issues, not all components could be analysed through this Sonar setup. However, the overall project results are the following.

Table 23. Source Code Quality main values overall current status

Component	Value per component
Number of bugs	< than 2 per component
Number of vulnerabilities	<10 per component
Technical Dept	< 10 days per component
Density of comment lines	46% average
Density of duplication	15% average

100% passed unit tests when deliver code for integration	Achieved for all components

Bugs shall be resolved by project duration; the vulnerabilities and technical dept should be taken into account for creating a solution with a higher TRL level (ANASTACIA aims for TRL5 according DoA[3]).

5.2 ANASTACIA Framework Scalability Aspects

Throughout the design, development and integration process ANASTACIA took into account the scalability aspects into account in order to offer a platform that can support large scale CPS/IoT deployments. By using a distributed architecture composed by many components, each of them able to be hosted separately we expect that the application can be adapted to the needs of the actual deployment. In order to further elaborate on the scalability aspects, we examined some of the operational characteristics of the components in order be able to identify the technical limitations enforced by the components' implementation. In the following table we collect all this information.

Table 24. Technical information per component

Component	Is component scalable (with or with a proxy)?	Can more than one user be supported?	Does it use multiple threads or processes?	Amount of data expected	Which database is used?
Policy Editor Tool	Yes	Yes	Yes	None. Data create by the UI	N/A
Interpreter	Yes	Yes	Yes	Minimal amount of data (policies created by the user)	N/A
Policy Conflict Detector	Yes	Yes	Yes	Minimal amount of data (policies created by the user)	N/A
Policy Repository	Yes	Yes	Yes	Minimal amount of data (policies created by the user)	SQL
Security Enablers Provider	Yes	Yes	Yes	Minimal (Number of Enablers is limited)	SQL
Security orchestrator Engine	No	Yes	Yes	Minimal (Mitigations actions are not constantly sent)	N/A

Security Orchestrator Optimizer	No	Yes	Yes	Minimal (Mitigations actions are not constantly sent)	N/A
Monitoring Agents	N/A, it is a distributed agent	N/A	N/A	N/A	N/A
Data Filtering and pre-processing Component	Yes	Yes	Yes	Large Amount of data used. Less than 10 Mbps in current testing setup (due to small size of events), but both Apache Kafka and Apache Storm can scale to great extent[14].	Kafka
Data Analysis Component	Yes	Yes	Yes	Large Amount of data used. Less than 10 Mbps in current testing setup.	N/A
Verdict and Decision Support System	Yes (core engine yes, UI no)	Yes	Yes	Up to 10 Mbps in current setup	mySQL
Mitigation Action Service	Yes	No	Yes (1 thread per alert)	Up to 10 Mbps in current setup	None
Security Alert Service	Yes	Yes	Yes (1 thread per alert)	Up to 10 Mbps in current setup	None
Dynamic Security and Privacy Seal Agent	Yes	N/A	No	Minimal	None
Security and Privacy Manager Analysis	Yes	Yes	Yes	Up to 10 Mbps in current setup	Redis (cache)

[14] https://engineering.linkedin.com/kafka/benchmarking-apache-kafka-2-million-writes-second-three-cheap-machines

Dynamic Security and Privacy Seal User Interface	Yes	Yes	Yes	Up to 10 Mbps in current setup	Redis (cache), Mongo DB
System Model Service	No	Yes	Yes	Minimal	Filesystem
Assets Model	No	Yes	Yes	Minimal	Filesystem
Security Resource Planning	No (same with assets model)	Yes (same with assets model)	Yes (same with assets model)	Minimal (same with assets model)	Filesystem (same with assets model)
Incident Detector	Yes	Yes	Yes	Up to 10 Mbps in current setup	mySQL
Performance Data Analytics	Yes	Yes	Yes	Up to 10 Mbps in current setup	N/A
Security Sensors	It is a distributed sensor network	N/A	N/A	N/A	N/A

5.3 Testing for the Integrated Platform

Integration testing is the phase in software testing in which individual software modules are combined and tested as a group. The main idea of integration testing is to start from two components to test the interface between them. In some cases, more than two components can participate in a common test.

