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

D3.4 Trust Manager IT2 integration

Document Identification			
Status	Final	**Due Date**	28/02/2023
Version	1.0	**Submission Date**	28/02/2023

Related WP	WP3	**Document Reference**	D3.4
Related Deliverable(s)	D3.1, D3.2, D3.3, D4.1, D4.2, D4.3, D4.4	**Dissemination Level (*)**	PU
Lead Participant	UPC	**Lead Author**	Eva Marin-Tordera
Contributors	XLAB, UMinho, POLITO, SYNELIXIS, STS	**Reviewers**	UMinho (Henrique Santos, André Oliveira)
			ATOS (Antonio Alvarez, Jorge Martinez)

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
André Oliveira	UMinho
Henrique Santos	UMinho
Pedro Magalhães	UMinho
Xavi Masip	UPC
Eva Marín	UPC
Jan Antić	XLAB
Hrvoje Ratkajec	XLAB
Daniele Canavese	POLITO
Silvia Sisinni	POLITO
Enrico Bravi	POLITO
Alexandra Lakka	SYNELIXIS
Guillermo Yuste	ATOS
Jorge Martínez	ATOS
Grigorios Kalogiannis	STS

Document History			
Version	Date	Change editors	Changes
0.1	29/11/2022	Eva Marín Tordera (UPC)	Table of Contents
0.2	10/01/2023	Jan Antić, Hrvoje Ratkajec (XLAB)	Added the First version of Section 3; 3.1, 3.2, 3.3 and 3.10
0.3	12/01/2023	André Oliveira (UMinho)	Added the First version of Section 2; 2.1; 2.2; 2.3; 3.6
0.4	16/01/2023	Daniele Canavese, Silvia Sisinni, Enrico Bravi (POLITO)	Added the First version of Section 3.8
0.5	16/01/2023	Eva Marín (UPC)	Added content to section 3.5, and first version of introduction
0.6	18/01/2023	Alexandra Lakka (SYN)	Added content to section 3.9
0.7	18/01/2023	Eva Marín (UPC)	Added section 4
0.8	3/02/2023	Guillermo Yuste, Jorge Martínez (ATOS)	Added content to sections 3.4 and 3.7
0.9	10/02/2023	Eva Marín (UPC)	Added content to section 5
0.91	13/02/2023	Pedro Magalhães	Added content to section 5

		(UMINHO) Silvia Sisinni (POLITO)	
0.92	14/02/2023	Pedro Magalhães (UMINHO)	Added content to section 2.1, 2.2 and 2.3.
0.93	14/02/2023	Jan Antić, Hrvoje Ratkajec (XLAB)	Updated Section 3 (3.1, 3.2, 3.3), Added content to Section 4
0.94	17/02/2023	Eva Marín (UPC)	Updated content in section 5
0.95	17/02/2023	Pedro Magalhães (UMinho), Eva Marín (UPC)	Add missing content in section 5, and to the list of acronyms
0.96	17/02/2023	Alexandra Lakka(SYN), Jan Antic (XLAB), Mounir Bensalem (TUBS), Eva Marín(UPC)	Added content to section 5 and conclusions
0.97	23/02/2023	Henrique Santos (UMINHO), Andre Oliveira (UMINHO), Eva Marín(UPC)	Revision from UMINHO, and addressed comments of this revision
0.98	24/02/2023	Jorge Martínez (ATOS), Guillermo Yuste(ATOS), Eva Marín (UPC)	Revision from ATOS, and addressed comments of this revision
0.99	27/02/2023	Antonio Álvarez (ATOS)	Revision from ATOS
0.991	27/02/2023	Eva Marín (UPC), Jorge Martínez (ATOS), Hrvoje Ratkajec (XLAB), Grigorios Kalogiannis (STS), Enrico Bravi (POLITO)	Addressed comments from revision
0.95	27/02/2023	Eva Marín (UPC)	Version for QA
1.0	28/02/2023	Antonio Álvarez, 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)	27/02/2023
Quality manager	Juan Andrés Alonso (ATOS)	28/02/2023
Project Coordinator	Antonio Álvarez Romero (ATOS)	28/02/2023

Table of Contents

List of Tables

List of Figures

List of Acronyms

Abbreviation / acronym	Description
AMQP	Advanced Message Queuing Protocol
API	Application Programming Interface
APol	Access Policy
CA	Cyberagent
CEF	Common Event Format
CID	Content Identifiers
D3.2	Deliverable number 2 belonging to WP3
DDoS	Distributed Denial-of-Service
EC	European Commission
FCS	FISHY Control Services
FRF	FISHY Reference Framework
GDPR	General Data Protection Regulation
GUI	Graphical Interface
HIDS	Host-based Intrusion Detection System
HTTP REST	HTTP Representational State Transfer
HTTPS	Hypertext Transfer Protocol Secure
IdM	Identity Manager
IPFS	InterPlanetary File System
JSON	JavaScript Object Notation
JWT	JSON Web Token
KPI	Key-Performance Indicators
ML	Machine Learning
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
SSL	Secure Sockets Layer

Abbreviation / acronym	Description
TIM	Trust & Incident Manager
TM	Trust Manager
VPN	Virtual Private Network
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 implementation of blocks in the Trust Manager (TM) module of WP3 for IT-2 and their integration. In the previous deliverable of this Work Package, D3.3 [1], the blocks in TM, TIM and SPI, have already been designed for IT-2, including new functionalities and tools for this second project iteration.

For IT-2, the following functionalities are implemented in TIM: vulnerability assessment, incident detection, mitigation, prediction and estimation of risks, remote attestation, trustworthy mechanisms and collaboration among stakeholders, extension/expansion scalability and global security events storage, being some of these functionalities new for this second project iteration, such as prediction and estimation of risks, remote Attestation, trustworthy mechanisms and collaboration among stakeholders and extension/expansion scalability.

Concerning SPI, the functionalities implemented are: identity Management, privacy enforcement, access control and data management. For this second project iteration being new in this second project iteration, privacy enforcement and access control are new. All these functionalities of TIM and SPI are matched with different tools, some of them also new for IT-2, as it will be shown in Table 1.

Taking as input the last design proposed in D3.3 [1], and considering the deployment of the different tools in the use cases and their specific implementation, the integration of the different tools in the Fishy Reference Framework (FRF) is described, as well as ad-hoc deployment in the use cases.

Finally, in this deliverable it is also described the interfaces between components of TIM and SPI, all in all related to the specific workflow of the Trust Manager (TM). Different tables shown extensively one by one the interfaces and the type of communication employed between all the components in TM, as well as the interfaces with components from other Work Packages.

