ANASTACIA

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

D4.3

Initial Agents Development Report

Distribution level	[PU]
Contractual date	31.11.2018 [M23]
Delivery date	31.11.2018 [M23]
WP / Task	WP4 / T4.3
WP Leader	UTRC
Authors	R. Trapero (ATOS), I. Vaccari, E. Cambiaso, E. Punta, S. Scaglione (CNR), D. Rivera (MONT), R. Marin-Perez (OdinS), A.Skarmeta (UMU), A.Molina Zarca (UMU), A.Mady, S.Vuppala, A. Balogh, P. Sobonski (UTRC)
EC Project Officer	Carmen Ifrim carmen.ifrim@ec.europa.eu
Project Coordinator	Softeco Sismat SpA Stefano Bianchi Via De Marini 1, 16149 Genova – Italy +39 0106026368 stefano.bianchi@softeco.it
Project website	www.anastacia-h2020.eu

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

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

Table of contents

List of figures

List of tables

PUBLIC SUMMARY

This document describes initial implementation status of ANASTACIA agents deployed both in local devices and remote servers. Work accomplished during this task was completed in parallel with tasks T4.1 and T4.2. The deliverable elaborates on following topics:

- Key innovation and progress beyond the state of the art from monitoring and detection agents' perspective.
- Brief description of monitoring agents within ANASTACIA demonstrator framework and their features with distinction into two types:
    - Local agents, implemented in the local network and IoT devices for monitoring security enforcement plane and feeding enablers for threat detection purposes.
    - Cloud agents, implemented in remote/cloud servers of ANASTACIA framework. These agents are implemented as system agents that captures relevant data and perform local analysis that can be communicated to other agents for distributed analysis or to the centralized monitoring service for global analysis (i.e. system level anomaly detection).
- Initial integration results of monitoring agents achieved during ANASTACIA framework with latest updates as of the release time of this deliverable. The section describes monitoring capabilities for each agent.
- Plan for future work which is formed for next steps to illustrate final realization of monitoring agents inside ANASTACIA framework.
- Summary of the current work completed for initial part of ANASTACIA technology demonstrator.

1 INTRODUCTION

1.1 AIMS OF THE DOCUMENT

The aim of this deliverable is to provide overview of the initial implementation and integration work that has been done for monitoring agents in ANASTACIA framework. In particular, the deliverable explains:

- State of the art and beyond state of the art on monitoring and detection agents.
- Current development status for ANASTACIA agents.
- Results of current agent integration work.
- Plan for remaining part of the project.

Chapter 2 explains main innovations and progress of the agents. Chapter 3 elaborates on agent placement in overall ANASTACIA architecture. Inside, reader will find information about the agents that were deployed in ANASTACIA test bed. Further in the same chapter more detailed information is included about monitoring capabilities of the agents with division to local and cloud types of agents. Next chapter gathers description of agent threat detection capabilities. Finally, the work is summarized with work plan towards final agent implementation report.

1.2 APPLICABLE AND REFERENCE DOCUMENTS

This document refers to the following documents:

- D1.3 – Architecture Design with latest architecture refinement ***[1]***.
- ANASTACIA architecture diagram ***[2]***.
- D2.2 – Attacks and Threats Analysis and Contingency Actions ***[3]***.
- D4.1 – Initial Monitoring Component Services Implementation Report ***[4]***.
- D4.2 – Initial Reaction Component Services Implementation Report ***[5]***.

1.3 REVISION HISTORY

Version	Date	Author	Description
0.1	30.10.2018	UTRC	Initial draft (Took)
0.2	31.10.2018	UTRC	Content for chapter 1
0.4	08.11.2018	ATOS/CNR/UTRC	Contribution for chapters 2-5
0.5	12.11.2018	CNR/UMU/UTRC	New contribution to chapter 2 and 3.
0.6	16.11.2018	MMT	MMT content for chapter 2 and 3.
0.7	19.11.2018	UTRC	Chapter 4 and 5 content
0.8	28.11.2018	CNR, OdinS, MMT	Chapter 4 contribution
0.9	07.12.2018	UMU, UTRC	Internal review

1.4 ACRONYMS AND DEFINITIONS

Acronym	Meaning
AAA	**A**uthentication, **a**uthorization and **a**ccounting
AT	**AT**tention
DCapBAC	**D**istributed **C**apability **B**ased **A**ccess **C**ontrol
DPI	**D**eep **P**acket **I**nspection
DoS	**D**enial **o**f **S**ervice attack
ECDSA	**E**lliptic **C**urve **D**igital **S**ignature **A**lgorithm
FENG	**Fe**ature **Eng**ineering (part of the Data Analysis agent)
IDS	**I**ntrusion **D**etection **S**ystem
IT	**I**nformation **T**echnology
JSON	**J**ava **S**cript **O**bject **N**otation
MiTM	**M**ain **i**n **T**he **M**iddle (Attack technique)
MRTG	**M**ulti **R**outer **T**raffic **G**rapher
NFV	**N**etwork **F**unctions **V**irtualization
OT	**O**perational **T**echnology
PCAP	**P**acket **Cap**ture
PoC	**P**roof **o**f **C**oncept
REST	**Re**presentational **S**tate **T**ransfer
SoC	**S**ystem **o**n **C**hip
SDN	**S**oftware **D**efined **N**etworking
SEP	**S**ecurity **E**nforcement **P**lane
VDSS	**V**erdicts and **D**ecision **S**upport **S**ystem
VNF	**V**irtual **N**etwork **F**unction

2 BEYOND STATE OF THE ART - AGENTS

With the advent of Internet of Things (IoT), it is well known that millions of devices will be connected to the existing network infrastructure. As a result, monitoring and detection are expected to get more complex for administrators as networks tend to become more heterogeneous. Therefore, the addressing for IoTs would be more complex given the scale at which devices will be added to the network and hence monitoring and detection are bound to become uphill tasks due to management of larger range and variety of nodes. On such occasions, agents can capture both wireless and wired segments for monitoring and detection purpose to reduce the complexity at infrastructure level.

All the nodes in the network provides monitoring data but an efficient system should able to collect this data in effective manner and present it to administrator. Also, intrusions should be identified effectively from such collected data. Some of the monitoring and detection agents which could meet this criterion are: Nagios ***[14]***, Ntop ***[15]***, Zenoss ***[16]***, MRTG ***[17]***, Snort ***[18]*** and Suricata ***[19]***. Zenoss and Nagios are mostly well known agents since they could monitor some aspects of the network. On the other hand, the functionality most of these agents are somewhat limited since they were primarily designed to monitor IP traffic only. To this point, the agents in ANASTACIA cover wide range of data including both operational and network data. This flexibility allows to dimension ANASTACIA agents to use, choosing and adapting it to the infrastructure layout. Moreover, the traditional agent security approach does not work for IoT. One should need to consider IoT supported light weight agent approach. Therefore, using our approach to one can get full visibility of all devices and be able to control and secure these devices ***[20]***.

Beyond state of art:

Our agents are deployed on IoT nodes where they can sniff wireless traffic and provide network information about L0-L7 network layers to cloud based agents. Cloud based agents are deployed inside ANASTACIA monitoring and reaction plane where received information is filtered, processed and used in attack verdict generation. Note that one set of agents serve lower layers of IoT network stack while other set of agents work on upper layers. The agents developed in ANASTACIA framework are independent of underlying monitoring structure and can be deployed to various parts of security perimeter that includes local and remote security infrastructures (local and cloud based IT systems).

Compared to state of the art on monitoring and detection agents, ANASTACIA brings following additional functionalities to agents,

- **On-demand**
  ANASTACIA agents could be delivered and deployed on demand (at the edge of the IoT network) using NFV with 5G slicing. The idea of deploying the monitoring agents as VNFs is planned to address in next year. However, ANASTACIA framework is capable of providing such on-demanded solutions.

- **Interoperability**
  ANASTACIA framework supports external agents and it will be instantiated by implementing proxy agent that will close gap with existing or new cyber security systems. ANASTACIA framework provides two paths to become interoperable with external agents. The external agents adapt either its own interface to ANASTACIA framework by sharing information with other components via Kafka broker or an adapter will be provided from ANASTACIA framework to adopt to given cyber security product.

- **Plug and play**
  ANASTACIA framework provides a flexible way such that any agent can deployed in various forms on multiple cross-platform applications. The agent has capability to be deployed either in stand-alone environments or cloud environments.

- **Reliability**
  Agent reliability has been one of the key building features during the implementation and integration phase of ANASTACIA project. Our agents has capability to work in 24/7 fashion thanks to modular design where each connector module responsible for establishing connections with other ANASTCIA components is working independently. This means that if one of the external connections fails the agent will wait until it will be re-established again. Those situations were present initially during integration when project partners were updating their components and this feature enabled agent to work under those circumstances. ANASTACIA reliability is achieved by implementing Kafka broker system, REST APIs, Buffer, and Logger.

- **Scalability**

  This is achieved by implementing two approaches:

  - **Single agent performance** – ability to consume events with keeping performance of the system is in good health.
  - **Multi-agent scenario** – As the amount information that needs to be check grows the multi-agent infrastructure can be deployed to cover large SEP. In this case SEP can be divided into sections. Section can be building floors or larger sections of the building (i.e. company premises – multiple floors of the building) or security perimeter under specific security clearance. In this case large event processing and anomaly detection is spread across multiple agents. This approach off-loads ANASTACIA framework and provides no single point of failure desired in critical infrastructure deployments.

- **Learning engine for detection**
  ANASTACIA developed a detection approach based on CP-based decision model that 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. After learning important relations, the model discards un-satisfied relations, and consequently update the model with best possible relations and features of IoT sensor nodes. The learning engine prunes this process until it achieves a best model with least number of relations and features. Specifically, this engine identify the sensors are involved in breaking the relation and what are the set of relations are broken Following this fashion, the model is further tuned. On top of this model, ANASTACIA build new detection rules in order to detect attacks in network data.

3 MONITORING AGENTS IN ANASTACIA FRAMEWORK

The monitoring component of ANASTACIA follows a flexible approach where new monitoring agents can be easily plugged and unplugged as depicted in Figure 1. Monitoring data can be extracted from many different locations, devices and other assets. It allows to manage both OT data (for example, temperatures measured by a temperature sensor) and IT data (for example, http requests to a web server or signalling data in a cellular network). This flexibility allows to dimension the agents to use, choosing and adapting it to the infrastructure layout. For example, a segment of an infrastructure where wireless devices are used might require an anti-jamming detector (in order to detect jamming attacks against the wireless spectrum). Wired parts of the infrastructure wouldn't need wireless detectors but would require network sniffers to detect intrusions or targeted attacks against a concrete asset. To this point, ANASTACIA provides agents which can capture both wireless and wired segments. For an example, MMT agent could be a potential candidate for such capabilities.

In ANASTACIA, monitored data is retrieved from different parts of the infrastructure and forwarded to a messaging broker which filters and transform the data to send it to the incident detector which looks for anomalies and security threats. The incident detector is capable of correlating events received from different monitoring agents, combining and analysing them to find potential threats. For instance, anomalies detected in the operational data of several temperature sensors combined with evidences from a network sniffer can be combined to infer an ongoing attack.

