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

D3.3 Trust Manager components design and implementation (IT-2)

Document Identification			
Status	Final	**Due Date**	31/10/2022
Version	1.0	**Submission Date**	07/11/2022

Related WP	WP3	**Document Reference**	D3.3
Related Deliverable(s)	D3.1, D3.2, D4.1, D4.2	**Dissemination Level (*)**	PU
Lead Participant	UMinho	**Lead Author**	Henrique Santos (UMinho) André Oliveira (UMinho)
Contributors	UPC, UPC, TID, SYN, POLITO, UC3M, XLAB	**Reviewers**	Eva Marin-Tordera (UPC)
			Jose M. Manjón (TID)

Keywords:
Integration, SPI workflow, TM workflow, validation strategy

Document Information

List of Contributors	
Name	Partner
Henrique Santos	UMinho
André Oliveira	UMinho
Pedro Magalhães	UMinho
Diego R. López	TID
Jose M. Manjón	TID
Xavi Masip-Bruin	UPC
Eva Marín-Tordera	UPC
Nelly Leligou	SYN
Cataldo Basili	POLITO
Daniele Canavese	POLITO
Silvia Sisinni	POLITO
Luis González	UC3M
Guillermo Yuste	ATOS
Antonio Álvarez	ATOS
Jan Antic	XLAB

Document History			
Version	Date	Change editors	Changes
0.01	01/08/2022	André Oliveira (UMinho)	ToC and Initial Structure of the Document
0.02	15/09/2022	Eva Marín (UPC)	First version of Section2
0.03	26/09/2022	André Oliveira (UMinho)	UMinho Contribution
0.03	28/09/2022	Pedro Magalhães (UMinho)	Add 4.6 and 5.3 contribution
0.04	03/09/2022	Nelly Leligou (SYN)	Added the First version of Section 4.7
0.05	04/09/2022	Ayaz (UPC)	Added the First version of Section 4.4
0.06	09/10/2022	Daniele Canavese, Silvia Sisinni (POLITO)	POLITO contribution
0.07	10/10/2022	André Oliveira (UMinho)	Adjusting sections and contributions to the document

0.08	14/10/2022	Luis Gonzalez (U3CM)	Added the First version of Section 5.1
0.09	14/10/2022	Guillermo Yuste (ATOS)	Added the First version of Section 4.1 and 4.3
0.091	17/10/2022	André Oliveira (UMinho)	Review of the Document
	24/10/2022	Antonio Álvarez (ATOS)	Refinement of the Document
0.095	24/10/2022	Jan Antic (XLAB)	Added the First version of Section 4; 4.2; 4.8; 4.9; 4.10 and 5.3
0.1	24/10/2022	André Oliveira (UMinho)	D3.3 First Version Completed and sent for review
0.11	26/10/2022	Eva Marín (UPC)	Review of D3.3
0.12	31/10/2022	André Oliveira (UMinho)	D3.3 Content and Style Review comments addressed
0.20	03/11/2022	André Oliveira (UMinho)	D3.3 Second Version Completed and sent for review
0.21	04/11/2022	Eva Marín (UPC)	Review of D3.3
0.22	04/11/2022	Jose M. Manjón (TID)	Review of D3.3
0.23	04/11/2022	André Oliveira (UMinho)	D3.3 Content and Style Review comments addressed
0.3	04/11/2022	André Oliveira (UMinho)	D3.3 Third Version Completed
1.0	07/11/2022	Antonio Álvarez, Juan Alonso (ATOS)	Quality assessment and final version to be submitted.

Quality Control		
Role	Who (Partner short name)	Approval Date
Deliverable leader	Henrique Santos (UMinho)	04/11/2022
Quality manager	Juan Alonso (ATOS)	07/11/2022
Project Coordinator	Antonio Álvarez (ATOS)	07/11/2022

Table of Contents

Document Information 2
Table of Contents 5
List of Tables 6
List of Figures 7
List of Acronyms 8
Executive Summary 10
1 Introduction 11
1.1 Purpose of the document 11
1.2 Relation to other project work 11
1.3 Structure of the document 11
1.4 Glossary adopted in this document 12
2 FISHY platform architecture 13
3 Security & Privacy Infrastructure Design 16
3.1 Identity Management and Access Policy 16
3.2 Data Management 17
3.2.1 Privacy enforcement 17
4 Trust & Incident Manager Design 18
4.1 XL-SIEM 18
4.2 Wazuh 19
4.3 RAE 20
4.4 PMEM 21
4.5 Trust Monitor 22
4.6 Zeek 26
4.7 Smart Contracts 26
4.8 Central Repository 29
4.9 VAT 30
4.10 LOMOS 30
5 Implementation in the Reference Framework 31
5.1 Introduction to FISHY Reference Framework 31
5.2 SPI Integration 31
5.3 TIM Integration 32
6 Conclusions 33
References 34

List of Tables

Table 1. TM functionalities *14*

List of Figures

Figure 1. Update of the FISHY architecture for IT-2 13
Figure 2. Architecture in IT-2 related to WP3 15
Figure 3. SPI Workflow 16
Figure 4. TIM tools workflow 18
Figure 5. Wood-based Panels Trusted Value-Chain XL-SIEM deployment 19
Figure 6. Securing Autonomous Driving Function XL-SIEM deployment 19
Figure 7. RAE workflow 21
Figure 8. ML Based Detection Module Overall Internal working 22
Figure 9. FISHY high-level architecture 23
Figure 10. Trust Monitor subcomponents 24
Figure 11. TM interaction workflow with the other components of the FISHY architecture ... 25
Figure 12. Smart contracts internal organisation and relation to the high-level architecture 26
Figure 13. Sequence Diagram for smart contracts component storing a detected security event in the F2F use case 28
Figure 14. Sequence Diagram for smart contracts component storing a policy to the blockchain in the F2F use case 29
Figure 15 - LOMOS internal log processing 30

List of Acronyms

Abbreviation / acronym	Description
AA	Attestation Agent
ABAC	Attribute-based access control
AC	Access control
API	Application Programming Interface
AS	Authorisation Server
AT	Access token
CA	Cyber Agent
CEF	Common Event Format
CVEs	Common Vulnerabilities and Exposures
D3.1	Deliverable number 1 belonging to WP3
D3.2	Deliverable number 2 belonging to WP3
D3.3	Deliverable number 3 belonging to WP3
DAPP	Distributed application
DI	Digital identity
DID	Distributed ID
EC	European Commission
EDC	Enforcement & Dynamic Configuration
F2F	Farm-to-Fork
FRF	FISHY Reference Framework
IBFT	Istanbul Byzantine Fault Tolerance
IDS	Intrusion Detection System
IdM	Identity management
IdAM	Identity and Access Management
IRO	Intent-based Resilience Orchestrator & Dashboard
JSON	JavaScript Object Notation
JWT	JSON Web Token
K8s	Kubernetes
LOMOS	Log Monitoring System
MAC	Mandatory access control
ML	Machine Learning
NED	Network Edge Device
OIDC	OpenID Connect
OCSVM	One Class Support Vector Machine
P2P	Peer-to-Peer
RA	Remote Attestation

Abbreviation / acronym	Description
RAE	Risk Assessment Engine
RBAC	Role-based access control
REST	Representational State Transfer
SADE	Securing Autonomous Driving Function at the Edge
SC	Smart contract
SCM	Security Assurance & Certification Manager
SIA	Secure Infrastructure Abstraction
SIEM	Security and Information Event Management
SPI	Security & Privacy Data Space Infrastructure
TIM	Trust & Incident Manager
TM	Trust Manager
VAT	Vulnerability Assessment Tool
VM	Virtual Machine
WP	Work Package
WBP	Wood-Based Panels
XACML	eXtensible Access Control Markup Language
XL-SIEM	Cross-Layer Security Information and Event Management
XML	eXtensible Markup Language

Executive Summary

This deliverable focuses on the update of the deliverable D3.1[1], that belongs to third work package of the FISHY Project. It describes the design and implementation tasks of the Trust Manager (TM) module envisioned for the second iteration (IT-2) of the FISHY Project. In previous deliverables D3.1[1] and D3.2[2] the blocks in TM (Trust Manager) have already been designed and described, namely TIM and SPI, and details about the integration of these components were given.

Deliverable D3.3 presents details about the component design and implementation for second iteration (IT-2).

1 Introduction

1.1 Purpose of the document

This deliverable describes the design and implementation of the Trust Manager (TM) module for the second iteration of the FISHY Project. It addresses the characteristics of the two blocks identified within the TM, namely the Trust & Incident Manager (TIM) and the Security & Privacy Data Space Infrastructure (SPI), at the second stage of the Project development iteration.

After the description of the design and implementation of each one of the blocks in TM realized in D3.1 [1], and the characterization of all aspects related to the integration of the TM component alongside the other components of the FISHY architecture described in D3.2[2], it is essential to review the work performed and do some improvements for the second iteration. In this deliverable, we present and describe the characteristics of the SPI and TIM outcomes to be integrated, as well as the design choices adopted for the second iteration of the Project. It also outlined/drafted the validation strategy and test to be realized.

