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

D3.1 Trust Manager components design and implementation (IT-1)

Document Identification			
Status	Final	**Due Date**	31/05/2021
Version	1.0	**Submission Date**	30/06/2021

Related WP	WP3	**Document Reference**	D3.1
Related Deliverable(s)		**Dissemination Level (*)**	CO
Lead Participant	TID	**Lead Author**	Diego López, Antonio Pastor, Luis Conteras
Contributors	TID, SYN, XLAB, UPC, UMINHO, ATOS, POLITO	**Reviewers**	Daniele Canavese, POLITO
			Eva Marín, UPC

Keywords:

Trust, authentication, privacy, access control, data infrastructure, threat repository, detection, mitigation, attestation

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.

Document Information

List of Contributors	
Name	Partner
Diego R. López, Antonio Pastor, Luis Contreras	TID
Daniele Canavese, Cesare Cameroni, Ignazio Pedone	POLITO
Ales Cervinec	XLAB
Henrique Santos	UMinho
Jose Javier Vicente Mohino	ATOS
Xavier Masip, Eva Marín, Sergi Sánchez	UPC
Nelly Leligou, Panos Trakadas	SYN

Document History			
Version	Date	Change editors	Changes
0.1	03/03/2021	Diego R. López, Antonio Agustín Pastor (TID)	ToC and initial structure
0.2	27/04/2021	Diego R. López, Antonio Agustín Pastor (TID)	First round of contributions (UPC, UMinho, Xlab. ATOS)
0.3	05/05/2021	Diego R. López, Antonio Agustín Pastor (TID)	Second round of contributions
0.4	17/05/2021	Diego R. López, Antonio Agustín Pastor (TID)	After first complete review by editors
0.5	26/05/2021	Diego R. López, Antonio Agustín Pastor (TID)	Ready for new round of inputs, after agreement on contents for design and implementation sections
0.6	27/06/2021	Diego R. López, Antonio Agustín Pastor (TID)	Ready for project internal review
0.7	29/06/2021	Diego R. López, Antonio Agustín Pastor (TID)	Final version for Quality Assessment
1.0	29/06/2021	Jose Francisco Ruiz (Atos) Juan Alonso (Atos)	Quality assessment and final version to be submitted.

Quality Control		
Role	Who (Partner short name)	Approval Date
Deliverable leader	Diego López, Antonio Pastor, Luis Contreras (TID)	27/06/2021
Quality manager	Alonso, Juan Andres (ATOS)	30/06/2021
Project Coordinator	Ruiz, Jose Francisco (ATOS)	30/06/2021

Table of Contents

Document Information 2
Table of Contents 3
List of Tables 5
List of Figures 6
List of Acronyms 7
Executive Summary 9
1 Introduction 10
1.1 Purpose of the document 10
1.2 Relation to other project work 10
1.3 Structure of the document 10
2 FISHY platform architecture 11
3 Design 14
3.1 Security & Privacy Data Space Infrastructure 15
3.1.1 Access Control 15
3.1.2 Identity Manager 18
3.1.3 Access Policy 19
3.1.4 Data Management 21
3.2 Trust & Incident Management 24
3.2.1 Vulnerability Assessment 25
3.2.2 Incident Detection 27
3.2.3 Impact Assessment 30
3.2.4 Mitigation 31
3.2.5 Threat/Attack Repository 32
3.2.6 Smart Contracts 33
3.3 Metric Gathering 34
4 Implementation 36
4.1 Security & Privacy Data Space Infrastructure 36
4.1.1 Identity Manager and Access Policy 36
4.1.2 Data Infrastructure 38
4.2 Trust & Incident Management 39
4.2.1 Vulnerability Assessment 39
4.2.2 Incident Detection 41
4.2.3 Impact Assessment 43
4.2.4 Mitigation 43
4.2.5 Threat/Attack Repository 43
4.2.6 Smart Contracts 44
4.3 Metrics Gathering Tools 45
4.3.1 Nagios 45
4.3.2 OSSEC 46
4.3.3 XL-SIEM 46
4.3.4 Wazuh 47

4.3.5 VAT (XLAB) 48
5 Conclusions 49
References 50

List of Tables

Table 1: Summary of TM internal architectural blocks and modules 14
Table 2: List of DAPPs provided by the SC module 44

List of Figures

Figure 1. Initial FISHY architecture — 11
Figure 2. Action areas in FISHY — 12
Figure 3. Mapping between FISHY components and action areas — 13
Figure 4. General SPI workflow — 15
Figure 5. Client Credentials Flow — 16
Figure 6. Authorization Code Flow — 17
Figure 7. Resource Owner Password Flow — 17
Figure 8. Policy-based Architecture — 20
Figure 9. Authorization Protocol Flow — 21
Figure 10. Data Infrastructure architecture — 23
Figure 11. Data Infrastructure Workflow — 24
Figure 12. Sending metrics to TIM — 24
Figure 13. Vulnerability assessment, exemplified with RAE — 25
Figure 14. Format of the vulnerability report by RAE — 25
Figure 15. Remote attestation workflow — 28
Figure 16. High-level Trust Monitor architecture — 29
Figure 17. Impact assessment, exemplified with RAE — 31
Figure 18. ML-based mitigation workflow — 32
Figure 19. High-level workflow involving the Threat/Attack repository — 33
Figure 20. FISHY Smart Contract sub-modules — 33
Figure 21. Realm management architecture — 37
Figure 22. Initial Data Fabric Node architecture — 38
Figure 23. Inputs and outputs of RAE — 39
Figure 24. Architecture of the XL-SIEM — 42
Figure 25. Example of security events graphic provided by XL-SIEM — 47
Figure 26. Example of a VAT report — 48

List of Acronyms

Abbreviation / acronym	Description
ABAC	Attribute-based access control
AC	Access control
AS	Authorisation Server
AT	Access token
CMM	Capability maturity model
DAC	Discretionary access control
DAPP	Distributed application
DI	Digital identity
DL	Distributed ledger
DoS	Denial of Service
EDC	Enforcement & Dynamic Configuration
IdM	Identity management
IdAM	Identity and Access Management)
IRO	Intent-based Resilience Orchestrator & Dashboard
JWT	JSON Web tokens
M2M	Machine to machine
MAC	Mandatory access control
OIDC	OpenID Connect
OrBAC	Organization-based access control
OSSEC	Open Source HIDS SECurity
RA	Remote Attestation
RAE	Risk Assessment Engine
RBAC	Role-based access control
RoT	Root-of-Trust
RP	Relying Party
RS	Resource Server
SAML	Security assertion markup language
SC	Smart contract
SCM	Security & Certification Manager
SDLC	Software development lifecycle

Abbreviation / acronym	Description
SIA	Secure Infrastructure Abstraction
SSEDIC	Scoping the Single European Digital Identity Community
SPI	Security & Privacy Data Space Infrastructure
TIM	Trust & Incident Manager
TM	Trust Manager
TMon	Trust Monitor
TPM	Trusted Platform Module
UCON	Usage control
UE	User equipment
XACML	Extensible access control markup language

Executive Summary

This document constitutes the first step in the generation of the Trust Manager (TM), one of the four main functional elements of the FISHY architecture. In general terms, the TM is the element in charge of collecting and forwarding evidence within the FISHY decision cycle. It processes the data made available by the *infrastructure abstraction* and by specific metric gathering tools, applying privacy principles and normalizing the data flows so they become suitable for the analytics elements that assess the security of the devices, components and systems in the supply chain controlled by the FISHY framework.

The TM is divided into two blocks, the Trust & Incident Manager (TIM) and the Security & Privacy Data Space Infrastructure (SPI). While the SPI is in charge of the collection, forwarding and storage of data, the TIM incorporates the security assessment mechanisms.

