Multi-layered Security Technologies

for hyper-connected smart cities

D4.2: M-Sec IoT Security

March 2021

Grant Agreement No. 814917

Multi-layered Security technologies to ensure hyper-connected smart cities with Blockchain, BigData, Cloud and IoT

Project acronym	M-Sec
Deliverable	D4.2 IoT security
Work Package	WP4
Submission date	31 March 2021
Deliverable lead	Aamir Bokhari (YNU), Arturo Medela (TST)
Authors	Aamir Bokhari (YNU), Arturo Medela (TST), Mathieu Gallissot (CEA)
Internal reviewer	Orfefs Voutyras (ICCS), Kenji Tei (WU)
Dissemination Level	Public
Type of deliverable	DEM

The M-Sec project is jointly funded by the European Union's Horizon 2020 research and innovation programme (contract No 814917) and by the Commissioned Research of National Institute of Information and Communications Technology (NICT), JAPAN (contract No. 19501).

Version history

#	Date	Authors (Organization)	Changes
v0.1	28 January 2021	Arturo Medela (TST)	Full ToC and assignments
v0.2	05 March 2021	Arturo Medela (TST)	Section 1, 2 content provided
v0.3	08 March 2021	Mathieu Gallissot (CEA)	Section 2 updated
v0.4	13 March 2021	Aamir Bokhari (YNU)	Section 2 updated
v0.5	16 March 2021	Arturo Medela (TST)	Section 2 updated
v0.6	18 March 2021	Aamir Bokhari (YNU)	Section 1, 2, 3 updated
v0.7	22 March 2021	Arturo Medela (TST)	Version for internal review
v0.8	23 March 2021	Kenji Tei (WU)	Internal Review
v0.9	26 March 2021	Orfefs Voutyras (ICCS)	Internal Review
v1.0	31 March 2020	Arturo Medela (TST)	Version ready for submission

Table of Contents

List of Tables

List of Figures

Glossary

Acronym	Description	Acronym	Description
AESS	Amazon Elastic-Search Services	OSS	Open Source Software
API	Application Programming Interface	OTP	One-Time Password
ARM	Advanced RISC Machines	PC	Personal Computer
Dx.y	Deliverable y of WP x	Tx.y	Task y of WP x
DoS	Denial of Service	TCG	Trusted Computing Group
IDS	Intrusion Detection System	TEE	Trusted Execution Environment
FG	Functional Group	TCP	Transmission Control Protocol
GPIO	General Purpose Input/Output	TLS	Transport Layer Security
HW	HardWare	TPM	Trusted Platform Module
IDS	Intrusion Detection System	UC	Use Case
IoT	Internet of Things	UDP	User Datagram Protocol
IP	Internet Protocol	USB	Universal Serial Bus
MAC	Media Access Control	VRT	Vulnerability Research Team
OISF	Open Information Security Foundation	WP	Work Package
OS	Operating System	XMPP	Extensible Messaging and Presence Protocol

Executive Summary

The work described in this deliverable (D4.2) was carried out in the framework of WP4 – "Multi-layered Security Technologies", and more specifically, in the framework of T4.1 – "IoT Security". The report presents the updated and final version of the document (the first version being D4.1), providing the technical details of the Functional Group and Functional Components related to the Task.

All technical partners involved in this task collaborated and developed the appropriate tools to meet the objectives set out in the project, especially with regard to novel Security aspects in IoT contexts. Every partner focuses on the individual modules that they are responsible for during the implementation phase of WP4 and supports the integration activities of WP2, while following the common Architecture framework set by WP3 in D3.4.

All of the updated versions of the WP4 technical deliverables (D4.2, D4.4, D4.6, D4.8, D4.10) follow the same approach and have the same structure. Section 1 provides an introduction to the scope of this document and its relation with other WPs and Tasks. Section 2, which aggregates all the main outcomes of the Task, presents extensively the FG and the Functional Components covered by the Task, by providing an extensive description of the corresponding functionalities, and details related to the API of the FG and its interactions with other FGs of the M-Sec solution. Finally, Section 3 concludes the document.

Regarding the differences between 'D4.1 M-Sec IoT Security Layer – first version' and 'D4.2 M-Sec IoT Security Layer – final version':

- **Section 1** has remained more or less the same. Nevertheless, the part referred to the relation to risks and threats grows thanks to the addition of additional risks not originally contemplated.
- **Section** 2 as a whole provides a more integrated view of the Components, as it focuses on their presentation from an FG perspective.
- **Section 2.2** corresponds to Sections 2 and 3 of the previous version of the document (D4.1). The Package Information, Installation Instructions, and the Licensing Information are also integrated in this new version, in order to provide readers with all this information in a single report. In addition, **Section 2.3** introduces the API. A new component was also researched and tested that helps in further addressing unknown attacks and zero-day threats.
- **Section 2.4** is a brand new section and is a result of the common integration activities between all of the technical Tasks of the project.
- **Section** 3 corresponds to Section 4 of the previous version of the document.

All in all, the deliverable is considered to have provided all of the information required to expose the M-Sec technical solutions related to T4.1 as well as the results of the integration and demonstration related activities.

1. Introduction

1.1 Scope of the document

The main focus of this task is to implement the M-Sec IoT security framework, which in turn will help to develop reliable and secure applications for the smart city context. The goal here consists in looking for techniques, methods, and design and operating principles that minimize the risk of suffering critical vulnerabilities in a wide range of IoT devices, which could be leveraged by hackers to carry out a number of nefarious activities.

This task has as its main objective the definition and ulterior implementation of the M-Sec IoT security layer and thus and thus starts with the devices and services it comprises, which have evolved from its initial description in Deliverable 4.1 [D41], in parallel to the execution of the Stage 1 of the different pilots. Retorting to the original M- original M-Sec architecture already introduced and discussed in previous reports such as Deliverable 3.3 [D3.3], the [D3.3], the layer addressed by this task can be identified as the IoT layer. What is more, given the final version of the version of the architecture presented in Deliverable 3.4 [D34], the components about to be discussed in this report report are part of the so-called Devices Functional Group (FG) and Devices Security Functional Group (see

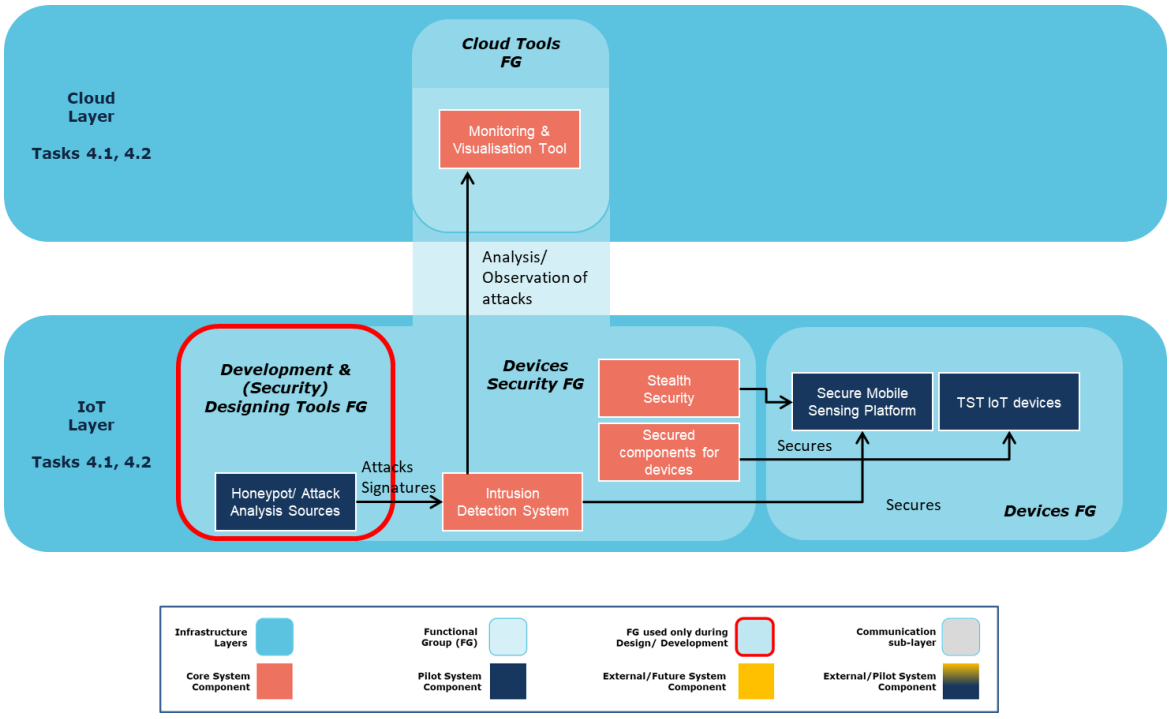

Figure 2).

This document addresses the main objectives of this task establishing the M-Sec components strengthening the IoT layer, which are one of the security layers in the overall Multi-layer Security (M-Sec) platform, providing the needed security and reliability for IoT devices as follows:

- **IoT devices** with increased security, an asset that further strengthens the current state-of-the-art security provision in IoT devices on a hardware level.

- **Perimeter Defense** (originally known as Intrusion Detection System (IDS)), a software-based asset that monitors communication between IoT devices and cloud in order to detect, prevent, and report any suspicious activity that may be a sign of an attack.
- **Stealth Security** feature is a newly added software component that further adds another layer of security by making the IoT device invisible to the Internet and responds only to authorized devices. It also provides power-savings to the resource-constraint IoT device.
- **Security Monitoring and Visualization Tool** provides 24/7 insights into the IoT layer by collecting logs and providing easy-to-understand graphical analytics.

This set of assets will act as the foundation coming out of Task 4.1 and feed the various project pilots, where they are put to a test in real-life situations, according to the respective use cases.

1.2 Relation to other work packages and tasks

Figure 1 summarises the relations of this deliverable (and the corresponding task) to other tasks and WPs.

