ANASTACIA

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

D4.1

Initial Monitoring Component Services Implementation Report

Distribution level	PU
Contractual date	30.08.2018 [M20]
Delivery date	30.08.2018 [M20]
WP / Task	WP4
WP Leader	MONT
Authors	D. Rivera (MONT), A. Riccio (MONT), R. Trapero (ATOS), Enrico Cambiaso (CNR), D.Mehta (UTRC), Piotr Sobonski (UTRC), Giannis Ledakis (UBITECH), Rafael Marín-Perez (ODINS)
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

Table of contents

PUBLIC SUMMARY

This document is the first deliverable of the WP4 – Monitoring and Alert/Reacting Enablers. It contains an explanation of the Monitoring Module of ANASTACIA, describing its positioning in the general architecture of the project.

Following this approach, this document also presents how the available tools of ANASTACIA partners had to be adapted to fit in the Monitoring Module design. Despite these efforts, each tool had to undergo an adaptation development in order to correctly integrate it into the ANASTACIA platform.

This document also specifies the integration development of each tool in order to cope with the four selected use cases: MEC.3, BMS.2, BMS.3 and BMS.4. Firstly, the Montimage Monitoring Tool developments are described, consisting in the implementation of the detection rules for MEC.3 and BMS.3 use cases. Secondly, it is described the adaptations done to the ATOS XL SIEM software, which are principally related to the specification of the messages to read from the ANASTACIA general broker, and the respective plugins developed to parse such messages and extract their information. Thirdly the document presents the adaptations done in the UTRC Data Analysis module in order to communicate its information using the general broker, following a format that is readable by the ATOS XL SIEM software. Finally, the development of the general broker is presented, which is used as a general communication channel (to transfer messages form Monitoring agents to the ATOS XL SIEM tool) as well as a pre-processing and filtering engine. The latter functionality is provided by a data streaming processor, enabling its classification and re-formatting in real-time.

This document has been developed in close collaboration with the Integration Work Package (WP6), allowing the communication and deployment of different tools on the same attack environment.

1 INTRODUCTION

1.1 AIMS OF THE DOCUMENT

This document represents the first deliverable of WP4, which contains detailed information about the status of the implementation of the Monitoring Module of ANASTACIA. Section 2 presents the general advancements of the task, organizing the content in the following manner. Section 2.1 presents the general architecture of the ANASTACIA Monitoring Module, showing its principal components and functionalities. Section 2.2 presents the status of the implementation of the modules, exposing how the different tools of the partners were developed and adapted to work in the ANASTACIA platform. Finally, Section 3 presents the conclusions of the document.

1.2 APPLICABLE AND REFERENCE DOCUMENTS

This document refers to the following documents:

- D1.3 – Initial Architectural Design
- D2.2 – Attacks and Threats Analysis and Contingency Actions
- MS12 a – Monitoring Components Services Specified and Agreed by the Board

1.3 REVISION HISTORY

Version	Date	Author	Description
V0.1	05/06/2018	D.Rivera (MONT)	Initial ToC
V0.2	29/06/2018	A. Riccio (MONT) R. Trapero (ATOS)	Initial contributions from Montimage and Atos.
V0.3	31/07/2018	A Trapero (ATOS) D.Mehta (UTRC) P.Sobonski(UTRC) E. Cambiaso (CNR) D. Rivera (MONT)	Updated section 2.2.2. Adding section to 2.2.3 section. Minor comments and modifications in section 2.2.1. Added Introduction, Sections 2.1 and 2.2.
V0.4	14/08/2018	G. Ledakis (UBITECH) D. Rivera (MONT)	Added UBITECH contributions in section 2.2.4 Added Conclusion and Public Summary
V0.5	21/08/2018	R. Marín-Perez (ODINS) D. Rivera (MONT)	Complete review and final Version

1.4 Acronyms and Definitions

Acronym	Meaning
MMT	Montimage Monitoring Tool
DoS	Denial of Service
DDoS	Distributed Denial of Service
DPI	Deep Packet Inspection
PAN	Personal Area Network
CoAP	Constrained Application Protocol
DFPB	Data Filtering and Pre-processing Broker

2 THE ANASTACIA MONITORING COMPONENT

One of the main goals of the ANASTACIA project is to provide to monitoring services in order to detect potential security breaches and attacks on cyber-physical networks. These services are partially covered by already-existing tools from the partners that required further development and adaptations to make it work in the general architecture of the project. The following sections describe the general architecture of the ANASTACIA Monitoring Component and give details about the status of the implementation of the required development and adaptations to the integrated tools.

2.1 ARCHITECTURE

Starting with the definition of the General Architecture of ANASTACIA (which has already been presented in D1.3), the Monitoring Module also started its initial design phases. The principal idea with the preliminary design (exposed in D1.3) was to offer a flexible design that would make easier the integration of the tools brought together in the ANASTACIA platform. Figure 1 shows the initial, general design of the Monitoring Component that was presented as part of the ANASTACIA general architecture.

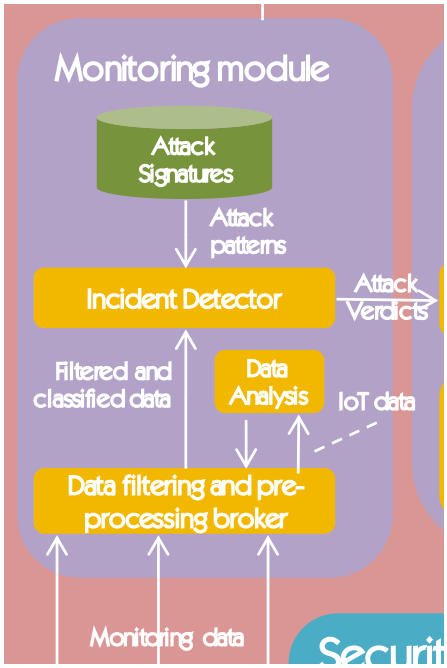

Figure 1 Initial Monitoring Module Design Presented in D1.3

In this initial design, four principal components are recognized:

- Data Filtering and Pre-processing Broker: The main objective of this component is to receive the data from the Monitoring Agents and provide an initial pre-processing and aggregation of the raw data. These data include, but are not limited to, data extracted by packet sniffers (MMT-Probe, for example) and data from cyber-physical sensors (temperature data, as is the case of UTRC sensors).
- Data Analysis: This module contains a behavioural-based analysis module that is used by UTRC to detect changes on the temperature sensors.
- Attack Signatures: This is the repository of attacks signatures that will be used by the Monitoring Agents to detect possible security threats.
- Incident Detector: The principal component of the design. It will collect all the processed data from the monitoring agents and raise alerts in case a security breach or attack has been detected.

2.2 IMPLEMENTATION

The initial design presented in Section 2.1 was later refined and put into perspective in order to clearly identify how the available tools can be inserted in the Monitoring Module of ANASTACIA. Figure 2 show how each one of the available tools were used in the proposed design, in order to fulfil the monitoring services of the ANASTACIA platform.

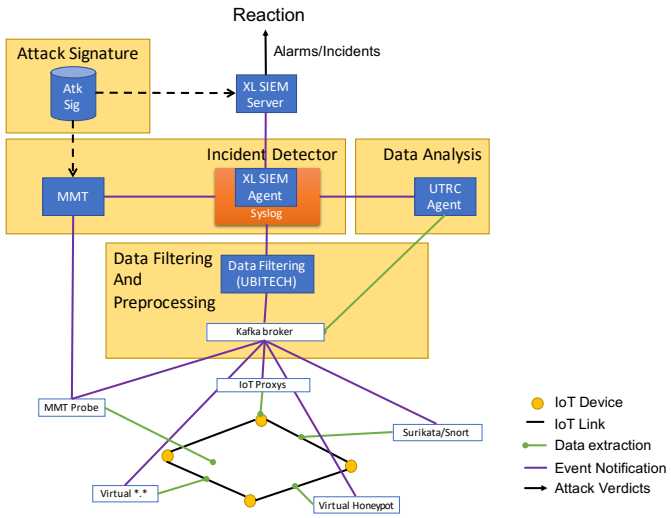

Figure 2 Mapped Design of the Available Tools on the Monitoring Module

Figure 2 shows how the components of the initial design of Figure 1 are mapped to the available tools provided by the partners. According to the figure, the Attack Signatures database was considered to be embedded in each member of the Incident Detector (MMT and XL-SIEM tools), since both detectors already supported this feature. Despite the simplicity of the presented solution, several adaptations have to be developed in order to correctly cope with the use cases of the ANASTACIA project. The following sections present the development made to integrate them into the proposed design.

2.2.1 Montimage Monitoring Tool

The *Montimage Monitoring Tool* (MMT) is characterized by a modular architecture which gives the software great flexibility and adaptability since modules can be assembled in several configurations according to stakeholders' requirements. Moreover, each module has been designed in order to be easily adaptable, adding new protocols and security rules as soon as it is necessary, hence improving the scalability of the systems

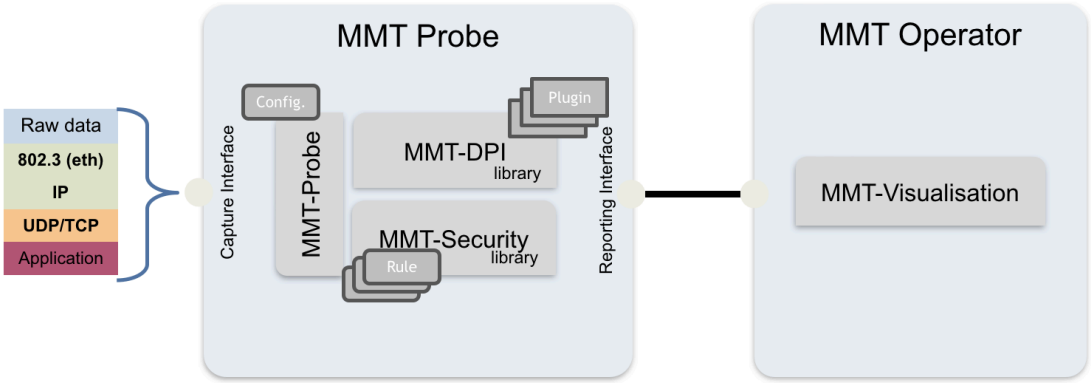

Figure 3 MMT Architecture General View

The Figure 3 shows the general organization of MMT software. Two principal components can be identified: MMT-Probe and MMT-Operator. The latter serves as the frontend of the platform, displaying the information extracted by MMT-Probe. For this reason, it does not add significant value to the ANASTACIA project and, therefore, will not be integrated in the platform, since it is in the scope of the user plane module of the project. However, *MMT-Probe* contains the principal logic and potential added value for the ANASTACIA platform: the DPI (by using the *MMT-DPI* library) and the Security Analysis capabilities (by using the *MMT-Security* library).