This document puts the blocks constituting the TM in the context of the whole framework, identifying the modules constituting them, describing their functional characteristics and required interfaces, and discussing the relevant workflows in which they are involved. As part of a practical approach, the applicable tools identified by the project team to implement the discussed functionalities are described, including the related features, and the necessary adaptations to interface them within the TM environment and with the rest of the FISHY framework.

1 Introduction

1.1 Purpose of the document

This document provides the initial design of the FISHY Trust Manager (TM), aligned with the first project development iteration. It addresses the characteristics of the two blocks identified within the TM, namely the Trust & Incident Management (TIM) and the Security & Privacy Data Space Infrastructure (SPI), along with their internal components.

It considers the functional aspects regarding data flow and identity management, applying privacy-by-design practices and incorporating mechanisms for monitoring data normalization and forwarding. The mechanisms for vulnerability and risks analysis, prediction and estimation and the incident detection and mitigation strategies are considered as well.

1.2 Relation to other project work

The initial TM design is tightly coupled with the design of the other FISHY functional elements in the context of the first development iteration, and especially with the Security and Certification Manager (SCM) element in WP4, particularly regarding the proper design of the interfaces connecting WP3 and WP4 developments that will be integrated in WP5, together with the *upper* (intent) and *lower* (infrastructure) layers enabling the application of the FISHY framework.

1.3 Structure of the document

The document is structured in the following chapters.

Chapter 2 introduces the overall principles of the FISHY architecture and puts the TM in its context.

Chapter 3 addresses the initial design of the TM, describing the design of each one of the blocks in the TM architecture, including their interfaces and relevant workflows.

Chapter 4 describes the base tools to be applied and how they have to be adapted to be applied in the FISHY framework.

Finally, **Chapter 5** provides some conclusions about the results reported in this document.

2 FISHY platform architecture

The FISHY platform architecture is being developed in task 2.3 of WP2, and a deliverable, D2.2, will be release in M12 with the FISHY architecture for IT-1. However, a preliminary version of the FISHY architecture has been described in the Internal report on the Architecture for IT-1 in M7 (MS7).

This preliminary version of the FISHY architecture is based on the architecture presented in the FISHY proposal and shown in Figure 1. The FISHY platform is divided into four main blocks: Trust Manager (TM), Security & Certification Manager (SCM), Intent-based Resilience Orchestrator & Dashboard (IRO) and Secure Infrastructure Abstraction (SIA).

The Trust Manager (TM) is designed and implemented in WP3 and will be described in detail in this deliverable. In summary, this component is in divided into two blocks, the Trust & Incident Manager and the Security & Privacy Data Space Infrastructure. The Trust & Incident (TIM) manager provides the tools for assessing the security of the stakeholder's device, component or/and system. The Security and Privacy Data Space Infrastructure (SPI) is responsible for the collection and storage of data generated from the devices, processes and components of the stakeholders' ICT systems being part of the supply chain.

WP4 is in charge of designing and implementing the two blocks within the SCM, namely, Enforcement & Dynamic Configuration (EDC) block which is responsible for making the entire system cyber resilient, even when including potentially insecure components, based on the concepts of dynamic self-configuration. The Security Assurance and Certification Management (SCM) which is responsible for the provision of auditable, evidence-based evaluation and certification of the assurance posture of complex ICT systems, based on identified security claims and metrics, setting the roots for the definition of a pan-European process for certification of devices, processes and systems, as required in today's in the European market.

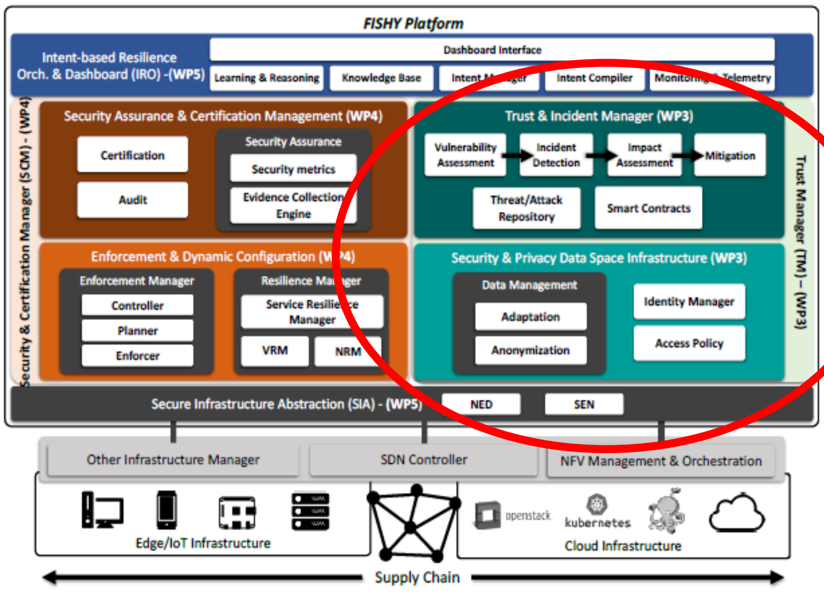


Figure 1. Initial FISHY architecture

Finally, in WP5, the components in the architecture shown in the top and the bottom of Figure 1 will be developed. The Intent-based Resilience Orchestrator and Dashboard (IRO) block, shown at the top of the figure, is designed to work as the user-centric interface to translating and to orchestrating input actions into intents, to be used by other components. And the Infrastructure Abstraction (SIA) is the infrastructure-centric interface and works as a data interface between different Edge/IoT or Cloud infrastructures and FISHY platform.

With the help of the use cases needs and requirements, the work in T2.3, in these first months of the project, has been the mapping between the described blocks and the different areas where FISHY will be deployed. These action areas are described as follows and shown in Figure 2:

- Organization: defined as the company, the supply chain consortium, the law enforcement, etc.
- Realm: defined as the environment from cybersecurity perspective, with same policies, rules, etc.
- Domain: a group of assets with certain relationship (same network, infrastructure, location, etc.)

In Figure 2, only different domains inside realm 1 are shown, but in general in a realm there can be one or more domains, and in each organization we can have only a realm or more than one realm. Even, the whole supply of chain can be composed by different organizations, the example in Figure 2 contains two possible organizations.

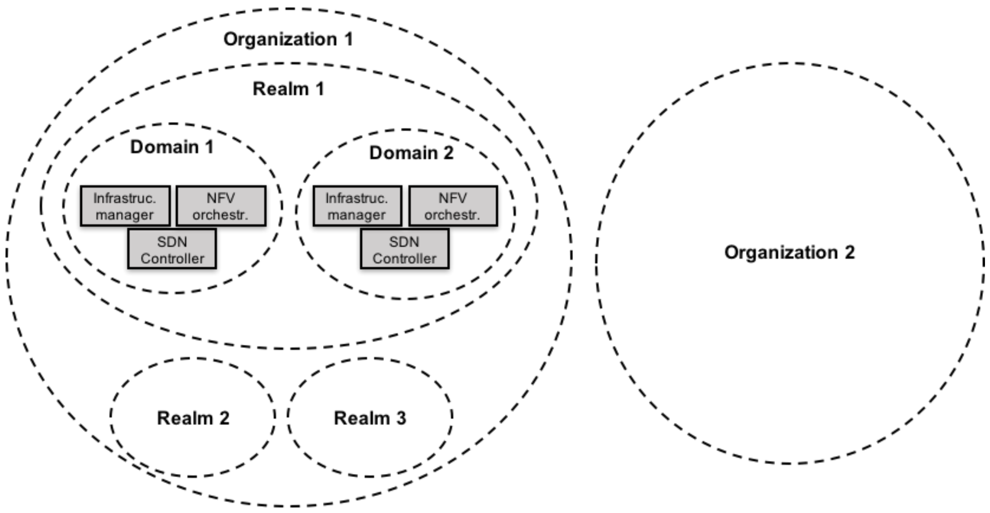


Figure 2. Action areas in FISHY

This mapping between FISHY components and action areas in FISHY is shown in Figure 3, where we observe that there are some components, such as the IRO, the TIM and the SCM, that are logically centralised. We consider that these centralized components will be a third-party service located at cloud (FISHY control services). Whereas there are other components, such EDC, SPI and SIA that will be replicated in each one of the domains, because they are directly connected to the infrastructure.