1.2 Relation to other project work

The design and architecture of the Trust Manager (TM) module had already started on the first iteration of the FISHY Project and is extensively described in Deliverable 3.1 [1]. Also, the integration aspects of the modules SPI and TIM are studied and presented in Deliverable 3.2[2]. All three tasks referent to this module, namely T3.1, T3.2, and T3.3, were addressed indifferently in this Deliverable, and therefore D3.3 is the first output in terms of reporting, of the Trust Manager module for the second iteration of the project, creating a path for the module final integration in the FISHY Platform. D3.3 marks the conclusion of T3.1 and T3.2, while T3.3 will run until the conclusion of WP3 in M30 (February 2023). T3.3 takes care of the local integration within the Trust Manager module. 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. Actually, in terms of timing WP3 and WP4 are totally symmetric and run fully in parallel. The output of both tasks/WPs will jointly feed the whole FISHY integration to be done in WP5 towards delivering the IT-2 version of the FISHY platform, to be documented in deliverable D5.2, due M32 (April 2023); 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 into 6 major chapters, being this one an initial introduction to the document and the remaining described as follows:

- **Chapter 2** presents the FISHY Platform Architecture for the IT-2 of the Project.
- **Chapter 3** presents the SPI design and architecture structure for the IT-2.
- **Chapter 4** presents the TIM design and architecture structure for the IT-2.
- **Chapter 5** presents integration into the Reference Framework of the Trust Manager Module, concerning IT-2 details.
- **Chapter 6** presents the conclusion thoughts of 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 FISHY platform architecture

According to the DoA and the deliverables D3.1[1] and D3.2[2], the TM (Trust Manager) component is divided into two blocks: TIM (Trust and Incident Manager) and SPI (Security and Privacy Data Space Infrastructure) [1,2]. In this deliverable, D3.3, the design and implementation of both blocks will be described in detail for IT-2. However, not only the internal improvements of each one of the components and its functionalities will be detailed, but also the main changes in the whole FISHY architecture impacting WP3.

After releasing the FISHY architecture for IT-1 in month 18, some improvements have been proposed for IT-2. The main updates in the FISHY architecture related to TM (WP3) are:

- Threat/attack repository becomes a Central repository and event-based messaging system. This change is mainly due to the need of having, on one hand, and a single repository where both output data from tools can be stored; on the other hand, it becomes the main way of communicating data among tools.
- SPI's modules Access Policy and Identity Manager becomes transversal to the whole architecture, as it was defined within IT-1. These modules deploy access control among all other architecture units.
- FISHY appliance to interact with the infrastructure. In order to smooth the deployment of the FISHY agents and their connection with the data collectors in the infrastructure, this new component has been proposed and added to the whole workflow in the architecture.
- FISHY Dashboard not only relates to IRO but the rest of FISHY tools. Although FISHY Dashboard does not belong to WP3 (but WP5), the architectural change is shown because it impacts on SPI-Identity Manager/Access Policy functionality. The GUI of all the TIM tools will authenticate through SPI to access the FISHY dashboard.

The matching between the requirement proposed by use cases providers and functionalities/modules in TM is shown in Table 1. Apart from these general architectural changes, at the TM internal level, new modules fulfilling new requirements or enforcing already considered requirements in IT-2 have

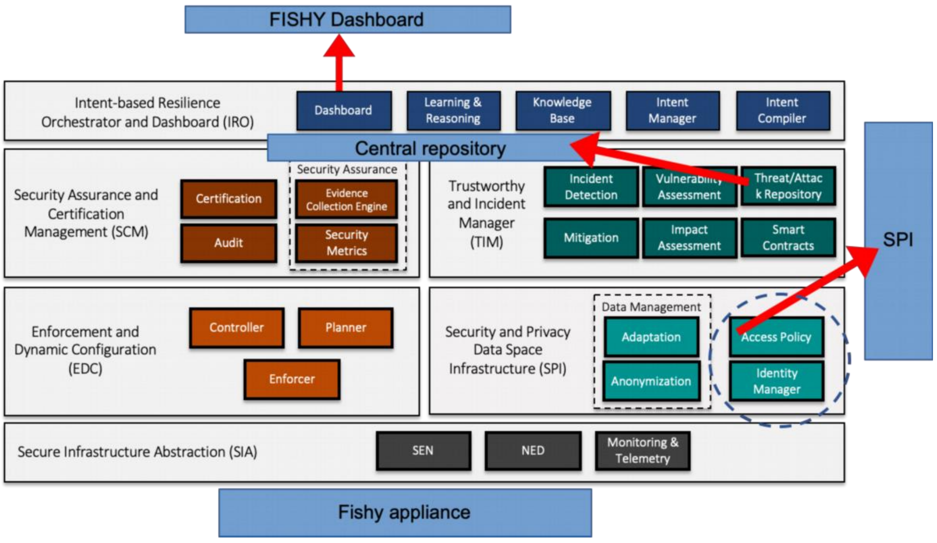

Figure 1. Update of the FISHY architecture for IT-2

been added. Specifically, Trust Monitor and Smart Contract modules have been designed and integrated in TIM for IT-2, the first one covering remote attestation functionality for integrity evaluation of the FISHY Appliance, and the latter covering trustworthy mechanisms and collaboration among different stakeholders. Also, the Zeek tool matches the network performance monitoring and anomaly detection requirements of the TIM module, also contributing to the development of security metrics.

Table 1. TM functionalities

Block	Functionality	Modules	Tools
SPI	Identity Management/ Privacy enforcement	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
TIM	Vulnerability assessment	Vulnerability assessment	Wazuh, VAT, LOMOS
	Incident Detection	Incident Detection	XL-SIEM, PMEM, Zeek (Network Monitoring)
	Mitigation	Mitigation	PMEM
	Prediction and estimation of risks	Prediction and estimation of risks	RAE
	Remote Attestation	Trust Monitor	TPM 2.0
	Trustworthy mechanisms and collaboration among stakeholders	Smart Contracts	Smart Contracts
	Extension/ Expansion scalability	Smart Contracts	Smart Contracts
	Global security events storage	Central Repository	Relational database Pub/Sub

Finally, in Figure 2, it is shown the detailed architecture for IT-2 of all the modules involved in Trust Manager (TM, WP3).

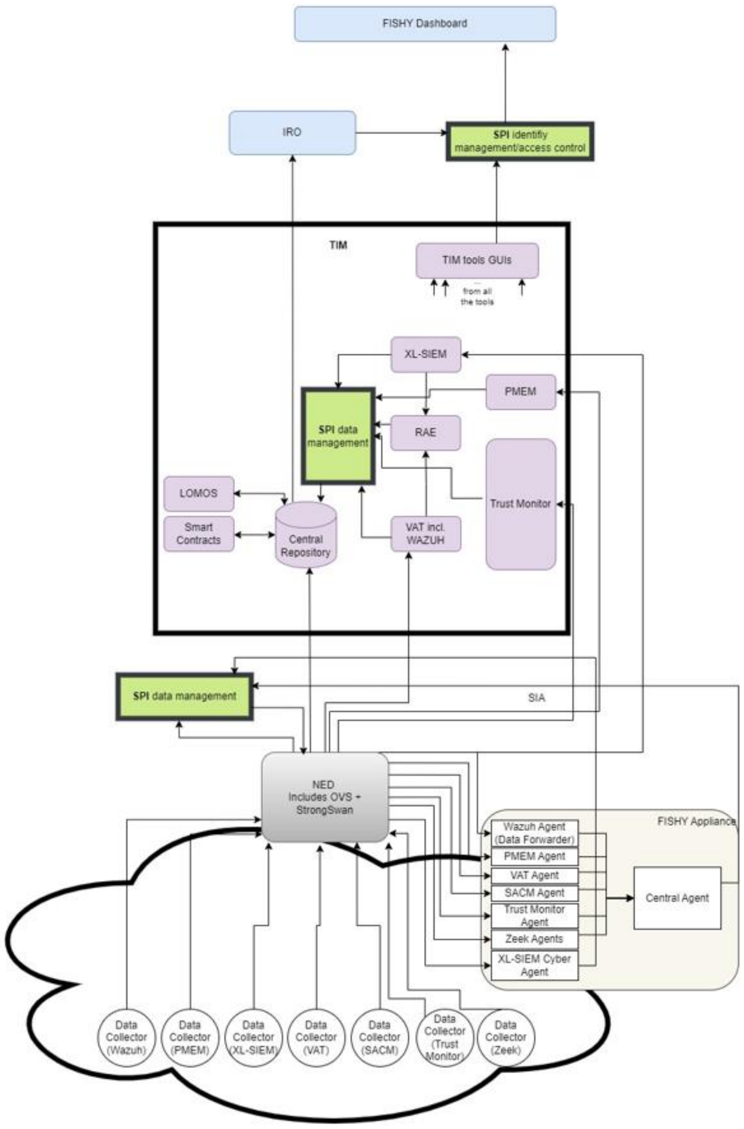

Figure 2. Architecture in IT-2 related to WP3

The general flow of data to the TIM tools is as follows, data collectors read data from the infrastructure and send it to their respective agents in the FISHY appliance, if necessary, the connection is provided by NED as shown in the figure. FISHY appliance sends this data to the SPI data management. This module may forward the data to each one of the TIM tools (and also raw data is stored in the Central repository) or can process the data to apply data formatting/anonymization or to apply privacy rules. The output from the SPI data management goes to each one of the TIM tools in the FISHY central services, again, if necessary, NED can provide the connectivity. Then data reach each one of the TIM tools, which process the data and writes the output to the Central repository. Depending on the tool this output can be alerts, events, logs, etc. Output data written in the Central Repository is available for other blocks in other work packages, such as IRO. Detailed workflows are shown in sections 3 and 4, Figure 3 and Figure 4.

