A coordinated framework for cyber resilient supply chain systems over complex ICT infrastructures

D3.2 Trust Manager IT1 integration

Document Identification			
Status	Final	**Due Date**	30/09/2021
Version	1.0	**Submission Date**	30/09/2021

Related WP	WP3	**Document Reference**	D3.2
Related Deliverable(s)	D3.1, D4.1, D4.2	**Dissemination Level (*)**	CO
Lead Participant	UPC	**Lead Author**	Eva Marin-Tordera
Contributors	XLAB UMinho	**Reviewers**	Ales Cernivec, XLAB
			Alexandre Santos, UMinho

Keywords:
Integration, TM workflow, validation strategy

This document is issued within the frame and for the purpose of the FHISY project. This project has received funding from the European Union's Horizon 2020 research and innovation programme under Grant Agreement No. 952644. The opinions expressed and arguments employed herein do not necessarily reflect the official views of the European Commission.

(*) Dissemination level: **PU**: Public, fully open, e.g. web; **CO:** Confidential, restricted under conditions set out in Model Grant Agreement; **CI:** Classified, **Int =** Internal Working Document, information as referred to in Commission Decision 2001/844/EC.

Document Information

List of Contributors	
Name	Partner
Henrique Santos	UMinho
André Oliveira	UMinho
Alan Jhones	UMinho
Jan Antic	XLAB
Xavi Masip	UPC
Eva Marín	UPC

Document History			
Version	Date	Change editors	Changes
0.1	30/08/2021	Eva Marín Tordera (UPC)	Contribution from XLAB, UMinho and UPC to sections 2, 3, 4 and 5.
0.2	31/08/2021	Eva Marín Tordera (UPC)	First compiled version with sections 2, 3, 4 and 5.
0.3	6/09/2021	Eva Marín Tordera (UPC)	Added introduction and glossary and improve figure 4 quality
0.4	8/09/2021	Xavi Masip (UPC)	Review content
0.5	10/09/2021	Henrique Santos (UMinho)	Review content and add glossary terms
0.6	15/09/2021	Jan Antic (XLAB)	Added content in section 5.3 and 5.4
0.7	16/09/2019	Eva Marín Tordera (UPC)	Added executive summary and re-written section 4.
0.8	21/09/2021	Eva Marín Tordera (UPC)	Added conclusions and list of acronyms
0.9	22/09/2021	Henrique Santos (UMinho) and Jan Antic (XLAB)	Revision of execution summary and conclusions. Version to be sent to reviewers
0.91	28/09/2020	Aless Cernivec (XLAB), Alexandre Santos (Uminho)	Internal review
0.95	28/09/2021	Eva Marín Tordera (UPC)	Version for Quality Check
1.0	22/09/2021	Jose Francisco Ruiz, Juan Alonso (Atos)	Quality assessment and final version to be submitted.

Quality Control		
Role	Who (Partner short name)	Approval Date
Deliverable leader	Eva Marin-Tordera (UPC)	28/09/2021
Quality manager	Alonso, Juan Andres (ATOS)	30/09/2021
Project Coordinator	Ruiz, Jose Francisco (ATOS)	30/09/2021

Table of Contents

Document Information 2
Table of Contents 4
List of Tables 5
List of Figures 6
List of Acronyms 7
Executive Summary 8
1 Introduction 9
1.1 Purpose of the document 9
1.2 Relation to other project work 9
1.3 Structure of the document 9
1.4 Glossary adopted in this document 9
2 SPI Integrated outcome 11
2.1 Data Management 12
2.2 Low-level interface (Downlink) 13
2.3 Up-level interface (Uplink) 13
2.4 Purposed technologies 13
3 TIM Integrated outcome 15
4 TM Global Integration 16
4.1 Functionalities to be integrated 16
4.2 TM Workflow 17
5 Testing and validation objectives 19
5.1 SPI functionalities 19
5.2 TIM functionalities 20
5.3 Metrics and KPIs for TM 21
5.4 Description of validation Strategy 22
6 Conclusions 23
References 24

List of Tables

Table 1: Summary of the internal architectural blocks and modules in TM 16

List of Figures

Figure 1- SPI architecture *11*
Figure 2. Flow Chart of TM Communication *17*
Figure 3: SPI testing Architecture *19*
Figure 4 - TIM PoC Testbed *21*

List of Acronyms

Abbreviation / acronym	Description
API	Application Programming Interface
CEF	Common Event Format
D3.2	Deliverable number 2 belonging to WP3
EC	European Commission
GDPR	General Data Protection Regulation
HTTP REST	HTTP Representational State Transfer
JSON	JavaScript Object Notation
JWT	JSON Web Token
KPI	Key-Performance Indicators
NED	Network Edge Device
PoC	Proof of Concept
REST	Representational State Transfer
SCM	Security Assurance & Certification Manager
SEN	Secured Edge Node
SPI	Security & Privacy Data Space Infrastructure
TIM	Trust & Incident Manager
TM	Trust Manager
WP	Work Package
XACML	eXtensible Access Control Markup Language
XL-SIEM	Cross-Layer Security Information and Event Management
XML	eXtensible Markup Language

Executive Summary

This deliverable describes the internal implementation and integration of blocks in the Trust Manager (TM) module of WP3. In previous deliverable D3.1[4], the blocks in TM (Trust Manager) have already been designed and described, namely TIM and SPI.

