ANASTACIA

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

D4.6

Final Agents Development Report

Distribution level	[PU]
Contractual date	31.10.2019 [M34]
Delivery date	31.10.2019 [M34]
WP / Task	WP4 / T4.3
WP Leader	UTRC
Authors	I. Vaccari, E. Cambiaso (CNR), D. Rivera (MONT), R. Marin-Perez (OdinS), P. Sobonski, S.Vuppala, (UTRC), G. Ledakis (UBITECH)
EC Project Officer	Carmen Ifrim carmen.ifrim@ec.europa.eu
Project Coordinator	Softeco Sismat SpA Stefano Bianchi Via De Marini 1, 16149 Genova – Italy +39 0106026368 stefano.bianchi@softeco.it
Project website	www.anastacia-h2020.eu

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

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

Table of contents

PUBLIC SUMMARY 4

1 Introduction 5

1.1 Aims of the document 5

1.2 Applicable and reference documents 5

1.3 Revision History 5

1.4 Acronyms and Definitions 6

2 Beyond State of The Art - Agents 8

3 Agents in ANASTACIA framework 10

3.1 Monitoring agents 10

3.1.1 OdinS firmware/Authentication Agent 10

3.1.2 MMT IoT Agent 12

3.1.3 UTRC Data Analysis agent 14

3.2 Adversary agents 15

3.2.1 UTRC MiTM Agent 15

3.2.2 CNR SlowDoS Agent 16

4 AGENTS results 17

4.1 Monitoring Agents 17

4.1.1 MMT IoT Agent and Probe 17

4.1.2 UTRC Data Analysis Agent 18

4.2 Adversary Agents 23

4.2.1 Hacked firmware Agent 23

4.2.2 UTRC MiTM Agent 23

4.2.3 CNR SlowDoS Agent 24

5 Summary 26

6 References 27

List of figures

Figure 1: Image from the AT88CK590 [13] 10
Figure 2: Main window of the ACES testing software 11
Figure 3: Example of the tools available in the software 11
Figure 4: Example running a SHA256 on a test string in Hexadecimal 11
Figure 5 MMT-IoT solution internal design 13
Figure 6 Design of the remote commands for MMT-IoT 14
Figure 7 Physical and Virtual Deployment of MMT-IoT 18
Figure 8 Monitoring agent detection results 19
Figure 9. UTRC Data Analysis agent deployment architecture. 21
Figure 10. UTRC Data Analysis MSPL configuration workflow. 22
Figure 11. MiTM deployment architecture for Y3 demonstration. 24
Figure 12: The effects of SlowComm against an Apache2 web server equipped with different security modules [19] 25

List of tables

Table 1. Average Confusion matrix over MiTM (online mode) 19

PUBLIC SUMMARY

This document describes final implementation status of ANASTACIA agents deployed on ANASTACIA framework. Work accomplished during the task T4.3 was completed in parallel with tasks T4.1 and T4.2. Moreover, this deliverable extends and leverages initial implementation status which was presented in D4.3 deliverable. In summary, this deliverable elaborates on following topics:

- Key innovation and progress beyond the state of the art from monitoring and adversary agents' perspective.
- Brief description of monitoring agents within ANASTACIA final demonstrator framework and their features with distinction into two types:
    - Monitoring agents, implemented on IoT devices for monitoring security enforcement plane and feeding enablers for threat detection purposes. These agents are implemented as system agents that captures relevant data and perform 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).
    - Adversary agents, designed and implemented to perform certain malicious actions on IoT network to facilitate monitoring agent capabilities with mitigation actions.
- Final integration results of monitoring and adversary agents achieved during ANASTACIA framework with latest updates as of the release time of this deliverable. The section describes monitoring capabilities for each agent.
- Summary of the work completed for final part of ANASTACIA technology demonstrator.

1 INTRODUCTION

1.1 AIMS OF THE DOCUMENT

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

- Beyond state of the art on monitoring and detection agents.
- Final development status for ANASTACIA agents.
- Results of agent integration work.

Chapter 2 explains main innovations and progress of the agents. Chapter 3 elaborates on agent placement including monitoring and adversary agents in overall ANASTACIA architecture. Inside, reader will find information about the monitoring and adversary agents that were deployed in ANASTACIA test bed. Further in the same chapter more detailed information is included about monitoring capabilities of the agents. Also, readers can find more information about adversary actions on typical smart building use-case with respect to final demonstration of the project. Next chapter gathers description of agent capabilities. Finally, the work is summarized with achieved work 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]***.
- D4.3 - Initial Agents Development Report ***[6].***

1.3 REVISION HISTORY

Version	Date	Author	Description
0.1	12.08.2019	UTRC	Initial draft
0.2.1	23.09.2019	OdinS	Integrating partner contributions
0.2.2	07.10.2019	MMT	Integrating partner contributions
0.3	09.10.2019	CNR	Integrating partner contributions
0.4	10. 10. 2019	UTRC, MMT, CNR, OdinS	First version
0.7	25. 10. 2019	UTRC	First Release
0.8	29. 10. 2019	UBITECH, UTRC, MMT	Ready for review
0.9	30. 10. 2019	UTRC	Addressing reviewer's comments

1.0	30. 10. 2019	UTRC	Final document for submission

1.4 Acronyms and Definitions

Acronym	Meaning
CPS	**Cyber-Physical Systems**
SIEM	**Security Information and Event Management**
PANA	**Carrying Authentication for Network Access**
CoAP	**Constrained Application Protocol**
DoS	**Denial of Service attack**
DDoS	**Distributed Denial of service**
DCapBAC	**Distributed Capability Based Access Control**
IDS	**Intrusion Detection System**
IT	**Information Technology**
JSON	**Java Script Object Notation**
MiTM	**Main in The Middle (Attack technique)**
MRTG	**Multi Router Traffic Grapher**
DTLS	**Datagram Transport Layer Security**
OT	**Operational Technology**
PCAP	**Packet Capture**
PoC	**Proof of Concept**
REST	**Representational State Transfer**
SoC	**System on Chip**
SDN	**Software Defined Networking**
SEP	**Security Enforcement Plane**
VDSS	**Verdicts and Decision Support System**

CSO	**Chief Security Officer**
TRL	**Technology Readiness Level**
MSPL	**Medium-level Security Policy Language**
SLIP	**Serial Line IP**
CSO	Chief Security Officer

2 BEYOND STATE OF THE ART - AGENTS

Security is becoming an increasingly paramount task in Cyber-Physical Systems (CPS), as contemporary security techniques use computationally complex cryptography approaches. The seminal works on CPS security have mainly focused on three tracks: creating lightweight security methods ***[7]***, implementing intrusion detection schemes ***[8]–[10]*** and building secure architectures ***[11]***. The CPS intrusion detection systems (IDS) can be divided into two categories: signature and behavior-based techniques. Signature-based intrusion detection approaches seek for runtime features that match a specific pattern of malicious behaviour and they have a low false positive rate. On the other hand, behavior-based intrusion detection approaches look for runtime features that are out of the ordinary. However, the later approaches are more susceptible to false positives. In the cyber-physical world, the attacker's goal is to disrupt the normal operations of the CPS, for example: sensor readings, safety limits violation, status reports, safety compliance violation, communication flows among devices, etc. The continued rise of cyber-attacks together with the evolving skills of the attackers and inefficiency of the traditional security algorithms to defend against advanced and sophisticated attacks such as Distributed Denial of service (DDoS), slow DoS and zero day, necessitate the development of novel defence and resilient detection techniques compared to traditional approaches like signature and behavior-based methods.