Document name:	D3.3 Trust Manager components design and implementation (IT-2)					**Page:**	15 of 34
Reference:	D3.3	**Dissemination:**	PU	**Version:**	1.0	**Status:**	Final

3 Security & Privacy Infrastructure Design

In the second iteration of the FISHY Project, the Security & Privacy Infrastructure (SPI) module acquired a more well-defined role with tasks that required constant and all-around communication with the other modules of the project. As it was described in Deliverable 3.1, SPI has the purpose to provide an interface between low-level components and higher-level modules, in the sense that it manages the raw data acquired by the sensors placed in the infrastructure under study and drives them to be measured comparatively with the security metrics relevant and defined according to the security objectives of the company under study. Besides managing data collected, SPI also performs a key role on FISHY related to identity and access management to the platform. It is on SPI that is defined who should have access to the platform and the different roles that can access it and what could be done according to the role that the user has associated with.

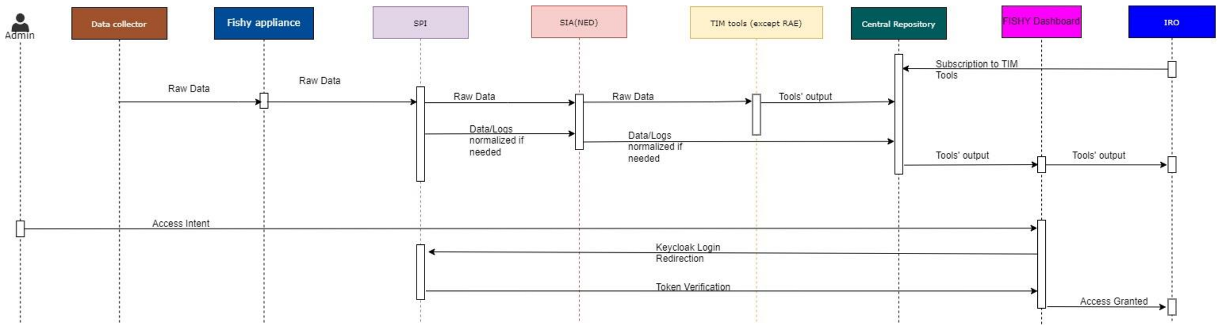

Figure 3. SPI Workflow

Figure 3 represents the SPI workflow and is representative of the main functions related to this module, namely identity management, access policy, and data management including privacy enforcement features (e.g., anonymization). As shown, there is provision for raw data normalization when necessary to format event data to a common representation that posteriorly is stored in a Central Repository or is processed by the TIM tools. Another workflow is associated with an access request triggered by the user and is managed by Keycloak system which perform authentication and authorization verification processes, granting or rejecting access accordingly to the policy rules defined.

3.1 Identity Management and Access Policy

The Identity and Access Management capabilities of FISHY are developed with the SPI module, which is the component responsible to coordinate and grant access to users who use all the other FISHY modules. In the second iteration of the Project, it is intended that this access control module extends its features to control the access to all the technologies and tools that are part of the FISHY Platform. As it was designed for the first iteration of the Project, the Access Control (AC) unit serves OpenID Connect (OIDC) based on RESTful Technology. This solution is designed to have a centralized authorisation server capable of implementing the OAuth2 standard, complemented by an authentication layer based on the OpenID Standard. OpenID Connect remains the implemented centralized authentication system, capable of requiring servers and clients, or tools and users, to possess a unique ID and shared key to access the platform. The protocol used specifies the mechanism to get Access Tokens (AT), to allow access to software services. The OpenID protocol is used with the intent to enforce authentication on the flow of information across the whole FISHY platform.

All data transactions across the platform are dependent on a correct Identity Management process, which in this case is performed through JWT (JSON Web Tokens) capable of providing authentic (signed) claims between entities and components.

Regarding access policies, these are being developed to define a set of conditions that should be satisfied to grant subject access to a desirable object, system, or information. During the development of this document, the access policies are mostly implemented following a role-based access control (RBAC), but with more refinement, it is possible that an attribute-based access control (ABAC) model would be also used. Following Deliverable 3.1[1], the Policy Architecture will be based on XACML, Extensible Access Control Markup Language, which is an open standard for access control architectures, responsible for the management of rights, evaluation, and enforcement of access policies.

3.2 Data Management

As seen on the SPI Workflow presented before, this module has the feature to receive raw data from the sensors or tools implemented in the premises of the organisation under study and transform them into normalized data. Is intended that the FISHY Platform supports the collection and sharing of a vast amount of data produced by different components and be capable of exchanging data within the elements of the FISHY architecture. To support this capability, it is essential to remember that it is supposed to exist a constant production of data in different formats and standards associated with the use of different tools. For that reason, is important to manage and adapt all data collected and transform those into a unique standard and form, such as the Common Event Format (CEF). The use of CEF was already proposed in Deliverable 3.1[1] of the FISHY Project and is characterised by being a format used by several monitoring tools and security devices. CEF works with key/value arrangement and their manipulation makes it easy to incorporate JSON/JWT and implementations over Syslog, whenever necessary.

3.2.1 Privacy enforcement

The privacy enforcement feature within the FISHY project is related to the implementation of security policies in XACML that can be able to guarantee the privacy of data and users accordingly to the security objectives and requirements defined by the object (organisation) under study. This privacy interface is also very intimate to the concepts of adaptation and anonymization presented in D3.1[1]. The adaptation feature will act as transformer, mapping the raw data received through the collectors implemented and the edge storing into the Central Repository. These will only include the necessary attributes to support both functional and non-functional system requirements. The anonymization concept is associated with a conversion of processed data to preserve the privacy of users and comply with regulatory requirements. Since data in a supply chain are intended to be transferred to more than one organisation it is important to secure that unnecessary information is shared with other organisations or entities, to reduce the risk of unauthorized disclosure of personal data.

4 Trust & Incident Manager Design

During the IT-1 phase of the project, the development of TIM focused on a minimum viable demonstrator that served as a validator of the architecture design and a proof-of-concept of the proposed data flow. The implementation and integration of a chosen tool, Wazuh, demonstrated the ability to consume data from a use case, forward it for analysis, store it in the Central Repository and send out a notification when an anomalous event is detected. IT-1 established a template for further developments of TIM in the IT-2 phase of the project, which now comprises the various modules and services offering holistic cybersecurity protection of the monitored infrastructures.

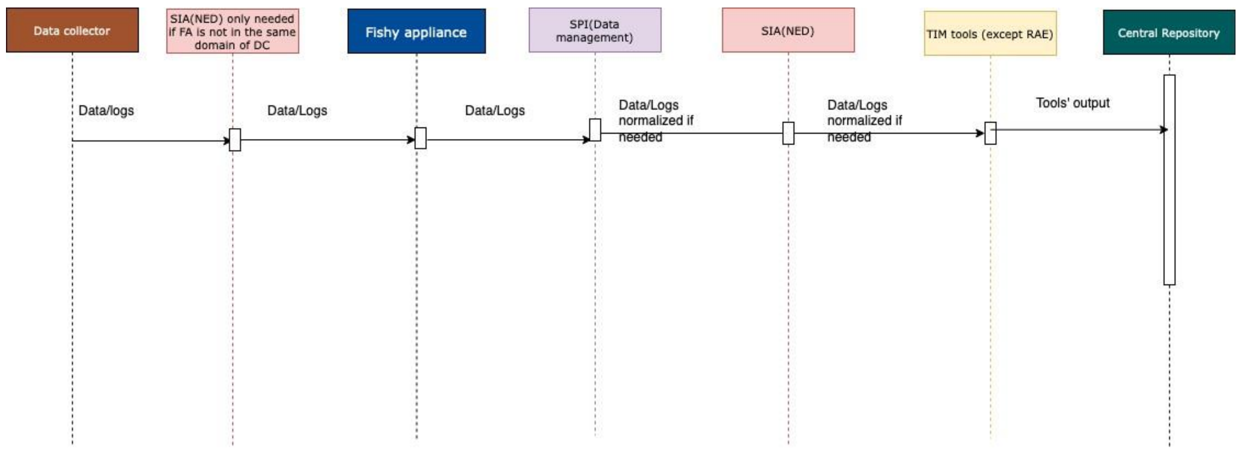

Figure 4. TIM tools workflow

Figure 4 presents the data flow of gathered metrics from the monitored infrastructure up to the platform, where they are analysed, correlated and the results are stored in the Central Repository, which, through its pub/sub mechanism, also allows instant notifications of new results. These notifications can be used by either other FISHY components for further analysis or informing system administrators of potentially malignant events.

This figure also includes Data Collectors as the first link in the chain. This concept was introduced during the IT-2 phase of the project. Data Collectors reside inside the monitored infrastructure and forward their collected data to tool agents residing on the FISHY Appliance, with SIA (NED) providing secure access to the underlying architecture.

4.1 XL-SIEM

The XL-SIEM is thoroughly described in D3.1 [1]. After this delivery, during the subsequent months, the XL-SIEM has significantly improved its capacities to better accommodate the FISHY use case needs, as described in D6.3[3].

The asset was improved both on the client and server sides. At the CyberAgent (CA), new sensors were fully integrated, making it possible to work with new data collectors such as IoT devices, RabbitMQ servers or Cisco controllers. Aside from that, during the deployment of the CA, we simplified some deployment steps.