Considering this, *MMT-Probe* will be used with two principal goals. On one hand, it will provide periodical statistics about the flows on the network and, on the other hand, it will provide security analysis about any detected attack. To this end, the tool has been configured with its security extension: *MMT-Security*, in order to bring a security analysis capability to the ANASTACIA Monitoring and Reaction module.

Given the IoT nature of the ANASTACIA platform, the work on MMT-Probe adaptation was principally focused on extending the tool in order to correctly parse IoT-specific protocols and extract meaningful data (such as the IP addresses, port numbers and payload of the packet) needed for security analysis. Consequently, most of the work has been spent on extending the deep packet inspection module of MMT: *MMT-DPI* library. This library has also been designed with a modular approach, which makes it easy to be extended with new protocols in form of *plugins*. Moreover, it is also possible to choose which plugins include during the compiling phase thus tailoring the software for the particular environment it will be deployed.

Considering the ANASTACIA platform, several plugins have been developed in order to cope with the parsing of the IEEE 802.15.4 stack protocols. In particular, the development moved toward *6LoWPAN* technologies, in order to match the technology deployed on the monitored network. In order to support the broadest range of implementations, the development process followed very carefully the most recent RFCs (RFC 4944[1] and RFC 6282[2] for 6LowPAN and RFC 7252[3] for CoAP Protocol) and standards (IEEE Standard 802.15.4-2015[4]) regarding all protocols involved in the ANASTACIA platform. Nevertheless, the complexity and varsity of the standards have not allowed a complete support so far. Indeed, development focused primarily on those features mainly related to the implementation of the use cases proposed. However, full standard support is foreseen to be ready for the next iterations of the project.

Security analysis is performed by the *MMT-Security* module. This module receives data from DPI library and use them to detect possible security breaches/evasions according to security rules. These rules are written in XML and can be loaded either statically and dynamically thus providing the possibility to add new rules when new threats are discovered without stopping the tool itself. Montimage analysed the four use cases chosen for the project first iteration and identified two of them where the tool can detect attacks. To this end, security rules have been developed to detect the attacks of the use cases that can be applied, that is MEC.3 and BMS.3. A deep explanation of the implementation of these security rules can be found in next subsections.

2.2.1.1 Use Case MEC.3

In this scenario, an attacker, external to the network, controls a set of internal nodes and instructs them to execute a *ping flood Denial of Service (DoS)* attack on the network. In this case the attacking hosts are compromised IoT devices and smart cameras that flood the targeted host with a large amount of *ICMP echo* requests. The victim is therefore induced to consume its resources, in order to reply to each received request. During a successful attack, all targeted hosts are unable to communicate with other network nodes thus reaching a DoS state. Moreover, the attack is characterized by the ability to spoof packet source *IP* address,

[1] https://tools.ietf.org/html/rfc4944
[2] https://tools.ietf.org/html/rfc6282
[3] https://tools.ietf.org/html/rfc7252
[4] https://standards.ieee.org/findstds/standard/802.15.4-2015.html

assuming no security mechanism has been adopted. Therefore, it is trivial for the attacker to execute a Distributed DoS attack (DDoS). According to the above considerations, it follows that a detection strategy cannot rely on the source of the attack, since the attacker might be spoofing the source IP of the packets, rending the adopted security mechanisms useless. An attacker might also change the source IP address periodically, e.g. each 500 ms, or even choose a random value for each packet. This technique adds more difficulties when detecting the attack using the source IP address; in the first case, the detection technique will work until the attacker changes the IP, while in the second one no attack will ever be detected. For this reason, it is important to rely on the destination address, instead of the source address of the packets.

Legitimate traffic is another aspect that has to be considered as well. *ICMP echo* requests are usually used for checking reachability of a remote node, therefore this traffic must be allowed somehow. One approach is allowing this traffic but limiting its available bandwidth. In particular, the deliverable D2.2 proposes to allow a maximum of ten requests each five seconds, allowing only two requests per second to the same destination. Such trade-off still lets legit traffic flow through the network and at the same time detect DoS or DDoS on the victim, since the maximum allowed bandwidth is extremely low, compared to the bandwidth needed to successfully lead a DoS.

In conclusion, a detection strategy might be summarized as follows: *no more than two ICMP echo requests are allowed towards the same destination in a second.* This implies that an attack will be detected by observing at least three consecutive *ICMP echo* requests directed to the same destination in less than a second. In contrast, there should not be an alert when the same requests are spread over more than the same period.

In terms of *MMT-Security*, all this corresponds to a security rule composed by three different events, being each of them a detection of an *ICMP echo* request. However, to raise an alert all these events must happen in less than one second. More specifically, the *context*[5] of the rule is related to reception of the first two requests, where the destination address of the second ping request must match the destination of the first one. As it was mentioned before, the attacker might spoof the source address (in the case of DDoS attack), so the rule does not introduce any constraint regarding the source IPv6 address of the ping requests.

Figure 4 represents the *context* of the security rule, which detect packets that contain the value *128* in the type field of the *ICMP* header, which characterizes a packet containing an ICMP ping request.

```
<operator value="THEN" delay_units="s" delay_min="0" delay_max="1">
        <!-- Context -->
        <event value="COMPUTE" event_id="1"
                description="Context: ICMP ping"
                boolean_expression="((icmpv6.type == 128) &amp;amp;&amp;amp;
                lowpan.iphc_dst == lowpan.iphc_dst))"/>
        <event value="COMPUTE" event_id="2"
                description="Context: 2nd ICMP ping"
                boolean_expression="((icmpv6.type == 128) &amp;amp;&amp;amp;
                (lowpan.iphc_dst == lowpan.iphc_dst.1))"/>
</operator>
```

Figure 4 Definition of the Context for the ICMP Flooding Attack.

[5] For the MMT toolbox, the “context” is the set of initial conditions that need to be met in order to trigger a security alert.

Finally, the last event – the *trigger*[6] of the rule – corresponds to receiving a third *ICMP echo* with the same characteristics of the message related to the second event. Its definition is shown in Figure 5 below.

```
<!-- Trigger -->
<event value="COMPUTE" event_id="3"
        description="Trigger: 3rd consecutive ICMP ping packet"
        boolean_expression="((icmpv6.type == 128) &amp;&amp;amp;
                (lowpan.iphc_dst == lowpan.iphc_dst.1))"/>
```

Figure 5 Definition of the Trigger for the ICMP Flooding Attack.

Nevertheless, and in order for the *trigger* to be valid, this event must happen in less than one second related to previous events. This fact is expressed in the configuration of the property, as shown in Figure 6: Both the context and the trigger must happen within a maximum delay of 1 second. Once the *trigger* becomes true an alert is generated.

```
<property value="THEN" delay_units="s" delay_min="0" delay_max="1"
        property_id="60" type_property="ATTACK"
        description="3 consecutive ICMPv6 ping packets in a second. Possibly ICMP ping flood.">
                <!-- Context -->
                <!-- Trigger -->
</property>
```

Figure 6 General Definition of the MMT Property for the ICMP Flooding Attack.

The aforementioned security property is now ready to be compiled for the MMT-Security library, in order to detect the attack of the MEC.3 use case.

2.2.1.2 Use Case BMS.3

In this scenario an external attacker exploits a web page vulnerability to inject malicious *SQL* code in order to access or manipulate a *SCADA* database that is used to manage an energy micro-grid. Such exploitation leverages a security vulnerability in an application's software, e.g. lacking incorrect user input filtering for string literal escape characters, thus allowing maliciously crafted queries be executed on databases and letting an attacker obtain complete control on stored data. According to the D2.2 deliverable, such control might have several aims (injected SQL is presented underlined):

- to alter/tamper database contents
  - SELECT * FROM users WHERE name = 'foo'; DROP TABLE users; SELECT * FROM usersinfo WHERE '1'='1'
- to bypass access restrictions (in order to accomplish privilege escalation)
  - SELECT * FROM users WHERE name = '' OR '1'='1';
- to access/steal sensitive data
  - SELECT * FROM customers WHERE id = '' OR '1'='1';

[6] Similar as the "context", the "trigger" is the set of conditions that should be met in order to raise a security alert. The trigger will raise an alert ONLY if the context and the trigger are satisfied.

In each of the above cases, the underlined text represents the malicious code inserted by the attacker. Here the main goal of a detection strategy would be to check the presence of those statements in packets payload in order to detect a possible attack.

It is important to remark that only requests might contain this type of threats – since they are issued by clients – it would be more efficient to check only the payload of requests instead of responses (if any). Remember that, as previously explained, the application protocol adopted in ANASTACIA testing sensor network is the *Constrained Application Protocol* (CoAP).