With this deployment of FISHY components in the different organizations, realms, and domains, and taking into account the blocks in WP3, the TIM will be centralized, instead the SPI will be deployed in each one of the domains.

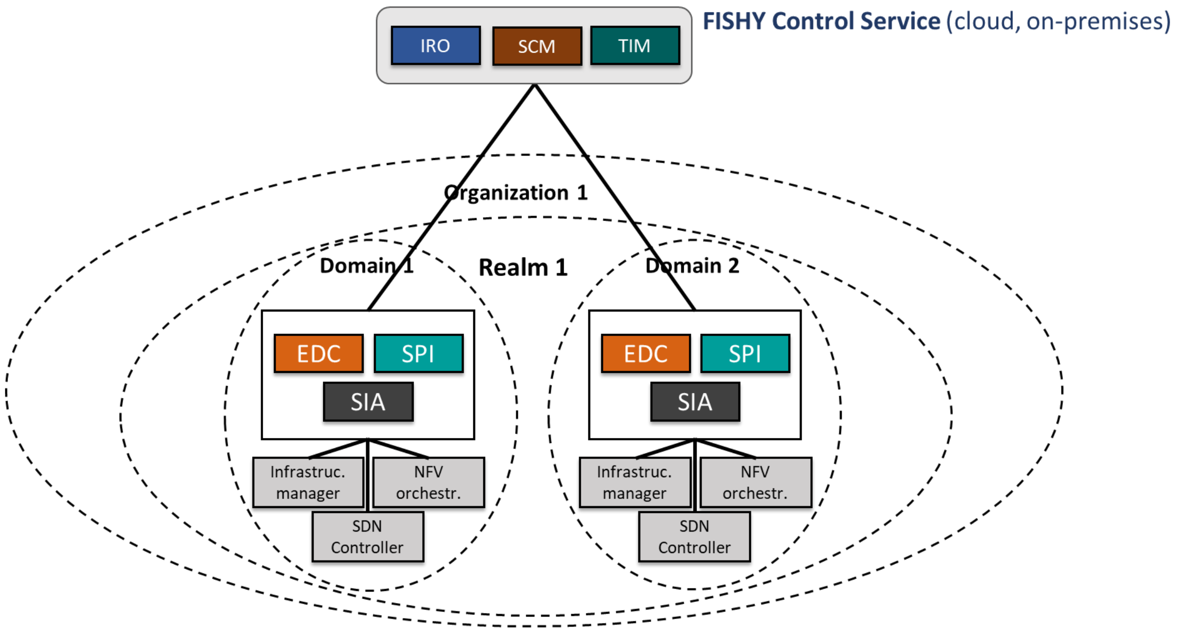


Figure 3. Mapping between FISHY components and action areas

3 Design

The next table presents a summary of the internal blocks and modules for Trust Manager. Each of them is detailed in following subsections, and in the corresponding ones in Section 4.

Table 1: Summary of TM internal architectural blocks and modules

Block	Module	Functionality	Interface with	Workflow	Tools
SPI	Identity Manager		Any internal FISHY resources	Figure 5 Figure 6 Figure 7	OpenID Connect
SPI	Access Policy				XACML – JSON Profile
SPI	Data Management / Adaptation	Access management. Syntax and access method adaptation	All potential data sources Data consumers: IRO, SCM, TIM	Figure 11	5Growth SDA, NGSI-LD Context Broker, Kafka, NiFi, Flink
SPI	Data Management / Anonymization	Personal data filtering and/or obfuscation			
TIM	Vulnerability Assessment			Figure 14	RAE
TIM	Incident Detection				XL-SIEM
TIM	Attestation	Infrastructure remote attestation	IRO, SIA, and infrastructure	Figure 16	Trust Monitor
TIM	Impact Assessment		Figure 18	Figure 17	RAE
TIM	Mitigation	Mitigation mechanisms based on ML algorithms which based on patterns of behaviours detects anomalies and classifies them.	IRO, Threat repository	Figure 19	PMEM
TIM	Threat/Attack Repository			Figure 20	Relational database Pub-sub
TIM	Smart Contracts				Ethereum

3.1 Security & Privacy Data Space Infrastructure

The SPI module aims to provide an interface between the low-level components (implemented within SIA) producing data related to metrics in general, and the higher-level modules that use those metrics (implemented within TIM and SCM). According to the analysis already done over the use cases and the requirements of the tools involved, we envisage the possibility of having two different types of low-level devices: those that are controlled by FISHY itself (implementing security functions specified EDC and enforced by SIA), which we refer as white boxes (WBox); and those that are not controlled by FISHY, allowing only monitoring, which we refer as black boxes (BBox) – Figure 4 shows the general workflows for both cases. Concerning the BBox case, the monitoring tools can be deployed in the infrastructure itself, or at the SIA level. The data interface (DInt) needs to handle any required data transformation (including format normalization and anonymization, when applicable – performed by DMng) as well as the enforcement of the Access Control rules established (performed by AC_IdM). The next subsections go into more detail in describing how these blocks interact for forwarding the relevant data, applying the appropriate access control mechanisms, and acting to preserve privacy.

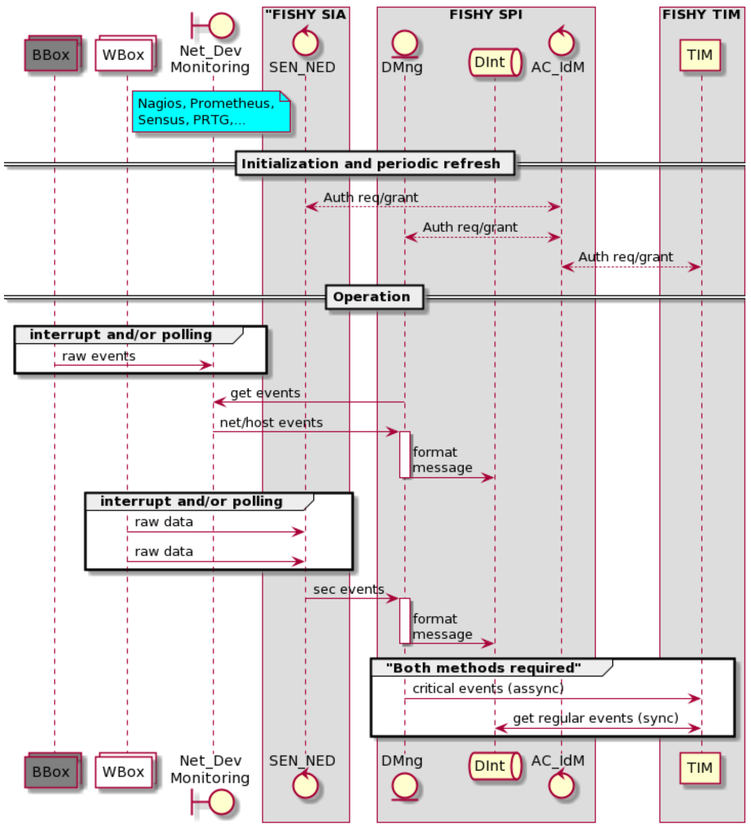


Figure 4. General SPI workflow

3.1.1 Access Control

FISHY is (will be) a modern distributed application. This means it will be highly modular, requiring a carefully designed Access Control (AC) unit. Among the available solutions OpenID Connect (OIDC) [1] is a de-facto standard widely used in similar software architectures, based on RESTful technology. It basically consists of a centralized authorisation server implementing the OAuth2 standard, complemented by an authentication layer based on the OpenID standard.