At the server, new rules have been developed, increasing the tool correlation capacity, and improving the RAE calculation algorithms (more detail in section 4.3)

In the Wood-based Panels Trusted Value-Chain (WBP), the XL-SIEM and the CyberAgent (CA) were deployed in the SONAE infrastructure in a dedicated Virtual Machine (VM). It serves as a bridge between the XL-SIEM and SONAE's IoT Devices and CISCO controller. Aside from that, a new correlation rule that compares the same users' login in different servers has sign-in.

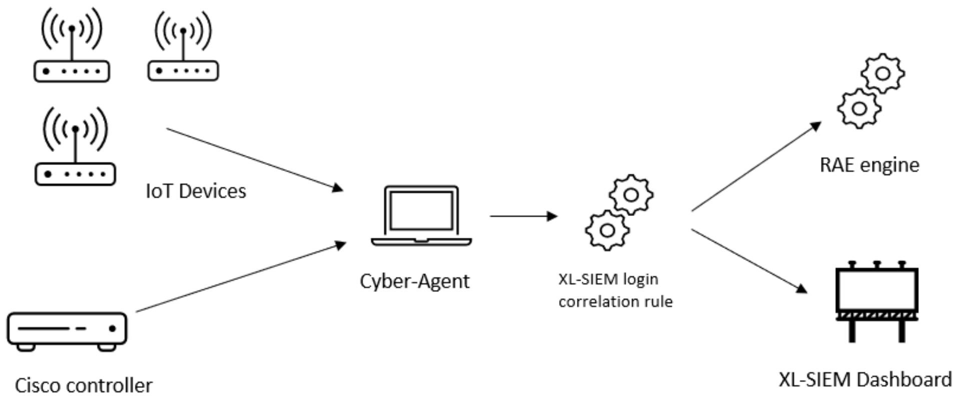

Figure 5. Wood-based Panels Trusted Value-Chain XL-SIEM deployment

For the Securing Autonomous Driving Function at the Edge (SADE) the CA was deployed together with the Kubernetes deployment and is receiving logs for the RabbitMQ and Nginx servers. We also worked on new rules to detect SQL-Injection attacks.

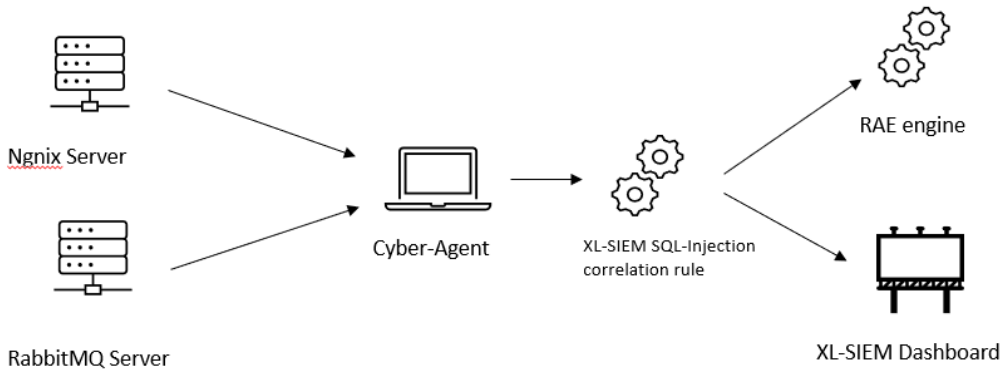

Figure 6. Securing Autonomous Driving Function XL-SIEM deployment

4.2 Wazuh

Wazuh has already been described in D3.1 [1]. During IT-2, developments of Wazuh are focused on integration into the platform architecture and improving its detection capabilities for threats specific to our use cases.

In its most common deployment mode, Wazuh uses agents deployed on devices to gather logs and monitor file integrity. This deployment mode already fits well with the established architecture of the

FISHY platform, as the data flow of Agent -> Receiver -> Server matches the FISHY concepts of Data Collector -> Appliance agent -> Server.

Wazuh also offers agentless monitoring, where the data collection process has to be provided by some other means. This deployment mode is usually used on devices like routers or switches, where agent cannot be installed, but it also provides an opportunity for a less intrusive deployment in production systems. For agentless data collection, Wazuh supports various protocols, such as SSH, WMI or NetFlow. During IT-2, support for data collection via RabbitMQ has been developed and used to retrieve data from the Farm-2-Fork use case (F2F).

We also worked on the implementation of new custom rules that can detect connection attempts of a device with an unauthorized DID (Distributed ID).

4.3 RAE

The Risk Assessment Engine (RAE) is a Python-implemented tool of ATOS that can perform risk assessment thanks to the information received from XL-SIEM, the Vulnerability Assessment Tool (VAT) and other tools.

The R-implemented risk assessment models that are part of RAE, are affected by several variables, such as target configurations, network topology, attack vector followed by the malicious agent, and indicators, that are a representation of some status, action or omission related to the cyber risk considered in the risk assessment model and vulnerabilities.

RAE internal workflow:

1. RAE receives information from the infrastructure from Wazuh and XL-SIEM using a Cyber Agent (CA).
2. This information updates the status of one or various indicators associated with one or various risk models.
3. The update of one or various indicators starts a new risk assessment evaluation, where all the indicators, that is, the status of the infrastructure and status of the attack, and asset and organization information are considered.
4. A Risk Report is generated that can be seen at the RAE user interface, with a global risk position with qualitative and quantitative values.
5. Some selection of information from the risk report is sent to the Central Repository in Common Event Format (CEF), as stated in Figure 7.

The events received from Wazuh and XL-SIEM are mapped to the indicators. A new risk assessment is automatically launched each time a status modification is detected for an indicator.

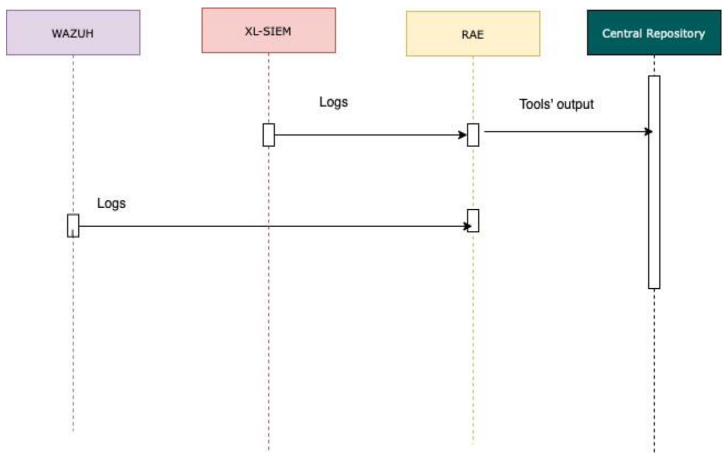

Figure 7. RAE workflow

4.4 PMEM

Currently, it is estimated that 65% of the attacks in a network are known. The access control, authentication mechanism and encryption algorithms work as a first line of defence against network cyber-attack. The Intrusion Detection Systems (IDS) works as the second line of defence against cyber-threat. PMEM can work as a second line of defence to detect anomalies in networks system. PMEM is an IDS based on Machine Learning approaches to detect known and unknown anomalies in a network system entry. Irregular network activity is considered an anomaly. New anomalies, not seen before, can appear anytime. Thus, an IDS needs to detect known and unknown anomalies. The workflow of the PMEM consists of the following steps, as already described in D3.1[1]:

Data Collection Agent: PMEM contains a data collection agent which captures the real network traces from the use case infrastructure. The network traces contain raw information captured from the main router of the organization which contains all the incoming and outgoing traffic within the network.

Feature Extractor: The features are extracted from the raw traffic for Machine learning (ML) model training.

API: An API is developed for getting these features to be used as Input for ML Model. These features will be forwarded to the PMEM tool with the help of the FISHY Appliance.

The previous three components are not shown in Figure 8 because they are part of the data collectors and FISHY appliance shown in Figure 2.

ML Based Detection Module: The prediction performed by the PMEM ML module workflow is shown in Figure 8. The first module is based on One Class Support Vector Machine (OCSVM). This module aims to classify normal and abnormal traffic. The entries that have not been classified as normal traffic are passed through supervised and an additional layer of OCSVM filter to classify the entries into a certain type of attacks (at least, 7 known different categories), unknown attack or normal. The predictions performed by PMEM will be stored in the central repository to be used by IRO and possible suggestions will be given for each type of attack which will be used by the IRO to create and intent to do the possible configuration in the network with the help of the EDC.

The improvements in PMEM for the second iteration of the project are focused on the ML Based Detection Module. The current version of the PMEM is using both supervised and unsupervised machine learning approaches to detect and classify known and unknown network attacks. The supervised machine learning models are trained to detect known cyber threats while an unsupervised model is trained to detect unknown or zero-day exploit attacks.