In contrast with the previous attack (that was detected by sniffing packets directly in the IoT network) this one will be detected by sniffing packets near the requests server. In the previous use case, both the attackers and the victim were located inside the IoT network, so the detection of the attack was made inside the same IoT network. In this case, it is important to protect the requests server (which is located outside the IoT network) not only from attacks coming from inside the IoT network, but also from external sources. Following this, the MMT-Probe instance will be located outside the IoT network, in order to analyse the packets that flow in the incoming link of the server. Considering these observations, the MMT-Security rule was built in order to examine the incoming CoAP requests on a normal ethernet network.

A security rule for this kind of threat would be composed by two events. In this case, these events are not bounded by time constraints, so they must be both valid in the scope of a single packet. The first event (whose code is shown in listing below) is the *context* of the rule, and it is related with the detection of a *CoAP* request (distinguished by a message of *class* 0). Hence this is the first parameter to be checked. The second condition has been added for performance reasons related to MMT internals. The definition of the context of the rule is shown in Figure 7.

```
<!-- Context -->
<event value="COMPUTE" event_id="1"
        description="Context: a CoAP request"
        boolean_expression="((coap.class == 0) &amp;&amp;
        (ipv6.dst == ipv6.dst))"/>
```

Figure 7 Definition of the Context for the SQL Injection Attack.

The second event, or the *trigger* of the rule, is checking requests payload looking for the *SQL* statements described previously. Fulfilling this task requires more than just checking header values, therefore an *embedded function* has been developed. In particular, this function scans the payload byte per byte performing a syntactic search in order to look for one of those *SQL* statements that have been previously explained. In order to be more general, those statements are expressed in a general form in order to cope with variations of the same attack. The embedded function returns a boolean value indicating the response of the search. In this way, if at least one of those patterns are found in the payload, the *trigger* is valid, and an alert is raised, indicating an ongoing attack. The definition of the trigger can be found in Figure 8.

```
<!-- Trigger -->
<event value="COMPUTE" event_id="2"
        description="Trigger: SQL statements in the payload"
        boolean_expression="(#em_check_sql_injection_on_coap(coap.payload) == true)"/>
```

Figure 8 Definition of the Trigger for the SQL Injection Attack.

Finally, both the context and the trigger are combined inside the property tag to create the MMT-Security Rule as shown in Figure 9.

```
<property value="THEN" delay_units="ms" delay_min="0" delay_max="0"
        property_id="61" type_property="ATTACK"
        description="SQL Injection detected in CoAP payload.">
                <!-- Context -->
                <!-- Trigger -->
</property>
```

Figure 9 General Definition of the MMT Property for the SQL Injection Attack.

2.2.2 ATOS XL SIEM

The support for monitoring capabilities within ANASTACIA that ATOS uses is based on the adaptation of its XL-SIEM. The ATOS XL-SIEM is an incident detector that supports distributed correlation of incidents. It is based on the correlation of events received from agents. Agents are separate components, extensible from the XL-SIEM perspective, that retrieve events from probes running in some organization. Several agents can be deployed within the same or different organizations, normalizing events received from probes and sending them to the XL-SIEM service for its correlation. Agents are also easily extensible with new probes. Agents are based on plugins that adapt the probe events to the information that the XL-SIEM requires. Therefore, for any new probe it is required the taxonomy of events that will be receive from it, highlighting the important fields.

At this stage of the project four probes are considered in ANASTACIA:

- AAA events (for BMS.2 use case). OdinS provides with AAA probes that detect any anomaly related to the unauthorized access to IoT devices.
- Deep Packet Inspection scanning (for BMS.3 and MEC.3. use cases). MMT provides a DPI scanning of network traffic which can detect potential threats and ongoing attacks
- IDS events. AALTO supports an IDS scanner which provide events related to suspicious activity within the infrastructure.
- Data Analysis (for BMS.4 use case). UTRC provide with an anomalous behaviour scanner which uses operational data from IoT devices and detect anomalies in the data they produce.

Figure 10 represents the internals of the XL-SIEM agent and the plugins developed within ANASTACIA. The final purpose of the plugins is to normalize events into a common format that is understandable by the XL-SIEM server, which is in charge of correlating them and trigger the corresponding alarms.

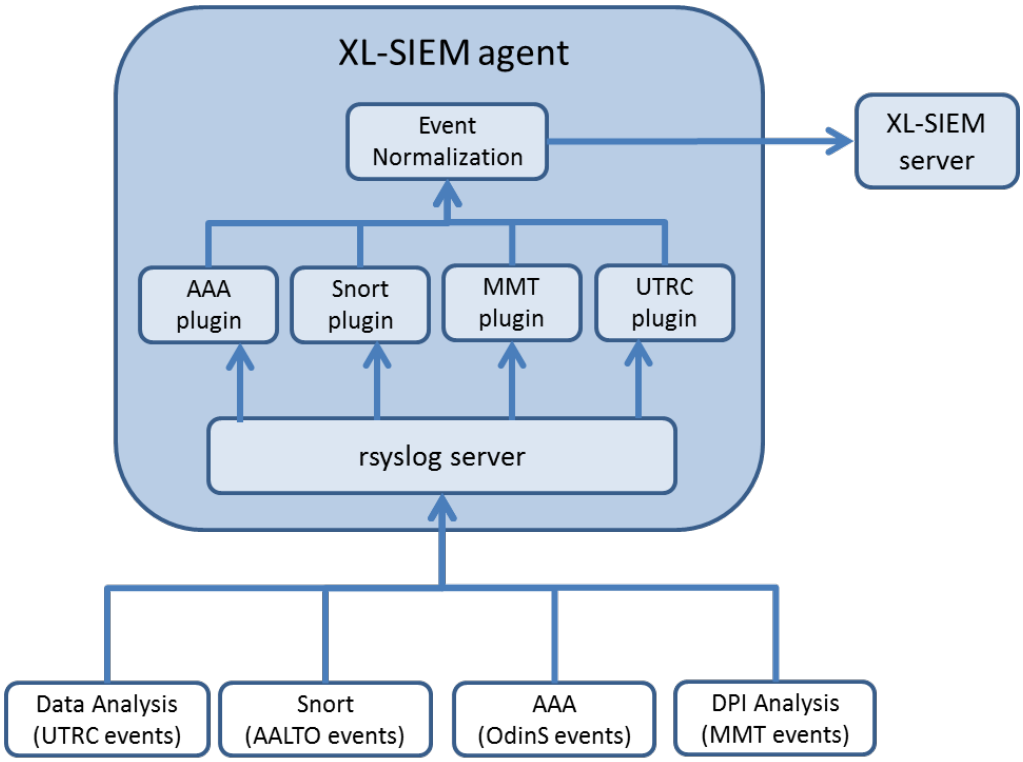

Figure 10 XL-SIEM plugin schema

The normalized events contain the following possible fields:

Table 1 Normalized events within the XL-SIEM agent

Normalized field	Description	Normalized field	Description
event_id	(mandatory)(internal) Used to uniquely identify the event	dst_ip	(optional) IP of the destination host identified in the event (default 0.0.0.0)
plugin_id	(mandatory)(internal) Used to identify the probe at the XL-SIEM server	src_port	(optional) Port at the source host identified in the event
plugin_sid	(mandatory)(internal) Used to identify the type of message within the same probe	dst_port	(optional) Port at the host destination host identified in the event
date	Timestamp of the event	userdata1…9	(optional) Custom fields to add information
interface	Network interface receiving the event	log	(internal) complete event
src_ip	(optional) IP of the source host identified in the event (default: 0.0.0.0)		

The following table represents the taxonomy of events received from the MMT probe:

Table 2 Events taxonomy received from MMT DPI analysis probe

Incident	Report field	Definition	Possible values
Security Report	reportType	The type of the report	"security"
	probeID	ID of the MMT Probe instance	Integer
	source	Name of the sniffed interface	String
	timestamp	Timestamp (in seconds.microseconds) of the report	Double
	propertyID	ID of the MMT-Property tested	Integer
	verdict	Respected or not	{"detected", "not_detected", "respected", "not_respected", "unknown"}
	securityType	Type of security property	{"attack", "securty", "evasion"}
	cause	Description of the property	String
	sourceIP	Source IP of the violaiton	String
	destIP	Destination IP of the violation	String
	sourceMAC	Source MAC of the violation	String
	destMAC	Destination MAC of the violation	String
	Example: <113>1 2018-06-08T08:46:03.377Z piscola.local MMT-Probe - - - {"propertyID":61,"verdict":"detected","securityType":"attack","cause":"SQL Injection detected in CoAP payload","sourceIP":"","destIP":"","sourceMAC":"","destMAC":"00:12:74:02:00:02:02:02","reportType":"security","probeID":3,"source":"tap0","timestamp":1.528447562E9}		
Statistics Flow with no session	reportType	The type of the report	"statistics-no-session"
	probeID	ID of the MMT Probe instance	Integer
	source	Name of the sniffed interface	String
	timestamp	Timestamp (in seconds.microseconds) of the report	Double
	protocolPath	Set of Detected protocols in the analysis of the flow	String
	dataVolume	Data amount including headers (in bytes)	Integer
	payloadVolume	Data amount excluding headers (in bytes)	Integer
	packetCount	Number of packets in the flow	Integer
	Example: <113>1 2018-06-08T09:34:40.330Z piscola.local MMT-Probe - - - {"protocolPath":"99.30","dataVolume":102,"payloadVolume":74,"packetCount":2,"reportType":"statistics-no-session","probeID":1,"source":"enp0s3","timestamp":1.528449753103438E9}		
Statistics Flow with session	reportType	The type of the report	"statistics-session"
	probeID	ID of the MMT Probe instance	Integer
	source	Name of the sniffed interface	String
	timestamp	Timestamp (in seconds.microseconds) of the report	Double
	protocolPath	Set of Detected protocols in the analysis of the flow	String
	dataVolume	Data amount including headers (in bytes)	Integer
	payloadVolume	Data amount excluding headers (in bytes)	Integer
	packetCount	Number of packets in the flow	Integer
	upDataVol	Uplink data volume in bytes	Integer
	upPaylVol	Uplink payload volume in bytes	Integer

	upPkgCnt	Uplink package amount	Integer
	downDataVol	Downlink data volume in bytes	Integer
	downPayVol	Downlink payload volume in bytes	Integer
	downPkgCnt	Downlink package amount	Integer
	clientIP	Origin IP	String
	serverIP	Destination IP	String
	sourceMAC	MAC address of the last hop	String
	destMAC	MAC address of the next hop	String
	clientPort	Origin Port of the flow	Integer
	serverPort	Destination Port of the flow	Integer
	Example: <113>1 2018-06-08T08:46:00.195Z piscola.local MMT-Probe - - - {"protocolPath":"99.178.354.0","dataVolume":114,"payloadVolume":4,"packetCount":2,"upDataVol":60,"upPayVol":4,"upPkgCnt":1,"downDataVol":54,"downPayVol":0,"downPkgCnt":1,"clientIP":"192.168.0.28","serverIP":"10.0.2.101","sourceMAC":"52:54:00:12:35:02","destMAC":"08:00:27:5c:93:58","clientPort":139,"serverPort":139,"reportType":"statistics-session","probeID":1,"source":"enp0s3","timestamp":1.528447554143143E9}		