More specifically, Task 4.1 relates to WP2 since the IoT devices there described as part of the diverse use cases are in need of security features to provide users with a safe and reliable service. This is where WP4 comes into action and delivers the techniques to bring an answer to those needs. There is also a direct relation to WP3. T4.1 receives as input, the system and user requirements from T3.1 and Risks-and-Threats-related information from T3.3. Moreover, it follows the common Architectural framework that has been identified in T3.2 for the coordination of all the technical activities.

Within this very same WP, T4.1 is closely related to the other Tasks that focus on other security layers. Communication between the WP4 tasks leads to an end-to-end security solution. For example, T4.1 is directly connected to T4.2, where the cloud/data security layer is discussed, given that all information produced by the IoT devices will travel through it, and also to T4.4, dealing with the application-level security, since data provided by IoT devices will be employed by the apps offered to users. When referring to M-Sec Functional Groups (see Figure 16), the Devices Security FG directly relates to the Secure & Trusted Storage FG, where data forwarded from the IoT devices and tools on the lower layer arrives and is properly and safely stored. In addition, there is a clear link to the IoT Data Marketplace, since these very same data will be classified and made available for potential stakeholders who may be interested in taking part in the M-Sec ecosystem and develop their own solutions.

Finally, the results of this report are directly provided as input to T2.3 which is focusing on the overall integration activities. Together with the other final deliverables of WP4, D4.2 provides all the information and functionalities required for an integrated security solution.

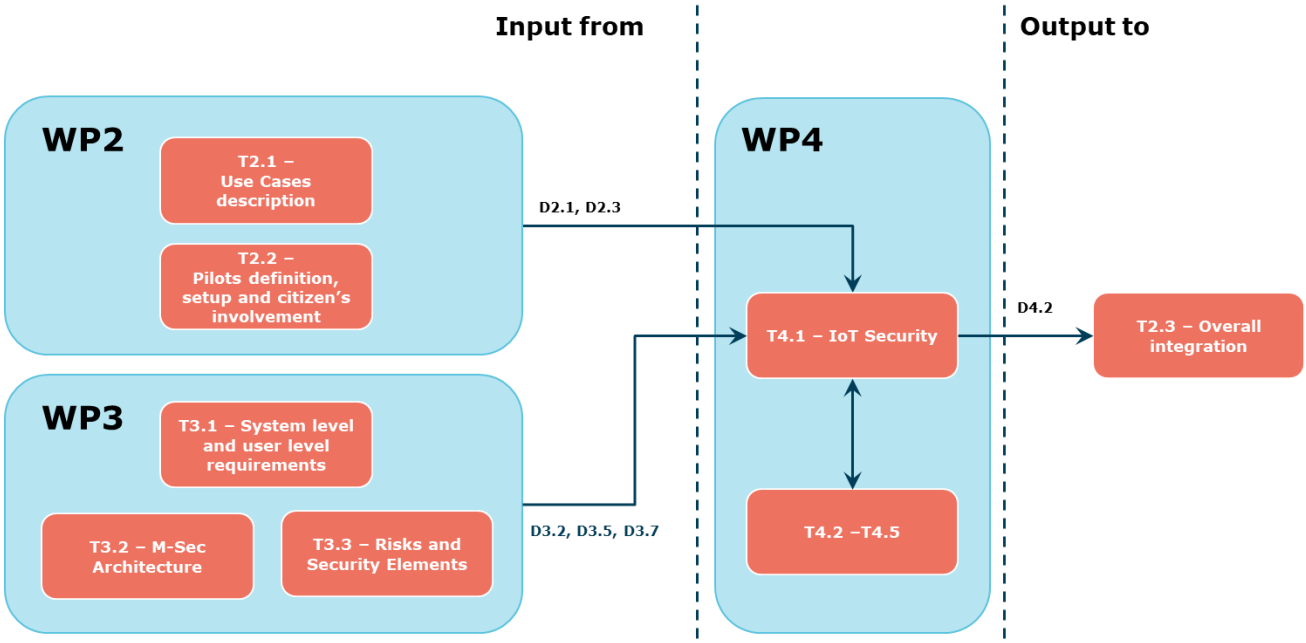

Figure 1. T4.1 and D4.2 relation to other WPs and Tasks

1.3 Relation to M-Sec Risks

The proliferation of IoT devices, not only in the workplace but also in everyday routines and environments, presents a huge security risk. Threats to IoT systems and devices translate to bigger security risks because of certain characteristics that the underlying technology possesses. These characteristics make IoT environments functional and efficient, but they are likely to be abused by threat actors.

There are different IoT attack surface areas, or areas in IoT systems and applications where threats and vulnerabilities may exist. Below is a summarization of the IoT attack surface areas, which for M-Sec's particular case have been addressed in WP3 as part of Task 3.3:

- **Devices.** Devices can be the primary means by which attacks are initiated. Parts of a device where vulnerabilities can come from are its memory, firmware, physical interface, web interface, and network services. Attackers can also take advantage of unsecure default settings, outdated components, and unsecure update mechanisms, among others.
- **Communication channels.** Attacks can originate from the channels that connect IoT components with one another. Protocols used in IoT systems can have security issues that can affect the entire systems. IoT systems are also susceptible to known network attacks such as denial of service (DoS) and spoofing.
- **Applications and software.** Vulnerabilities in web applications and related software for IoT devices can lead to compromised systems. Web applications can, for example, be exploited to steal user credentials or push malicious firmware updates.

The complete list of potential risks and threats that may affect M-Sec's IoT layer can be checked in Table 1, as extracted from Task 3.3. However, it is worth noting that this list includes an extra entry which registers the potential trouble an unknown attack may cause over the IoT devices that conform this layer. All of these threats are of Type *"IoT/Edge"*, and Sub-Type *"Device"* or *"Management"*. Specific interfaces are provided in D3.5 [D35].

Table 1: M-Sec IoT layer risks and threats

Threat #	Description	STRIDE Threat Class	M-Sec Asset	Source	Probability	Criticality	Rating	Comments/ Mitigation
Thr.IoT.1	Data stored in the device can be read by an intruder	I	EnMon, Crow, Caburn	UC1,2	3	3	9	TPM will be designed to reduce the probability by securing the IoT device itself.
Thr.IoT.2	An unauthorized party can modify data on the device	T	EnMon, Crow, Caburn	UC1,2	3	3	9	TPM will encrypt data to reduce this risk.
Thr.IoT.3	Man in the middle attack: a third party puts itself between the entity that communicates with the device and the device itself, without them noticing	S	EnMon, Crow, Caburn, KEIO Mobile Sensing Platform	UC1,2,3	3	3	9	Only PCB with hard coded sensors. TPM will encrypt data to reduce this risk. Serial connection is to be clamped and Locked. Physical security to mitigate risk by lowering likelihood.
Thr.IoT.4	Unauthorized modification of configuration parameters of the device or the sensor	E, T, D	EnMon, Crow, Caburn, KEIO Mobile Sensing Platform	UC1,2,3	3	3	9	Hard coded circuitry. Someone need to steal and replace with modified PCB. Few affected IoT devices/sensor box not critical. Security cameras in the park/surroundings lower likelihood.
Thr.IoT.5	An attacker can overload the device by injecting many requests	D	EnMon, Crow, KEIO Mobile Sensing Platform, Caburn	UC1,2,3	-	-	-	No services active, so no requests can be processed
Thr.IoT.6	Jamming of the wireless communication link	D	EnMon, Crow, KEIO Mobile Sensing Platform, Caburn	UC1,2,3	-	-	-	No wireless interface
Thr.IoT.7	Accidental or intentional physical damage to any device	D	EnMon, Crow, Caburn, KEIO	UC1,2,3	3	3	9	Risk will be mitigated by physically put devices in secure enough locations. Risk

	part may cause device failure		Mobile Sensing Platform					will be mitigated by Physically clamping and locking device securely. Few sensor failure is not critical
Thr.IoT.8	Insecure firmware update mechanism: the firmware has been retrieved at a non-valid source	S	EnMon, Crow, KEIO Mobile Sensing Platform	UC1,3	-	-	-	No firmware
Thr.IoT.9	If installed, malware has full access to the whole device	T	EnMon, Crow, KEIO Mobile Sensing Platform	UC1,3	-	-	-	No OS
Thr.IoT.10	If installed, malware has access to data in the device	I, T	EnMon, Crow, KEIO Mobile Sensing Platform	UC1,3	-	-	-	No OS
Thr.IoT.11	If installed, the malicious firmware may cause device operation failure	D	EnMon, Crow, KEIO Mobile Sensing Platform	UC1,3	-	-	-	No OS
Thr.IoT.12	Insecure firmware update mechanism: the firmware is corrupted	D	EnMon, Crow, KEIO Mobile Sensing Platform	UC1,3	-	-	-	No firmware
Thr.IoT.13	A device designed to be used by several users and keeping history per user discloses information on the other users	I, E	EnMon, Crow, KEIO Mobile Sensing Platform	UC1,3	-	-	-	No such capability
Thr.IoT.14	A weak authentication method is very likely to be used (short and simple passwords, if any), opening a door to data and	S, Implementation issue	EnMon, Crow, KEIO Mobile Sensing Platform,	UC1,2,3	-	-	-	No OS, no firmware