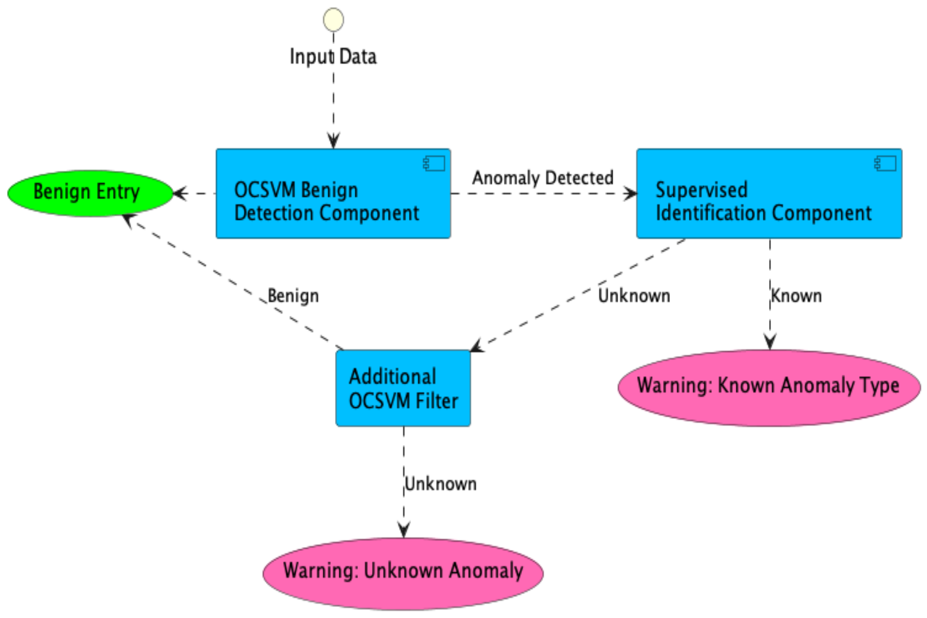

Figure 8. ML Based Detection Module Overall Internal working

PMEM was already deployed in one of the use cases, F2F, for IT-1 and doing detection in a real-time scenario. The data collection agent will be deployed in the FISHY appliance for IT-2 and data will be forwarded with the help of the SPI to the ML Based Detection Module. The final prediction and recommended suggestion will be stored in a Central Repository to be used by other FISHY modules.

4.5 Trust Monitor

This section describes the implementation details of the Trust Monitor (TM), a subcomponent of the TIM module in the FISHY architecture, which has the aim of providing periodic remote attestation of the FISHY appliance, and the infrastructural nodes present in the enterprise domain and protected by the FISHY solution. Figure 9 represents a high-level overview of the FISHY architecture and highlights the logical positioning of the TM inside it.

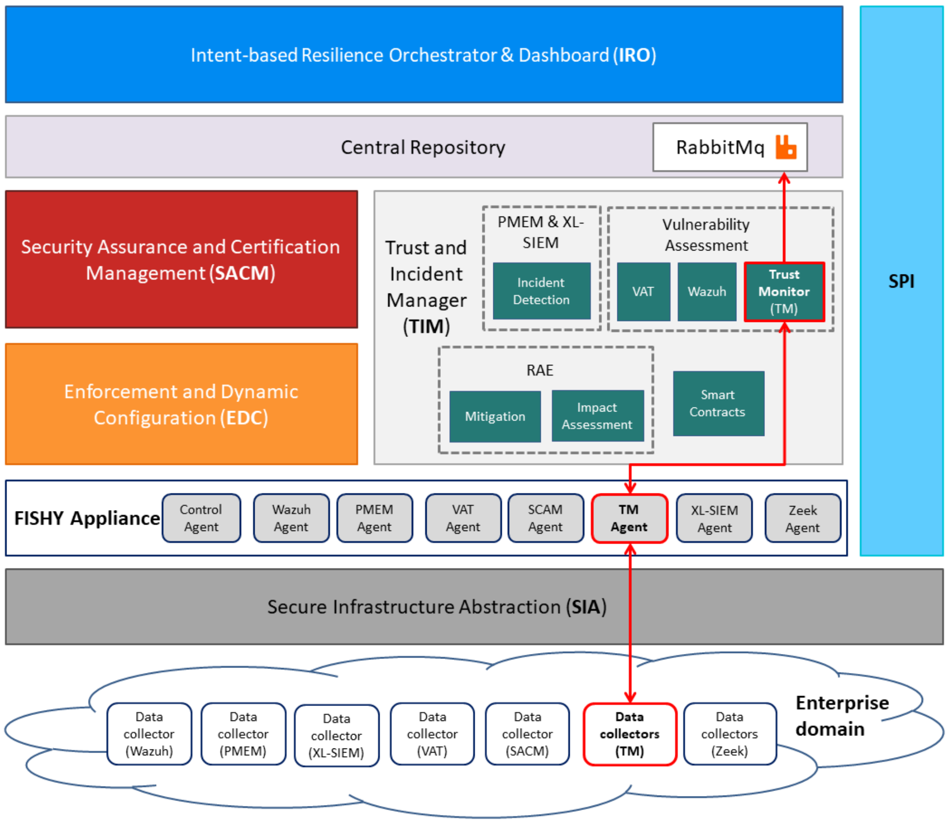

Figure 9. FISHY high-level architecture

The Trust Monitor (TM) has been developed with a modular architecture, whose subcomponents are represented in Figure 9 and described in the following:

- *TM Core Application* is the main component in the TM as it manages the central high-level logic of the TM framework. It has the task of 1) receiving registration requests for the various entities to be attested (e.g. physical or virtual compute nodes, IoT devices), hierarchically organized in order to reflect the real configuration of the enterprise infrastructure; 2) starting the attestation process on each registered entity; 3) building the attestation reports for the various entities by aggregating the attestation results related to each of its sub-components and finally 4) publishing the attestation results to the Central Repository via a RabbitMQ queue.
- *Connectors* allow the TPM Core Application to interact with Databases and Attestation Adapters. In particular, *Database Connectors* expose APIs for accessing instances, whitelists or integrity verification information; the *Adapters' Connector* allows the TM to interact with different attestation frameworks by dynamically loading the corresponding Attestation Adapters specified in a configuration file.
- *Attestation Adapters* allow to instantiate different remote attestation workflows, each one with its verification logic depending on the type of node; this enables the TM to attest nodes with different architectures (e.g. x86, ARM, RISC-V) and Root of Trusts (e.g. hardware TPM, ARM TrustZone, Intel SGX). Attestation Adapters enclose all the interaction logic with a specific attestation framework, exposing a common interface to the Adapters' Connector, and standardizing the format of the attestation results for the various entities attested through different attestation solutions.

- *Databases* store all the information regarding the various entities to be attested, the whitelists and other information used by the attestation frameworks to carry out the integrity check, the reliability policies and the attestation results produced by the TM framework.
- *Attestation Result Queue* collects all the attestation results produced by the various attestation frameworks and published by the Attestation Adapters in a standard format, making them available to the TM Core Application, which in turn will use them to create aggregate attestation reports for entities organized hierarchically, based on the configured integrity policies.

