ANASTACIA

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

# D4.2

## Initial Reaction Component Services Implementation Report

| | |
|---|---|
| **Distribution level** | CO |
| **Contractual date** | 31.10.2018 [M22] |
| **Delivery date** | 31.10.2018 [M22] |
| **WP / Task** | WP4 / T4.2 |
| **WP Leader** | MONT |
| **Authors** | I. Vaccari (CNR), E. Cambiaso (CNR), E. Punta (CNR), S. Scaglione (CNR), D. Rivera (MONT), R. Trapero (ATOS), P. Sobonski (UTRC), D. Belabed (THALES), A. Molina Zarca (UMU) |
| **EC Project Officer** | Carmen Ifrim<br>carmen.ifrim@ec.europa.eu |
| **Project Coordinator** | Softeco Sismat SpA<br>Stefano Bianchi<br>Via De Marini 1, 16149 Genova – Italy<br>+39 0106026368<br>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.

# Table of contents

# Table of figures

# PUBLIC SUMMARY

This document is the second deliverable of the WP4 – Monitoring and Alert/Reacting Enablers. It is focused on the Reaction Module of ANASTACIA platform. In this deliverable, the functionalities of the Reaction Module are described and analysed. The deliverable also describes the actual state of the component of the Reaction Module and the internal communications between the components in order to define a set of countermeasures to mitigate the threats detected by the Monitoring Module.

Firstly, the Security Alert Service is analysed. The aim of this component is to retrieve alerts from the Verdicts and Decision Support System (VDSS), analyse and enrich them with further information retrieved by the Mitigation Action Service (MAS), and send the expanded output to the seal management plane (DSPS). Secondly, the Verdicts and Decision Support Service (VDSS) is considered. It is the module that evaluates the threats detected by the Monitoring module and decides, upon an analysis of the available security capabilities, the possible countermeasures to adopt, among the ones that are supported by the ANASTACA framework. Thirdly, the Mitigation Action Service (MAS) is in charge of processing the decisions taken by the Verdict and Decision Support System and generate the appropriate MSPL file that specifies the countermeasures to be deployed. The MSPL is sent to the Orchestrator Plane to deploy the countermeasures selected. Finally, the Security Model Analysis is implemented to store all the capabilities of the platform and to share the database with the VDSS.

Finally, a section describing the innovative scientific and technological aspects is reported, in order to better explain the functionalities of the Reaction Module. In this context, the Reaction Module represents a particularly innovative component of the system, due to the fact it's able (i) to make autonomous decisions, without human intervention, (ii) to deploy reactions in (almost) real-time, and (iii) to manage large numbers of detection events. Because of this, and because of the importance of this component for the ANASTACIA architecture, the Reaction Module represents a crucial element of the system, implemented through security-by-design approaches and characterized by innovative aspects.

# 1 INTRODUCTION

## 1.1 AIMS OF THE DOCUMENT

This document describes the second deliverable of WP4, which contains detailed information on the implementation of the ANASTACIA framework Reaction Module. Section 2 introduces the current status of the ANASTACIA framework architecture related to the Monitoring and Reaction plane. Section 2.1 presents the general architecture of the ANASTACIA Reaction Module. Section 2.2 describes the module developed by explaining in detail each component and the flow of information exchanged. Section 3 and 4 show a summary of internal and external communication. Finally, Section 5 presents the conclusions of the document.

## 1.2 APPLICABLE AND REFERENCE DOCUMENTS

This document refers to the following documents:

- D1.3 – Architecture Design;
- T2.2 – Attacks and Threats Analysis and Contingency Actions;
- MS12b – Reaction component services specified and agreed by the board;
- D4.1 – Initial Reaction Component Services Implementation Report.

## 1.3 REVISION HISTORY

| Version | Date | Author | Description |
|---|---|---|---|
| **0.1** | 28/08/2018 | I. Vaccari (CNR) | Initial draft |
| **0.2** | 04/09/2018 | E. Punta (CNR) | First revision |
| **0.3** | 12/10/2018 | I. Vaccari (CNR), E. Cambiaso (CNR) | Integration of contributions received from partners |
| **0.4** | 15/10/2018 | E. Cambiaso (CNR), S. Scaglione (CNR) | Minor fixes |
| **0.5** | 17/10/2018 | Alejandro Molina Zarca (UMU) | Added information related to the available capabilities database |
| **0.6** | 22/10/2018 | Diego Rivera (MONT) | Added information for the innovative aspects |
| **0.7** | 25/10/2018 | I. Vaccari (CNR), E. Cambiaso (CNR) | Production of the final version of the document |

## 1.4 Acronyms and Definitions

| Acronym | Meaning |
|---|---|
| **SAS** | Security Alert Service |
| **VDSS** | Verdicts and Decision Support System |
| **MAS** | Mitigation Action Service |
| **DSPS** | Dynamic Security and Privacy Seal |
| **SO** | Security Orchestrator |

# 2 THE ANASTACIA REACTION COMPONENT

One of the most interesting aims of ANASTACIA framework is to implement a strategy able to react when threats are made to the system. The framework is composed by two different modules. The monitoring module, described in D4.1, is developed in order to detect when a threat is attacking the system.

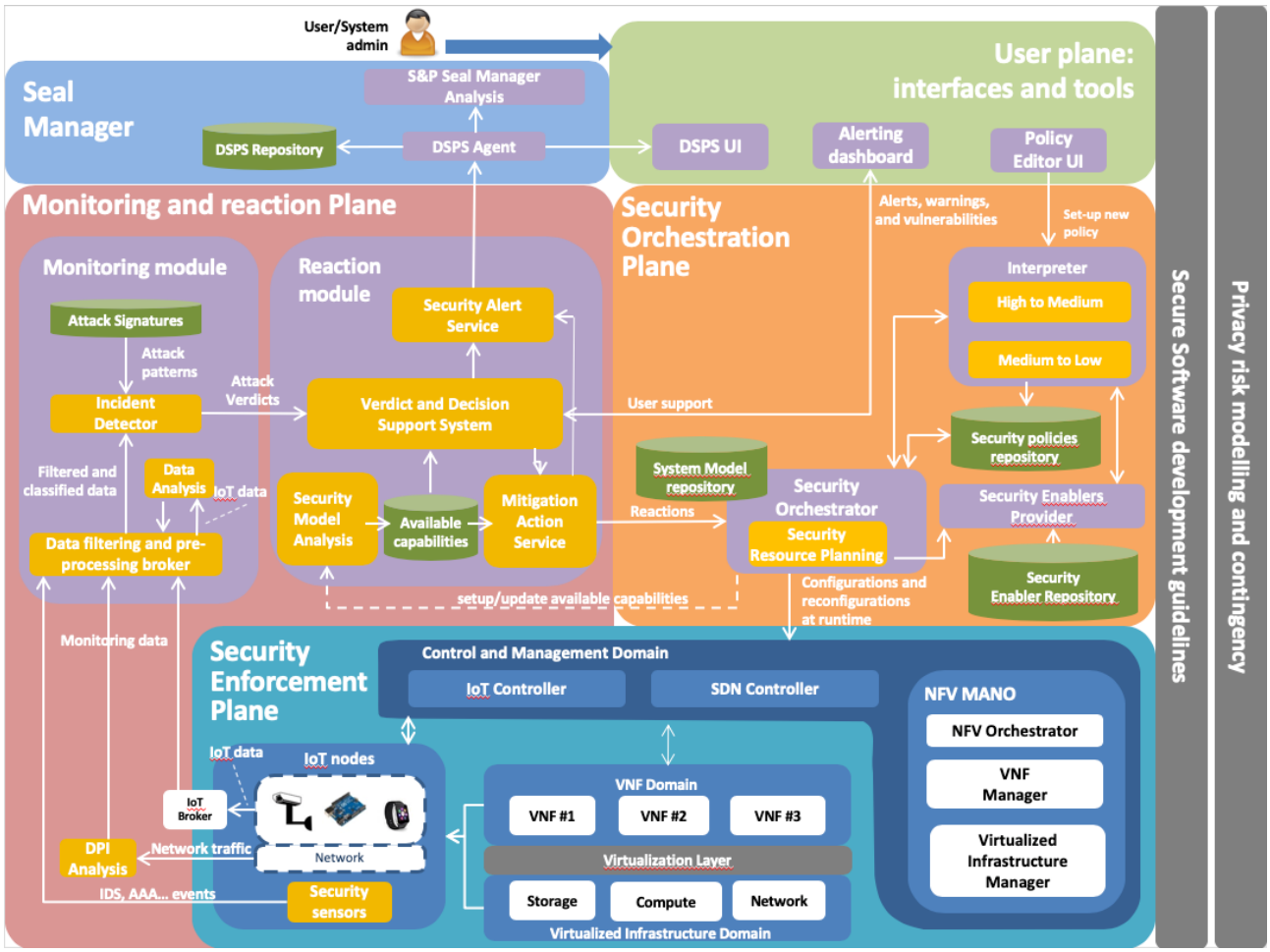


Figure 1 - ANASTACIA Framework

As can be seen from Figure 1, the Reaction module is directly connected to the Monitoring module, the Seal Manager and the Security Orchestrator Plane. In collaboration with the Monitoring module, the aim of the Reaction module is to provide the countermeasures that could be adopted by the system in order to mitigate the detected attack and to share the defined mitigation plane with the other module of the framework in order to protect the system from threats.

In the next sections, the current status of each component of the Reaction Module is described in detail.

## 2.1 ARCHITECTURE

The first version of the architecture, defined in the deliverable D1.3, was used as a basis for starting the design of the individual components and the iterations that the components have internally and externally with other modules of the framework. The main idea is to develop a lightweight and reliable module that quickly selects the various countermeasures that can be implemented when a threat is identified by the Monitoring module to prevent the damage caused to the system to increase dramatically.