Deliverable D3.2 presents the TM integration for iteration 1 (IT-1). For IT-1, the following blocks are implemented in TIM: Vulnerability Assessment, Incident Detection, Mitigation and Threat/Attack Repository. Concerning SPI, this document describes the implementation of the Data Management and the Identity Manager — despite having a reference late to the Access Policy editor, that description indicates what we expect to do, but the Access Policy module will not be included in IT-1.

Starting from designs in D3.1, first of all the inputs and outputs of each of the blocks, as well as the technology used in their implementation are described. Taking these inputs and outputs into account, the integration between SPI and TIM is proposed, including a specific integration workflow.

As a summary, the whole input to TM comes always from different agents' tools in the use case infrastructure. The different tools implementing Vulnerability Assessment, Incident Detection and Mitigation in TIM analyse and process this data, producing an output, which is stored in the Threat/Attack Repository (jointly with the raw data that tools have received). The Threat/Attack Repository will be part of a general Central repository available for all the FISHY components, as the design idea of a pub/sub layer over storage is useful across the whole platform. The output could be an alert, a vulnerability detected, as well as the detection of attacks and/or incidents or the suggestion of a mitigation action. These outputs will be available to other FISHY components and tools through the Central repository.

In this deliverable, the validation strategy for TM is also proposed; first for each of the blocks, SPI and TIM, as well for the whole TM module. Different metrics are proposed to validate TM, such as latency and accuracy. From these metrics, different Key-Performance Indicators, KPIs, will be proposed, to be considered in WP6.

In TM integration for IT-1, the simplicity has been the key, and in this sense the input to the three TIM blocks (and their corresponding tools) has been simplified, and they receive the data directly from the agents in the infrastructure. Regarding the whole TM output, in form of alerts, incident detection or mitigation action suggestion, it will be available to other FISHY modules through the access to a Central Repository.

Finally, in this deliverable we have proposed different metrics to evaluate the TM performance. Based on these metrics, different KPIs will be proposed in the work in next months in WP3.

1 Introduction

1.1 Purpose of the document

This deliverable describes the final and integrated Trust Manager (TM) module release for IT-1, ready to be integrated with the Security and Certification Manager (SCM) module from WP4, as well as with IRO in WP5.

After the description of the design and implementation of each one of the blocks in TM, namely SPI and TIM, done in D3.1 [4], in this deliverable we describe on one hand, the SPI and TIM outcomes to be integrated, as well as the global TM integration. It is also outlined/drafted the validation strategy and test to be realized.

1.2 Relation to other project work

The integration efforts in T3.3 started in month M9, therefore this is the first output of this task, both in terms of reporting, this deliverable, and also as milestone, the TM module integrated. This task works in parallel with its counterpart task, task T4.3, which integrates all the blocks within the Security and Certification Manager in WP4. The output of both tasks will jointly feed the whole FISHY integration to be done in WP5 towards delivering the IT-1 version of the FISHY platform in month 15; as well as the whole proof-of-concept to be deployed in the three use cases.

1.3 Structure of the document

This document is structured in 5 major chapters

- **Chapter 2** presents the SPI integrated outcome.
- **Chapter 3** presents the TIM integrated outcome.
- **Chapter 4** presents the TM global integration.
- **Chapter 5** presents the testing and validation objectives.
- **Chapter 3** concludes the deliverable.

1.4 Glossary adopted in this document

- **Bbox**: Black Box approach to system development, when dealing with modules provided by third-party without any implementation details besides the input/output function (opposite to Wbox).
- **Wbox**: White Box approach to system development, when dealing with modules for which there is a full detailed implementation document along with source-code (opposite to Bbox).
- **Gbox**: Gray box approach to system development, when dealing with modules for which we have the output/input function and some information about internal details (a middle stage between Bbox and Wbox).
- **Pub/Sub**: Publication Subscription solution for data transaction systems providing some sort of middle storage and data organization mechanism.
- **OpenId Connect**: open specification of a simple identity management layer on top of the OAuth2 authorization protocol.

- **OAuth2**: OAuth 2.0 is an industry-standard protocol for authorization; it defines several flows to accommodate different Access Control requirements and implementations.
- **RabbitMQ**: a highly flexible open-source message-broker software that supports several well-known queuing and data streaming protocols.

2 SPI Integrated outcome

SPI (Security & Privacy Data Space Infrastructure) is the FISHY component responsible for handling in the proper way all measurements, metrics and events provided by the infrastructure (in the premises), making them available, in a controlled way, to upper FISHY components, namely TIM and SCM.

According to the project proposal, SPI is a subsystem based on the concept of decentralized and distributed data storage, in which users keep part of the data, creating resilient file storage and providing data sharing mechanisms, with Access Control capability. Towards that goal, it is necessary to identify, define and propose the different aspects required to facilitate the proper interfacing with the Security and Certification Manager module (SCM) and TIM modules – at a minimum –, particularly addressing the aspects related to data and identity management, and privacy-by-design practises (as imposed by GDPR).