1 Introduction

1.1 Purpose of the document

This deliverable describes the final and integrated Trust Manager (TM) module release for IT-2, ready to be integrated with the Security and Certification Manager (SCM) module from WP4, as well as with IRO and SIA in WP5.

After the description of the design and implementation of each one of the blocks namely SPI and TIM for IT-2, done in D3.3 [1], in this deliverable we describe on one hand, the SPI and TIM outcomes to be integrated, considering the new added functionalities in IT-2. On the other hand, we describe for each one of the modules the integration in the FISHY reference Framework (FRF), as well as the interfaces between these modules.

1.2 Relation to other project work

The work done in WP4 is done completely in parallel to the work in WP3 Trust Manager (TM). The outcome of both Work Packages in month 30 is the release of the corresponding blocks integrated in the FRF to be jointly tested with also IRO and SIA in WP5. Moreover, the integration of each block of TM, task T3.3, has been done in collaboration and with the support of T5.3, task devoted to the whole FISHY integration.

1.3 Structure of the document

This document is structured as follows:

- **Chapter 2** presents the SPI integrated outcome.
- **Chapter 3** presents the TIM integrated outcome.
- **Chapter 4** presents the integration of the Fishy Appliance in the FRF.
- **Chapter 5** presents the interfaces of TM.
- **Chapter 6** concludes the deliverable.

2 SPI Integrated outcome

Security and Privacy Infrastructure (SPI) module establishes a constant and all-around communication with all modules of the FISHY Project, and specially with the Trust and Incident Manager (TIM) and Security and Assurance Certification Manager (SACM) modules. SPI provides an interface between low-level components, higher-level modules, and users, providing both local and federated Access Control services along different domains and some data management functions including privacy preserving mechanisms. It is important to note that SPI also performs a key role in identity and access management, defining who should have access to the platform and the different roles that should be granted different types of access, through a dedicated Security Policy manager.

As specified on the Deliverable 3.3 [1], the SPI module can be subdivided into three components, namely identity management (IdM), access policy (APol) and data management (DM) including privacy enforcement features (e.g. anonymization). All these components have their unique characteristics and their specific requirements concerning integration into the FISHY Reference Framework (FRF). Those details are specified in the following subsections.

2.1 SPI identity management integration

In the previous project iteration (IT-1), the SPI Access Control component (Keycloak) was deployed using docker technology through a docker-compose deployment file. This made the deployment process very straightforward, whereby simply running the docker-compose up command, a Keycloak docker image would be directly downloaded from the Docker hub and the necessary changes to the container would 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, among other parameters). To integrate the Keycloak instance into 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 configurations inherent to docker-compose were sacrificed. Namely, while using docker-compose images it was easy to edit and automatically rebuild before runtime by specifying a custom Dockerfile file, this is not so straightforward in Kubernetes where images need to be manually built by the user beforehand.

After a successful migration and testing phase with the sandbox, the SPI IDM component was deployed in the FRF, using the credentials provided by U3CM. It's currently running the Fishy Access Control domain services at the domain #1 (as it is identified within the project). The console GUI is working and accessible from outside the FRF. Test SSL Certificates were generated in house to provide HTTPS support. The SPI-IDM component is already deployed in all the use-cases through the FCS instance in the FRF. It's currently being used to authenticate users using the tools' dashboards.

2.2 SPI data management integration

As it was described in D3.2 [3], the Common Event Format (CEF) is a standardized data format that allows for event data normalization. It was agreed between all partners that the CEF format would be used to write events into the central repository. One of the functionalities of the SPI DM component is to do this, operating as a service. The SPI DM component has the capability to receive events from the tools, normalize them for the CEF format and send them to the central repository.

Along with the development of IT-2 a research task was initiated with the tool partners to specify each tool event format to integrate the respective event normalization functionality into the SPI Data Management component. The work is still in process, and currently the SPI Data Management component has the ability to normalize events from The Zeek, PMEM & Trust Monitor tools.

Integration with the Central Repository is underway where the SPI DM component will write the normalized outputs after the events are processed.

The normalized events written in the repository will have the following generic format elements:

- device_product: string,
- device_version: string
- event_name: string,
- device_event_class_id: string,
- severity: string
- extensions_list: string

After the SPI DM integration with the central repository, it is expected to be deployed into the FRF in the FISHY Control Services domain, and other domains, if needed, with relative ease using the experience gained when deploying the IDM component. The SPI-DM component will be deployed in all the Use Cases to convert the events from the tools from that particular Use Case to the CEF format and store it in the central repository.

2.3 SPI access policy integration

The access policy component (APol) is presented in D.3.3 as being responsible to manage all policies related to subject granting access to a desirable object, system, or information according to a set of conditions previously defined and agreed upon. As initially described in D.3.1, the APol architecture is 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.

Currently, this component is still under development and for this reason, the integration with FISHY Platform and the FISHY Reference Framework is a topic under discussion. Architecturally, it is expected that an administrator will submit the security policies or set of access policies in natural language (with necessary restrictions), including privacy policies. Then, an agent referred to by “Policy Administrator” is responsible to convert these policies into XACML. By the time these policies are converted, they are also evaluated against each other, using testing scenarios, to see if there exists some incompatibility or contradiction between them. In the case that a policy set converted in XACML is correct and error-free, it will be stored in the FISHY Central Repository or in an external database (still under evaluation), and uploaded to IdM and DM. These tools are responsible for Access Control and Privacy rules enforcement.

The SPI Access Policy component will be deployed in all the Use Cases through the FCS instance in the FRF and it will be used to manage the access policies of each Use Case.

3 TIM Integrated outcome

In this section the different functionalities of TIM and their integration are described as they are listed in Table 1.

Table 1. TM functionalities

Block	Functionality	Modules	Tools
SPI	Identity Management/ Privacy enforcement	Identity Manager	Keycloak
	Access Control/Privacy enforcement	Access Policy	XACML
	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

3.1 WAZUH integration

A detailed description of Wazuh can be found in D3.1 [2] and D3.3[1].

In the IT-2 process, the development of Wazuh was focused on integration into the platform architecture and improving its detection capabilities for threats specific to the use cases.

Thus, the deployment of Wazuh was fully adjusted to the platform architecture. The data flow Collector – Appliance agent – Server was fully implemented and is in line with the FRF. Additionally, the automatic generation of a custom Wazuh rule based on intent was developed and implemented in the TIM component.