In Figure 2 is reported the actual version of the Reaction Module, after some revision activities adopted to refine the architecture.

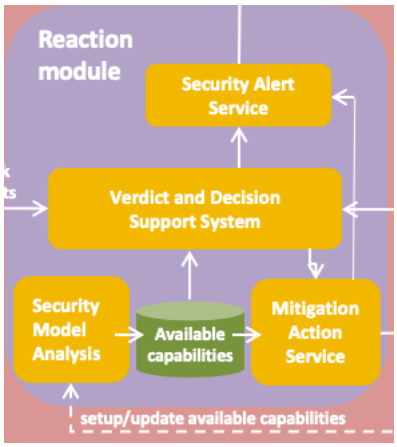


Figure 2 - Reaction module

In the current version of the architecture, the Reaction Module consists of the following components:

- Security Alert Service: this component is adopted to retrieve alerts form the VDSS component, analyse and enrich them with further information (retrieved by the MAS), and send the expanded output to the seal management plane (DSPS);
- Verdict and Decision Support System: this component is in charge to evaluate threats detected at the Monitoring module and decide the most convenient countermeasure among the ones that are supported by the platform;
- Security Model Analysis: This component continuously collects and updates the existing capabilities and their related threats and costs from the Security Orchestrator.
- Available capabilities database: database used to store the capabilities of the ANASTACIA framework and to share the information with other component of the Reaction module
- Mitigation Action service: This component is in charge of analysing the outputs of the Verdict and Decision Support System and transform these alerts into a MSPL file that specifies the countermeasure that as to be deployed to respond against the detected attack.

## 2.2 IMPLEMENTATION

The implementation phase was carried out in several steps, to allow partners to implement and refine the logic of the components in an accurate and precise manner. Initially, internal components and interfaces were developed to connect the various modules inside the Reaction Module. Subsequently, the Reaction module was interfaced with the other modules of the framework to send the processed information.

The current version of the module is able to process the alert received from the Monitoring Module containing information on the identified threat. Subsequently, it recovers from the various modules the possible countermeasures to be applied, in terms of policy, and sends this information to the modules responsible for applying the countermeasures on the system in order to protect it against attacks.

The countermeasures currently available on the system are those defined in T2.2 for the use cases under analysis. In the next versions they will be updated and expanded to help protect the system from more threats.

### 2.2.1 Security Alert Service

The Security Alert Service is a component of the Reaction module used in this initial phase to integrate information from the other components of the module, group them together and send to the **DSPS**.

**SAS** module chains the detailed information related to the threat detected with the policies that the other components of the module have selected to protect the system from attacks. Using this module will therefore lead to a link between the identified threat and the selected protection system. The module communicates with the components as shown in Table 1.

| Abbreviation | Alias | Component | Developing partner | Communication |
|---|---|---|---|---|
| **VDSS** | XL-SIEM | Verdicts and Decision Support System | ATOS | Dedicated RabbitMQ queue |
| **MAS** | - | Mitigation Action Service | MONTIMAGE | Web Server |
| **DSPS** | - | Dynamic Security and Privacy Seal | DG, MANDAT, ARCHIMEDE | Dedicated RabbitMQ queue |

Table 1 - Communication of the Security Alert Service

In particular, the communication between the **SAS** and **VDSS** modules is implemented on a dedicated RabbitMQ queue where the **SAS** component listens to the queue waiting for a threat identified by the Monitoring module and transmits it to the **VDSS**.

After receiving detailed information from the **VDSS** about the identified threat, the **SAS** component waits until all other components of the module perform their activities to identify and select the countermeasures necessary to mitigate the threat. When the activities are completed, the MAS sends the selected policies through a Web Server.

The SAS then combines the threat information with the policies that the other components of the module have selected and sends the combined information externally, via another dedicated RabbitMQ queue, to the Dynamic Security and Privacy Seal. In fact, the communication based on the two dedicated RabbitMQ is implemented with SSL encryption, whereas instead the Web Server is only an http connection. Next release will implement SSL.

The SAS module then combines the alert received from the VDSS with the policies from the MAS producing an output for the DSPS module with the format reported below.

```
{
    <alert data, as received by the VDSS component>,
    'status_complete': <a boolean value specifying if the alert is enriched or not with the data received by the MAS component>,
    'policy': <policy data, as received by the MAS component (optional, only if status_complete is true)>,
    'request_id': <deployment identifier, as received by the MAS component (optional, only if status_complete is true)>
}
```

## 2.2.2 Verdicts and Decision Support Service

The Verdicts and Decision Support Service (VDSS) is in charge of evaluating the incident detected at the Monitoring module and decide, upon an analysis of the available security capabilities, on the most convenient countermeasure among the ones that are supported by the IoT infrastructure. The VDSS is the consumer of the Incident Detector at the Monitoring module, which exports the incidents detected to the VDSS. Those incidents are evaluated based on the information provided by the Security Model Analysis (SMA). The result of the evaluation is distributed to the consumers of such information (namely the Security Alert Service, SAS, and the Mitigation Action Service, MAS).

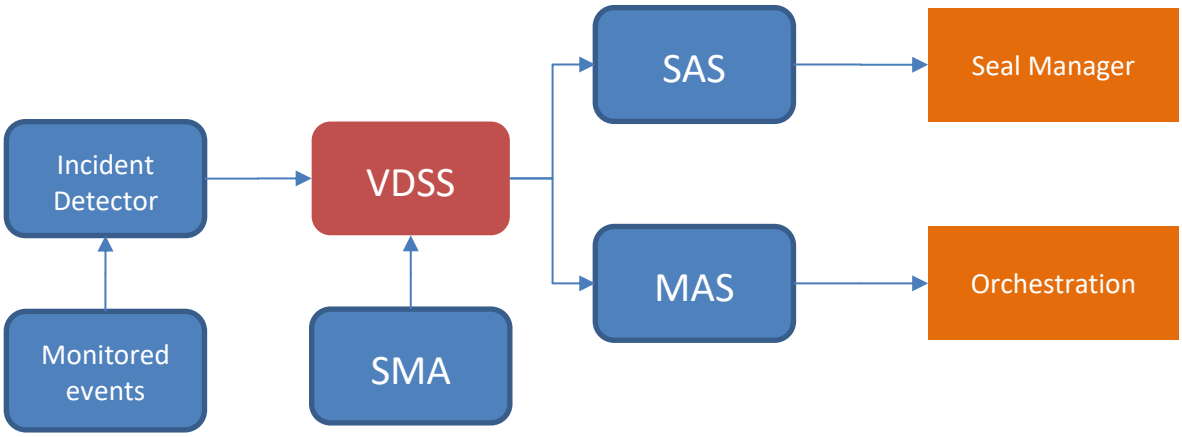


Figure 3 - VDSS relationships

The VDSS component is built upon several subcomponents (see figure below).

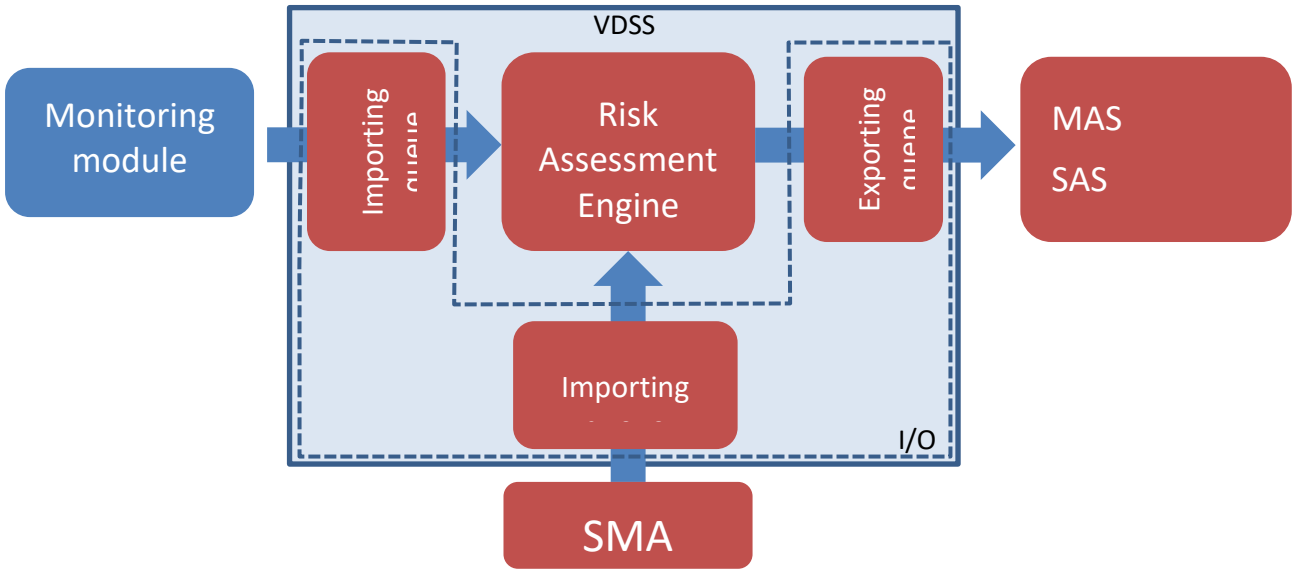


Figure 4 - Internals of the Verdict and Decision Support Server

The core of the VDSS is the Risk Assessment Engine. Two main parts supports it:

- I/O: Infrastructure for exchanging information between the VDSS and other components of the ANASTACIA infrastructure. Two sources are used as input for the Risk Assessment Engine:
  - Incidents detected at the monitoring module. The monitoring module correlates monitoring events and reports about incidents detected. A Storm-DRPC[1] based mechanism is used to report these incidents to the Risk Assessment Engine.
  - Information about mitigations and capabilities. The Security Model Analysis (SMA), using information retrieved from the Orchestrator, reports to the VDSS about the available mitigations that are supported by the IoT infrastructure according to the security policy and the capabilities enforced. A RabbitMQ messaging queue is used to share this information with the VDSS. Details will be given below.