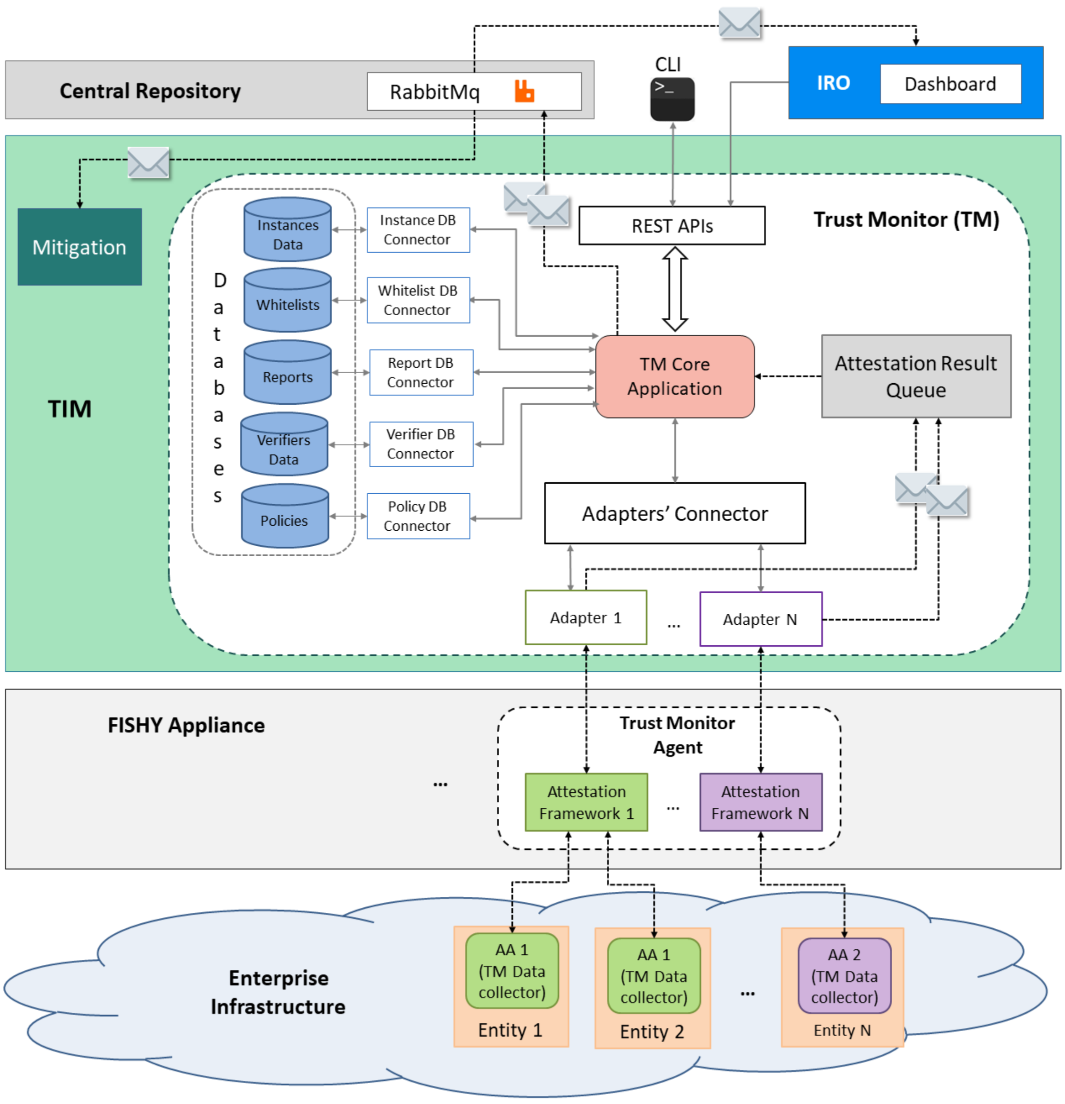

Figure 10. Trust Monitor subcomponents

Figure 10 represents the interaction workflow of the TM with the other components of the FISHY architecture, adaptable for the different use cases of the FISHY platform. The TM starts the attestation process on the infrastructural nodes by sending the Attest Infrastructure command to the TM Agent deployed in the FISHY Appliance. At this point, the attestation frameworks installed in the FISHY Appliance will start the periodic remote attestation process of the customer infrastructure; such a process consists of sending an attestation request to each node deployed in the infrastructure and evaluating the integrity report, which contains tamper-proof evidence on the health state of the node in question. For this to happen, an Attestation Agent (AA) must be installed on each infrastructural node to be attested, i.e. a service with the task of receiving attestation requests and responding to them with an integrity report, whose reliability is based on the root of trust provided by the specific node. When the attestation framework has assessed the integrity state of the entity, it sends the result of the attestation to the corresponding Adapter, which converts it into a standard format and publishes it on the internal queue available to the TM. As soon as all the attestation reports relating to the various entities present in the infrastructure are available, the TM aggregates an attestation report for the entire infrastructure, keeping also into consideration the attestation policies established for the overall assessment of the trustworthiness level of the infrastructure. Then, the TM publishes the aggregate report on the Central Repository via a RabbitMQ queue, in order to make it available to all components of the FISHY architecture interested in it. In particular, the FISHY Dashboard will use the report to view in real-time the current health status of the infrastructure nodes, while the Mitigation module will analyse the result of the report and, in the case of integrity failure, will send the IRO-specific intents as a remedy for the breach. The IRO is then responsible for transmitting them to the Enforcement & Dynamic Configuration (EDC) module in the form of high-level security policies.

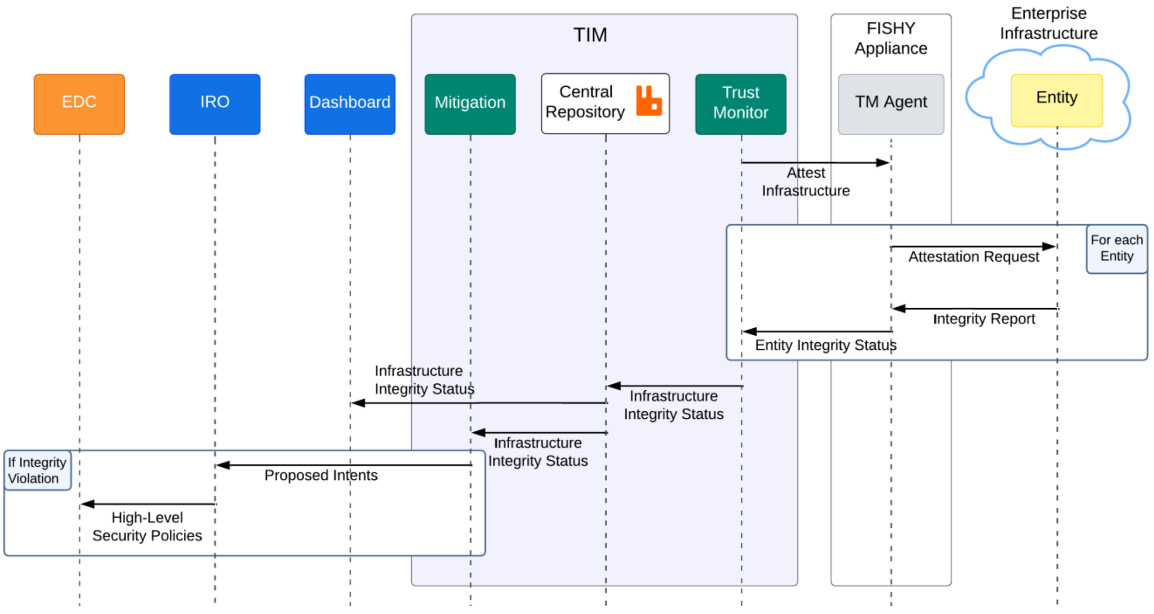

Figure 11. TM interaction workflow with the other components of the FISHY architecture

Trust Monitor deployment

The Trust Monitor has been implemented as a set of microservices, where the Trust Monitor Core Application holds the central role. These services are deployed as Docker containers using the Docker Compose tool, which allows for quickly and efficiently instantiating all the Trust Monitor sub-components and easily enables network interaction among them. In particular, the Trust Monitor is composed of four containers: one for NoSQL databases, one for relational databases, one for the internal queue and the last one for the TM core logic. As regards the Trust Monitor Agent, it will contain the Keylime framework for the attestation of physical nodes, containers deployed with different attestation technologies (e.g. Docker, CRI-O, container) and Kubernetes pods; other attestation frameworks will be available based on the type of nodes present in the infrastructure to be attested. Finally, the Trust Monitor Data Collectors present in the customer's infrastructure will be the Attestation Agents corresponding to the attestation technology used (e.g. the Keylime Attestation Agent), which will be installed on each of the nodes to be attested.

4.6 Zeek

Zeek is a passive open-source network traffic analyser that can output an extensive set of logs describing the network activity. Zeek comes with multiple built-in functionalities for a range of analysis and detection tasks. It also provides a domain-specific scripting language for expressing arbitrary analysis tasks[4]. Zeek is not an active security device, like a firewall or intrusion prevention system, rather it sits on a "sensor," that quietly and unobtrusively observes network traffic in a compact, high-fidelity transaction logs, file content, and fully customized output, suitable for manual review on disk or in a more analyst-friendly tool like a security and information event management (SIEM) system [5]. As it has been said, Zeek outputs rich information-filled logs about a wide range of protocols, such as HTTP, DNS, and DHCP, but it also outputs a log with Zeek-generated alerts which can be triggered by modules written in its scripting language. These modules can be written to track any kind of metric with the measures gathered by Zeek which can then generate a notice when an anomaly in the traffic is detected. These alerts can then be forwarded through the SPI either to a tool or the Central Repository, allowing FISHY to closely monitor any kind of anomalies detected in the network.

4.7 Smart Contracts

The Smart Contracts component is responsible for ensuring the integrity of a) the recorded security events and b) the enforced mitigation policies. The Smart Contracts component communicates directly with the Central Repository of the FISHY platform (to receive this information) and stores it in the blockchain. The components that generate this information include SACM, TIM-WAZUH, TIM-PMEM and IRO while the Smart Contracts component communicates with the Central Repository through a RabbitMQ.

Figure 12 depicts a high-level architecture of the FISHY platform and gives attention to the Smart Contracts' placement in it. The figure emphasizes the place of the component in the general architecture as well as the different sub-components it consists of.

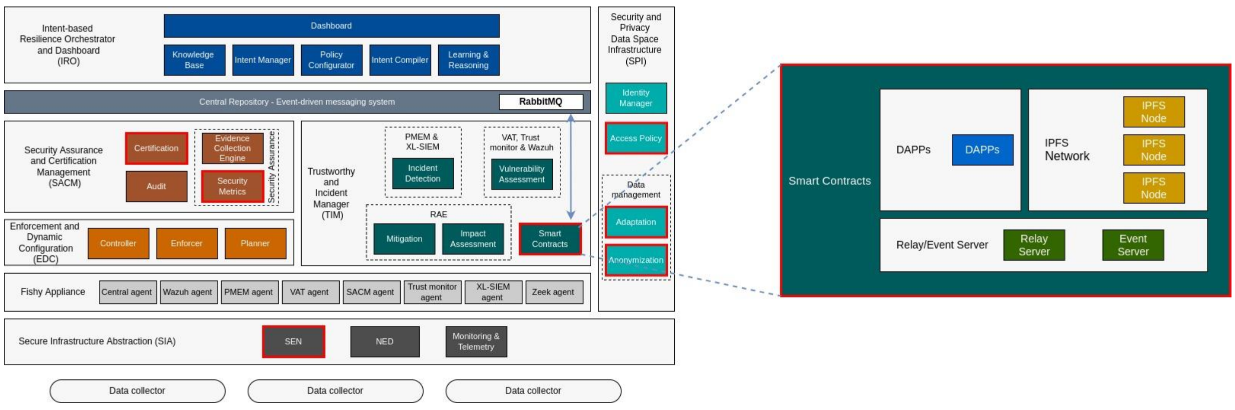

Figure 12. Smart contracts internal organisation and relation to the high-level architecture

The component consists of the following sub-components:

- IPFS
- DAPPs (Decentralized Apps)
- Relay Server
- Event Server

IPFS

The IPFS is a P2P (peer-to-peer) distributed file system that can be used to store and access any type of data (e.g. files, JSON, jpeg etc.). Considering that the information that accompanies/describes a security event and a policy may be large (e.g. larger than just a few bytes) and thus not appropriate for being stored in the blockchain, we have decided to integrate an IPFS infrastructure. The different events/policies along with the relevant information will be stored in a private IPFS network and we store in the blockchain a) the ID of the security event/policy and b) the corresponding link in the IPFS system. This way, the required time to store the information is reduced, compared to the case where we store information in the blockchain, and allows for larger data sizes. The events/policies become accessible via a link, which is stored in the blockchain. The IPFS guarantees that if any change happens to the source data, it will be detectable.