Also, Keycloak integration and authentication was successfully implemented.

For the deployment of Wazuh at the Farm-2-Fork use cases, an additional adapter was developed that is able to consume data from RabbitMQ and transfer it to the Wazuh manager via syslog.

3.2 VAT integration

Vulnerability Assessment Tool (VAT) provides the capabilities to detect vulnerabilities of both web services and infrastructure, described in more detail in D3.1 [2], D3.2 [3] and D3.3 [1].

At the start of IT-2 process, VAT was adjusted for seamless functioning within the FISHY platform by implementing various tweaks to its webUI.

Further on, a new post-hook adapted was developed and implemented to allow data flow from monitored infrastructure to the FISHY platform fully in-line with the platform architecture. The post-hook enables 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.

Also, Keycloak integration and authentication was successfully implemented.

For the deployment of VAT at the Farm-2-Fork use case, no additional tunning was necessary.

3.3 LOMOS integration

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.

The integration process in IT-2 encompassed model training based on a log analysis of a system's normal operations. Namely, LOMOS consumes raw logs, that are fed into its Log parser module, which structures the logs into a format ready for analysis. The structured logs output by the Log parser are then analysed by the Anomaly detector module, which produces an individual, per-log anomaly score.

This second layer of analysis enables the FISHY platform to flag activities normally perceived as normal, benign events for additional analysis or investigation. Logs with an anomaly score that surpass 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.

In the integration process to the FRF, LOMOS was deployed on XLAB premises with a trained model. An additional adapter, based on Elasticsearch, was developed and implemented. The adapter enables the flow of live data coming from the Farm-2-Fork use case partner (through the consumer) to LOMOS as well as polling of Elasticsearch indexes in which LOMOS keeps the anomaly scores. The adapter can then send the polled data from Elasticsearch indexes with their anomaly scores to the Central repository, notifying and alerting the use case partner about the out-of-the-ordinary events.

For deployment of LOMOS at use cases, an additional consumer was developed that can fetch raw log data from Elasticsearch. While an adapter based on Elasticsearch is already mentioned above, these two systems serve different functions. Architecturally, the adapter resides in the north-bound domain, FISHY Control Services, and acts as an interface between LOMOS and the FISHY platform, sending the received data for analysis and fetching the anomaly scores. The consumer, on the other hand, lies in the south-bound domain, at the monitored infrastructure level, runs on the Appliance, collects raw log data and forwards it in the north-bound direction.

3.4 XL-SIEM integration

The XL-SIEM purpose is analyzing huge volumes of security information. It is composed of two main parts, the agents collect relevant information for the system, and the server analyzes, correlate the data, and could raise security alerts. A complete description of the XL-SIEM can be found in D3.1 [2], IT-1 improvements documented in D3.3 [1].

In the **Wood-based Panels Trusted Value Chain** (WBP), new sensors were developed and integrated at the Cyber Agent (CA) to detect the relevant SONAE log entries and meet the specific security requirements for this chain.

Some of the sensors were developed from the very beginning, other were refined. More specifically, on the client side the following sensors were deployed.

DDoS attack, a sensor capturing the telemetry was deployed, and specific calculations were made to establish the threshold for alarms. In both cases, where telemetry exceeds a certain value, or do not reach a minimum, an alarm is created.

The CA had already a sensor that detects a **Brute Force** attack, and new sensor was deployed to check directly in the IoT devices and detect it without a HIDS.

Before implementing it, installing an HIDS sensor was needed to detect brute force attacks, as shown in Figure 1:

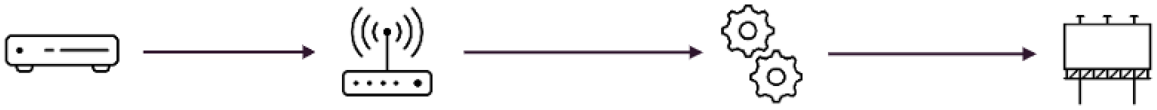

Figure 1: IoT device logs -> HIDS -> XL-SIEM engine -> Alarm

After that, the logs are sent directly from the IoT device to the engine, and we can detect the attack without 3rd party programs. The flow is simplified as shown in Figure 2:

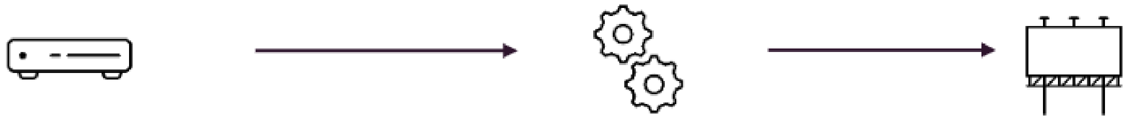

Figure 2: New flow for Brute Force attack

For the **Session hijacking**, a new sensor for monitoring all the information about new sessions being started in the different devices were developed.

And finally, a sensor that detects any access to admin pages from an outside network was created.

In the server new rules were developed to integrate into the engine the new sensors data, that includes:

- DDoS alarms related to the agreed threshold.

- A rule that compares information between the servers and raise the alarm if the same user login in a short time.
- Some existing rules for common attacks like SQL injection or brute force were adapted for the use case.

In the Securing Autonomous Driving Function at the Edge (SADE), new sensors were integrated at the Cyber Agent (CA) to detect “started car” status, “authorized car” status,

“Driver facial recognition” status. Data comes from both the server and the car itself.

Some of the new rules that will raise an alarm in this use case are:

- Three level alarms when an unauthorized driver try to start a car one, two or three times respectively.
- Every car started without the previous authorization.
- When a driver tries to start a non-existing car 50 times.
- The rules for common attacks like SQL injection or brute force were adapted for this use case as well.

For authentication purposes in the Fishy dashboard, Keycloak authentication has been implemented. The XL-SIEM connects to the Keycloak server, validating the token received via the Fishy Dashboard URL.

If the Keycloak server response is valid, authentication in XL-SIEM is complete.

The XL-SIEM integration consist mainly of two tasks: export the alarms, with a new added functionality to export the alarms to an external AMQP queue and connect to the FRF.

The XL-SIEM integration in the FRF consist of the following steps:

- Initially, we installed a VPN client in our machine and, using our profile, login to the 5TONIC environment.
- Then we have added the corresponding configuration to our Kubernetes config files.
- Finally, and to interact with the other partners components, we have added the needed virtual networks to the XL-SIEM.
- Deployment into the FRF, testing and validation.