To fulfil its function and according to the requirements imposed by the other FISHY components, D3.1 [4], that will use the data, the SPI proposed architecture includes the following functions/modules (see also Figure 1):

- **Data Adaptation,** responsible for the organization of data, which may come from different intervals, in different data models (e.g., XML, JSON, sensor data, logs) and existing different means of communication (e.g., REST, Pub/Sub and APIs);
- **Access Control and Identity Manager,** is responsible for authenticating user/processes connected to the secure and distributed data space;
- **Access Policy**, consists of a set of rules and associated manager subsystems, responsible for supporting the definition and control of the needed policies to assure the access control and privacy requirements;
- **Data Anonymization** is responsible for dealing with the privacy of the data set shared by the interested parties.

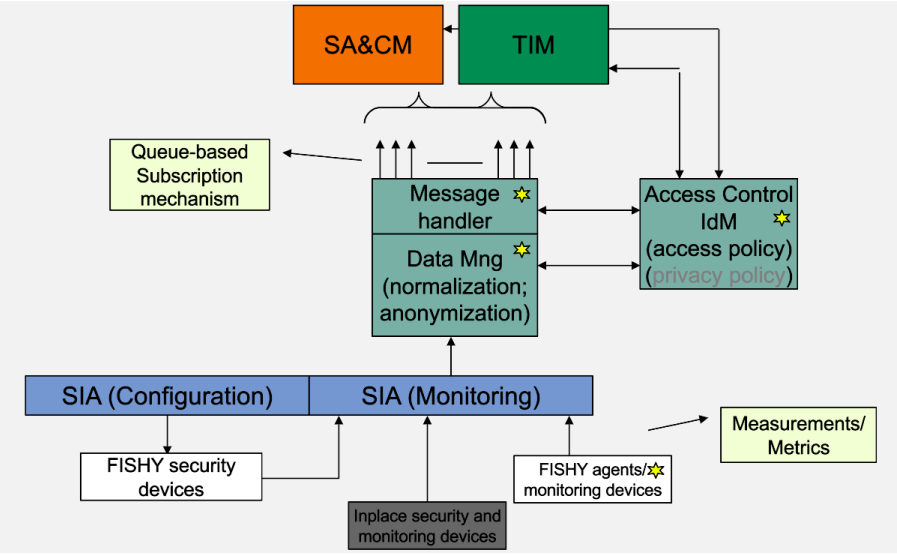

Figure 1- SPI architecture

As an intermediate component, SPI receives (or gets) raw data from the infrastructure – we will call this the downlink – normalizes its representation since data sources are expected to be very variable, applies access control and privacy rules, and makes it available for upper-level FISHY components – we will call this the uplink. Each of these functions will be detailed in dedicated subsections bellow.

Together with the SPI, a taxonomy of metrics is being developed. A taxonomy is used as it is the preferred method for data classification in hierarchical structure [1]. To elaborate a taxonomy, it is

necessary a systematic view about the systems / organization, and also a detailed description of its components. As described by Savola in its work on taxonomy of security metrics [2], organizational or management metrics are linked to attributes of organizational programs and processes, technical metrics are related to software and hardware artefacts, and operational metrics applicable to systems running in their environments [3]. The process of elaborating a taxonomy of information security metrics can become very hard and delicate, as it is necessary to understand the level of maturity of the organization and the existence of guides or processes that have already been defined.

In this context, the taxonomy under development follows the model recommended in NIST SP 800-55, based on the three main classes considered: Operational, Technical, and Organizational. For FISHY, there is a need for subcategories and a greater granularity of metrics, due to the complexity and heterogeneity of the project. Furthermore, by their specific functions, it is expected both SCM and TIM require diverse metrics classes.

2.1 Data Management

Within the Data Management block, adaptation elements act as transformer functions, mapping events, and raw data received from different systems in the premises and with different formats, as mentioned before, into a normalized format. Security event data, provided by security devices, is typically used for identifying threats, evaluating risks, and determining the impact of malicious actions (e.g., attacks). Benign actions (e.g., performance indicators) may also be used to evaluate the behaviour of the system and also the impact of any type of operations (including malicious ones). This data is often extracted from log files, and the log format must be lightweight and capable of encapsulating all network and host logs of large data sets.

Taking into account the variety of data formats and standards that may exist among the events generated by the various systems that will be operating in FISHY, and after analysing the main data standards, the use of the Common Event Format (CEF) was suggested. CEF is already used by several monitoring and security devices, works with key/value arrangement, and brings some important advantages for the project, such as the ease of incorporating JSON/JWT and implementations over Syslog, if necessary. The format is quite expressive and can incorporate information about sensors or devices, attack origin and destination, time, files, process and even users.

The CEF has in its structure a key/value representation mechanism, based on a dictionary provided by the developers. The generic format is:

- CEF host date/time: Version | Device Supplier | Device Product | Device Version | Device Event Class ID | Name | Gravity | [Extension]

where ‘Extension’ denotes a sequence of key-value elements, as shown in this example:

- 2 May 04:16:11 localhost CEF: 0 | nxlog.org | nxlog | 2.7.1243 | Executable code detected | Advanced exploit detected | 100 | src = 192.168.255,110 spt = 46117 dst = 172.25.212.204 dpt = 80

As highlighted before, for Syslog and similar data sources, adjusting the format will be very simple. However, for other specific monitoring systems (adopted or developed in the project) it may be necessary to add some header information pertaining FISHY framework. But the Extension mechanism should be flexible enough to support all core data. In some cases, we envisage the possibility of correlating measurements to provide more significant metrics – this will be addressed in a case-by-case basis. As far as the data segmentation is concerned, it will be performed by a Message Broker (discussed below) with the capacity to organize data following the taxonomy of metrics under development.