The event normalization performed at the XL-SIEM agent is extracting the most relevant fields in a common format as depicted in the following table:

Table 3 Normalized fields for MMT events

Incident	Report field	Normalized event: field (plugin variable)
Security Report	reportType	<<implicit in the type of event>>
	probeID	userdata5 (probeID)
	source	userdata6 (src_ifce)
	timestamp	date
	propertyID	userdata7 (property_id)
	verdict	userdata3 (verdict)
	securityType	<<implicit in the type of event>> {"attack", "securty", "evasion"}
	cause	userdata4 (cause)
	sourceIP	src_ip
	destIP	dst_ip
	sourceMAC	userdata1 (src_mac)
	destMAC	userdata2 (dst_mac)
Statistics Flow with no session	reportType	<<implicit in the type of event>>
	probeID	userdata1 (probeID)
	source	userdata2 (src_ifce)
	timestamp	date and userdata3 (timestamp)
	protocolPath	userdata4 (protocolPath)
	dataVolume	userdata5 (dataVolume)
	payloadVolume	userdata6 (payloadVolume)
	packetCount	userdata7 (packetCount)
Statistics Flow with session	reportType	<<implicit in the type of event>>
	probeID	userdata5 (probeID)
	source	userdata3 (src_ifce)
	timestamp	date and userdata6 (timestamp)
	protocolPath	userdata4 (protocolPath)
	dataVolume	userdata7 (statistics)
	payloadVolume	
	packetCount	
	upDataVol	

	upPaylVol	
	upPkgCnt	
	downDataVol	
	downPayVol	
	downPkgCnt	
	clientIP	src_ip
	serverIP	dst_ip
	sourceMAC	userdata1 (sourceMAC)
	destMAC	userdata2 (destMAC)
	clientPort	src_port
	serverPort	dst_port

The following table represents the AAA probe events:

Table 4 Events taxonomy received from AAA probe by OdinS

Incident	Report field	Definition	Possible values
Forbidden Network Authentication	source_ip	Source of Attack, in IPv6 and including the port	String
	affected_ip	Device affected, in IPv6 and including the port	String
	type_of_device_affected	Type of device affected, such as PAA, IoT node or PEP	String
	event_type	Indicates the type of event, in this case “network authentication” labeled as “na”	“na”
	`Sep 1 17:02:38 {"source_ip":"aaaa::1", "source_port":"4000","affected_ip":"aaaa::2","affected_port":"716","type_of_device_affected":"PAA","event_type":"na"}`		
Forbidden Device Access	source_ip	Source of Attack, in IPv6 and including the port	String
	affected_ip	Device affected, in IPv6 and including the port	String
	type_of_device_affected	Type of device affected, such as PAA, IoT node or PEP	String
	event_type	Indicates the type of event, in this case “device access” labeled as “na”	“da”
	url	URL access attemp	String
	`Sep 1 17:02:38 {"source_ip":"aaaa::1", "source_port":"4001","affected_ip":"aaaa::2","affected_port":"5683","type_of_device_affected":"IoT_node","event_type":"da","url":"coap://[aaaa:2]:5682/<resource_access_url>"}`		
Forbidden Data Publication	source_ip	Source of Attack, in IPv6 and including the port	String
	affected_ip	Device affected, in IPv6 and including the port	String
	type_of_device_affected	Type of device affected, such as PAA, IoT node or PEP	String
	event_type	Indicates the type of event, in this case “data publication” labeled as “na”	“dp”
	url	URL access attemp	String
	`Sep 1 17:02:38 {"source_ip":"aaaa::1", "source_port":"4001","affected_ip":"aaaa::3","affected_port":"1026","type_of_device_affected":"PAA","event_type":"dp","url":"http://[aaaa:3]:1026/<data_publication_url>"}`		

The event normalization performed at the XL-SIEM agent is depicted in the following table:

Table 5 Normalized fields for AAA events

Incident	Report field	Normalized event
Forbidden Network Authentication	Source of Attack (IPv6)	src_ip
	Source of Attack (port)	src_port
	Device affected (IPv6)	dst_ip
	Device affected (port)	dst_port
	Type of device affected: PAA	userdata1 (device_type)
	Event type: [na]	<<implicit in the type of event>>
	Timestamp (taken from the log)	date
Forbidden Device Access Forbidden	Source of Attack (IPv6)	src_ip
	Source of Attack (port)	src_port
	Device affected (IPv6)	dst_ip
	Device affected (port)	dst_port
	Type of device affected: IoT-node	userdata1 (device_type)
	Event type: [da]	<<implicit in the type of event>>
	URL access attemp: `url`	userdata2 (url)
	Timestamp (taken from the log)	date
Data Publication	Source of Attack (IPv6)	src_ip
	Source of Attack (port)	src_port
	Device affected (IPv6)	dst_ip
	Device affected (port)	dst_port
	Type of device affected: PEP	userdata1 (device_type)
	Event type: [dp]	<<implicit in the type of event>>
	URL access attemp: `url`	userdata2 (url)
	Timestamp (taken from the log)	date

For the Data Analysis tool by UTRC, the taxonomy of events is as follows:

Table 6 Normalized fields for UTRC events

Incident	Report field	Definition	Possible values
Attack verdict	origin	Indicates the origin of the event	String
	timestamp	Indicates the date when the event occurred	String
	attack	Indicates whether there is an attack or not associated to this event	{“True”, “False”}
	severity	Indicates the level of severity of the anomaly detected	low, medium, high
	score	Indicates a score associated to the anomaly detected	numerical value represented as a string
	explanation	Textual description describing the anomaly detected	string
	events	List of events that have been correlated to generate this anomaly event. For every event the ts (timestamp of the event) and the val (value of the measure data) is included	json: - ts: string - val: numerical value represented as a string
	`Jul 10 11:19:29 10.0.2.2 [UTRC] {'verdict': {'origin': 'IoT', 'timestamp': 1531218480229, 'attack': True, 'severity': 'low', 'score': '0.90', 'explanation': 'Probing attack'},'events': [{'ts': '2018.02.23 17:02:38', 'val': '23.5'}, {'ts': '2018.02.23 17:04:16', 'val': '13.0'}, {'ts': '2018.02.23 17:12:38', 'val': '23.0'}]}`		

The event normalization performed at the XL-SIEM agent is depicted in the following table:

Table 7 Normalized fields for AAA events

Incident	Report field	Normalized event
Attack verdict	origin	userdata5 (origin)
	timestamp	data and userdata4 (timestamp)
	attack	<<Implicit in the type of event>>
	severity	userdata1 (severity)
	score	userdata3 (score)
	explanation	userdata2 (explanation)
	events	-not used-

Once the events are normalized they are sent to the XL-SIEM server. There are several connections possibilities. ANASTACIA uses a secure socket in port 41000. The following diagram shows the deployment architecture currently used in the project. The Incident Detector component of the monitoring module consists of two separate machines: the XL-SIEM agent and the XL-SIEM server. The XL-SIEM agent collects events from probes, normalize them and submit them to the XL-SIEM server that correlates them and trigger alarms. These alarms are exported to a RabbitMQ queen in order to be consumed by the Reaction module components.

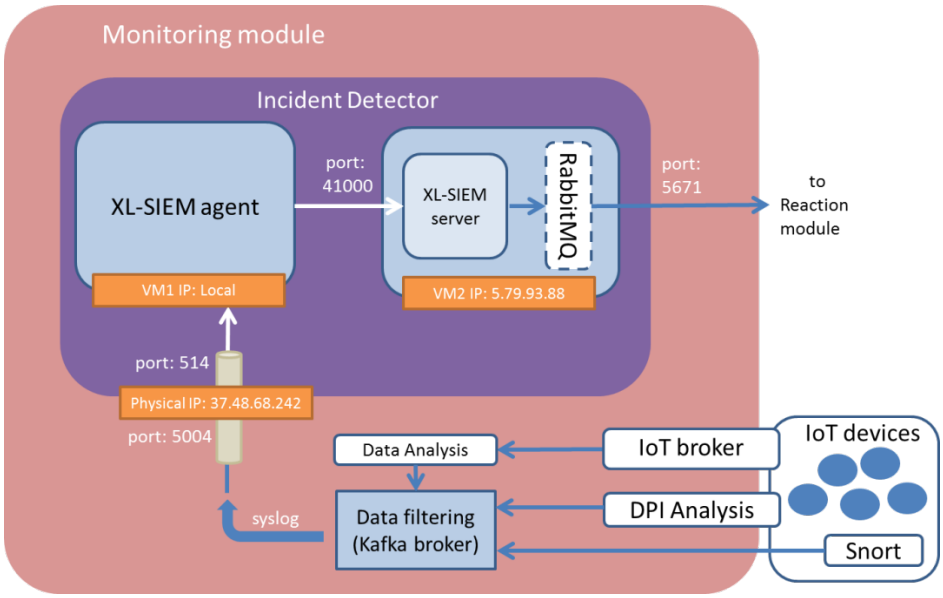

Figure 11 Internals of the monitoring module

Upon the reception of normalized events by the XL-SIEM server they are correlated, and alerts are generated. To do so, several rules and directives are necessary to be set-up at the XL-SIEM. More specially, at this stage of the development process only events from the same source are correlated in order to infer potential attacks. The following table represents an excerpt of the rules applied to the events received from the AAA probe, MMT sensor and the UTRC Data Analysis module. Additionally, many different rules are already set-up at the XL.-SIEM for the correlation of Suricata/Snort events. Rules are defined using the EPL[7] (Event Processing Language) syntax. The values Priority and Reliability are set-up by the system admin using an XL-SIEM control panel and represents the importance of the rule. These values, and the importance of the individual events are used to calculate the risk associated to the alert that is generated.