3.5 PMEM integration

A brief description about the PMEM functionalities can be found in the previous deliverable D3.1 [2] section 4.2.4.1 and D3.3[1], section 4.4. The integration process for IT-2 involves the implementation of the PMEM different modules in the FISHY framework and improving the detection and prediction capability according to use case specific attacks. The data collector is now adopted to work with the FISHY appliance and the implementation process is still in progress. The authentication process is achieved with the help of Keycloak mechanism provided by the SPI identity management to support the Single Sign-on mechanism and it is successfully integrated in the FISHY dashboard. The connection with the Central Repository is made to store the results generated by the machine learning module which classifies and stores the attack entries detected by the PMEM. The output of PMEM will be converted into CEF with the help of SPI data management which is in progress. The connection with the IRO is accessible using a Central Repository. The logs stored in the repository are accessible by IRO and can be shown in the FISHY dashboard. The detection results generated by PMEM are also available on a separate GUI of the PMEM which is already integrated into FISHY dashboard. The integration process with the FRF is also in progress and PMEM is already tuned to be installed and tested with the FRF.

The next steps in the PMEM integration in FRF are:

1) PMEM Dockerizing,

2) Generation of scripts for Kubernetes implementation
3) Testing locally in the deployment of FISHY sandbox
4) Migration of final PMEM version to FRF
5) Finally test PMEM inside FRF with other FISHY components.

In the Farm to Fork (F2F) use case, a PMEM agent was deployed in the infrastructure to get the relevant network flows from the F2F use case. The data transformation is performed on these flows to generate the relevant features needed for the machine learning models to perform the detection and classification. After the data transformation, these features were sent to the PMEM detection module, in the FISHY Central Services, which is responsible for detection of the Normal/Malicious network flows and classification of the known networks attacks which can happen in the network. The detection and classification models were deployed from scratch.

The **detection module** is trained to detect the attacks and label the flows as normal or malicious flows. The flows captured using the PMEM agent are then analyzed by this module to label the network logs as Normal or Malicious network flows. The malicious network flows are further processed by the classification's module.

The **classification module** is responsible for the detection and classifications of the known attacks occurring in the network. The malicious flows from the detection modules are further processed to classify them into known attacks or Zero-day attacks. The corresponding alert messages are shown on the PMEM dashboard.

In F2F use case, PMEM goal is to detect the Brute force attacks and Denial of service attacks in their network infrastructure. The PMEM's role is to identify new attacks in supply chains along with these known attacks.

3.6 Zeek integration

As described in D3.3 [1] Zeek is a passive network traffic analyser part of the TIM. 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. This tool 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 are written to track any kind of metric with the measures gathered by Zeek which generate notices when a recognized anomaly in the traffic is detected. These alerts are then 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.

A data-collector module has been integrated. It can be easily deployed using a docker-compose file which initiates all the appropriate containers. After the events are generated inside the Zeek engine, this will be forwarded to its correspondent agent inside the FISHY Appliance. A simple python script was developed to allow this. The Zeek agent receives the data from the internal data-collector and forwards it to the SPI Data Normalization component so it can be normalized into the CEF Format. After, the normalized data is stored in the central repository and can be viewed using the IRO component. Since Zeek is a network data-collector that needs to attach itself to a host network interface to collect network data, deploying it inside the FRF would hinder the ability to sniff data from the network. Also on the other hand, the tool does not have a data processing component inside the TIM, so deployment to the FRF was not necessary. The FISHY appliance agent will allow the data to be sent inside the FRF to the SPI, acting as the de-facto FRF integration. In the Wood-Based Panel (WBP) use case, Zeek will be deployed to monitor the activity of the production line IoT devices network, gathering data about the traffic and generating notices in case an attack/anomaly is detected.

3.7 RAE integration

The Risk Assessment Engine (RAE) is a Python-based and R-based (models) tool that evaluates in near real time the qualitative and quantitative risk for a company. It has a dashboard, with a general risk position, a more detailed (per asset, per model, per risk) risk view, and an engine where the R models are evaluated. RAE receives as inputs the company asset configuration, the business questionnaire (that can be filled from the dashboard), and the network events and alarms as indicators, that are sent mainly by the XL-SIEM. RAE is described in detail in D3.1 [2], and IT-1 improvements documented in section 4.3 of D3.3 [1].

In the Wood-based Panels Trusted Value Chain WBP, a new asset configuration for the Web dispatcher was added to align with use-case necessities, so that the new asset can be evaluated in the RAE models and configure and receive the XL-SIEM alarms to match them to the RAE indicators.

In the Securing Autonomous Driving Function at the Edge (SADE), the RAE indicators are matched to the XL-SIEM alarms that report the facial recognition status. For this specific attack, where multiple facial recognition logins are attempted, an adaptation of the Brute Force attack model is used. This model considers the threshold/s for facial recognition detection that the SADE edge applies to the in-vehicle camera information.

For authentication purposes in the Fishy Dashboard, Keycloak authentication was integrated. The RAE connects to the Keycloak server, validates the token obtained through the URL of the Fishy Dashboard and if the Keycloak server response is valid, the authentication at RAE is completed for the user the server informed the token belongs to.

Apart from the XL-SIEM, the integration in the FRF consist of the following steps:

- Install a VPN client in our machine and login to the 5TONIC environment.
- Add the corresponding configuration to our Kubernetes config files.
- Add the needed virtual networks to the RAE.
- Deployment into the FRF, testing and validation.

3.8 Trust Monitor integration

The Trust Monitor is a monitoring entity of a network infrastructure whose purpose is to verify that all physical and virtual nodes, deployed in the infrastructure, have a sufficient level of trustworthiness to be used in a given application domain, according to the attestation policies configured for the infrastructure. A detailed description of the Trust Monitor functionality can be found in D3.3 [1].

The Trust Monitor can be easily deployed using a **docker-compose.yml** file which instantiates all the appropriate containers. The Trust Monitor is expected to be deployed into the FRF inside the FISHY Control Services domain through a Kubernetes deployment file. The translation from docker-compose to kubernetes deployment file was done as part of the IT-2 integration process.

The Trust Monitor outputs attestation events describing the current integrity status of an entire infrastructure or IoT ecosystem under monitoring. The attestation events need to be normalized in CEF format and saved in the Central Repository to be made available to the system administrators or other tools of the FISHY Reference Framework. To this end, the Trust Monitor makes use of the service offered by the SPI DM, whose integration was carried out as part of the IT-2 integration process.