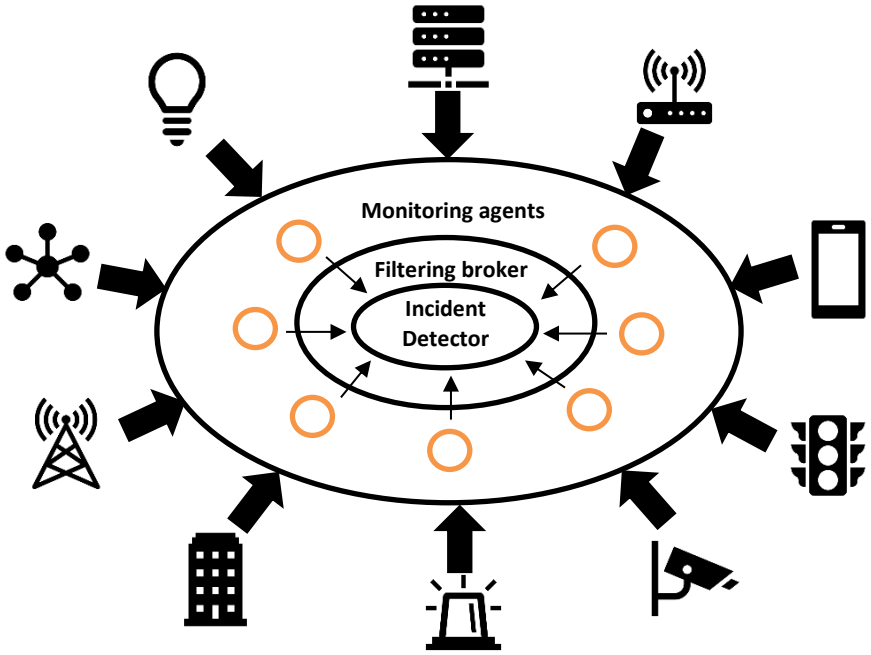

Figure 1 – ANASTACIA monitoring layers.

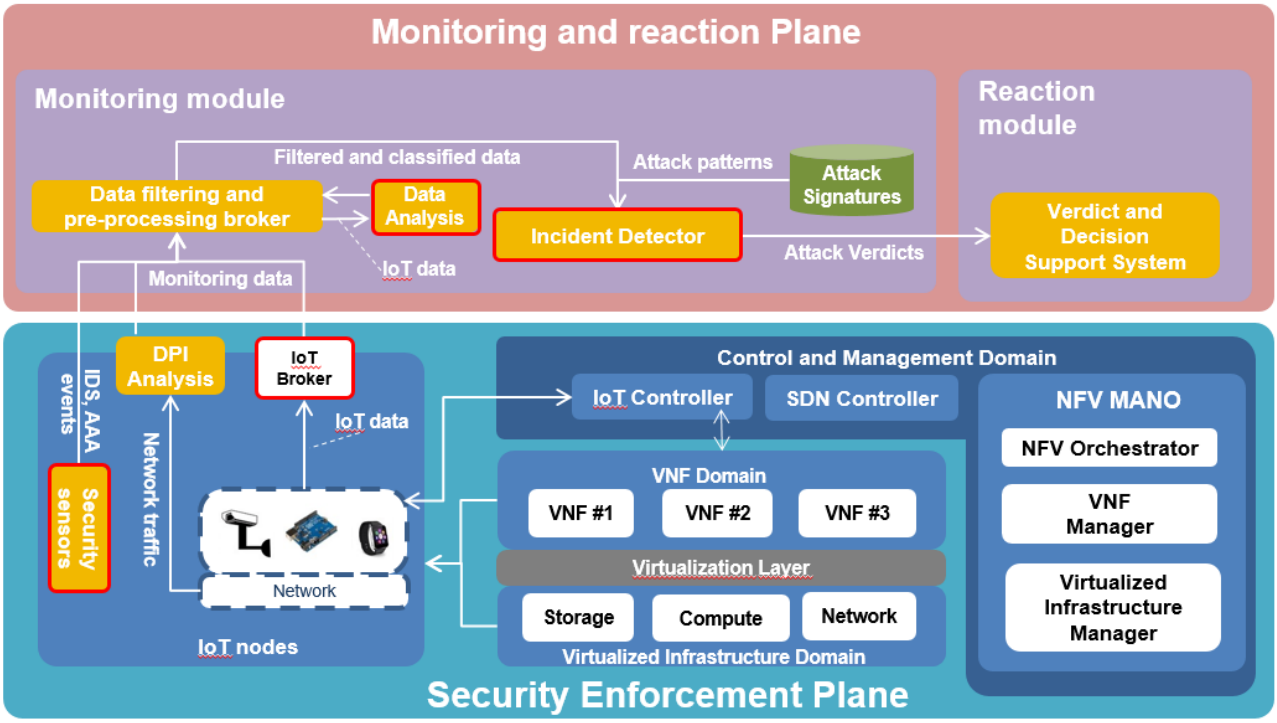

Figure 2 - ANASTACIA architecture (simplified) with monitoring agents marked red.

On Figure 2 we can observe assembly of local and cloud based agents that are used in different parts of ANASTACIA system to monitor SEP and enable detection of advanced threats. Local agents are deployed on IoT nodes where they can sniff wireless traffic and provide network information about L0-L4 network layers to cloud based agents. Cloud based agents are deployed inside ANASTACIA monitoring and reaction plane where received information is filtered, processed and used in attack verdict generation.

In the current version of the architecture, following agents were deployed to enable cyber security threats detection in ANASTACIA framework. Note that our agents are categorized into two parts 1) local agents 2) cloud agents. Local agents serve lower layers of IoT network stack while cloud agents work on upper layers. ANASTACIA prepared three types of agents to help detect malicious activities at network level:

- Zigbee IoT agent – monitors network level communication between devices using Zigbee protocol.
- MMT IoT local agent – looks into 6LowPAN traffic for abnormal activity by using DPI techniques and sending network activity to ANASTACIA framework for further analysis,
- Local IoT agent – Protects IoT devices and networks against unauthorized action in SEP.

On cloud/remote side ANASTACIA framework provides system protection by implementing two agents:

- MMT probe cloud agent – cooperates in tandem with MMT IoT local agent to deliver high accuracy of anomalies detection in SEP on a network level.
- Data analysis agent – check overall system state for any anomalous behaviour and reports detections to reaction part of ANASTACIA framework.

3.1 LOCAL AGENTS

ANASTACIA aim is to provide pro-active and semi-autonomous security for wide range of cyber-physical networks against potential security breaches and attacks. Considering the ANASTACIA platform, several plugins have been developed over IEEE 802.15.4 stack protocols in order to cope with different detection and reaction capabilities. In particular, we consider different IoT network technologies like 6LoWPAN and Zigbee as well as CoAP as IoT application protocol for IoT networks. Therefore, due to the complexity and

variety of considered, IoT may pose serious threats. The developed local agents consider such different technologies into account. This section illustrates local agents developed for initial ANASTACIA framework demonstrator. Each of them is monitoring IoT SEP in a different way to provide full coverage against known and unknown security threats. For an example, Zibbee local agent supports IoT network over Zigbee protocol while MMT agent provides monitoring capabilities over 6LoWPAN and CoAP protocols. Both Zigbee and MMT agents monitors at operational level. However, local IoT agent implements the monitoring capabilities at authorization level of devices using CoAP. Therefore, the considered local agents cover a wide range of monitoring and detection capabilities in IoT networks.

3.1.1 Zigbee Agent

In order to identify possible attacks on IoT networks, the ANASTACIA team has developed a set of agents mainly used to monitor messages exchanged in dedicated IoT environments. In particular, CNR investigated and implemented IoT security by focusing on wireless protocols and by proposing innovative cyber-attacks and related countermeasures. By embedding a dedicated scenario on the ANASTACIA framework, it is possible to evaluate the efficiency of such threats and in general to protect IoT devices and networks from innovative threats.

In particular, CNR investigated IoT networks based on the Zigbee protocol. Such study led to the identification of an unknown vulnerability that provides a malicious user the ability to disconnect a sensor from the network (for instance, to lead a DoS, or to make it connect a different network), by exploiting AT Command packages ***[Vaccari, 2017]***. By considering the physical hardware adopted during the study, his kind of packets is automatically processed by the Zigbee controller, hence, it is not able to manage it from the micro-controller of the IoT device. Because of this, protection from such threat should be accomplished through some sort of wrapping techniques.

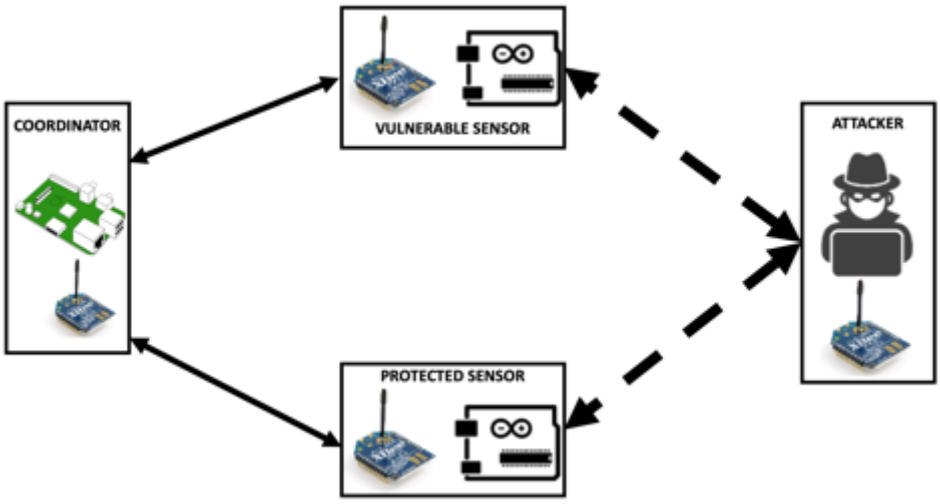

Figure 3: The scenario depicting the remote AT commands IoT attack

In this context, local agents are represented by IoT devices connected to the Zigbee network. Such devices are supposed to autonomously monitor their status, in order to identify possible exploitation of the proposed threat. When an attack is identified, the alert is sent to the Zigbee coordinator that notifies other ANASTACIA platform modules. Similarly, disconnection of IoT devices will be monitored and notified, by the Zigbee coordinator.

3.1.2 MMT IoT Agent

The Montimage Monitoring Tool (MMT) was designed as a piece of software to inspect and monitor security properties on traditional Ethernet networks. Since MMT is a DPI-based tool, it analyses the packets on the network and extracts information from them by using specific parsing and extraction rules for each protocol analysed. In this sense, the parsing and extraction rules that are used in traditional Ethernet network cannot be applied to the IoT networks, since the latter make use of particular protocols adapted for devices with low memories and low power consumption, thus compressing the transmitted information between devices. Moreover, the DPI technology of MMT makes use of Linux network interfaces to capture packets from the network. Since the Linux kernel does not know how to handle these packets, it drops them, making impossible to use Linux-based technologies to provide DPI features.

Considering these difficulties, an adapting layer has been added to the MMT architecture, in order to correctly capture packets from IoT networks using the IEEE 802.15.4 and 6LowPAN protocols. Figure 4 summarizes the architecture of this adaptation layer, which includes both the Local Agent and the Cloud Agent of MMT.

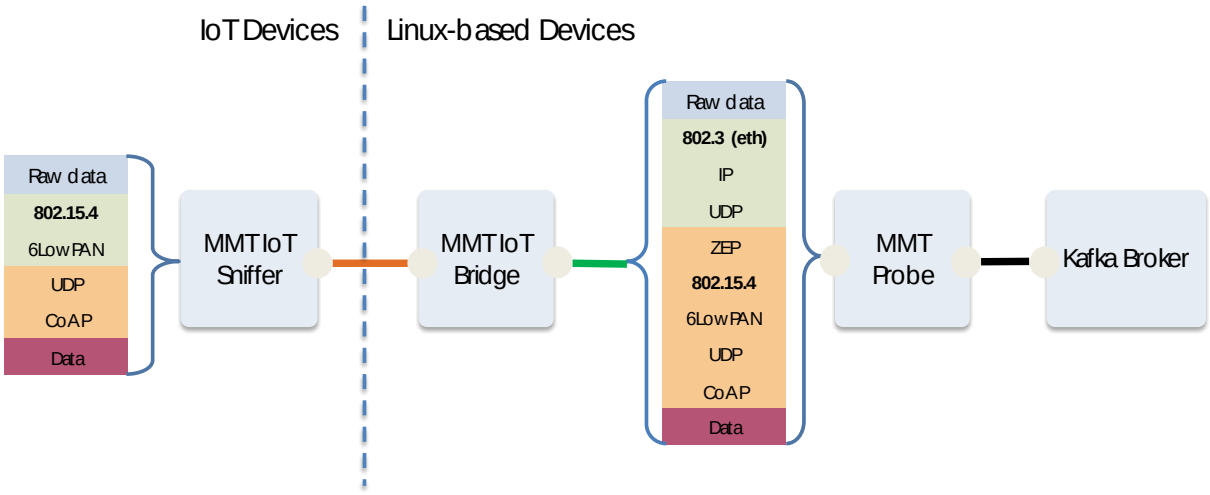

Figure 4 Architecture of the IoT Adaptation of MMT.

