ANASTACIA

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

D1.5

Final Architectural Design

This deliverable presents the final results of task 1.3. This task is in charge of designing the ANASTACIA architecture. This deliverable includes the final version of the ANASTACIA architecture, with details of the components included, the main processes that the ANASTACIA framework supports and details about the main interfaces.

Distribution level	PU
Contractual date	31.12.2019 [M36]
Delivery date	31.12.2019 [M36]
WP / Task	WP1/T1.3
WP Leader	Atos
Authors	Ruben Trapero (ATOS), Diego Rivera (MONT), Miloud Bagaa, Boukhalfa Mohammed, Tarik Taleb, Mada Badr, Nadir Zinelaabidine, Abada Abderrahmane (AALTO), Dallal Belabed (THALES), Eunah Kim, Robert Narvaez (DG) (MAND), Antonio Skarmeta, Jorge Bernal, Alejandro Molina, Jordi Ortiz (UMU), Rafael Marín Pérez (ODINS), Satyanarayana Vuppala (UTRC), Stefano Bianchi (SOFT), Ivan Vaccari, Enrico Cambiaso (CNR), Eunah Kim, Robert Narvaez (DG)
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

PUBLIC SUMMARY 5
1 Introduction 6
1.1 Aims of the document 6
1.2 Applicable and reference documents 6
1.3 Revision History 6
1.4 Acronyms and Definitions 8
2 Design methodology for second revision 9
3 ANASTACIA final activity models 11
3.1 Policy Management 11
3.1.1 Policy set-up 11
3.1.2 Policy enforcement 12
3.2 Monitoring activities 13
3.2.1 Monitoring data acquisition 13
3.2.1.1 Monitoring by MMT probe 13
3.2.1.2 Monitoring by CNR probes 14
3.2.1.3 Data Analysis 15
3.2.1.4 MiTM agent – adversary emulator 16
3.2.2 Incident detection 17
3.3 Reaction activities 18
3.3.1 Decision Support 18
3.3.2 Reaction enforcement 19
3.4 Dynamic Security and Privacy Seal 20
3.4.1 Seal Creation 20
3.4.2 DSPS Agent 21
3.4.3 Secured storage 21
4 ANASTACIA final architecture 22
4.1 Global view 22
4.2 Monitoring and Reaction 23
4.2.1 Monitoring components 23
4.2.1.1 Incident detector 23
4.2.1.2 Data Filtering and pre-processing Broker 24
4.2.1.3 Data Analysis 25
4.2.1.4 Resource and QoS Monitoring 26
4.2.2 Reaction components 27

4.2.2.1 Verdict and Decision Support System 27
4.2.2.2 Assets Model 28
4.2.2.3 Security Alert Service 29
4.2.2.4 Mitigation Action Service 30
4.3 Orchestration and Enforcement 30
4.3.1 Security Orchestrator Engine 30
4.3.2 Policy interpreter – M2L Translator 31
4.3.3 Policy conflicts and dependencies detector 32
4.3.4 Security Orchestrator Optimizer 33
4.3.5 Performance Data Analytics 34
4.3.6 Security Resource Planning 34
4.3.7 System Model Service 35
4.3.8 IoT, SDN, NFV Drivers 35
4.4 Dynamic Security and Privacy Seal 36
4.4.1 DSPS Server 37
4.4.2 Dynamic Security & Privacy Seal Agent 37
4.4.3 Dynamic Security & Privacy Seal User Interface 38
5 Main interfaces 40
5.1 Reaction – Enforcement 40
5.2 Monitoring – Enforcement 41
5.3 Monitoring and Reaction – DSPS 41
5.4 Orchestration – IoT infrastructure 42
6 Conclusions 44
References 45

Index of figures

Figure 1. Refinement architecture procedure 9
Figure 2. High level ANASTACIA framework (extended from D1.3) 10
Figure 3. ANASTACIA activities related to the ANASTACIA planes 11
Figure 4. Policy set-up model 12
Figure 5. Policy Enforcement model 12
Figure 6. Monitoring flow followed by MMT-Probe 13
Figure 7. An overview of a remote AT command IoT attack detection approach 14
Figure 8. Monitoring flow by CNR probe 15
Figure 9. Monitoring process inside UTRC Data analysis agent. 15

Figure 10. MiTM attack process inside UTRC MiTM agent 17
Figure 11. Complete final monitoring process 18
Figure 12. Complete final decision support process 19
Figure 13. Reaction Enforcement process 20
Figure 14. DSPS overview process 20
Figure 15. DSPS event processing 21
Figure 16. ANASTACIA final architecture design 22
Figure 17. Incident Detector internal architecture 23
Figure 18. Data Filtering and pre-processing internal architecture 24
Figure 19. Data analysis agent internal monitoring architecture 25
Figure 20. Resource and QoS Monitoring (RQM) interaction with SOE and SM components 26
Figure 21. Verdict and Decision Support System internal architecture 27
Figure 22. THALES Asset Model internal architecture 28
Figure 23. Security Alert Service internal design 29
Figure 24. High-level design of the Mitigation Action Service 30
Figure 25. Security Orchestrator Engine (SOE) interaction with different security enablers 31
Figure 26. M2L Translator high-level design 32
Figure 27. Conflict detector high-level design 32
Figure 28 Security Orchestrator Optimizer (SOO) interaction with different security orchestration components 33
Figure 29. Security Resource Planning Module Interactions with the Security Orchestrator 34
Figure 30. System Model Data Structure [8] 35
Figure 32. Internal architecture of the DSPS Server 37
Figure 33. Internal architecture of the DSPS agent 38
Figure 34. Internal architecture of the DSPS User Interface 38
Figure 35. Main interfaces represented over the high level architecture 40

Index of tables

Table 1. Document revision history 6
Table 2. List of acronyms 8
Table 3. Incident Detector details 23
Table 4. Data Filtering and pre-processing Broker details 24
Table 5. Data Analysis details 25
Table 6. Resource and QoS Monitoring details 26
Table 7. Verdict and Decision Support System details 27
Table 8. Assets Model details 28
Table 9. Security Alert Service details 29

Table 10. Mitigation Action Service details 30
Table 11. Mitigation Action Service - Security Orchestrator Engine Interface detail 40
Table 12. Interface Resource and QoS Monitoring and Security Orchestrator Engine 41
Table 13. Data Analysis and Security Orchestrator Engine interface detail 41
Table 14. Security Alert Service to DSPS agent interface detail (CNR, MAND) 41
Table 15. SDN/NFV drivers and control management domain interface detail 42
Table 16. IoT Controller Driver - Control and Management Domain interface detail 42

PUBLIC SUMMARY

This deliverable includes the final design of the ANASTACIA architecture along with additional details related to the complete ANASTACIA framework. The contents reported in this deliverable evolve from D1.3 [1] , delivered in M9 where an early design of the ANASTACIA architecture was provided. The progress on the work carried out from M9 to the current deliverable has fed from different sources, namely the feedback received from the EC reviews, from the Advisory Board, and the continuous feedback from the development activities. The ANASTACIA architecture has gone through nine revisions since the initial report in D1.3 to the latest version reported in this deliverable. The result is a consolidated and consistent ANASTACIA architecture capable of dealing with the main processes identified: monitoring, orchestration, reaction, policy management and security and privacy seal generation.

This deliverable is structured in three main parts:

- Detail of the main processes supported by the ANASTACIA framework: set-up and enforcement of policies, acquisition of monitoring data, incident detection, mitigation decision support, reaction enforcement and dynamic security and privacy seal management.
- Details of the ANASTACIA final architecture, including the global overview of the complete architecture and details of the individual components that are part of it.
- Details of the main interfaces that enable the interaction between components belonging to different planes of the ANASTACIA architecture.

1 INTRODUCTION

1.1 AIMS OF THE DOCUMENT

This document describes design insights followed to produce the final ANASTACIA architecture. To document describes the most important aspects of the design process that have derived into the final architecture, detailing the sources used (feedbacks received, outcomes from other tasks, etc), the activities covered by the framework, the components that compose the architecture and their interfaces.

1.2 APPLICABLE AND REFERENCE DOCUMENTS

The following documents have been used as reference in this deliverable.

From WP1:

- D1.4 Final User-centred Requirement Analysis [2]

WP2 deliverables, especially the final reports of WP2 tasks:

- D2.5 Policy-based Definition and Policy for Orchestration Final Report [3]
- D2.6 Attack Threats Analysis and Contingency Actions Final Report [4]
- D2.7 Privacy Risk Modelling and Contingency Final Report [5]
- D2.8 Secure Software Development Guidelines Final Report [6]

WP3 deliverables, especially the final reports of WP3 tasks:

- D3.4 Final Security Enforcement Manager Report [7]
- D3.5 Final Security Orchestrator Report [8]
- D3.6 Final Virtual Resources Manager Report [9]

WP4 deliverables, especially the final reports of WP4 tasks:

- D4.4 Final Monitoring Components Services Implementation Report [12]
- D4.5 Final Reaction Component Services Implementation Report [13]
- D4.6 Final Agents Development Report [14]

From WP5

- D5.1 Dynamic Security and Privacy Seal Model Analysis Report [15]

From WP6, the final reports of WP6 tasks:

- D6.4 Final Technical integration and validation Report [18]
- D6.5 Final Use cases implementation and tests Report [19]
- D6.6 Final End-user validation and evaluation Report [20]

1.3 REVISION HISTORY

Table 1. Document revision history

Version	Date	Author	Description
0.1	23.9.19	Ruben Trapero (ATOS)	ToC
0.2	24.9.19	Ruben Trapero (ATOS)	Section 1.1 Section 1.2 and Section 2

0.3	2.10.19	Ruben Trapero (ATOS)	Section 3.2.2, Section 3.3.1, Section 4.1, Section 4.2.1.1, Section 4.2.2.1
0.4	10.10.19	Alejandro Molina (UMU)	Section 3.1.1, Section 3.1.2, Section 5.3
0.5	11.10.19	Ruben Trapero (ATOS)	Public summary, front page
0.6	15.10.19	Diego Rivera (MONT)	Section 3.2.1.1, Section 3.3.2, Section 4.2.2.4, Section 5.1
0.7	15.10.19	Ivan Vaccari (CNR)	Section 3.2.1.2, Section 4.2.2.3
0.8	21.10.19	Diego Rivera (MONT)	Extended Section 5.1
0.9	22.10.19	Satyanarayana Vuppala (UTRC)	Section 3.2.1.3
0.10	24.10.19	Ruben Trapero (ATOS)	Integration: split content in section 3.2.1.3 Updated architecture figure Change structure of interfaces in section 5
0.11	12.11.19	Ruben Trapero (ATOS)	Added first version of Section 6
0.12	13.11.19	Piotr Sobonski (UTRC)	Updates and content augmentation based on review comments for sections 3.2.1.3, 3.2.1.4, 4.2.1.3, 4.2.2.2, 5.2.
0.13	13.11.19	Federico Sismondi (MI), Robert Narvaez (DG)	Section 3.4, Section 4.4
0.14	21.11.19	Dallal Belabed (THALES)	Added section 4.3.4
0.15	22.11.19	Giannis Ledakis (UBITECH) Eunah Kim (MANDAT)	Added section 4.2.1.2 by UBITECH
0.16	28.11.19	Ivan Vaccari (CNR)	Filled Table 15
0.17	29.11.19	Dallal Belabed (THALES) Miloud Bagaa (AALTO)	Filled Section 4.3.4 by Thales Edited Sections 3.3.2, 4.2.1.4, 4.3, 5.2 and Table 16 by AALTO
0.18	10.12.19	Ruben Trapero (ATOS)	References
0.19	16.12.19	Ruben Trapero (ATOS)	Edits based on IAB feedback
0.20	18.12.19	Alejandro Molina (UMU)	Internal review
0.21	18.12.19	Enrico Cambiaso (CNR) Ruben Trapero (ATOS)	Edits based on internal review

0.22	19.12.2019	Miloud Bagaa (AALTO) Boukhalfa Mohammed (AALTO) Tarik Taleb (AALTO) Mada Badr (AALTO) Nadir Zinelaabidine (AALTO) Abada Abderrahmane (AALTO)	Edited Sections 3.3.2, 4.2.1.4, 4.3, 5.2 and Table 16
1.0	20.12.2019	Ruben Trapero (ATOS)	Final version ready for submission

1.4 Acronyms and Definitions

Table 2. List of acronyms

Acronym	Meaning
DSPS	Dynamic Security and Privacy Seal
GDPR	General Data Protection Regulation
HSPL	High Security Policy Language
IoT	Internet of Things
MANO	Management and Orchestration
MMT	Montimage Monitoring Tool
MSPL	Medium Security Policy Language
NFV	Network Function Virtualization
SDN	Software Defined Networking
SIEM	Security Information and Event Management

2 DESIGN METHODOLOGY FOR SECOND REVISION

The design methodology followed in ANASTACIA to produce the final architecture feeds from several sources and departs from the initial architecture produced in M9 and reported in D1.3. This process is depicted in Figure 1. Firstly, there has been a continuous feedback between the development activities and the revision of the architecture. At the same time, additional sources have taken part in this redesign process. Firstly, latest deliverables from WP1 added additional inputs in terms of latest requirements and latest studies about new threats and attacks. The feedback received by the EC through the intermediate and technical reviews, and the received from the Advisory Board, was considered for the several revisions of the architecture that have been created between the first architecture reported in M9 and the latest one reported in this deliverable. In total 9 revisions of the architecture have been produced, very often including just minor changes and sometimes with a major revision of some of the modules. In parallel to such revisions, and also used as feedback for them, it has been carried out the development activities that have resulted in the prototypes produced during the whole project period. This has been possible due to the flexibility of the first design of the ANASTACIA architecture, with flexibly designed interfaces between modules that allowed to extend, when required, the functionalities provided by every module.

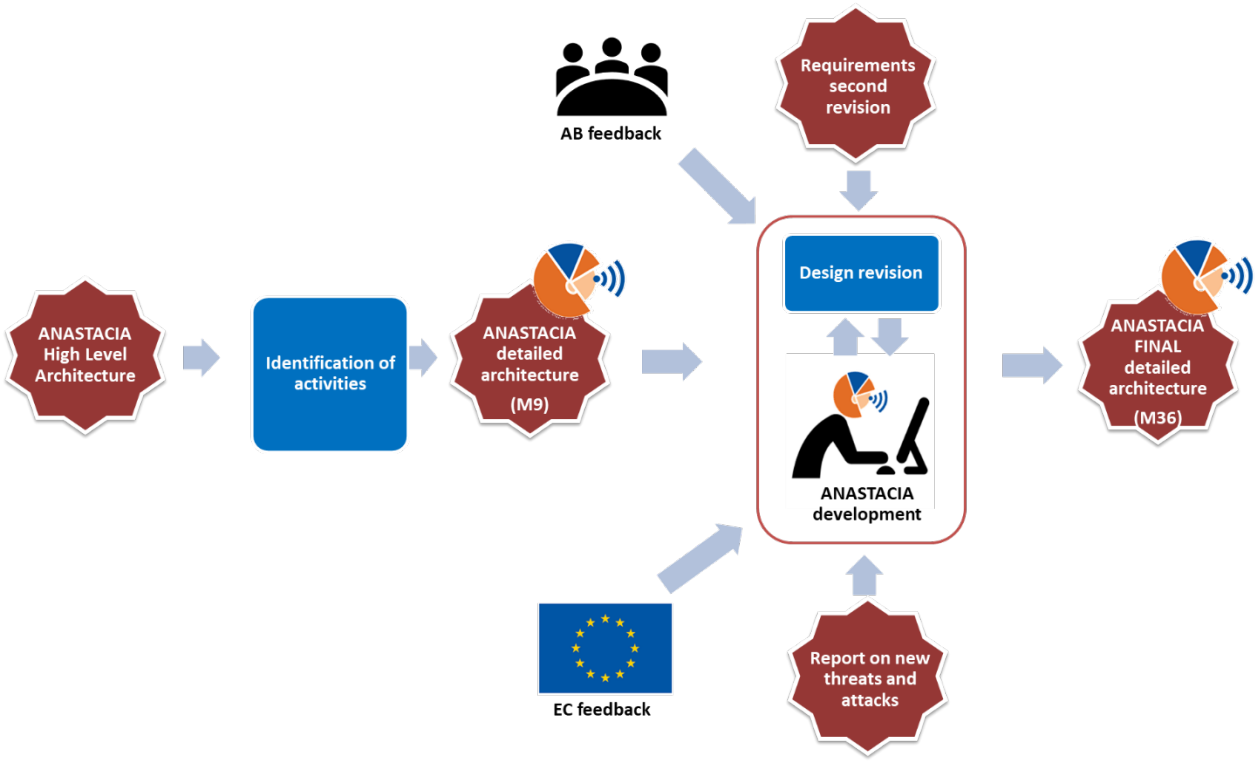


Figure 1. Refinement architecture procedure

The architecture of the ANASTACIA framework has evolved significantly during the second half of the project and with respect to the architecture designed and reported in D1.3 in M9. It is worth noticing that the high-level view of the ANASTACIA framework, as reported in D1.3 and attached here in Figure 2, remains mostly the same. Administration posibilities have been added from the User Plane by the CISO (Chief Information Security Officer), to configure DSPS at the Seal Management Plan, components for monitoring and reaction at the Monitoring and Reaction plane and Policy editors at the Security Orchestration Plane.