Advanced persistent attacks (APT) are combined with MiTM attack to provide maximum survivability of the threat within a new environment. Generally, these attacks are correlated and use low bandwidth communication to avoid suspicion of security systems that rely on statistical detection methods to determine system security. To this end, we can summarize the detection challenges on such advanced attacks as a) finding relation among operational attacks and network attacks, b) learning the nature of attacks and c) APTs that are hard to detect with normal Security Information and Event Management (SIEM) tools. Considering challenge (a), in this deliverable, we propose a "multi-layer" detection model that includes operational, system, and network data to detect advanced correlated attacks. Our developed behavioral data analysis detection engine relates states of different layers so that attacks can be identified with higher confidence level. Now taking challenge (b) into account, our approach adapts relations in order to capture the nature of attacks. Finally, APTs and advanced attacks can be easily detected with our learning approach by incorporating challenges (a) and (b). We leveraged an approach for learning a constraint-programming (CP) ***[12]*** based decision model by learning a set of constraints/relations from the data that conjunctively defines the normal operations and communication flows of a CPS. The malicious operations are detected when CPS fails to abide by the learnt decision model.

In summary, the developed agents 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. Compared to state of the art on monitoring and detection agents, ANASTACIA brings the following additional functionalities to agents. Please note that some of following functionalities has been discussed in D4.3, however, we have updated them during final agents' implementation.

- On-demand

ANASTACIA agents could be delivered and deployed on demand (at the edge of the IoT network) using containerization techniques.

- 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 using containers such as Kubernetes.

- Reliability

Agent reliability has been one of the key building features during the implementation and integration phase of ANASTACIA project. Our agents have 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 presented 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 resilient communication/message passing systems such as Kafka broker system, REST APIs, Buffer, and tracking tools such as Logger. More details on agent's reliability is explained in later part of sections.

- Scalability

  This is achieved by implementing two approaches:

- Single agent performance – ability to consume events with keeping performance of the system in good health.

- Multi-agent scenario – As the amount information that needs to be checked grows, the multi-agent infrastructure can be deployed to cover large Security Enforcement Plane (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 Constraint programming (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 AGENTS IN ANASTACIA FRAMEWORK

Monitoring agents in ANASTACIA framework are designed to detect known and unknown threats. In section below, final state of agent's integration within ANASTACIA framework is illustrated with up to date results discussion. Compared to D4.3 deliverable, we have designed and implemented adversary agents in ANASTACIA framework. These newly added agents play a crucial role in final demonstration to facilitate attack methodology. In this end, we have categorized agents into two parts 1) Monitoring agents presented in section 3.1, and 2) Adversary agents to facilitate the analysis, presented in section 3.2.

3.1 MONITORING AGENTS

3.1.1 OdinS firmware/Authentication Agent

During the second cycle of ANASTACIA development, OdinS has worked on enhancing the following features of the developed agent, investigation cryptographic hardware acceleration aimed to reduce the execution time of the authentication agent developed for IoT devices with constrained resources.

We looked at the AT88CK590 UNIT, evaluation kit from Microchip, as seen in the image in Figure 1. We tested the ATECC508A chip to perform ECC operations.

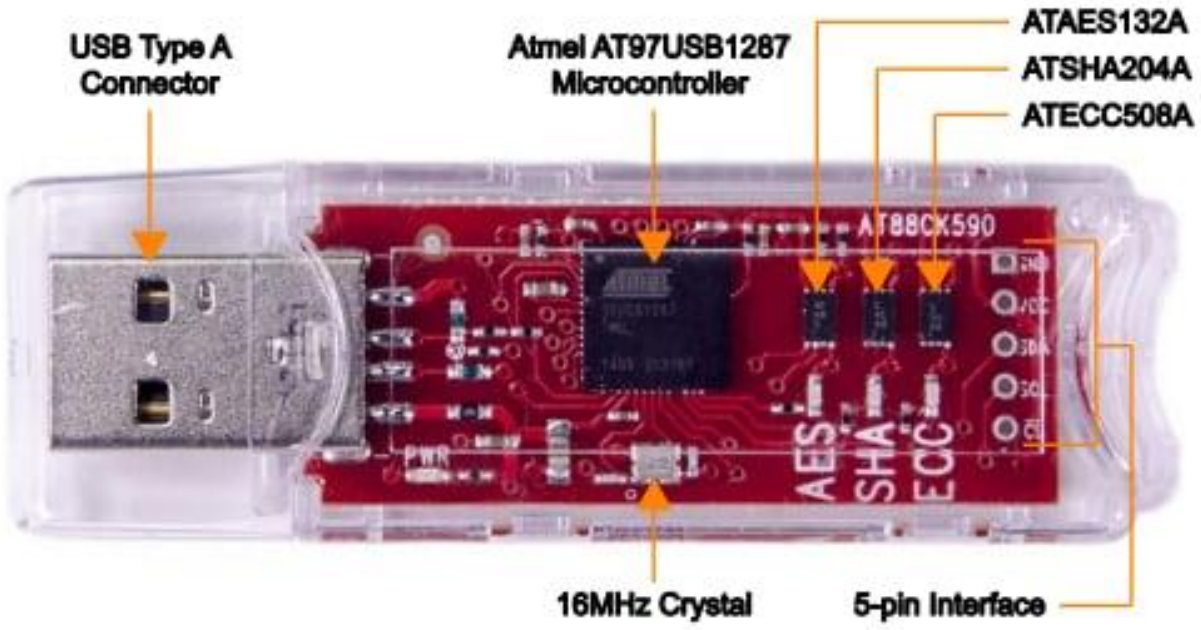


Figure 1: Image from the AT88CK590 [13]

We used the software to test the evaluation unit kit from Microchip, called ACES version 5.0.0 ***[14]*** and perform some testing on the crypto-chip, as we show below.

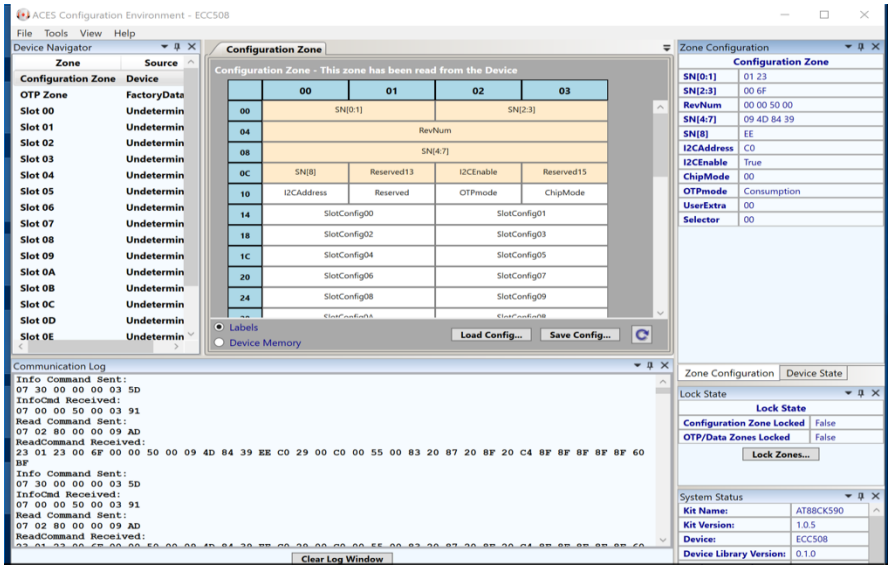


Figure 2: Main window of the ACES testing software

The software allows us to interact directly with the crypto chip, as Figure 3 shows, sending commands on demand, or using predefined tools to perform crypto operations, such as signing and verifying a signature, validating and calculate a MAC. In Figure 4, we perform an HASH operation using SHA256 in the crypto-chip using the string “this is a test” in hex, which translates to “0x746869732069732061207465737 4” and gives the result passing the operation through the crypto suite. The communication is done using the I2C interface that is exchanging the information with the crypto-chip.