The Figure shows the principal components of the whole MMT solution adapted to work in IoT environments, being two of them the principal components acting as local agent:

- MMT-IoT Sniffer: This is a Contiki-based[1] firmware capable of being installed in IoT devices. Adaptations of this firmware have been made, consisting mainly in modifying the network driver to clone the traffic from the radio interface and transmit it via a USB communication line (orange line between the sniffer and the bridge).
- MMT-IoT Bridge: This is the first Linux-based component of the architecture. It is a software tool that reads the packets sent by the sniffer – through the USB line – and sends them to the MMT-Probe. Since MMT-Probe uses Linux interfaces as the source of packets, it is not possible to directly send the extracted packets through a network interface, since the Linux kernel (where MMT-IoT bridge is running) does not know how to handle the IoT protocol, and therefore discards the packets. To cope with this problem the IoT bridge tool performs an encapsulation of the packets, using the Zigbee

[1] http://www.contiki-os.org/

Encapsulation Protocol (ZEP) – a standard format for encapsulating IoT packets. After this layer, the packets are encapsulated using normal UDP, IP and Ethernet headers, finalizing the encapsulation process. Finally and once the packets have standard Ethernet headers (see the format between MMT-IoT Bridge and MMT Probe in Figure 4), they are ready to be handled by the Linux kernel, therefore they can be sent using one of the following options: via a TUN/TAP interface for local delivery of the data, when both Bridge tool and MMT Probe are running in the same machine; or using Linux raw sockets, to forward the packets to a remote machine, when the Bridge toll and MMT-Probe are running in different machines.

The two components described above have to be deployed in two different devices (the sniffer on an IoT device and the bridge on a USB-enabled, Linux-based machine) and, both of them have to be deployed in the premises of the IoT network. This is required due to the fact that the sniffed packets are transferred from the sniffer to the bridge using a USB line, which limits the possibility of splitting these two components. It is worth mentioning that despite this USB requirement of the device running MMT-IoT bridge, it has been designed to be executed even in SoC devices – like Raspberry Pi – offering more flexibility for deploying this technology.

3.1.3 Local IoT agent

The local IoT agent is able to enforce the access control to the resources it offers, by means of validating a DCapBAC token. DCapBAC provides the user with the necessary authorization permission to access resources hosted by different devices not having to verify the permissions every time with a centralized entity, which is where the distributed nature of the process comes into play.

The DCapBAC process, exemplified in Figure 5, involves the *Capability client*, the *Capability manager* and a *Policy Decision Point* (PDP). The process is as follows:

- The Capability Client requests a capability token for a specific target and a set of permissions (i.e., actions to perform on some resources or services).
- The Capability Manager asks the PDP if the subject has the permissions requested.
  – The PDP responds to the Capability manager request if the subject has the specified permissions
  – The Capability Manager generates the Capability Token. An example can be seen in Figure 6.
- The Capability Manager sends the Capability Token to the Capability Client.
- The Capability Client can use the token at its discretion not having to verify again if it has permissions to access the resources during the validity period of the token.

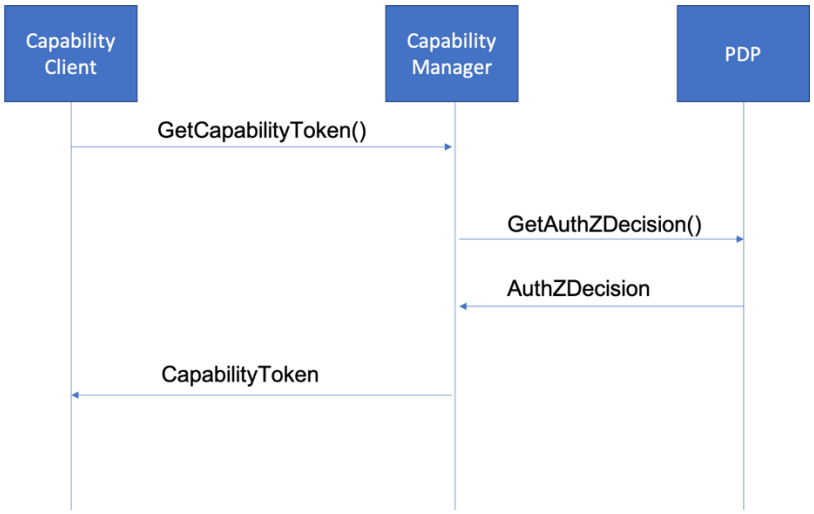

Figure 5 Process of Obtaining a Capability Token

The Capability Token is instantiated in a JSON document that contains all the necessary information to state the permissions of the client, as well as to verify the validity of the token itself.

An example of a Capability Token is shown in Figure 6.


```
{
  "id": "c3akpikk34qn04oq142lha43ii",
  "ii": 1535733427,
  "is": "capabilitymanager@um.es",
  "su": "MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAEAhPN60KBnrwsANw0dP6W11AAU60VUVed/Mpj8R1OOtC/
  Hmhdu2RxnZF6IgpqgWXzmBH0WVxA==",
  "de": "coaps://ca:ffe::de:caf",
  "si": "MEYCIQCEt4wS5uP6wQ+pY+XfnBW5ibWp0h7fEWLGQlQaUDs5VwIhAO889S4g/CZhqBbZEObnB6ShIlN7lNjww7rfLSmtBID/",
  "ar": [
    {
      "ac": "POST",
      "re": "/service"
    }
  ],
  "nb": 1535734427,
  "na": 1535744427,
}
```


Figure 6 Example of Capability Token

The Capability Token provides the fields to verify and state the permissions of the client. The "*id*" field is used to unequivocally identify a capability token. The "*ii*" field states the time at which the token was issued. The "*is*" field states the identity of the entity that issued the token and, therefore, the signer of the capability token. The "*su*" field references to the subject to which the rights from the token are granted. The "*de*" field is a URI used to unequivocally identify the device to which the token applies. The "*si*" field contains the digital signature of the capability token. The content is a signature in ECDSA that is represented by two values. Each half of the "*si*" field corresponds to one of these values using Base64. The "*ar*" field represents the set of rights the subject has been granted. The "*ac*" field represents the action (e.g., GET, POST, PUT, DELETE) and the "*re*" field represents the resource in the device for which the action is granted. For further details about the Capability Token specification and it use can be found in ***[1]***.

In the next figure we can see a simple diagram that shows how a client interacts with the IoT device using a Capability Token. A Client uses the Capability Token it just received from the Capability Manager, sending it along in a Request to the IoT device that hosts the resources or services it intends to use. This capability token is then processed by the IoT device, that parses it and validates it according to the lifetime and signature from Capability Manager with asymmetric cryptographic based on ECC suite for constrained devices. In the case the validation is correct, the IoT device sends a response to the client with the requested information or the confirmation that the action that had to take place has been performed.

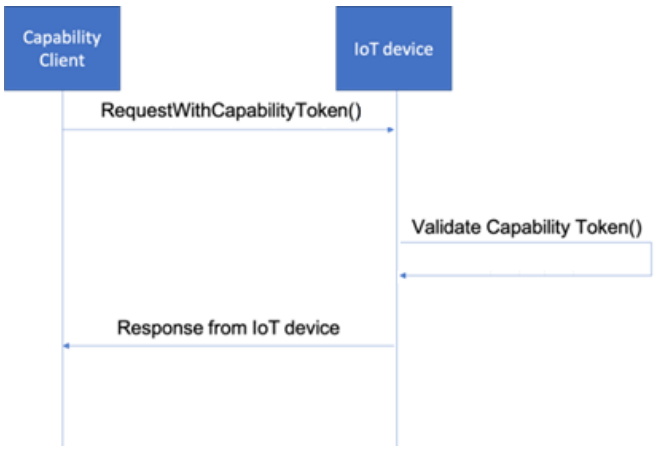

Figure 7 Diagram of the use of a Capability Token

3.2 CLOUD AGENTS

Cloud agent chapter describes agents developed in ANASTACIA framework that are independent of underlying monitoring structure and can be deployed to various parts of security perimeter that includes local and remote security infrastructures (local and cloud based IT systems).

3.2.1 MMT Agent

As mentioned before, Figure 4 shows the components acting as Local Agents. However, it also shows two other components which are part of the MMT adaptation for the IoT network that can be considered as cloud agents:

- MMT Probe: This module executes the incident detection logic. It receives the encapsulated packets via a Linux interface (either coming from the same or a remote machine) and parses it to extract the information of the transferred packets. Using this data, the MMT-Probe module evaluates rules in order to detect security issues in the analysed flows, using the embedded MMT-Security library. Depending on the requirements and configurations, MMT-Probe can detect SQL injection and ICMP flooding among other attacks. Finally, MMT-Probe is also capable of aggregating the extracted information in order to offer statistics about the detected flows, information that can be used by other incident detectors for their security analysis.
- Kafka Broker: This module is the general communication broker of the ANASTACIA platform. All the detected incidents and the generated statistics will be published in this channel. This final step makes the extracted information available to other modules, allowing complementary security analysis using information extracted from different sources.

3.2.2 Data Analysis agent

Data analysis agent performs system level monitoring by aggregating information from SEP using Kafka broker. Details of data consumption scenario were described in detail in D4.1 ***[4]***. In this deliverable we will focus on information processing inside agent. The processing part is being divided into 2 parts:

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

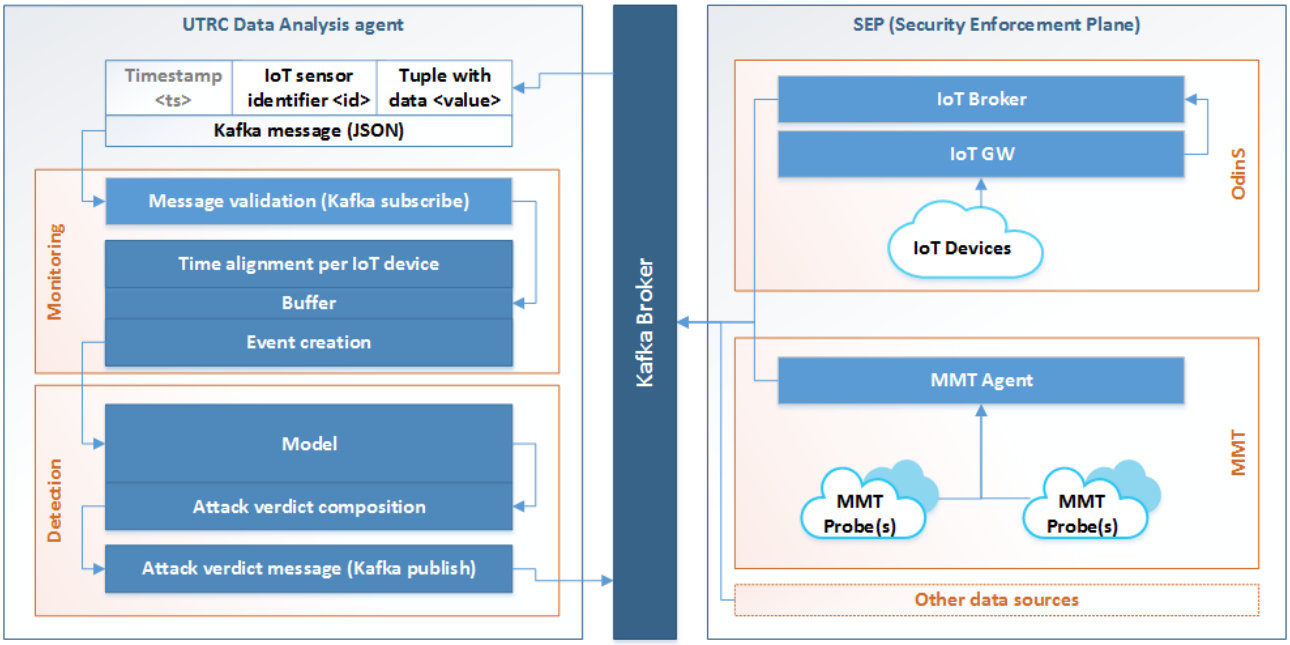

Figure 8 Data analysis agent internal monitoring architecture.