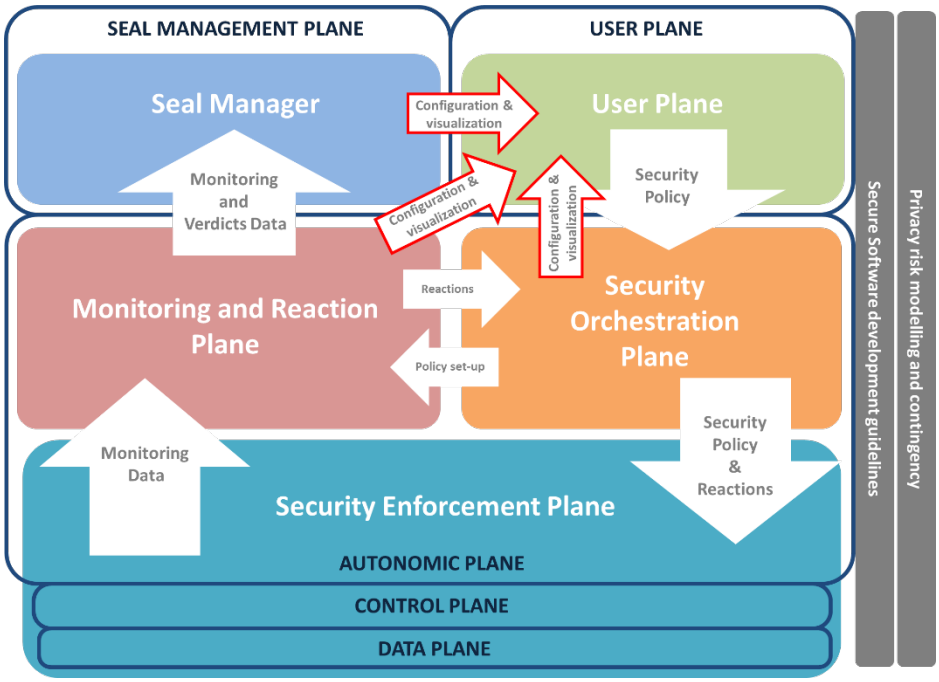


Figure 2. High level ANASTACIA framework (extended from D1.3)

The following sections describe different aspects of the ANASTACIA platform, all of them related to the architecture designed for ANASTACIA. Firstly, it is described the activity models that represents the main processes carried out by the ANASTACIA platform. They are an evolution and refinement of the models presented in D1.3. However, some of them have been extended or completely redefined.

Once defined the main activities carried out by the platform it is presented the final architecture of the ANASTACIA platform. In the current deliverable it is also included the detailed architecture of the main components of the architecture, including their description and internal specification, inputs received, and output produced.

In the next part it is detailed the main interfaces considered in ANASTACIA. In this deliverable it is detailed just the interfaces that involve the interaction between the big planes depicted in Figure 2. The details of the interfaces between components within a module are described in the deliverables corresponding to the description of their respective modules in WP3, 4 and 6.

Finally, for the sake of completion, it has also been included in this deliverable a study of the requirements covered by every component of the architecture.

3 ANASTACIA FINAL ACTIVITY MODELS

This section describes the final version of the main activities carried out by the ANASTACIA framework. These activities are derived from the evaluation of use cases, requirements and considering the early results of the developments. Figure 3 represents the ANASTACIA activities described in the following subsections, showing their relation to the different planes of the ANASTACIA high level architecture.

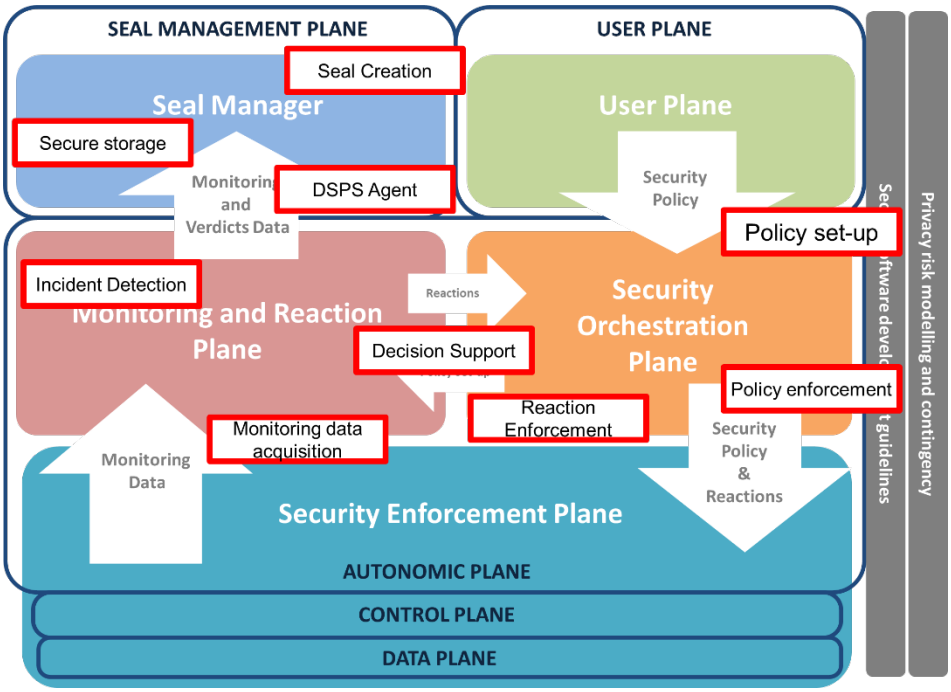


Figure 3. ANASTACIA activities related to the ANASTACIA planes

3.1 POLICY MANAGEMENT

3.1.1 Policy set-up

Policy set-up process allows security administrator defining and enforcing orchestration security policies in High-level Security Policy Language (HSPL) by using the Policy Editor Tool. Figure 4 shows this process where these high-level policies are refined during the policy refinement process in the Policy interpreter which generates orchestration security policies in Medium-level Security Policy language (MSPL) according to the information gathered from the ANASTACIA system model.

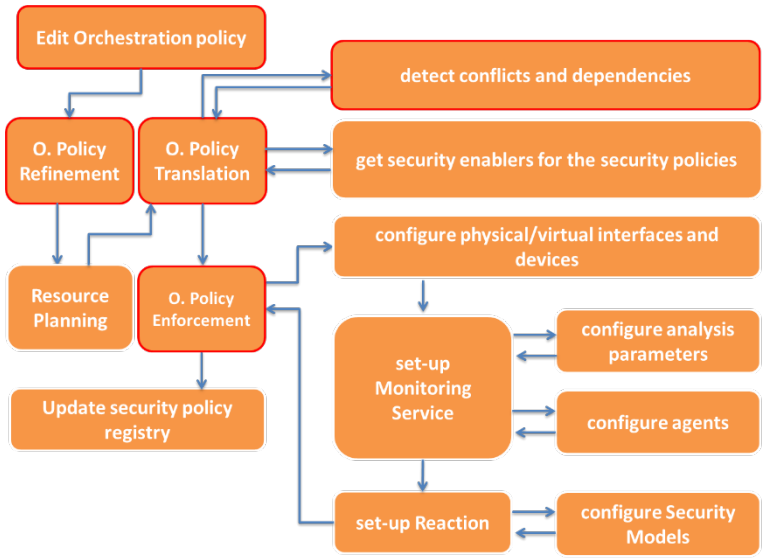


Figure 4. Policy set-up model

The generated MSPLs for orchestration are sent to the Resource Planner in the Orchestrator who determines the best security enablers that can enforce the MSPL policies. Then it requests a policy translation to the Policy Interpreter in order to get the final configurations for the selected security enablers. Now the Policy interpreter also performs a policy conflict and dependencies detection by using the Policy Conflict Detector service. In this way, the Policy Interpreter returns not only the policy translations but also the conflicts and dependencies inter/intra security policies as well as between security policies and the system model. When the orchestrator receives the configurations, conflicts and dependencies, it manages them in order to perform the enforcement carefully in the different security enablers taking into account priorities, dependencies and conflicts.

3.1.2 Policy enforcement

Policy enforcement process starts when the Security Orchestrator receives the correspondence between the Orchestration MSPL policies and the final configurations for the previously selected security enablers, as well as the possible conflicts and dependencies. Figure 5 shows the policy enforcement model at high level.

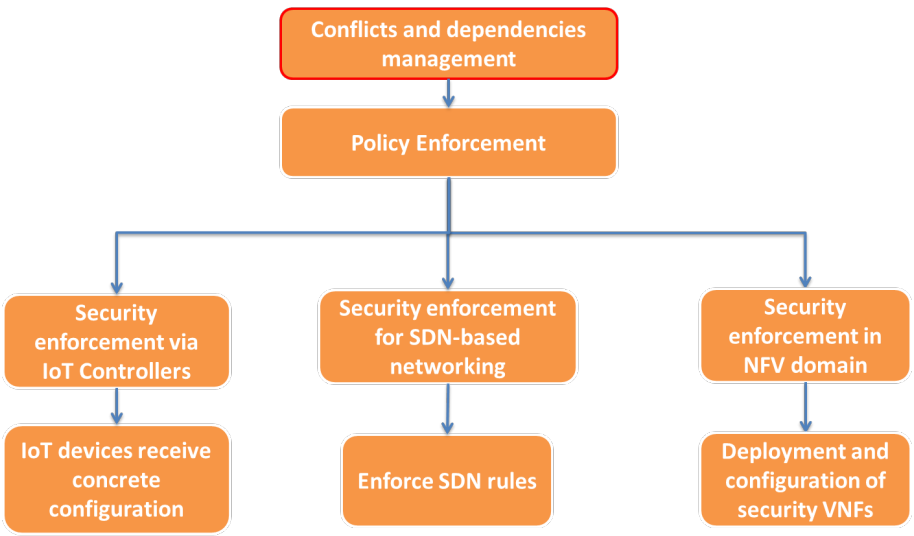


Figure 5. Policy Enforcement model

When the Orchestrator receives the data, it manages priorities and dependencies, queuing those policies that are not already satisfied or even cancelling the enforcement and notifying the result of those policies

that present conflicts. If the policy enforcement is affordable, the Security Orchestrator uses specific drivers for each involved security enabler in order to apply the new configurations. For instance, if the security policy is related to IoT management the Orchestrator will use the IoT Controller driver in order to apply the IoT configurations though the IoT Controller. If it is required to modify the configuration of the SDN network, it can be done by using the SDN driver in order to enforce the configuration through the SDN controller. In case the security enabler is not up and running in the system it will be dynamically deployed by using the NFV-MANO driver.

3.2 MONITORING ACTIVITIES

3.2.1 Monitoring data acquisition

This section describes the components that are responsible of retrieving data from the IoT infrastructure that can be used to infer a potential ongoing incident.

3.2.1.1 Monitoring by MMT probe

The data acquisition performed by the MMT software is performed by analysing the traffic packets from the network. To this end, MMT-Probe relies on the MMT-IoT software in order to facilitate the extraction of the packets from an IoT network. Figure 6 shows the main flow of the process followed to perform the monitoring activity.

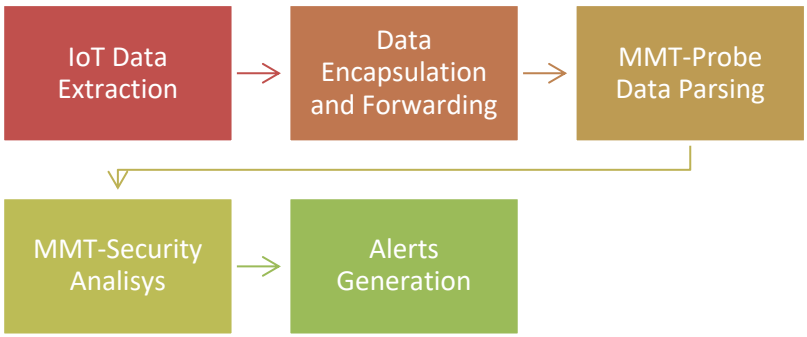


Figure 6. Monitoring flow followed by MMT-Probe

1. IoT-Data Extraction: At this step, MMT-IoT is deployed on the network and actively receiving the traffic from the network. The packets received by the MMT-IoT Sniffer are transferred to a machine with higher capabilities using a USB line.
2. Data encapsulation and Forwarding: Once an extracted packet is received via the USB line, the data packet is encapsulated using the ZEP, IP and ethernet protocols. This step is required to forward the packet to loopback interface – the most common case – or to a remote security analyser using a common Linux socket.
3. MMT-Probe Data Parsing: The packet is then received by MMT-Probe via the selected interface. This software captures the packet (by sniffing the corresponding network interface) and invokes the DPI engine to extract data about the network protocols present in the data, which includes the data from the encapsulated IoT protocols. These data are the main input for the next step.
4. MMT-Security Analysis: As mentioned before, MMT-Security relies on the data extracted by MMT-Probe in order to perform the security analysis. This software treats the extracted information as network events and performs a correlation analysis, testing security properties that have been loaded as plugins.
5. Alerts Generation: Each loaded plugin will generate a verdict each time it is processed. Depending on the nature of the plugin, an alert will be risen depending on the verdict of the property; if the plugin codifies a security property, the alert will be risen when the property is not correctly verified

(negative verdict); if the property codifies an attack definition, MMT will raise an alert if the property is verified (therefore, the occurrence of the attack is confirmed)

More details about how the MMT-Security properties have been given in D4.1 [10] and D4.4 [12].

3.2.1.2 Monitoring by CNR probes

The agent implemented detects an innovative attack on IoT contexts based on the AT Commands exploitation. The attack aims to reconfigure IoT devices in order to disconnect them from the legitimate Zigbee network [Vaccari2017]. The agents are directly installed and implemented inside the IoT devices. The agent is able to monitor the device status and verify that the network parameters are correctly configured in order to communicate on the network. In case the device is affected by a reconfiguration attack exploiting Remote AT Commands, the agent is designed to autonomously reconfigure itself, hence alert the IoT coordinator. Since not all the devices may embed an autonomous detection and mitigation system, the IoT coordinator is also supposed to monitor devices status to identify disconnections. Each generated alert is forwarded to the ANASTACIA monitoring plane through a Kafka broker, as shown in Figure 7.