The operational workflow includes a preliminary phase in which the nodes that need to be monitored, together with their respective attestation policies, are registered in the Trust Monitor. Architecturally, it is expected that a use case administrator will register the set of nodes to be monitored by means of the Trust Monitor GUI, which will be accessible through the FISHY Dashboard. The authentication of the administrator takes place via the Keycloak server, which is part of the FRF and is contacted to validate the tokens received from the GUI.

The Trust Monitor tool will be integrated into the "Securing Autonomous Driving Function at the Edge" (SADE) use case. The objective of the FISHY platform in the SADE use case is to create a security layer by means of the tools made available by the FRF, in order to protect the information of the sensors and actuators installed in the autonomous vehicle (REMOTIS). Within a network of IoT nodes in general, and in the automotive sector in particular, it is important not only to guarantee the security of the communication between the IoT devices and the edge servers that collect data from such devices and send commands to them, but also to ensure that the IoT devices themselves are "trusted", i.e. that they have not been compromised by an attacker, thus the data they produce are trusted. The task of the Trust Monitor tool within the SADE use case is to monitor the integrity status of the software running on the IoT devices deployed in the autonomous vehicle and to raise "integrity failure" alerts as soon as an unauthorized modification is detected on any of them. To this end, each IoT device, in addition to the functionality normally implemented, will also provide an "attestation functionality", through which the device will be able to produce attestation reports and send them to the Trust Monitor. The latter will analyze the attestation reports and evaluate the level of integrity of the devices based on the attestation policies associated with them.

3.9 Smart Contracts integration

The Smart Contracts component is used for the validation of the data sent by the rest of the FISHY components. More specifically, the data can be:

a) the recorded security events
b) the enforced mitigation policies.

The following figure provides an overview of the Smart Contracts component sub-components and its placement in the general architecture.

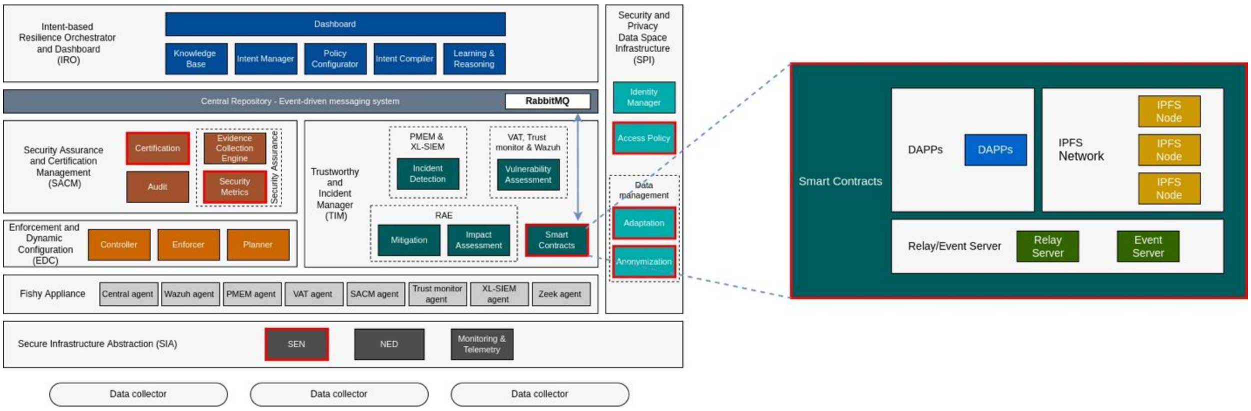

Figure 3 .Smart contracts component high-level architecture

Since the full description of the sub-components' functionality is given in the deliverable D3.3, the basic workflow will be presented, briefly.

The Relay Server reads from the Central Repository of FISHY the events/policies the components send. These data, then, are stored in an IPFS network, while the Content Identifiers (CID) that index back to the files placed in IPFS, are stored in a blockchain network. For clarity, a CID is the hash of the stored data. This approach protects against attacks that can compromise the integrity of the data, since any change will produce a new hash, making the attack easily detectable. To complete the process, the Relay Server writes back to the Central Repository its answer with details about the stored event/policy, for other components to consume and use appropriately (e.g. SACM, IRO).

FRF Integration

The Smart Contracts component can be installed either in the FRF framework or outside of it. The following segment will present an analysis of the available options and what each one has to offer to the FISHY platform and the client who utilizes the capabilities of the platform.

Deploy on client premises

In this case, the Smart Contracts component can be installed separately from the rest of the FISHY platform, and more specifically on the premises of the organization the FISHY platform will be monitoring. There are two options for the IPFS and blockchain networks:

A) Use public IPFS and blockchain networks

B) Use private IPFS and blockchain networks

Public IPFS/Blockchain

In this scenario, the Relay/Event Server of the Smart Contracts component can be installed on premises, while the IPFS and blockchain networks can be public. This allows for the client to have full access to the logs of the Relay/Event Server and monitor the requests. By using the public IPFS and blockchain networks, the client can rely on the peers of the network to validate the data.

This approach is useful if the data is deemed not sensitive and if the client does not want to install their own IPFS/blockchain network, with their resources.

Figure 4 gives a simplified view of the scenario.

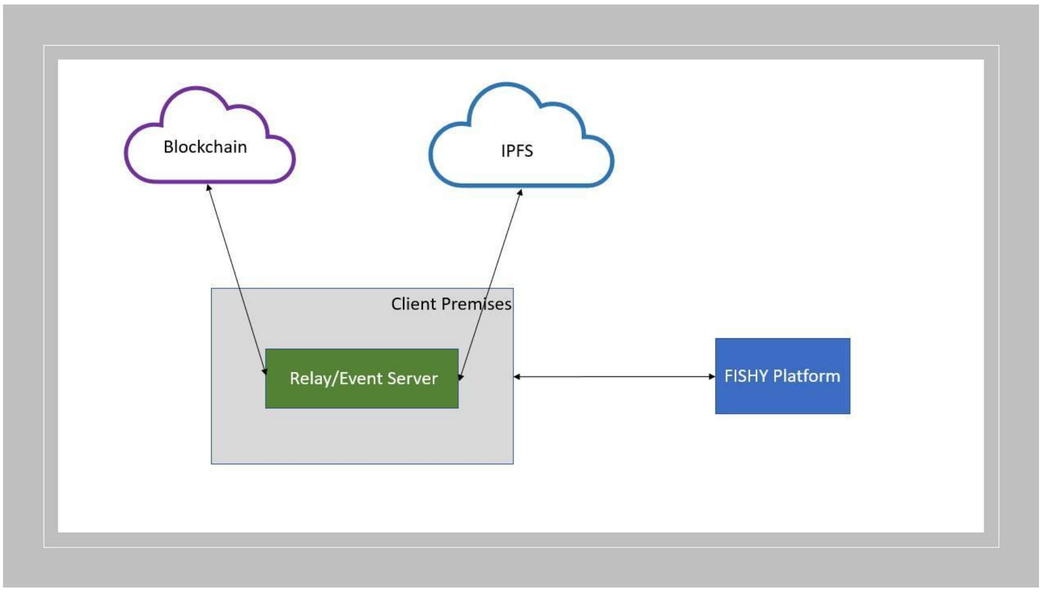

Figure 4. Smart Contracts component – Installation on premises for public IPFS/blockchain

Private IPFS/Blockchain

In this scenario, again the Relay/Event Server of the Smart Contracts component is installed on premises, as well as the IPFS and blockchain networks. The two networks are working in private mode, meaning that no external nodes will be able to participate in the validation of the data. The nodes that can join the networks are monitored by the client.