OpenID Connect implements a centralized authentication system, initially requiring servers and clients (software modules) registration, when unique ID and shared key are generated. The protocol specifies the mechanisms to get Access Tokens (AT), which are tickets to allow access to software services. Those mechanisms are referred to as Flows. The OpenID protocol is used to enforce authentication on those flows through an extra set of operations. Each flow corresponds to a possible scenario involving (all or in part):

- A **user** (data owner).
- A **client** (as **Relying Party** -- RP) that needs to access the data on behalf of the user, but that does not know him/her.
- The **Authorisation Server** (AS).
- Another **client** that provides data (**Resource Server** - RS)

Among all possible flows, in the case of the FISHY project and taking preliminary discussions on the requirements, there are three main flows to consider:

- **Client Credential Flow** – when the client is the "resource owner" (no need to get user authorization). This is a typical machine-to-machine (M2M) scenario, which is the case when dealing with CLIs, daemons, or services running on the backend (see Figure 5).
- **Authorisation Code Flow** – this is the typical case when the client is a server-side application. The user (typically through a browser or App) first triggers the login process within the client, which calls AS to authenticate the user and to validate the request. Eventually the AS will redirect the call to a third-party authentication system (when using a Federated Identity provider). In case of success, the AS returns an **Authorisation Code** to the client. Next, the client will ask the AS to provide the **Access Token**, passing its ID and secret key - that is why this flow should only be used with server-side clients since their code and the key will never be exposed - (see Figure 6).
- **Resource Owner Password Flow** – a special case of the previous one, when it is not possible (or desirable) to redirect the user for authentication and there is absolute trust in the client capacity to keep (safely) user credentials. This is frequently considered a poor security practice, but may provide a very effective programming practice, if client trust is very high. When dealing with IoT and machine-to-machine architectures (which is the case with FISHY), sometimes this is the only possible way to deploy AC. But it should never be an option when dealing with third-part clients (see Figure 7).

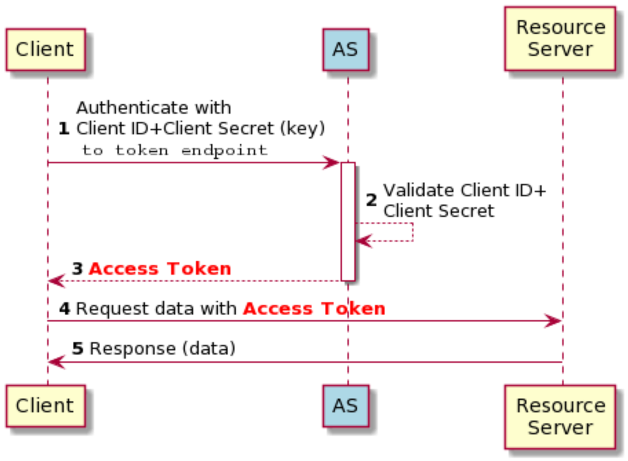
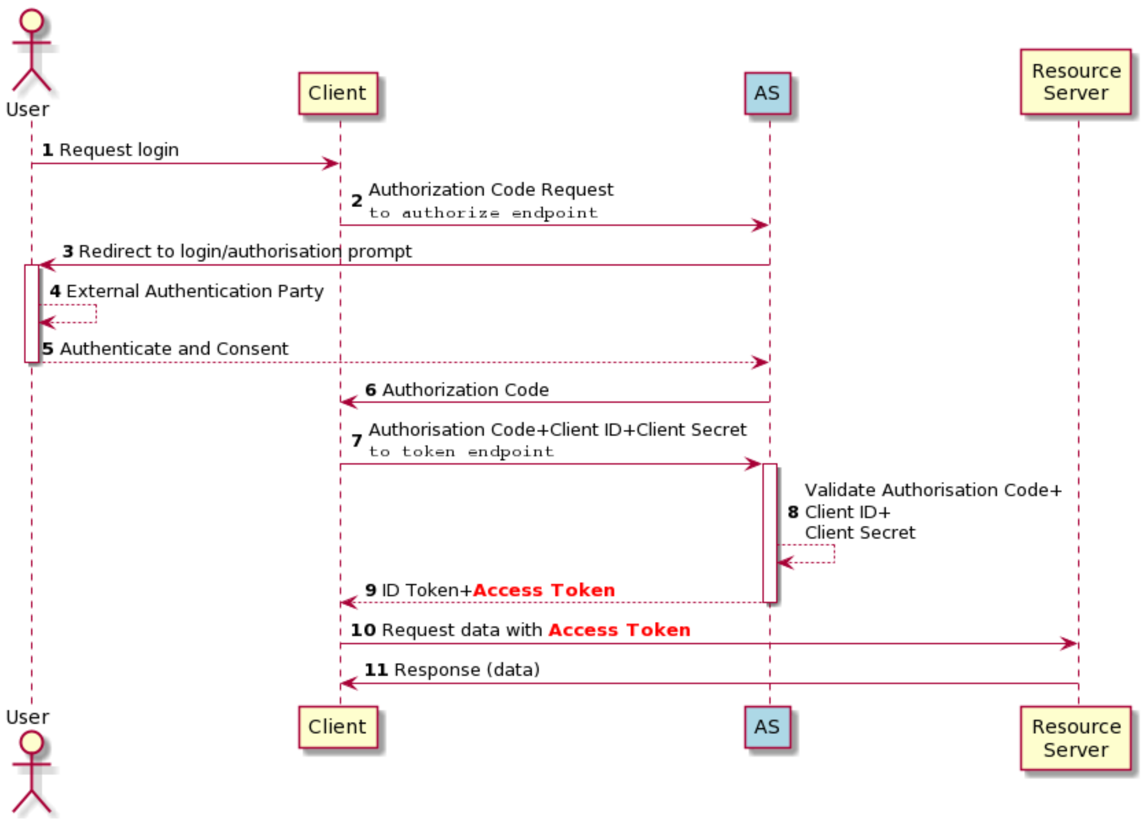
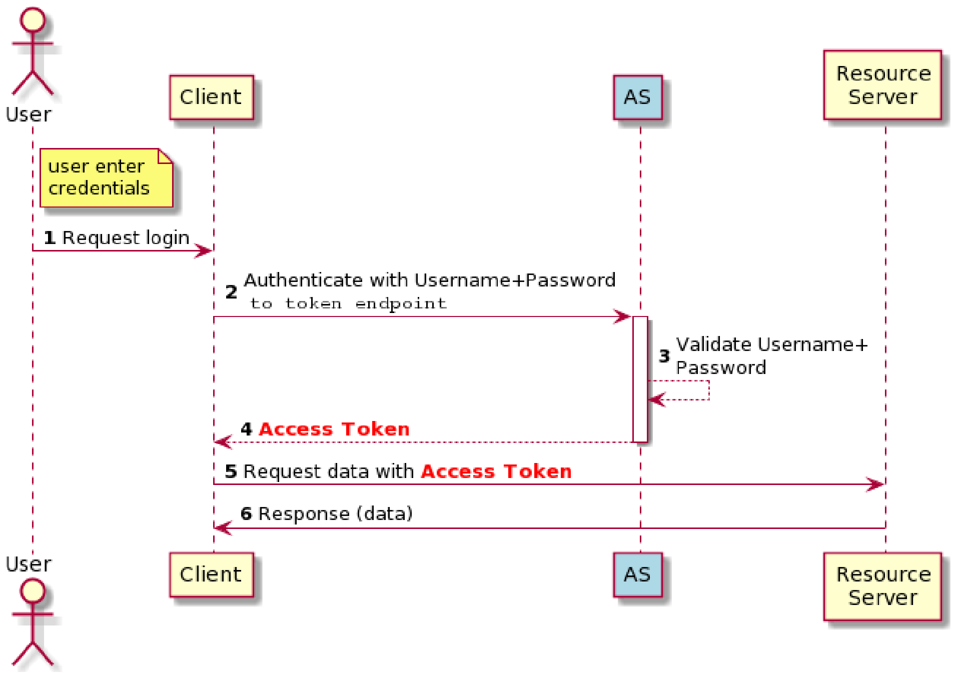