  The output of the VDSS is consumed by the MAS and the SAS. The MAS will use the VDSS output to create a MSPL message to be interpreted by the Orchestration to enforce the Mitigation. The SAS adapts the VDSS output to be processed by the Seal Manager module. The output of the VDSS are

[1] http://storm.apache.org/releases/1.1.2/Distributed-RPC.html

alerts about incidents detected and mitigations proposed. The VDSS produces security alerts, which include information about the severity of the incident detected. This information is shared through a RabbitMQ messaging queue. Details will be given below.

- Risk Assessment Engine: component in charge of evaluating the incidents detected at the monitoring module and the mitigations supported by the orchestrator. The purpose of the Risk Assessment Engine is to evaluate the trade-off between mitigating the incident and the cost associated to it. The cost is understood here as the resources required to enforce a mitigation, considering resources either computational, human, time resources, complexity, time to mitigate, time that the service need to be up and running again, and, ultimately, monetary costs. All those factors are evaluated using a mathematical model that produces a list of recommended mitigations, including details about what are the best ones in terms of cost. It is up to the system admin decision to automatically enforce the best possible mitigation or to choose among the list of mitigations, evaluating the scores and information produced by the risk assessment engine.

The I/O has been completely developed during the first period, while the Risk Assessment Engine has produced with a very simple functionality that will be improved during the second period. The following subsection describes the details of the I/O components of the VDSS.

**Implementation details for I/O queues**

A set of input and output elements were developed to feed the Risk Assessment Engine. The development was divided into two parts:

- Input from the Monitoring module. The VDSS receives information about incidents detected by the Monitoring module. The monitoring module and the VDSS are very deeply integrated based on Apache Storm clusters. Several storm workers are deployed (which might be running in different machines, thus improving scalability, performance and flexibility). The Incident detector, at the Monitoring module, deploys several correlation bolts in charge of correlating the events received from the agents that are monitoring the security probes available at the IoT infrastructure. These events are received by this correlation bolt through a storm spout, which is a source of stream used to transfer data in the form of tuples (this is, pairs of values representing a label and value). The functionality provided by the correlation bolt can be distributed between different storm workers, which might be running in different machines. The result of the correlation is passed as a storm spout, by using a storm-drpc server, to another correlation bolt (called actionBolt) in charge of preparing the data to be processed by the risk assessment engine. Again, this actionBolt can be processed by one or more workers which can be running in different machines, thus sharing computational resources. The internals of this are represented in Figure 5.

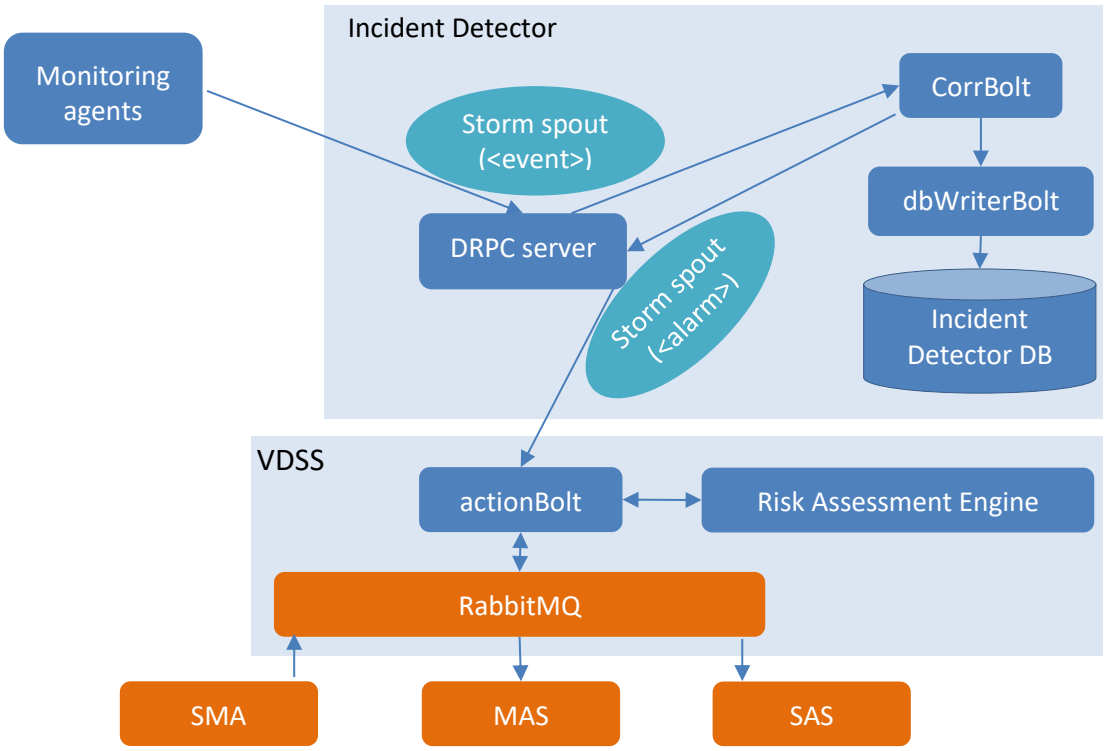


Figure 5 - Details of the VDSS inputs from the Incident Detector

- Input from SMA, Output to MAS. The VDSS uses a different mechanism to communicate with the SMA and MAS. The main reason for this is due to the deep integration of the VDSS within the Incident detector, which are both integrated in a storm topology that is part of the Atos XL-SIEM asset. The VDSS manages the exchange of information between SMA and MAS through secure RabbitMQ queues. The VDSS uses a fanout schema for configuring the queues (Figure 6). This means that an exchange queue is created (X in the figure), where the messages are put but by the Producer (P). The exchange queue allows to be attached to different queues (amq.gen queues in the figure) which are created by Customers ($C_i$). The fanout schema broadcast the messages sent by the Producer to the Exchange queue to all the attached queues. The Exchange queue empties when the messages are broadcasted while the attached queues retain the messages till they are consumed by the consumers.

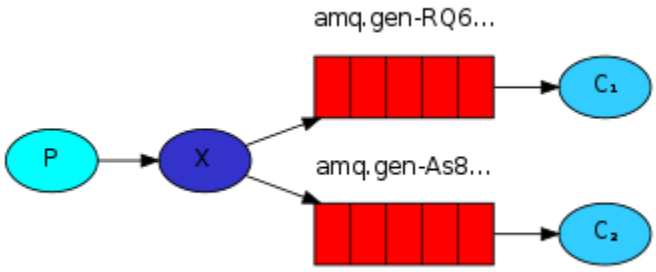


Figure 6 - RabbitMQ configured as fanout (source: https://www.rabbitmq.com)

This fanout schema has been used in Anastacia to control de input and output of the VDSS between the MAS and the SAS. A RabbitMQ server is running at the VDSS, which used by the, SAS, MAS and SMA to manage the exchange of information. Figure 7 represents the queues deployed in the Reaction module. Two exchange queues are configured:

  - The exchange queue *eu.anastacia_server_output* is used to push incident alerts to the attached queues:

- The queue *eu.anastacia.dw_input.cnr* is used by the SAS to read incident alerts and modify them to be sent towards the Seal Manager.
- The queue *eu.anastacia.dw_inputmontimage* is used by the MAS to read incidents and prepare MSPL files to be interpreted by the orchestrator for its enforcement.

- The exchange queue *Anastacia_SecurityModelAnalysis* is used by the SMA to push mitigations and capabilities supported by the platform. The *SMAtoVDSS* queue is attached to this queue to provide that information as input to the Risk Assessment Engine.

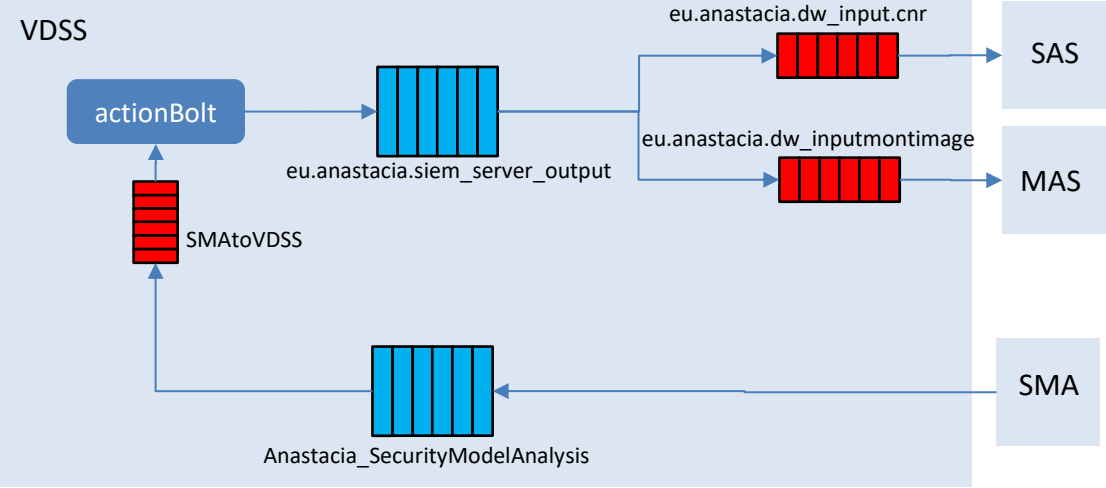


Figure 7 - Deployment of queues to manages I/O with VDSS

The following screen represents the RabbitMQ management pane, which shows the Exchange queues available (Figure 8) and the attached queues (Figure 9).