2.2 Low-level interface (Downlink)

The downlink part of SPI must be capable of supporting very different systems, subsystems, or components. To this end, we need to find common definitions to facilitate the proper interface and to help specify correctly the API requirements. After analysing the different case studies and the available documentation, we end up categorizing the low-level components into the three following scenarios:

- Black box (Bbox) Scenario: this scenario pertains those components from which no direct security related data can be used. Those components were developed without security in mind and do not implement any type of security control mechanism. Even so, their behaviour, or data patterns generated by their functions can be linked somehow to security objectives. Monitoring tools such as Nagios or Prometheus, can be deployed to capture the relevant metrics and make them available. Otherwise, equivalent components can be developed within FISHY, mainly when dealing with specific data and simple environments. The main sources of data will be network devices and segments, as well as logs related to computational elements performing any type of function.
- White box (Wbox) Scenario: in this scenario we mostly deal with components developed either in house or based on open-source code. Therefore, it is possible to adapt the system making it generate the data that FISHY will analyse. This scenario potentiates a better utilization of the FISHY resources, but it will not be the most frequent scenario, since in most cases industries use proprietary technological solutions. Within FISHY, a particular pair of modules fits the Wbox scenario (SEN / NED), since its key functionality demands for a deeper system integration with the capacity to generate security related functions at either the system software or hardware level.
- Gray box (Gbox): in cases where both previous scenarios coexist.

It is also important to differentiate low-level devices by their function type, according to three dimensions:

- Asynchronous and synchronous;
- Passive or active, in the sense of the way the low-level device handles the data event. Active devices are those capable of trigger a data transaction, in contrast with the passive ones which work by polling; and
- Time constrained, in particular if there are real-time requirements.

2.3 Up-level interface (Uplink)

The uplink block is responsible for making available security related data in an organized way, to both TIM and SCM (in WP4). As such, it will use a message broker mechanism using the taxonomy for metrics under development to organize dedicated data queues. According to the discussions already taken within the FISHY technical teams, and the system design adopted with legacy systems, a REST architecture style should be chosen whenever possible. Among the available solutions, RabbitMQ is a well-known technology that fulfils the requirements imposed.

2.4 Purposed technologies

For the Access Control and Identity Manager, the selected tool is Keycloak, an open-source software that fulfils the specific requirements, including the access control from devices, users, and clients. It deploys the well-known protocols OpenId Connect + OAuth2 and its integration with most FISHY components should be almost immediate.

For the message broker module, as already mentioned, it is proposed to use the RabbitMQ, also an open-source software very flexible and efficient to implement a message queuing protocol.

Concerning the Access Policies (not included in IT-1), we are proposing a hybrid model, where we will use the XACML standard as the basis, in addition to an authentication process based on JSON Web Token. JSON is characterized as a lightweight and easy to work format, that should be integrated with XACML to provide a standardized interface between PEP and PDP structuring the request and response task.

3 TIM Integrated outcome

TIM (Trust & Incident Manager) is the FISHY component that performs the analysis of metrics collected by the lower-lying FISHY components, such as the monitoring module of SIA and the metrics-gathering and processing tools. TIM is responsible for determining the vulnerabilities, detecting attacks and/or incidents and generating mitigating actions for the purpose of hardening the cybersecurity level of a monitored infrastructure.

TIM is composed of several modules that facilitate its functionality, listed and briefly described below:

- **Vulnerability Assessment,** automated vulnerability and risk analysis, estimation and detection;
- **Incident detection,** detection of anomalous events from data gathered by monitoring, supported by ML methods;
- **Impact Assessment**, inferring the scope of possible damage (data loss/theft, service downtime…) posed by the vulnerabilities and incidents detected by the previous components;
- **Mitigation,** providing actions and recommendations for cybersecurity hardening and minimizing the scope of incidents;
- **Central Repository,** storage for both incoming monitoring metrics and results of TIM analysis tools. By using a pub/sub layer over storage, the Threat/Attack Repository allows responsible components to have immediate access to new data for analysis and also serves as the integration point between TIM and other architectural blocks within FISHY;
- **Smart Contract,** manages service-level agreements and notifies stakeholders in case of violations and ensures the temporal succession of stored events.

In order to jump-start the technical development of the TIM component, a Proof-of-Concept (PoC) testbed has been created. It is used to apply the architectural principles defined in WP2, investigate communication channels and protocols most suited for each segment of the FISHY platform and serve as a starting point for integration with the metrics-gathering tools, task T3.3, Integrated Trust Manager, and task T5.3, Cross-functional platform integration into the FISHY PoC. The PoC Testbed is described in further detail in section 5.2.

4 TM Global Integration

4.1 Functionalities to be integrated

The TM (Trust Manager) is one of the main blocks of the FISHY architecture. The TM is divided into two blocks, TIM and SPI. The TIM consists of different modules responsible for providing the security of the stakeholder's devices and systems. For IT-1 four modules from TIM are selected to be integrated, as shown in Table 1 . The main TIM functionality is to create different security related messages which can be accessed by IRO and further components in WP4 and WP5. The second component, SPI, is responsible for collecting and storing the data generated by the different FISHY agents, processes and components from the ICT systems in the use cases infrastructure. For IT-1, there are four modules working in SPI; each one is responsible for performing different functionalities, as also shown in Table 1. Table 1 represents the summary of the internal blocks, modules and tools of TM to be integrated in IT-1.