Internal agent architecture was illustrated on Figure 8. It depicts two main parts of agent (monitoring and detection). Both sides are preconfigured to adjust to various system monitoring capabilities for anomaly detection (temperature, pressure, IR presence, etc…). The common interface between monitoring and detection part enables agent to consume and process various data sources that are available on SEP. The only condition is that message should contain IoT device identifier and value that will contain device current monitored state (both are marked blacked inside JSON message). Optionally if the security perimeter is not co-located with ANASTACIA framework timestamp can be added for higher detection accuracy at detection level.

This mechanism enables other data sources to deliver more information that can be used in more accurate anomaly detection and IT system security evaluation. Agent process each message in three stage routine by implementing:

- **Validation** – each message is validated in accordance to predefined data exchanged protocol. Each local agent provides data in different format. If validation fails it will be logged for further investigation. The validation process is protected against multiple failures so it will continue to provide validation for all types of messages defined in agent configuration.
- **Parsing** – This step will divide each message into at least three main parts:
- Timestamp – will contain local time stamp that is reception time of JSON message from Kafka broker. Message supplied timestamp may be used at detection phase where it will be combined with monitored IoT sensor data value(s),

- Identifier – IoT device uniquely identifier in SEP. The IoT device ID needs to be also added into *feng* (feature engineering) section of detection model for accurate verdict generation. Example of model configuration with IoT sensor identifiers in *feng* section is included in Annex 1,
- Value – represents parsed value of monitored data extracted from JSON message. In case that external time source information is included the parser will add it automatically to the value that will be stored in the buffer.

- **Conversion** to buffer event – Once parsing step is completed the parsed values are assembled into event that is pushed to buffer. Each event will contain three fields mentioned above.

Event is then added to data buffer that will store all events. The number of events stored per each IoT measured parameter is automatically determined based on feature engineering configuration for each of them. Figure 9 illustrates internal event processing and internal buffer structure used in data analysis agent. At some pre-set time intervals, monitoring part will fetch information for each of the parameters stored in the buffer. Next, data buffer will construct frameset that will contain current parameter sample and N historical samples of data set for given parameter. N represents number of historical samples of the same parameter and it is preconfigured in agent configuration file for each of the parameters in feng section (See Annex 1 – feng section of the configuration).

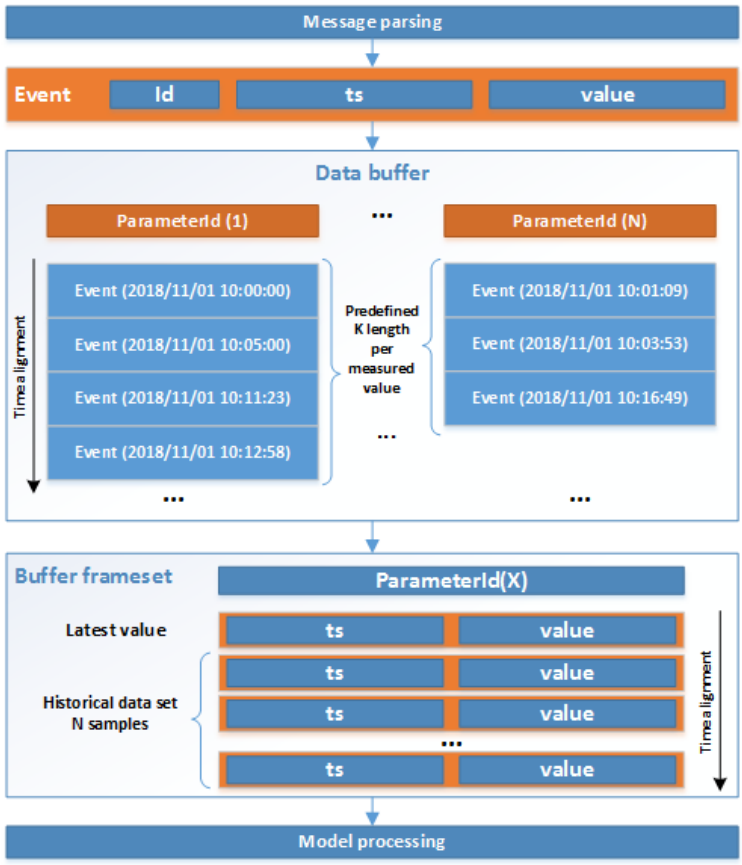

Figure 9 Data buffer processing and internal structure.

Once system state with historical data is created, it is passed to the model for detection evaluation and later the agent will generate appropriate attack verdict that will be sent to VDSS component via Kafka broker. All events are stored in time ordered fashion so when model requests them they are fetched without need to sort them out. The idea behind the scenes is to abstract model processing from various data sources and uniform system view (data cleaning) before sending information to detection process.

From data buffer configuration perspective, the buffer is initialized based on model configuration (*feng* configuration). This way when agent starts the buffer automatically adopts to model parameters (features) configuration to deliver required framesets for detection analysis.

For example, with configuration presented on Figure 10, the maximum length (marked red) for parameter (Sensor1) in the buffer will be assessed as 5.

```
# Features configuration
features:
  Sensor1:
    type: "temp"
    location: "Ground Floor Visitors Wait Hall"
    temp:
      raw:
        function: "raw"
        length: 1
        clusters: 3
        dtype: continuos/int/str
        distFunc: enum/euc
        min:
        max:
      diff:
        function: "diff"
        length: 2
        clusters: 4
        arguments: [
          "Sensor1_temp_raw"
        ]
      avg:
        function: "avg"
        length: 5
        clusters: 4
        arguments: [
          "Sensor1_temp_raw"
        ]
```

Figure 10 Example configuration listing with parameter size buffer allocation.

In order to be able to analyse the proposed data an agent application, it is essential to setup the IoT network with real nodes to judge the feasibility of application, because a network of real nodes provide reliable data about the application scenarios. The real network set-up is suggested for a network of small number of nodes as it is economical to setup smaller networks of application. If the network consists of hundreds of nodes, then it proves to be expensive to setup a physical network even before predicting the behaviour. Moreover, our data analyis agent framework needs continuous and large portions of data from larger network for training purpose. To this end, we used the network simulators that can be used to understand the behaviour of the communication networks by using mathematical models to reproduce different environmental conditions and run experimental simulations of various device frameworks. In a similar fashion, the network simulators can be used to replicate the IoT devices and large network of hundreds of nodes in different topology can be created to understand the behaviour of these devices.

Simulation experiments for training:
We considered the NS-3 ***[7]*** network platform which is a complex, yet flexible, robust, reliable and efficient simulator. It was specifically designed for research and academic purposes. Based on certain comparisons, analysis and inferences, the NS-3 simulator is found to be suitable for network simulations related to large networks. The network behaviour can be analysed in different perspectives such as performance, flexibility, robustness, customization etc. The performance of a network is basic and crucial aspect of a network.

Therefore, analysing a network in terms of performance can be considered as the first step towards complete understanding of network with data analysis agent.

NS-3 has sophisticated simulation features, which include extensive parameterization system and configurable embedded tracing system, with standard outputs to text logs or PCAP (tcpdump). It is an object oriented for rapid coding and extension. It has an automatic memory management capability as well as an efficient object aggregation/query for new behaviours & states, like adding mobility models to nodes. Moreover, NS-3 has new capabilities and modules ***[12]***, such as handling multiple interfaces on nodes correctly, efficient use of IP addressing and more alignment with Internet protocols and designs and more detailed 802.11 models, etc. The Simulation Network Architecture looks just like IP architecture stack. The nodes in NS-3 may or may not have mobility. The nodes have "network devices", which transfer packets over channel and incorporates Layer 1 (Physical Layer) & Layer 2 (Data Link layer). The network devices act as an interface with Layer 3 (Network Layer: IP, ARP). The Layer 3 supports the Layer 4 (Transport Layer: UDP, TCP), which is used by the Layer 5 (Application Layer) objects.

Attack generation using NS-3:
Main experimentation and simulation effort was focused on generating denial of service and man-in-middle attacks in the simulated network. Flooding attack is a denial of service attack, in which a compromised node floods the network by sending large number of fake control/data messages to existent/non-existent nodes in the network or by streaming large volumes of useless DATA packets to the other nodes of the network. Man-in-middle attacks is also performed similarly using NS-3 framework.

After attack generation, we have simulated IoT network resembling real nodes in NS-3. The simulated data has been fed to Data agent for anomaly detection. Whole simulation system with agent data feed was illustrated on Figure 13.

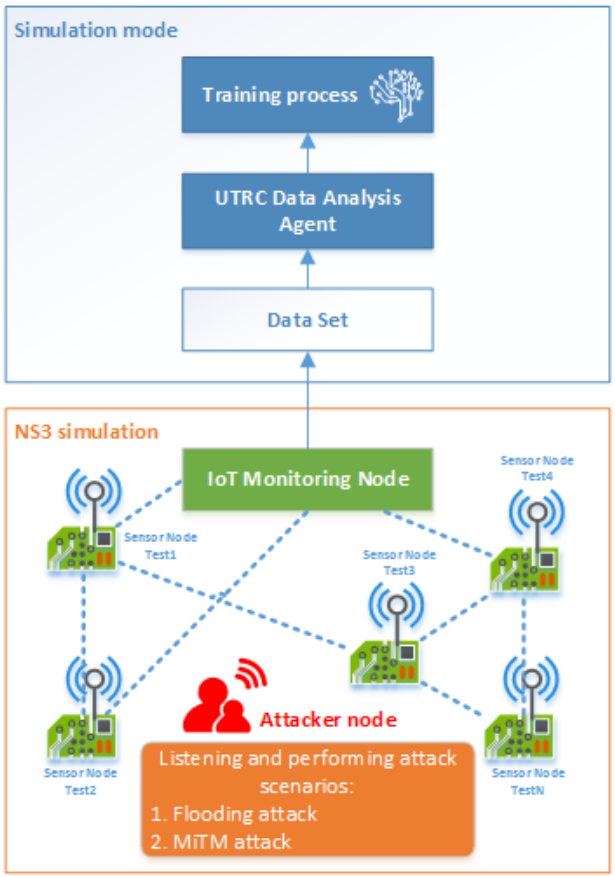

Figure 11: NS-3 simulation test bed with agent data feed to simulation mode.

4 AGENTS INTEGRATION RESULTS

Monitoring agents in ANASTACIA framework are designed to detect known and unknown threads. In section below, current state of agent's integration within ANASTACIA framework is illustrated with up to date results discussion.