The goal of integration testing is to identify problems that occur when components are combined. By using a test plan that suggests the usage of tests (unit test or manual tests) before combining components, the errors discovered when in integration tests are most probably related to the interface between them. This method reduces the number of possibilities of errors to a far simpler level of analysis. In D6.1[1] a number of tests were defined and in some extend tested. In this deliverable we provide updated versions of the tests and check how the actual tests have been performed. Furthermore, in the ANASTACIA deployment for the Y3 demo, the integration of the components is further tested by dedicated test cases; the tests presented here however helped to identify issues early and proceed further with the test cases that are covered in deliverables D6.2[11] /D6.5[7] .

5.3.1 Integration Tests

In the following tables the tests that have used in order to cover the integration aspects of the project, are presented. All tests are successfully tested.

Table 25. Integration Test for SDN, NFV, the IoT controllers

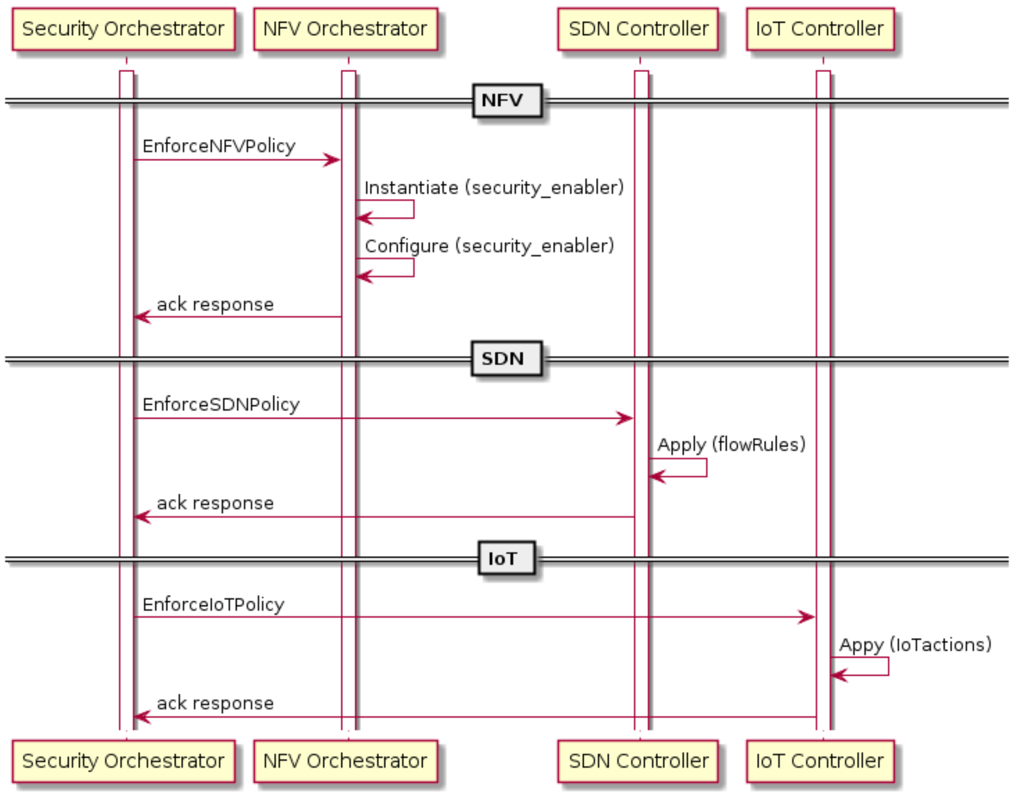
Tests	Enforce-NFV, Enforce-SDN, Enforce-IoT	
Interface(s) Tested	**Components Used**	**Short Description**
SDNI, NFVI, IOIT	Security orchestrator, SDN	Testing of the proper functioning of controllers
The security orchestrator is responsible for enforcing relevant security policies in the data plane. This can be either using SDN, NFV, the IoT controller or a combination of those. For the NFV part, the security orchestrator makes sure that the security enablers are up and configured by interacting with the NFV orchestrator. For SDN, using the northbound API, receiving an acknowledgment of the request means that the security policy has been properly enforced. The same is applicable for the IoT mitigation actions (The reception of the acknowledgement confirms the policy enforcement). 		