Table 1: Summary of the internal architectural blocks and modules in TM

Block	Requirement	Modules	Tools
SPI	Identity Management/ Access Control	Identity Manager	Keycloak
	Access Control/Privacy enforcement	Access Policy	XACML
	(Low-level raw) Data Management	Data Management / Adaptation	RabbitMQ
	(Low-level raw) Data Management	Data Management /anonymization	Transformational data module (to be developed, along with the metrics' taxonomy)
TIM	Vulnerability assessment	Vulnerability assessment	Wazuh
	Incident Detection	Incident Detection	XL-SIEM
	Mitigation	Mitigation	PMEM
		Central Repository	Relational database Pub/sub

The TM integration done in IT-1 demands for a communication mechanism between SPI and TIM modules built upon the RabbitMQ protocol, as detailed in D3.1 [4]. Moreover, also for the IT-1 FISHY delivery, three tools (Wazuh, XL-SIEM and PMEM) are implemented in TIM, requiring two FISHY agents

to be deployed in the infrastructure; these FISHY agents send log files, assisted by the SIA functionalities, to Data Management in SPI, as described in next section.

4.2 TM Workflow

The workflow of the TM components communication and interconnection is discussed next and also shown in Figure 2. As an architectural decision, WP3 partners decided that the tools working in TIM, needs to get access to the log files and data only from a Threat/Attack repository. This Threat/Attack repository will be in a future integrated in a Central repository, which will be part of the whole FISHY deployment[1]. In step (4) in Figure 2, the FISHY agents working right below SIA, collect the data from the use case infrastructure and it is forwarded to the SIA. This collected data through SIA will be passed to both the Data management and the Message Handler working in the SPI using (5). The Data management block converts these logs files into the Common Event Format (CEF) and also handles an anonymization mechanism. The message Handler, in collaboration with the access control mechanism, is responsible for delivering these raw log files to the Central repository using (6). The data will be stored in the repository using the Pub/Sub or REST API mechanism. Then, after storing data in the repository, it will be available to the TIM tools, as well as to other FISHY modules outside TM.

However, in a first approach and for IT-1 implementation, the workflow has been simplified. This TM simplified workflow works as follows; only steps 1,2 and 3 are considered, where the FISHY agents are sending data directly to the tools working in the TIM. The detailed discussion is described as follows:

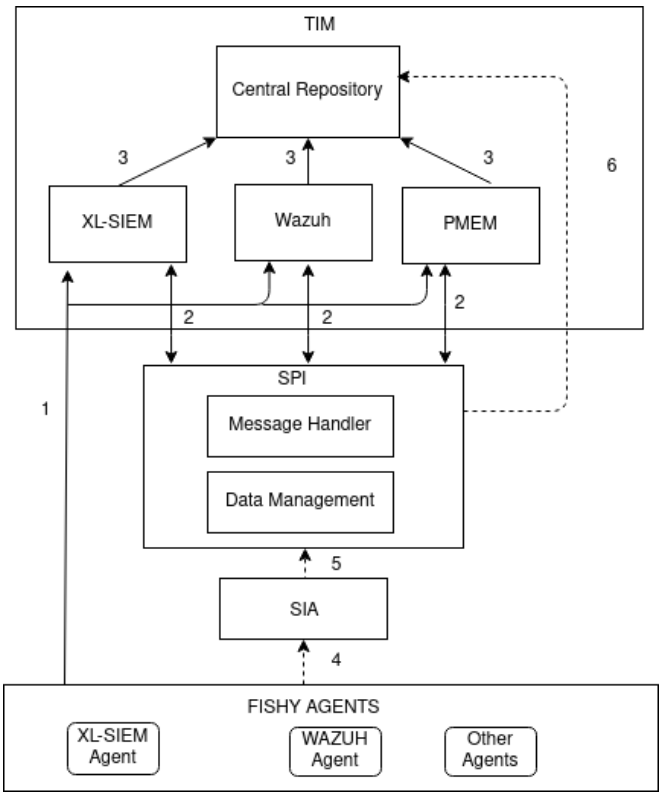

Figure 2. Flow Chart of TM Communication

[1] In this document, we use indistinctly Threat/Attack repository and Central Repository.

1. Data Gathering:

In IT-1, three tools within the TIM should read the log files (and raw data) coming from the use cases infrastructure. In the simplified workflow for IT-1, tools are directly gathering the log files from the FISHY agents. Currently, two FISHY monitoring agents are deployed in the infrastructure. Indeed, Wazuh and XL-SIEM tools have their own agents, while PMEM is currently using the logs files collected by these agents. Thus, these FISHY agents directly forward the data to the tools working in the TIM.

2. Authentication:

Respective tools working in the TIM needs to access the Central repository to store the log files and their output for future analysis. The request to access the repository is forwarded to the SPI. The Access control and Identity manager module are responsible for handing this request using a specific tool named Keycloak. Each tool in TIM needs to first register with the Keycloak. An access token is provided to each tool which latter is to be used for checking the authentication and authorization of the tool to write the data in Central repository.