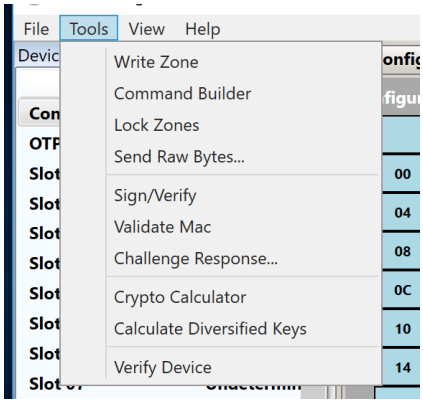


Figure 3: Example of the tools available in the software

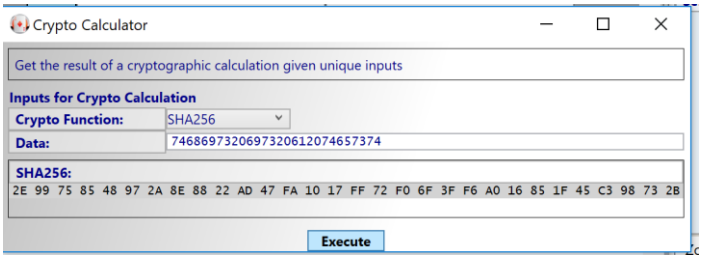


Figure 4: Example running a SHA256 on a test string in Hexadecimal

The software integrated in the IoT authentication agent consists of the following standardized protocols:

- The Protocol for Carrying Authentication for Network Access (PANA) RFC5191. By running PANA, the IoT devices perform what is called a bootstrapping, becoming a trusted party in the security domain of the deployed network, and being able to perform their normal operation in the network.
- The DTLS protocol enables to establish a secure channel, for which the TinyDTLS software is integrated.
- Furthermore, the IoT agents are able to retrieve Distributed Capability Based Access Control (DCapBAC) tokens to access different resources, as well as the OdinS IoT agent is able to enforce the access control to the resources it offers, by means of validating a DCapBAC token when acting as a server.
- CoAP (Constrained Application Protocol) is integrated to exchange information at application layer as a consistent RESTful protocol for constrained IoT devices.

Given the presented protocol stack we are able to secure the communications and access to the network resources as well as resources hosted by different IoT devices in the network.

3.1.2 MMT IoT Agent

During the first half of the ANASTACIA project Montimage has developed the MMT-IoT solution. This novel asset aims to provide an IoT network sniffing solution that can be easily deployed in any IoT deployment in order to perform security analysis inside the IoT network. Despite these advancements, the development reached the Proof of Concept state during the first part of the project, providing a software that run in an IoT environment emulated with Cooja.

During the second period, Montimage continues the development of this asset, in order to further improve the TRL of MMT-IoT by adapting this software to run on real IoT devices, and by validating its behaviour using real IoT traffic.

3.1.2.1 MMT-IoT Adaptations

With the goal of demonstrating the security analysis capabilities of the MMT-IoT solution, Montimage focused the efforts in extending the development to allow the MMT-IoT software to run on a physical IoT device.

3.1.2.1.1 IoT Platform Selection

To this end Montimage has selected the Zolertia ReMote ***[15]*** device as the base IoT device to run MMT-IoT. This device was designed in the frame of the H2020 project RERUM ***[16]*** aiming to offer a flexible platform to develop IoT applications and connect them to already-deployed networks. In particular, these devices provide support for different features:

- Both 2.4GHz and 900Mhz radio support.
- Both IEEE 802.15.4 and ZigBee networks.
- Wide variety of OS supported: Contiki, RIOT and OpenWSN.

Considering these features, the Zoleria ReMote showed a high potential to port the MMT-IoT software. Moreover, the support for ContikiOS reduced the development efforts required, since the Proof of Concept (PoC) version of MMT-IoT was developed using the same OS.

Despite the advantages, the design of the MMT-IoT solution – as presented in D4.3 – depends on low-level system software modifications to correctly work.

3.1.2.1.2 Zolertia Network Drivers Adaptation

As presented in D4.3, the MMT-IoT has been conceived as a two-part software: one running on the IoT device (MMT-IoT Sniffer) and another running on a Linux machine (MMT-IoT Bridge). The solution has been designed considering the low power restrictions of the IoT device. Firstly, the MMT-IoT Sniffer reads the packet from the radio signals and transfers it to the Linux machine using the Serial Line IP (SLIP)[1] library over a USB line. Secondly, the MMT-IoT Bridge reads the information from the USB line and puts it on a loopback network interface. In this sense, the extracted packet can be read by the MMT-Probe software, which performs the security analysis on the extracted packages. Following this approach, MMT agent avoids performing a complex task – the security analysis – on an energy- and resources-restricted device such as the IoT node.

In this design, the MMT-IoT Sniffer is a sniffing solution that is capable of extracting IoT packages in real-time. To this end, MMT-IoT relies on hook functions inserted in the radio drivers, bypassing Contiki' s network stack to reduce the power consumption and maximizing the extraction throughput. The general changes introduced are shown in Figure 5 and they can be summarized as follow:

- Radio driver in *Promiscuous Mode.*
- Retain received packets with bad checksum.
- Insertion of call-backs to directly transfer the network buffer to the USB line.

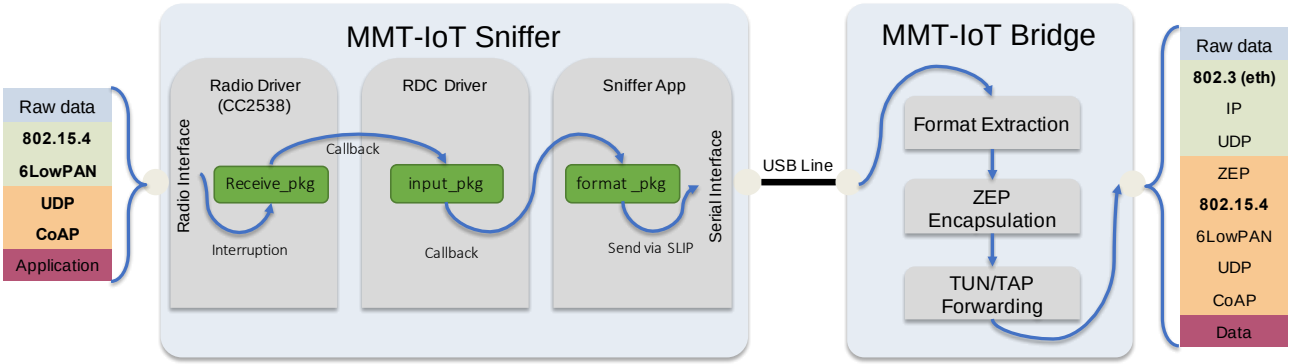


Figure 5 MMT-IoT solution internal design

All the mentioned modifications were implemented in a PoC and tailored for the network drivers of a *Sky mote* – a particular IoT device type emulated by the Cooja software. In the case of the deployment for the Zolertia Remote devices, it was required to insert the same modifications in the CC2538 radio driver (to support the 2.4GHz band), respecting the original design of the emulated solution. The results of this process are two files with the modified drivers of the CC2538 radio chip and a modified version of the Contiki RDC driver.

3.1.2.1.3 Remote Commands Support

In the PoC version, the MMT-IoT Sniffer (running on an IoT device) activated and deactivated its sniffing function by pressing the *user button* on the emulated device. This solution is not optimal when deploying the sniffer on a real environment, where the physical access might not be possible.

To cope with this problem, Montimage modified both the MMT-IoT Bridge and Sniffer software to support basic commands to activate and deactivate the sniffing feature. Figure 6 shows the implemented mechanism in both ends to support this behaviour.