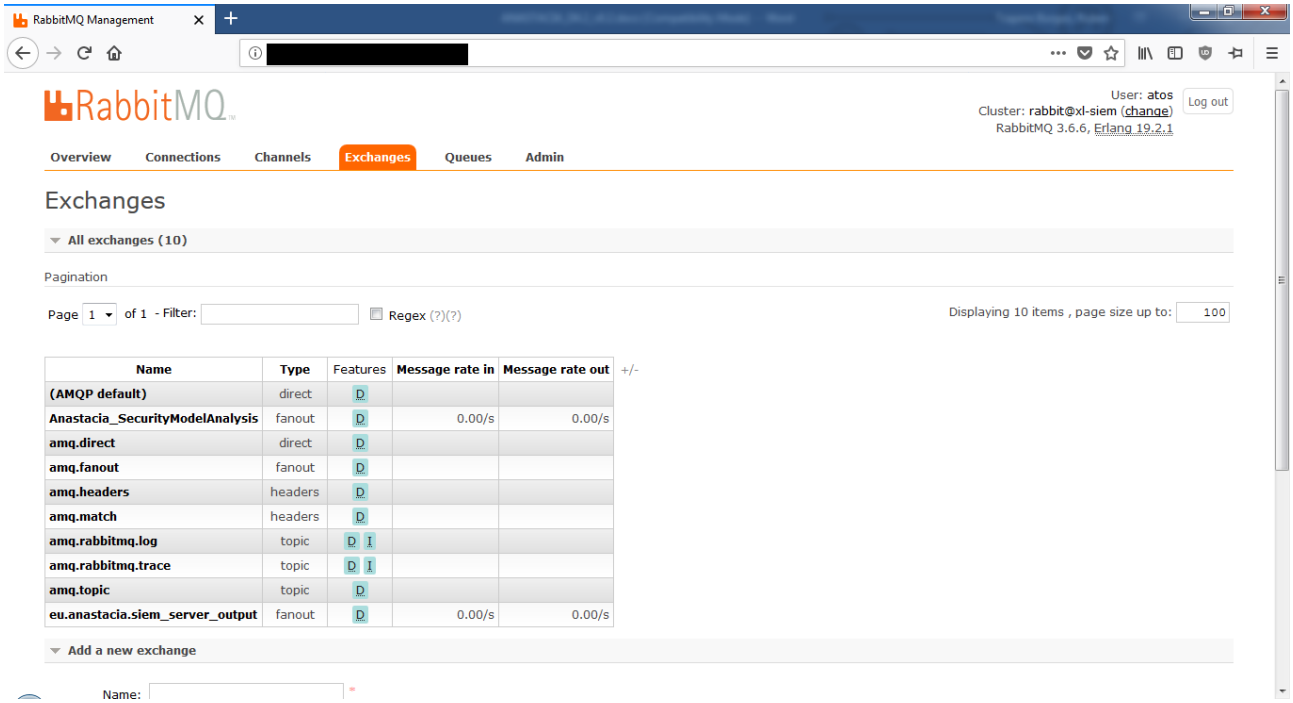


Figure 8 - Exchange queues at the RabbitMQ server

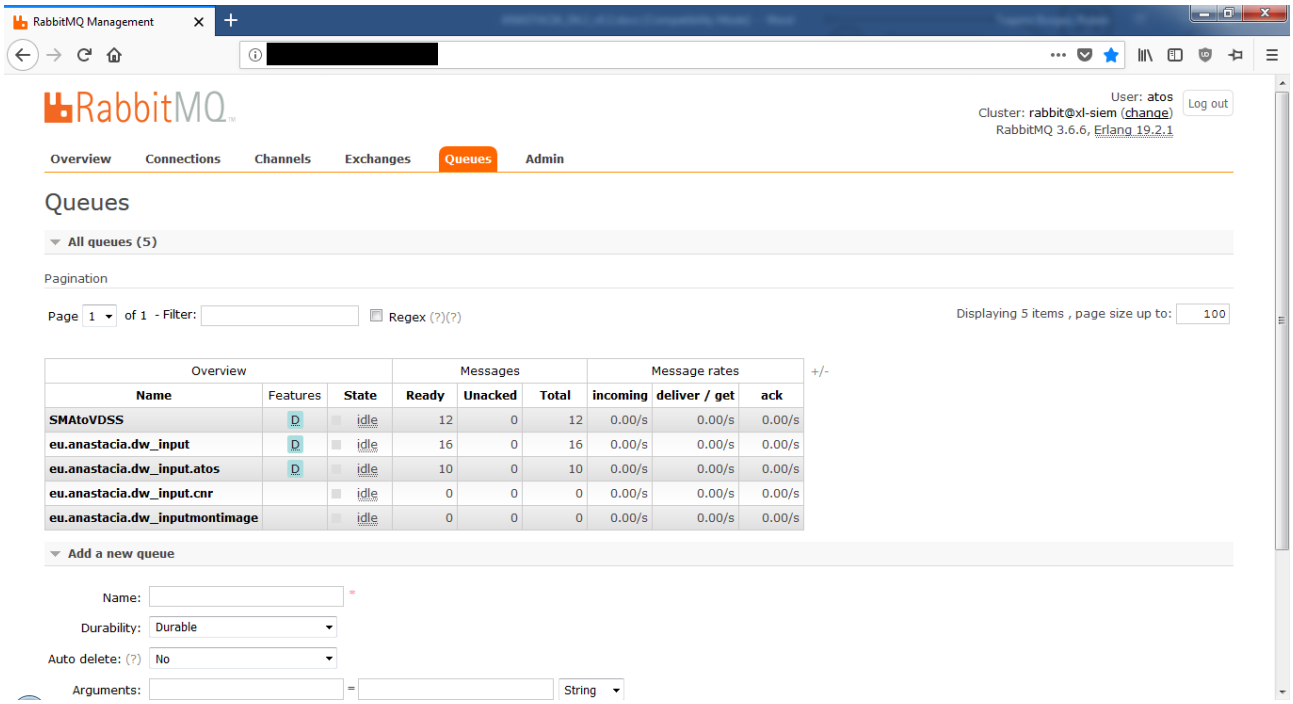


Figure 9 - Attached queues seen at the RabbitMQ server

### Incident alerts exported by VDSS

The VDSS exports security incident alerts to other components of the Reaction module. These alerts are exported to the queues specified in the previous subsection. The format of the message exported is very similar to the format used for events exported by the Monitoring module (see D4.1 for details). The message is a JSON message that includes:

- Information about the alert, including internal fields used by the VDSS (such as several IDs), and information that describe the alert such as the destination and source (IP and Port) of the incident, metrics (priority, reliability, risk) to evaluate the severity of the incident, textual descriptions of the alert, timestamp, category and subcategory of the alerted incident and several custom fields (USERDATA) with specific information about the alert.
- Embedded events that have been correlated to generate the alert. These embedded events follow the same JSON format as the one specified in D4.1 for the events exported by the monitoring module.

The following is an example of one of the alerts exported by the VDSS. It corresponds to a Forbidden Network Authentication which has been generated by correlating one event (identified as “a” inside the “RELATED_ELEMENTS_INFO” field). It is worth noticing that every event and alert is identified by the VDSS with a log. The alert specifies the events that have been correlated by including a list of event ids inside the “RELATED_EVENTS” field. The RELATED_ELEMENTS_INFO contains these events (also identified by their event_id field).


```
{"AlarmEvent":{
        "DST_IP_HOSTNAME":"00000000",
        "RELATED_EVENTS":"[b69f11e8a9b9080027ea052c27ca689c]",
        "DST_IP":"aaaa::2",
        "PLUGIN_NAME":"cyber-monitor",
        "SRC_IP":"aaaa::1",
        "PRIORITY":4,
        "RELIABILITY":6,
        "SUBCATEGORY":"Bruteforce",
        "USERDATA3":"",
        "USERDATA4":"",
        "PLUGIN_SID":"5",
        "USERDATA1":"PAA",
        "USERDATA2":"",
        "ORGANIZATION":"ATOS",
        "CATEGORY":"Authentication",
        "PLUGIN_ID":"70000",
        "USERNAME":"",
```

```
        "FILENAME":"",
        "BACKLOG_ID":"4fb66f54938f4c59be24e175c51b1e55",
        "RELATED_EVENTS_INFO":{
                "a":{
                        "date":"1536765531",
                        "plugin_id":31000,

                        "log":"IlNlcCAxMiAxNToxODo1MSAxOTIuMTY4LjU2LjEgW0FBQV0geyJzb3VyY2VfaXAiOiJhYWFhOjoxIiw
                gInNvdXJjZV9wb3J0IjoiNDAwMCIsImFmZmVjdGVkX2lwIjoiYWFhYTo6MiIsImFmZmVjdGVkX3BvcnQiOiI3MTYiLCJ0eX
                BlX29mX2RldmljZV9hZmZlY3RlZCI6IlBBQSIsImV2ZW50X3R5cGUiOiJuYSJ9ICI=",
                        "interface":"enp0s3",
                        "dst_ip":"aaaa::2",
                        "src_ip":"aaaa::1",
                        "userdata7":null,
                        "fdate":"2018-09-12 15:18:51",
                        "userdata8":null,
                        "userdata5":null,
                        "userdata6":null,
                        "userdata9":null,
                        "userdata3":null,
                        "userdata4":null,
                        "userdata1":"PAA",
                        "userdata2":null,
                        "src_port":null,
                        "plugin_sid":1,
                        "event_id":"b69f11e8a9b9080027ea052c27ca689c",
                        "filename":null,
                        "organization":"ATOS",
                        "dst_port":716,
                        "tzone":null,
                        "device":"10.0.2.4",
                        "username":null}},
        "PROTOCOL":6,
        "RISK":4,
        "SRC_PORT":0,
        "SENSOR":"",
        "SRC_IP_HOSTNAME":"00000000",
        "SID_NAME":"AAA Probe - Forbidden Network Authentication",
        "USERDATA7":"",
        "DATE":"2018-09-12 15:18:51",
        "USERDATA8":"",
        "USERDATA5":"",
        "USERDATA6":"",
        "PASSWORD":"",
        "USERDATA9":"",
        "DST_PORT":716,
        "EVENT_ID":"84dd346249904bfc89b1719a21bacb93"}}
```


The main fields used to trigger a suitable mitigation are the CATEGORY, SUBCATEGORY and SID_NAME. The SID_NAME contains the type of security incident detected while the category and subcategory classifies them. A list of pre-defined categories is available at the VDSS. Annex I lists them. It is expected that new categories and subcategories can be added in order to better fine tune the mitigation.