Client
AS
Resource Server
1 Authenticate with Client ID+Client Secret (key) to token endpoint
2 Validate Client ID+ Client Secret
3 Access Token
4 Request data with Access Token
5 Response (data)
Client
AS
Resource Server

Figure 5. Client Credentials Flow



User
Client
AS
Resource Server
1 Request login
2 Authorization Code Request to authorize endpoint
3 Redirect to login/authorisation prompt
4 External Authentication Party
5 Authenticate and Consent
6 Authorization Code
7 Authorisation Code+Client ID+Client Secret to token endpoint
8 Validate Authorisation Code+ Client ID+ Client Secret
9 ID Token+Access Token
10 Request data with Access Token
11 Response (data)

Figure 6. Authorization Code Flow



User
Client
AS
Resource Server
user enter credentials
1 Request login
2 Authenticate with Username+Password to token endpoint
3 Validate Username+ Password
4 Access Token
5 Request data with Access Token
6 Response (data)

Figure 7. Resource Owner Password Flow

All information transactions pertaining to the Identity Management operation are performed using JWT (JSON Web Tokens) [2], which is a data structure widely spread by the software community to provide authentic (signed) claims between parties, specifically when using RESTful solutions.

But any AC solution requires as well proper Data Management and policy enforcement functions. Data Management must assure data is formatted and categorised in the correct way, according to system requirements. Following good practices and standards, data should be classified by critically and a domain-origin dimensions, besides other possible attributes. Those attributes are essential to define privacy related functions and access rules, which will be detailed in the next subsections.

3.1.2 Identity Manager

Within the Cyberspace, Digital Identity (we will refer to it just as ID), embodied by our credentials and associated characteristics, has taken on an increasingly important role. Access Control is the gateway to this world. But unlike the real world, in Cyberspace we can (and usually do) have multiple IDs. Those IDs may share some attributes with real personas, but can also create completely different personalities, according to the context and including privacy concerns, among other possible motivations. But in the end, behind all IDs, there are real subjects who will be accountable for what their IDs do (hopefully!).

With the increased criticality of all activities in Cyberspace, ID management (IdM) -- also referred by IdAM (Identity and Access Management) -- has become a priority. Standardization organizations have produced frameworks for this area [3]. In Europe some initiatives have been promoted, such as the Scoping the Single European Digital Identity Community (SSEDIC) [4] -- also to propose a framework for the management of IDs to be applied in all European countries --, the FutureID[1] infrastructures project, or the PICOS[2] project, and not forgetting the eIDAS regulation. And even NATO, through its Information Assurance Product Catalogue (NIAPC), created a Security mechanism Group (SG05)[3] dedicated to this topic. All these activities were also fuelled by the increase in cybercrimes related to identity theft, fraud, and privacy breaches, and its impact on citizens, in general.

The work around those frameworks generated some new concepts, besides those of ID and IdM:

- **Service Provider** (SP), is an organization that provides an information service over the Internet; with more or fewer requirements, the SP demands an authenticated ID before delivering the service.
- **Digital Identity Provider** (IdP), is a specific service provider, which handles user authentication for several users and SPs. There are several ways of delivering this service, mainly concerning the way authentication is performed, and four models emerged along the research done:
    - **Credential Identity Service**, are those using as credentials some formal resource, like certificates.
    - **Identifier Identity Service**, are those using any user identification, such as the username, an email address, an ID card number, or something equivalent.
    - **Attribute Identity Service**, are those using any type of attribute that describes the user identity, like residence address, age, contact information.
    - **Pattern Identity Service** (less frequent), are those using patterns, usually related to user reputation or recognizance from others (humans or systems), like honour, trust records, or history access records.

  We may find systems using more than one type of identifier, of course. Sometimes it is not easy to map one of the above types. Anyway, the decision about what to use and why, should always be based on the trust and security levels within the target environment.

Besides the type of identity information used, an IdM can also be categorized by the implementation model. The chosen model has a substantial impact on architectural decisions concerning the development of an Information System that uses an IdM:

- **Isolated Model**, the SP and IdP functions are kept together in one server, and there is no sharing across domains. This is the simplest case (and more frequent), where administrators

[1] https://cordis.europa.eu/project/id/318424 [3]
[2] http://picos-project.eu
[3] See also https://www.ia.nato.int/niapc/SecurityMechanismGroup/Identity-Management-and-Access-Protection

have full control over IdM operation. The biggest drawback is forcing users to have specific credentials for each SP they access. As already mentioned, this situation pushes users to use the same credentials in several SPs and to choose simple passwords that are easy to remember. Clearly, this model has scalability issues.

- **Centralized Model**, the SP and IdP functions are separated, and credentials are stored in the IdP. But they are both local, and there is no cross-domain sharing. This way, the Centralized Model share with the Isolated Model the same advantages and drawbacks, except the possibility to use the same credentials to all SPs that are local. The classic example is the Kerberos system.
- **Federated Model**, contrary to the above models, this one aim to address the cross-domain operation. It uses protocols and standards to establish agreements between groups of SPs and a remote and independent IdP (operated by a third party). There are a number of well-established technologies to support the Federated Model [5].

3.1.3 Access Policy

Traditionally, Access Policies are developed with the purpose of defining a set of conditions that should be satisfied to grant a subject the access to a desirable object. When designing a system for access control, three elements should be considered [6]:

- **Security policy**: it defines the general rules governing all information access requests.
- **Security model:** it provides a formal representation of the access control security policy and how it works.
- **Security mechanism**: it defines the low-level (software and hardware) functions that implement the controls imposed by the policy and formally stated in the model.

Regarding to access control models, the available literature refers several different models, each one with specific characteristics and focus. Working with these models, it is possible to implement attribute-based access controls (ABAC), discretionary access control (DAC), mandatory access control (MAC); role-based access control (RBAC); organization-based access control (OrBAC); usage control (UCON), besides some other variants that explore specific subject's characteristics and rules enforcement points.

The implementation of these models should take into consideration the architecture designed and at the same time be supported by a standard as a guide for best practices. In this approach, three implementation possibilities arise, an architecture based on XACML, an architecture based on token, or a hybrid architecture that can some the best of both worlds.

The EXtensible Access Control Markup Language (XACML) [7] is an open standard for access control architectures, responsible for the management of rights, evaluation, and the enforcement of access policies. XACML traditionally is based on ABAC model which means that the attributes of entities will be used to authorize or reject an access. This architecture is characterized by four main components[8]:

- Policy Enforcement Point (PEP) provides an interaction system and is responsible for enforcing access decisions.
- Policy Decision Point (PDP) evaluates access re-quests against access control policies and determines whether access should be granted or denied.
- Policy Administration Point (PAP) acts as a policy repository and provides resources for policy management.
- Policy Information Point (PIP) denotes the source of information (for example, context information) needed for policy evaluation.