	device exposure.		Caburn					
Thr.IoT.15	The device was reset to its default settings, which does not include security.	E	EnMon, Crow, KEIO Mobile Sensing Platform, Caburn	UC1,2,3	-	-	-	No OS, no firmware
Thr.IoT.16	Nobody is responsible for device maintenance.	Management issue	EnMon, Crow, KEIO Mobile Sensing Platform, Caburn IoT Devices	UC1,2,3	1	3	3	M-Sec partners will play this role and assign a responsible person. Partners have already assigned people responsible for the maintenance to lower the likelihood and impact.
Thr.IoT.17	Nobody is responsible for system management and maintenance (e.g. system: device network)	Management issue	EnMon, Crow, KEIO Mobile Sensing Platform, Caburn IoT Devices	UC1,2,3	1	3	3	M-Sec partners will play this role and assign a responsible person. Partners have already assigned people responsible for the maintenance to lower the likelihood and impact.
Thr.IoT.18	Attack on Power Management.	D	Crow, KEIO Mobile Sensing Platform, Caburn IoT Devices	UC1,2,3	3	3	9	No Firmware or data storage. Physical security & clamping to mitigate risk by lowering likelihood.
Thr.IoT.19	A visitor is playing with the device (e.g. a blood pressure monitor) and wrongly records measurements that are not those of the intended user.	S: Identification rather than authentication	Caburn IoT Devices	UC2	3	5	15	Not blood pressure monitor, but other monitors can have wrong measurements if someone touch it, even without knowing it.
Thr.IoT.20	Old persons may not know how to handle electronic	Usability	Caburn IoT Devices	UC2	3	5	15	Users' concern about the difficulty in the use technological devices. Devices that

	devices efficiently. They make errors and quit easily.							do not require human interaction, such as door/window opening sensor, smart plug sensor, were selected.
Thr.IoT.21	An old person denies having recorded a measurement (e.g. blood pressure rate).	R	Caburn IoT Devices	UC2	3	3	9	Deployed devices do not require human interaction.
Thr.IoT.22	A member of caring personnel denies having administered a treatment.	R	N/A	N/A	--	--	--	Out of the scope of Use Case 2
Thr.IoT.23	Risks from unknown attacks.	S, T, R, I, D, E	EnMon, Crow, KEIO Mobile Sensing Platform, Caburn	UC1,2,3	5	5	25	Mitigation will be a NICT based process and through the Stealth Security feature.

2. Devices Security FG

2.1 General Description of the FG

Personal devices like smartphones, tablets, and personal computers (PCs) are getting more and more secure, but hackers are getting better at attacking them too. Users on the know are aware of these dangers and would like to make sure they are protected against the latest threats. M-Sec provides easy-to-use guidance and solutions aimed to secure these devices, starting and making an example of the ones developed and tested during the course of the project, along with the applications that run on or through them.

Therefore, this report focuses on the security incorporated into the IoT devices themselves, being Task 4.4, the one dealing with the techniques to proceed within the application side of the equation in order to increase the overall security in the whole service. The Functional Groups related to this Task and the corresponding components are shown in the following Figure 2.

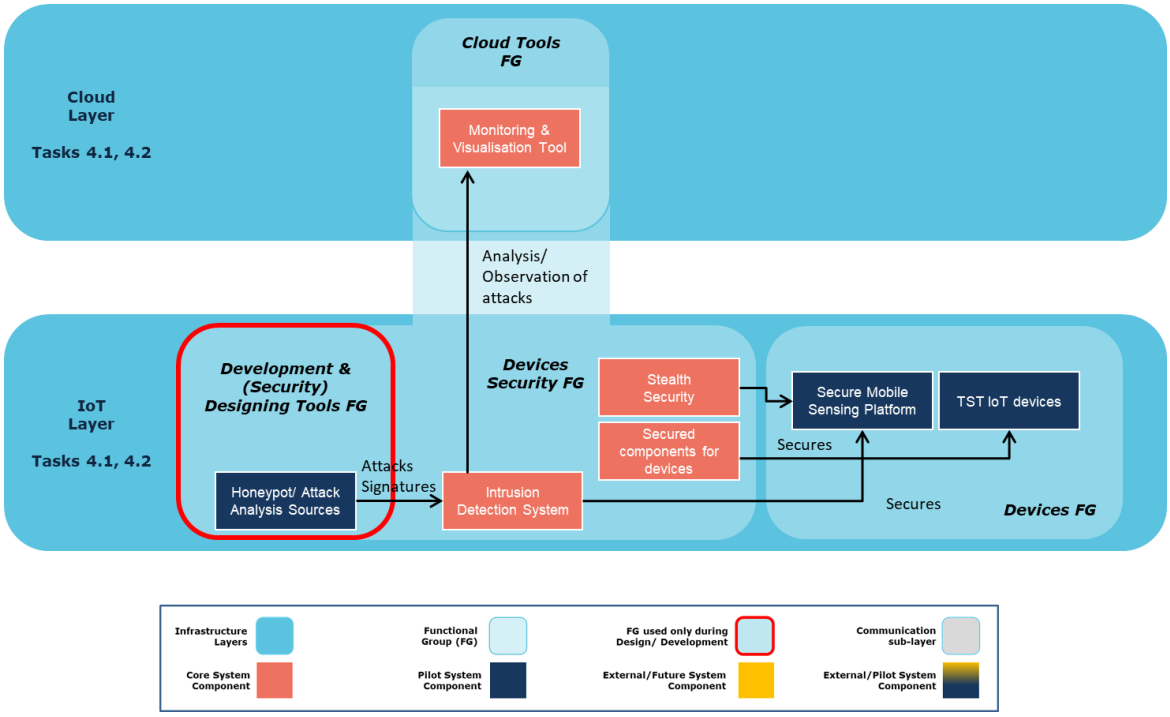

Figure 2. The M-Sec Devices and Devices Security FGs

"Devices" in the M-Sec context are physical artifacts with which the physical and virtual worlds interact, as well as software-based services that operate at the lowest levels of the architecture (Devices FG). Devices can also be the entities for certain types of applications, such as management applications, when the interesting entities of a system are the devices themselves and not the surrounding environment. For the Devices Seucrity FG that secures in M-Sec, the following key security components are described:

- Secured components for devices
- Perimeter defense (Intrusion Detection System)

- Stealth security
- Security monitoring and visualisation tool

In the following section, these four components are described in detail. Section 2.4 presented the interactions of these components with components of other M-Sec FGs. Finally, the Annex presents the position of the Devices Security FG within the whole M-Sec Architecture.

2.2 Components of the FG

Reliable and secure IoT devices

The EnMon & Crow IoT devices represent the hardware (HW) solution designed and developed to give an answer to the requirements posed by Use Case 1. The initial design ambitioned to reach the security objectives through the integration of a TPM2 device in the main boards: one of them, the former, designed in-house and relying on an STM32-L4 microprocessor, and the latter retorting to the employment of a Raspberry Pi properly programmed and wired. Figure 3 shows the rough appearance of these two solutions.

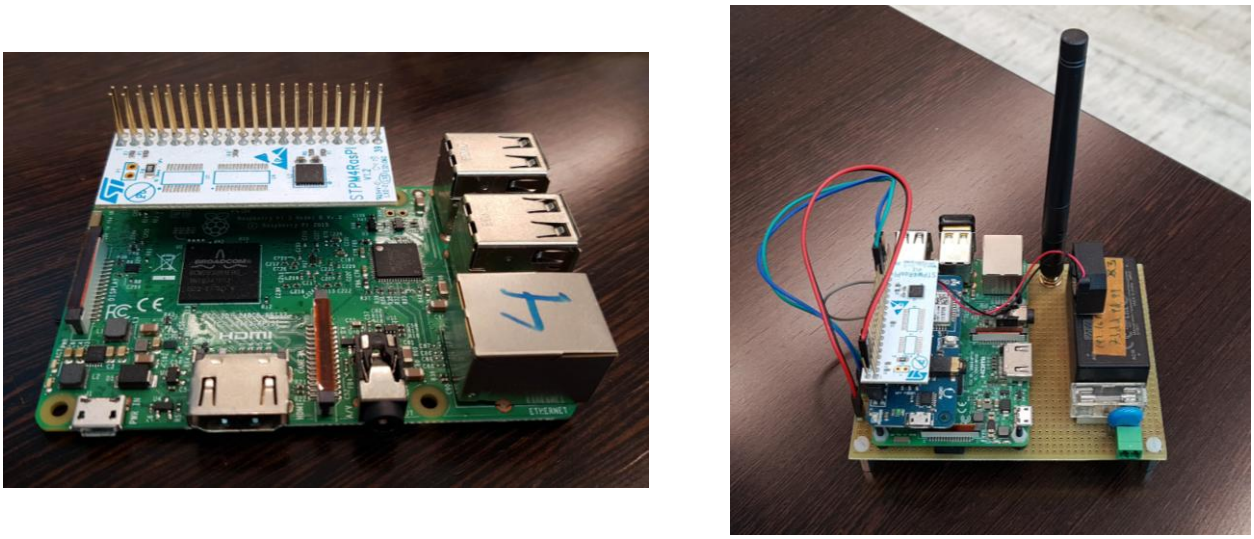

Figure 3. EnMon and Crow design integrating a TPM module

The envisioned use of the TPM implied employing its internal features and tools to provide encryption of the data that both devices send to the M-Sec servers. In order to take the better option, two different TPM modules were tested: one property of ST Microelectronics and the other one supplied by Infineon. Thus, a just comparison is established while the benefits and expected return of such integration are explored.

However, the corresponding trials and lab demonstrators performed showed the results were not exactly as initially expected, mainly due to the fact these TPM modules were not suited to provide the kind of security envisioned. In addition, the kind of support provided to Operating Systems (OSs), such as the one employed in the Crowd Counting device, buildroot, is not equal to the one offered to alternatives such as Debian or Devuan. Finally, another blocking issue directly refers to the interaction of a TPM with an L4 microprocessor, since ST itself is promoting the STSAFE-A100 for this type of interaction.

All in all, a new course of action was chosen, not involving the TPM to provide the desired security.

Figure 4. EnMon device in its encasement

The EnMon device not including a TPM was discussed and developed (see Figure 4), including a new microprocessor, an alternate version of the STM32-L4, which includes a module specifically devoted to performing encryption processes over the data captured by the sensors and thus sends them over the air in a safe and secure way. These encryption mechanism and keys are replicated in the other end of the communication channel, where after the decryption process is conducted, data can be presented in the web application that users employ to interact with the deployment.