## 2.2.3 Mitigation Action Service

The Mitigation Action Service (MAS) is the implementation of the principal interface with the Security Orchestrator of the ANASTACIA platform. Considering this functionality, the MAS is the Reaction component that is in charge of processing the decisions taken by the Verdict and Decision Support System and generate the appropriate MSPL file that specifies the countermeasures to be deployed.

### 2.2.3.1 Mitigation Action Service Design

The Mitigation Action Service (MAS) was conceived as an always-running software that actively monitors the output of VDSS. It has been implemented in Java, with the support of the JSVC library[2], in order to provide continuously operating software rather than single execution code.

The design of the MAS is presented in Figure 10. It presents the principal classes that are used by this module, as well as the dependencies between them.

[2] https://commons.apache.org/proper/commons-daemon/jsvc.html

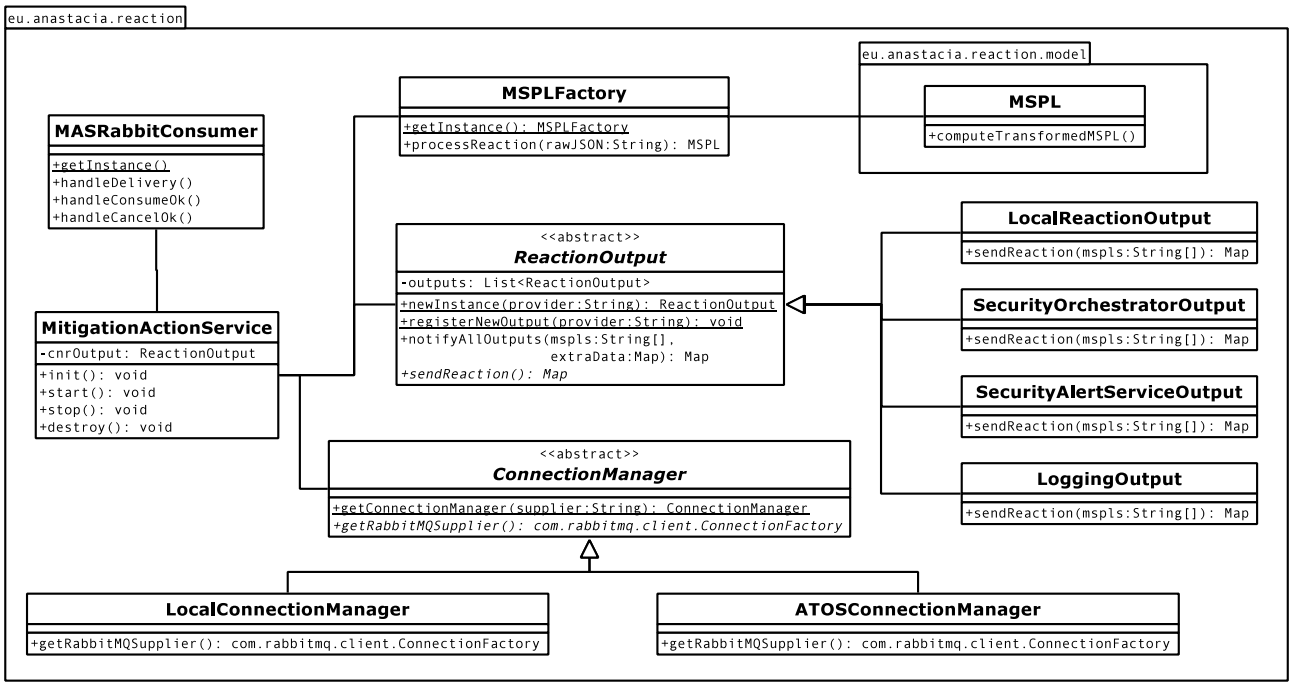


Figure 10 - Class diagram for Mitigation Action Service

The Principal class is “MitigationActionService”, which is in charge of actively look for messages in the RabbitMQ server of the VDSS. This class defines four methods (init, start, stop and destroy, all of them required by the JSVC library) that are called accordingly in different stages of the service. In these methods, the Class will coordinate the interactions with the rest of the classes in order to correctly read the messages from the VDSS, generate the corresponding MSPL and send it to the registered outputs. The interactions at each step of the execution are explained in the next section.

## 2.2.3.2 Implementation Details

As it is shown in the class diagram of Figure 10, the MAS is formed by a set of Java classes. The MitigaitonActionService class is the principal one, which implements the “Daemon” class of the JSVC library, defining four methods that are invoked automatically by the jsvc command.

### 2.2.3.2.1 Init

The “init” method is the starting point of the service. It is automatically invoked by JSVC when starting the service. The sequence diagram of the init method is depicted in Figure 11.

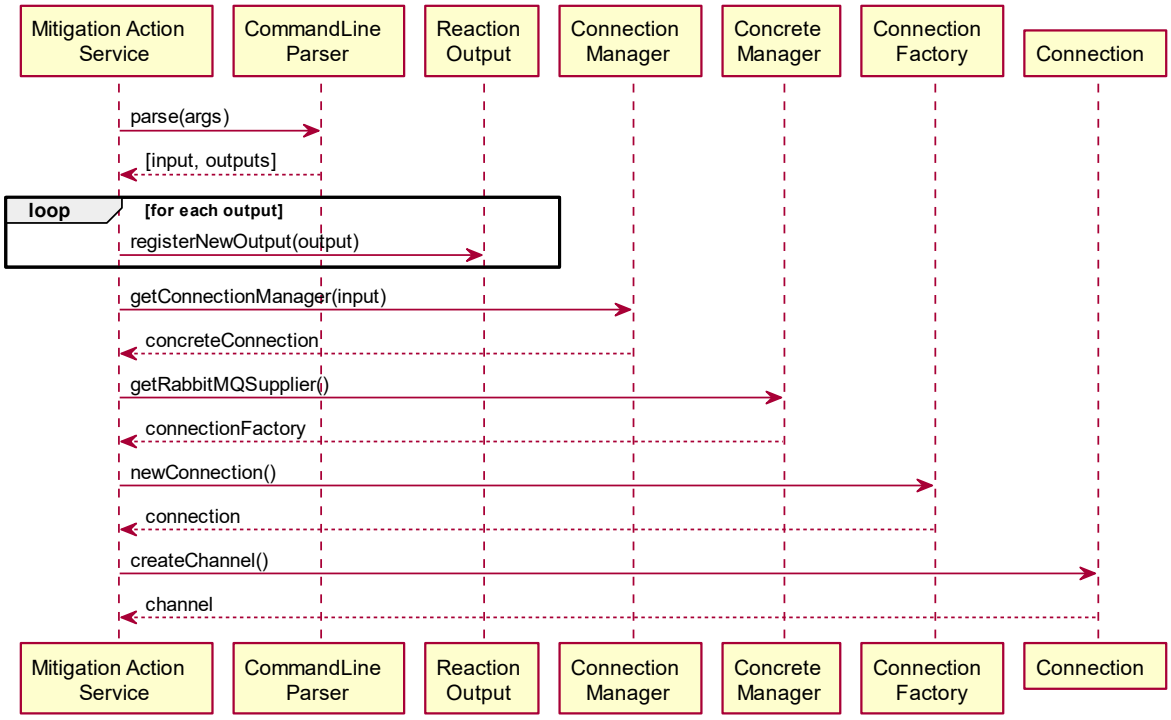


Figure 11 - Sequence Diagram for init Method

As mentioned before, this method is in charge of initializing the required variables and connections. It starts by parsing the arguments, determining which is the input of the MAS and the outputs. Then for each detected output, it registers it in the ReactionOutput class. Then, the MitigationActionService class starts configuring the input of the server, by obtaining the Connection Manager of the input specified in the arguments and obtaining the RabbitMQ supplier (an instance that handles how to connect to the server) for the specified input. Finally, the MAS class creates a new Connection object and the respective Channel to start the communication.

At this stage, the Mitigation Action services is still not ready to process messages, since the Rabbit MQ client has not been configured yet. This will be done in the “start” method.

#### 2.2.3.2.2 Start

Once connection objects were initialized the connection objects, it is required to start the client and actively listen for messages in the RabbitMQ server. To this end, the MitigationActionService class performs the interactions shown in Figure 12.