[7] https://docs.oracle.com/cd/E13157_01/wlevs/docs30/epl_guide/overview.html

Table 8 Sample of rules for generating alarms

Name	Rule	Priority	Reliability
MMT Probe - SQL Injection Detected	pattern [every-distinct(a.src_ip, 60 seconds) a=MMT_Probe_SQL_Injection -> b=MMT_Probe_SQL_Injection ((b.src_ip=a.src_ip) and (b.dst_ip=a.dst_ip))]	5	10
MMT Probe - ICMPv6 Ping	pattern [every-distinct(a.src_ip, 60 seconds) a=MMT_Probe_Ping_ICMP -> b=MMT_Probe_Ping_ICMP ((b.src_ip=a.src_ip) and (b.dst_ip=a.dst_ip))]	5	10
AAA Probe - Forbidden Network Authentication	pattern [every-distinct(a.src_ip, 60 seconds) a=AAA_Forbidden_Network_Auth]	4	6
AAA Probe - Forbidden Device Access	pattern [every-distinct(a.src_ip, 60 seconds) a=AAA_Forbidden_Device_Access]	4	7
AAA Probe - Forbidden Data Publication	pattern [every-distinct(a.src_ip, 60 seconds) a=AAA_Forbidden_Data_Publication]	4	6
Man in the Middle on IoT data	pattern [every-distinct(a.src_ip, 60 seconds) a=UTRC_MitM]	5	10

As an example, the first row represents SQL injection attacks detected by the MMT probe. Events collected from the MMT probe are compared in terms of source and destination IP. If the events arrive within 60 seconds, an alert is generated. The threshold of 60 second is used to prevent the flood of the XL-SIEM with redundant alerts.

The following figure represents a screenshot of the Atos XL-SIEM modified for its usage in Anastacia. More specifically it represents the alerts once different events from the current available probes are correlated.

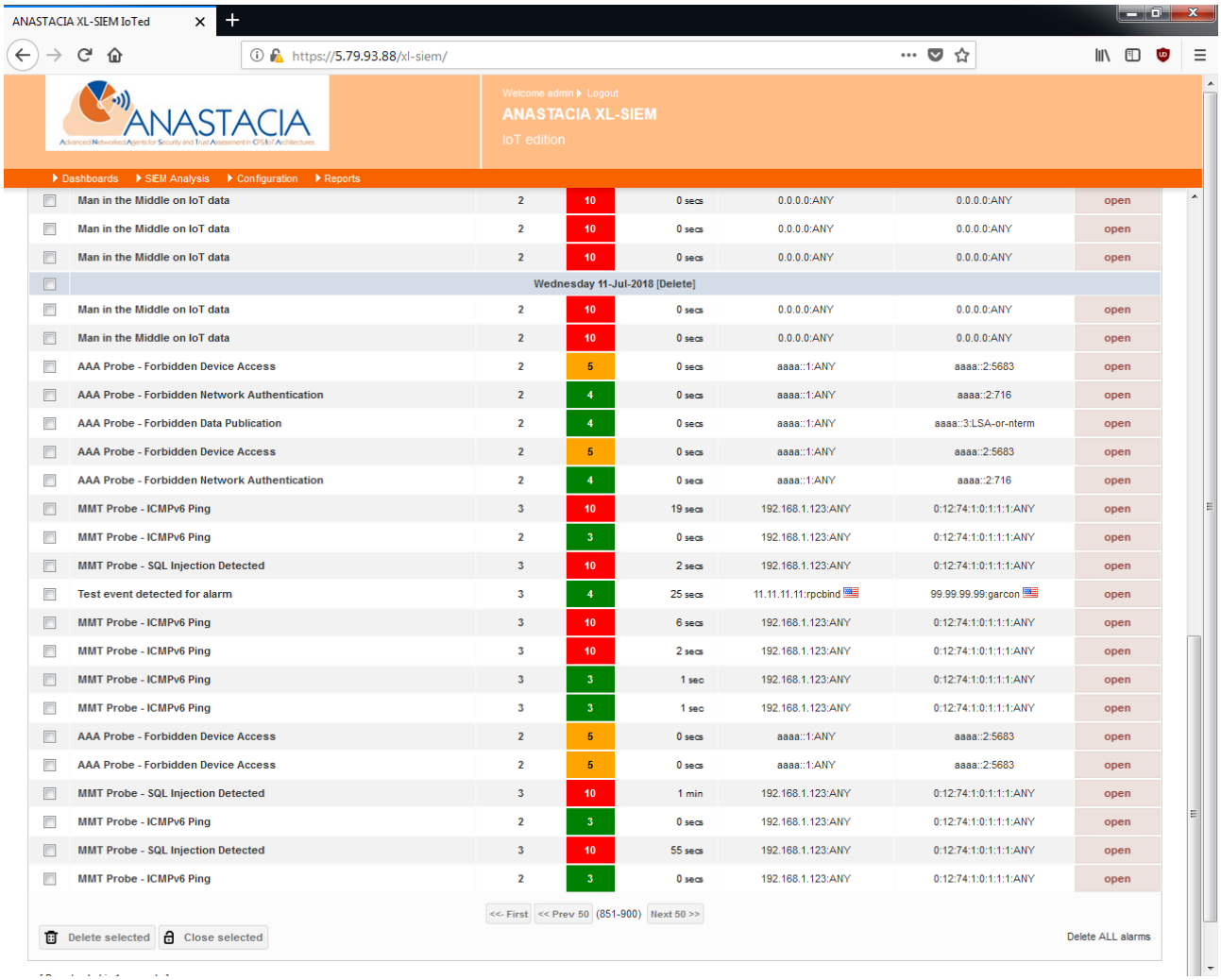

Figure 12 XL-SIEM dashboard showing alerts detected in Anastacia

The risk value represents the importance of the alerts, which considers the priority and reliability of the events and importance of the assets affected by the incident:

$$Risk = \frac{Reliability * Priority * Asset_{Importance}}{25} with \begin{cases} Reliability = \{0,1,2,3,4,5\} \\ Priority = \{0,1,2,3,4,5,6,7,8,9,10\} \\ AssetImportance = \{0,1,2,3,4,5\} \end{cases}$$

In addition to the information reported in the dashboard, every alert is exported to a RabbitMQ messaging queue in the form of a JSON message. This JSON is collected by the reaction module which uses it to decide on the mitigation to be enforced by the orchestrator, and to report it to the Dynamic Security and Privacy Seal. More details about these steps will be reported in D4.2.

Apart from the development of additional plugins to support the information shared by the monitoring probes deployed in the project and the new rules to correlate them, several modifications have been required to be done in the XL-SIEM for its use in Anastacia and in IoT platforms. Some examples are the adaptation of the correlation engine to use IPv6 addresses or the modification of the alerts exported for its use by the Reaction module and the Dynamic and Privacy Seal, adding additional information to support mitigations actions.

2.2.3 UTRC Data Analysis (Use case BMS.4)

The core function of any intrusion detection system (IDS) is to gather and analyse information in order to identify any intrusion. When the context is cyber-physical system or Internet of Things, IDS should not only

monitor cyber-related metrics (e.g. network activity, CPU speed, and log files) but also physical processes/measurements that govern the behaviour of physical devices. IoT or sensor data consists of a continuous stream of data (i.e. time-series) where the time interval between successive updates could vary from milliseconds to minutes. The data produced, usually pertains to the information about the physical state of a system e.g., temperature, pressure, voltage, power consumption, flow rate, speed, acceleration etc. The goal is to detect intrusion not only in cyber space but also in physical space. For example, the data reported by an IoT sensor could be far from its normal behaviour or an actuator could behave in a highly erratic manner.

The anomaly-based intrusion detection system builds a profile (or a data-model) of the normal behaviour using either statistical or unsupervised machine learning methods. It then uses the normal profile to flag any deviations from that profile as alerts. The advantages of anomaly-based IDS are that it can identify new attacks. We have developed an approach learn a constraint programming-based decision model consisting of a set of relations to detect misbehaviour of the system. More specifically, the idea is to learn a set of relations which together when satisfied defines the normal behaviour of the system. The workflow for learning the model is defined below (Figure 13).