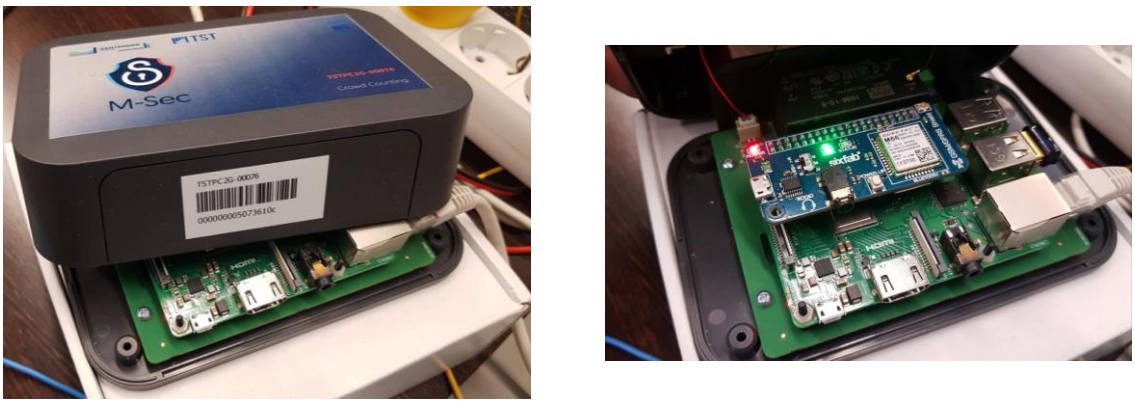

Figure 5. Crow device in its encasement

In parallel, another version of the Crow device providing encryption through programming features and using TPM for the booting mechanism was developed during this period. Figure 5 shows the result that acts in the first stage of the Pilot 1 deployment. When putting it to work, we noticed it seems that it has given TCP/IP sending failures and has stopped working. Attached is part of the log:

```
...
2020-10-28 12:02:58 - [INFO] End stop BT process: 20202810-12:02:58
2020-10-28 12:03:00 - [INFO] 6 BT devices detected
2020-10-28 12:03:03 - [INFO] Sent BT results: OK
2020-10-28 12:03:03 - [INFO] Reset WiFi
2020-10-28 12:03:06 - [INFO] Start WiFi process: 20202810-12:03:06
2020-10-28 12:03:06 - [INFO] Start BT process: 20202810-12:03:06
```

```
2020-10-28 12:08:09 - [INFO] Init stop WiFi process: 20202810-12:08:09
2020-10-28 12:08:10 - [INFO] Waiting for stopping WiFi process: 20296
2020-10-28 12:08:10 - [INFO] End stop WiFi process: 20202810-12:08:10
2020-10-28 12:08:13 - [INFO] 103 WiFi devices detected
2020-10-28 12:08:38 - [ERROR] Sending data by TCP to 212.83.139.102:7698
2020-10-28 12:08:48 - [ERROR] Sending data by TCP to 212.83.139.102:7698
2020-10-28 12:09:08 - [ERROR] Sending data by TCP to 212.83.139.102:7698
2020-10-28 12:09:28 - [INFO] Sent WiFi results: OK
2020-10-28 12:09:28 - [INFO] Init stop BT process: 20202810-12:09:28
2020-10-28 12:09:29 - [INFO] Waiting for stopping BT processs: 20328
2020-10-28 12:09:29 - [INFO] End stop BT process: 20202810-12:09:29
2020-10-28 12:09:41 - [INFO] 14 BT devices detected
2020-10-28 12:09:52 - [ERROR] Sending data by TCP to 212.83.139.102:7698
2020-10-28 12:10:00 - [INFO] Sent BT results: OK
2020-10-28 12:10:00 - [INFO] Reset WiFi
2020-10-28 12:10:02 - [INFO] Start WiFi process: 20202810-12:10:02
2020-10-28 12:10:02 - [INFO] Start BT process: 20202810-12:10:02
1970-01-01 01:00:19 - [INFO] Init people_counterd SVN: 120
1970-01-01 01:00:19 - [INFO] IP server: 212.83.139.102
1970-01-01 01:00:19 - [INFO] Port server: 7698
1970-01-01 01:00:19 - [INFO] Wait time: 300
1970-01-01 01:00:19 - [INFO] Advanced mode activated
1970-01-01 01:00:19 - [INFO] TCP connection
1970-01-01 01:00:19 - [INFO] Encrypt data no activated
1970-01-01 01:00:19 - [INFO] Logger mode activated: /root/pc_000000000f518faa.log
1970-01-01 01:00:19 - [INFO] WiFi device activated: wlan0
1970-01-01 01:00:19 - [INFO] BT device activated: hci0
1970-01-01 01:00:19 - [INFO] Device /sys/class/net/wlan0mon detected
1970-01-01 01:00:19 - [INFO] Device /sys/class/bluetooth/hci0 detected
1970-01-01 01:00:19 - [INFO] Waiting for device /sys/class/net/ppp0 to appear
1970-01-01 01:00:30 - [INFO] Retry [1/5] ppp conexion
1970-01-01 01:00:32 - [INFO] Waiting for device /sys/class/net/ppp0 to appear
2021-01-15 10:43:53 - [INFO] Init people_counterd SVN: 120
2021-01-15 10:43:53 - [INFO] IP server: 212.83.139.102
2021-01-15 10:43:53 - [INFO] Port server: 7698
2021-01-15 10:43:53 - [INFO] Wait time: 300
2021-01-15 10:43:53 - [INFO] Advanced mode activated
2021-01-15 10:43:53 - [INFO] TCP connection
2021-01-15 10:43:53 - [INFO] Encrypt data no activated
...
```

After an analysis, the fault was found. It seems that, from time to time, the USB WiFi in monitor mode does not work properly, but if in three consecutive retries we see that it is not able to locate any MAC, we restart the device. With this action, we solved this possible problem.

We realized that there are things that can be improved in the python script that does the sending of frames to the backend. Currently, it is opening and closing a TCP socket for each send, so this has to be changed to

be more efficient. Maybe in the backend, opening and closing so many sockets may cause some overload. To solve this, we proceed to conduct the development of a python class, with the aim of managing the encryption and sending of data in the people counter. It will be possible to select the sending of encrypted data, as well as to select the type of communication: "TCP" or "UDP". This class shall also be in charge of the management of possible errors derived from the type of connection selected. In addition, a method is implemented to force a memory release to avoid leak memory type phenomena, due to the large number of sockets being opened and closed.

The final objective implies achieving the combination of these techniques with the use of the TPM device to provide a secure booting mechanism, which in the end wraps up the whole and complete security features for these kinds of IoT devices.

TPM2 device or equivalent

One approach of the security for embedded devices that have been used in M-Sec is to attach secure elements as a trust anchor to these devices. We used an EAL4+ certified component in the first part of the project to be attached to a device in order to provide security functions. The goal is to go from an unsecured device to an intermediate level of secured device without refactoring the entire hardware.

Measured boot

Measured boot is defined by the Trusted Computing Group (TCG) as the ability to, during boot time, measure the operating system prior its execution. Measurement in this case consists of hashing the loaded code and extend a PCR of a TPM with this hash.

The measurement taken in this manner can be used to:

1) Seal or unseal keys or very small amount of data, which will be stored within the TPM, offering a EAL4+ storage. The value will be readable only if the measurement matches a reference value.
2) Attest the integrity of the platform remotely, with a quote of the PCR signed by the TPM.

OS Security

The OS security uses mechanisms provided by the TPM to strengthen the security mechanisms such as encryption. It uses mechanisms such as IMA and EVM within Linux and relies on the PCR provided by the TPM.

This security setting has been interfaced with the security manager described in D4.10 *"End to end security"* [D410].

Package Information and installation instructions

This demonstrator will be part of the two pilots that constitute Use Case 1, to be conducted in Santander (Spain).

Required Tools and dependencies

Assuming the build machine is debian-based (debian, ubuntu, etc.), the following dependencies shall be installed :

```
apt-get install crossbuild-essential-arm64 fakeroot git kernel-wedge quilt ccache flex bison
libssl-dev rsync libncurses-dev bc patchutils dh-python dh-exec libelf-dev device-tree-compiler
```

TPM physical connection

In production, the TPM would be directly soldered and routed on the board, but in our case, the devices to secure are legacy, or we are in a prototyping phase. The pinout of the board we have been using is designed for the Raspberry PI GPIO (General Purpose Input/Output) port. It can be plugged directly on top of the Rasbperry, as shown in Figure 6.

Figure 6: Raspberry PI with the STPM4Raspi extension in White

For other devices, we can use either SPI (Serial Peripheral Interface) bus or I2C (Inter-Integrated Circuit) bus to connect the TPM to the micro-controller or processor. We have done a temporary wiring in order to conduct trials using an STM32L4 (Nucleo L476RG development board), as shown in Figure 7.

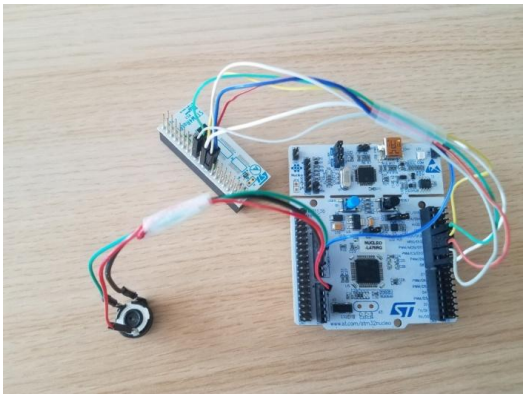

Figure 7: External wiring for microprocessor developments

Bootchain

The bootchain is made of two components: the ARM trusted firmware and the U-Boot bootloader. In some cases, the Advanced RISC Machines (ARM) Trusted Firmware may be optional, its main functionality is to enable the TrustZone®, which is the ARM ‘s implementation of a Trusted Execution Environment (TEE).