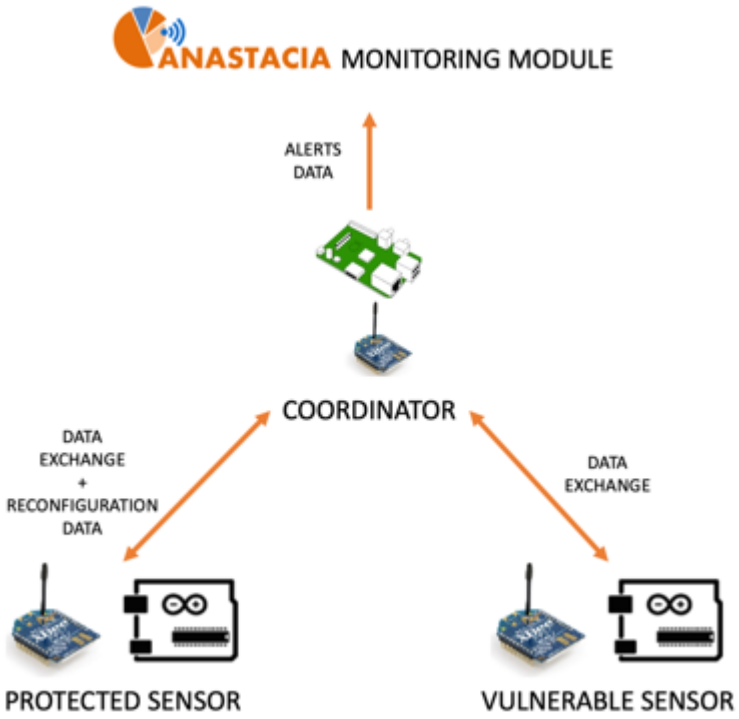


Figure 7. An overview of a remote AT command IoT attack detection approach

A workflow of the agent behavior is reported in the diagram flow reported in the following.

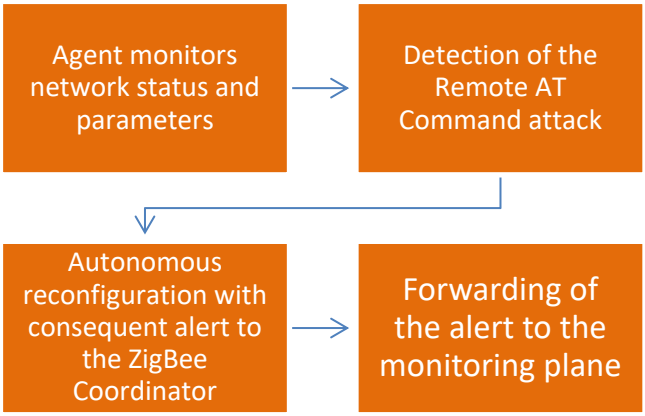


Figure 8. Monitoring flow by CNR probe

By adopting such approach, the attack is autonomously identified and mitigated, hence, damages are reduced by ensuring security of the entire infrastructure.

3.2.1.3 Data Analysis

Data analysis agent performs system level monitoring by aggregating information from different ANASTACIA components using Kafka broker from ANASTACIA SEP.

The component monitors each IoT sensor using combination of CP and ML techniques in a comprehensive model that is able to detect anomalous behaviour of each monitored sensor. The model is created through supervised learning technique where ground truth data (label) is mixed with malformed data set that is emulating adversary behaviour.

In ANASTACIA project ground truth information was gathered via OdinS IoT GW that was reporting sensor information through Kafka broker. After training and cross-validation completion model is deployed to production environment where verdict generation is enabled through MSPL policy configuration interface for each ANASTACIA SEP sensor separately. The verdict computation and assessment are fast as it is similar in speed to computing hash function.

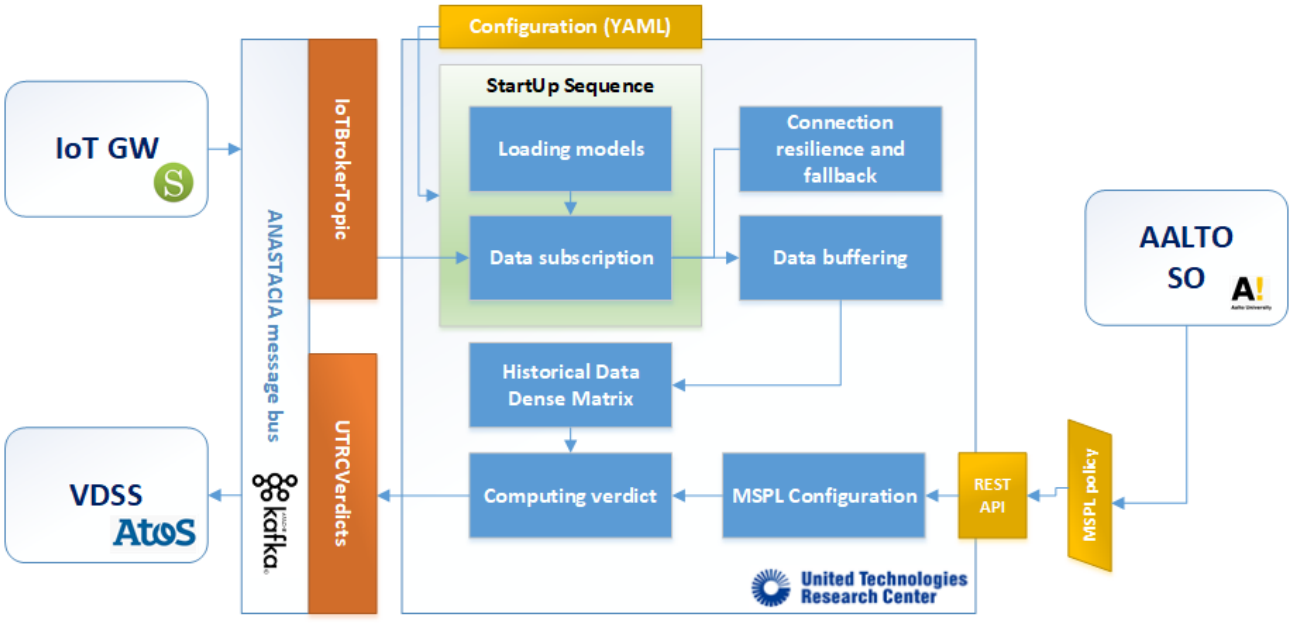


Figure 9. Monitoring process inside UTRC Data analysis agent.

Figure 9 illustrates monitoring steps inside agent to perform anomaly detection within ANASTACIA framework. During start-up sequence (green box) agent is configured via configuration provided through YAML formatted file where ANASTACIA environment connectivity parameters are placed to enable the agent to communicate with other project components. During start up sensor models are loaded to agent memory and new data subscription on Kafka broker is started. To ensure correct data flow OdinS IoT GW is consuming data via two channels that is REST API and Kafka message broker. Internally agent have some resiliency mechanisms to enable operation even in case of failure of both communication channels.

Upon data reception irrespective of communication channel agent start process of data buffering and composition of data dense matrix that will contain timestamp and each sensor current temperature information. In case if MSPL reconfigures agent to run verdict (anomalous behaviour detection), the agent will compute sensor verdict and will send it back to ATOS VDSS component for further investigation and remediation of detected attack. Further details on how agent is constructed can be found in D4.3 [11], D4.6 [14].

3.2.1.4 MiTM agent – adversary emulator

In order to emulate attacker behaviour in controlled environment UTRC team created MiTM agent, that can attack OdinS IoT temperature sensor infrastructure deployed to ANASTACIA SEP. The agent action steps were illustrated on Figure 10. When MiTM agent starts the first thing is trying to perform is to read current temperatures from OdinS IoT GW to enable attack to cover its tracks on the system that control has been taken over for attack duration. Once temperature is loaded to memory and no attack is performed the agent will read temperature values to keep track of the system behaviour. This is passive persistence within ANASTACIA SEP perimeter.

Active phase starts when after self-deployment the agent would “call home” to notify attacker about ability to control IoT sensor system to further penetrate given cyber-physical system due to IoT sensor exploit. Call home feature wasn’t implemented in ANASTACIA framework to enable safe execution of this scenario. The agent has three command and control instructions that can be executed remotely via REST API interface to emulate adversary behaviour:

- **Start** – this instruction starts attack on predefined sensor range. The algorithm will down select sub-set to attack and start changing temperature sensors based on random temperature selection for each sensor. In this phase MiTM agent is actively generating actively attack.
- **Stop** – this instruction will stop attack, leaving sensors with false temperature readings. When this command is issued APT will stop any active malicious behaviour this can be considered as hide phase.
- **Recover** – this command recovers previous temperature settings observed by the agent before attack. In this step the attacker is trying actively cover tracks of the attack by restoring values on the sensors.

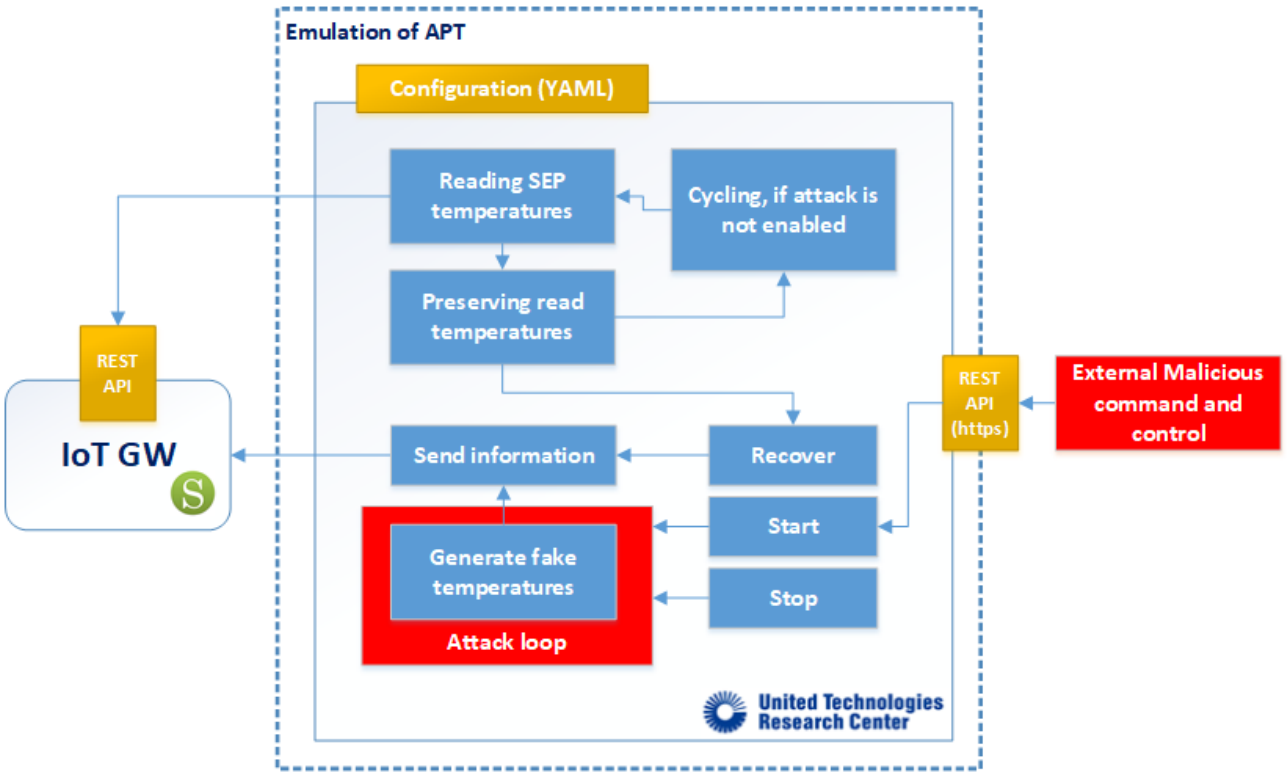


Figure 10. MiTM attack process inside UTRC MiTM agent

3.2.2 Incident detection

The incident detection process is part of the monitoring activities that retrieve monitoring data from security probes, triggering alerts when the analysis of the events denotes an ongoing incident. Figure 11 represents the detailed process for the detection of incidents. This process is divided into four main steps: retrieve events, analyse aggregated data, calculate risk and emit verdicts about attacks.

The step “retrieve events” compiles events from observed agents, aggregating events produced by the different security probes deployed within the infrastructure. In this step, raw information is received from different probes, which might be in different format and containing different type of information depending on the type of probe that has generated it. This information is filtered, to remove unnecessary information, classified, to identify the type of events and normalized to a common format used by the following steps of this process. In this normalization phase the raw events are transformed to a unified format where the information is mapped to common fields for all types of events. The set of normalized events is submitted to the following step for processing.

The next step analyses the aggregated data received from the “retrieve events” phase. In this event the information is extracted from the normalized events. With the events received and the information extracted from them it is evaluated several correlation rules that identify patterns, check for certain information and, in general, several criteria is analysed to identify evidences of ongoing incidents. In this phase it is also generated an alert in case an incident is identified. However, before triggering it is calculated the risk associated to such incident, which is part of the following phase.

The phase “calculate risk” evaluates several metrics associated to the events that have triggered an alert. The risk is calculated combining different scores for different aspects of those event, such as the priority of the type of events, the reliability of the source that has generated such event and the importance of the asset affected by the incident, which depends on factors such as the location of the device and the type of device. The result is a numerical score that represents the risk of an ongoing incident over the infrastructure when certain device of a certain type is affected by it. Once obtained this value the step “analyse the aggregated

data" generates the alert which, in the final step, is pushed to be collected by the consumers of this information (for example to the VDSS).

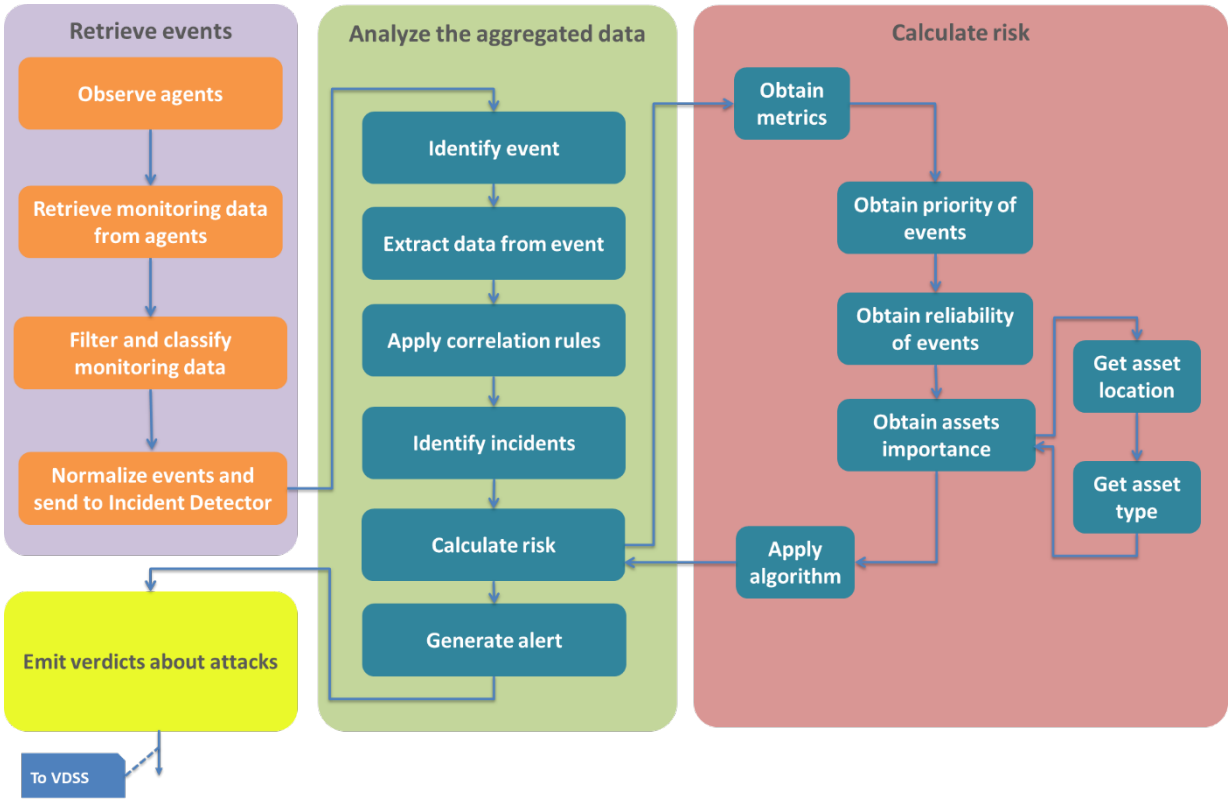


Figure 11. Complete final monitoring process

3.3 REACTION ACTIVITIES

3.3.1 Decision Support

This activity carries out the evaluation of the different available mitigations strategies that can mitigate an ongoing incident, providing with a suitability score per mitigation and a mitigation ranking with the more suitable mitigations ranked on top. Figure 12 represents the complete process for deciding on the most suitable mitigations. This process is composed of two main steps: Calculation of mitigation suitability and Mitigation selection.