[1] http://contiki.sourceforge.net/docs/2.6/a01693.html#_details

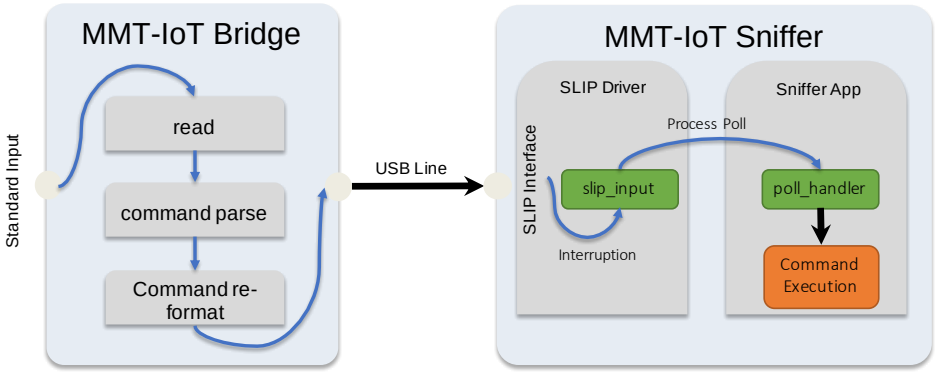


Figure 6 Design of the remote commands for MMT-IoT

Implementing these commands required changes in both the MMT-IoT Bridge and the Sniffer. On the Bridge side, a modification was introduced to read the command from the console, parse it and transform it into a short sequence of character that are sent through the USB line.

On the Sniffer side, a series of changes were made in order to read the characters sent (the command) and implement its behaviour in the device. First, it was required to insert a modified input SLIP driver in order to add a call back once a message is received on the IoT device from the USB line. Secondly, the call back function was implemented to parse the received characters and identify the received command and generate an event in the sniffer functionally equivalent to a button press.

The described solution eliminates the dependency on physical buttons, allowing the sniffer to be activated remotely by simply sending a command to the device. It is worth mentioning that the activation and deactivation of the sniffer using a command is a complimentary method that does not deactivate the manual activation via the system button. In this sense, it is completely possible to activate the sniffer using a command and deactivate it by pressing the hardware button.

3.1.3 UTRC Data Analysis agent

The developed data analysis agent performs system level monitoring and detection by aggregating information (messages) from SEP using Kafka broker. In this deliverable, we will focus on message processing, monitoring and detection using the developed agent. The agent processing part is being divided into two parts, a) monitoring and b) detection.

Monitoring:

We implement message passing mechanism that enables data sources to deliver measurements at remote location that can be used in more accurate anomaly detection and IT system security evaluation. We process each message in two stage routine by implementing a) validation – each message is validated in accordance to predefined data exchanged protocol, b) parsing – this step will divide each message into three main parts: timestamp, identifier of the device, the parsed value of monitored data that extracted from JSON message. Such messages are then added to data buffer that will store all events. More details on such messaging queue processing techniques can be found in D4.3 ***[6]***. Using messaging queue processing, data buffer will construct frameset that will contain current N historical samples of data set for given parameter. We cleanse the data during validation and parsing. Initially, we have considered OdinS IoT lab for sensor measurements. The temperature readings were recorded using messaging queuing techniques. The missing temperature readings can be addressed using linear interpolation. However, this temperature motes doesn't provide information regarding network layers. In order to mimic a CPS, we have created a synthetic data set using the temperature readings and a discrete event simulator. We used NS-3 ***[17]*** 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.

Synthetic dataset*:* We considered the NS-3 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 our proposed model. In order to evaluate the presented approach, we construct a CPS consisting of temperature nodes (i.e., temperature as one of feature of the simulator node) using NS-3 simulator. Each NS-3 node is fed with measured/collected temperature readings from previous step.

Detection:

Generally, most of the anomaly based intrusion systems build a model of the normal behaviour of the monitored system. The built model is then used to compare a currently observed behaviour to the normal behaviour to determine whether the current state is a normal or an anomaly mode. Similarly, we have followed three steps to get the best detection model: a) building an initial model, b) learning the best model, and c) verification of the best model. In the first step, we create an initial model capturing all features and all possible relations using a historical data-set that describes the normal behaviour of the monitored system. Later in the second step, we use another labelled data-set that captures a mixture of normal and anomalous behaviour in order to select the best sub-model in terms of performance. In the third step, we use another unseen data-set to verify the performance of the selected sub-model.

3.2 ADVERSARY AGENTS

In this section of the document, the adversary agents implemented to emulate the attack functionality in order to facilitate use-cases considered in ANASTACIA. It is important to evaluate considered innovative attacks with respect to developed monitoring agents. In summary, monitoring agents are closely aligned with adversary agents to demonstrate their scalability, and attack resiliency. In particular, the man-in-the-middle (MiTM) agent implemented by UTRC is explained. Later, the slow DoS attacker implemented by CNR is described.

3.2.1 UTRC MiTM Agent

UTRC developed a scenario to mimic the MiTM attack. To this purpose, we have a created a malicious script to attack the temperature sensors located in UMU lab. Specifically, our adversary script resides in container environment, which sits in between IoT router and temperature sensor node. The developed adversary script can be triggered from anywhere outside.

To validate and test our models, we have also worked on simulators to mimic CPS system. As explained earlier, we have used NS-3 simulator to generate attack scripts for testing monitoring data.

Attack generation using NS-3: Our experimentation and simulation effort is focused on generating denial of service and man-in middle attacks in the simulated CPS. Flooding attack is a denial of service attack, in which a compromised device floods the network by sending large number of fake control/data messages to existent/non-existent nodes in the network. Synthetic data set creation with NS-3 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 on temperature readings. In turn, it can cause disruption to the operation of temperature nodes, i.e., change in temperature values. It can be performed by either tampering the temperature values or changing the internal state of temperature sensor node. The simulated data has been fed to our proposed learning model for anomaly detection. Whole simulation system with the external data feed from simulator.

3.2.2 CNR SlowDoS Agent

Denial of service attacks (DoS) are executed with the aim of exhaust victim's resources, compromising the targeted systems' availability, thus affecting availability and reliability for legitimate users. The term Slow DoS Attack (SDA), coined by the CNR research group involved in the project, concerns a DoS attack which makes use of low-bandwidth rate to accomplish its purpose ***[18]***. The innovative attack proposed in the context of the project and implemented as an attack agent is called SlowComm ***[19]***, sending a large amount of slow (and endless) requests to the server, saturating the available connections at the application layer on the server inducing it to wait for the (never sent) completion of the requests. SlowComm is able to target multiple TCP-based services like HTTP, HTTPS, FTP, SMTP, or SSH. Such feature is called protocol independence ***[18]***. In particular, SlowComm works by creating a set of predefined connections with the victim host. For each connection, a specific payload message is sent (the payload is typically endless), one character at time (one single character per packet), by making use of the Wait Timeout ***[18]*** to delay the sending. In this way, once the connection is established with the server (at the transport layer), a single character is sent (hence, establishing/seizing the connection at the application layer, hence, with the listening daemon). At this point, the Wait Timeout is triggered, to delay the sending of the remaining payload, and to prevent server-side connection closures. The effects of the attack against an unprotected and vulnerable server are that after a few seconds from the beginning of the malicious activity, the DoS is reached on the server ***[19]***. Such state is kept until the attack is interrupted.

The malicious agent implemented by CNR is represented by a network node designed to target web services through the SlowComm attack. As mentioned, such attack was designed, implemented and validated in the context of the project. SlowComm makes use of minimum attack bandwidth to saturate the resources of the targeted daemon, through the establishment and management of TCP connections, exploiting specific weaknesses of the server. Due to such low requirements for the adversary, in terms of network bandwidth, computational power, disk usage, the adversary can be represented on a non-powerful node. In order to prove that, we represented it as a Raspberry PI node running the malicious software. In particular, although even a virtual machine, a laptop or a desktop node could also be adopted, the adoption of a low-resources host like a Raspberry PI demonstrates how the attack can be executed with ease even on non-powerful hosts. The attacker is equipped by a Raspbian operating system, based on GNU/Linux Debian OS, and it runs the malicious tool, implemented in Java programming language. The target is instead represented by a virtual machine equipped of an Apache2 web server.