U-Boot has been patched in order to support the physical TPM. Patches consist in:

1) declaring the TPM component on the SPI bus in the device tree

2) adding SPI support for the BCM2738 chipset.

A predefined configuration enabling the TPM commands, libraries, and associated security dependencies have been added. Once this code has been retrieved, the compilation occurs as follows

```
make CROSS_COMPILE=aarch64-linux-gnu-
```

Optionally, the TrustZone® features can be enabled using the OPTEE OS in the secure world. OP-TEE can be compiled with the following command:

```
make -C ../optee_os O=out/arm CFG_ARM64_core=y \
CROSS_COMPILE="aarch64-linux-gnu-" \
CROSS_COMPILE_core="aarch64-linux-gnu-" \
CROSS_COMPILE_ta_arm64=aarch64-linux-gnu- \
CROSS_COMPILE_ta_arm32=arm-linux-gnueabihf- \
CFG_TEE_CORE_LOG_LEVEL=3 \
DEBUG=1 CFG_TEE_BENCHMARK=n PLATFORM=rpi3
```

In order to compile the ARM Trusted Firmware (ATF), sources can be retrieved from the mainline repository and compiled with the following command:

```
make   PLAT=rpi3 DEBUG=0   CROSS_COMPILE=aarch64-linux-gnu- \
 BL33=../u-boot/u-boot.bin \
 RPI3_PRELOADED_DTB_BASE=0x01000000 all fip
```

If Open Portable Trusted Execution Environment (OPTEE) is to be added, we can use this other command

```
make PLAT=rpi3 DEBUG=0  CROSS_COMPILE=aarch64-linux-gnu-  NEED_BL32=yes \
 BL32=../optee/optee_os/out/arm/core/tee-header_v2.bin  \
 BL32_EXTRA1=../optee/optee_os/out/arm/core/tee-pager_v2.bin \
 BL32_EXTRA2=../optee/optee_os/out/arm/core/tee-pageable_v2.bin \
 BL33=../u-boot/u-boot.bin CRASH_REPORTING=1   SPD=opteed \
 RPI3_PRELOADED_DTB_BASE=0x01000000  all fip
```

Linux Kernel

In order to be fully available to application and to ensure the OS integrity, the Linux Kernel has to be recompiled with specific additional features such as:

- Declaring the TPM device within the device tree
- Enabling TPM2 driver (as a character device)
- Enabling the Integrity measurements and extended verification module (IMA and EVM)
- If needed, enabling the OPTEE driver for trusted applications

On debian-based systems, the kernel can be recompiled with the following suite of commands:

```
ARCH=arm64
FEATURESET=none
FLAVOUR=arm64
CROSS_COMPILE=aarch64-linux-gnu-

export $(dpkg-architecture -a$ARCH)
export PATH=/usr/lib/ccache:$PATH
export DEB_BUILD_PROFILES="cross nopython nodoc pkg.linux.notools"
export MAKEFLAGS="-j$(($(nproc)*2))"
```


```
export DEBIAN_KERNEL_DISABLE_DEBUG=
[ "$(dpkg-parsechangelog --show-field Distribution)" = "UNRELEASED" ] &&
  export DEBIAN_KERNEL_DISABLE_DEBUG=yes

fakeroot make -f debian/rules cleanfakeroot make -f debian/rules orig
fakeroot make -f debian/rules source
fakeroot make -f debian/rules.gen setup_${ARCH}_${FEATURESET}_${FLAVOUR}
fakeroot make -f debian/rules.gen binary-arch_${ARCH}_${FEATURESET}_${FLAVOUR}
fakeroot make -f debian/rules.gen binary-libc-dev_arm64
```

Linux Security

In order to monitor the execution of Linux and its application, we use the IMA/EVM module that we need to configure for our demonstration. The modules have been activated within the kernel configuration in the previous step, and then, we need to configure them with a list of files to monitor. We propose a script to deploy the parameters to be run after the installation of the tools.

```
apt-get install autoconf libtool libssl-dev libattr1-dev libkeyutils-dev asciidoc ima-
evm-utils
/opt/ima/deploy.sh
```

The policy regarding this module can be edited in */etc/ima/ima_policy* while the measurements can be monitored in the */sys/kernel/security/ima/ascii_runtime_measurements* file.

Download and Run Demonstrator

The following files compose the demonstrator:

- *arm-trusted-firmware.tar.bz2* which contains the ARM's Trusted Firmware, which acts as a hypervisor between the Linux OS and the TrustZone during runtime.
- *u-boot.tar.bz2* which contains the Linux boot loader with a patch to support the SPI bus and the proper configuration in order to enable the TPM2 (device tree) and the measurements (boot script)
- *optee.tar.bz2* (optional) which contains the OP-TEE system for the TrustZone® tailored for the Raspberry PI.
- *linux-4.19.12.tar.*bz2 which is a patched version for the debian kernel for the Raspberry PI with TPM2, IMA/EVM, and OP-TEE driver. The device tree matched the TPM2 development board we have been using.
- *rpi-sdcrypt.tar.bz2* which is a set of files to be deployed on the operating system in order to manage the encryption of the partition and decryption using the TPM2 NVRAM based on a PCR policy. IT handles the provisioning phase as well as the *initramfs* generation for the decryption.

At the time of edition of this deliverable, the development files have not been published, and this publication is under review at CEA.

User Manual

In order to successfully run this demonstrator, no specific user manual is required since the deployment of the secured IoT devices will be rather straightforward, and the user will just need to employ a mobile application to interact with them.

Licensing (if applicable)

Even if most of the developments have been made using open source code, the status of the development includes certain blocks of proprietary code (non-free). Nevertheless, the future steps will, for sure, imply developing open source code that could be made available.

Currently, some review process is being conducted in order to push some patches into mainstream repositories.

Perimeter Defense - Intrusion Detection System (IDS) for IoT devices

Besides the hardware-based internal security for the IoT devices, M-Sec technical partners also focused on developing software-based security options that can be easily implemented on existing IoT gateway devices without any additional hardware. For this purpose, Use Case 3 was selected as an example that portrays a common approach in the field to implement mobile sensor-based SMART solutions for managing various beneficial tasks by the municipality in a smart city. The perimeter defense security solution represents a software solution designed and developed to give an answer to the requirements posed by the Use Case 3. The "Mobile Sensing Platform" being used was directly connected to the Internet without any security features. The reliability and authenticity of the data from the sensors could not be protected.

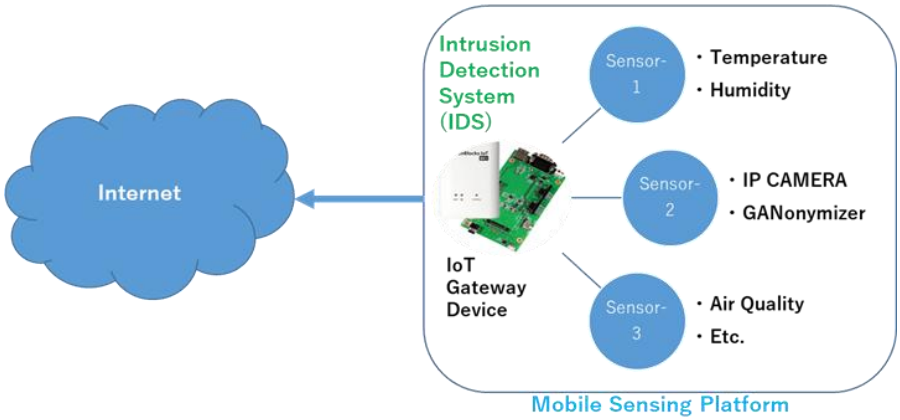

Figure 8. Secured mobile sensing platform

The initial design was aimed at the security objectives through the integration of software-based multi-layered security on the IoT gateway device, which was the entry/exit point for being used for providing the trustworthy data from various IoT sensors. A threat and risk assessment was conducted to understand the vulnerable points and appropriate measures were implemented to mitigate the risks. Hardening was done to reduce the attack surface, TLS encryption was chosen for securing data, and perimeter defense was provided via an open-sourced intrusion detection system (Suricata) [SRC] supported by Open Information Security Foundation (OISF) [OISF], covering known threats, policy violations, and malicious intents. The solution was further strengthened with attack analysis from the IoT honey pot during the designing phase. Though the results in Pilot-1 were promising, they also revealed some areas for improvement. In order to reduce the risk from **unknown attacks or zero-day threats**, the IDS was further customized and configured to keep its signatures up-to-date by linking it to a number of signature-providing sources, such as emerging threat intelligence and Talos (Vulnerability Research Team, VRT, ruleset) [TAL].

Package Information & Installation Instructions

For the sake of easiness, we have compiled the customized OS hardening commands and the IDS software together as a package for easy installation on IoT devices used in Use Case 3.

Required Tools and dependencies

The demonstrator has been designed for securing mobile sensing platform having the following specifications:

- Intel Atom Processor 500 MHz Dual Core, 1GB RAM, 4GB Flash, Debian GNU/Linux.