3. Storing Output:

In addition to the Central repository, three main modules are envisioned in IT-1 operating in TIM, aimed at producing security related messages. These modules, i.e., Vulnerability assessment, Incident detection and Mitigation module, leverage on different tools: Wazuh, XL-SIEM and PMEM respectively. Each module, after getting raw files from the infrastructure and access token from SPI, processes it and stores the output in the Central repository. These outputs are available to other FISHY modules.

5 Testing and validation objectives

5.1 SPI functionalities

Taking into account the features and ideas mentioned in section 3.1, a prototype for the SPI module was proposed and tested through a demo to exemplify the interconnections between all modules. The main objective at this point was to create a simulation of events that could exemplify what would happen in a real case scenario. The simulation should respect the SPI architecture and exercise the essential features, namely Data Adaptation, Access Control, Identity manager, and Data Anonymization. It is important to mention that the proposed solution was developed in a test scenario without a proper connection to a real system. That was possible using some data files (logs) extracted from the use cases of the FISHY project to simulate a client and feed the demo, which in future could be an embedded process with a monitoring module, such as Prometheus or Nagios.

From a higher-level perspective, the test was carried out through a client that generates formatted data from the log files. The client fetches the data (log files) and performs its normalization, consisting on the conversion into CEF format. After this step, the client performs its authentication into the OpenID server (deployed by Keycloak) and finally sends the data to the RabbitMQ.

The RabbitMQ, serves as a message broker and facilitates the data organization between the publisher and the consumer component of the SPI. Also, it serves the purpose of providing important insights into the system performance giving information to be analyzed through the security metrics, namely the latency of messages sent by the client and consumed by RabbitMQ. Figure 3 shows the system testing architecture, including the sequence of operation steps performed, as explained next.

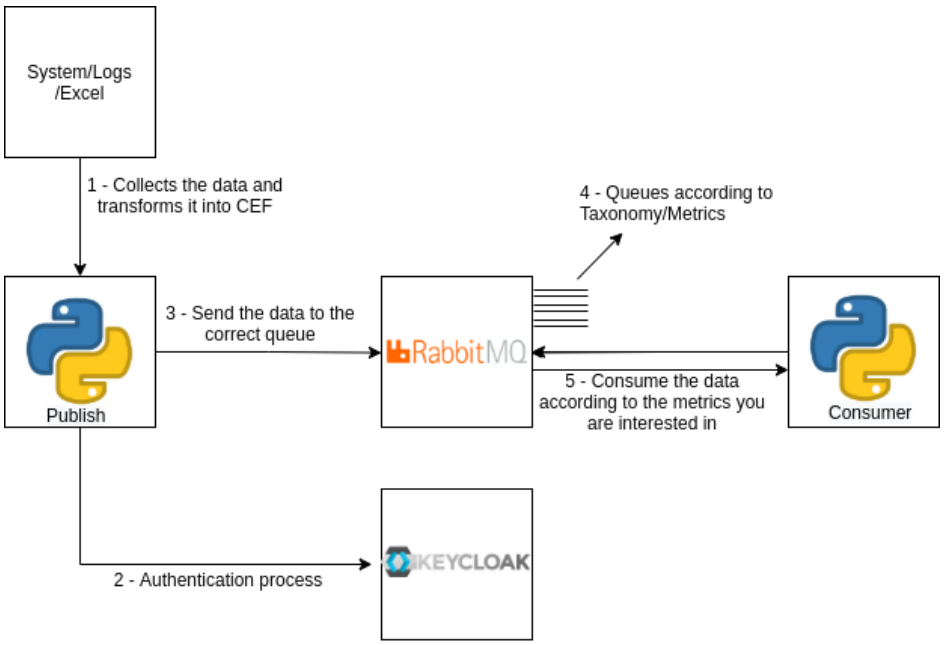

Figure 3: SPI testing Architecture

1. Collects the data and transforms it into CEF

Reads raw data from logs files and performs data normalization using the CEF format.

2. Authentication Process

Requests an access token to Keycloak, sending authentication information. The protocol used is the OpenID and OAuth2, **through the implicit flow**.

3. Send the data to the correct queue

In order to send the data to the correct queues, a dictionary was created and, at this moment, all the metrics are mapped into the taxonomy, thus creating a pattern for the name of the queues (for example, Tx.tec.performance.protocols.NETWORK), so that each metric can go to the correct queue.

4. Queues according to Taxonomy/Metrics

The message-broker (RabbitMQ) exposes the necessary queues to accommodate all the metrics. The queues' names are defined by each metric's class, as described in step 3; following the previous example, one of the exposed queues is Tx.tec.performance.protocols.NETWORK.

5. Consume the data according to the required metrics

To guarantee the effectiveness of the test, we run a consumer to simulate the interface with TIM and SCM, accessing the message broker (for testing purposes, we used only one queue named Tx.tec.performance.protocols.NETWORK). Besides, within the testing environment, the simulated consumer does not perform authentication, which a real FISHY component must do.

5.2 TIM functionalities

The starting point of development of TIM is the creation of a PoC testbed, that uses a combination of mock-ups and initial versions of components to simulate a deployment of the FISHY platform in a use-case environment and the flow of information from the monitored infrastructure to the upper-level services that analyzes, stores and displays the data.