The addition of the smart contracts component to the FISHY platform allows:

1. The verification that a security event/policy is detected by FISHY and
2. The guarantee that the details of an event/policy are not tampered with (these details are accessed through the link stored in the blockchain).

DAPPs

The DAPPs sub-component consists of the Smart Contracts that contain the logic for storing the original source of the data (IPFS link) and retrieving it. The DAPPs component is deployed in a private blockchain network, namely Quorum. This private blockchain network solution uses the IBFT (Istanbul Byzantine Fault Tolerance) consensus mechanism. This mechanism is one of the best regarding performance and transaction speed, therefore making the overall implementation very fast.

Relay Server

The Relay Server subscribes to the RabbitMQ of the Central Repository and is notified of every new event/policy added. The Server then composes the information in a file to store in the IPFS network. The Relay Server also exposes REST API endpoints connecting to the DAPPs, for the components to access the link for the details of the events/policies in the IPFS.

Event Server

The Event Server is mainly responsible for keeping track of all the events that are emitted from the smart contracts of the DAPPs sub-component. Instead of querying the blockchain directly to ensure that an event has been triggered (e.g when a new IPFS link is stored), the log the Event Server creates to keep track of the emitted events can be used to determine whether a call to the DAPPs was successful and check briefly its most basic details (e.g. the ID of the event/policy).

It should be noted that all the above sub-components are designed to work as Kubernetes deployments. The Quorum blockchain platform is available as a Helm chart that will also be deployed in a Kubernetes cluster.

Smart Contracts Component Deployment

The Smart Contracts component and the relevant blockchain network will be deployed in Synelixis' cloud for the lifetime of the project. However, the blockchain network which is necessary for the FISHY smart contracts component operation will in general be a private blockchain network as is the case for numerous applications/solutions in the market. This means that one or multiple nodes could be deployed in Synelixis, another (or another set) in ATOS, or XLAB. The FISHY clients do not need to contribute to the maintenance or deployment of the blockchain network.

This method ensures that:

1. A decentralized solution is offered since the nodes can be hosted on different premises.
2. The data stored are secure since they cannot be deleted, and any tampering attempts are detectable and
3. There is no single point of failure, since all the nodes hold replicas of the stored data, and one failed node cannot compromise the entire network.

It is worth stressing that: i) the fact that FISHY relies on a private blockchain network does not decrease the value of the decentralized solution and ii) protecting FISHY operations through the integration of blockchain techniques increases the security of the FISHY platform; this does not mean that this is 100% secure as such a security level does not exist; it means that a higher security level is reached and this should be considered keeping in mind the value of the protected data.

Farm-to-Fork (F2F) Use Case

In order to better demonstrate the role of the Smart Contracts Component, it would be useful to examine a use case on the F2F example. Let's assume that a malicious actor is trying to login into the F2F platform by enacting a brute-force attack. In this case, we expect two components, SACM and IRO, to perform the following actions:

SACM

1. The SACM component detects the brute-force attack
2. It composes a related security event
3. The event is stored in the Central Repository
4. The Smart Contracts Component gets notified by the Central Repository's RabbitMQ about the addition
5. The Smart Contracts Component produces a file with the details of the event and stores it in the IPFS network
6. The link pointing to the event file is stored in the blockchain

The above steps are depicted in the Sequence Diagram of Figure 13.

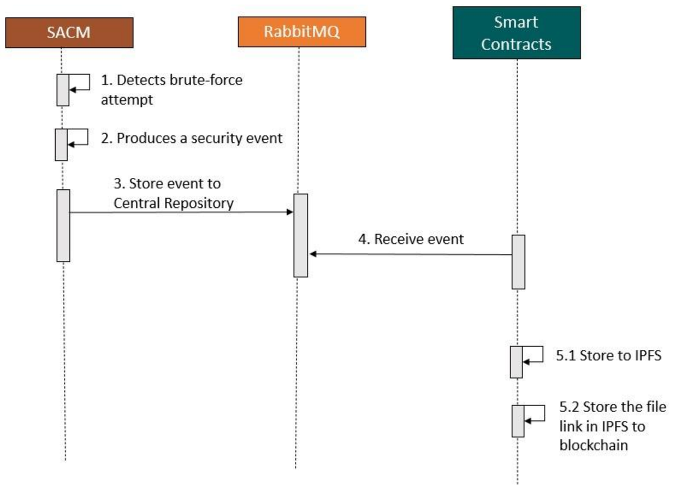
SACM
RabbitMQ
Smart Contracts
1. Detects brute-force attempt
2. Produces a security event
3. Store event to Central Repository
4. Receive event
5.1 Store to IPFS
5.2 Store the file link in IPFS to blockchain

Figure 13. Sequence Diagram for smart contracts component storing a detected security event in the F2F use case

IRO

1. The IRO component, after being notified of the threat, composes a mitigation policy, which blacklists the IP from where the login attempts have been detected
2. The policy is stored in the Central Repository
3. The Smart Contracts Component gets notified by the Central Repository's RabbitMQ about the addition
4. The Smart Contracts Component produces a file with the details of the policy and stores it in the IPFS network
5. The link pointing to the policy file is stored in the blockchain

The above steps are depicted in the Sequence Diagram of Figure 14

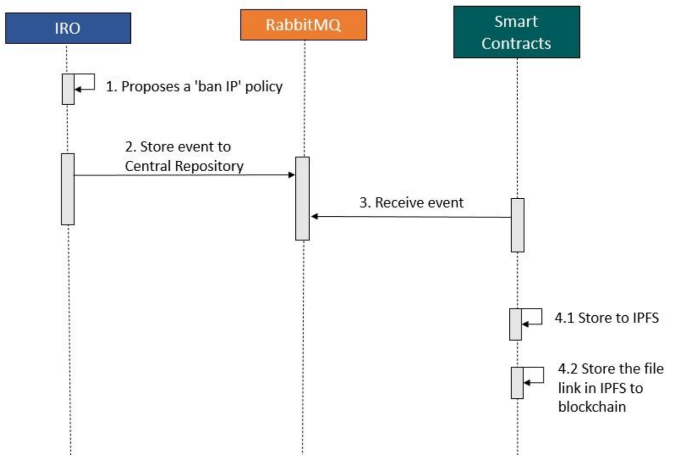
IRO
RabbitMQ
Smart Contracts
1. Proposes a 'ban IP' policy
2. Store event to Central Repository
3. Receive event
4.1 Store to IPFS
4.2 Store the file link in IPFS to blockchain

Figure 14. Sequence Diagram for smart contracts component storing a policy to the blockchain in the F2F use case

The Smart Contracts Component is used to persist the events/policies the FISHY components produce and ensure the integrity of the data. The data for these events/policies are immutable for every decision made or action enforced, and the Smart Contracts Component is a method to keep track of all actions taken and protect the information from malicious actors.

4.8 Central Repository

Formerly known as the Threat/Attack Repository, this component was renamed to Central Repository when it was made transversal, rather than a TIM component only. The design of merging a traditional CRUD-based REST API for managing data storage with a pub/sub system to allow any interested party to be instantly notified when new data is available for processing was a requirement most components shared, and it became obvious that it would be inefficient if every component needs its own storage solution to be developed.

Central Repository now facilitates communication between components from every technical work package. In IT-2, its data model definitions have been expanded from only dealing with events and alerts, to the ability to store various policies, certification data and facilitate immediate responses from the platform thanks to the pub/sub system providing instant notifications.

Further developments of the Smart Contracts component also allowed the development of Central Repository to be less dependent on verifiable data immutability. While amending data after the original write into the Central Repository is not a common requirement and in most cases is not even supported, leveraging Smart Contracts persistence of data on the blockchain gives an assurance that data has not been tampered with.

4.9 VAT

Vulnerability Assessment Tool (VAT) provides the capabilities to detect vulnerabilities of both web services and infrastructure. Its suite of tools includes W3AF[6], OWASP [7] and NMAP [8]. These tools can be employed in various profiles and configurations that can be provided via VAT's webUI and its scheduling capabilities enable multiple rounds of scans.

This tool is especially useful to anyone who's running a custom-built webpage or webUI, as the VAT tool can monitor both a development sandbox or a CI/CD environment, allowing developers to catch vulnerabilities early, or even production systems, showing the necessity for security patches of an external access point. OWASP ZAP and W3AF provide an itemized list of CVEs (Common Vulnerabilities and Exposures) found by scanning a web service, while the NMAP module is able to scan multiple targets in one scan and identify exposed services that should be connectable only on the private network, such as misconfigured Redis databases[9]. In IT-2, VAT received various tweaks to its webUI to allow better integration into the FISHY Dashboard, OWASP and W3AF have been updated to their newest lists of CVEs and a new post-hook adapter was developed that allows the results of VAT scans to be propagated to the Central Repository, where they are available for further analysis and notifying the system administrators of the status of their infrastructure.