Download and Run Demonstrator

The installation file is named as "*installer.sh*" that can be downloaded securely over 3G or internet connection with the following command:

```
$ scp -P 64295 -i <key> yyyyyyyy@xxx.xxx.xxx.xxx:~/iot-k/installer.sh .
```

After downloading, check and confirm the integrity of the downloaded file using Message-Digest Algorithm 5 (MD5) hash, as follows:

```
$ md5sum installer.sh
MD5: 5cbxxxxxxxxxxxxxxxxxxxxx
```

Installation Steps:

1) The "*installer.sh*" file should be downloaded into the "*/root*" directory and the directory should look something like this:

```
root@iot:~# ls -l
total 40712
-rw-r--r-- 1        root root         0 Sep 27  2017 1
-rwxr-xr-x 1        root root         41678664 Nov 26 16:59 installer.sh
-rw------- 1        root root         1675 Nov 26 16:32 key.pem
root@iot:~#
```

2) Launch installer shell as follows:

```
root@iot:~# chmod +x installer.sh
root@iot:~# ./installer.sh
installer/
installer/start_ips.sh
installer/suricata-4.1.5.tgz
installer/root.tgz
:        :        :        :
```

3) Wait for it to complete updates, download, and install the program. It will finish installation and return the prompt as follows:

```
:        :        :        :
Index setup finished.
Loading dashboards
```

```
Loaded dashboards
Loaded machine learning job configurations
Loaded Ingest pipelines
root@iot:~#
```

4) Now check root directory contents and you should see something like below:

```
root@iot:~# ls -l
total 40724
-rw-r--r-- 1      root root         0 Sep 27  2017 1
-rwxr-xr-x 1      root root         41678664 Nov 26 16:59 installer.sh
-rw------- 1      root root         1675 Nov 26 16:32 key.pem
-rw-r--r-- 1      root root          801 Nov 26 17:50 readme.txt
-rwxr-xr-x 1      root root         187 Nov 26 17:56 start_ids.sh
-rwxr-xr-x 1      root root         343 Nov 26 17:58 start_ips.sh
```

The "*start_ids.sh*" is for monitoring mode, whereas, "*start_ips.sh*" is for preventing attacks mode.

Run Demonstrator Guide:

Demonstrator runs in the backgroud, sending log events to the visualization tool in the cloud. On IoT gateway devices, IDS program will be initiated at startup with the following command:

```
root@iot:~# ./start_ips.sh
```

Licensing (if applicable)

The software solution developed as IDS for the IoT devices is based on Open Source Software (OSS) and, therefore, does not need any licensing. Whereas, IoTPOT is a proprietary asset that is used for study and analysis purposes only during the research and development phase.

Stealth Security

A new feature was researched and tested that helps in further addressing the unknown attacks and zero-day threats. By using the port knocking methodology (see Figure 9), an IoT device was able to hide its available ports and enable the port only on receiving an authenticated known sequence of knocks on specific ports, being randomly generated but at sync between client and serving IoT device.

As the attackers need to know which ports are open and what service is being provided for conducting an attack, this stealth security feature further strengthens the security by hiding the ports. Another advantage of this security feature is that it helps the IoT device to consume less power than without it. As the device is stealthily hidden from the Internet, the number of unauthorized packets also decrease that contributes to power-savings.

Due to hidden/closed ports, the device is invisible to the attackers. This also helps with avoiding any known/unknown attacks. Ports are opened for only authenticated requests after receiving a correct sequence of knocks on specific ports.

The setup employed in the lab to conduct an experiment related to this feature can be checked in Figure 10.

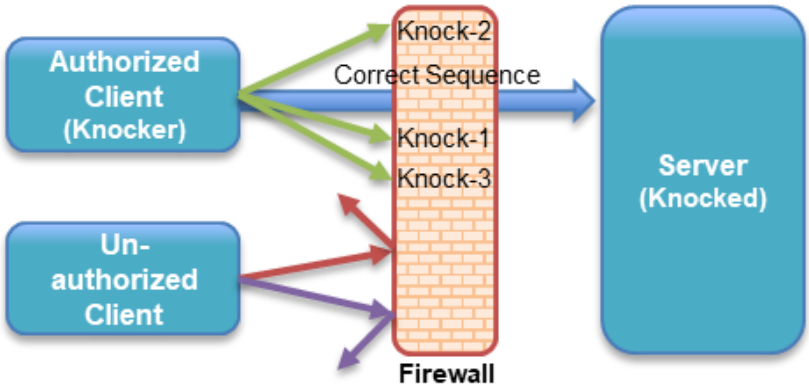

Figure 9. Port knocking methodology

Figure 10. Lab experiment

Therefore, unauthorized scans are not able to find open ports and are not able to attack either. At the same time, this solution helps the IoT device in reducing its power consumption. Details can be found in the research paper *"Empowering resource-constraint IoT gateways with port knocking security"* at Computer Security Symposium 2020 (CSS2020) [PKS] and peer-reviewed *paper "Empirical analysis of security and power-saving features of port knocking technique applied to an IoT device"* – Journal of Information Processing (JIP) [SPS].

Table 2. Port knocking security effectiveness (6-weeks test results).

Security Effectiveness of Port Knocking Feature	With Port Knocking	Without Port Knocking
Unauthorized SSH Login Attempts	0 Times	431,142 Times
Source IP Addresses	0 Hosts	5,424 Hosts

Package Information & Installation Instructions

For the sake of easiness, we have compiled the customized stealth security feature together as a package within the installer (as explained previously) for easy installation on IoT devices used in Use Case 3.

Security Monitoring and Visualisation Tool

As this tool interacts very closely with IoT gateway devices via the embedded agent, it has been included in the Devices Security FG, instead of Cloud Tools FG. In order to stay vigilant and monitor security threats to the IoT devices layer from anywhere in the cloud, an analysis tool was designed based on the Kibana elastic-search running on Amazon Elastic-Search Service [AESS] that can translate the data into easy-to-understand graphical information. The tool collected and examined the log activity from embedded agents in the IoT gateway devices. Figure 11 depicts the whole visualization process.

Figure 11. Visualization process

Embedded data collection agents (*filebeat*) in the IoT gateway devices upload the security logs to the aggregator module in the AWS elastic cloud for further analysis. The aggregator module collects all the data from various IoT gateways and passes it to the data analysis engine, which examines all the data with the help of the threat detection module. The results are flagged as alerts/events in the visual graphics module of Kibana for further investigation and easy-to-understand security threat monitoring. Examples of these features are available in the following figures (Figure 12, Figure 13 and Figure 14)