As it can be seen in the figure below, PAP provides policies for the PDP (1). Upon receiving an access request (2), the PEP forwards the request to the PDP (3), which evaluates the request in relation to the policies obtained from the PAP. If additional information is needed for the evaluation, the PDP questions the PIP (4;5). The PDP evaluates the request against the policies and returns a response specifying the decision to access the PEP (6), which enforces the decision.

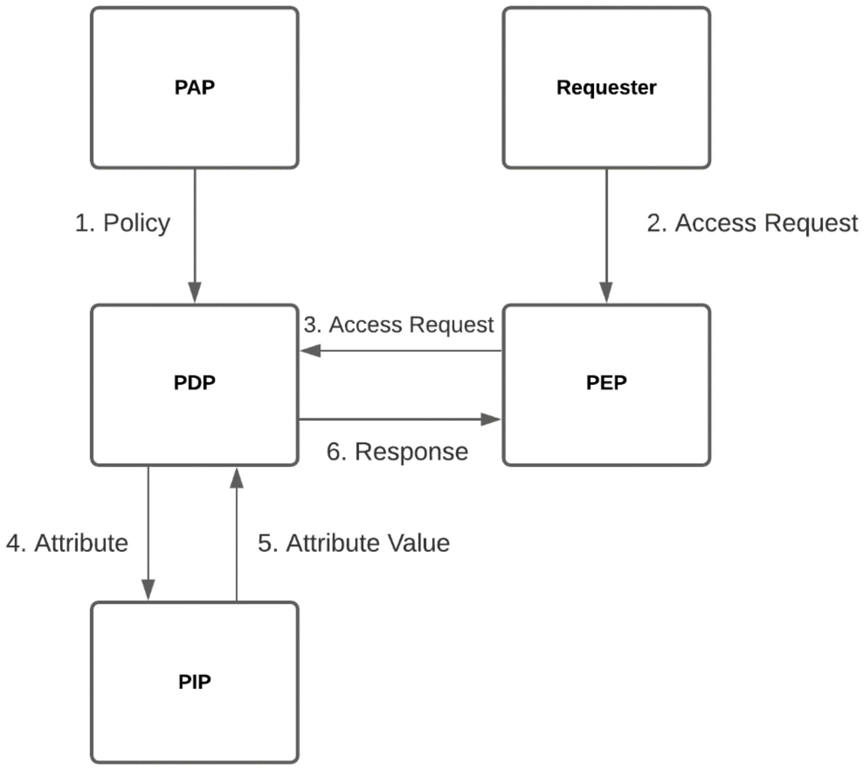


Figure 8. Policy-based Architecture

OAuth2 [9] is the most recent and used application of a token-based architecture. Token-based authentication works by ensuring that each request to a server is accompanied by a signed token that the server checks for authenticity and returns a response to the request. The standard model for implementing this architecture is OAuth2, which defines four roles, as shown in Figure 9:

- **Resource owner:** An entity capable of granting access to a protected resource. When the resource owner is a person, it is referred as an end-user.
- **Resource server:** The server hosting the protected resources, capable of accepting and respond to resource requests through access tokens.
- **Client:** An application that makes requests on behalf of the resource owner and with its authorization. The term "client" does not force any particular implementation characteristics (e.g., whether the application executes on a server, a desktop, or other devices).
- **Authorization server:** The server issuing access tokens to the client after successfully authenticating the resource owner and obtaining authorization.

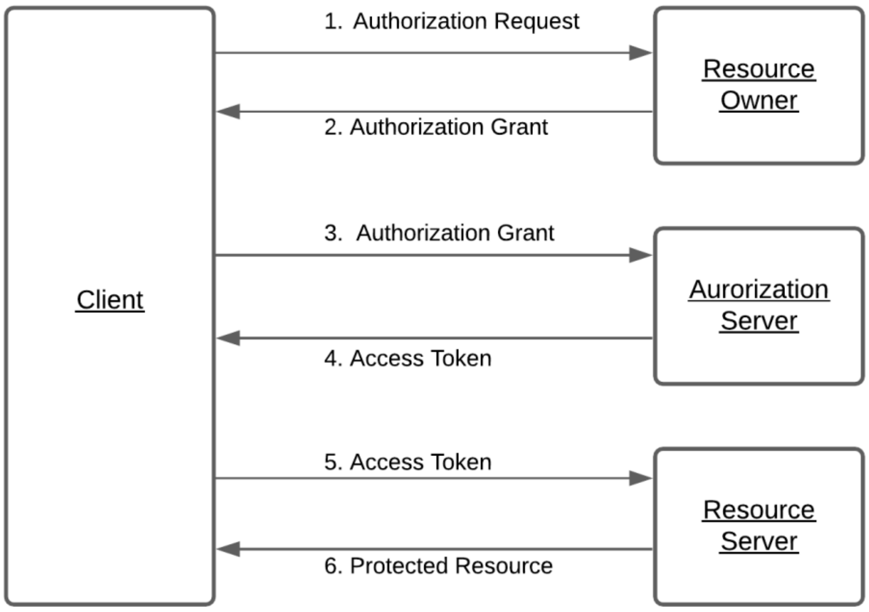


Figure 9. Authorization Protocol Flow

Solutions adopting a policy-based architecture typically provide a single, centralized point for the evaluation and enforcement of access control policies. This solution may not be suitable when resources are distributed across different nodes, which is a typical situation in many IoT applications. In the case of FISHY, which is a distributed system, it seems better to implement a hybrid model, where we will use the standard XACML as a base, plus an authentication process based on JSON Web Token. JSON is characterized for being a lightweight and relatively easy to work format that should be integrated with XACML to provide a standardized interface between the PEP and the PDP structuring the request and response task.

3.1.4 Data Management

3.1.4.1 Adaptation

Adaptation elements will act as transformer functions, mapping the raw data received through the infrastructure abstractions and the edge elements onto data structures able to include all necessary attributes to support both functional and non-functional system requirements. The non-functional requirements are typically related to performance and security functions, including the privacy and AC policy rules. The functional requirements are related to the main FISHY functions comprising security events and related operational data. Higher level modules, especially those already available will naturally impose data format requirements and, possibly, data cleaning and aggregation demands.

3.1.4.2 Anonymization

The concept of anonymization is frequently associated with data and refers to the personal data conversion process into “anonymized data” by the application of a range of techniques with the main purposes of preserving privacy of users and complying to regulatory requirements. The anonymization process can be reversible or irreversible and, in this project, it focuses on shared data among organizations or entities, where additional administrative and technical controls may need to be imposed in order to reduce the risk of unauthorized disclosure of personal data. This process has no ideal solution to all cases, so it must be adapted to select the most adequate approach for the circumstances.

There are some elements that should be taken into consideration when determining a suitable anonymization technique and an appropriate anonymization level. The purpose of anonymization should be clear, especially under the consideration that anonymization tends do decrease information, so when the anonymization impact increases the utility of the dataset decreases. Another important element is inferred information, the possibility for certain information to be inferred from anonymized data. The expertise with the subject matter is also an element to be considered since the anonymization techniques usually reduce the risk of identifiability to a level acceptable by the organizational risk portfolio but that can imply a degradation of the value of data for subject experts.