The step "calculation of mitigation suitability" is divided into two main parts: extraction of the information associated to the alert to mitigate (i.e., assets affected, type of incident detected, and risk associated to it) and the calculation of the suitability scores for the mitigations. For extracting the information related to an alert there are two extra steps:

- Calculate the impact of the mitigations when enforced within the infrastructure. To this end it is evaluated the type of incident, and aspects such as the number of devices affected by the mitigation or the resources required to enforce it.
- Obtain information about the assets to calculate the criticality of the assets affected by the mitigation, which is obtained considering aspects such as the type of device affected by the mitigation or the location where it is running.

With this information it is calculated the suitability score of the mitigation and the corresponding ranking.

Finally, the step Mitigation Selection includes the final selection of the mitigation to enforced, which can be done either manually or automatically depending on CISO and mitigations preferences.

Further details of this process are given in deliverable D4.5.

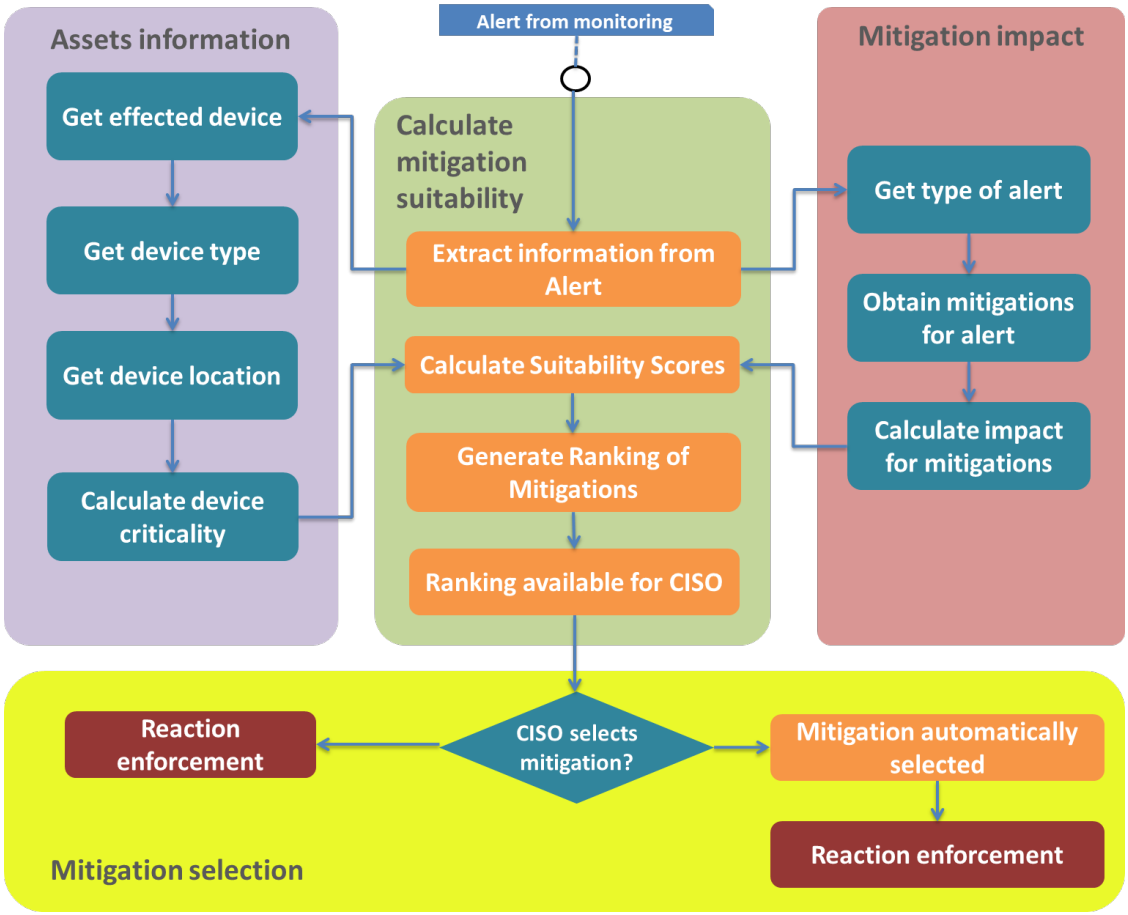


Figure 12. Complete final decision support process

3.3.2 Reaction enforcement

This activity, represented in Figure 13, uses the output of the Decision Support activity as the main input. The principal goal is to generate a tailored mitigation plan based on the assessment process carried out by the VDSS.

The MAS takes the output of the VDSS that contains the evaluation of the mitigation strategies. With this information, the MAS will start generating the orchestration policy that will contain one or more mitigation actions that need to be enforced. To this end, the MAS will make use of the information stored in the System Model and Assets Model components in order to create a tailored instance of the mitigation that codifies the countermeasures that need to be executed by the Security Orchestrator.

Once the mitigation plan has been generated, the MAS will send these countermeasures to the Security Orchestrator. To this end, the MAS will use the particular API provided by the SO in form of an HTTP server: the MAS will send a POST request to the SO whose payload will contain the MSPL orchestration containing the mitigations to be enforced.

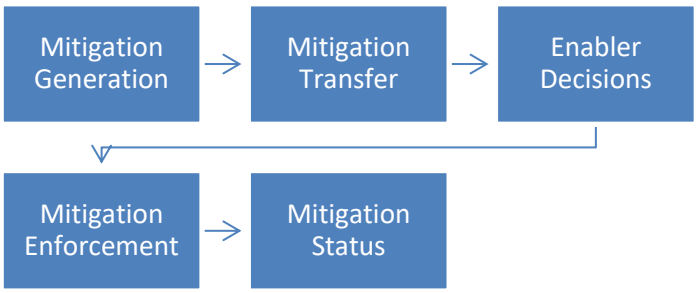


Figure 13. Reaction Enforcement process

3.4 DYNAMIC SECURITY AND PRIVACY SEAL

This activity, represented in Figure 14, provides a solution to privacy and security monitoring process, addressing both the organizational and technical requirements enshrined by the GDPR through the implementation of a two-step process during which:

- the end-user (DPO/CISO) will receive information from the DSPS (based on the alerts and other data provided by ANASTACIA's monitoring and reaction service); and
- the end-user will provide feedback on the alerts and data obtained from the DSPS and use the DSPS tools to securely store data to support any organizational claims for compliance with relevant regulations or proof of security/privacy control performance.

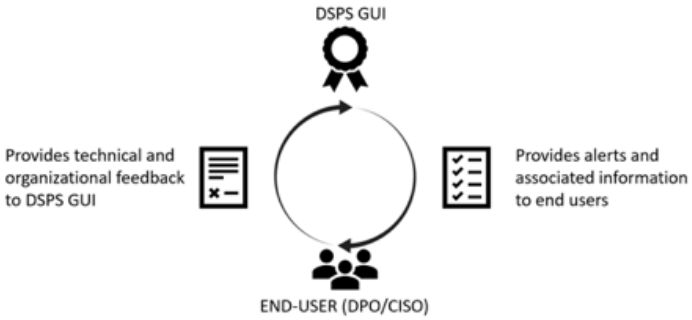


Figure 14. DSPS overview process

3.4.1 Seal Creation

Seal Creation is the activity taking place at the end of the pipeline of ANASTACIA's event processing. As described before, ANASTACIA carries out activities including monitoring for detecting incidents, deciding which mitigation plan is more suitable to handle the incident, enforce the mitigation plan, etc.

Seal Creation activity, represented in Figure 15, is to process security incidents and mitigation plans described in messages received from SAS for generating "Seals". Seals describe system status at a given moment in time, notably when new security incidents have been detected, it provides real-time status to the user and at the same time it provides corresponding privacy risks which is performed by AI module harmonizing DPOs' inputs. Seals and associated logs are then are saved in the DSPS Storage and shared with the user when applicable.

The following schema depicts the simplified flow of actions within the DSPS component.

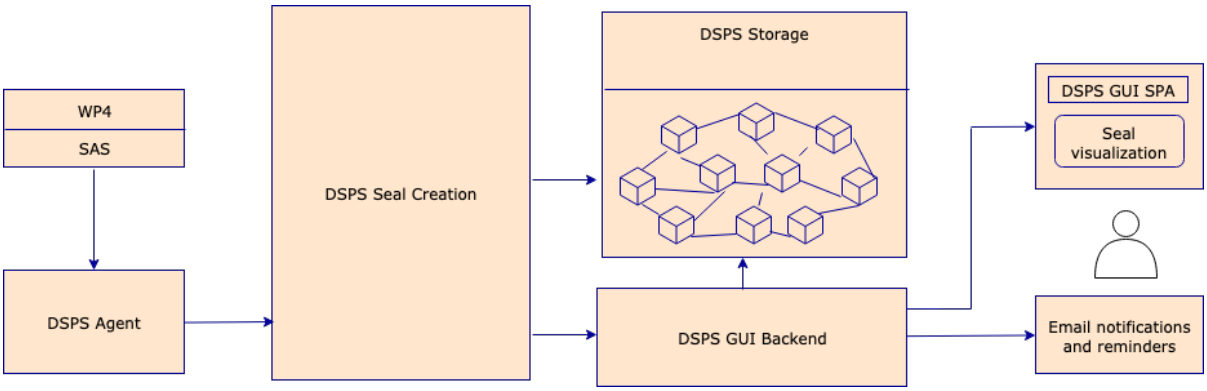


Figure 15. DSPS event processing

3.4.2 DSPS Agent

The Agent listens to all events produced by the SAS and triggers the consequent actions of DSPS Seal Creation by using the DSPS Creation Service API. The DSPS Agent changes the format of the received message from SAS to standard format of Structured Threat Information Expression (STIX) so that the DSPS services can be provided on top of other security monitoring systems.

3.4.3 Secured storage

This activity provides a secure and reliable storage of the continuous privacy and security seals which are measured by the ANASTACIA platform, and to enable the authentication of the information stored in the off-chain log. For the sake of auditability seals must be saved in an immutable way, this provides also a non-refutable seal history.

Put in simple words, after Seal Creation activity issues a seal and logs, the secure storage activity takes a signed seal and stores it in the blockchain, and the encrypted log which will be split and put in the Shamir storage.

Further details on this activity is described in D5.2 [16].

4 ANASTACIA FINAL ARCHITECTURE

4.1 GLOBAL VIEW

Figure 16 depicts the final architecture of the ANASTACIA framework, which includes the latest changes with respect to the version delivered at M9 in D1.3. The most important aspects that have evolved since then are summarized as follows:

- Policy conflict detector for orchestration policies has been added to the Policy Manager.
- Reaction plane, where the complete VDSS is defined, including components required by the VDSS to fetch input from different parts of the ANASTACIA framework.
- Security Orchestrator, which include additional components to interact with NFV, SDN and IoT functions and components to manage resources used by the orchestrator.
- Seal Manager, where the components included have been refined.
- New components added to the User plane, where User Interfaces have been decoupled and included in this plane.

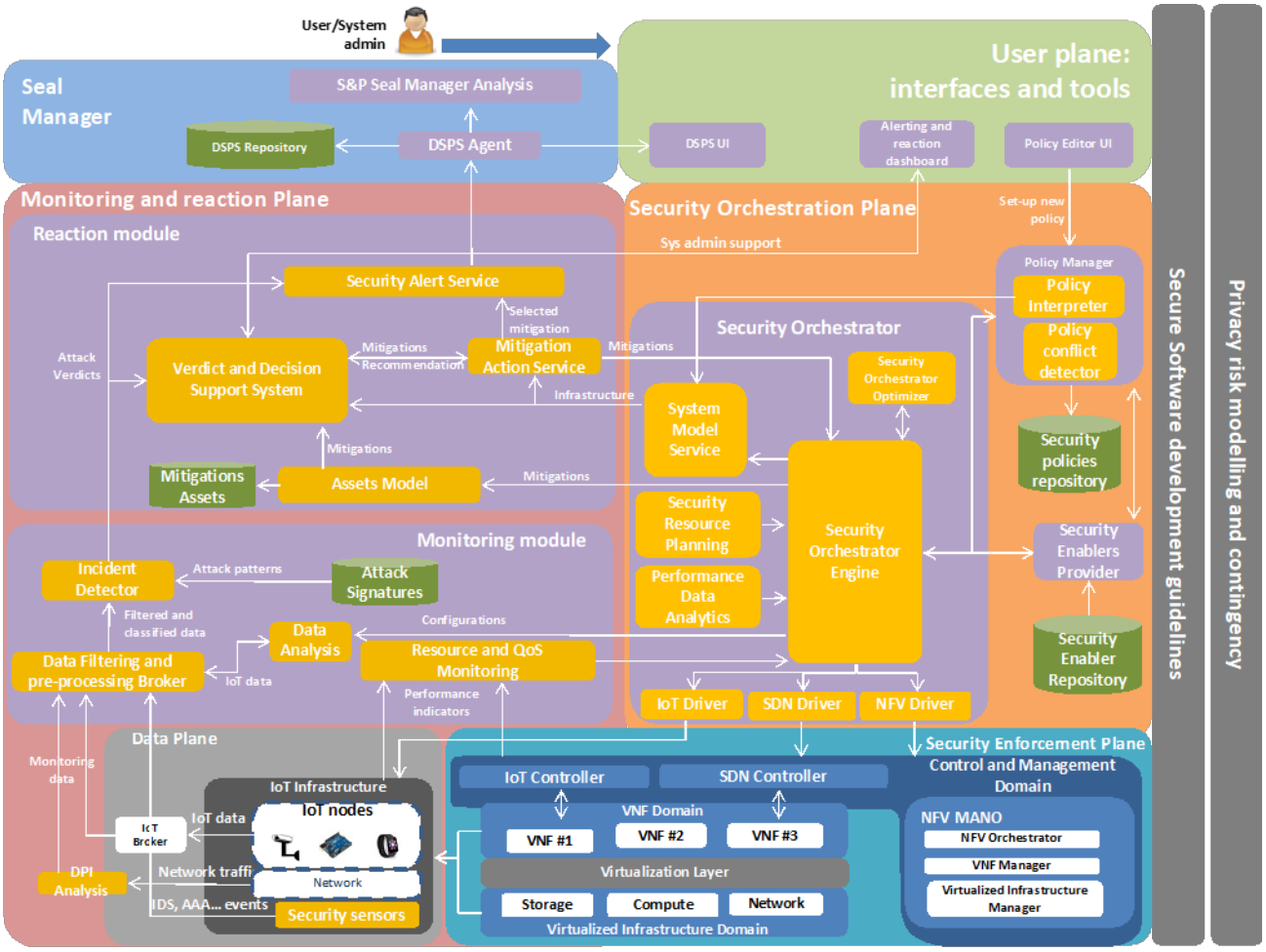


Figure 16. ANASTACIA final architecture design

In the following subsections a detailed description is given for the individual modules included in the ANASTACIA framework.

4.2 MONITORING AND REACTION

4.2.1 Monitoring components

4.2.1.1 Incident detector

The internals of the Incident Detector are depicted in Figure 17. It is composed of two parts: the XL-SIEM agent and the server.

The XL-SIEM agent receives raw events from security probes through a rsyslog server. These events are processed by individual plugins adapted to the type of information received. The XL-SIEM architecture can be extended with additional plugins which allows to easily incorporate information from additional probes. These plugins parse the raw events, extract the relevant information which is used by the agent to generate a normalized event with common fields, which are sent through a socket connection, to the XL-SIEM server. It is worth noticing that XL-SIEM agent and XL-SIEM server can be deployed as separated instances in different remote machines or can be deployed on the same machine.