Currently, the PoC includes: i) a mock-up of a webUI frontend (in place of the IRO Dashboard component); ii) an initial version of the Threat/Attack Repository, that already has database connectivity and a Pub/Sub layer implemented by using RabbitMQ, and; iii) an initial version of the FISHY Appliance Agent, capable of deploying Wazuh server and agents on-demand (via an HTTP REST request) using Ansible scripts, while also serving as a data forwarder for the events and alerts generated by Wazuh, which has been achieved by leveraging Wazuh's support for custom integration scripts. The custom integration scripts allow Wazuh to forward data to another source, while also providing the ability to specify the type and severity of events that should be forwarded.

A similar approach will be taken when integrating other tools into the platform, as having a single point of integration (namely the FISHY Appliance Agent) provides greater security and easier development of authentication mechanisms with the rest of FISHY.

The advantages of the Threat/Attack Repository's Pub/Sub layer are demonstrated in the PoC by a websocket server that consumes notifications of new data from RabbitMQ and alerts the webUI to refresh the displayed data without the need for users to manually refresh their browsers.

Figure 4 shows the deployment diagram, the connections and protocols used in the TIM PoC Testbed. Described below are the numbered connections and their associated functions:

1 - Trigger Ansible deployment of Wazuh, check deployment status
2 - Get Wazuh reports
3 - Push notification for new Wazuh report to RabbitMQ
4 - Subscriber receives notification via RabbitMQ
5 - Websocket notification
6 - Wazuh agent forwards report to Manager
7 - Wazuh Manager forwards report to FISHY Agent
8 - FISHY Agent stores report in Central Repository

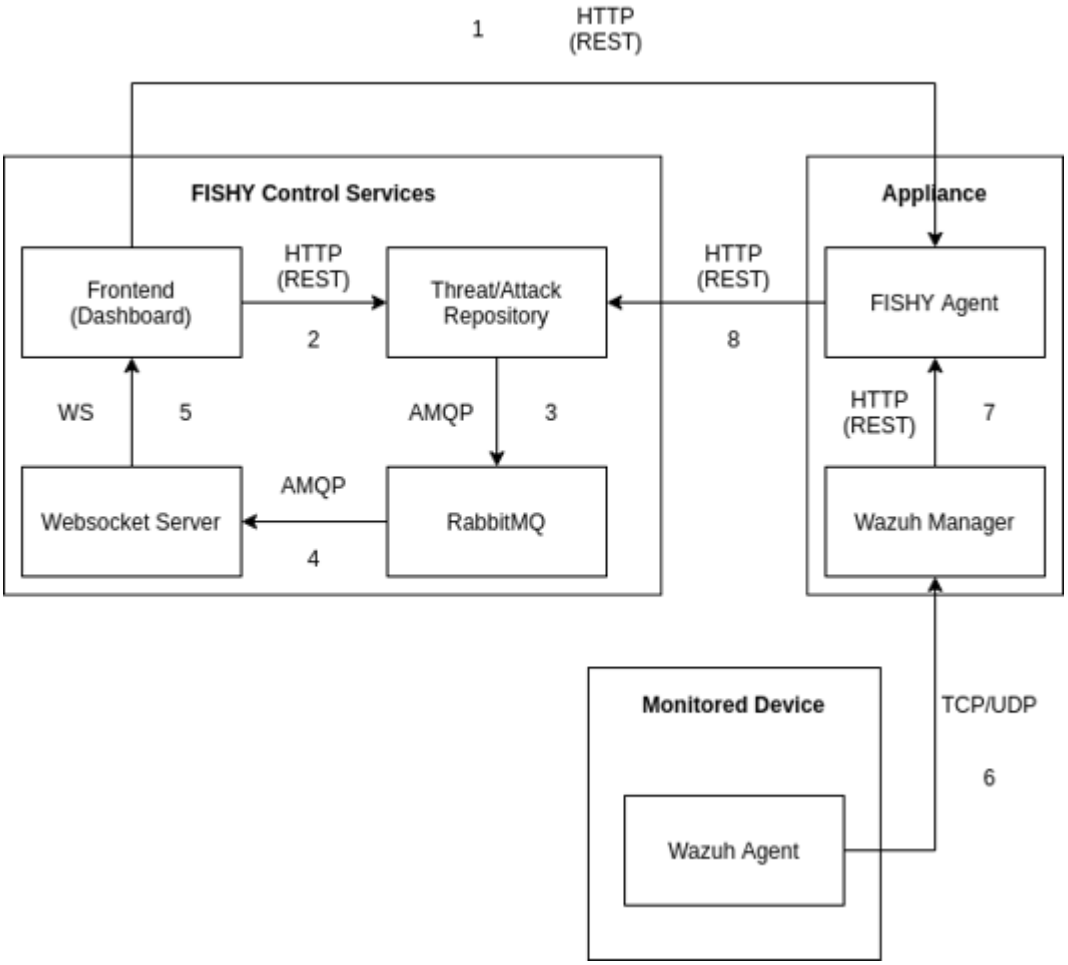

Figure 4 - TIM PoC Testbed

During task T3.3, the biggest emphasis will be placed on connections 1 and 8, between the FISHY Control Services and the Appliance, as they are the ones that must use SPI as their mode of transport. The requests to deploy tools and check the installation progress must be properly authenticated and authorized, while the gathered metrics must be normalized and anonymised before being transferred and stored in the Central Repository.