4.1 ZIGBEE AGENT

As previously mentioned, the proposed innovative attack on IoT contexts is based on the AT Commands exploitation. Such attack aims to reconfigure IoT devices to disconnect them from the Zigbee network ***[6]***[2]. The agents implemented on the IoT devices are responsible for monitoring the device status and verifying that all the parameters are correct. In case the device is affected by a reconfiguration attack, such alert information is forwarded to the IoT coordinator, and the device is designed to mitigate the attack (by autonomously reconfiguring itself). Since not all the devices may embed a detection and mitigation system, the IoT coordinator is supposed to monitor devices status to identify disconnections, hence report them to the other ANASTACIA modules through Kafka broker.

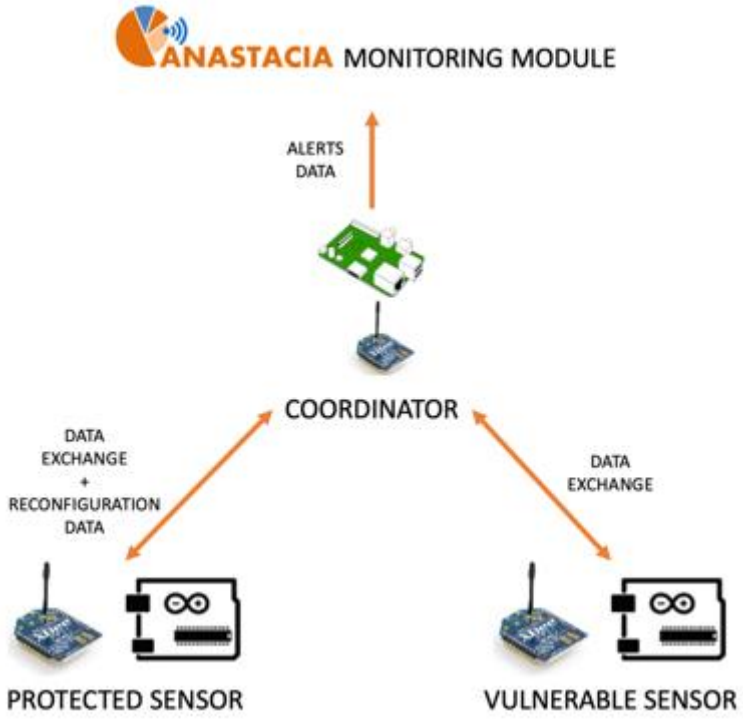

Figure 12: An overview of a remote AT command IoT attack detection approach

By using this approach, the attack is autonomously identified and mitigated and the ANASTACIA platform is notified of the exploitation, thus reducing the possible damage to the network and by ensuring security of exchanged information.

[2] Although a variation of the attack may lead to different effects, our study was focused on disconnection of the device from the network (e.g. to lead a DoS, or to make the device connect a different network).

4.2 MMT IoT Agent and Probe

MMT-Probe is capable of detecting security incidents by verifying security properties. Such properties are expressed in XML files, which are compiled into MMT-Security plugins. Finally, these plugins are loaded by the MMT-Probe and actively verified using its DPI and ML technologies. This feature is supported by the modular approach applied of the MMT-Security library, which allows installing new plugins – security rules – and loading them dynamically.

For the first iteration of the ANASTACIA project, two use cases have been covered by the MMT-Probe, which are covered in the following subsections.

4.2.1.1 MEC.3

This use case comprises an ICMP flooding attack on a Smart Camera. The attack is performed using compromised IoT devices towards a Smart camera within the same IoT network. Being this said, the detection of this attack has to be done inside the IoT network, since the traffic generated by the attack will never leave the IoT network.

Considering this, both the MMT-IoT Sniffer and Bridge components were deployed in the testing environment, in order to sniff the IoT traffic from the IoT network and forward it to the MMT-Probe component. With this deployment, the innovative MMT-IoT components were correctly integrated with the rest of the ANATSACIA platform, allowing to effectively sniff and process IoT packets using adapted DPI tools such as MMT-Probe.

Once having access to the IoT traffic, an ICMP flooding detection rule was specified using the requirements found in D2.2 ***[3]***. In that document, it is specified that any flow containing more than 3 ICMP ping requests packets within a second has to be considered as an attack. Considering this, the detection rule was written to consider the first two ICMP packets seen in the flow as a context[3], and then detect the third within a second as a trigger[4] of the rule. More details (including the XML code that defines this rule) can be found in D4.1 ***[4]***.

All the aforementioned features were integrated in the ANASTACIA platform and connected to the Kafka Broker as shown in Figure 4. This level of integration allowed testing MEC.3 test case deployment, which permitted extracting IoT packets in real time, and detect the MEC.3 attack within one second of the starting of the attack.

4.2.1.2 BMS.3

This use case copes with an SQL injection attack performed by a compromised IoT device. The attack is performed towards an SQL server located outside the IoT network, since it outside this restricted network a where a data storage server is more likely to be deployed. In this sense, the malicious traffic will go through the IoT router, being transformed to normal Ethernet frames (according to the routing protocol) in order to make the traffic reach the SQL server. This particular characteristic suggests that the detection has to be done in the incoming link of the server – which is a normal Ethernet link – thus requiring only MMT-probe (with no local agents) deployed to sniff the incoming traffic of the server.

Considering this situation, a new detection rule has been developed in order to detect the SQL injection attack in IPv6 traffic. In particular, the address translation (and the protocol change) made by the IoT RPL router made this task particularly challenging, requiring translating back the IP addresses to identify the attacker device.

[3] In an MMT-Security rule, a context is a condition that has to be fulfilled before checking the trigger of the attack. It can be seen as the "requisite" that needs to be accomplished before checking for the actual attack

[4] In an MMT-Security rule, the trigger is the condition that allows identifying the attack and, therefore, allows detecting it, given that the context has already been tested and checked.

On the other hand, the detection rule has to also consider the CoAP protocol of the use case in order to correctly read the packet and detect any SQL query. To this end, the MMT-Probe tool was extended with a CoAP protocol parsing plugin. This allowed us to extract all the information from this protocol (request type, resource requested, among others) including the payload, where the SQLi detection was performed. More details about how this detection is performed can be found in D4.1 ***[4]***.

Considering the particular requirements of this use case, the MMT-IoT agents (local agents) were not required and only the MMT-Probe (acting as cloud agent) was deployed in the same network as the CoAP server, outside the IoT network. This configuration maximized the protection against attacks coming not only from the IoT network but also from other sources. In addition, the MMT-Probe was also integrated to the Kafka Broker, dumping the extracted information and the detection verdicts on the common ANASTACIA channel.

4.3 DATA ANALYSIS AGENT

Data Analysis agent is developed to detect anomalies based on data fed into the model. From current state of integration in BMS.4 use case following chapters will discuss developments in areas of:

- Model detection results – where agent model accuracy result will be presented.
- Agent reliability – new built in capabilities to enable agent work in 24/7 fashion.
- Agent scalability – capability to scale up event processing and anomaly detection.
- Agent deployment capability – cloud deployment capabilities.

4.3.1 Model detection results

The agent is capable of detection threats which are defined in the Table 1 Mainly detection in real deployment scenario will depend on how much data for training and validation is available and how many model constraints are predefined to enable high level model adherence to reality.

Data analysis agent is composed of messaging wrappers, constraint programming (CP) models and buffered sensor data from IoT networks. Mainly, CP model is core component of data analysis agent, and it gather and analyse information in order to identify any intrusion. Moreover, CP model built on IoT continuous stream of data (i.e. time-series) where the time interval between successive updates could vary from milliseconds to minutes. UTRC CP model consists of network of relations between IoT sensor data. An example illustration of such network is shown in Figure 13.Using UTRC CP model, we aggregate the different types of IoT sensor data to truly model the normal behaviour of the system that is being supervised. This model is built for monitoring at system level, but it does not prevent from including in the model information about network performance if that is exposed to it. For an example, CPU consumption of a device can be included along its actual sensor data. The variety of data that we can aggregate allows the model to be as generic or as specific as the end-user required it to be. Since the model is built on relations, we can leverage from the fact that what data effects what other data type and also if the end-user is aware of this.

UTRC developed an approach learn a CP-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. After learning important relations, the model discards un-satisfied relations, and consequently update the model with best possible relations and features of IoT sensor nodes. In each iteration, *the relation between the sensor features and all other network features further verified.* Also, *we identify the sensors are involved in breaking the relation and what are the set of relations are broken Following this fashion, the model is further tuned.* The workflow for learning the model is explained in D4.1 ***[4]***.

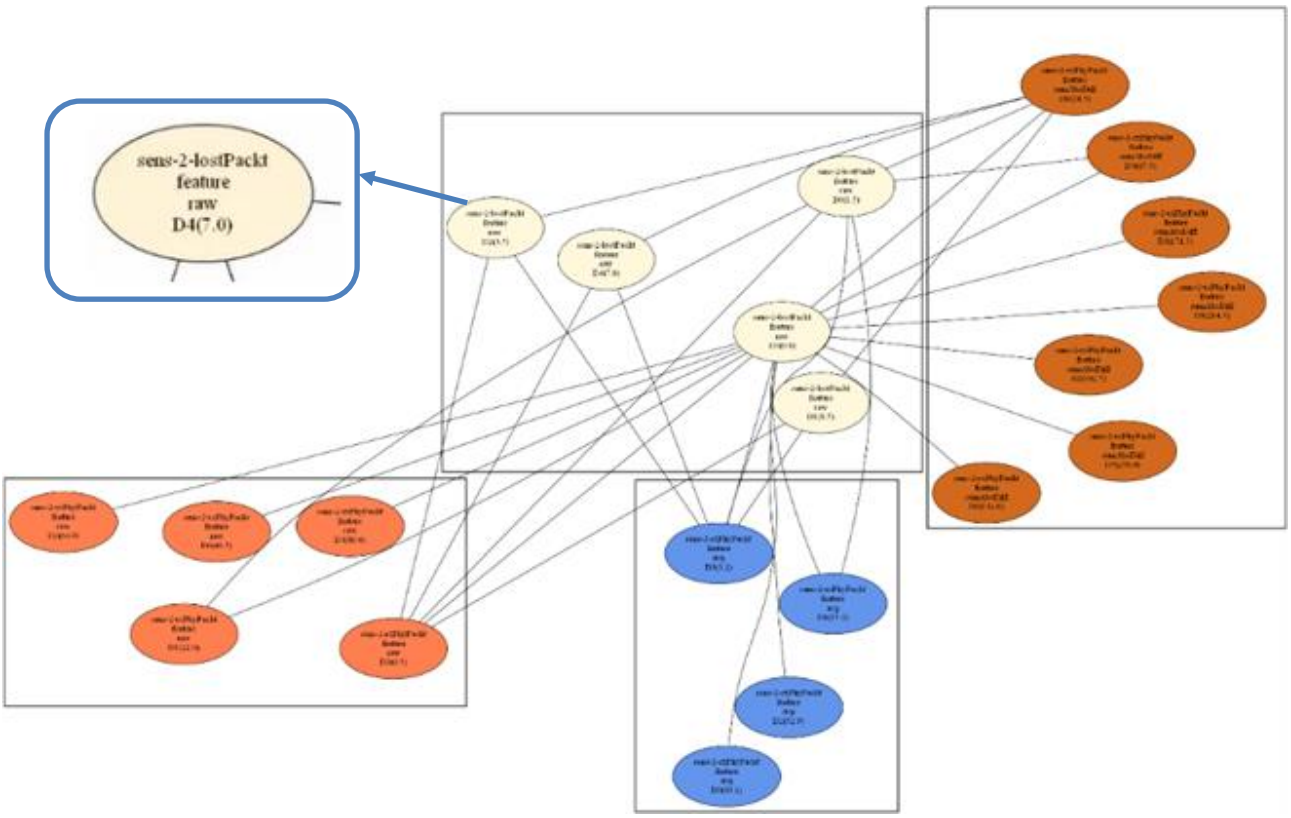

Figure 13: CP Model with relations with highlighted sensor parameter.