Table 26. Integration Test for IoT data retrieval

Test	SubscribeData-from-IoT-network	
Interface(s) Tested	**Components Used**	**Short Description**
	IoT broker and Data Filtering and pre-processing Component	Testing of the data exchange of sensors values and attack notifications

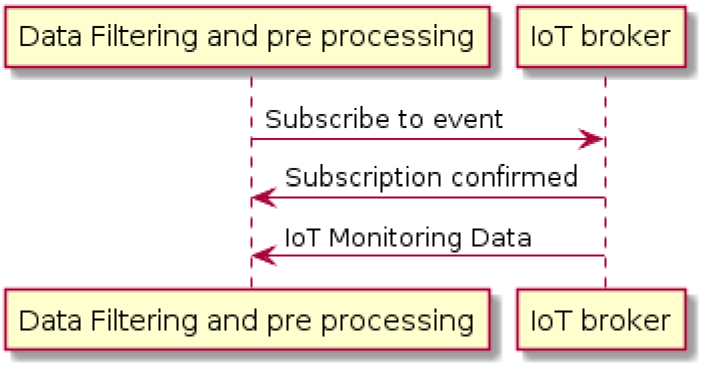
The Data Filtering and pre-processing Component subscribes to the IoT broker to receive new data from sensors. IoT Broker return subscription response. This response is used to exchange new data between IoT broker and Data Filtering and pre-processing Component.  Data Filtering and pre processing IoT broker Subscribe to event Subscription confirmed IoT Monitoring Data Data Filtering and pre processing IoT broker

Table 27. Test for Data Analysis integration

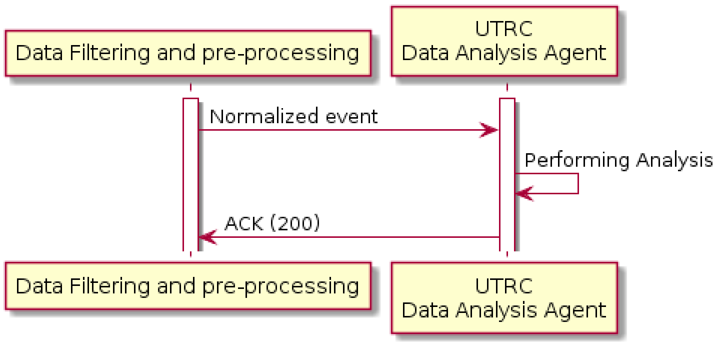
Test		
Interface(s) Tested	**Components Used**	**Short Description**
Kafka based interface between the Data Filtering component and the Data analysis component	Data analysis and Data Filtering pre-processing	Testing that the security event are correctly normalized when received from the Data Filtering and pre-processing
The Data Filtering component sends a probe event to the Data analysis including the basic information: source of attacked devices, destination of the attack, type of attack, timestamp.  Data Filtering and pre-processing UTRC Data Analysis Agent Normalized event Performing Analysis ACK (200) Data Filtering and pre-processing UTRC Data Analysis Agent		

Table 28. Testing if VDSS receives Data Analysis alerts

Test		
Interface(s) Tested	**Components Used**	**Short Description**
MVI	Data Analysis and Verdict and Decision Support system	Testing that security events are correctly correlated and alarms are generated

The Data Analysis correlate events that derive into the generation of alarms. The test uses a simple rule set at the Decision Support system that triggers a simple alarm for a single event sent from the Data Analysis.

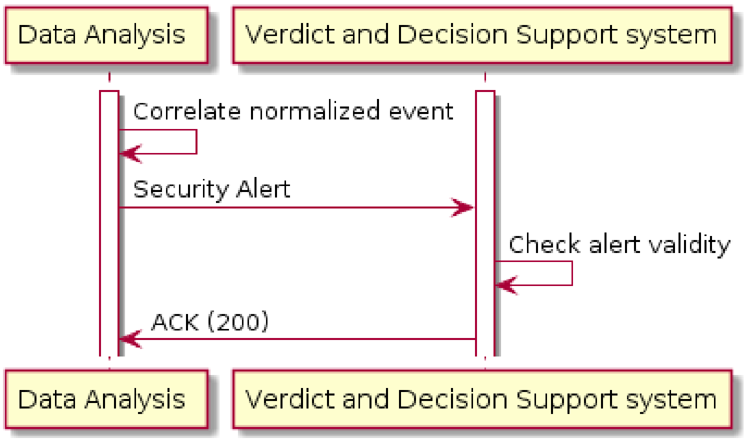

Data Analysis
Verdict and Decision Support system
Correlate normalized event
Security Alert
Check alert validity
ACK (200)
Data Analysis
Verdict and Decision Support system