The client, apart from having full access to the Relay/Event Server, can, also, have full access to every file/transaction stored in IPFS/blockchain. This approach can prove useful if the client wants to store sensitive information and requires limited access to it.

Figure 5 gives a simplified view of the scenario.

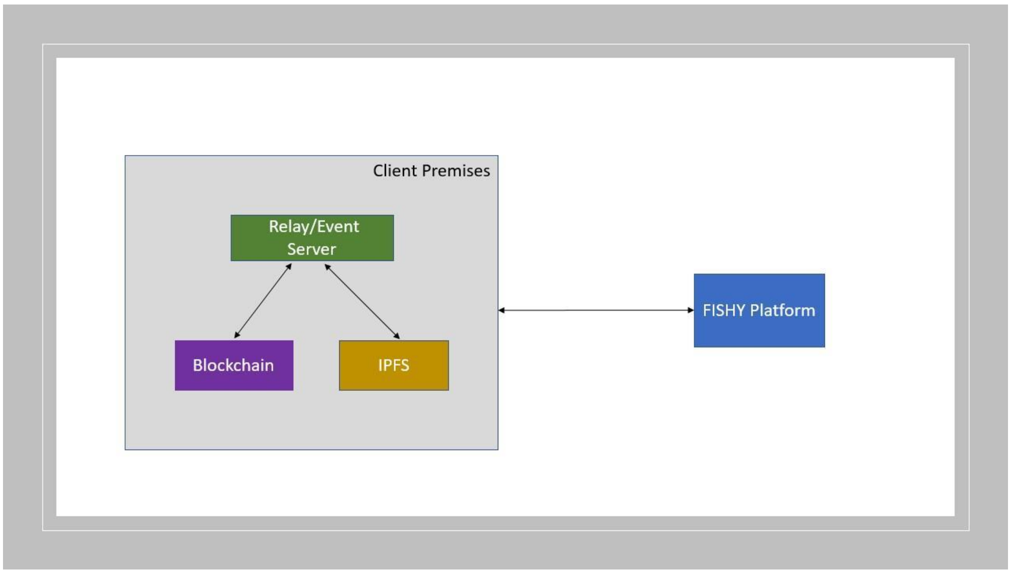

Figure 5. Smart Contracts component–Installation on premises for private IPFS/blockchain

Deployment on FRF

Another installation approach is to integrate the Smart Contracts component in the FRF framework. The FRF framework is used to deploy the FISHY components and manages the integration of the platform. Therefore, the Smart Contracts component comes prepackaged with the rest of the FISHY platform. The IPFS and the blockchain networks will be already deployed in the FRF network.

This approach has the benefit that the communication of all the FISHY components will happen internally, through a common framework. Additionally, no involvement from the client is required for the installation of the Smart Contracts component. However, the client can have limited access to the stored information in the component.

Figure 6 gives a simplified view of the scenario.

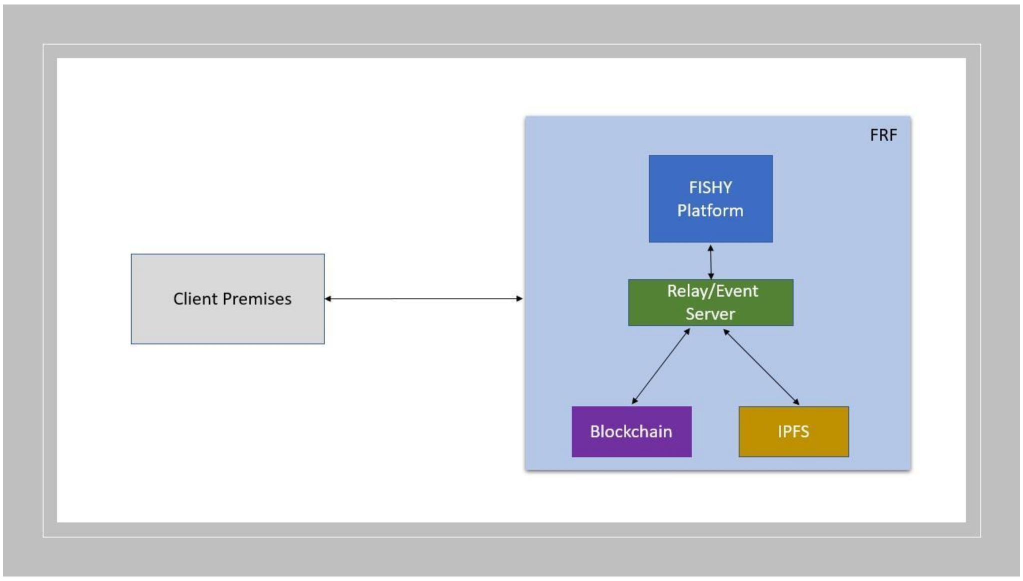

Figure 6. Smart Contracts component–Installation in FRF

3.10 Central Repository integration

In the IT-2 process, the component was renamed from Threat/Attack Repository to the Central Repository to be a more transversal rather than a TIM component only.

In this regard, in IT-1 it was strictly a WP3 component and in IT-2 it was expanded to serve as a central repository and facilitate the communication between the components (modules) of all technical WPs (WP3 – WP5). The Central repository has been deployed to the FRF and modified to enable the storage of the tool reports in the CEF format.

4 FISHY Appliance

The FISHY Appliance provides a runtime for the various tool agents and assists in tool integration and data flows. It is situated in the lower, domain level and is the first recipient of data collected from the monitored infrastructure. Elements of SIA provide secure networking between the Appliance and the private infrastructure, while the SPI provides secure connectivity with the FISHY platform.