As said before, there is no ideal technique for all cases, so the anonymisation process should apply different techniques regarding to every specific situation with the purpose of reducing the risk of disclosure of personal data. The most common data anonymization techniques are:

- Attribute or Record Suppression – This technique refers to the removal of a section of data (e.g., a column in a table) or the removal of an entire record in a dataset. This technique is applied when an attribute is not needed in the final anonymized dataset and it is an easy way to decrease identifiability at the beginning of the anonymization process.
- Character Masking – This technique refers to the change of the characters of a data value using a constant symbol. Typically, the masking technique is only partially being applied, only to some characters in the attribute, and it is used when hiding part of a record is sufficient to provide the extent of anonymity required.
- Pseudonymization – This technique refers to the coding or replacement of identifiable data with made up values and it is applied when the data values need to be uniquely distinguished and where no character of the original attribute should be kept.
- Generalisation – This technique refers to a deliberate reduction in the precision of data or recoding, it is applied to values that can be generalised and still be useful for the intended purpose.
- Swapping – This technique aims to rearrange data in the dataset such that individual attribute values are still represented in the dataset but do not correspond to the original records, this is applied only when there is no need for analysing relationships among attributes.
- Data Perturbation – This technique is characterized for slightly modifying the values of the original dataset and it is applied when quasi-identifiers may potentially be identified when combined with other data sources.
- Synthetic Data – This technique refers to the generation of synthetic datasets directly and separately from the original data, instead of modifying the dataset.
- Data Aggregation – This technique refers to the conversion of dataset from a list of records to summarised values. It is usually applied when records are not needed, and the aggregated data is sufficient for the purpose.

3.1.4.3 Data Infrastructure

The Data Infrastructure proposed for FISHY is intended to support the collection and sharing of the diverse data produced by the different components, from the individual network elements and devices to events and knowledge as produced by elements higher in the policy enforcement hierarchy. In particular, it is committed to provide a common, model-driven approach able to addresses the following goals:

- Capacity to collect and forward the diverse information to be exchanged within the FISHY architecture.
- Support for the integration of heterogeneous formats for log data, events and alarms, knowledge sharing, policy expressions, etc.

- Incorporate the capacity to process (by extracting, transforming, and selecting) relevant information (i.e., alerts from events, errors from info, flows from samples…)
- Mechanisms for adaption and enrichment of data sets: label data, format unification, anonymization.
- Open interfaces suitable to be used by the different FISHY components consuming data flows, irrespectively of their nature: TIM, IRO, SCM...

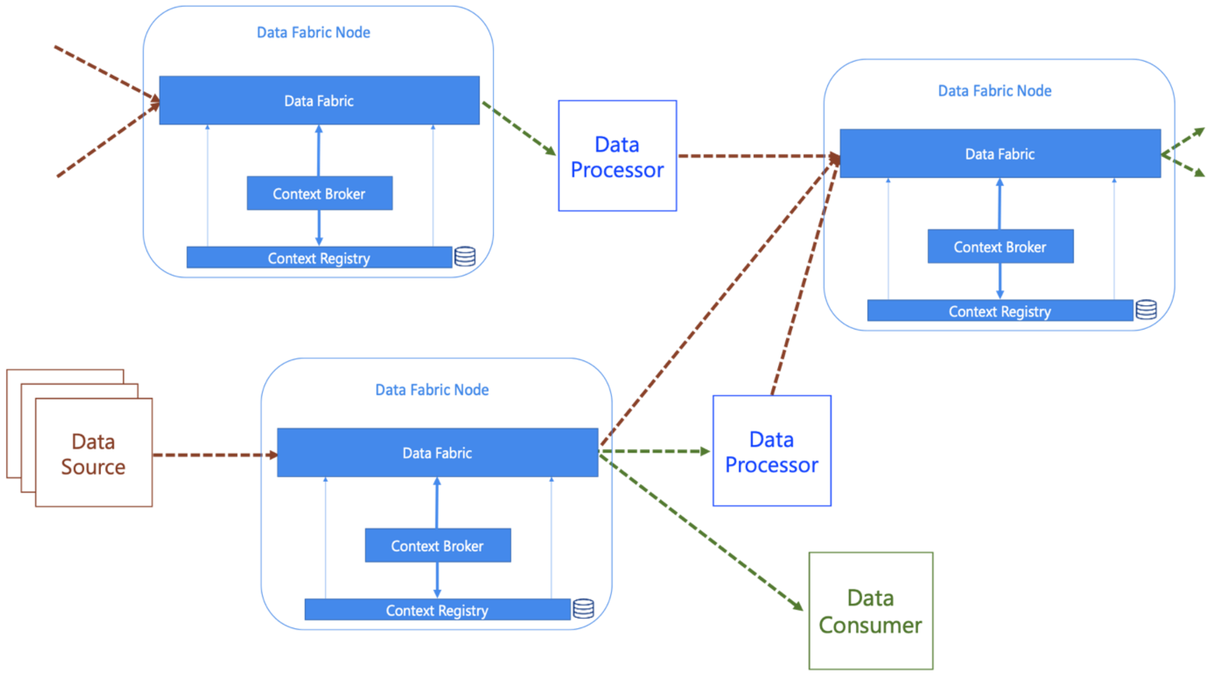


Figure 10. Data Infrastructure architecture

The FISHY Data Infrastructure, as shown in the figure above, is built by the composition of data forwarding (*Data Fabric*) nodes that connect different *Data Sources* (providing raw data to the Fabric nodes), *Data Consumers* (using the data forwarded by the Fabric nodes) and *Data Processors* (acting as Consumers for a Fabric node, and Sources for a following Fabric node). Sources, Consumers (and Processors, in their double-sided role) are described according to a metadata model, describing the characteristics of the data flows they produce and consume in terms of access method, data models, permissions and verification mechanisms. Fabric nodes use the metadata of the attached Sources and Consumers to collect data, combine, transform, and forward them. Processors are used to apply transformations on the data flows that required specific capacities or additional information.

The Data Infrastructure is the backbone of the SPI, in charge of data transformation and forwarding Northbound within the FISHY architecture. The Data Infrastructure mechanisms will apply metadata-based enforcement of access control, assessment on data provenance, and the mechanisms to guarantee privacy preservation, as described in the sections above. The metadata-based approach supports an open management of Sources and Consumers, and the dynamic orchestration of data flows.

The following workflow illustrates the process for registration and process of Sources and Consumers at a Fabric node, as well as their further interactions. Note the adaptation process implies there is not a necessary one-to-one mapping between the dataflow elements sent by the Source and those forwarded to the Consumer.

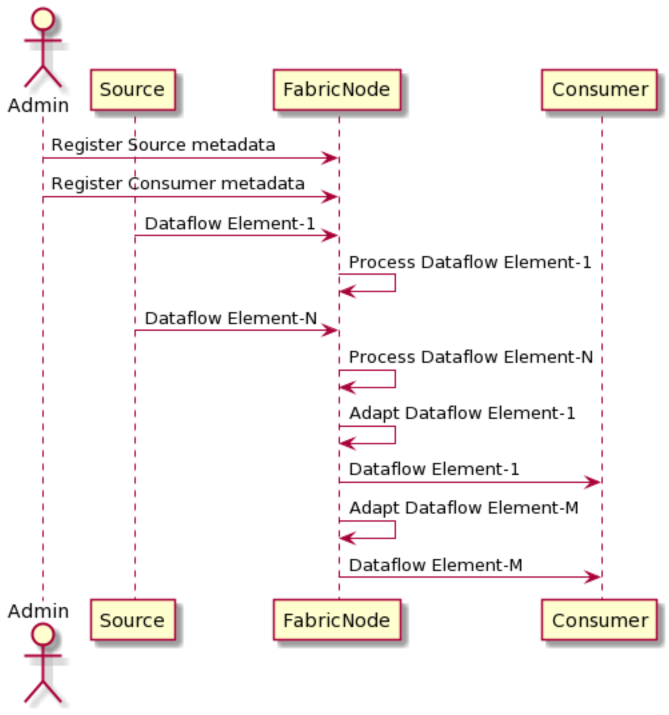


Figure 11. Data Infrastructure Workflow

3.2 Trust & Incident Management

The TIM module is responsible for assessing the security posture of the monitored infrastructure, detection of anomalous events, assessing the impact of present vulnerabilities and incidents and providing actions and recommendations for mitigation of cybersecurity risks. Facilitating the analysis and assessment of the cybersecurity status of the monitored infrastructure is the FISHY Appliance, enabling the deployment and configuration of the various metrics-gathering tools that forward data to the analysis components of TIM. The deployment and configuration of tools will be performed by a managing component referred to as FISHY Agent remote configuration scripts [10]. The suite of tools to be deployed and their configurations on each node of a supply chain will be derived from the intents and policies TIM receives from IRO. The proposed flow of this operation is presented in Figure 12. The Appliance is situated in the lower domains of FISHY architecture, close to the monitored infrastructure, along EDC, SPI and SIA.