The XL-SIEM server analyses the normalized events and look for evidences of incidents based on a set of correlation rules that identify patterns or information included in such events. It is composed of several subcomponents, such as the Input spout, which listen for new events received from the XL-SIEM agent, the Event processor, which extracts the information included in the events received, the correlation bolt, which applies rules looking for patterns denoting incidents, a RabbitMQ server to push alerts triggered and the DB write bolt, which acts as interface between the rest of the components of the XL-SIEM server and the database.

Further details about the monitoring process carried out by the XL-SIEM within the ANASTACIA framework are given in the deliverables of WP3.

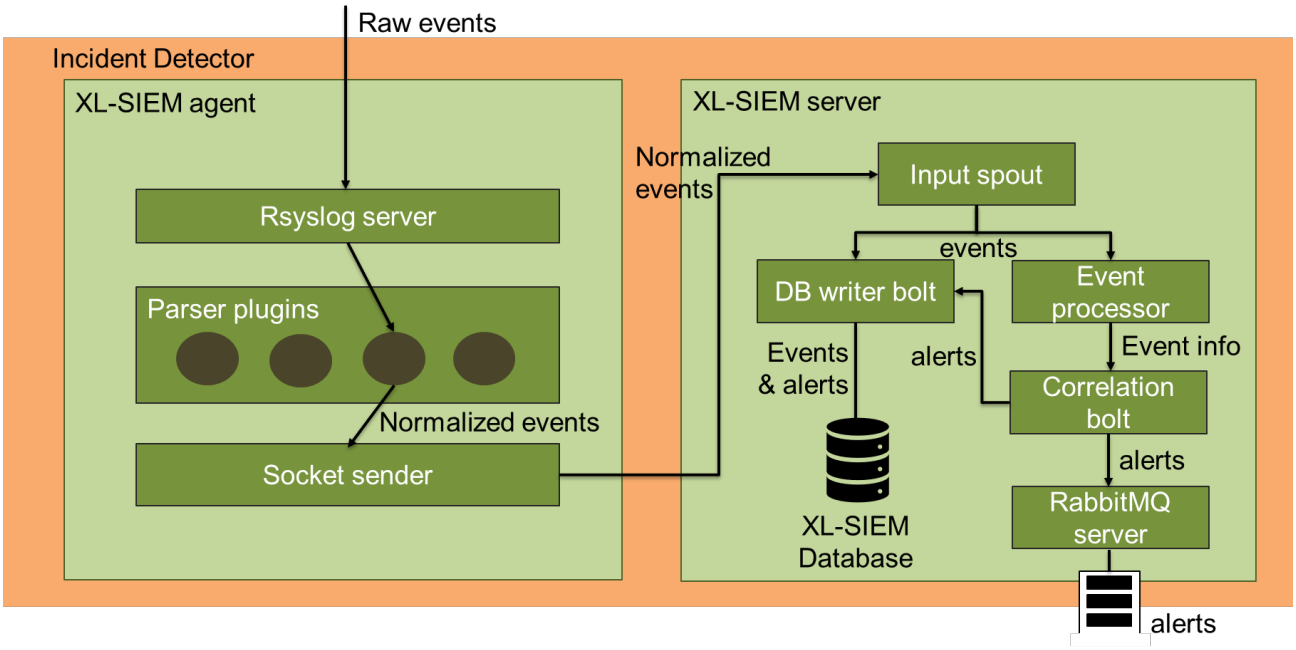


Figure 17. Incident Detector internal architecture

Table 3. Incident Detector details

Incident Detector	
Description	Tigger security alerts based on events received from security probes

Inputs	Raw events received from Security probes
Outputs	Alerts, as a JSON message pushed to a queue which is consumed by the VDSS at the Reaction module and the DSPS plane

4.2.1.2 Data Filtering and pre-processing Broker

Data Filtering and pre-processing component broker is a component that is used to collect, aggregate and filter the raw monitoring data received from the underlying IoT platform and monitoring agents. Based on Apache Kafka and Apache Storm, this component is carrying out an initial pre-processing of the received information in order to be correctly analysed by the Incident Detector. The internals of this component are represented in Figure 18.

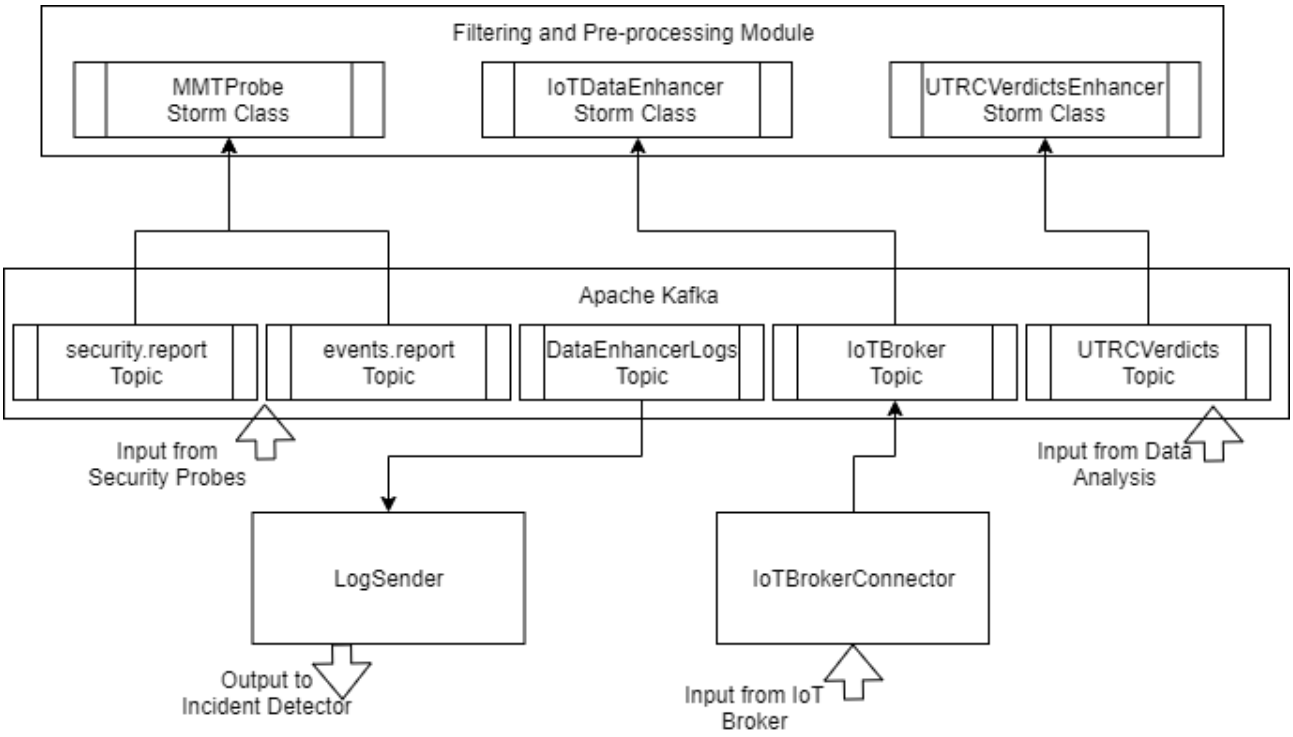


Figure 18. Data Filtering and pre-processing internal architecture

Table 4. Data Filtering and pre-processing Broker details

Data Filtering and pre-processing Broker	
Description	Retrieve monitoring data from the IoT platform, filtering and processing in order to provide it ready for the Incident Detector.
Inputs	- AAA events. OdinS provides with AAA probes that detect any anomaly related to the unauthorized access to IoT devices. - Deep Packet Inspection scanning. MMT provides a DPI scanning of network traffic which can detect potential threats and ongoing attacks

	- Data Analysis. UTRC provide with an anomalous behaviour scanner which uses operational data from IoT devices and detect anomalies in the data they produce.
Outputs	Normalized and filtered events sent to Incident Detector

4.2.1.3 Data Analysis

Details of data consumption and message queuing techniques were described in detail in D4.3. However, in this deliverable, we provide a short description on information processing inside monitoring agent. The processing part is being divided into 2 parts as shown in Figure 19:

- Monitoring – received messages are processed, filtered and cleaned to enable data recording for future model training and attack verdict generation that will be sent to reaction components.
- Detection – system level analysis of current security state of SEP based on trained model and current information stored in monitoring buffer.

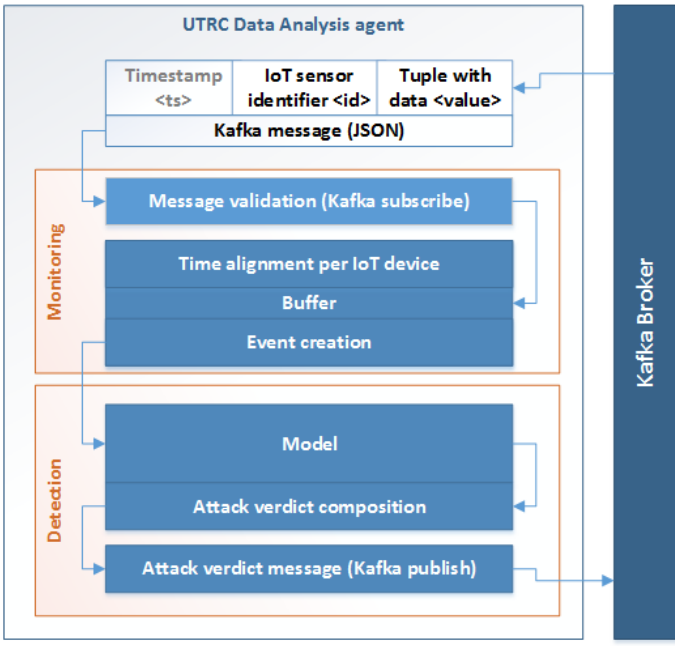


Figure 19. Data analysis agent internal monitoring architecture

An internal agent architecture was illustrated on Figure 1, D4.3. It depicts two main parts of agent (monitoring and detection). Both sides are preconfigured to adjust to various system monitoring capabilities for anomaly detection (temperature, pressure, IR presence, etc...). The common interface between monitoring and detection part enables agent to consume and process various data sources that are available on SEP.

Table 5. Data Analysis details

UTRC Data Analysis Agent	
Description	Agent analyse IoT sensor information (temperature) and provides detection verdict on whether sensor data was tampered with.
Inputs	OdinS IoT sensor temperature data shared on Kafka broker. Format is JSON

Outputs	Verdict information that is sent to ATOS VDSS component via Kafka broker. Format is JSON and adjusted to ATOS VDSS component requirements.

4.2.1.4 Resource and QoS Monitoring

This component is responsible for monitoring the resource utilization and QoS of the SDN network and the deployed virtual network functions (VNFs). This component helps both the security orchestrator optimizer and the data analytics components for making their internal decisions. It is mainly used for ensuring efficient and low-cost life cycle management of the SDN network and VNFs. The collected information helps the security orchestrator for deploying, reusing or destroying VNFs according to the network state and workload at each VNF.

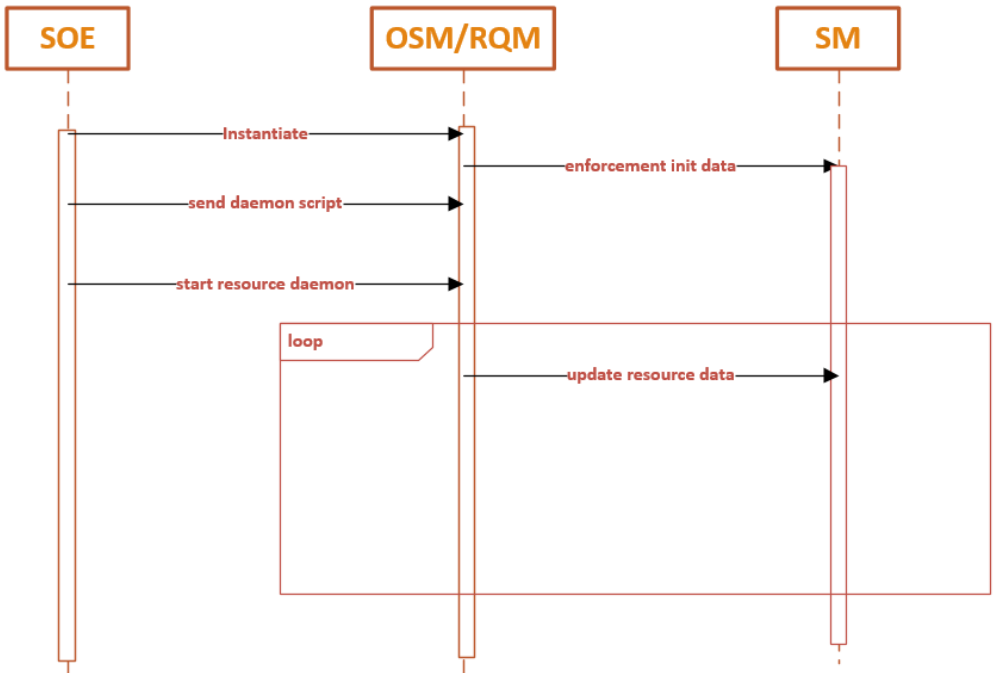


Figure 20. Resource and QoS Monitoring (RQM) interaction with SOE and SM components

Figure 20 shows the interactions between the RQM module with the Security Orchestration Engine (SOE) and System Model (SM). After taking the decisions at the level of Security Orchestrator Optimizer (SOO), the SOE enforces the taken decisions by communicating with different security enablers that include SDN controllers, NFV orchestrators and/or IoT controllers. In case a security VNF should be instantiated, the SOE communicates with the NFV orchestrator (OSM) to instantiate that VNF. Then, the SOE will communicate with the RQM for configuring that VNF by deploying a daemon agent that should keep sending the state of that VNF to the RQM component. From another side, the RQM keeps updating the SM about the state of that VNFs that include the available and used CPUs, RAM and disk.

Table 6. Resource and QoS Monitoring details

Resource and QoS Monitoring	
Description	This component serves for monitoring the resource utilization of VNFs in terms of delay, bandwidth, CPU usage percentage, memory details available, used and total. This component gives the possibility for the security orchestrator for enabling the life cycle management (LCM) of deployed VNFs.

Inputs	None
Outputs	JSON format that should be used by the orchestrator.

4.2.2 Reaction components

4.2.2.1 Verdict and Decision Support System

The internals of the VDSS are depicted in Figure 21. The core of the VDSS is the VDSS Engine Model, which calculates Suitability scores associated to every mitigation that can mitigate an incident. The suitability score is a value that quantitively indicates how good or bad a mitigation is for a given incident, which depends on many different factors, such as the type of incident, the type of device affected by the incident, the devices affected by the mitigation, their location or the resources required to enforce it.

The rest of the components of the VDSS supports the calculation of the suitability scores by providing with information required. The exchange of information from/to external components is managed by the Backend Controller, which acts as frontend to fetch data such as available mitigations, information about assets or configurations from the CISO. An input/output manager handles the interaction with RabbitMQ servers for consuming information such as alerts or to receive feedback about selected mitigations. The VDSS also includes a GUI which is used by the CISO to modify certain parameters related to the mitigation activity, and to insert manually importance scores for certain aspects of the evaluation, which allows to finetune the VDSS analysis and to provide with a more accurate evaluation. All components are supported by a VDSS Database which stores information about mitigations, impact evaluations, and other pieces of information relevant for the VDSS Engine Model.