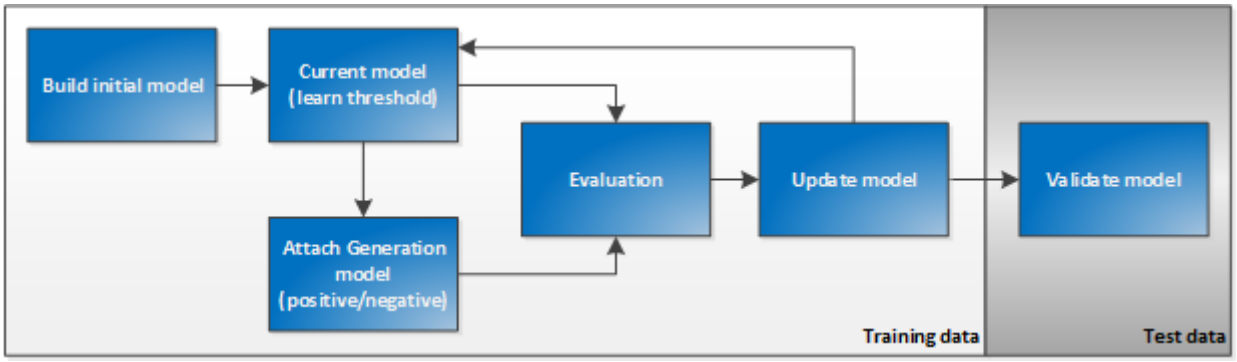

Figure 13 Workflow for model learning.

Data Analysis component generates attack verdicts based on SEP events received via Kafka broker and mapping them into pre-trained model. The event is sent by OdinS IoT broker to Kafka bus on topic **IoTBrokerTopic**. Event taxonomy is illustrated in detail in Table 9. UTRC agent subscribes to topic with OdinS IoT events and process them internally. In first step IoT data is aligned with other events and saved in local database for further processing (model training and evaluation). After recording IoT events they are sent to the model. Finally, wrapper responsible for sending verdict information sends request to the model to get current system state verdict. Once verdict is calculated the message is send to **UTRCVerdicts** topic for further analysis. Generated event taxonomy message is depicted in Table 9. UTRC verdict message is being intercepted by UBITECH proxy component that enables ATOS XL-SIEM to consume messages being posted on Kafka. It converts Kafka messages into syslog format messages that are scanned by ATOS XL-SIEM monitoring module. Integrated monitoring data flow with other ANASTACIA components has been illustrated on Figure 14.

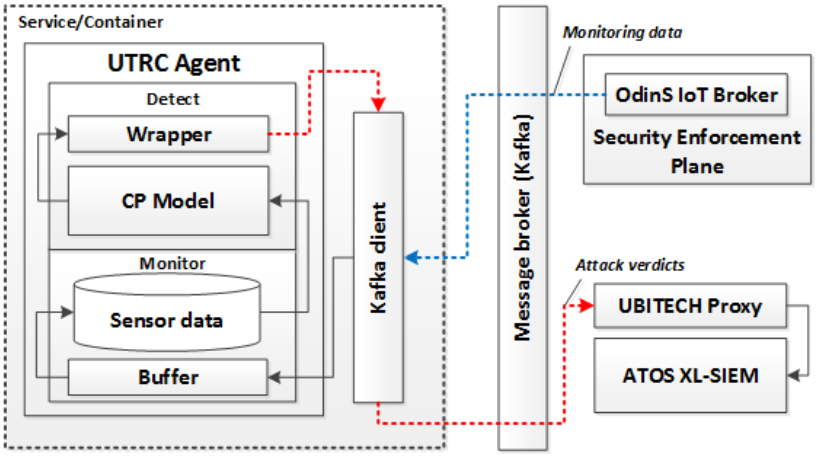

Figure 14 UTRC data analysis monitoring flow (blue – IoT data, red – attack verdicts).

Table 9 Events taxonomy received from OdinS IoT broker

Event	Report field	Definition	Possible values
IoT temperature sensor information	subscriptionId		String
	originator	Message origin – machine name	String
	contextResponses	List of context elements with IoT sensor data	List of dictionaries
	contextElement	IoT sensor data captured by OdinS IoT broker	Dictionary
	type	IoT device type	String
	isPattern	Is Pattern flag	String
	id	IoT device name	String
	attributes	List of IoT device attributes	List of dictionaries
	name	Attribute name	String
	type	Attribute type	String
	value	Attribute value	String
	statusCode	Code status for current message operation executed on OdinS IoT REST API	Dictionary
	code	HTML status code	Integer
	reasonPhrase	Textual representation of HTML status code	String
	{"subscriptionId": "5b44f0808f9da35934daef10", "originator": "localhost","contextResponses": [{"contextElement": {"type": "IoTdevice", "isPattern": "false", "id" : "IoTdevice/2001:720:1710:4:0:0:0:1001", "attributes": [{"name": "identificator", "type" : "string", "value": "/2001:720:1710:4:0:0:0:1001" },{"name": "temperature", "type": "float", "value" : "27.88"},{ "name": "timestamp", "type": "time", "value": "2018-07-26/06:53:59"}]}, "statusCode": {"code": "200", "reasonPhrase": "OK"} }] }	

From integration perspective UTRC Data Analysis component accomplished and demonstrated monitoring capabilities during integration meeting in UMU in Murcia. From test case scenario perspective additional implementation effort is required to complete internal event logging to enable distributed logging of test cases using Kafka message broker.

2.2.4 Data Filtering and Pre-processing Broker

The Data Filtering and Pre-processing Broker (DFPB) is a module of Anastacia is designed and implemented mainly by Ubitech in order to collect the monitoring information from multiple streams, filter and classify the incoming information, before making it available for analysis to the Incident Detector and the Data Analysis Module. Data Filtering can be considered the entry point of the Monitoring Module of ANASTACIA framework for the data supplied by the probes used in ANASTACIA. From the four probes presented already as inputs of the ATOS XL SIEM tool, at this stage of the project the following three probes are passing through the DFPB:

- 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.

DFPB can be considered as a middleware layer, that filters, prepares and unifies the provided monitoring information, converts input to messages following the syslog standard with the agreed formatting and forwards them to the Incident Detector monitoring module that based on ATOS XL-SIEM. The exact fields, the way that their values are represented and their arrangement as part of the syslog message must be guaranteed by DFPB in order to avoid issues on the Incident Detector. The structure of the messages has been described in section 2.2.2. The normalized data can even be enriched with semantic information, if needed. To achieve this, DFPB is based on the usage of Apache Kafka[8] as a message broker that collects the RAW data created from the probes using a publish/subscribe mechanism, and Apache Storm[9] as the framework to execute the real-time pre-processing of the RAW data. This way DFPB should be capable to collect streaming raw data from multiple sensors, allow streaming data access to the Data Analysis component, process the data streams and sent the output though syslog format, while being scalable through distribution.

The overall architecture of the component is depicted in the following Figure 15.

[8] https://kafka.apache.org
[9] https://storm.apache.org

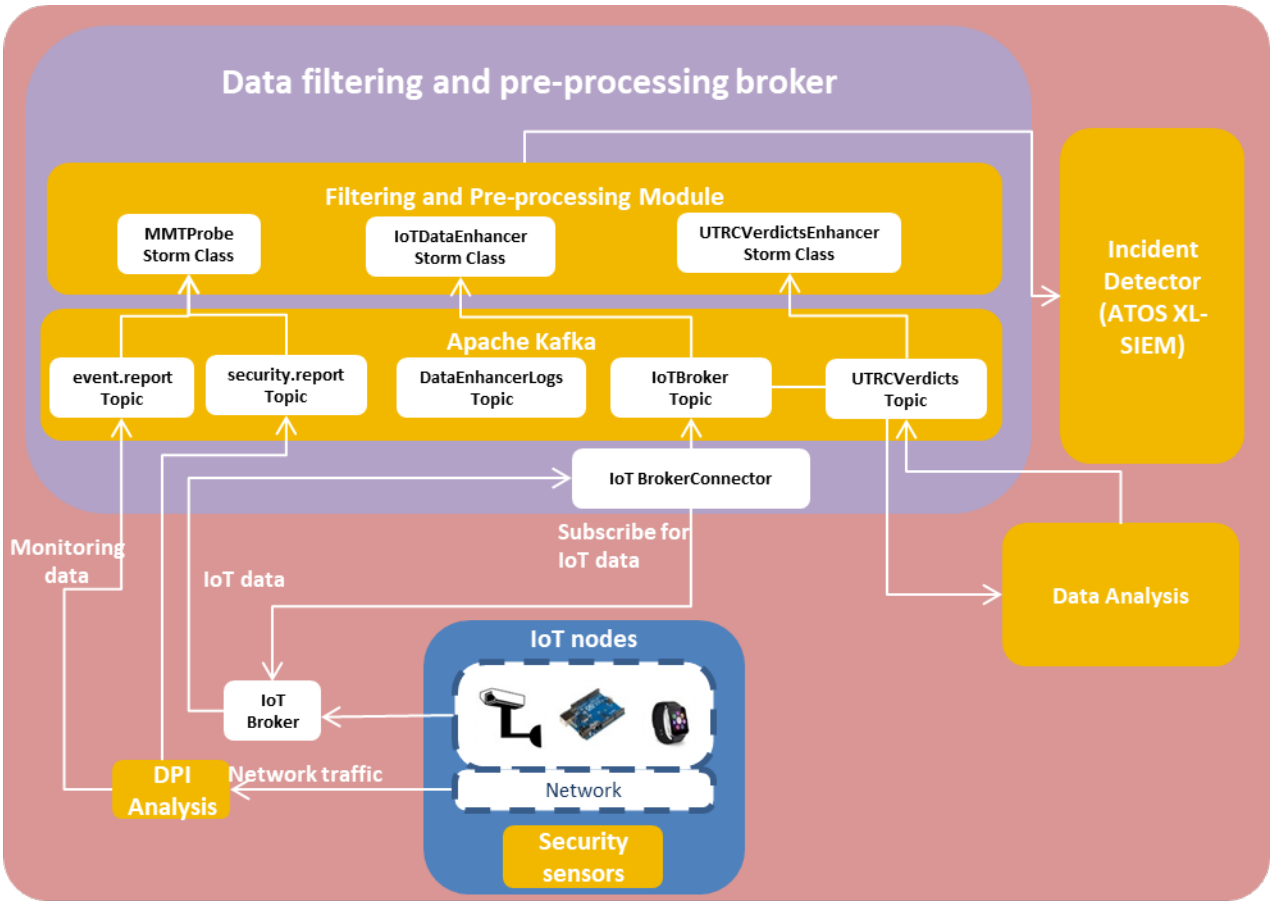

Figure 15 Data Filtering and pre-processing component internals and interactions

As mentioned already, for the implementation of the broker of Data Filtering and pre-processing mechanism that collects the data from the probes of ANASTACIA, we are using Apache Kafka. Apache Kafka is a distributed stream processing platform that has capabilities of a massively scalable message broker and also capabilities of stream processing through Kafka Streams. The publish/subscribe (pub/sub) messaging pattern is realized using destinations known as topics. Publishers send messages to the topic and subscribers register to receive messages from the topic, and when messages sent to the topic are automatically delivered to all subscribers. As depicted in Figure 15, appropriate topics for each of the project are created. For the easier monitoring of the topic and their contents, we have installed in addition to the Kafka, a dedicated UI that allows to browse topics and their contents[10].