Architecturally, the tools are split into 3 components, data collectors, agents and processors (servers where actual data analysis takes place). Data collectors reside within the monitored infrastructure and forward the collected data to their corresponding tool agents, that run on the Appliance. The collected data must then be forwarded to the tool processing servers within the FISHY Control Services via the SPI. Some tools are able to interface with the SPI directly, while others have the option of using the Appliance Agent and its REST API data collection endpoint to ease the process of integration.

The deployment of the Appliance and tool agents (at use cases) is facilitated by Ansible scripts, which allows a lot of flexibility when it comes to delivery and staging. If the target deployment environment is a virtual machine, the Appliance can be preconfigured and packaged as a VM image, or in the case of a physical device, the Ansible scripts can be used for provisioning directly.

5 Interfaces

Considering the workflow shown in Figure 7, in this section we describe the main interfaces between the components in WP3 (Trust Manager). Component per component we have the next tables describing those interfaces.

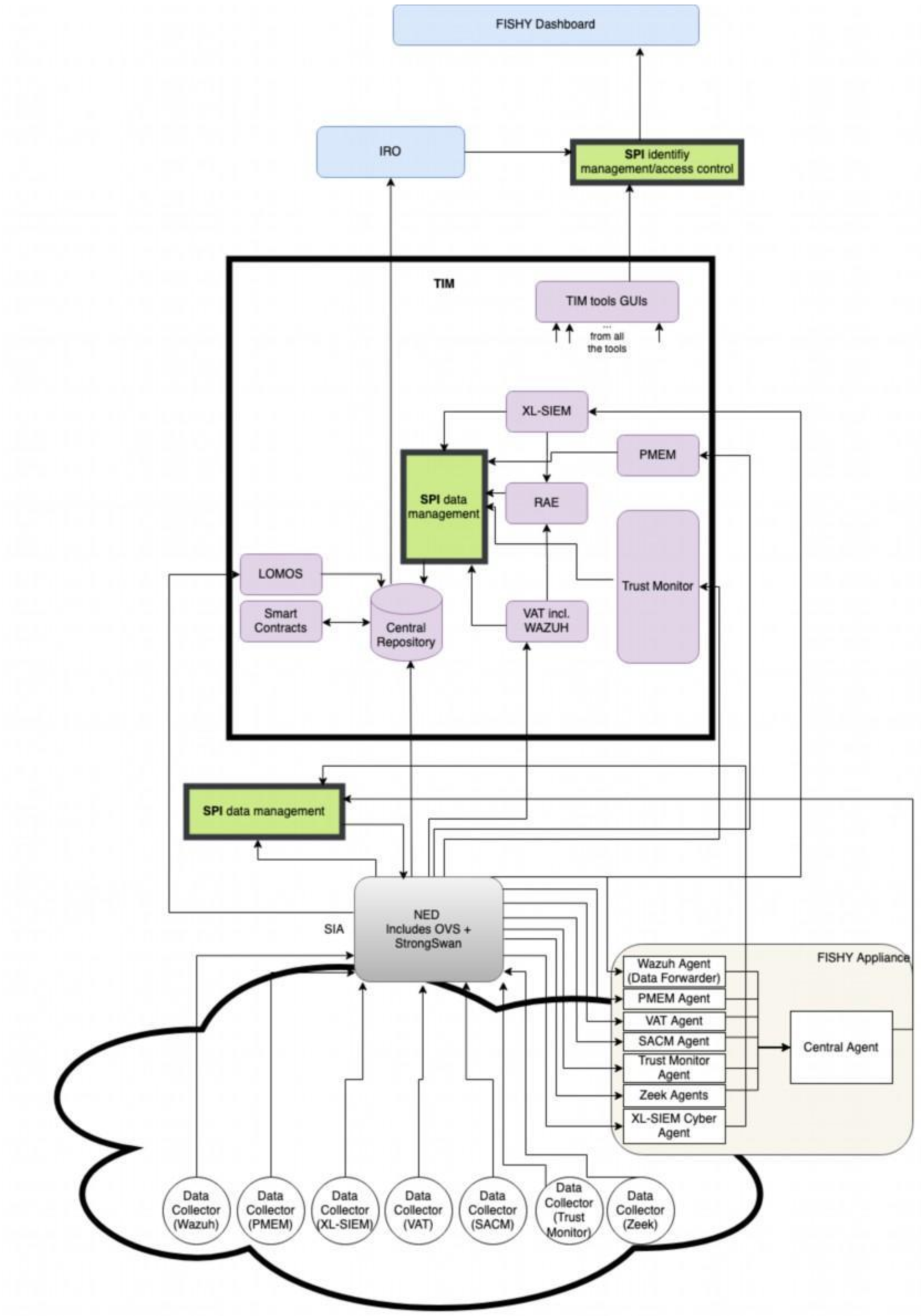

Figure 7. TM whole workflow

Table 2. Inbound interfaces

Component	Origin	Data	Type of communication	Status
XL-SIEM	NED	Data collected at the data collectors	No specific one. Interfaces *use * SIA connectivity	Ongoing
RAE	XL-SIEM	Alarms in native format	AMQP	Ongoing
RAE	VAT	Reports of detected vulnerabilities	AMQP	Not started
TIM VAT (Including WAZUH)	NED	Data collected at the data collectors	No specifc one. Interfaces *use* SIA connectivity	Currently data collectors from WAZUH send the data to the FISHY appliance and then to the WAZUH tool in the Central Services (the process of sending it through NED is ongoing)
PMEM	NED	Data collected at the data collectors	No specifc one. Interfaces *use* SIA connectivity	Currently data collectors send directly the data to the PMEM tool in the Central Services (the process of sending the data to the FISHY appliance and later through NED to the tool in the Central Services is ongoing)
Trust Monitor	NED	Data collected at the data collectors	No specifc one. Interfaces *use* SIA connectivity	Currently data collectors (attestation agents) send directly the data to the Trust Monitor tool in the Control Services domain (ongoing the process of sending the data to the FISHY Appliance and later, through NED, to the tool in the Control Services domain)