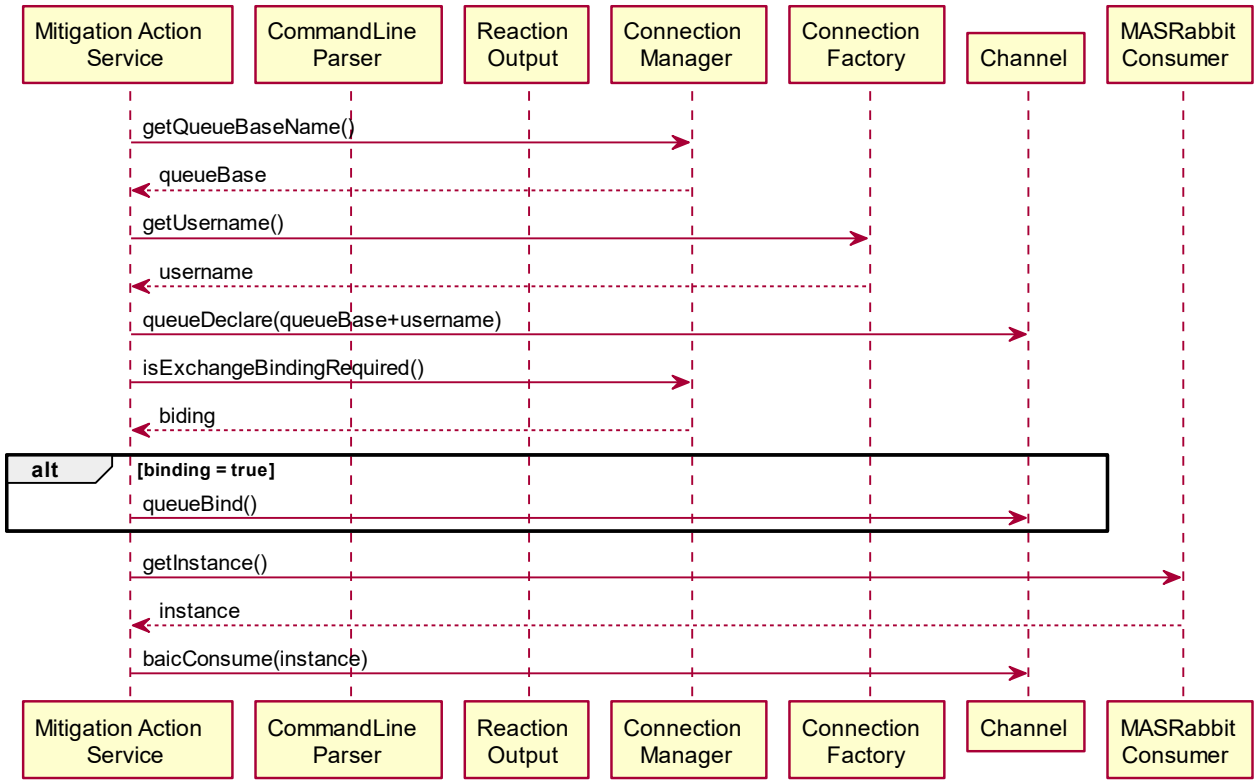


Figure 12 - Sequence Diagram for start Method

The figure above shows the principal interactions of the Mitigation Action Service to start the process. It first retrieves the basic information to declare the name of the RabbitMQ queue. Later, it determines if it is required to bind the queue to an exchange queue (this information is specified by the provider) and it performs the binding in case it is required. Then, it retrieves the instance of MASRabbitConsumer class, in order to invoke the "basicCosume" method on the channel, defining the callbacks of the instance the MAS just retrieved from the Rabbit consumer.

After this process, the MA relies on the RabbitMQ java library that will invoke the callbacks defined in the MASRabbitMQ class. Once a message arrives to the broker, the library will invoke the "handleDelivery" method, which will be in charge of processing the message (a JSON string) to generate the MSPL. Figure 13 shows the sequence diagram of the MSPL computation process.

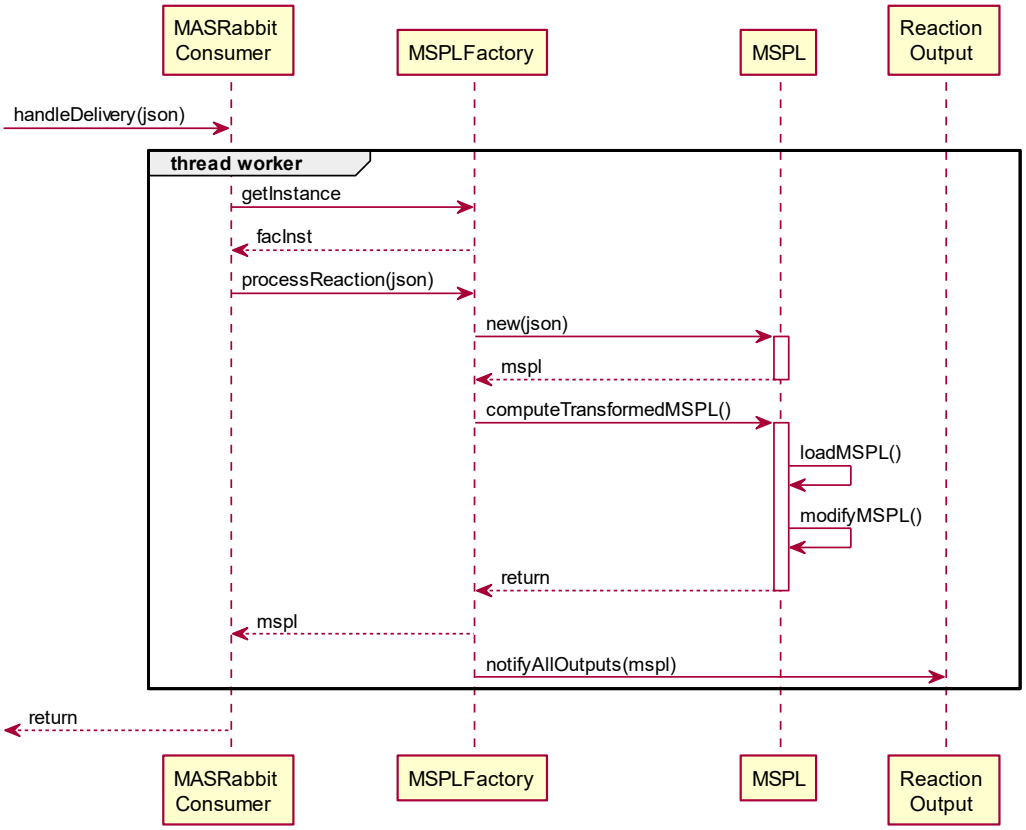


Figure 13 - Sequence Diagram for handleDelivery Method

Once the handleDelivery method has been called in the MASRabbitConsumer class, the Mitigation Action Service submits a new thread worker to process the received message. This worker will call the MSPLFactory class, obtaining the current instance. Then the MAS will send the raw JSON string to the returned MSPLFactory instance in order to process the reaction and generate the MSPL.

The process of the reaction starts in the MSPLFactory class, which parses the JSON (by means of using the Jackson fasterXML library) and creates a new instance of an MSPL object using the parsed JSON object. The constructor of the MSPL class will use the parsed JSON to find keyword on specific fields of the VDSS alert, in order to identify the use case involved. In addition, the constructor will also extract all the key information that will be later used when computing final MSPL.

Once the use case has been identified, the MSPL factory invokes the “computeTransformedMSPL” method. This invocation will load the corresponding MSPL template and modify the required fields according to the information extracted in the constructor. This invocation finished the creation of the MSPL file for the use case. Moreover, the design of the whole MSPL creation engine has been conceived to be easily extensible for the second period of the project, in which the MSPL will be computed dynamically with the information of the Security Model Analysis Component.

Once the MSPL has been computed, the MASRabbitConsumer class will invoke the method “notifyAllOutputs” on the ReactionOutput class in order to send the computed MSPL to all the activated outputs of the service. This includes (but are not limited to) the Security Orchestrator, the DSPS and the local terminal (for testing purposes).

#### 2.2.3.2.3 Stop

Sometimes it is required to stop the service without completely closing all the connections. This is needed by the JSVC library that might decide to restart the service by calling first stop, and then start after it. In this scenario, it is important to stop looking for new messages in the RabbitMQ server, avoiding generating new MSPL files.

To this end, the stop method invokes a cancel in the RabbitMQ channel, stopping the execution of the "handleDelivery" callback method and, therefore, not allowing reading more messages from the VDSS.

#### 2.2.3.2.4 Destroy

This method is automatically invoked after the stop, just before exiting the Java Virtual Machine. Considering this, the implementation of this method will simply close the connection to the RabbitMQ server.

### 2.2.3.3 Deployment

Once the development has been completed, the MAS was deployed in a dedicated machine in order to actively process the output of the VDSS. In this sense, the deployment of the MAS is continuously receiving the alerts from the VDSS and generating the corresponding MSPL for each use case treated in the project.

To support the development process of the different use cases of the project, the MAS design (as shown in Figure 10) includes the implementation of different outputs by means of implementing a single output interface. This design also allows activating and deactivating the different implemented outputs, which allows debugging and test other parts of the ANASTACIA platform without impacting other already-tested modules.

The deployment of the MAS takes advantage of this feature, by means of activating and deactivating the communications channels of the MAS by demand of the partners. This decision was taken in order to avoid introducing noise during the testing of the individual components, and therefore obstructing the debugging process with test messages coming from other parts of the platform.

## 2.2.4 Security Model Analysis

The Security Model Analysis (SMA) is a component of the Reaction module. This component continuously collects and updates through Kafka broker the existing capabilities and their related threats and costs. In fact, the SMA component retrieved from the Orchestrator the available capabilities that are supported by the IoT infrastructure according to the security policy and the capabilities enforced, and collects from the VDSS the corresponding threats. Moreover, it updates the cost of each capability computed by the VDSS. Finally, the output will be sent by the VDSS to the MAS in order to generate the accurate MSPL file, the Figure 14 shows the SMA interactions.