4 AGENTS RESULTS

4.1 MONITORING AGENTS

4.1.1 MMT IoT Agent and Probe

Section 3 shows the implementation details of the MMT-IoT solution. This section expands the information by giving details about how the software was deployed and applied on the ANASTACIA general platform.

4.1.1.1 Model detection results

As stated before, the MMT-IoT solution was conceived as a sniffing solution that allows analysing IoT traffic from within the IoT network.

In the ANASTACIA platform, Montimage took advantage of the sniffing and security analysis technologies to detect potential privacy issues cause by data leakages from the IoT network. In this particular case, an attacker can take control of an IoT device (or even insert a compromised device in the network) with the aim of redirecting sensible information being collected by other nodes. Using the MMT-IoT software Montimage will extract the traffic and detect the data leakage by determining if the destination of the data is trusted or not. This is done by a security rule implemented on the MMT-Probe which actively monitors the IoT traffic.

The initial tests conducted by Montimage allowed to detect 100% of the cases where sensible information was sent to an address that does not belong to the trusted network.

4.1.1.2 Agent reliability

Maintaining an already deployed MMT-IoT instance is crucial on real IoT networks. This maintenance is supported by the introduction of remote commands that can be executed from the Linux machine that the MMT-IoT sniffer is connected to.

This mechanism allows simple management of the MMT-IoT Sniffer, supporting the activation and deactivation of the sniffing process, aiming to reduce the power consumption of the sniffing device. In addition to these commands it is also possible to remotely update the software of the sniffing device, allowing to discard any potential changes that an attacker can introduce in the MMT-IoT sniffer. This technique is also used when deploying new instances of a virtual MMT-IoT; since the Sniffer will be running on an emulated environment, the code running will be loaded from a trusted repository rather than cloning an already-running instance, thus reducing the chances of introducing malicious software in the control plane.

4.1.1.3 Agent scalability

As mentioned in the previous sections, MMT-IoT has been originally designed to run on an emulated environment using Cooja. This emulator can be run on a standalone virtual machine that contains the MMT-Probe security software, which complements the MMT-IoT solution. The Virtual Machine can then be deployed dynamically in the same manner as an IoT Honeynet. This allows using the MMT-IoT software in an on-demand basis and deploying it dynamically in a network.

Moreover, the orchestration policies supported by the ANASTACIA platform offer even more flexibility for MMT-IoT, being able not only to deploy new instances of the MMT-IoT sniffer, but also redirect traffic towards this new deployed VM and improve the security analysis with new rules or analysis capacity.

4.1.1.4 Agent deployment capability

MMT-IoT can be deployed in two different ways: physical deployment on the IoT network or virtual deployment of a dynamically allocated VM containing MMT-Probe and MMT-IoT. Both deployments are illustrated in Figure 7.

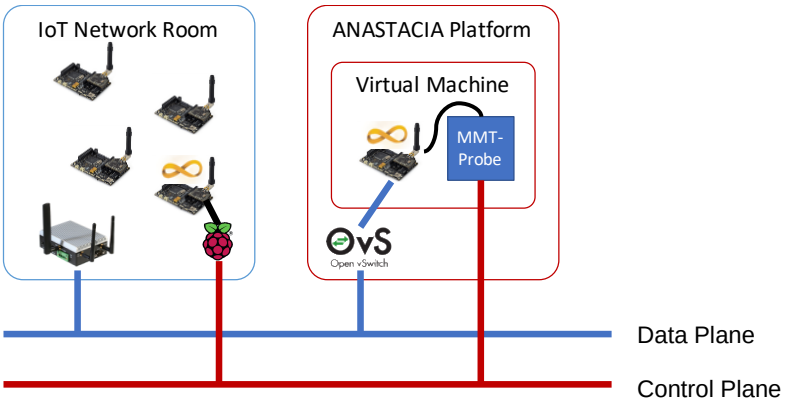


Figure 7 Physical and Virtual Deployment of MMT-IoT

The physical deployment is composed of two pieces: an IoT device (which hosts the MMT-IoT Sniffer software) and a Linux machine (which hosts the MMT-Probe and MMT-IoT Bridge software). Due the physical nature of this agent, its deployment implies gaining physical access to the IoT network, connecting both deployed devices and connecting the Linux machine to the control plane using a wired connection. The embedded software supports all the remote management capabilities mentioned above, allowing to start and stop the sniffing (and the security analysis) as required by the system operator.

On the other hand, a virtual deployment of MMT-IoT allows more flexibility in the sense of deploying instances as needed. These deployments can be triggered manually by the system administrator or automatically by reactive security policies generated by the ANASTACIA platform. As mentioned before, the virtual instances have no differences from the physically deployed agents, supporting also remote management commands to facilitate the access to these agents.

Finally, it is important to remark that both types of deployment can be re-configured by means of security policies. These policies can be generated as reaction to new attacks, in order to instruct the analysis of new security rules, analyse new network interfaces, or even change the output of the potentially generated security alerts.

4.1.2 UTRC Data Analysis Agent

4.1.2.1 Model detection results

In section 3.1.3, UTRC developed the second phase of data analysis agent. This section briefs recent detection results. Once the agent detects anomaly using UTRC CP model and a decision is returned to VDSS system to determine this behaviour as normal or abnormal. The decision (verdict) is sent to “UTRCVerdict” topic on Kafka broker for further analysis. We have performed several attacks including man-in-middle, flooding etc. We consider attack windows and non-attacked points, where an attack window is detected if at any time point the model raised an alarm. We compare the expected verdict and calculate the following confusion matrices for considered attacks. 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. Average Confusion matrix over MiTM (online mode)

	Detected	Not detected
Intrusion	88.89% (1)	11.11% (2)
No intrusion	17.67% (3)	82.33% (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.

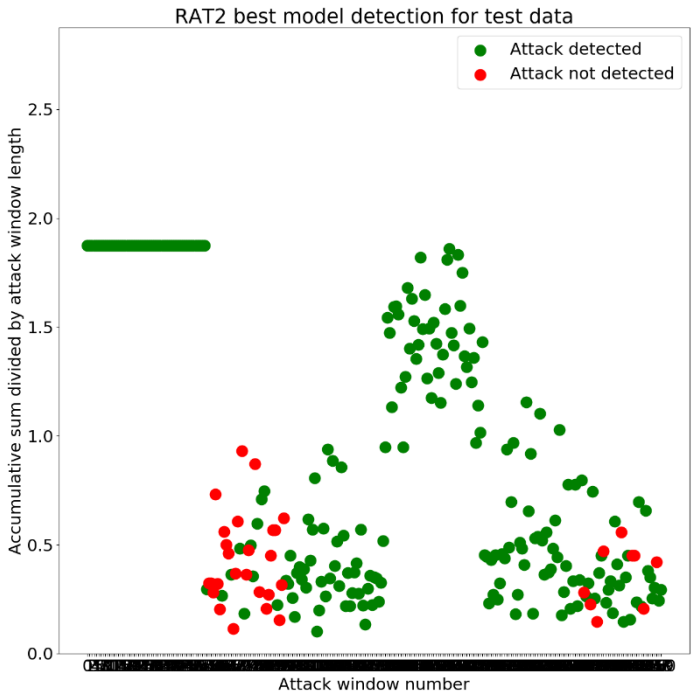


Figure 8 Monitoring agent detection results

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.

Figure 8 demonstrates the detection results for a temperature mote called 'RAT2' under UTRC data analysis agent. It is clear from the figure that our developed technique is capable of detecting anomaly events over time windows.