Table 29. Testing proper triggering of reactions

Test		
Interface(s) Tested	**Components Used**	**Short Description**
CSI	Mitigation Action Service and Security Orchestrator	Testing that the information required to trigger a reaction is correct.

The Mitigation Action Service creates a reaction message describing the incident to mitigate and the reaction chosen. The test consists in checking that all the information required to enforce the mitigation is correctly included in the reaction message.

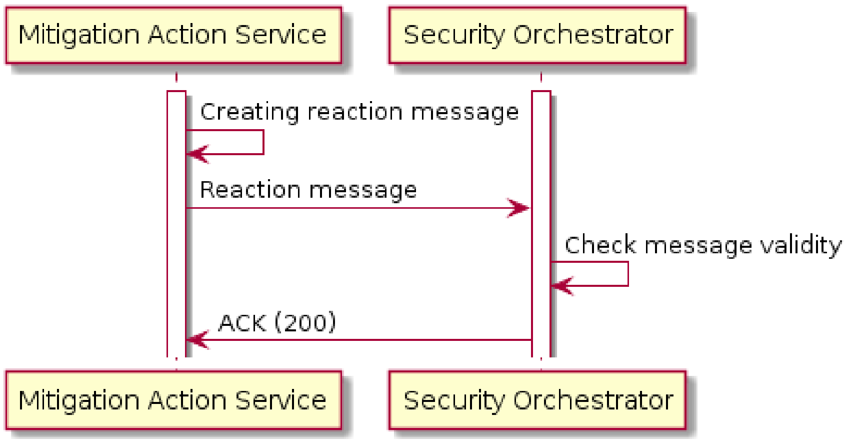


Table 30. Testing of alert visualization

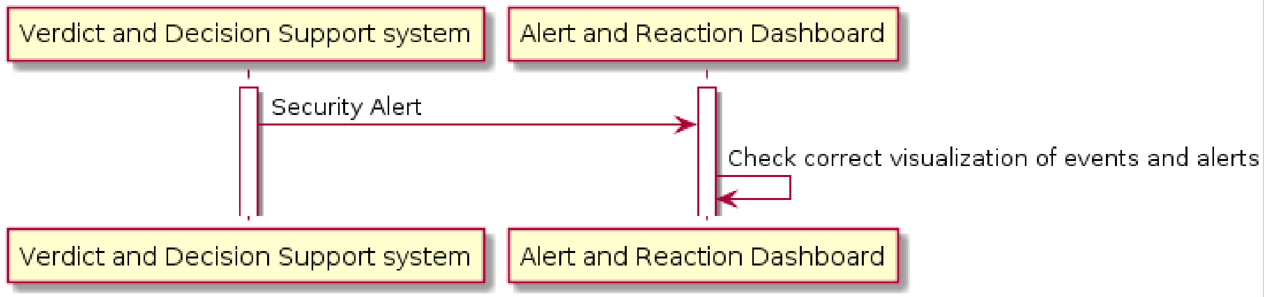
Test		
Interface(s) Tested	**Components Used**	**Short Description**
SAWI	Verdict and Decision Support system and Alert and Reaction Dashboard	The test will show a simple event and a simple alert in a dashboard at the system admin side.
The incidents detected and the event received by the monitoring module will be displays in a GUI. A simple event is sent to check the correct visualization of the events.  Verdict and Decision Support system Alert and Reaction Dashboard Security Alert Check correct visualization of events and alerts Verdict and Decision Support system Alert and Reaction Dashboard		