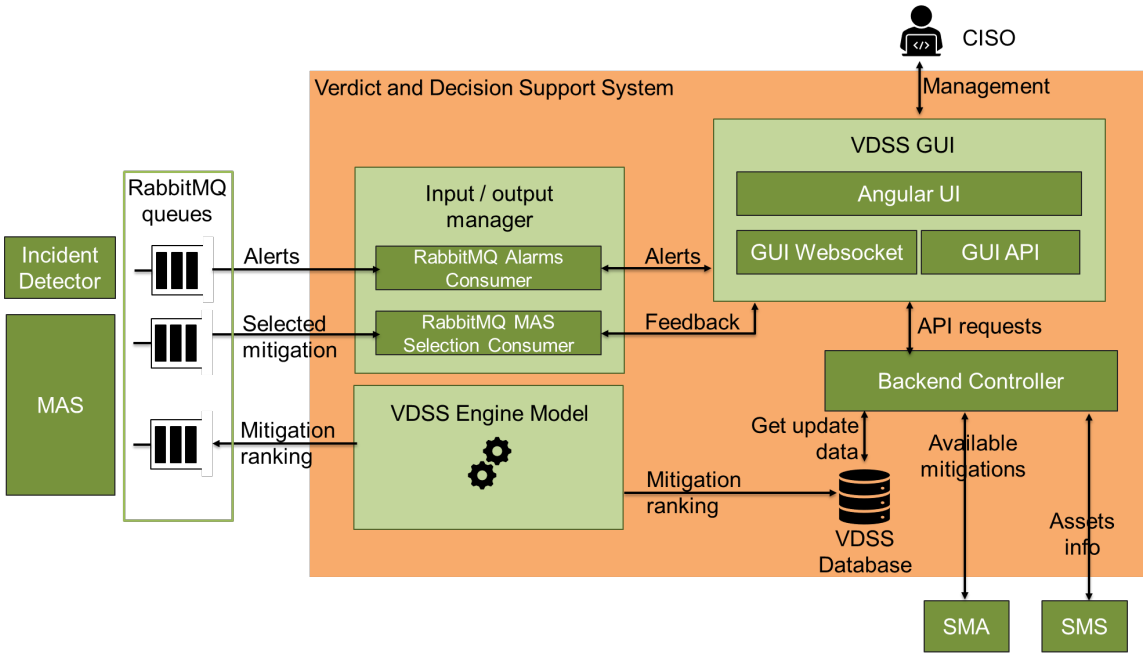


Figure 21. Verdict and Decision Support System internal architecture

Table 7. Verdict and Decision Support System details

Verdict and Decision Support System	
Description	Evaluates mitigations for a given incident, calculating a score representing the suitability of the mitigation and generating a ranking of mitigations depending on the suitability scores obtained.

Inputs	Alerts detected at the Incident Detector, received through a RabbitMQ server List of available mitigations received from the Security Model Analysis. Information about affected assets received from the Security Model Service. Feedback on the selected mitigation received from the Mitigation Action Service Configuration parameters inserted by the CISO
Outputs	Mitigation ranking pushed to a RabbitMQ queue to be consumed by the Mitigation Action Service.

4.2.2.2 Assets Model

THALES Assets Model is responsible for enabling search and updating of database in which mitigation strategies, capabilities and actions are stored. Figure 22 illustrates internal architecture of the components which is comprised out of three main REST API that provide access to related mitigation entities. Interaction between SO and Assets Model is done through pooling mechanism that will fetch from SO information model updates or changes applied to SEP environment and updates corresponding mitigation entities to what is currently available (flavour optimization).

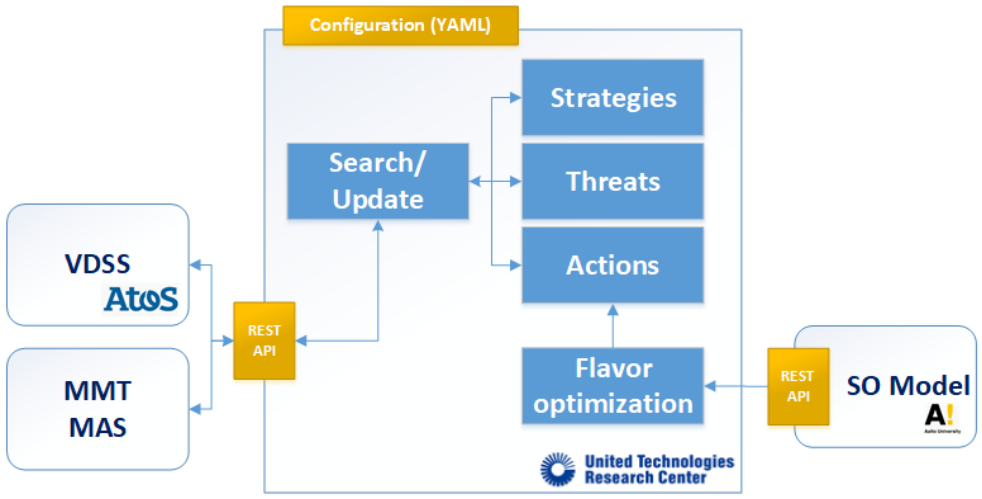


Figure 22. THALES Asset Model internal architecture.

Table 8. Assets Model details

Assets Model	
Description	Preserves and enables searching and filtering for current ANASTACIA framework mitigation strategies, observed threats and mitigation actions for other components to use.
Inputs	SO Model – flavours information (REST API). Format is JSON.
Outputs	For each of the data set a corresponding REST API is designed to provide required information to other components: Mitigation strategies - https://10.79.7.172/strategies/ Capabilities – https://10.79.7.172/capabilities/ Mitigation actions – https://10.79.7.172/actions/ For all above REST API is used and data is stored in JSON format.

4.2.2.3 Security Alert Service

The Security Alert Service is the component of the Reaction module aimed to combine and integrate information retrieved from other internal components of the module, group them and propagate/send to the DSPS the combined data.

The SAS module, represented in Figure 23, combine the detailed information about the threat detected by the VDSS with the reaction policies, generated by the MAS, to mitigate and protect the system from the threat. The adoption of this strategies will therefore lead to a link between the identified threat and the selected mitigation.

The Security Alert Service is composed mainly by three modules: two daemons and one engine. The daemons are adopted to communicate with the VDSS, the MAS and the System Model components, while the engine elaborates the information and communicates with the DSPS. In particular, when a threat alert is received by the VDSS daemon, it is automatically forwarded to the engine, that starts to prepare the data packets for the DSPS. Then, when the MAS communicates the policies applied to protect the system to the MAS daemon, the SAS engine correlates the policies and the alert and pushes this information to the DSPS. At the same time, the SAS starts polling the System Model to get policies enforcement information (using a shared identifier received by the MAS). As enforcement information is received, it is forwarded to the DSPS (optionally, information received after each polling can be forwarded to the DSPS as well), and polling is interrupted.

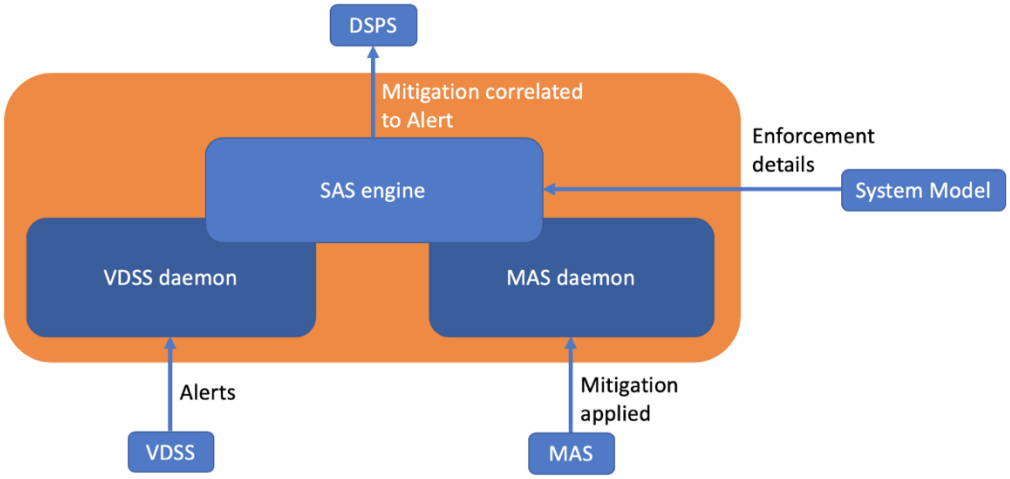


Figure 23. Security Alert Service internal design

Table 9. Security Alert Service details

Security Alert Service	
Description	Security Alert Service integrates and enriches the alert information received from the VDSS with the deployed mitigation selected and applied by the MAS, plus policy enforcement information received by the System Model; then, the SAS sends the complete information to the DSPS.
Inputs	Alert from VDSS, mitigation applied from the MAS, enforcement information from the System Model
Outputs	Integration of alerts with mitigations and enforcement information

4.2.2.4 Mitigation Action Service

Once the security alerts have been correctly assessed by the VDSS and the possible mitigation plans have been evaluated, it is required to generate the corresponding mitigations that will be applied on the network. The ANASTACIA platform relies on the Mitigation Action Service (MAS) to do this task.

The MAS acts as a central point that correlates the information contained in other ANASTACIA components in order to generate a crafted version of the mitigation strategy evaluated by the VDSS: To this end, the MAS will rely on the information contained in the System Model Service, which contains an up-to-date snapshot of the status of the monitored network. The MAS will make use of this information in order to correctly configure the evaluated mitigation plan and determine how best to apply it. At the end of this process, the MAS will generate an *orchestration security policy* that contain one or more security policies that will be enforced by the Security Orchestrator.

Figure 24 shows the general, high-level design of the MAS, showing the principal submodules of this component. In general, the MAS is composed of four principal components: the *MAS-Core* component that contains the main classes that enable the basic functionality of the MAS, and three *model* components that provide the classes to access the data contained in external APIs consumed by the MAS. Further details of the MAS (including a more detailed class diagram) can be found in D4.5 [13].

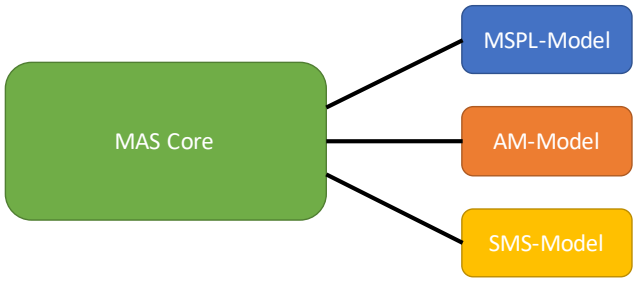


Figure 24. High-level design of the Mitigation Action Service

Table 10. Mitigation Action Service details

Mitigation Action Service	
Description	Generates the mitigations of the detected attack that will be enforced by the Security Orchestrator, crafting the countermeasures using the information stored in the System Model Service.
Inputs	• Security Alerts received from the Verdict and Decision Support System. • Mitigation Strategies, Capabilities, Actions received from the Assets Model. • System Model Information received from the System Model Service – Security Orchestrator.
Outputs	• Reactive Mitigation Policies, consumed by the Security Orchestrator.

4.3 Orchestration and Enforcement

4.3.1 Security Orchestrator Engine

The security orchestrator engine is the main sub-module of the security orchestrator component that has the responsibility for receiving the different policies either from the end-user or from the monitoring and reaction plane. It has the responsibility to take into account the policy requirements and the available

resources in the underlying infrastructure for mitigating the attacks while reducing the expected mitigation cost and without affecting the QoS requirements of different verticals. It has also the reasonability for deploying and configuring different security enablers using SDN controllers, NFV orchestrators and IoT controllers.

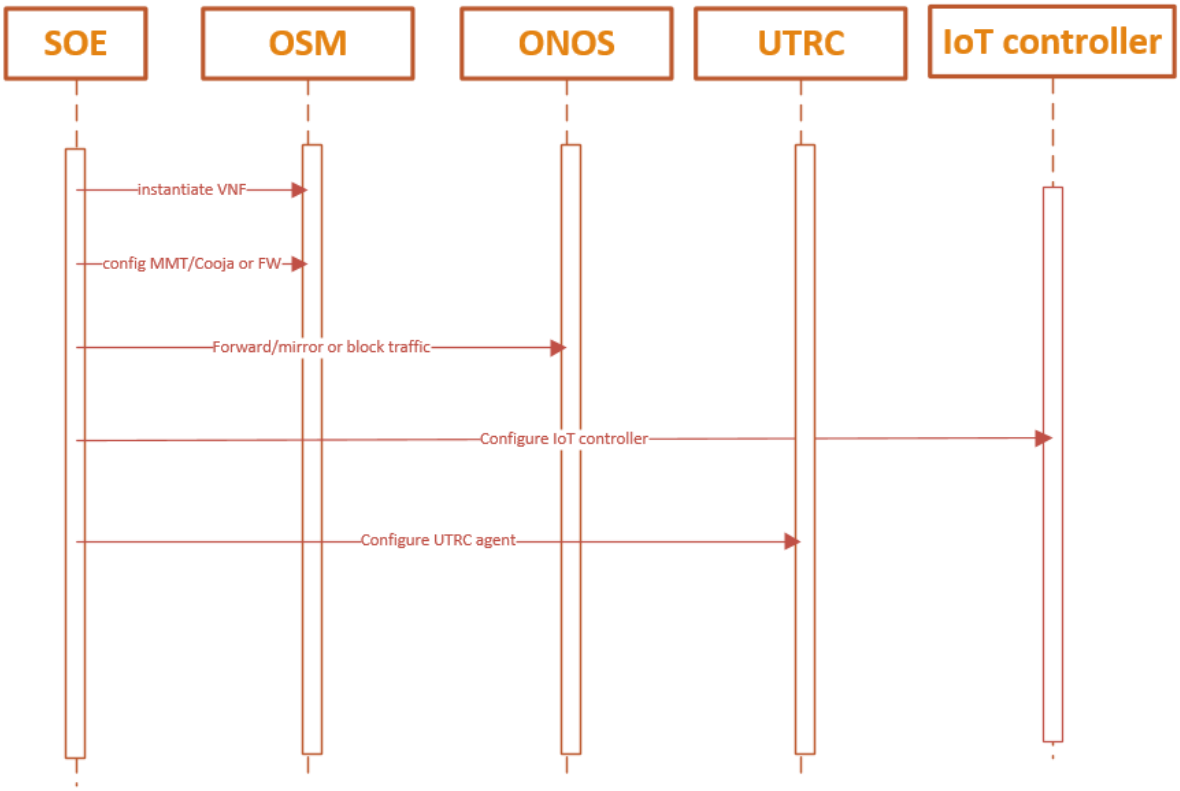


Figure 25. Security Orchestrator Engine (SOE) interaction with different security enablers

Figure 25 shows the interactions between the SOE module with different security enablers. The SOE enforces the taken decisions at the SOO by communicating with different security enablers that include SDN controllers, NFV orchestrators and/or IoT controllers. In case a security VNF should be instantiated, the SOE communicates with the NFV orchestrator (OSM) to instantiate that VNF. In case a new SDN rule should be pushed, the SOE will forward the required requests. Also, according to the taken decisions, the SOE could communicate with IoT controller or UTRC agent for enforcing the adequate security rules.

Security Orchestrator Engine	
Description	This module ensures the end-to-end service orchestration and provisioning. It offers REST API to communicate with Mitigation Action Service (MAS) and third-party component for mitigating different attacks using different enablers.
Inputs	MSPL file (XML)
Outputs	JSON message that shows the status of the enforced policy. In addition, the security orchestrator engine updates the system model about the new deployed enablers for further use.

4.3.2 Policy interpreter – M2L Translator

Policy interpreter receives the orchestration policies and the selected security enablers from the orchestrator, and it performs the policy translation by using the specific plugin for each security enabler. The

conflict and dependencies detection relies on the policy conflict detector module. Figure 26 shows the high-level design of this module.

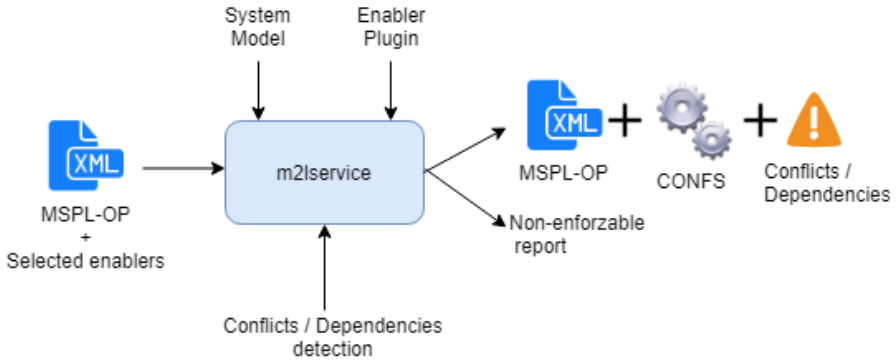


Figure 26. M2L Translator high-level design

Policy Interpreter – M2L Translator	
Description	This module allows translating orchestration security policies into security enablers configurations through a REST APIs.
Inputs	MSPL file (XML)
Outputs	JSON message which includes the correspondence between security policies and translations as well as possible conflicts and dependencies.