4.1.2.2 Agent reliability

Agent reliability is one of the primary objectives of agent development. Without correct ability to recover from errors and agent engine might suffer failures that will be more expensive to repair looking further down on the SW lifecycle. To prevent this and enable high reliability during agent code development following best practices were used to deliver high quality SW:

- **SOLID principles (**https://en.wikipedia.org/wiki/SOLID**)** – were used when designing internal agent architecture. The focus was put on modular design, dependency injection and frequent code review and refactoring to adhere to the high SDLC standard.
- **Unit testing** – white and black box tests were designed to test each agent subcomponent that will become part of production agent software.

Outcome of the work was reliable agent that was able to run without any significant failures for Y3 duration of the ANASTACIA project. Internally to reinforce reliability following features were implemented:

- **Tight I/O exception capture isolation** – in case of interaction with external component, file system, network error traps were set to ensure that none of the errors will propagate through SW leading to agent failure. This feature was also expanded in case of communication with Kafka broker.
- **Network reliable loop** – in case of failure of Kafka message broker and inability of consuming messages agent will repeat logging attempts until it will again establish communication. Network connection failures is handled in a way that when new data sensor is not present another direct REST API is used to get data from OdinS IoT broker and still generate verdicts. When data cannot be fetched by either REST API or Kafka queue the agent will stop generating verdicts. Also, for Kafka message broker a dead letter pattern was implemented to keep track record on N latest messages in the buffer in case if message broker will become online. When this happens agent will re-publish all N messages to the queue so VDSS component can process them.
- **Internal processes separation** – monitoring decisions and REST APIs designed to deliver control and status updates are kept as separate processes. Failure of one of them will not crash other. This was done to make sure that agent will be still able to respond despite different circumstances in which it might be working.
- **Health management** – the agent can be inquired about its status. In ANASTACIA to keeping track of each container health CONSUL Service is used. Agent status can also be monitored via second REST APIs to get detailed update on current agent health.
- **Logging** – last line defence is circular log buffer that stores last N MBs of logging information. In case of crash or error the operator can still review logs and send it for further investigation.

4.1.2.3 Agent scalability

UTRC data analysis agent scalability has been addressed in two way fashion. From external point of view, agent has ability to deploy multiple instances in order to cover more sensor network traffic directed from malicious behaviour detection. As mentioned in section 4.1.2.4 agent is wrapped inside Docker container that can be deployed using Docker technology and orchestrated in multiple environments that supports it. This gives agent ability to process IoT sensor network data at large scale.

Internally agent can be configured to scan any number of sensors against which sensor model was prepared. In this sense the agent can process each sensor separately from each other. Thus, it enables flexible configuration to scale up or scale down sensor monitoring.

Based on the new MSPL policy configuration the agent offers on-the fly scalability by turning on or off sensor behaviour investigation that can be sent remotely and will change monitoring capability without need to stop agent. This also in turn enables agent to be used within ever changing cybersecurity monitoring conditions inside ANASTACIA framework.

Internal and external scalability allows agent to be configured in any IoT environment conditions. The recommendation to use external scalability before deployment to assess maximum amount of sensors for monitoring and adjust it to one agent per 100 sensors. Internal scalability can be used after initial deployment to tune in CSO needs for infrastructure monitoring.

4.1.2.4 Agent deployment capability

Since UTRC data analysis agent has been wrapped inside Docker container it can be run in any environment that can host Docker container engine. In ANASTACIA project there were two types of deployment completed. They are following:

- **Geo distributed** – in geographically distributed components the agent has been deployed in UBITECH machine provided for distributed scenario testing.
- **UMU local deployment** – co-located ANSTACIA components on University of Murcia.

In both scenarios same container was used to test agent detection and it monitoring capabilities. Figure 9 illustrates UTRC data analysis agent deployment architecture.

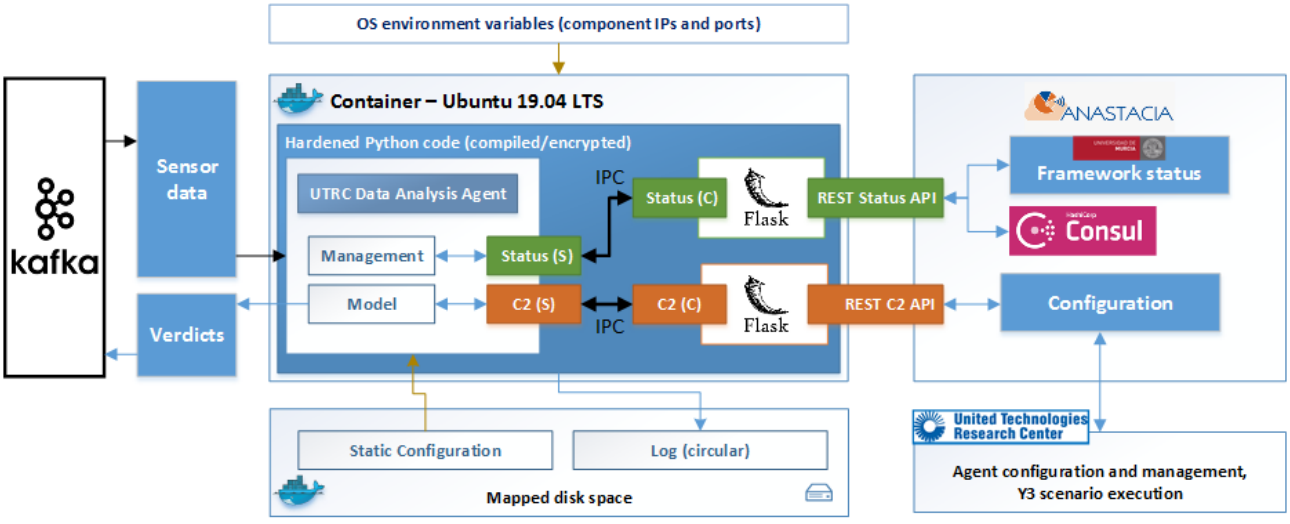


Figure 9. UTRC Data Analysis agent deployment architecture.

The agent was coded using Python 3.7+ programming language. In order to enforce security around the code before deployment the container with the agent has been hardened. There were few techniques used to enforce security:

- **Code compilation** – Python code was complete using pyinstaller (https://pypi.org/project/PyInstaller/) module that offers direct bytecode compilation to either ELF (Linux) or EXE (Windows) format. As part of compilation additional options were used to obfuscate binary code. The obfuscation removed all code comments that could give away better understanding and de-compiled readability as well as obfuscate variable names and classes to make decompiled code difficult to read.
- **Encryption** – further protection of the compiled binary code was achieved through encryption of the code, so only during execution code is open. The encryption use AES-128 bit encryption algorithm.
- **External configuration** – agent container can be configured in two ways. Frist it uses external IP address and port configuration to simplify deployment in different environments (i.e. other ANASTACIA component IP and/or port change). This configuration is achieved by implementing agent ability to read OS environments that points to specific component configuration. The agent is using following IP/port addresses OS variables to achieve this:
    - KAFKA_IP - Kafka broker IP address – address of ANASTACIA message broker on which all sensor data is presented,
    - KAFKA_PORT - Kafka port number – port number on which agent can subscribe to Kafka broker to subscribe to events (both items create a pair that will be used for data ingestion),
    - UTRC_C2_PORT - UTRC Agent configuration port – this is command and configuration port used by other component to reconfigure MSPL policies that are targeted to agent (enable/disable monitoring for specific sensors),
    - UTRC_ECHO_PORT - UTRC Agent echo status port – this port is used to report current status of an agent to Consul Service discovery.

Local IP address of the agent is automatically detected and exposed with ports via Docker engine for other components to use during communication.