Table 31. Testing of MMT monitoring data full circle

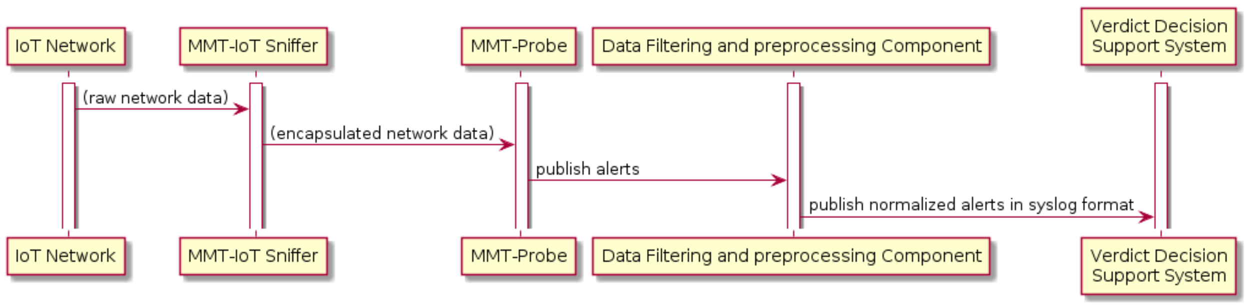
Test	MMT Integration test (Monitored Data General Interface – MDGI)	
Interface(s) Tested	**Components Used**	**Short Description**
	IoT Network, **MMT-IoT Sniffer**, MMT-Probe, MMT-Security, Data Filtering and preprocessing Component	Testing of the integration of MMT solution into the ANSTACIA Platform, using the Monitoring Data General Interface
 IoT Network MMT-IoT Sniffer MMT-Probe Data Filtering and preprocessing Component Verdict Decision Support System (raw network data) (encapsulated network data) publish alerts publish normalized alerts in syslog format The general ANASTACIA Architecture establishes that the data should be sent from the Agents to the Data Filtering and Pre-processing component, to be sent from this module to the analysis tool. This approach utilizes MMT's capability to use Kafka Brokers to publish the information extracted by MMT-Probe (using DPI techniques), making it available to all analysis tools. In particular, the MMT-Security uses this information from the Kafka Channel in order to test the security properties. The verdicts of these analyses are also published in the Kafka Broker in order to be used by VDSS.		

Table 32. Testing events receiving by the DSPS

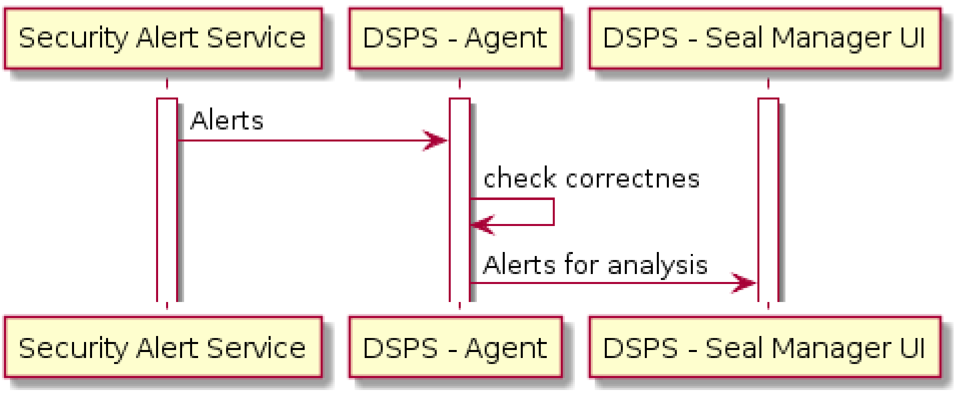
Test		
Interface(s) Tested	**Components Used**	**Short Description**
SMMI	Security Alert Service, Dynamic Security and Privacy Seal	The test will display the different kinds of data provided by the modules developed in the frame of WP4.
The detected threats and the corresponding events transmitted from WP modules to the DSPS server should be shown Sin the DSPS GUI. The test is separated in several parts, depending on the source of the data, their format and the information contained. This test includes the validation of the data described using STIX/TAXII.  Security Alert Service DSPS - Agent DSPS - Seal Manager UI Alerts check correctnes Alerts for analysis Security Alert Service DSPS - Agent DSPS - Seal Manager UI		