Component	Origin	Data	Type of communication	Status
Zeek	Data collectors	Network interface (Raw Traffic Data)	Network interface sniffing (Network packets are read as they arrive to the host)	Zeek currently receives traffic data by attaching to a host network interface. A port-map can be put in place in order to direct traffic to Zeek
Smart Contracts	Central repository	Receive events/policies/alerts from FISHY components	AMQP	Completed integration with RabbitMQ to receive events/policies/alerts
LOMOS	NED	Raw log data	Use case log data is stored in Elasticsearch, data collector on Appliance fetches it.	Model training and agent implementation in progress
Central Repository	SPI data managemen t	Outcome of the TIM and SACM tools in CEF format and RAW data	REST API	Currently TIM tools and SACM write directly in the Central Repository (ongoing the process of sending it through SPI data management)
Central Repository	LOMOS	Processed data with anomaly score	REST API	Integration in progress
Central Repository	Smart Contracts	Report when an event/policy is stored in IPFS and Blockchain	AMQP	Completed integration with Synelixis RabbitMQ, need to integrate with RabbitMQ of Central Repository
Central Repository	IRO	IRO generates policies to be enforced by EDC and saves them in the Central Repository.	REST API	The generated policies are automatically sent to the Central Repository
Central Repository	SACM	Reasoning results	REST API	Implementation in progress, SACM will write its reasoning results to the Central Repository
SPI data management	FISHY appliance	Raw data from data collectors and Zeek processed data	HTTPS (REST API) and/or Pub-sub, if required	Under development

Component	Origin	Data	Type of communication	Status
SPI data management	TIM tools and SACM	Processed data in tools own format to be converted in CEF format	HTTPS (REST API) and/or Pub-sub, if required	Zeek, PMEM and Trust Monitor endpoints are integrated. RAE and XL-SIEM endpoints are under-development
SPI identity management/ access control	Tools' dashboards	User credentials/Access tokens	HTTPS (API)	SPI-IDM has been deployed to the FRF. The tools have integrated their user login mechanisms with Keycloak
SPI identity management/ access control	Zeek	Client credentials/Acess token	HTTPS (API)	SPI-IDM has been deployed to the FRF. Zeek has integrated it's communication with the FISHY appliance agent with Keycloak's authentication
FISHY Appliance	Data collectors	Raw data from data collectors and Zeek processed data	Data collectors interface with agents on the appliance in their own formats and interfaces, connectivity is facilitated by SIA.	Interfacing of data collectors and agents is done in isolation, agent integration into t he Appliance is in progress
FISHY Appliance	NED	Raw data from data collectors and Zeek processed data if data collectors are not in the same domain as FISHY appliance it is necessary to go through NED (SIA)	No specific one. Interfaces use SIA connectivity	Interfacing of data collectors and agents is done in isolation, agent integration into t he Appliance is in progress

Table 3. Outbound interfaces

Component	Destination	Data	Type of communication	Status
XL-SIEM	SPI data management	Alarms generated CEF format	AMQP	Ongoing
RAE	SPI data management	Risk report in native format and CEF format	AMQP/REST API	Ongoing
TIM VAT (Including WAZUH)	SPI data management	Alarms generated CEF format	AMQP /REST API	Currently VAT/WAZUH writes directly in the Central Repository (the process of sending it through SPI data management is ongoing)
TIM VAT (Including WAZUH)	RAE	Reports of detected vulnerabilities	AMQP	Not started
PMEM	SPI data management	Detected attack: Type and time stamp	REST API	Currently PMEM writes directly in the Central Repository (the process of sending it through SPI data management is ongoing)
Trust Monitor	SPI data management	Attestation reports on enterprise infrastructure	AMQP /REST API	Currently Trust Monitor writes integrity reports in the Central Repository through the SPI data management
Zeek	FISHY Appliance	Alarms generated by Zeek	HTTPS (REST API)	Zeek agent has been developed and is waiting to be deployed in the fishy appliance in order to forward data to the SPI
Smart Contracts	Central repository	Report when an event/policy is stored in IPFS and Blockchain	AMQP	Completed integration with Synelixis RabbitMQ, need to integrate with RabbitMQ of Central Repository
LOMOS	Central repository	LOMOS analyses the log data and assigns it an anomaly score. Entries that surpass an anomaly threshold are stored in Central Repository.	REST API	Central Repository integration in progress

Component	Destination	Data	Type of communication	Status
Central Repository	IRO	Reports and events from Smart Contracts	AMQP	IRO consumes reports from different tools and events from Smart Contracts
Central Repository	Smart Contracts	Receive events/policies/alerts from FISHY components	AMQP	Completed integration with Synelixis RabbitMQ, need to integrate with RabbitMQ of Central Repository
SPI data management	Central repository	Raw data in CEF, with pseudoanonimization, when required, and processed data from TIM tools and SACM in CEF	REST API	Process of adding the CEF data model to the central repository is ongoing. Once it's added, SPI-DM will post the normalized data
SPI identity management/ access control	Tools' dashboards	Access tokens & Verification	HTTPS (API)	SPI-IDM has been deployed to the FRF. The tools have integrated their user login mechanisms with Keycloak
SPI identity management/ access control	Zeek	Access token & Verification	HTTPS (API)	SPI-IDM has been deployed to the FRF. Zeek has integrated its communication with the FISHY appliance agent with Keycloak's authentication
FISHY Appliance	SPI data management	Raw data in CEF, with pseudoanonimization, when required, and processed data for Zeek	HTTPS (REST API) and/or Pub-sub, if required	Under development
FISHY Appliance	TIM tools and SACM	Raw data	Different kind of communication including AMQP, etc.	Interfacing of data collectors and agents is done in isolation, agent integration into the Appliance is in progress

6 Conclusions

This deliverable provides the description of the integration for the second and final iteration of the project of the Trust Manager (TM) module. The current implementation of the different blocks of SPI and TIM is described covering all the functionalities envisioned for FISHY, some of them added for this second iteration of the project.

The adaptations and specific deployments for the use case of the tools matching these functionalities are also described, as well as the status of integration of these tools in the FISHY Reference Framework.

Finally, a detailed description of all the interfaces between the different blocks/tools of SPI and TIM is provided, also including the current status of deployment of these interfaces.

This deliverable D3.4 is written in parallel with D4.4 for WP4, and the output of these two deliverables will feed the final deliverable of integration D5.2. This parallel work describes the functionalities, the deployment in the use cases and the integration of all the blocks of FISHY, except those of WP5; as well as specifies deeply the interfaces between them. For this reason, the output of these two deliverables will help the final integration of FISHY to be reported in D5.2, and also the final deployment in the use cases in D6.4.

References

[1] **[FISHY] - *D3.3 Trust Manager components design and implementation***. *(IT-2)* Henrique Santos, André Oliveira. 2022.

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

[3] **[FISHY] -*D3.2 Trust Manager IT1 integration***. *Eva Marín Tordera 2021*