Second part of configuration is ability to receive commands and MSPL configuration via predefined Flask REST API designed for this purpose. In this scenario agent is waiting for commands and/or MSPL policies that require attention and performs actions based on requests coming on the management interface. This interface is used to in automatically reconfigure agent on the fly by other ANASTACIA components to generate verdict events for given IoT sensors deployed on SEP. Figure 10 depicts example of MSPL configuration workflow.

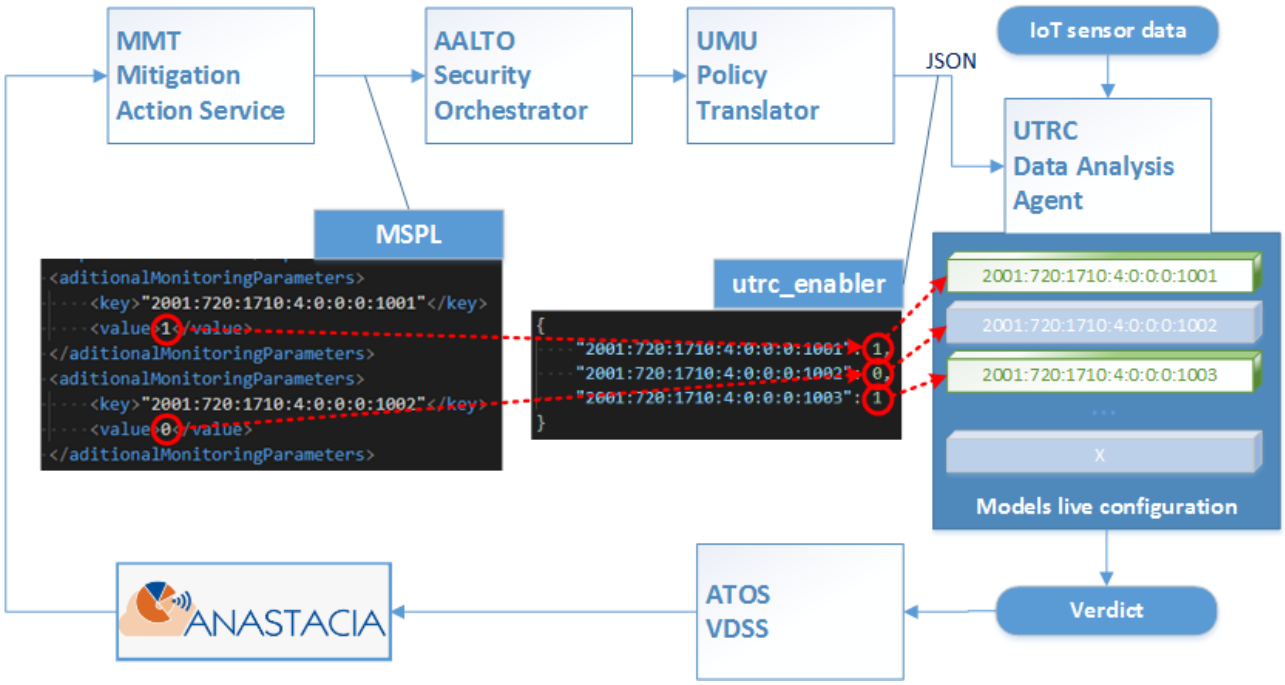


Figure 10. UTRC Data Analysis MSPL configuration workflow.

- **Status management** – agent exposes status port on which via separate REST API that is used later inside ANASTACIA framework by Consul Service discovery to monitor ANASTACIA framework health status and provide feedback to end customer (CSO, DPO),
- **External logging** – as agent performs monitoring actions all important information is logged into externally mounted file. To minimize disk space usage on host machine log configuration is set to be circular buffer of last 5MBs of logging information. The size of the logging can be changed depending on the type of platform on which agent is deployed.

Any attempts to decrypt and decompile container content at rest are severely hampered with first two techniques described above that implements defence in depth principle.

Overall deployment capability of the agent has been brought to current state of the art in SW engineering where use of container based technologies is widely used. This approach enables UTRC data analysis and other UTRC developed components to be deployed in wide range of IT environments starting from closed, disconnected and on premise infrastructure to cloud enabled premises that support containers (i.e. AWS, Azure, Google cloud, IBM cloud).

4.2 Adversary Agents

4.2.1 Hacked firmware Agent

As example of an attacked device, we will propose the use case where an IoT device tries to establish a DTLS connection with an outside server to leak information to an attacker. To do this, an attacker leverages an existing vulnerability in the software that makes it sends the information intended for internal use, to an outside target.

In this sense, an IoT device is infected and starts the mentioned malicious behaviour and the ANASTACIA framework is able to detect the situation and orchestrate a set of actions to mitigate and or palliate the effects of a misbehaving device.

In this scenario, the hacked device includes the TinyDTLS ***[20]*** software that is compatible with Contiki O.S., the operative system used in the developments of OdinS IoT devices. TinyDTLS is a software library that provides a datagram server and client with DTLS support ***[21]***. TinyDTLS has support for PSK and PK authentication with ECC. This gives the flexibility to select the most appropriate exchange according to the capabilities of the network and devices.

4.2.2 UTRC MiTM Agent

MiTM agent has been created to emulate APT threat that stays dormant inside victim network and is activate upon https encrypted traffic from external source to active attack at attack request. Figure 11 illustrates MiTM container architecture prepared for to emulate the considered attack. The layout has been prepared in accordance to same approach used in UTRC Data Analysis agent with different internal functionality to take commands via https server to attack OdinS IoT GW. For purpose of correct component management within UMU deployment monitoring port has been added to enable component mapping in Consul SW.

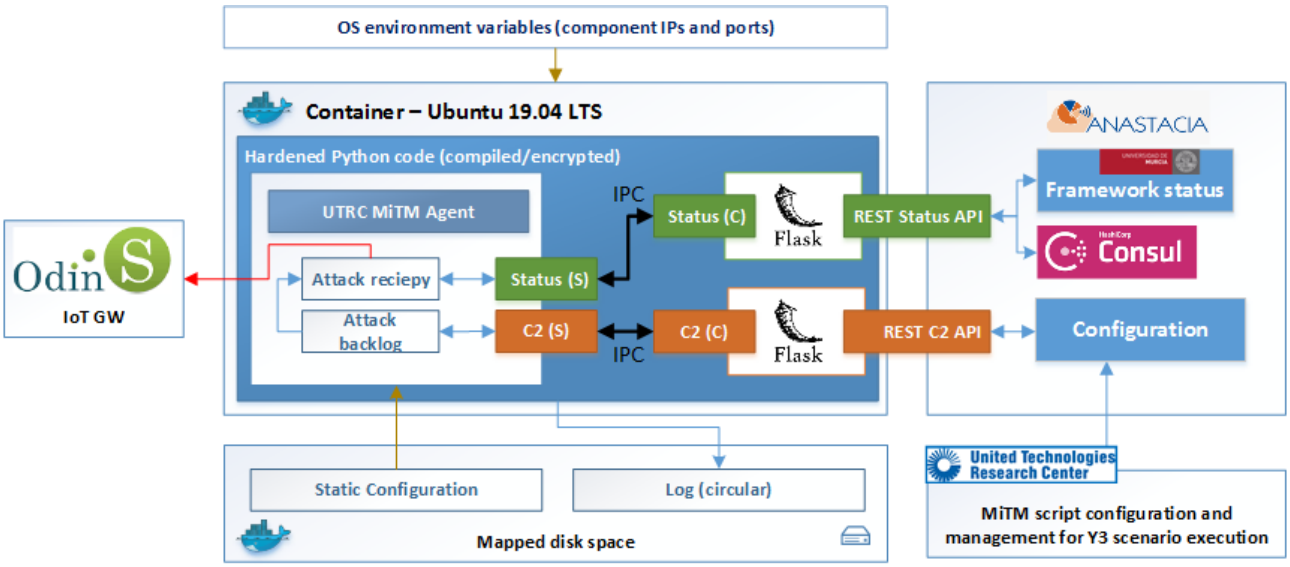


Figure 11. MiTM deployment architecture for Y3 demonstration.