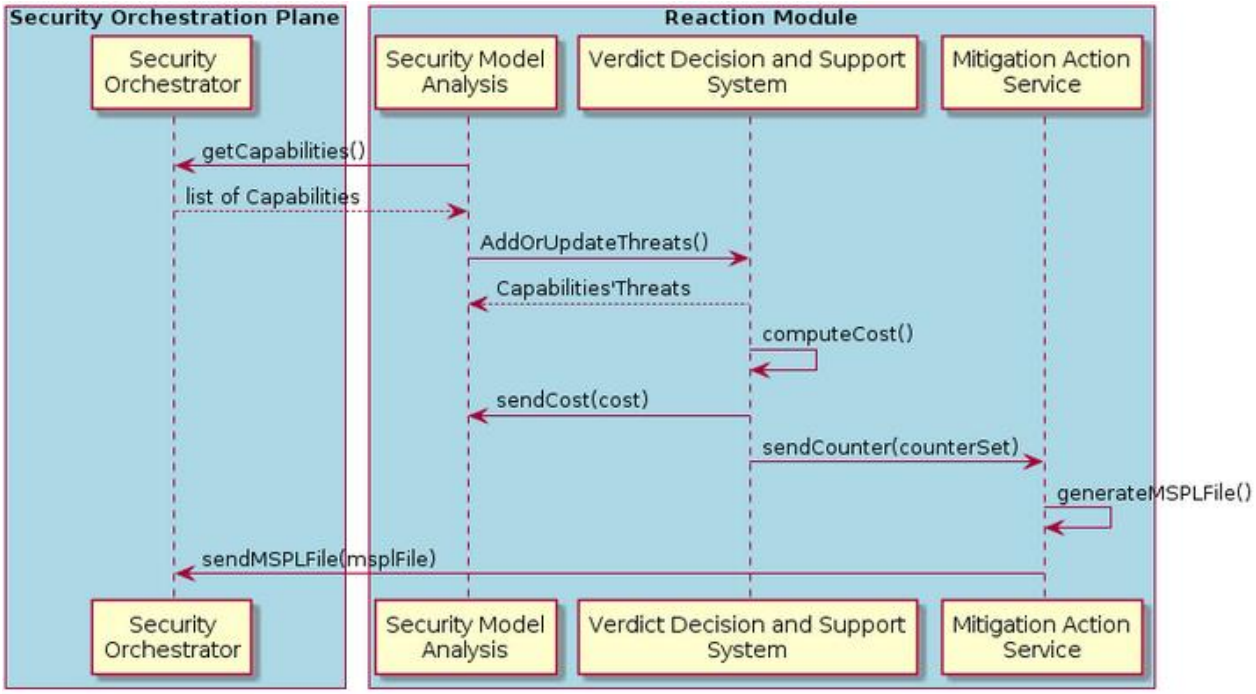


Figure 14 - The Security Model Analysis interactions

### 2.2.4.1 SMA deployment

The component is deployed in wrapper fashion to simplify interactions between complex communication environment and computational part of SMA. The data wrapper provides external connectivity to AALTO security orchestrator via REST API and ATOS XL SIEM component using SSL secured RabbitMQ channel.

The operations of deployed SMA component (Figure 15) are executed in four steps. During component initialization wrapper will request security models from SO component. Next SMA component will start receiving analysing thread costs provided by ATOS XL SIEM component. After computing information SMA will provide security model analysis that is then sent to ATOS XL-SIEM component on step 3. Lastly in step 4 ATOS based on the security model analysis and attack verdict messages will generate reactions that will be send to SO component for execution in SEP.

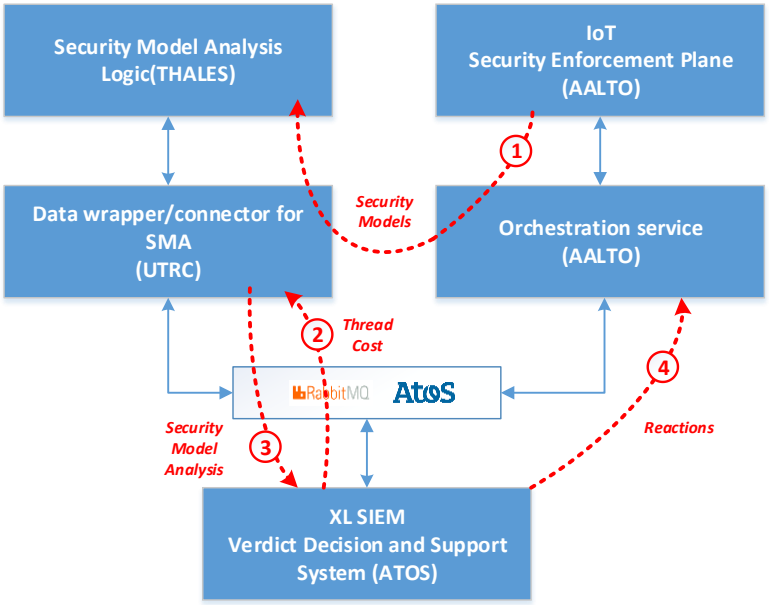


Figure 15 - Interaction between SMA

Example of the JSON file generated by the SMA that aggregate the information provided by the security orchestrator and VDSS.

```
{
   "Capability1":{
      "cost":cost1,
      "threats":[
         "threat1-1",
         "threat1-2"
         "threat1-3"
      ],
      "id":0
   },"Capability2":{
      "cost":cost2,
      "threats":[
         "threat2-1",
         "threat2-2"
      ],
      "id":1
   },
…
}
```

### 2.2.5 Available Capabilities

Currently the ANASTACIA framework is able to implement a series of countermeasures to protect the IoT networks from threats of different kinds and with different objectives. The possible countermeasures are reported in Table 2.

| Threats | Capabilities |
|---|---|
| Insider attack (malware) | "Authentication",<br>"AuthoriseAccess_resource",<br>"DTLS_protocol",<br>"Filtering_L3",<br>"Filtering_L4",<br>"Traffic_Divert",<br>"IoT_honeynet",<br>"IoT_control" |
| Man In The Middle | "Authentication",<br>"DTLS_protocol"<br>"Anonimity" |
| SQL Injection | "Filtering_L3", |

| | |
|---|---|
| | "Filtering_L4",<br>"Traffic_Divert",<br>"IoT_control" |
| Denial of Service | "Filtering_L3",<br>"Filtering_L4",<br>"Traffic_Divert", |
| 0-day | “Anonymity”<br>"Authentication",<br>"AuthoriseAccess_resource",<br>"DTLS_protocol",<br>"Filtering_L3",<br>"Filtering_L4",<br>"Traffic_Divert",<br>"IoT_honeynet",<br>"IoT_control" |
| Traffic analysis/Sniffing | “Anonymity”<br>"DTLS_protocol",<br>"Filtering_L3",<br>"Filtering_L4",<br>"Traffic_Divert", |

Table 2 - Available capabilities

In the next release of the Reaction Module, the table will be upgraded with more countermeasures and information related to the threats that the framework is able to detect and mitigate.

# 3 SUMMARY OF INTERNAL COMMUNICATION

This section shows the internal communication between the components of the Reaction module. Details can be found in Table 3.

| From | To | Data | Format | Services |
|---|---|---|---|---|
| Verdict and Decision Support System | Security Alert Service | Information about detected threats in form of Alerts. | JSON | Dedicated RabbitMQ queue with SSL |
| Mitigation Action service | Security Alert Service | Information on policy selected to mitigate the detected threat | HTTP POST request with JSON data | Web server, currently only HTTP connection |
| Verdict and Decision Support System | Mitigation Action Service | Information about detected threads in form of alerts. | JSON | Dedicated RabbitMQ queue with SSL |
| Security Model Analysis | Verdict and Decision Support System | Information about available capabilities | JSON | Dedicated RabbitMQ queue with SSL |

Table 3 - Internal communication Reaction Module

Future implementation of the communication is reported in the table below.

| From | To | Data | Format | Services |
|---|---|---|---|---|
| Capabilities Database | Mitigation Action Service | Available capabilities of the network and allowed actions specified in the Security Policy | - | Dedicated database shared between the Mitigation Action Service and the System Model Analysis |
| Mitigation Action Service | Security Alert Service | Computed MSPL and extra data about the deployment by the Security Orchestrator | HTTP POST request | Web Server on the Security Alert Service side |

Table 4 - Future implementation

# 4 RELEVANT FEATURES OF THE REACTION MODULE

The Reaction Module is a critical component of the framework of ANASTACIA, and it concerns the definition of the reactions to consider to protect the system from identified cyber-attacks. The attacks are detected by the Monitoring module that generates an alert and sends the alert to the Reaction Module.

The main objective of the module is, starting from the received alert, to execute a procedure that automatically defines possible countermeasures to protect the system from attack. As mentioned previously, the VDDS component carries out a threat risk assessment and, based on the capabilities available on the platform, decides the most convenient countermeasure to protect the system.

Subsequently, this information is shared with the MAS and SAS components: the MAS component, on one hand, is responsible for automatically generating an MSPL file and sharing it with the OS which will be responsible for deploying the protection system autonomously and thus mitigating the attack. This process is conducted by the MAS using the information provided by the VDSS and, in a second release of this component, it will use the information of the deployed security policy and the security capabilities deployed in the network. Using this data, the MAS will be able to dynamically compute the countermeasures, augmenting the risk analysis – performed in the VDSS – with the information on the security capabilities available in the Security Enforcement Plane. This feature will bring a key innovation to the MAS, which will be able to suggest a set of proper countermeasures that can actually be deployed by the security orchestrator.

The SAS component, on the other hand, enriches the information received from the VDSS with data regarding the selected policies and shares them with the DSPS. The information on the capabilities supported by the system are retrieved and stored by the SMA, associating the threats and the relative cost to the available policies.