The developed agent is capable of detection threats that are defined in the table below. Mainly detection in real deployment scenario will depend on how much data for training and validation is available and how many model constraints are predefined to enable high level model adherence to reality. The observed data in monitored agent is compared to anomaly detection model (UTRC CP model) and a decision is returned weather the model considered this behaviour as normal or abnormal. The decision (verdict) is sent to "UTRCVerdict" topic on Kafka broker for further analysis. In online usage in every predefined amount of minutes per time window a verdict will be returned weather the model has seen an anomaly or not. However, for simulation purposes we aggregate all verdicts and calculate the models accuracy in form of a confusion matrix. We consider attack windows and non-attacked points, where an attack window is detected if at any time point the model raised an alarm. Once a simulation is performed and we obtain the sequence of verdicts from the model we compare this with the expected verdict and calculate the following confusion matrices for man-in-middle attack at specified in BMS. 4 use case.

Specifically, we have followed two approaches for calculating the confusion matrix. First, point-wise: we compare the expected verdict with the predicted verdict time-point by time-point. Second, attack-wise: an attack is defined as a time window where consecutive time-points that who's expected verdict is attack. An attack is recognized if at least one time-point from the corresponding time window was correctly recognized as an attack. Non-attack points are considered as before, point wise comparing the predicted verdict to the expected one. Attack points are not considered individually but the attack windows are counted. Moreover, point-wise is suitable for online mode, whereas attack-wise counts for simulation mode.

Table 1. Man-in-middle attack confusion matrix for BMS. 4 (online mode)

	Detected	Not detected
Intrusion	97.55% (1)	2.45% (2)
No intrusion	1.53% (3)	98.47% (4)

Table 1 illustrates the confusion matrix for UTRC CP model. This confusion matrix can act as the baseline for calculation of various metrics. The confusion matrix shows the accuracy of the solution to intrusion problem. A confusion matrix as presented in Table 1 contains details about the actual and predicted verdicts done by a CP model. Mainly, we have four fields in this table:

- Detected/Intrusion (1) – Percentage of attack windows correctly predicted as attacks,
- Detected/No intrusion (2) – Number of attack points wrongly predicted as attacks,
- Not detected/Intrusion (3) – Percentage of attack windows wrongly predicted as non-attacks,
- Not detected/No intrusion (4) – Number of attack points correctly predicted as non-attacks.

The field (3) refers to security systems that incorrectly see legitimate data points as threat or security breaches. In basic terms, the IDS will detect something it is actually not supposed to. Alternatively, the IDSs are prone to false negatives, where the system fails to detect a request it should have. Both are problematic issues associated with IDS. Following footprints of Table 1, we show denial of service attack for simulation mode in Table 2.

Table 2. Flooding (Denial of service) attack confusion matrix for BMS. 4 (simulation mode).

	Detected	Not detected
Intrusion	84% (1)	16% (2)
No intrusion	9.3% (3)	90.7% (4)

For the case of Denial service attack, CP model predicts verdicts in Table 2. Specifically, the field (1) represents that the percentage of attack windows are correctly detected as intrusion in the network. On the other hand, field (2) represents the number of attack points are not detected with CP model. We plan to improve CP model by large set of training data to avoid such false negatives.

4.3.2 Agent reliability

Agent reliability has been one of the key building features during initial implementation and integration phase of ANASTACIA project. The agent has capability to work in 24/7 fashion thanks to modular design where each connector module responsible for establishing connections with other ANASTCIA components is working independently. This means that if one of the external connections fails the agent will wait until it will be re-established again. Those situations were present initially during integration when project partners were updating their components and this feature enabled agent to work under those circumstances.

Data Analysis Agent has implemented this imperative into following sections of agent:

- **Kafka connector** – by implementing circuit breaker ***[8]*** pattern thus enabling Kafka client to operate event during Kafka broker outage time without any significant impact. The implementation of connector is build based on asynchronous event processing meaning that

each event will trigger function that handles processing for received message. Even if routine fails the error information will be logged via logger component to agent trace file. Next event will trigger again same routine for processing and the message processing will continue.

- **REST API connector** – by using protected REST pooling with GET command. The protection means in this case that REST client connection timeout what configured with proper exception handling in the loop. Event in case Web server error the loop will continue to operate without interruption.
- **Buffer** – by building adjustable configuration explained in previous chapter to handle memory requirements and protecting access to the buffer from multiple threads as well as adding message validation and information extraction. Only correctly formatted messages are passed to buffer for processing. Other messages are dropped and logger for further operator investigation.
- **Model** – anomaly detection is being performed in a separate thread thus is isolated from other agent components. In case of failure the model is restarted, loaded and ready to use. At the same time information about failure will be logged with all details to log file. Another due diligence step is executed when agent is working simulation mode when model is trained. The training process is also used to check that configuration and all sensor information gathered so far is not failing detection process,
- **Logger** – by adding logging to every aspect of agent activity especially exception handling and any incorrect data flows that might be related to incorrect operation from external data sources perspective.

4.3.3 Agent scalability

Scalability of the agent in BMS.4 test case is achieved by implementing two approaches:

- **Single agent performance** – ability to consume events with keeping performance of the system in check. Current implementation is working under this scenario as changes in temperature doesn't require high processing power either from event processing side and anomaly detection component (model). This scenario is designed for small deployments on non-critical infrastructure or for running PoC for new deployments,
- **Multi-agent scenario** – As the amount information that needs to be check grows the multi-agent infrastructure can be deployed to cover large SEP. In this case SEP can be divided into sections. Section can be building floors or larger sections of the building (i.e. company premises – multiple floors of the building) or security perimeter under specific security clearance. In this case large event processing and anomaly detection is spread across multiple agents. This approach off-loads ANASTACIA framework and provides no single point of failure desired in critical infrastructure deployments.

4.3.4 Agent deployment capability

Since agent is written in Python language it can be deployed in various forms on multiple cross-platform applications. The agent has capability to be deployed in:

- **Stand-alone environments** – The machine on which agent operate can be in form of Windows or Linux machine capable of running Python 3.6 ***[9]***. The scenario includes case where on premise machines are required to be used to protect the perimeter and physical access to them is fully controlled. To further sanitize Python environment it is recommended to use virtual environment ***[10]*** recommended by Python community.
- **Cloud environments** – agent can be deployed into either VM machine (Windows or Linux) with Python 3.6 support or on top of Python 3 container within micro-service architecture under Docker ***[11]*** engine. This scenario is targeted into large scale high-availability scenarios where multiple agents are providing verdicts to various sections of security perimeter. The VM or

container based deployment can be used on PaaS infrastructure clouds offerings (Amazon AWS, Microsoft Azure, Google Compute Engine, IBM Bluemix etc.).

4.4 IoT LOCAL AGENT

In this section we will explain how the local IoT agent is integrated with the ANASTACIA infrastructure, though the explanation of the BMS.2 use case where an insider attacker tries to interact with an IoT device.

The insider attacker tries to activate remotely a fire alarm connected to an IoT device. To do that, the attacker sends a CoAP message that contains fire alarm activation query towards the IPv6 address of an IoT device (victim). Figure 14 shows the attack detection using the local IoT agent based on DCapBAC protocol. In particular when attacker sends a CoAP message with an unauthorized actuation to a fire alarm the local IoT broker wired to IoT device using DCapBAC protocol detects unauthorized actuation and notifies ANASTACIA framework.

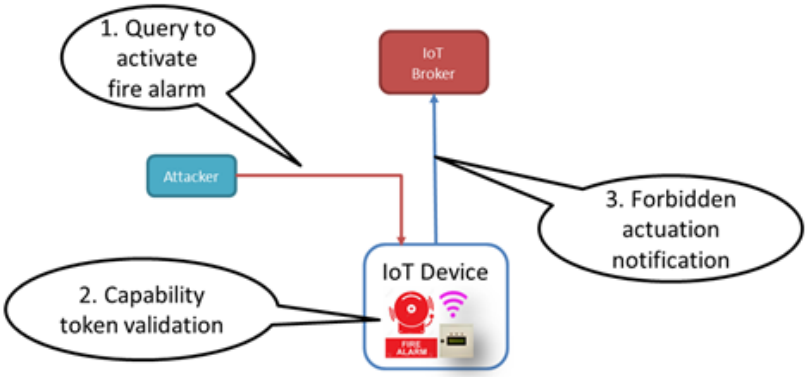

Figure 14: BMS2 use case phases.

The next figures show the data captured from the Insider-Attacker, IoT device and IoT broker to do the three main interactions of the threat notification:

1. Insider attacker sends the fire activation request in a CoAP message including the capability token towards the IPv6 address of a specific IoT device.
2. The IoT-device receives the message and validates the capability token. In this case, the token is invalid due to the lifetime expired and the IoT devices sends a threat notification to the IoT-broker.
3. The IoT broker receives the threat notification and sends a threat message indicating "Unauthorized Device Access" towards the Kafka broker in the Monitoring-Reaction plane of ANASTACIA framework.

Figure 15 shows the script from the client that sends an erroneous Capability Token (marked red), to try to activate the alarm connected to the local IoT agent.

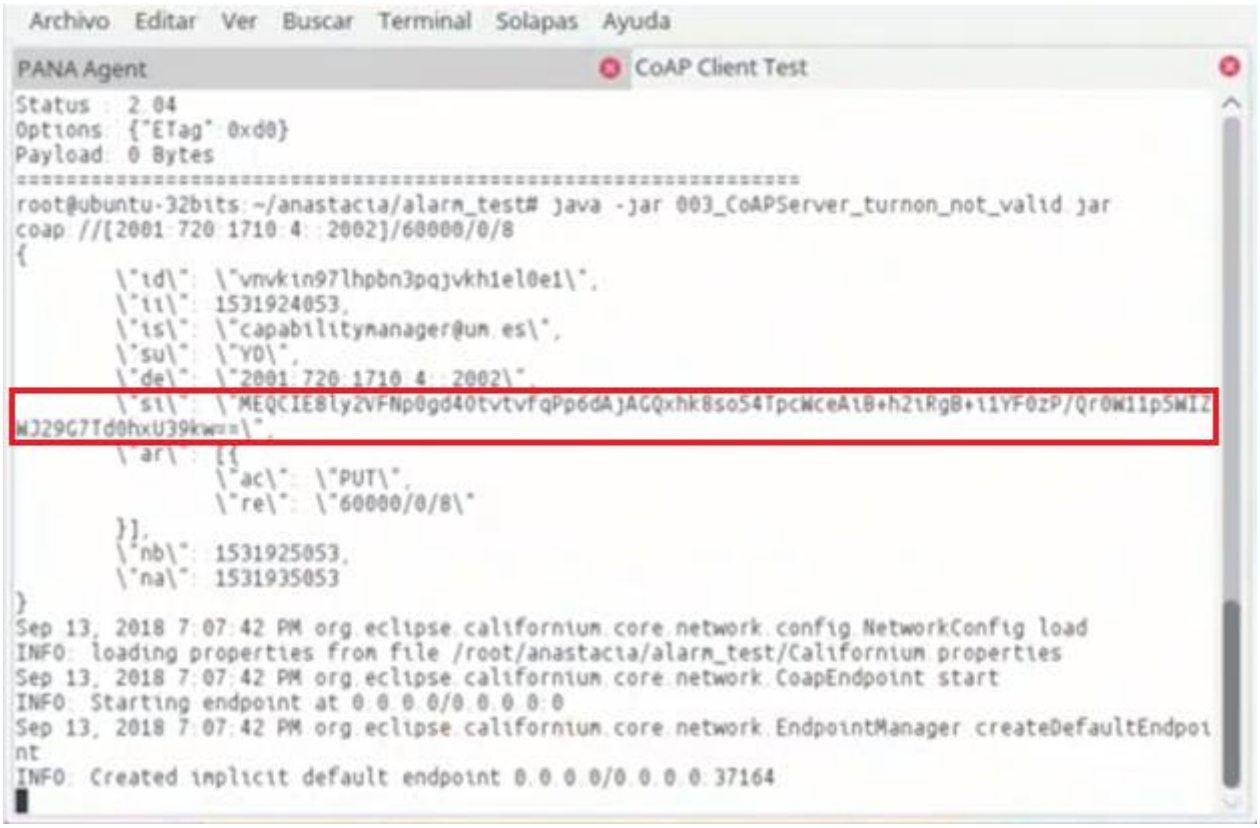

Figure 15: Script sending invalid Capability Token to the local IoT agent

Figure 16 shows the output of the IoT agent as it receives and processes the requests from the client. The IoT agent processes the CoAP request, gets the Capability Token, parses it, and validates the token. As we can see the output says, "*The capability Token is valid: 0*" (Figure 16), stating that the token was not successfully validated, hence sending an error to the IoT broker.