Upon start MiTM agent will record current temperature readings periodically to make sure in case of attack it will be able to clean up the tracks of attack if needed. Agent can be triggered using commands from external perimeter via https connection. From FW perspective this threat is undetectable as it behaves like normal user trying to reach some web service. There are plenty of standard applications that are using this approach to monitor and get updates from the internet.

MiTM configuration has three commands to perform MiTM attack:

- **Start** – Issues commands to take over values over particular sensor set to trigger fire alarm,
- **Stop** – stops/interrupts MiTM attack. When command is issued it will stop sending malicious change temperature commands to OdinS.
- **Restore** – brings back original sensor information to enable multiple testing scenario or evaluate performance of the monitoring components.

4.2.3 CNR SlowDoS Agent

By running the SlowComm attack against a vulnerable and unprotected server, as mentioned, the attack is successful and the DoS is reached just a few seconds after the beginning of the attack ***[19]***. During our tests, by focusing on targeting an Apache2 web server, we also tried to equip the service with security modules available. Results are shown in Figure 12.

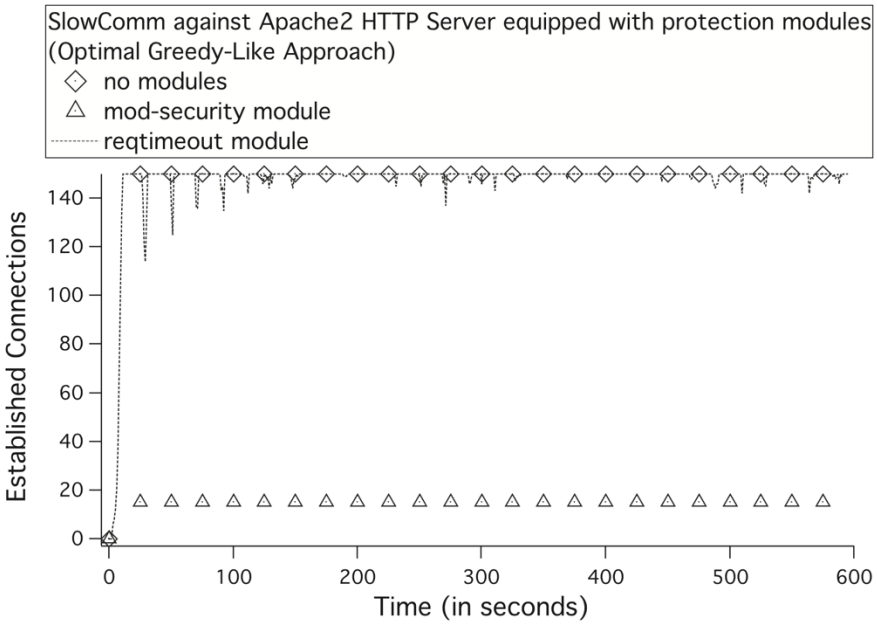


Figure 12: The effects of SlowComm against an Apache2 web server equipped with different security modules [19]

It is possible to notice the statement related to an unprotected server (no modules scenario). Concerning the other modules, mod-security limits the number of simultaneous connections related to the same source IP address: in this case, the attack is not successful. Nevertheless, a coordinated and distributed attack will lead a DoS, since connections would be distributed on multiple nodes, related to separate IP addresses. Instead, regarding the reqtimeout module, it is designed to automatically close connections that do not respect predefined minimum bandwidth rates. Being SlowComm a low-rate attack, connections are closed. Nevertheless, the attack is able to re-establish them as soon as the closure is identified, hence keeping an attack influence near to 1 ***[19]***.

Regarding other protocols targeted by SlowComm, since tested modules only work with Apache2, it is important to consider that similar protection methods may not be available. Because of this, CNR and MONT worked on the design and validation of an anomaly-based intrusion detection system that is able to identify a running attack independently on the targeted protocol.

5 SUMMARY

This document provides overview of final phase of agent development and integration. Chapter 2 provides beyond state of the art and research carried out during second phase of ANASTACIA project. In chapter 3, multiple agent capabilities of ANASTACIA are described. There are two types of agents were developed during final phase in the framework:

- Monitoring agents – deployed to monitor and detect malicious activities:
    - OdinS firmware agent – authentication agent developed for IoT devices with constrained resources,
    - MMT IoT probe – deep packet inspection and network event correlation on a network level,
    - Data Analysis agent – anomaly detection at multi-layer networks,
- Adversary agents – performs malicious actions on victim devices:
    - Hacked firmware Agent – IoT device is infected at firmware level,
    - UTRC MiTM Agent – MiTM attack is performed on IoT device,
    - CNR SlowDoS Agent – SlowDoS attack is launched against web-server

Moreover, ANASTACIA has developed following monitoring agents, however, we haven't presented them in this deliverable as they have been discussed in D4.3 already.

- Zigbee Agent – detection of ZigBee innovative attacks monitoring capability,
- Local IoT agent – new AAA wireless detection capability for IoT networks.

In section 4 agent integration results were presented with focus on main ANASTACIA story board for final demo. Final 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 before final demo.

6 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. D4.3 – "Initial Agents Development Report" – https://seafile.gruppoitaleaf.com/f/9c21d5a184/
7. F. Wu, L. Xu, S. Kumari, X. Li, J. Shen, K.-K. R. Choo, M. Wazids, and A. K. Das, "An efficient authentication and key agreement scheme for multi-gateway wireless sensor networks in iot deployment," Journal of Network and Computer Applications, vol. 89, pp. 72–85, Jul. 2017.
8. V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," ACM computing surveys (CSUR), vol. 41, no. 3, p. 15, 2009.
9. R. Mitchell and I.-R. Chen, "A survey of intrusion detection techniques for cyber-physical systems," ACM Comput. Surv., vol. 46, no. 4, pp. 55:1–55:29, Mar. 2014.
10. S. Haider, D. M. Shila, and M. van Dijk, "Security agents for embedded intrusion detection," Circuit Cellar Magazine, Mar. 2015.
11. C. Stergiou, K. E. Psannis, B.-G. Kim, and B. Gupta, "Secure integration of IoT and cloud computing," Future Generation Computer Systems, vol. 78, pp. 964–975, Jan. 2018.
12. L. D. Raedt, T. Guns, and S. Nijssen, "Constraint programming for itemset mining," in ACM SIGKDD, 2008, p. 204– 212.
13. Image from https://www.mouser.es/new/atmel/atmel-at88ck590-eval-kit/
14. https://www.microchip.com/developmenttools/ProductDetails/atmel%20crypto%20%20studio%20 (aces)
15. https://github.com/Zolertia/Resources/wiki/RE-Mote
16. https://ict-rerum.eu
17. B. Alokparna, S. Vuppala, and P. Choudhury, "A simulation analysis of flooding attack in MANET using NS-3," in International Conference on Wireless Communication, Vehicular Technology, Information Theory and Aerospace & Electronic Systems Technology, 2011, pp.1–5.
18. Cambiaso E, Papaleo G, Chiola G, Aiello M. Slow DoS attacks: definition and categorisation. International Journal of Trust Management in Computing and Communications. 2013 Jan 1; 1(3-4):300-19.
19. Cambiaso E, Papaleo G, Aiello M. Slowcomm: Design, development and performance evaluation of a new slow DoS attack. Journal of Information Security and Applications. 2017 Aug 1; 35:23-31.
20. https://sourceforge.net/projects/tinydtls/
21. Şenol, S., & Arslan, K. S. Secure Trusted IoT Gateway (STIG) - Conference: Information Security Conference 2016, At Ankara, Turkey. https://www.researchgate.net/publication/324057002_Secure_Trusted_IoT_Gateway_STIG