In specific the potential threats and ongoing attacks detected by the Deep Packet Inspection of MMT probe is published in the topics security.report and event.report (see Figure 16).

[10] https://github.com/Landoop/kafka-topics-ui

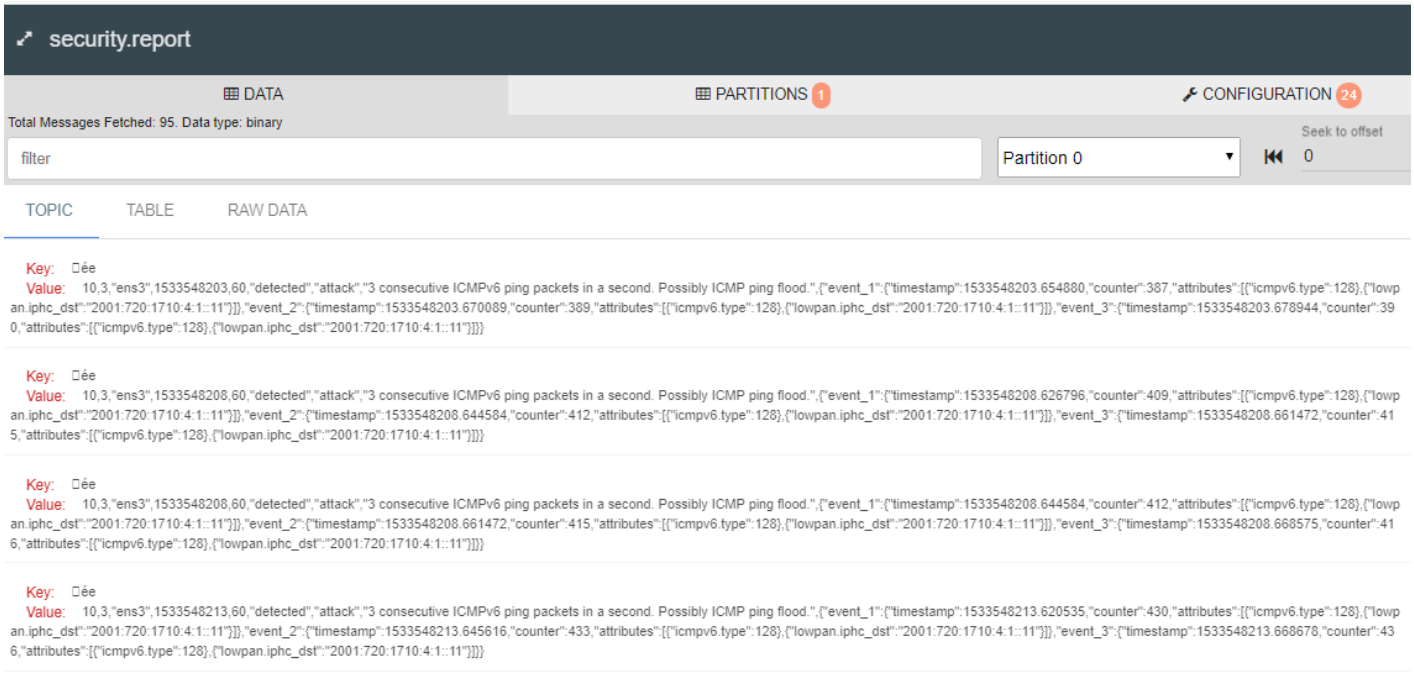

Figure 16 Contents of the security.report topic

The raw data from the IoT world are provided through the IoTBrokerTopic. The IoT data include information like temperature but also AAA events, related to the detection of any unauthorized access to IoT devices. It has to be mentioned that the IoT Broker was only capable to provide the aggregated information of the IoT nodes through subscription through REST calls, we created a module called IoTBrokerConnector that consumes data from the IoTBroker and push them to the Kafka topic.

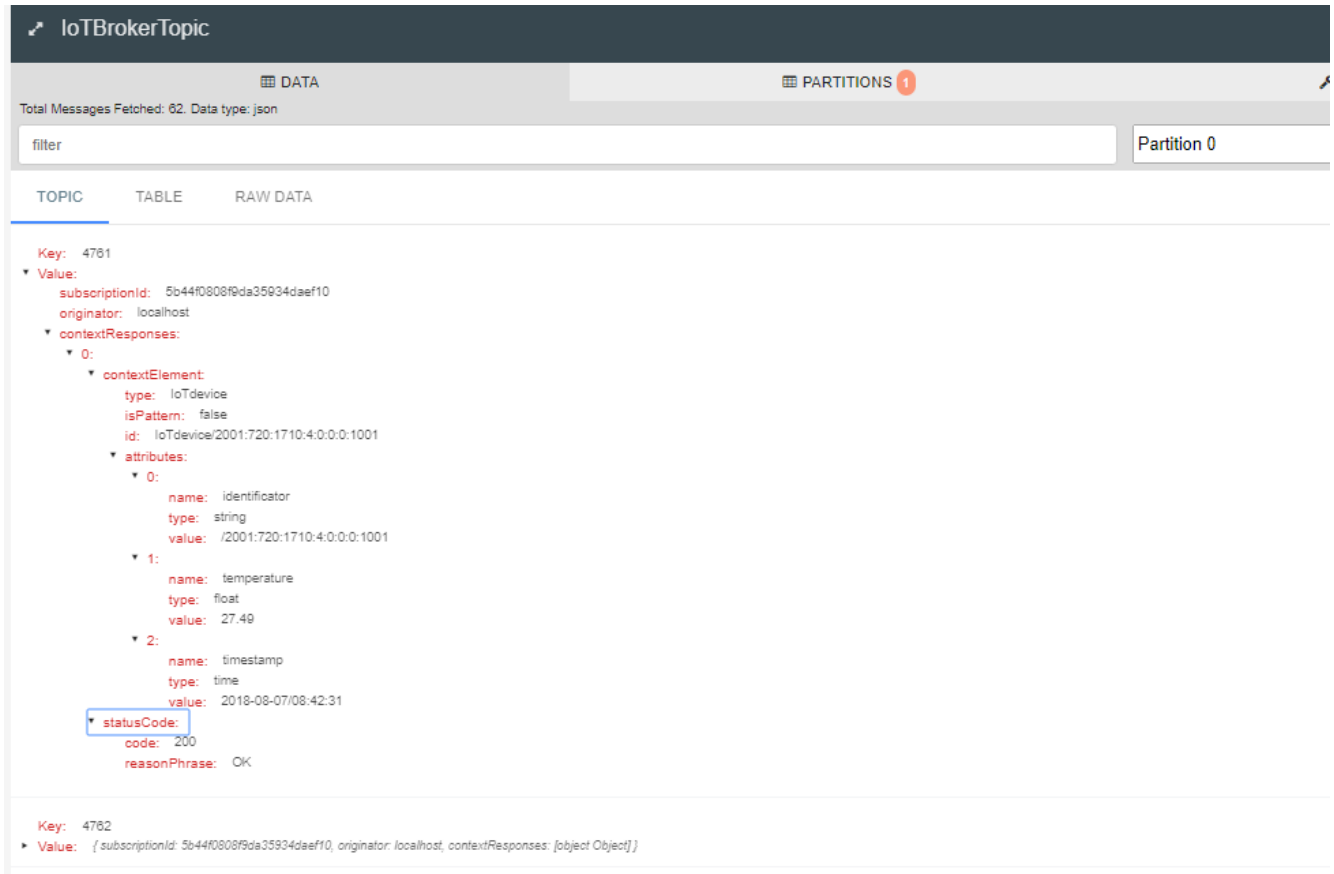

Figure 17 Contents of the IoTBrokerTopic topic

Finally, the Data Analysis component consumes events from the IoTBrokerTopic of the Kafka broker, map them into pre-trained model, and store them in local database for further processing (model training and evaluation). Based on this data stream, verdict of the data analysis is calculated and send again to the broker, to the dedicated UTRCVerdicts topic.