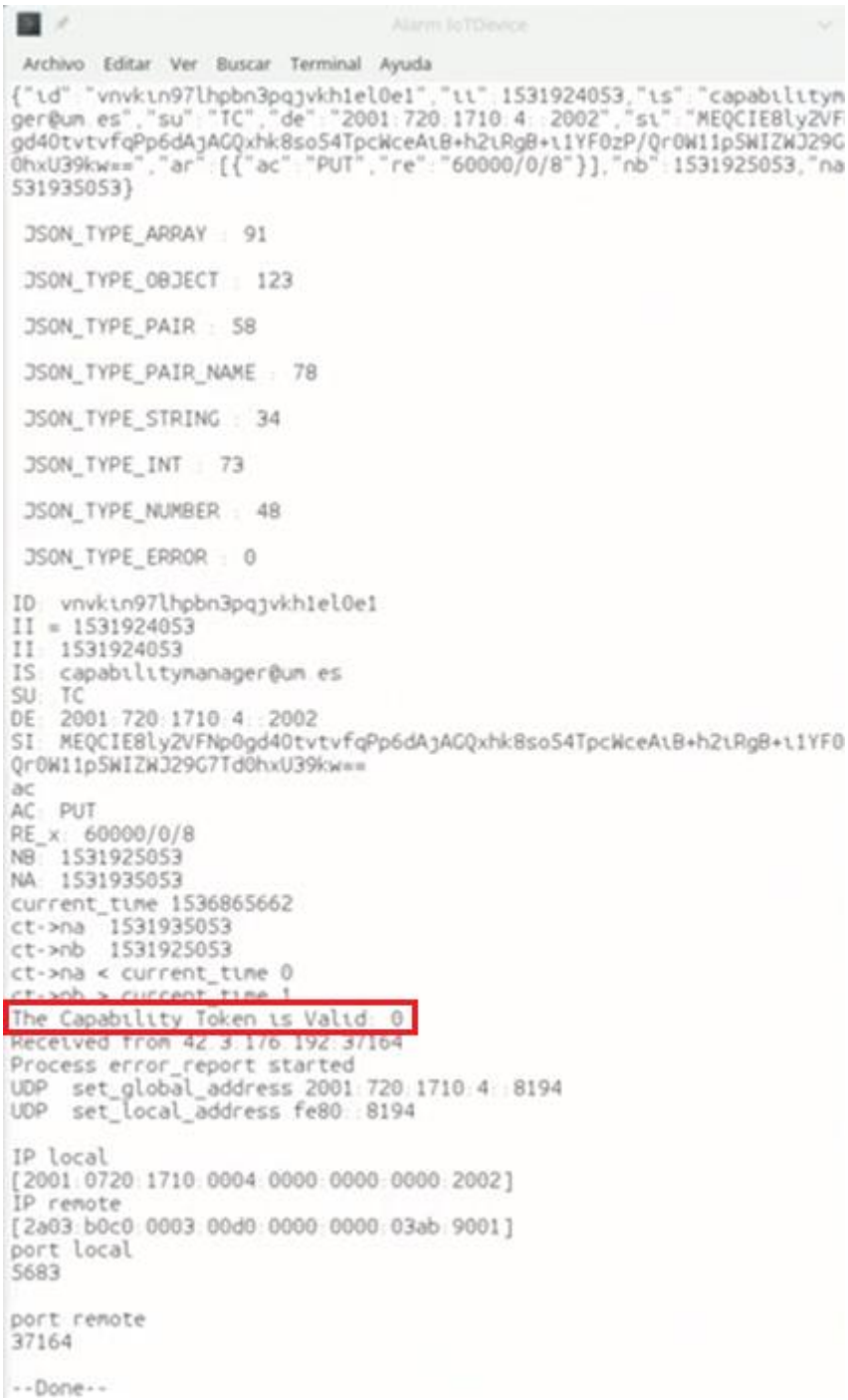

Figure 16: IoT Agent output, validating the Capability Token

Figure 17 shows the output from an entity that is subscribed to the IoT broker. We can see how after the local IoT agent sends the error message to the IoT broker, the subscribed entity receives the notification about the error the IoT agent just sent. After this, the entities in charge of steering or analysing the behaviour of the system after errors are received come into play.

Figure 17: IoT Broker subscribed entity output

5 PLAN AND NEXT STEPS

Further development of agent infrastructure is planned to complete final ANASTACIA agent's implementation. Based on Agile SW development methodologies the plan is divided into multiple sprints that will be executed until end of the project. Each sprint will contain three main parts:

- **Developing features** – described in backlog. This can take form of addressing detection gaps, agent faults observed during initial platform integration process etc.,
- **Testing** – newly deployed capabilities (QA) and running new features in ANASTACIA framework deployment (operations),
- **Retrospection** – on completed development operation (DevOps) cycle and building task list for next iteration (backlog).

At this stage in accordance to the plan there are 14 months to execute final agent development. This gives between 13 and 26 sprint cycles – with one month to two weeks' time span. Figure 18 illustrates example mapping of 13 sprints of agent development into ANASTACIA milestones.

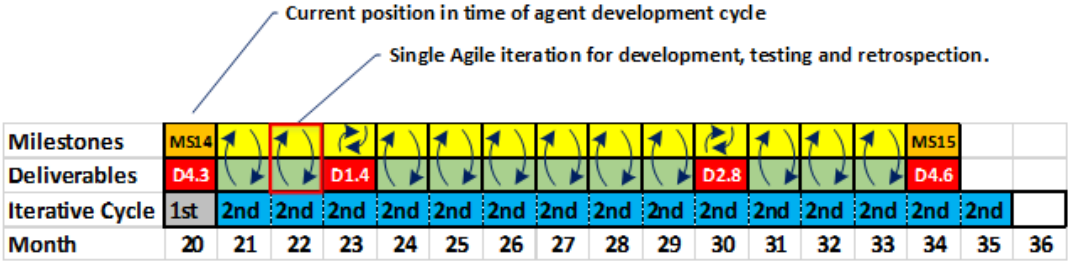

Figure 18. Example of agent development sprint mapping into ANASTACIA milestones.

Looking back at ANASTACIA milestones, in M23 of the project deliverable D1.4 (User Centred Requirements) will be completed and this work will be used to further drive agent requirements. In M30 agent development will be impacted by secure software development guidelines (D2.8 - Secure Software Development Guidelines). The guideline will enable finalization of agent implementation validation from security perspective based on security by design concept. Final agent implementation and integration is planned to be completed in M34 and ready for Y3 review. In subsections below project partners include more detailed plan for reminder of the project that will be aligned with plan from Figure 18.

5.1 ON-DEMAND DEPLOYMENT - NFV/VNF DEVELOPMENTS

Local Agents in ANASTACIA could be delivered and deployed on demand (at the edge of the IoT network) using NFV with 5G slicing. The idea of deploying the monitoring agents as (Virtual network security function) VNFs is not addressed in this document. However, ANASTACIA framework is capable of providing such on-demanded solutions. In this sub-section, we provide a plan to enable such functionality.

Network slicing is one of core building block of 5G networks which allows segmentation of user and control plane traffic within a 5G network. Network slicing can be instantiated in multiple ways with varying degree of cooperation and pervasiveness.

In ANASTACIA framework, we will consider network slicing such that:

- A network slice is a user plane data pipe with varying number of VNFs (Virtual Network Functions).
- Each network slice is defined by a blueprint that defines what the services are and how they are interconnected. Each network slice can be instantiated one or more times. Each instantiation defines a specific configuration for all the services and the network infrastructure within.

For example, an IoT slice could be defined such that it includes a single local agent and a network gateway to internet.

- Any cyber physical device with corresponding local agents can belong simultaneously to one or more network slices. It is left up to the operating system in the device to decide how to forward the application traffic to correct slices.

We use network slicing as a tool to enable on-demand local agents at IoT edge nodes.

5.2 INTEROPERABILITY

ANASTACIA framework external agent interoperability will be enabled by implementing proxy agent that will close gap with existing or new cyber security systems. For each external system or agent that has different interface a proxy data mediator is required to enable cyber security information exchange between bridged systems. Overall interoperability architecture has been illustrated on Figure 19.

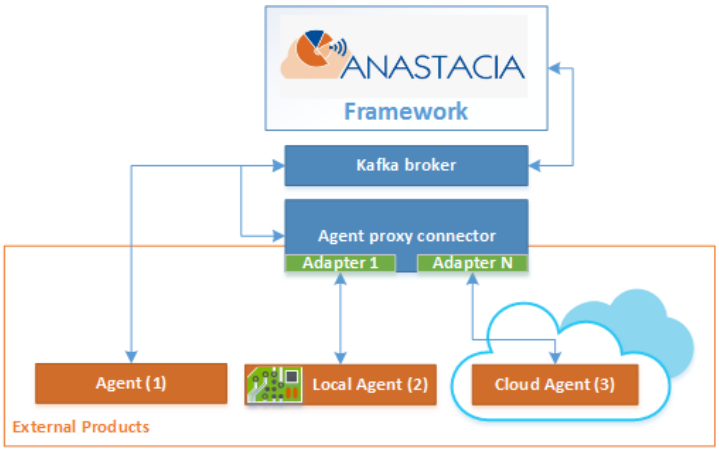

Figure 19. ANASTACIA external agent interoperability architecture.

Each new system will have two paths to become interoperable with ANASTACIA framework. Either adapt its own interface to ANASTACIA framework by sharing information with other components via Kafka broker (case 1) or an adapter will be provided from ANASTACIA framework to adopt to given cyber security product (case 2 and 3). The adapter is simple piece of code that will transfer cyber security information between given product interface and Kafka broker that will relay this information internally to ANASTACIA framework.

As an example of already demonstrated ANASTACIA framework flexibility the consortium already completed integration and interoperability with existing ATOS XL-SIEM ***[13]*** which is commercially available product. Thanks to this effort ATOS XL-SIEM become integral part of reaction module inside ANASTACIA. In the future we envision similar integration process for other commercially available tools to follow the suit. This capability ensures interoperability and adaptability of ANASTACIA framework with external cyber-security systems.

5.3 ZIGBEE AGENT

CNR plans to use the time remaining in the ANASTACIA project to improve the characteristics of the developed agents. In particular, the objective is to improve the following aspects:

- **Investigation of better detection algorithms:** by refinement and improving the techniques adopted to identify a running attack, it is possible to enhance detection efficiency, in order to reduce false positives and correctly identify running threats.
- **Integration of detection capabilities on ANASTACIA:** by providing a communication link between the IoT coordinator and the Detection module of ANASTACIA, it is possible to monitor threats and integrate detection capabilities between the IoT network and the ANASTACIA Detection module.
- **Validation of the developed technologies:** through accurate testing phases, the efficiency and performance of the agents developed for ANASTACIA will be evaluated.