Table 33. Testing Policy Editor Tool

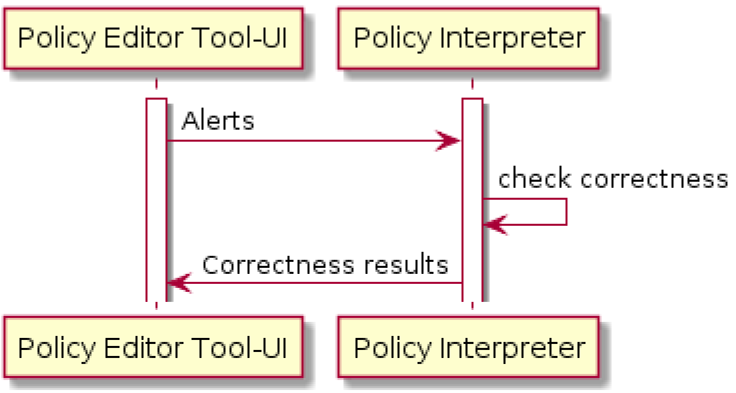
Test	HSPL policy definition	
Interface(s) Tested	**Components Used**	**Short Description**
/editor	Policy Editor Tool	Testing the proper HSPL definition from the Policy Editor Tool
To instantiate different HSPL Orchestration Policies through the Policy Editor Tool and verify the file generated is compliant with the HSPL scheme.  Policy Editor Tool-UI Policy Interpreter Alerts check correctness Correctness results Policy Editor Tool-UI Policy Interpreter		

Table 34. Testing policy refinement

Test	HSPL policy refinement	
Interface(s) Tested	**Components Used**	**Short Description**
H2MI	**Policy Interpreter**, Security Enabler Provider	Testing the proper HSPL refinement

To perform HSPL policy refinements, verifying:

1. Non-enforceable results
2. Enforceable-results, verifying MSPL Orchestration Policies generated are correct and compliant with the MSPL scheme.

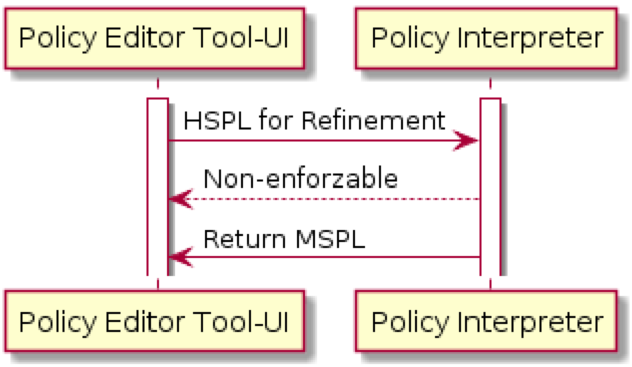
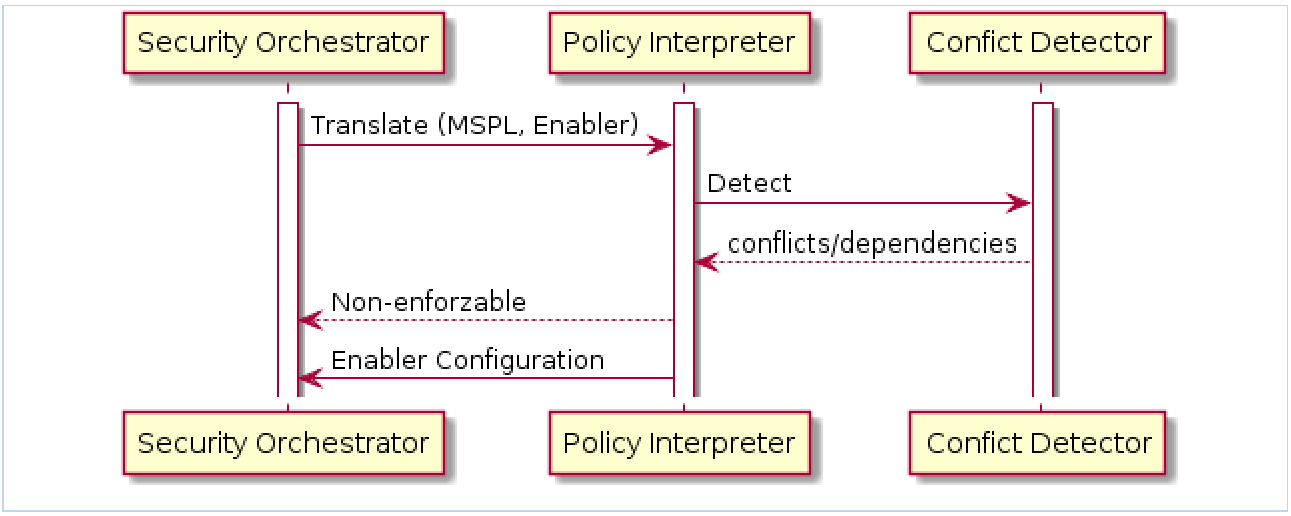