4.3.3 Policy conflicts and dependencies detector

Policy conflicts and dependencies detector is in charge to detect conflicts and dependencies inter and intra orchestration security policies also taking into account the current status of the system model. The detection process can be embedded as part of the refinement/translation process or it can be accessed independently. Figure 27 shows the high-level of this component.

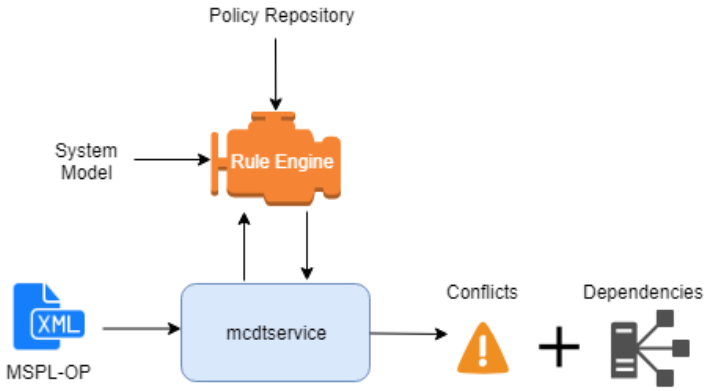


Figure 27. Conflict detector high-level design

Policy Conflicts and Dependencies detector	
Description	This module allows detect conflict and dependencies between orchestration policies through a REST API.

Inputs	MSPL file (XML)
Outputs	JSON message that includes the correspondence between the security policy identifiers and the conflict or dependency types.

4.3.4 Security Orchestrator Optimizer

This component is responsible for defining different policies that ensure the life cycle management (LCM) of the VNFs. To reduce the cost, this component uses as much as possible the existing VNFs. Also, this component is responsible to ensure the QoS while preventing different attacks. It also has the reasonability for preventing the conflicts between the different security policies already enforced at different VNFs and PNFs.

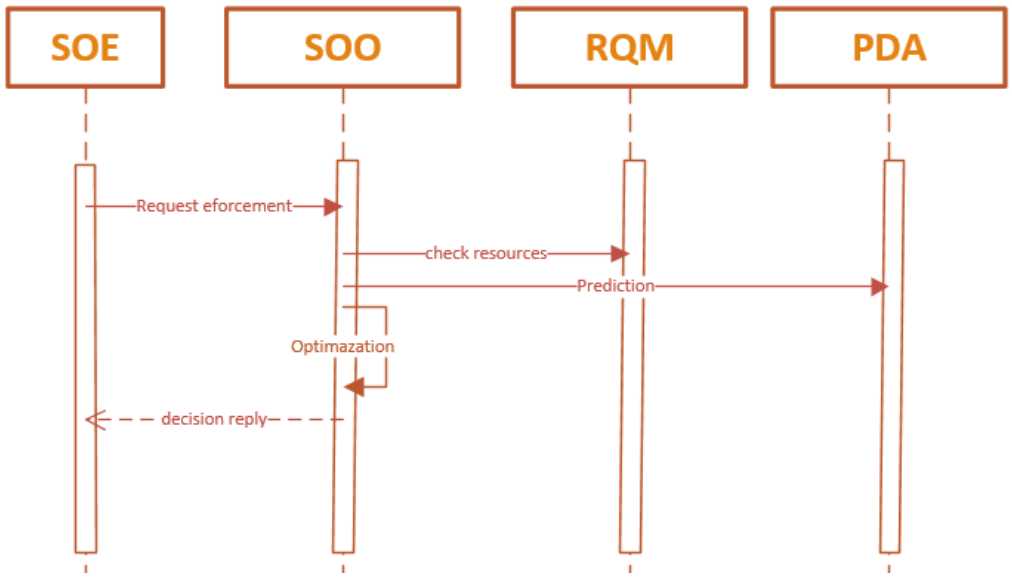


Figure 28 Security Orchestrator Optimizer (SOO) interaction with different security orchestration components

Figure 28 shows the interactions between the SOO and the other security orchestration components. As depicted in this figure, SOE sends the enforcement request to SOO. The latter checks the availability and predictions of resources and QoS from Resource and QoS Monitoring (RQM) and Performance Data Analytics (PDA) agents. After receiving the feedbacks, SOO generates optimal configurations and decisions using linear integer optimization technique. The decisions could include the instantiation or deletion of VNFs and/or the redirection of ongoing flows through existing VNFs.

Security Orchestrator Optimizer	
Description	This component uses linear integer programming optimization in order to select deploy and manage the different enablers for mitigating various attacks.
Inputs	Network information from the system model and information about the policy from the policy interpreter and the mitigation action service.
Outputs	JSON message that describes the network configuration to be enforced by the security orchestrator engine.

4.3.5 Performance Data Analytics

This component is responsible for predicting the resources that could be used by different VNFs in the future. Also, it predicts the network traffic and state in the future to proactively prevent the network and VNF overload in the future. This component is used by the SOO component for the right decisions and configurations.

Performance Data Analytics	
Description	This component predicts the amount of resources utilization of VNFs in order to make the decisions early and preventing the dysfunctionality of the system.
Inputs	VNF resource information received from the Resource and QoS Monitoring component.
Outputs	Information about predicted resource utilization that should be used by the security orchestrator optimizer for taking the timely right decisions.

4.3.6 Security Resource Planning

The definition of the strategies related to the Security Orchestrator are done by this specific submodule "Security Resource Planning" Figure 29. This submodule is in charge of efficiently selecting the available security enablers to meet the required MSPL security policies. Mainly, two strategies have been developed; the security enablers selection through Resource Planning algorithms and Service function routing and placement through dynamic Service Function Chain (SFC) requests placement. This module can be extended and reinvented indefinitely by introducing and/or updating the algorithms.

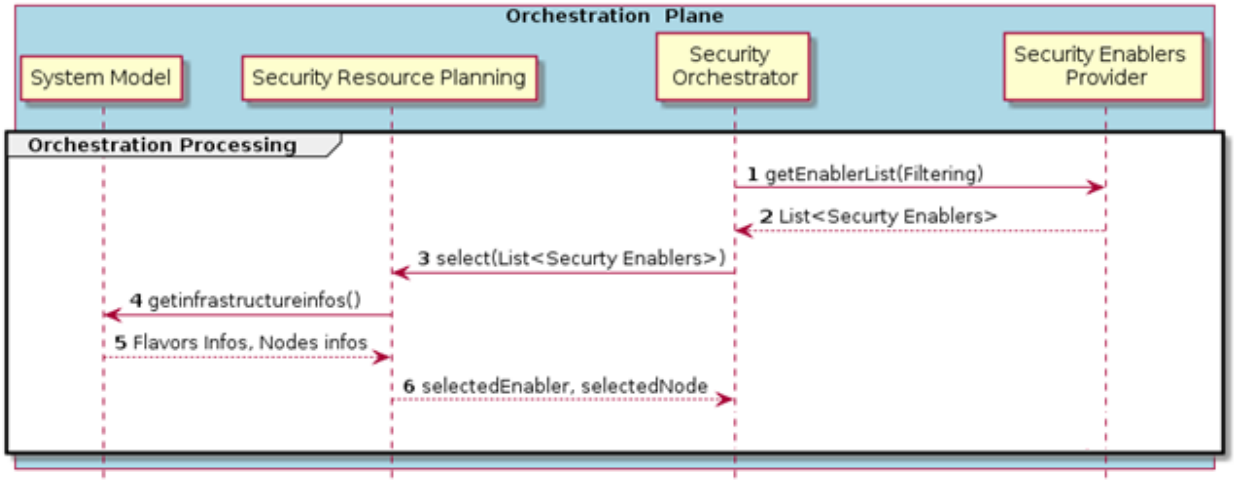


Figure 29. Security Resource Planning Module Interactions with the Security Orchestrator

Security Resource Planning	
Description	This submodule is in charge of efficiently selecting the available security enablers to meet the required MSPL security policies. This submodule will define and develop appropriate strategies to enforce security by exploring the SDN, NFV, and IoT technologies, accounting for the available resources in the underlying infrastructure and the policies requirements. In this vein, it provides functions like optimal path selection, load balancing, traffic rerouting, etc.

Inputs	System model
Outputs	Security orchestrator

4.3.7 System Model Service

The System Model is an independent component that provides swagger open API interface for managing and storing information about data plane that includes different enforced policies and network information. This data is made available to all of the ANASTACIA components in order to further refine/translate the security policies and improve the detection mechanisms. The data structure of the System Model Service is depicted in Figure 30.

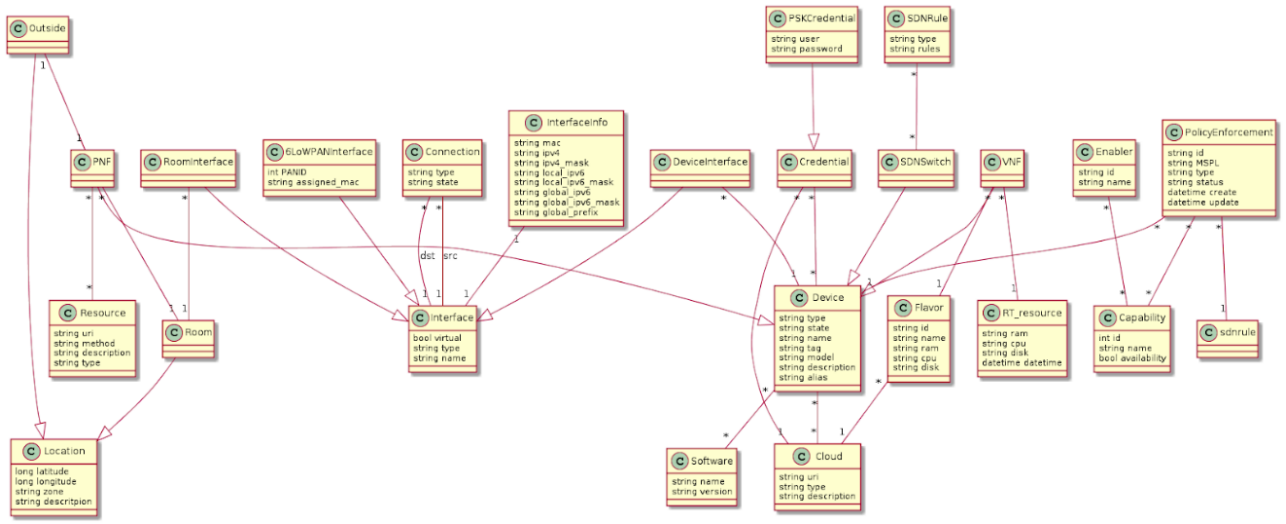


Figure 30. System Model Data Structure [8]

Depending on the type of the desired security policy, the System Model stores the relevant VNF details, SDN rules, IoT actions and related policies. More information about this component is available in [8].

System Model Service	
Description	This component is responsible on storing and managing information about different ANASTACIA components. It also gives these components the possibility to get more information about the status of other components and enforced policies.
Inputs	Receive different information from almost all ANASTACIA components using JSON format.
Outputs	Different ANASTACIA components also retrieve different information from the system model in a JSON format.

4.3.8 IoT, SDN, NFV Drivers

The mitigation action system leverage SDN and NFV enabling technologies to prevent the attacks efficiently. Different virtual security components would be instantiated and managed for preventing attacks using NFV technology. Also, the attacks could be filtered at the network level using SDN technology. Even more, the security orchestrator can mitigate the attacks at the source (affected IoT devices) thanks to the IoT controllers.

IoT Driver	
Description	This driver gives the possibility to the security orchestrator engine to communicate with the IoT controller at high-level of abstraction in independent way of the underlying technologies.
Inputs	A message received from the security orchestrator engine in a JSON format.
Outputs	A message that shows the status of enforced action in a JSON format.

SDN Driver	
Description	This driver gives the possibility for the security orchestrator engine to communicate with different SDN controllers in an abstract manner. Using this interface, the security orchestrator engine will be able to enforce different SDN rules in the network for redirecting, limiting, blocking or mirroring the traffic between different data plane components.
Inputs	A message received from the security orchestrator engine in a JSON format.
Outputs	A message that shows the status of enforced action in a JSON format.

NFV Driver	
Description	This driver enables the security orchestrator engine to communicate with different NFV orchestrator in an abstract manner. If the mitigation plan requires the instantiation of new VNFs, the security orchestrator instructs the NFV orchestrator to instantiate and configure the required VNFs.
Inputs	A message received from the security orchestrator engine in a JSON format.
Outputs	A message that shows the status of enforced action in a JSON format.

4.4 Dynamic Security and Privacy Seal

The component is built using several distributed processes interfacing with REST APIs, for simplicity they have been aggregated and described below as:

- DSPS-Server
  - Seal Creation service
  - Privacy-mapping service (OpenAPI)
- DSPS-Storage
  - Blockchain for Seals
  - Shamir Secure storage for Logs
- DSPS-Agent
- DSPS-User-Interface

The detailed architecture of the DSPS component can be found in D5.2 and the most up-to-date implementation details are in D5.3 [17].

4.4.1 DSPS Server

The DSPS Server is a core component for DSPS service that handles mapping with security alert to the seal logics, privacy risk mapping corresponding the security alerts received from SAS and management of agent key. It also controls overall service components such as LDAP, security sharing, communication with DSPS storage and GUI backend. The internals of this component are represented in Figure 31.

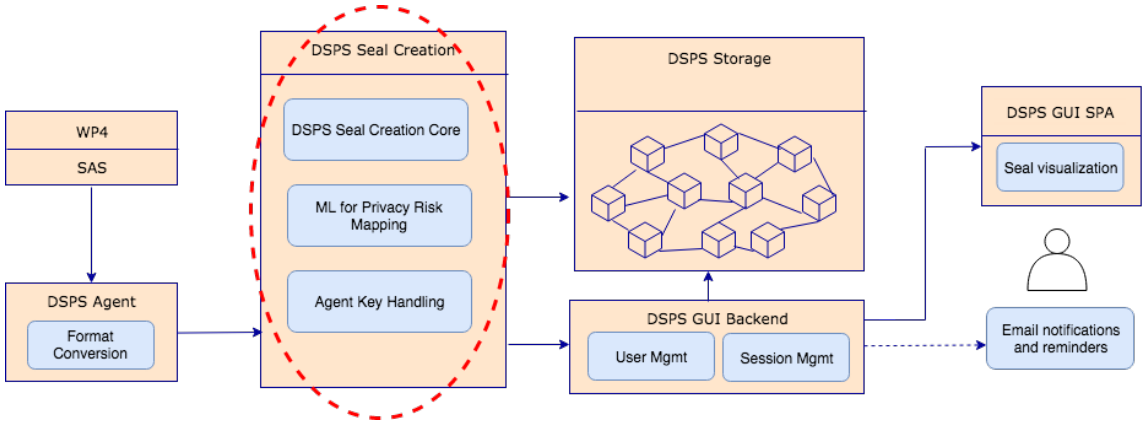


Figure 31. Internal architecture of the DSPS Server

Dynamic Security & Privacy Seal Server	
Description	The goal of this component is to manage new seals (JSON objects) given another object containing some information related to the new seal (called log) and given authorisation credentials. The log can be any kind of data but must contain some necessary information used to forge a new seal. The seal is then signed with the key of the Agent that belongs to the organisation the seal is referring to. The signed seal is then stored in the blockchain while the log is encrypted, split and put in the Shamir storage.
Inputs	Messages (alerts and mitigations) in STIX format, extra information necessary to make a seal may be contained also
Outputs	Sends Seal and associated Logs about privacy and security state of organisation to secured storage and GUI.