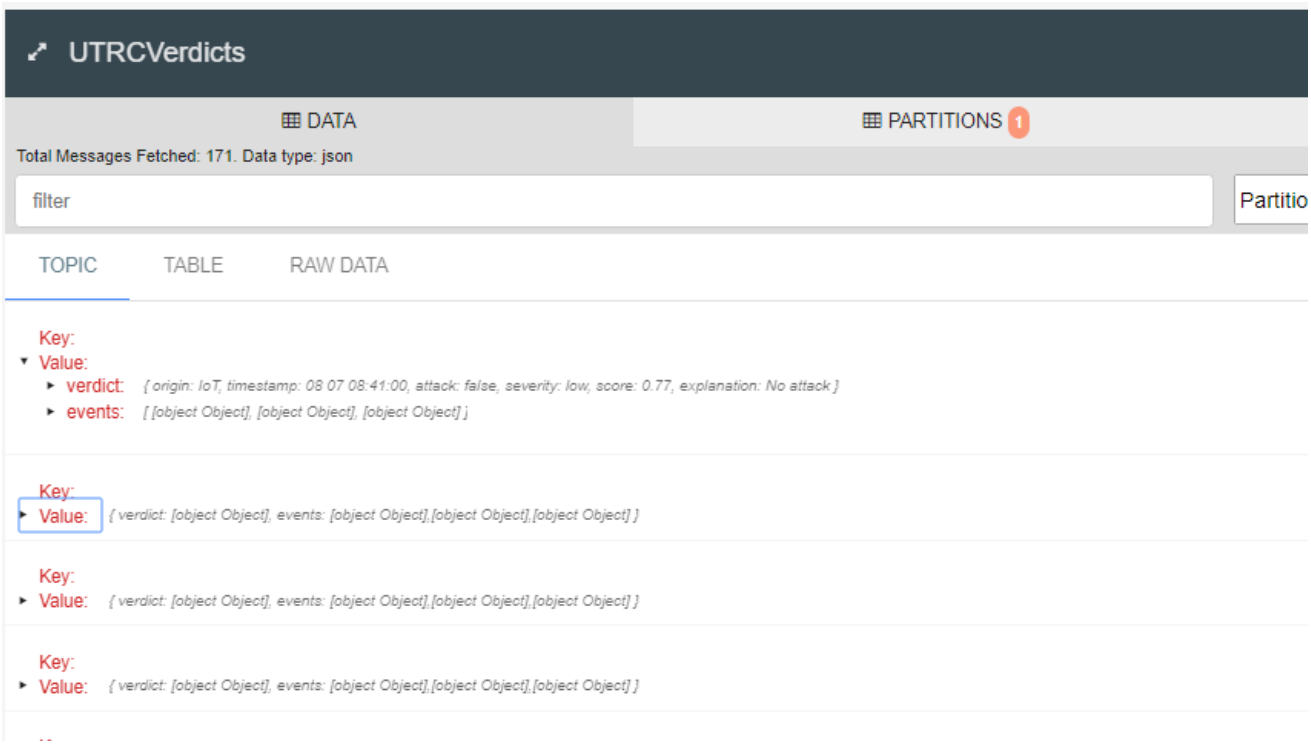

Figure 18 Contents of the UTRCVerdicts topic

All the messages that are published to the Kafka broker topics are processed in order to find missing fields and even fill them if possible, to make changes to formatting (e.g. in formatting of numbers or dates), properly format them and converts the information from the RAW messages into syslog format messages that are sent to ATOS XL-SIEM monitoring module for scanning. Finally, for testing, debugging and for logging reasons, we use a topic called MonitoringDataEnhancerLogs, in order to log all the processed data as there are sent to ATOS XL-SIEM.

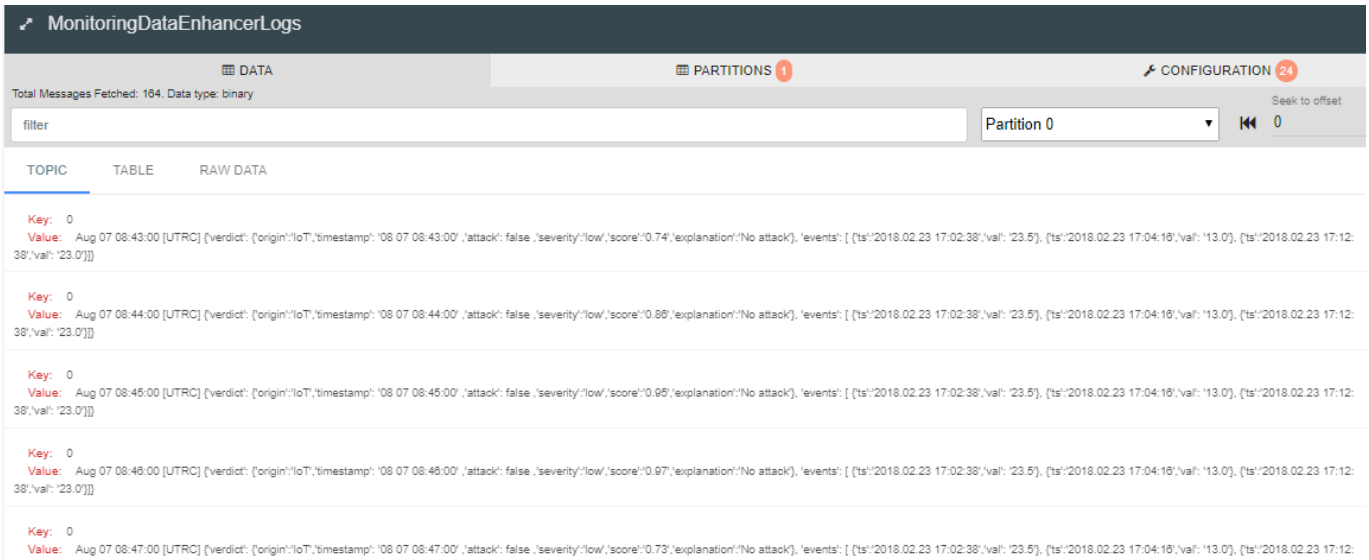

Figure 19 Contents of the security.report topic

This processing is done through Apache Storm, a distributed streaming and real-time computation framework. The usage of Apache Kafka and Apache Storm provide distributed storage and computation capabilities, thus ensuring the scalability of the developed solution.

For each of the probes that we have been incorporated in this initial version of the monitoring module, a dedicated Storm based application has been created. Currently, these applications share the same codebase and executable .jar file but use different runnable classes. For the deployment of these applications in both testing and production environments, we use Docker[11] and we create dedicated Dockerfiles[12] that are executed whenever we need to run the data filtering application.

In the following listing, we provide the Dockerfile content for the IoTDataEnhancer, Apache Storm based java application.

```
FROM storm:1.2.1
ADD ./target/data-enhancer-1.1.1.jar /app/
CMD storm jar /app/data-enhancer-1.1.1.jar eu.anastacia.monitoring.IoTDataEnhancer
```

In a similar way the application for the pre-processing of MMTProbe has been created, by executing the appropriate runnable class.

```
FROM storm:1.2.1
ADD ./target/data-enhancer-1.1.1.jar /app/
CMD storm jar /app/data-enhancer-1.1.1.jar eu.anastacia.monitoring.MMTProbe
```

Finally, for the Data Analysis verdicts the pre- processing component is deployed by executing the runnable class UTRCVerdictsEnhancer.

```
FROM storm:1.2.1
ADD ./target/data-enhancer-1.1.1.jar /app/
CMD storm jar /app/data-enhancer-1.1.1.jar eu.anastacia.monitoring.UTRCVerdictsEnhancer
```

The described mechanism allows not only provides scalability characteristics, but also it is easy to extension for the support of additional monitoring probes. Also, in the case that storage of the data is needed this can be performed using a Kafka connector to a MongoDB database. The code developed for the Data Filtering and Pre-processing Broker, including the code for the deployment of Kafka, and the Storm based pre-processing applications are available at the common code repository of the project[13].

[11] https://www.docker.com
[12] https://docs.docker.com/engine/reference/builder/
[13] https://gitlab.com/anastacia-project/

3 CONCLUSIONS

This document presented the advancements on the development of the Monitoring Components of the ANASTACIA Platform. To this end, this document presents the contents in a top-down manner.

Firstly, the design of the ANASTACIA's Monitoring Component was presented – which is part of the general architecture of the ANASTACIA platform, specifying which are the functionalities of each part of the Monitoring component and how they interact with other parts of the platform. It is presented then how the partner's available tools are used to fulfil the exposed design, covering all the sub-modules of the Monitoring Component. However, it was also required to modify the tools in order to meet the project's requirements.

The document then goes deep in the details about how the available tools had to be adapted for the project's use cases of the first iteration, namely MEC.3, BMS.2, BMS.3 and BMS.4:

- Montimage Monitoring Tool: For use cases BMS.3 and MEC.3, the MMT software had to be adapted in order to detect such threats. Section 2.2.1 describes how the new attack rules have to be developed in order to detect the attacks mentioned in these two uses cases:
- ATOS XL SIEM: As the ATOS XL SIEM was used as a general detector, several modifications have to be introduced in this software. Section 2.2.2 describes how different XL SIEM plugins have to be developed in order to correctly read the extracted information from different Monitoring Agents. In particular, this section describes how the plugins for the different sensors (MMT probe, AAA detector, UTRC detector and Snort) were developed in order to interact with the XL SIEM software.
- UTRC Data Analysis: In the particular case of BMS.2, a specific UTRC-proprietary sensor had to be used in order to detect a tampering attack. Section 2.2.3 shows how this proprietary software was adapted in order to generate detection alerts in a readable format, in order to be integrated with the technologies of the ANASTACIA platform.
- Apache Kafka/Storm Communication Broker: The Data Filtering and Pre-processing Broker (DFPB) was implemented as an entry point and middleware in order to normalize the data coming from different monitoring agent sources, using the streaming processor technology Apache Storm. This component also serves as a general communication broker, provided by an Apache Kafka sever, on which all the monitoring technologies can publish security information of the monitored network.

This deliverable closes the first development iteration of the Monitoring Module, which aimed to deploy a working version of the ANASTACIA platform for the four use cases mentioned above. The main idea behind this decision was to have a working version that could be used as a proof of concept, yet flexible enough to be extended for the rest of the use cases in the second iteration of the project.