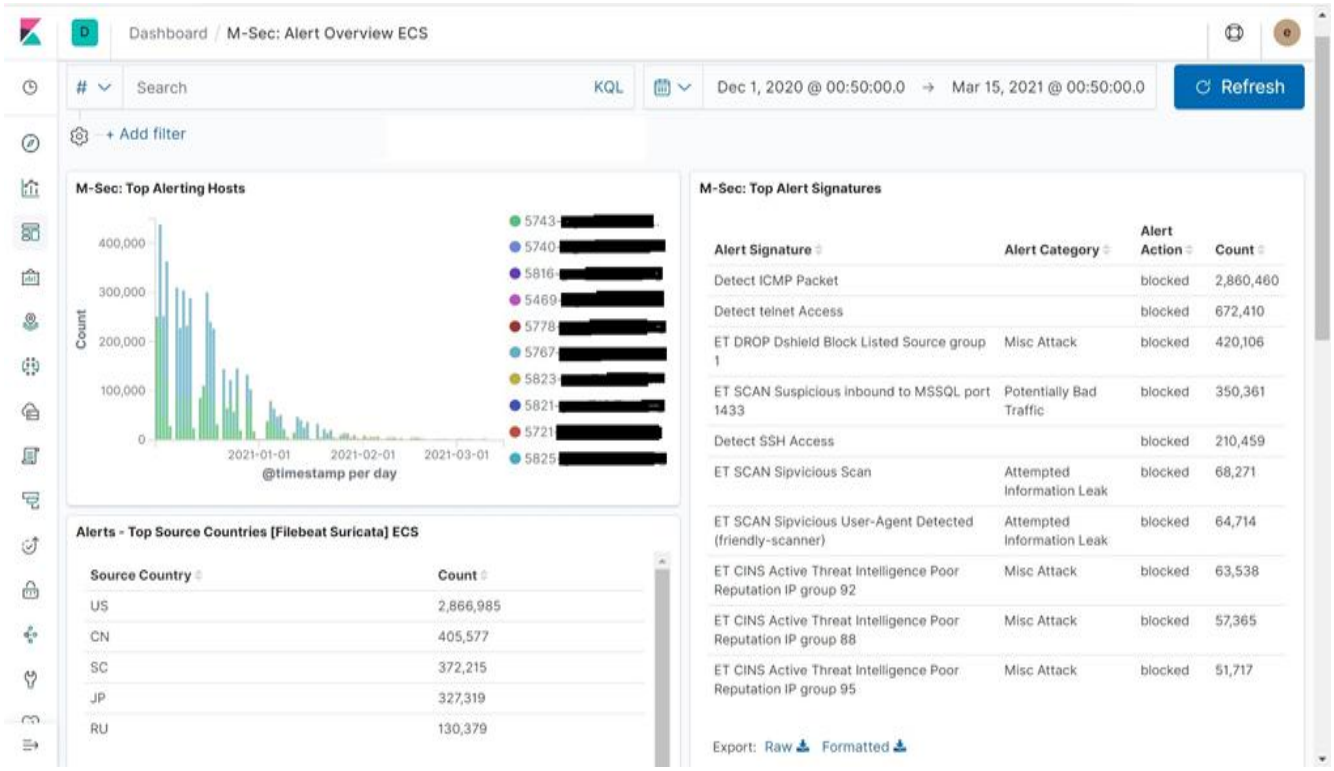

Figure 12. Alerts – Security monitoring & visualization tool

Figure 13. Events - Security monitoring & visualization tool

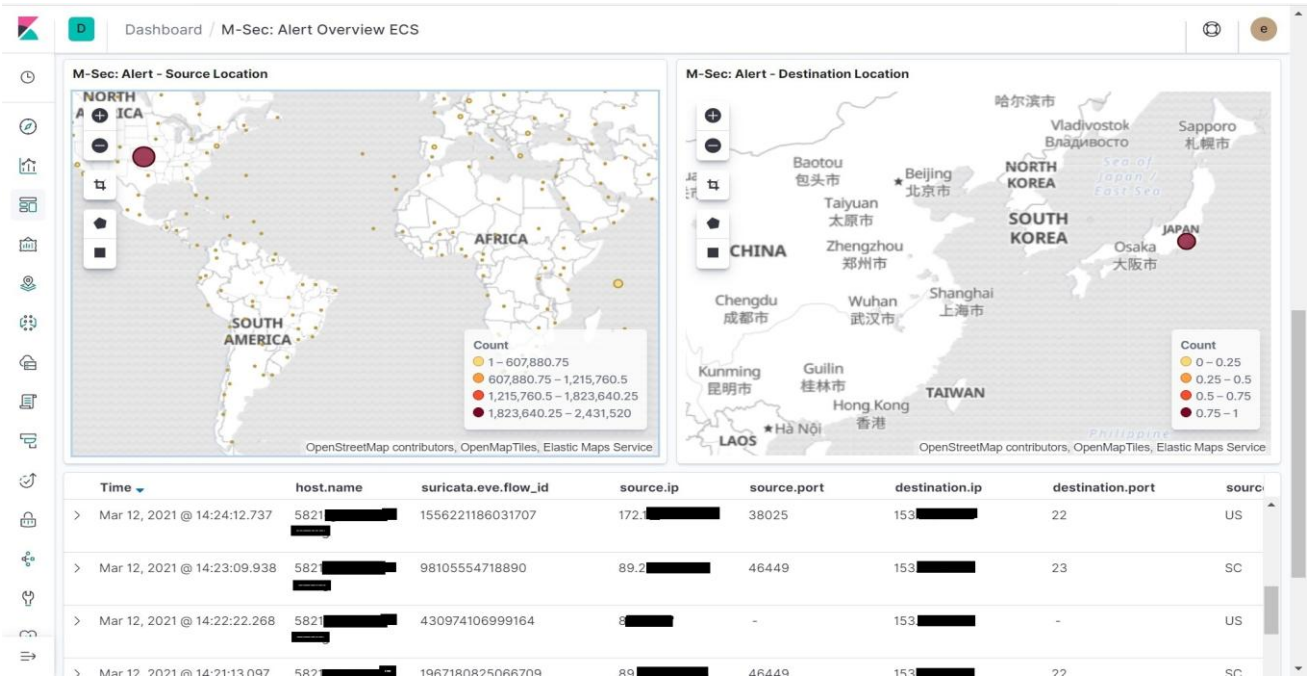

Figure 14. Geo-Location – Security monitoring & visualization tool

Few areas of improvement were found in the pilot-1 testing. Further improvements will be made to correct the errors in pilot-2. The geographical location of the IoT gateway device was also shown incorrectly as it was based on the source IP address of the cellular network provider. This will also be improved to reflect the geo-location of devices in Fujisawa city correctly.

2.3 API

The security features for the Devices Security FG are directly embedded into the devices. As such, an API is not needed. The monitoring and visualization tool has a web-based interface (see Figure 15) that runs on Amazon elastic-search service utilizing Kibana analytics engine. This is an open-source tool and can be obtained from github [KIB] along with the API-related information. The most easy way is to simply create deployment on Amazon elasticsearch services. This way you don't need to worry about the APIs and can access it from anywhere on the web. A getting started guide is also available.

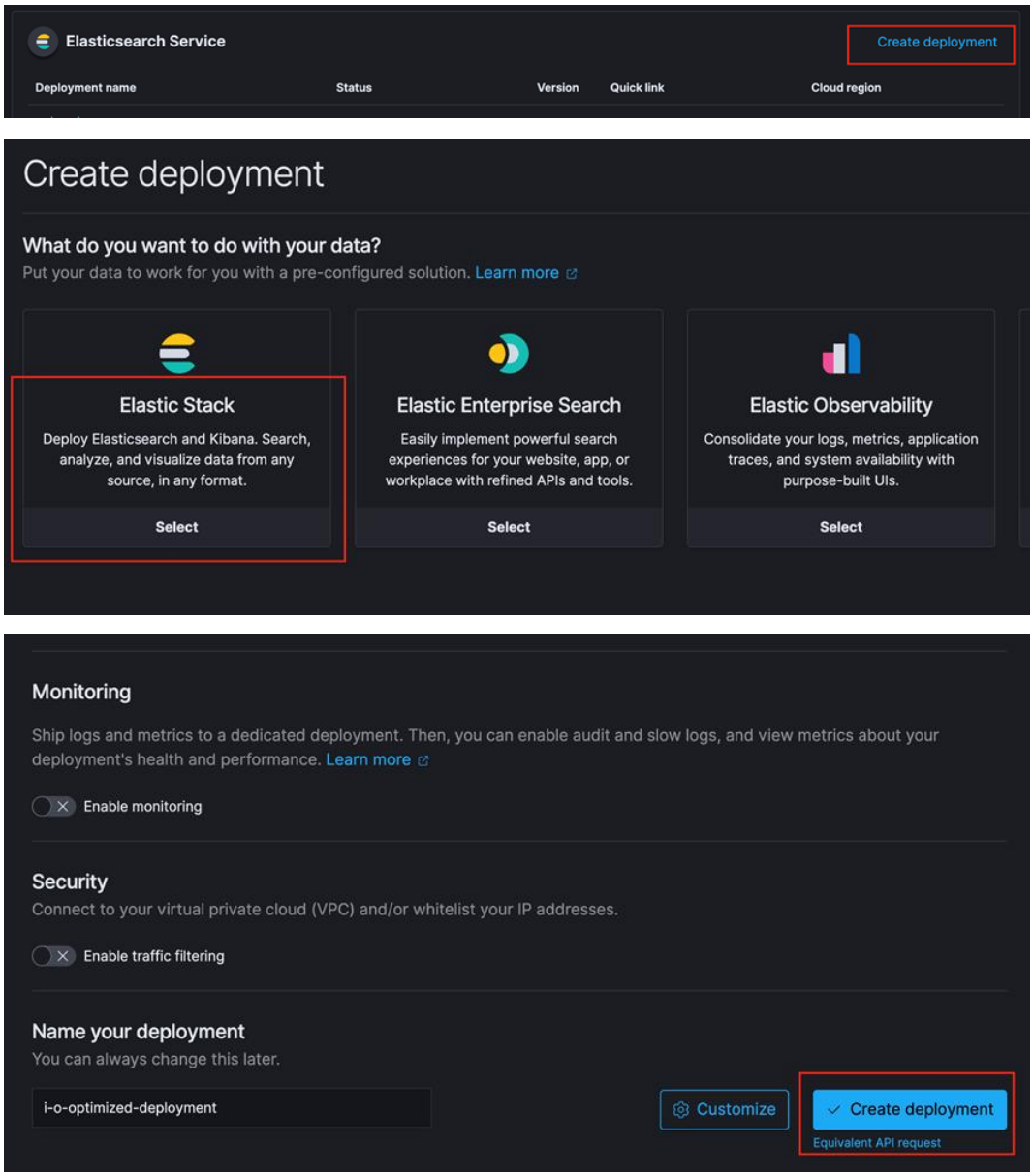

Figure 15. Monitoring and visualization tool web-based interface

Note your "Cloud ID" and access information. Then import this info into the agent "filebeat" configuration file on the IDS so that each IoT device can report to this deployment.

2.4 Interaction with other FGs

The following Figure 16 presents the interactions of the Devices FG and Devices Security FG with other FGs and components of the M-Sec solution.

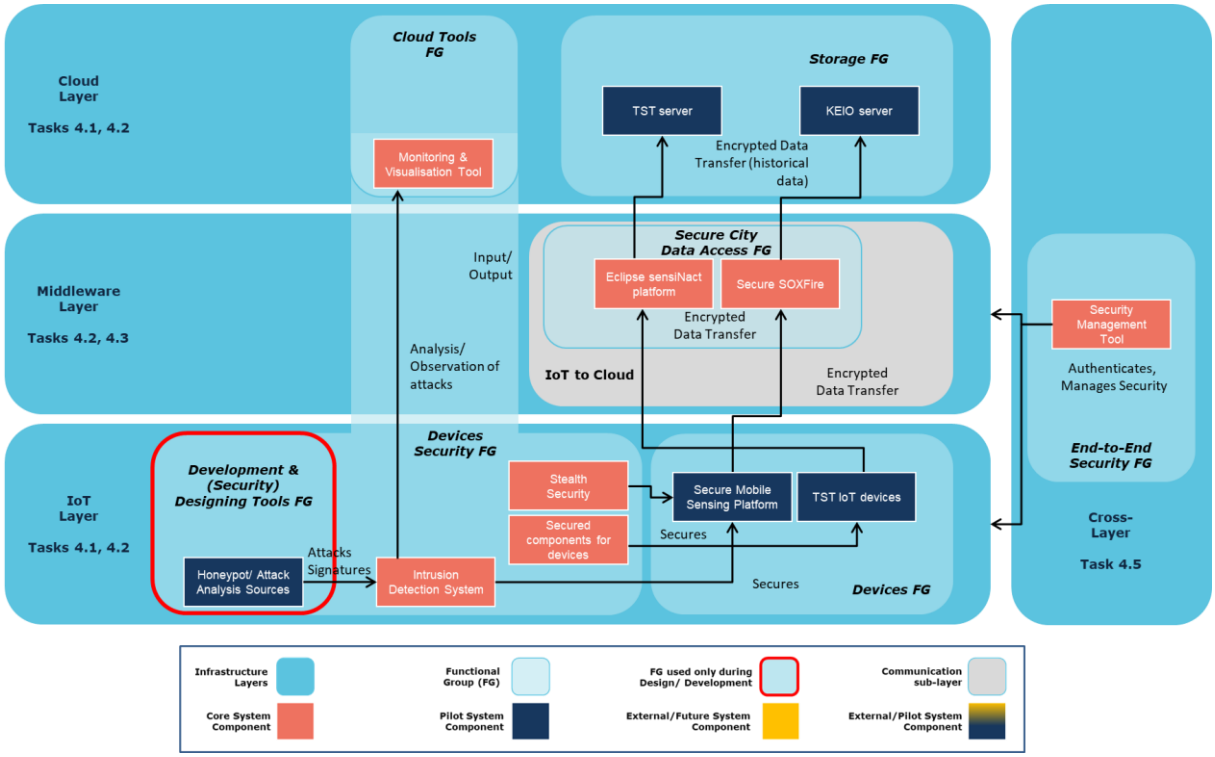

Figure 16. Interaction of the Devices and Devices Security FGs with other FGs

Interaction with Development & Security Designing Tools FG

During the designing phase, the IoT honeypot attack analysis system was leveraged to analyze and strengthen security parameters for the IoT devices.