4.4.2 Dynamic Security & Privacy Seal Agent

DSPS Agent, highlighted in Figure 32, takes in charge of communication with SAS to receive security alerts and mitigation information of the alerts. It converges the received data into STIX format.

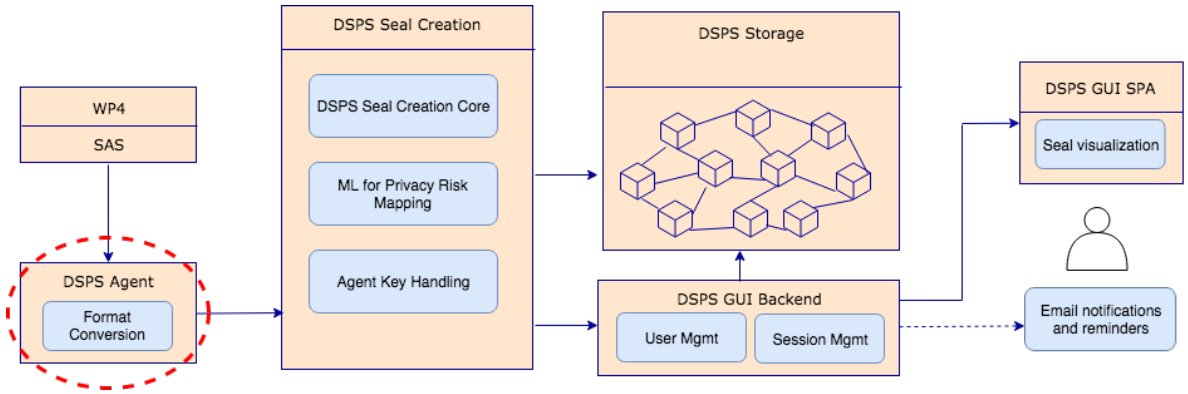


Figure 32. Internal architecture of the DSPS agent

Dynamic Security & Privacy Seal Agent	
Description	The DSPS-Agent is a component of the ANASTACIA framework. It receives and process alerts and mitigations from the ANASTACIA Alert Service and then it sends to the DSPS-Server(s) - using the standardised alert format STIX- the necessary information to make the seal.
Inputs	Messages describing security alerts and mitigations.
Outputs	Messages (alerts and mitigations) in STIX format, extra information necessary to make a seal may be added.

4.4.3 Dynamic Security & Privacy Seal User Interface

DSPS User Interface, highlighted in Figure 33, provides different types of user management with their credential, session management and functional support for DPOs and CISOs activities. It also provides human intuitive and user-friendly seal visualization.

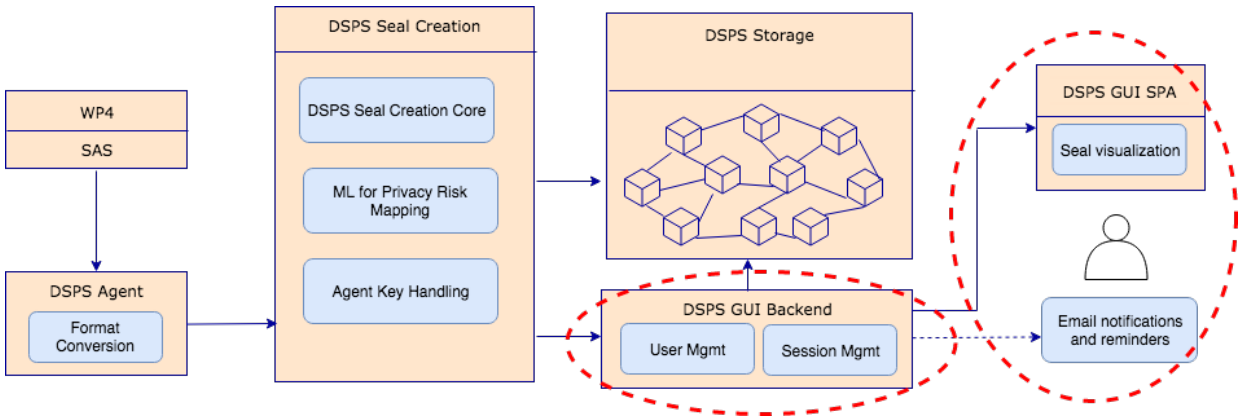


Figure 33. Internal architecture of the DSPS User Interface

Dynamic Security & Privacy Seal User Interface	
Description	The DSPS Graphical User Interface aims to fulfil the double goal of enabling end user's access to the DSPS information in an easy to understand and trustworthy manner. Additionally, the GUI introduces a decentralized tool for verification of the information received from the

	DSPS blockchain log and the distributed validation of DSPS blocks. Through these two goals, the end-user adopts an active role in the verification of the privacy and security of the deployed systems, and ultimately strengthening the DSPS's position as a trust-enhancing tool.
Inputs	Seal and seal log data from the DSPS Seal Creation Service and the DSPS Storage; user inputs from users (DPO, CISO, Administrator).
Outputs	Graphical and textual representations of seal and seal log data.

5 MAIN INTERFACES

This section describes the detail of the main interfaces used to exchange information between the planes. The interfaces described in this section are represented in Figure 34.

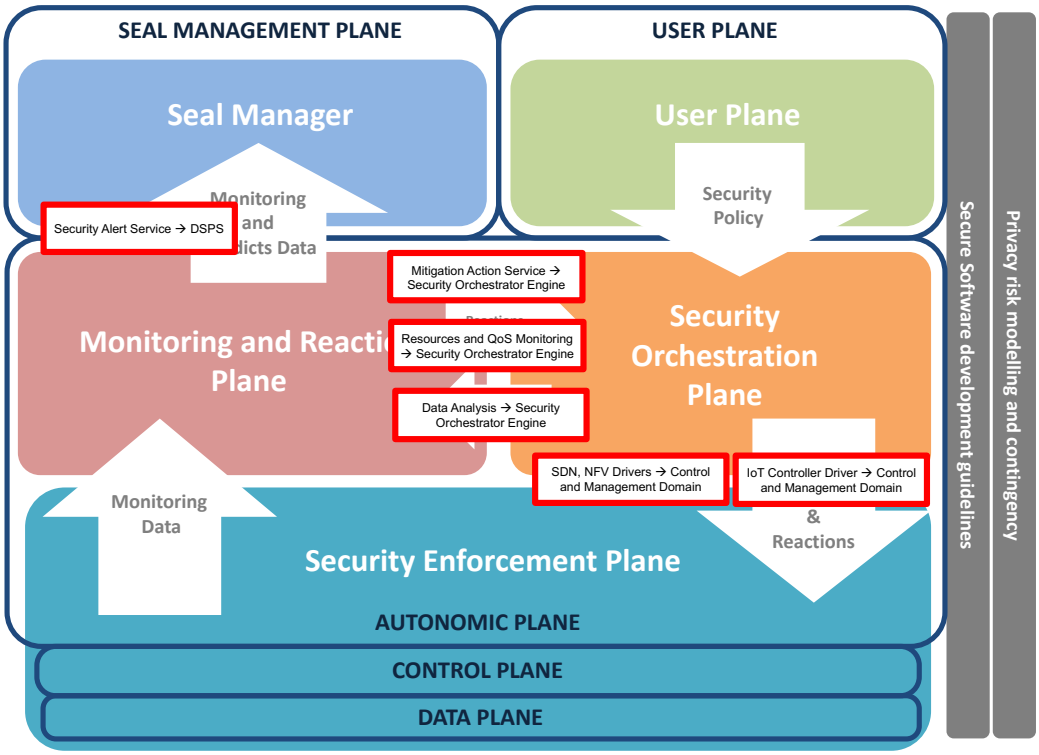


Figure 34. Main interfaces represented over the high level architecture

The interfaces between components of the same plane are not described here as they are described in D6.4 [15].

5.1 REACTION – ENFORCEMENT

Table 11. Mitigation Action Service - Security Orchestrator Engine Interface detail

Mitigation Action Service -> Security Orchestrator Engine	
Purpose	Two principal interfaces have been defined between these two components: • MAS – System Model: Intended to allow the MAS to access the System Model information that is used to generate the MSPL containing the mitigations to be deployed by the Security Orchestrator. • MAS – Security Orchestrator Engine: Intended to allow the MAS to send the final set of mitigations to the Security Orchestrator. The latter will use these mitigations (in form of MSPL) to execute the changes on the network and mitigate the detected threat.
Technology	• REST API – Swagger-based endpoint (MAS – System Model) • REST API – HTTP server endpoint (MAS – Security Orchestrator Enforcement)
Available functions	Two principal functions are available with this interface:

	Submit Reactive Policies for Enforcement (/enforce): • POST: The MAS sends the reactive security policy using the HTTP-based REST API using a POST request to the Security Orchestrator. **Retrieve System Model Information (multiple endpoints):** GET: The MAS sends GET requests to the swagger AI in order to retrieve information about the System model from multiple endpoints exposed by the Security Orchestrator.

5.2 MONITORING – ENFORCEMENT

Table 12. Interface Resource and QoS Monitoring and Security Orchestrator Engine

Resource and QoS Monitoring -> Security Orchestrator Engine	
Purpose	The security orchestrator optimizer for taking the right decisions will use the received information from the Resource and QoS Monitoring system.
Technology	Celery Queue broker
Available functions	not applicable

Table 13. Data Analysis and Security Orchestrator Engine interface detail

Data Analysis -> Security Orchestrator Engine	
Purpose	Dynamic UTRC Data Analysis Agent configuration to enable/disable given sensor(s) monitoring – verdict generation in case of enablement.
Technology	REST
Available functions	**GET**, **PUT** https://10.79.7.172/agent/mspl/ - enables security orchestrator to get or set current MSPL agent monitoring policy **GET** https://10.79.7.172/agent/status/ - returns current state of the UTRC Data Analysis Agent

5.3 MONITORING AND REACTION – DSPS

Table 14. Security Alert Service to DSPS agent interface detail (CNR, MAND)

Security Alert Service -> Dynamic Security and Privacy Seal	
Purpose	Share with the DSPS the status of the Reaction procedure. In particular, three different information are forwarded to the DSPS: 1. Alerts: the SAS forward the alerts received from the VDSS

	2. Alerts + MAS data: the SAS enriches the alerts with the mitigations data received from the MAS 3. Alerts + MAS + SM data: the SAS polls the System Model to obtain the status of the mitigation.
Technology	AMQP (RabbitMQ queue)
Available functions	**POST:** http://10.79.7.176:8080 Adopted by the MAS to send information to the DSPS
	GET: http://10.79.7.175:9000/api /policyenforcement/ Adopted by the SAS to get the information about the status of the mitigation

5.4 ORCHESTRATION – IOT INFRASTRUCTURE

Table 15. SDN/NFV drivers and control management domain interface detail

SDN, NFV Drivers -> Control and Management Domain	
Purpose	For what is this interface and what information is exchanged
Technology	Celery Queue broker
Available functions	not applicable

Table 16. IoT Controller Driver - Control and Management Domain interface detail

IoT Controller Driver -> Control and Management Domain	
Purpose	The interface exposes different services in order to manage resources and behaviour of the IoT devices.
Technology	REST
Available functions	Receive HTTP enforcement requests and enforce the configurations by using the specific protocols of the affected devices, e.g. CoAP, MQTT… **IoT Register (event subscription)** - Once an IoT device has been properly authenticated in the system the IoT Register receives the event and it registers the IoT device in the system model (or it updates the information) as well as it requests the default policies enforcement for the registered device. **Resource Management (/api/resources)** - GET: It allows to retrieve information of the available resources of the IoT devices. - POST: It allows to enforce the IoT configuration by using the available resources of the IoT devices, e.g., Power management resource.

	IoT Honeynet Model (/api/honeynet) - GET: It allows to retrieve honeynet information in order to build IoT honeynet MSPL policies.

6 CONCLUSIONS

This deliverable has presented the final results of the task T1.3, which objective was the design of the ANASTACIA architecture. This deliverable evolves from the deliverable D1.3 reported in M9 which introduced the initial architecture and the design methodology followed. During the rest of the project the ANASTACIA architecture has evolved thanks to the feedback received from the development of the framework, from the EC reviews, end-user validation carried out through questionnaires in WP7 and from the Advisory Board recommendations.

To this end, this document details the final processes carried out by the ANATACIA framework, which includes the activities related to policies definition and enforcement, monitoring, data acquisition and analysis, incident detection, decision support for reaction, reaction enforcement, and DSPS activities.

The final architecture is also presented, detailing the internals of the main subcomponents of the architecture, characterized by six planes:

- Data plane, where IoT devices are running and providing information about their activities.
- Monitoring and Reaction plane, where monitoring data is interpreted for the detection of incidents and the decision on mitigations to apply
- Security orchestration plane, where security policies are defined, reactions are triggered and NFV/IoT Controllers are orchestrated.
- Security Enforcement Plane, where reactions are deployed directly on IoT devices or the underlying infrastructure.
- Seal Manager plane, where the general status of the ANASTACIA platform and the IoT infrastructure is tracked and evaluated.
- User plane, where CISO interaction is managed.

For every plane, a set of subcomponents are defined in detail in this deliverable, used as a high-level overview that is further detailed in WP3, WP4 and WP5 deliverables.

Finally, this deliverable provides with an overview of the main interfaces that allows the exchange information between planes. As a high-level overview, the specific details about these interfaces and its usage in integration activities are given in WP6.

In summary, task 1.5 has provided with a comprehensive analysis of information coming from different sources (from developers to experts' feedback to name just a few) to produce with a consistent, robust and at the same time flexible architecture, which can be easily extended to incorporate additional security probes, new agents and easily adaptable to different types of IoT/CPS infrastructures. From the early results produced in D1.3 (highlighting a conceptual model that has guided the rest of the design activities, together with the analysis of use cases, requirements and threat analysis produced in WP1 and WP2), to the consolidated architecture reported in D1.5 (which main input has been derived not just from the activities carried out during the first period of the project, but also from the technical activities and feedback received during the second period), task 1.3 has successfully achieved all its expected objectives as planned at the beginning of the ANASTACIA project.

REFERENCES

[1]. ANASTACIA Deliverable 1.3 Initial Architectural Design. 2017
[2]. ANASTACIA Deliverable 1.4 Final User-centred Requirement Analysis. 2018
[3]. ANASTACIA Deliverable 2.5 Policy-based Definition and Policy for Orchestration Final Report. 2018
[4]. ANASTACIA Deliverable 2.6 Attack Threats Analysis and Contingency Actions Final Report. 2019
[5]. ANASTACIA Deliverable 2.7 Privacy Risk Modelling and Contingency Final Report. 2019
[6]. ANASTACIA Deliverable 2.8 Secure Software Development Guidelines Final Report. 2019
[7]. ANASTACIA Deliverable 3.4 Final Security Enforcement Manager Report. 2019
[8]. ANASTACIA Deliverable 3.5 Final Security Orchestrator Report. 2019
[9]. ANASTACIA Deliverable 3.6 Final Virtual Resources Manager Report. 2019
[10]. ANASTACIA Deliverable 4.1 Initial Monitoring Component Services Implementation Report. 2018
[11]. ANASTACIA Deliverable 4.3 Initial Agents Development Report. 2018
[12]. ANASTACIA Deliverable 4.4 Final Monitoring Components Services Implementation Report. 2019
[13]. ANASTACIA Deliverable 4.5 Final Reaction Component Services Implementation Report. 2019
[14]. ANASTACIA Deliverable 4.6 Final Agents Development Report. 2019
[15]. ANASTACIA Deliverable 5.1 Dynamic Security and Privacy Seal Model Analysis Report. 2017
[16]. ANASTACIA Deliverable 5.2 Dynamic Security and Privacy Seal Monitoring Service Demonstrator. 2018
[17]. ANASTACIA Deliverable 5.3 Dynamic Security and Privacy Seal User Interface. 2019
[18]. ANASTACIA Deliverable 6.4 Final Technical integration and validation Report. 2019
[19]. ANASTACIA Deliverable 6.5 Final Use cases implementation and tests Report. 2019
[20]. ANASTACIA Deliverable 6.6 Final End-user validation and evaluation Report 2019