An important innovative aspect of the ANASTACIA Reaction Module concerns the automatic selection of possible countermeasures against an identified attack. The system will automatically take care of identifying, deciding and implementing countermeasures to avoid damage to the system on which the framework is installed.

As for the Reaction Module, this component automatically selects possible countermeasures available on the system to implement a protection system in the shortest time possible to avoid creating critical damage to the system by exploiting the dynamic generation of MSPL files.

Furthermore, by exploiting the VDSS module, a risk analysis is performed on the identified attack, to subsequently share this information with the Seal Manager module.

# 5 Conclusions and Future Work

This document presented the advancements on the development of the Reaction Module components of the ANASTACIA Platform. Initially, an overview of the actual state of the infrastructure of the ANASTACIA framework is presented. After the overview, the deliverable describe in details the Reaction Module and its component.

For the different component of the Reaction module, a technical description is provided. In particular, the objective of the component and the communication with the connected entity is presented in order to understand the flow of information and how the module define the countermeasures for a detected threat.

The aim of this first version of the Reaction module is to develop and interconnect the different component in order to create an initial version of the module. Some modules are developed for this purpose instead other modules are modified in order to adapt their functionalities to the activities required by the Reaction Module. Also, future implementations are proposed to enhance the quality of this ANASTACIA framework.

The Reaction Module is also characterized by innovative aspects, implemented to optimize and protect networks and devices of the system more efficiently. By adopting a security by design approach to improve security aspects of the system, the Reaction Module is indeed able to autonomously decide (without the need of human actions) the countermeasures to be applied when a cyber-attack is detected. Furthermore, once an attack is detected, its impact on the system is reduced to very short time, since the deployment of security countermeasures requires minimum amounts of time, close to real-time. Another interesting characteristic of the Reaction Module is that the components are able to manage large number of detection events and, in case multiple threats are identified by the ANASTACIA framework, the Reaction Module is able to manage a cluster of devices, in order to elaborate large numbers of events coming from the other ANASTACIA modules.

This deliverable closes the first development iteration of the Reaction Module. The aim of this document is to develop a first working version that could be used as a proof of concept, adopted to basis for the second iteration of development to implement more functionalities.

# 6 ANNEX I. CATEGORIES AND SUBCATEGORIES OF SECURITY ALERTS

| Category | Subcategory_id |
|---|---|
| **Exploit** | ActiveX |
| | Attack_Response |
| | Browser |
| | Buffer_Overflow |
| | Command_Execution |
| | Cross_Site_Scripting |
| | Denial_Of_Service |
| | Directory_Traversal |
| | DNS |
| | File_Inclusion |
| | Format_String |
| | FTP |
| | Linux |
| | Mail |
| | Misc |
| | PDF |
| | Samba |
| | Shellcode |
| | Spoofing |
| | SQL_Injection |
| | Windows |
| **Authentication** | Account_Lockout |
| | Account_Unlocked |
| | Admin_Access |
| | Auth_Required |
| | Bruteforce |
| | Bypass |
| | Cleartext |
| | Default_Credentials |
| | Disclosure |
| | Error |
| | Failed |
| | FTP_Login_Failed |
| | FTP_Login_Succeeded |
| | Group_Added |
| | Group_Changed |
| | Group_Deleted |
| | Login |
| | Logout |
| | Misc |
| | Password_Change_Failed |
| | Password_Change_Succeeded |
| | Policy_Added |
| | Policy_Changed |
| | Policy_Deleted |
| | User_Changed |
| | User_Created |
| | User_Deleted |

| | |
|---|---|
| **Access** | ACL_Deny |
| | ACL_Permit |
| | Connection_Closed |
| | Connection_Opened |
| | File_Access |
| | File_Blocked |
| | Firewall_Deny |
| | Firewall_Misc_Event |
| | Firewall_Permit |
| | Misc |
| | Timeout |
| | Traffic_Inbound |
| | Traffic_Outbound |
| | Tunnel_Closed |
| | Tunnel_Connection |
| | Web_Application_Access |
| **Malware** | Adware |
| | Backdoor |
| | Fake_Antivirus |
| | Generic |
| | KeyLogger |
| | Spyware |
| | Trojan |
| | Virus |
| | Virus_Detected |
| | Worm |
| **Policy** | Anonymity |
| | Check_Failed |
| | Check_Passed |
| | Games |
| | Instant_Messaging_Chat |
| | Other |
| | P2P |
| | Phishing |
| | Porn |
| **Denial of Service** | Application |
| | DDoS |
| | Flood |
| | Other |
| | Web_Application |
| **Suspicious** | Bad_Traffic |
| | Blacklist_Address |
| | Database_Activity |
| | DNS_Activity |
| | DNS_Protocol_Anomaly |
| | FTP_Protocol_Anomaly |
| | HTTP_Protocol_Anomaly |
| | Mail_Activity |
| | Mail_Protocol_Anomaly |
| | Netbios_Activity |
| | Network_Activity |
| | Network_Anomaly |

| | |
|---|---|
| | NFS_Activity |
| | RPC_Activity |
| | Scada_Activity |
| | SSH_Activity |
| | SSH_Protocol_Anomaly |
| | Telnet_Protocol_Anomaly |
| | Threshold_Exceeded |
| | Web_Attack_or_Scan |
| **Network** | BOOTP_Activity |
| | DHCP_Activity |
| | FTP_Activity |
| | H.323_Activity |
| | High_Load |
| | IGMP_Activity |
| | IKE_Activity |
| | IPSEC_Activity |
| | L2TP_Activity |
| | Misc |
| | NTP_Activity |
| | OCSP_Activity |
| | PKI_Activity |
| | PPP_Activity |
| | PPTP_Activity |
| | RIP_Activity |
| | SIP_Activity |
| | SMTP_Activity |
| | SNMP_Activity |
| | SSL_Activity |
| | Telnet_Activity |
| | TFTP_Activity |
| **Recon** | Misc |
| | Scanner |
| **Info** | Misc |
| **System** | Alert |
| | Configuration_Changed |
| | Configuration_Error |
| | Critical |
| | Debug |
| | Emergency |
| | Error |
| | Information |
| | Locked |
| | Notification |
| | Process_Started |
| | Process_Stopped |
| | Restart |
| | Service_Started |
| | Service_Stopped |
| | Software_Installed |
| | Started |
| | Stopped |
| | Unlocked |

| | |
|---|---|
| | Warning |
| **Antivirus** | Definitions_Updated |
| | Definitions_Updated_Failed |
| | Disabled |
| | Error |
| | Scan_Finished |
| | Scan_Started |
| | Started |
| | Unknown_Event |
| | Virus_Detected |
| | Virus_Quarantine |
| | Virus_Quarantine_Failed |
| **Application** | DHCP_Error |
| | DHCP_Lease |
| | DHCP_Misc |
| | DHCP_Pool_Exhausted |
| | DHCP_Release |
| | DHCP_Request |
| | DNS_Misc |
| | DNS_Succesful_Zone_Tranfer |
| | DNS_Zone_Transfer_Failed |
| | FTP_Command_Executed |
| | FTP_Connection_Closed |
| | FTP_Connection_Opened |
| | FTP_Error |
| | FTP_Misc |
| | Mail_Dropped |
| | Mail_Received |
| | Mail_Sent |
| | Mail_Server_Misc |
| | Spam_Detected |
| | VPN_Closed |
| | VPN_Denied |
| | VPN_Misc |
| | VPN_Opened |
| | Web_Closed |
| | Web_Denied |
| | Web_Error |
| | Web_Misc |
| | Web_Modified |
| | Web_Not_Found |
| | Web_Opened |
| | Web_Proxy |
| | Web_Redirected |
| | Web_Reset |
| | Web_Terminated |
| **Voip** | Call_Ended |
| | Call_Started |
| | Misc |
| **Alert** | HostIDS_Alert |
| | IDS_Alert |
| | IPS_Alert |

| | |
|---|---|
| **Availability** | State_Critical |
| | State_Down |
| | State_Unknown |
| | State_Unreachable |
| | State_Up |
| | State_Warning |
| **Wireless** | Anomaly |
| | Association |
| | Authentication |
| | Client_Associated |
| | Deauthentication |
| | Disassociation |
| | Flood |
| | Misc |
| | New_Network |
| | Probe |
| | Scanner_Detected |
| | Spoofing |
| **Inventory** | Mac_Change |
| | Mac_Detected |
| | Mac_Misc |
| | Operating_System_Change |
| | Operating_System_Detected |
| | Operating_System_Misc |
| | Service_Change |
| | Service_Detected |
| | Service_Misc |
| **Honeypot** | Attack_Detected |
| | Connection_Closed |
| | Connection_Opened |
| | Misc |
| **Database** | Error |
| | Login |
| | Login_Failed |
| | Logout |
| | Misc |
| | Query |
| | Start |
| | Stop |
| **Alarm** | Attacks |
| | Bruteforce |
| | Dos |
| | Malware |
| | Misc |
| | Network |
| | Policy |
| | Scada |
| | Scan |
| **FastFlux** | |
| **Spam** | |
| **Mobile** | Bot |
| | C&C |

| | |
|---|---|
| | Event |
| | Malicious |
| | Suspicious |
| | Bot |
| | C&C |
| **Websites** | Malicious |
| | Suspicious |
| | Vulnerable |
| **Attack** | abuse |
| | compromise |
| | data |
| | dos |
| | dos.dns |
| | dos.http |
| | dos.tcp |
| | dos.udp |
| | login |
| | malware |
| | other |
| **Bot** | fast_flux |
| | other |
| **CCH Report** | |
| **Botnet** | c2 |
| | other |
| | p2p |
| **C2_Server** | http |
| | irc |
| | other |
| **Malicious** | exploit |
| | malware |
| | other |
| | phishing |
| **Vulnerable** | |

Table 5 - Categories and subcategories of security alerts at the VDSS