4.10 LOMOS

LOg MOnitoring System or LOMOS, is an ML-based anomaly detection solution. Its role in the FISHY platform is to provide a second layer of analysis of gathered data and metrics. While most tools and services employed by FISHY form their own chain and process of gathering and analysing data, LOMOS is employed in a more passive, observational role. Instead of deploying its own data collectors and agents in the monitored infrastructure, it taps into the data stream of already collected metrics flowing to the FISHY platform and establishes a baseline for normal activity and searches for anomalous behaviour that a more specialised, rule-matching solution can miss.

LOMOS consumes raw logs, that are fed into its Log parser module, which structures the logs into a format ready for analysis. This part of the process is unsupervised learning, meaning the incoming logs do not need to be in a known format or require any pre-processing, also known as "cooking". The structured logs output by the Log parser are then analysed by the Anomaly detector module, which produces an individual, per-log anomaly score.

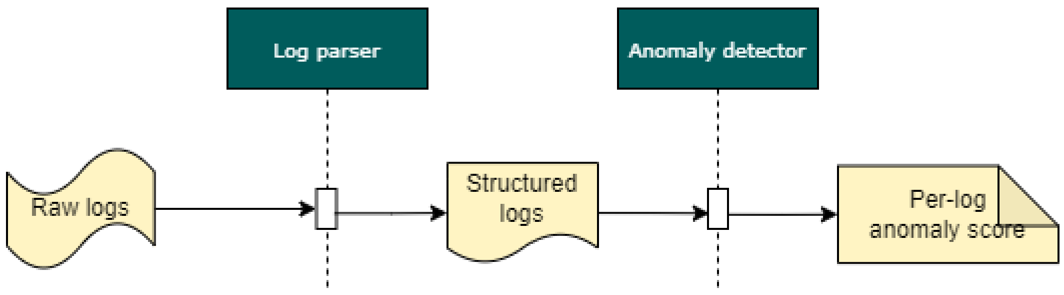

Figure 15 - LOMOS internal log processing

This second layer of analysis provided by LOMOS and the anomaly scores enables the FISHY platform to flag activities normally perceived as normal, benign events for additional analysis or investigation. Logs with an anomaly score that surpasses a threshold are persisted in the Central Repository and since data coming into the FISHY platform is always labelled by its source, the alerts generated by LOMOS allow system administrators to drill down into the sequence of events flagged as out-of-the-ordinary.

5 Implementation in the Reference Framework

5.1 Introduction to FISHY Reference Framework

The FISHY Reference Framework (FRF) is a testbed environment that provides a stable platform where all FISHY Components and FISHY use cases can be deployed to test their functionality and integration with the rest of the components of the project. The FRF is available at the 5TONIC[10]. This FRF is composed of a set of domains (known as FISHY domains) that represent the locations where the FISHY components and modules are deployed to provide their functionalities.

These FISHY Domains can be flexibly incorporated into the testbed in the form of Kubernetes (K8s) clusters or OpenStack [11] domains. Moreover, other external domains can also be incorporated into the testbed (e.g., a Domain composed of one or more Virtual Machines) using a VPN service. This service is also being considered to support the connectivity of external devices to the FRF, such as the data collector (it is currently being developed as part of IT-2).

One of the reasons that provide the flexibility to incorporate heterogeneous domains into the FRF is the approach that the SIA utilizes for the communications between these domains. In this case, the Network Edge Device (NED), as part of the SIA module, provides secure communication between the different domains. These NEDs are present in each one of the domains, being each NED connected with other desired neighbouring NEDs through the use of IP tunnels [12]. This way, NEDs build a network overlay, which provides end-to-end connectivity across all the domains. The communications can be protected between each other through IPsec [13] mechanisms, adding an extra layer of security in the process. Further information about the NEDs and their overlay can be found in the official repository of the FISHY Sandbox [14], described in Deliverable 5.1 [15]. At the time of writing, a monitoring tool is being developed to verify the status of this overlay and the inter-domain communications, and its basis will be based on other tools, like Zabbix [16]. The development of this monitoring tool is also being considered as part of IT-2.

5.2 SPI Integration

In the previous iteration IT-1, the SPI Access policy component (Keycloak) was deployed using docker through the docker-compose deployment file. This allowed for a very straightforward deployment were using docker-compose, a Keycloak docker image could be directly downloaded from the Docker hub and the necessary changes to the container could be applied through the docker-compose.yml file instructions or by specifying a custom Dockerfile build file (e. g., environmental variables, import custom files, specify open ports, etc…). To integrate the Keycloak instance in the FISHY Reference Framework it was necessary to translate this deployment file to a Kubernetes deployment file. To accommodate this change some of the ease of use of using docker-compose was sacrificed. Namely, while using docker-compose images could be easily edited and automatically rebuilt before runtime by specifying a custom Dockerfile file, this was not as straightforward in Kubernetes, where images would need to be manually built by the user beforehand.

For instance, for Keycloak to have SSL encryption, certificates need to be provided. Using docker-compose, these could be simply copied to a directory in the host machine which would then be linked inside the container using a shared volume. In the Kubernetes deployment version these certificates need to be directly built inside the image beforehand, which makes the deployment process more cumbersome.

5.3 TIM Integration

Integration of TIM tools and components into the FISHY Reference Framework was relatively straightforward. The FISHY Appliance is deployed on a VM, instead of Kubernetes pods, so the deployment process is virtually unchanged from its normal operation, save for defining a new inventory file for already existing Ansible scripts.

Other TIM components, that reside in the FISHY Control Services domain rather than the Appliance, were containerised from the earliest stages of implementation and as such were also no issue to deploy in Kubernetes instead of plain docker.

What proved slightly challenging was the specific networking model of the FRF, the division of management and data network interfaces and the way they must be configured inside a pod at runtime. While this subject was responsible for the biggest change in the services, its impact was limited to the containerisation build process, not the implementation of the services themselves. When specifying a Dockerfile for a service, essentially describing how an image for a container should be built, it is usually sufficient for the last command, running the service on container startup, to be a simple one-liner, running a binary or calling an interpreter and providing a few parameters. The specifics of the FRF networking model however required an additional Bash script to be provided to the container that configured the network interfaces based on provided environment variables, however, the provided guides and examples of components that were integrated into FRF sooner simplified this process.

6 Conclusions

This deliverable provides a complete description of the Trust Manager (TM) Module of the FISHY Platform for the 2nd iteration of the project. As it has been designed on the 1st iteration, the TM module is composed of two main blocks, the Security and Privacy Data Space Infrastructure (SPI) and the Trust and Incident Management (TIM).

The implementation of these blocks had already been described in the WP3 documents published before (namely, D3.1[1] and D3.2[2]) but due to the advancements of the project and the decisions taken by the development team, these ideas must be adapted and updated taking into account the constraints that have been appearing along the development and implementation phase.

Initially, this document describes the architecture issues of the module and the approach taken to satisfy all the requirements raised by the Use Case partners, and it is also presented all the functionalities offered by these modules correlated with the tools used. Posteriorly, it is presented the updated workflow of the two main blocks that composed the TM module, namely the SPI and TIM. The SPI section is focused on the features added to be implemented on this 2nd iteration of the project, as the TIM section is presented all the tools that the FISHY Platform will be served. The final section of the document reveals some adjustments needed to be taken towards the phase of implementation of the approach presented and traces the path to complete integration with the remaining modules of the project and with the Reference Framework.

References

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

[2] [FISHY] – D3.2 Trust Manager Integration. (IT-1) Marin-Tordera, Eva; Ruiz, Jose Francisco; Alonso, Juan Andres. 2021.

[3] [FISHY] – D6.3 Use cases settings and demonstration strategy. Gonos, Antonis; Alvarez, Antonio. 2022

[4] Zeek Documentation. https://docs.zeek.org Retrieved 2022-09-28

[5] Zeek - The Zeek Network Security Monitor. https://zeek.org/. Retrieved 2022-09-28

[6] W3AF, Open-source web application security scanner, https://w3af.org/ (15 October 2022).

[7] OWASP, Zed Attack Proxy, https://owasp.org/www-project-zap/ (15 October 2022).

[8] Nmap: The Network Mapper, https://nmap.org/ (15 October 2022)

[9] Many Internet-Exposed Servers Affected by Exploited Redis Vulnerability, https://www.securityweek.com/many-internet-exposed-servers-affected-exploited-redis-vulnerability (28 October 2022)

[10] The Linux Foundation. Kubernetes. Available at: https://kubernetes.io (12 October 2022)

[11] The OpenStack project. Openstack. Available at: https://www.openstack.org (12 October 2022)

[12] M. Mahaligam et al. Virtual eXtensible Local Area Network (VXLAN): A framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks. RFC 7483. August 2014.

[13] T. Chown et al. IPv6 Node Requirements. RFC 8504. January 2019.

[14] Luis F. Gonzalez et al. FISHY-Sandbox-development. Available at: https://github.com/H2020-FISHY/FISHY-Sandbox-development [14 October 2022]

[15] [FISHY] – D5.1 IT-1 FISHY Release integrated. Manjón, Jose Manuel; Alonso, Juan; Alvarez, Antonio. 2022.

[16] Zabbix LLC. Zabbix 6.2. Available at: https://www.zabbix.com (12 October 2022)