The implementation process is also guided by the communication between the technical and use-case project partners and identifying the requirements that will give the most value to the IT-1 FISHY prototype. In particular, talks with the F2F use-case partner, Synelixis, led to the development of a RabbitMQ consumer able to forward the received data via syslog, which is used by multiple metrics-gathering tools, and the implementation of custom Wazuh rules that address the more specific events and threats expected in their environment. The custom rule implementation is particularly useful, as it offers an excellent jumping-off point for the development of an intent-based, automatic rule generation functionality, targeted for Iteration 2.

5.3 Metrics and KPIs for TM

Metrics and Key-Performance Indicators have a special purpose in the SPI component of the Fishy Project. For example, in the authentication component through Keycloak they will be used to validate all processes. The use of metrics to analyze the information produced in this component can improve the security of all systems, distinguishing correct authentications from incorrect ones.

In this particular case, when a client tries to authenticate four different scenarios may come up.

- The first and the simplest one is a valid user that correctly authenticates into the system.
- The second one reflects the incapacity of a valid user to be authenticated into the system. This one can reflect the case of wrong behavior of the authenticator component or a breach of security by the client side.
- The third one reflects the capacity of an invalid user to be correctly authenticated into the system. This one reflects a serious breach of security by the system that grants access to an incorrect user.
- The last one reflects the capacity of the system to reject the authentication of an invalid user. This is a security measure that should be always active and put in place to distinguish the users that should be granted access to those who do not.

Another component that serves from a KPI is the RabbitMQ. As said before, RabbitMQ is a message broker that has the capacity to measure the latency of messages. This KPI can be considered as an indicator of performance. Indeed, an analysis of the measured latency can indicate the delay that the SPI module adds to the system.

Regarding project-specific KPIs for TIM, we can define one as the number of use-cases TIM can support with both custom-developed and out-of-the-box rulesets. For IT-1, the goal is to develop TIM to the point where it can analyse the traffic from Synelixis, the farm-to-fork use-case provider. The data that will be gathered from their infrastructure comes from PFSense, which Wazuh already supports, and also custom events from their SOFIE platform, which required custom implementation of rules. The custom-implemented rules are able to detect the following events:

- Unauthorized device, wallet ID level
- Unauthorized device, DID level
- Unauthorized user
- Attack on Blockchain node

Also for TIM, a very useful metric in TIM is the accuracy of predictions. Currently, one tool, referred to as PMEM is using machine learning approaches to predict the security attacks in terms of accuracy. Accuracy can be considered as a valuable metric to determine the specific functionality mitigation performed by TIM. For a particular use case, we can classify the threats into three different levels based on the accuracy, namely Normal, Threat and Attack.

Regarding these two selected preliminary metrics, delay and accuracy, different KPIs can be defined; although this is still a work in progress.

5.4 Description of validation Strategy

Figure 3 and Figure 4 show the test pattern designed for testing the functionalities of SPI and TIM separately. In order to design the validation strategy for the whole TM, the combination of both tests will be as follows. The test flow of SPI, Step 1 from Figure 3, which **Collects the data and transforms it into CEF,** can be connected to the Step 8 of TIM's test (Figure 4), where FISHY agents reports are stored at the repository using the SPI Step 5. The rest of the flow in both test cases remains the same for testing TM functionality. Thus, after combining these two test flows in a single flow, we can test the whole functionality of the TM.

The TIM-specific functionalities will also be validated by processing data from the actual infrastructure of the farm-to-fork use-case partner, Synelixis. The main objective of this validation step will be the confirmation of the ability of TIM to successfully detect the custom-defined events, that are not supported out-of-the-box by the integrated tools.

6 Conclusions

This deliverable provides the description of the initial integration of all the blocks in the Trust Manager (TM) module. First of all, the current implementation of the main blocks (SPI and TIM) in TM for IT-1 has been presented, as well as the tools used to implement these blocks.

From these implementations of SPI and TIM, the integration of both blocks has been proposed by means of a workflow. This workflow represents the integration of TM for IT-1, however the next steps towards the integration in IT-2 have been already drafted.

The main idea of the proposed integration for IT-1, is on one hand, that the FISHY agents in the infrastructure directly feed the blocks in TIM (and then their corresponding tools); however, the output of these tools will be stored in a Central Repository, to be available to other components outside TM in FISHY. These tools must request access to be able to write the output in the Central Repository; this access control is guaranteed by means of the SPI block.

The validation of the SPI and TIM prototypes for IT-1 has also been described, as well as the testing and validation strategy, of both, individual blocks and TM.

Finally, in this deliverable different metrics to evaluate each of the modules have been proposed, mainly based on accuracy and latency. In next deliverables, the KPIs associated to these metrics will be presented.

References

[1] Savola R. (2009), *A Security Metrics Taxonomization Model for Security-Intensive Systems* Journal of Information Processing Systems, pp. 197, Dec 2009

[2] Savola R. (2013), *Quality of security metrics and measurements*, Comput. Secur., vol. 37, pp. 78–90, Sep. 2013, doi: 10.1016/j.cose.2013.05.002.

[3] Morrison P., Moye D., Pandita R., and Williams L. (2018), *Mapping the field of software life cycle security metrics*, Information and Software Technology, vol. 102. Elsevier B.V., pp. 146–159, Oct. 2018, doi: 10.1016/j.infsof.2018.05.011.)

[4] [FISHY] - *D3.1 Trust Manager components design and implementation. (IT-1)* Diego López, Antonio Pastor, Luis Conteras. 2021.