Interaction with Cloud Tools FG

IoT gateway devices were embedded with filebeat agents for sending the security logs to the monitoring and visualization tool in the elastic cloud. The https-based tool provides the collected information in an easy-to-understand graphical manner for monitoring purposes.

Interaction with End-to-End Security FG

The Security Management Tool manages the overall security from authentication, authorization, accounting, and auditing of the IoT devices. Details are available in D4.10.

The security manager interacts with the secured devices in the following manners:

- During built-time of the device, it provides accounting and certificates for the provisioning step thru an enrolment procedure. This enrolment procedure uses a One-Time Password (OTP) to validate the inclusion within the M-Sec instance. Once completed, the device is properly configured with the same authentication backend thus facilitating a secure communication with these other FGs.
- During runtime, it audits the devices using the intrusion detection capabilities of the devices such as the TPM quoting or the output of the IDS. The security manager relies on these inputs in order to automate response action such as putting the device in quarantine and notifying its user of a potential breach.
- Option is also available to manage encryption by Security Manager's PKI module.

Interaction with Secured Data City Access FG

The Secured Data City Access FG is on the Northbound. The data are securely transferred across the cloud. For example, in case of Use Case 3, the data are securely transferred across the cloud using XMPP (Extensible Messaging and Presence Protocol) protocol that has a built-in encryption support in it.

Relation to the rest of the FGs

The Devices Security FG does not interact directly with the Privacy Management FG. The privacy management tool (Ganonymizer) is used directly with the IP camera. Therefore, the tool only sends the anonymized data via the secured IoT gateway device. Details are available in D4.4.

There is also no direct interaction with the Secure & Trusted Storage FG's components. Data travels through other FG before reaching that module in the architecture. Similarly, the FG does not interact directly with the IoT Marketplace FG. Only data collected through the secured sensors are exchanged in the marketplace.

3. Conclusion

The current report depicts the prototypes developed in the most recent stages of the project execution and their features to address the security concerns and risks reflected in WP2 & WP3 materials. These prototypes have been tested in real-life scenarios during the initial phases of the diverse M-Sec pilots, starting the integration processes and interactions with other WP4 elements. Table 3 below summarizes the discussion reflected in this document.

Table 3: Demonstrators and their correlation with Use Case Pilots

Demonstrator	Type	Use Case Pilots	Purpose
IoT Device with increased security	Hardware-based solution	Use Case 1 Pilot 1.1 Pilot 1.2	Provide embedded security layer to IoT devices
Perimeter Defense (IDS)	Software-based solution (Python)	Use Cases 3 & 4 Pilot-3.1 Pilot-4.1	Secures IoT mobile sensing platform by monitoring and preventing cyber-attacks
Stealth Security	Software-based solution (Python)	Use Cases 3 & 4 Pilot-3.1 Pilot-4.1	Secures IoT mobile sensing platform by hiding ports, saving power consumed, and preventing cyber-attacks
Security Monitoring and Visualisation Tool	Amazon Elastic Search (Kibana - JavaScript)	Use Cases 3 & 4 Pilot-3.1 Pilot-4.1	Security monitoring of IoT mobile sensing platform

Further validation tests and improvements will follow in the months remaining, based on the activities related to the second stage of the use cases. As in previous stages, the input received from end users, namely citizens and visitors from both smart cities, as well as representatives of the cities involved in the trials and interacting with the IoT equipment, is crucial for the consortium to evaluate the work done and extract useful conclusions.

On the other hand, the moment the second stage of the pilots becomes active, the consortium will continue keeping an eye on the kind of cyber-attacks the Devices FG may suffer from, in order to validate the countermeasures implemented and thus evolve the M-Sec platform as a whole, if required.

References

[AESS] Amazon elasticsearch service, https://aws.amazon.com/elasticsearch-service/?did=ft_card&trk=ft_card

[D33] M-Sec project Deliverable 3.3, *"M-Sec Architecture: Functional and technical specifications – first version"*, July 2019

[D34] M-Sec project Deliverable 3.4, *"M-Sec Architecture: Functional and technical specifications – final version"*, June 2020

[D35] M-Sec project Deliverable 3.5, *"Risks and security elements for a hyper-connected smart city"*, June 2020.

[D41] M-Sec project Deliverable 4.1, "*M-Sec IoT Security*", December 2019

[D410] M-Sec project Deliverable 4.10, "*M-Sec End to end security"*, March 2021

[KIB] Kibana Analytics Engine available in GitHub, https://github.com/elastic/kibana

[OISF] Open Information Security Foundation, https://oisf.net/

[PKS] Inoue, Y., Kato, S., Bokhari, A.H., Yoshioka, K. and Matsumoto, T.*: "Empowering resource-constraint IoT gateways with port knocking security",* Proc. Computer Security Symposium 2020(CSS2020), pp.362–367 (2020)

[SRC] Suricata, Open-sourced intrusion detection system, https://suricata-ids.org/

[SPS] *"Empirical analysis of security and power-saving features of port knocking technique applied to an IoT device"* – to be published in Sep-2021 in Special issue of *"Computer Security Technologies for Realizing Society 5.0"* - Journal of Information Processing (JIP)

[TAL] Talos, formerly the VRT, https://www.snort.org/talos

Annex

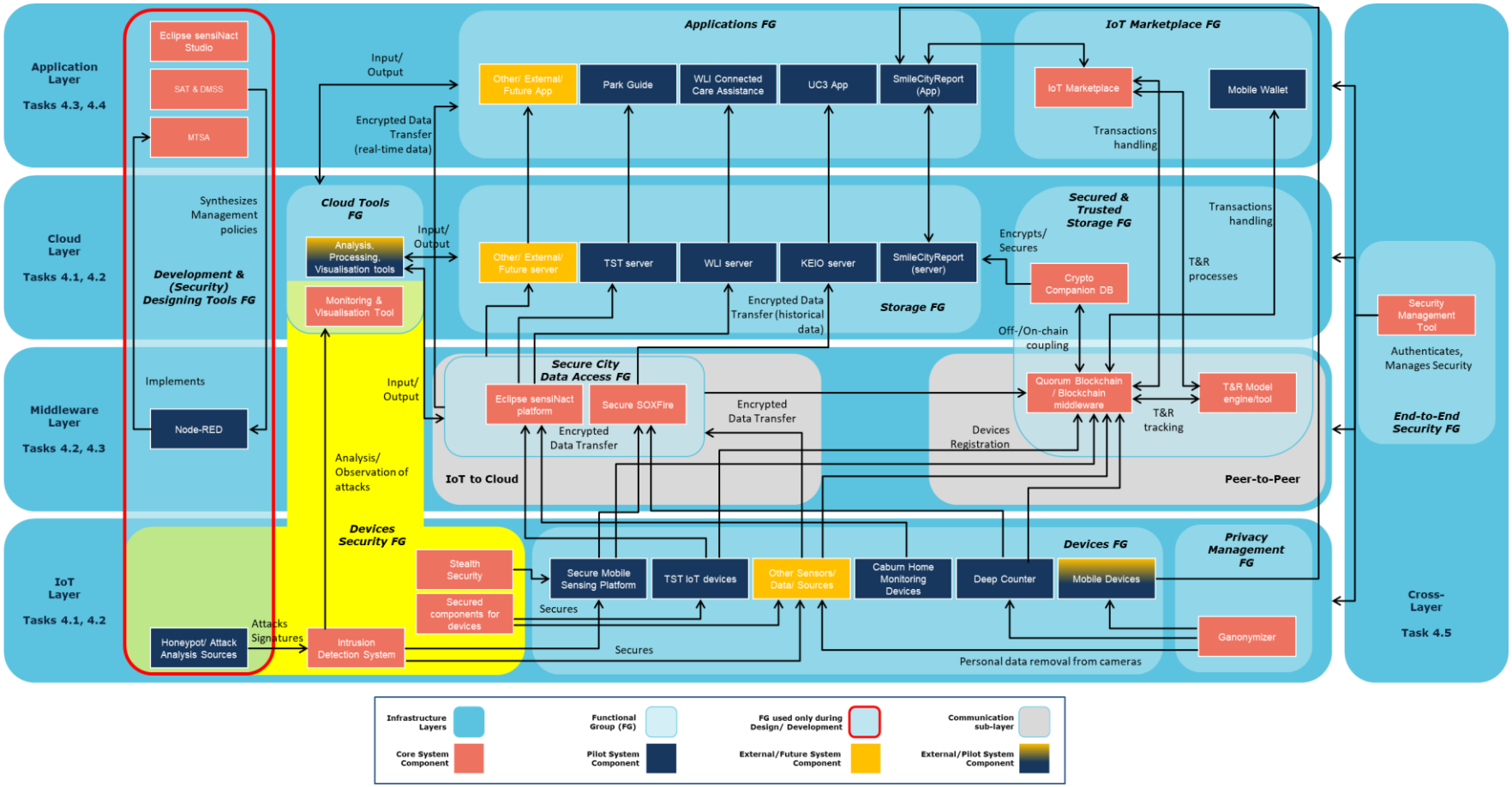

Figure 17. The M-Sec Architecture (T4.1 FG in yellow)