5.4 MMT AGENT

During the second development phase of the ANASTACIA project, Montimage will focus its efforts in the testing and validation of the extended MMT-IoT Agent. In particular, the activities will be focused on:

- Adaptation to 5G scenarios: the future 5G networks will be conceived to operate with IoT devices. In this sense, Montimage will investigate the security requirements on 5G network in order to adapt this solution to this new environment.
- Performance Evaluation of MMT-IoT: The IoT adaptation of MMT is currently in TRL 5. To raise this level and prepare this product for the market, a performance analysis of the solution is required. The main goal of this study will be to determine the maximum throughput of packets that MMT-IoT is capable of extracting from the IoT network. This final step will also allow us to provide a final validation of the technology in real testing environments.
- Validation of the technologies in real environments: To complement the last point, Montimage also plans to validate the technology using the ANASTACIA use cases proposed by the partners. With the validation in real environments, Montimage aims to raise even further the TRL of the novel MMT-IoT result. During the second development phase of the ANASTACIA project, Montimage will focus its efforts in the testing and validation of the extended MMT-IoT Agent. In particular, the activities will be focused on:
- Adaptation to 5G scenarios: the future 5G networks will be conceived to operate with IoT devices. In this sense, Montimage will investigate the security requirements on 5G network in order to adapt this solution to this new environment.
- Performance Evaluation of MMT-IoT: The IoT adaptation of MMT is currently in TRL 5. To raise this level and prepare this product for the market, a performance analysis of the solution is required. The main goal of this study will be to determine the maximum throughput of packets that MMT-IoT is capable of extracting from the IoT network. This final step will also allow us to provide a final validation of the technology in real testing environments.
- Validation of the technologies in real environments: To complement the last point, Montimage also plans to validate the technology using the ANASTACIA use cases proposed by the partners. With the validation in real environments, Montimage aims to raise even further the TRL of the novel MMT-IoT result.

5.5 DATA ANALYSIS AGENT

UTRC plans to use remaining time on three main concepts for agent development that will be aligned to ANASTACIA test cases:

- **Model accuracy improvements** – further development on higher model accuracy, especially by minimizing in false/positive detection and reducing alarms when nominal behaviour of the system is observed (40% effort),
- **Final tuning and integration efforts** – Data analysis agent will go through series of checks and tests to align integration efforts with rest of ANASTACIA framework (30% effort),
- **Validating agent scalability and resilience concepts** – Adding new agent tests related to scalability and resilience to validate agent characteristics in larger deployments scenarios (30% effort).

5.6 IOT LOCAL AGENT

During the second cycle of ANASTACIA development, OdinS plans to enhance the following features of the developed agent.

- **Investigation of cryptographic hardware acceleration** to reduce the execution time of the agent developed for IoT device with constrained resources in terms of computing and memory (30% effort),
- **Integration of cryptographic hardware** in the IoT devices to improve the validation time of the Elliptic Curve signature included in the authorization capability-based token (50% effort),
- **Validation of the integrated hardware** through accurate tests to evaluate the computing performance and the time reduction of cryptographic operations for attack detection (20% effort).

6 SUMMARY

This document illustrated current state of agent development and integration. In chapter 2 agents were described from their capabilities perspective. Local and cloud/remote agents were presented in two subsections. There are two types of agents were developed during initial phase in the framework:

- Local agents – deployed at SEP level to enable close to device malicious activity detections:
  - Zigbee Agent – detection of Zigbee innovative attacks monitoring capability,
  - MMT IoT probe – deep packet inspection and network event correlation on a network level,
  - Local IoT agent – new AAA wireless detection capability for IoT networks,
- Cloud/Remote agents – enable event correlation and security monitoring from holistic point of view:
  - MMT Agent – event correlation from multiple MMT IoT probes,
  - Data Analysis agent – anomaly detection at application layer.

In section 4 agent integration results were presented with focus on main ANASTACIA use cases (MEC.3, BMS.2, BMS.3 and BMS.4). Initial agent implementation inside ANASTACIA framework was successful and helped project to realize following goals:

- **Learn** how new detection mechanisms implemented in agents can work together and complement each other to provide good cyber-security monitoring cover in SEP.
- **Test** case coverage feedback from agent perspective. ANASTACIA provides coverage for WiFi, Zigbee and high-level detection performed on data level by MMT and data analysis agents.
- **Understand** agent integration and implementation requirements for final ANASTACIA framework deployment and develop plan to complete integrations in final year of ANASTACIA.
- **Identify** gaps in performance, scalability, resilience and prepare plans to address any pending challenges. Each project partner after first integration is aware of the gaps and will work on all observed integration issues and close them before final integration phase will be commenced.

Based on above new plan was formed that was presented in previous section. Action plan for further agent development was presented in chapter 5.

Current agent development and integration is on good track and it will be adopted to meet new threats such as IoT zero day attack, advanced persistent attacks etc. and advanced detection capabilities in the final part of this report. The final version of agent implementation will become backbone of ANASTACIA monitoring module.

7 REFERENCES

1. D1.3 – “Architecture Design” – https://seafile.gruppoitaleaf.com/f/309c94f69b/,
2. ANASTACIA Architecture Diagram – https://seafile.gruppoitaleaf.com/f/87d5c06002/,
3. D2.2 – “Attacks and Threats Analysis and Contingency Actions” – https://seafile.gruppoitaleaf.com/f/fa9d752539/,
4. D4.1 – “Initial Monitoring Component Services Implementation Report” – https://seafile.gruppoitaleaf.com/f/a194d52dfe/,
5. D4.2 – “Initial Reaction Component Services Implementation Report” – https://seafile.gruppoitaleaf.com/f/4ef7b936f8/.
6. "Remotely Exploiting AT Command Attacks on Zigbee Networks." – *Security and Communication Networks* 2017 Vaccari, Ivan, Enrico Cambiaso, and Maurizio Aiello.
7. NS-3 simulator – https://www.nsnam.org/
8. Circuit breaker pattern – https://microservices.io/patterns/reliability/circuit-breaker.html
9. Python – https://docs.python.org/3.6/
10. Python virtual environment – https://virtualenv.pypa.io/en/latest/userguide/
11. Docker engine – https://www.docker.com/
12. NS3 modules – https://www.nsnam.org/doxygen/modules.html
13. ATOS XL-SIEM – https://atos.net/en/prescriptive-security
14. Nagios – https://www.nagios.org/
15. Ntop – https://www.ntop.org/
16. Zenoss – https://www.zenoss.com/
17. MRTG – https://oss.oetiker.ch/mrtg/
18. Snort – https://www.snort.org/
19. Suricata – https://suricata-ids.org/
20. Salman Taherizadeh, Andrew C. Jones, Ian Taylor, Zhiming Zhao, Vlado Stankovski, “Monitoring self-adaptive applications within edge computing frameworks: A state-of-the-art review”, Journal of Systems and Software, Volume 136, 2018, Pages 19-38

8 ANNEX

1. Example configuration for Data Analysis agent

```
main:
  name: "ANASTACIA agent"
  vendor: "UTRC"
  version: "1.0"
  description: "UTRC monitoring agent"

# External data handler connectors - they must be initialized first
# without them agent will not work properly
connectors:
  # Cassandra database connectivity
  # Local data storage
  bfo:
    ips: [ "192.168.1.1" ]
    # Site for ANASTACIA machine IP
    keystore: "test"

  # Generic broker connection configuration
  kafka:
    type: "kafka"
    name: "IoT Broker client"
    ips: [ "192.168.1.1" ]
    port: "9092"
    retries: 5
    # Timeout in seconds after which attempt to reconnect to Kafka broker
    # will happen
    reconnect: 300
    # List of active consumer topics that will be used by the monitor
    consumers:
  databee:
    type: "buffer"
    # List of active consumer topics that will be used by the monitor
    consumers:
      IoTBrokerTopic:
        topic: iotqueue
        group: iotqueue
        timeout: 1000

components:
  # ANASTACIA monitor component configuration
  monitor:
    # Component information
    info:
      name: "ANASTACIA UTRC Monitor"
      vendor: "UTRC"
      version: "1.0"
      description: "UTRC monitoring agent"

    clients:
      rest:
        type: "rest"
        ips: ["192.168.1.1"]
        port: 1026
        timeout: 120
        link: "/v1/queryContext"
        headers: {
            'Content-Type': 'application/json',
            'fiware-service': 'ANASTACIA',
            'fiware-servicepath': '/pilot1',
            'Accept': 'application/json',
```

```
        }
        sensors: [
            "10:0:0:0:0:0:0:1",
            "10:0:0:0:0:0:0:2",
            "10:0:0:0:0:0:0:3",
            "10:0:0:0:0:0:0:4",
            "10:0:0:0:0:0:0:5"
        ]
      event_log:
        type: "event_log"
        topic: "UTRC_BMS4_Log"

  # ANASTACIA reaction component configuration
  detect:
    # Component information
    info:
      name: "ANASTACIA UTRC React"
      vendor: "UTRC"
      version: "1.0"
      description: "UTRC reaction component"

    clients:
      # Verdict thread that will evaluate system state every N seconds and
      # will send the information to ATOS XL-SIEM
      vedict:
        type: "verdict"
        # Timeout for verdicts expressed in seconds
        timeout: 60
        # Topic on which verdict messages will be sent
        topic: "verdicts"

# Internal feature engineering(FENG) configuration
feng:
    # Two modes are allowed:
    #  sim - simulation offline mode for training and model validation
    #  online - asynchronous data acquisition through ANASTACIA framework
    mode: "online"
    model: "./data/training_data.csv"
    model_of_sensor: "Test_1"
    outFolder: './data/outputModel/'
    # Determines whether to how to operate model loading:
    # 0 - do not reload model during agent start/stop
          # 1 - reload model
          model_load: 0
          train_data: 100
          model_data: 100
          tuning_data: 0
          min: 0
          max: 1024
          online:
          clusters: 2
          max_sensor_len: 5
          simulation:
            # Attack generator used to emulate adversary behavior in simulation mode
            attack_generator:
              # Attack type 0-N specific attack, -1 will be used for random selection
              file: ""
              type: 0
              number_of_runs: 1
              cp:
                selected_sensors: []
                start: 0
                count: 24
              # List of patterns used in attack generator and their
              # respective configuration settings
              patterns:
```

```
                constant_offset:
                  max_window_size: 24
                  max_no_of_attacks_per_sensor: 5
                random_offset:
                  max_window_size: 24
                  max_no_of_attacks_per_sensor: 5
                flat_line:
                  max_window_size: 25
                  max_no_of_attacks_per_sensor: 5
                no_noise:
                  max_window_size: 25
                  max_no_of_attacks_per_sensor: 3
                noise:
                  max_window_size: 24
                  max_no_of_attacks_per_sensor: 3
                sudden_change_then_noactivity:
                  max_window_size: 20
                  max_no_of_attacks_per_sensor: 5
                shift_pattern:
                  max_window_size: 30
                  max_no_of_attacks_per_sensor: 3

            # Relations between features are described in a N-tuple list fashion
            relations: [ [ "Test_1_temp_diff", "Test_1_temp_avg" ] ]

            # Features configuration
            features:
              # List of features to be extracted from the model
              Test_1:
                type: "temp"
                location: "Ground Floor Visitors Wait Hall"
                temp:
                  raw:
                    function: "raw"
                    length: 1
                    arguments: [
                      "Test_1_temp_raw"
                    ]
                  diff:
                    function: "diff"
                    length: 2
                    arguments: [
                      "Test_1_temp_raw"
                    ]
                  avg:
                    function: "avg"
                    length: 5
                    arguments: [
                      "Test_1_temp_raw"
                    ]
```