Table 35. Testing policy translation

Test	MSPL policy translation	
Interface(s) Tested	**Components Used**	**Short Description**
M2LI MCDT	**Policy Interpreter**, Policy Conflict Detector, Security Enabler Provider, Security Orchestrator	Testing the proper MSPL translation into a security enabler configuration as well as policy conflicts and dependencies detection.

To perform MSPL policy translations for the main identified security enablers, verifying:

1. Non-enforceable results
2. Enforceable-results, verifying the generated configuration is correct for the specific technology as well as the conflicts and dependencies are properly identified.


Security Orchestrator
Policy Interpreter
Confict Detector
Translate (MSPL, Enabler)
Detect
conflicts/dependencies
Non-enforzable
Enabler Configuration
Security Orchestrator
Policy Interpreter
Confict Detector

6 Conclusions

This document presented the final technical integration of ANASTACIA framework, as planned and executed in the scope of WP6, along with the technical testing and validation outcomes. First, we provided an updated view on ANASTACIA architecture and the interfaces between the components. Both the ANASTACIA architecture and the interfaces definition was a continuous task in order to keep up to date and adapt the design according to newly identified needs and the same time to ease the overall development. In addition, we provided a short overview on the status of the components developed and deployed for demonstration purposes.

Finally, an important aspect of this document is the validation reporting. Based on the analysis and plan that was done in the first months of WP6 and documented in D6.1, in this deliverable we present the validation results. As part of the development lifecycle defined in D6.1 we presented the setup of the pipeline for Jenkins and SonarQube and the overall quality results. Furthermore, we tried to analyse the components in order to find possible bottlenecks. According to the analysis made most components are not expecting high amounts of loads, while the ones responsible for dealing with the highest load (such as the Data Filtering and pre-processing Broker) are scalable by design, therefore can be used even in large scale deployments. Finally, we covered the integration tests that have been defined and used in order to ensure the proper integration between the components. This was a critical step in order to early identify issues and ease the execution of the test cases that were planned to be performed in the demonstration site.

References

[1] ANASTACIA Deliverable D6.1 – Initial Technical Integration and Validation Report, 2018

[2] ANASTACIA Deliverable D1.3 - Initial Architectural Design, 2017

[3] ANASTACIA Grant Agreement N°731558 – Annex I (Part A) – Description of Action, 2016

[4] ANASTACIA Deliverable D1.2 – User-centred Requirement Initial Analysis, 2017

[5] ANASTACIA Deliverable D1.4 - Final User-centred Requirement Analysis, 2018

[6] https://www.confluent.io/blog/apache-kafka-for-service-architectures/

[7] ANASTACIA Deliverable D6.5-Final Use cases implementation and tests Report, 2019

[8] ANASTACIA Deliverable D5.3 - Dynamic Security and Privacy Seal User Interface, 2019

[9] ANASTACIA Deliverable D2.8 - Secure Software Development Guidelines Final Report, 2019

[10] ANASTACIA Deliverable D6.3 - Initial End-user validation and evaluation Report, 2018

[11] ANASTACIA Deliverable D6.2 - Initial Use cases implementation and tests Report, 2018