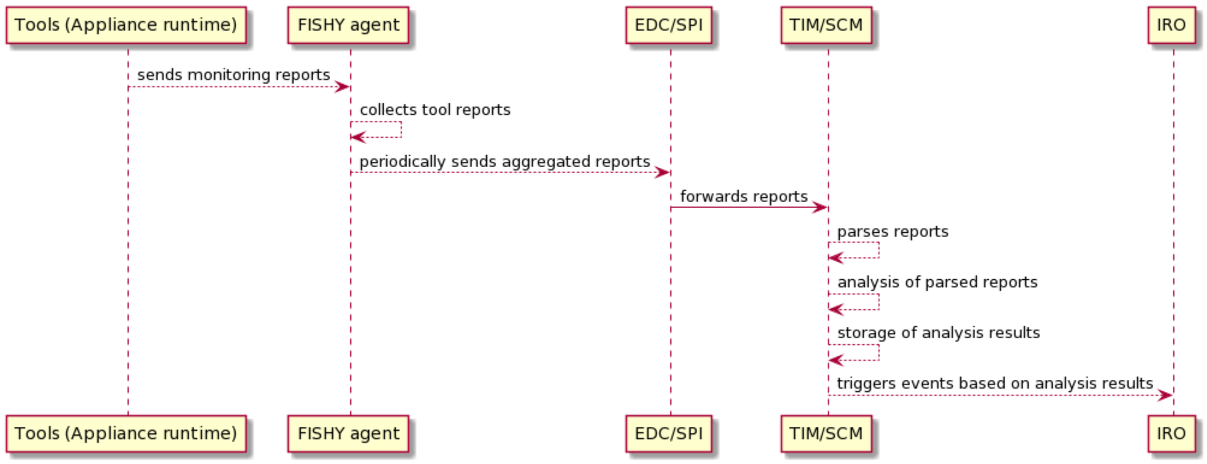

Tools (Appliance runtime)
FISHY agent
EDC/SPI
TIM/SCM
IRO
sends monitoring reports
collects tool reports
periodically sends aggregated reports
forwards reports
parses reports
analysis of parsed reports
storage of analysis results
triggers events based on analysis results
Tools (Appliance runtime)
FISHY agent
EDC/SPI
TIM/SCM
IRO

Figure 12. Sending metrics to TIM

The outputs of tools will be collected by the FISHY Agent and forwarded to the SPI component for data normalisation, anonymisation and secure transmission back to TIM, where the gathered metrics are first saved in the Threat/Attack Repository and then processed and analysed. The results of the analysis of the gathered data are again saved in the Threat/Attack Repository, where they are made available to components able to take mitigation actions as input and provide new intents and policies that can be applied to the monitored infrastructure by the EDC.

3.2.1 Vulnerability Assessment

The aim of the Vulnerability Assessment component is to provide a proactive, rather than reactive, assessment of the security status of a given network and/or infrastructure. The inputs of the various metrics gathering tools are collected, analysed, and stored for further review and based on the results of the analysis, recommendations are issued on how to harden the system or network and increase its security level.

The Risk Assessment Engine (RAE) is a Python-implemented tool of ATOS that can perform vulnerability assessment thank to its incorporated vulnerability scan. RAE can gather data, that is, indicators, from the monitoring of the targeted infrastructure. After performing the vulnerability scan, a report is generated by RAE. This is done with the purpose of updating the indicators associated to the vulnerabilities that are detected in the monitored platform. The workflow is expected to be like the one depicted in Figure 13:

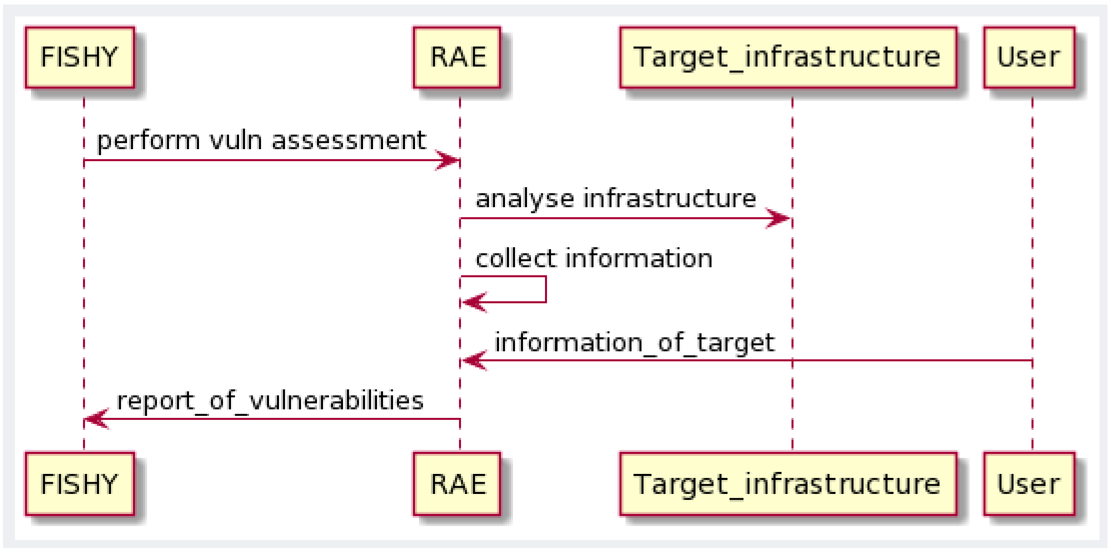
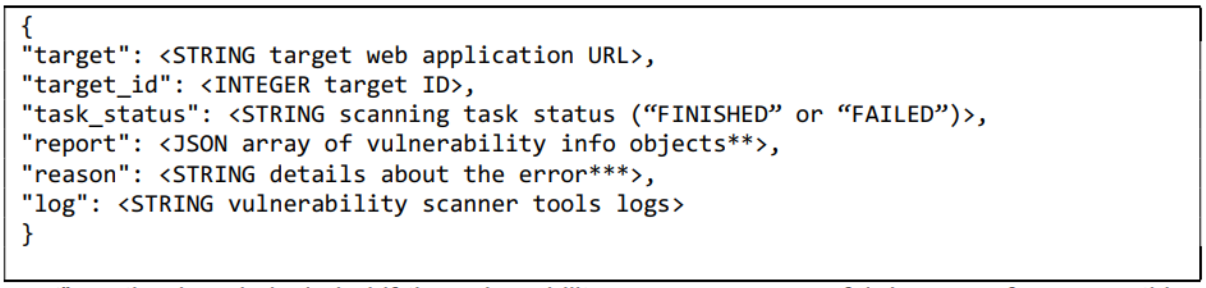


Figure 13. Vulnerability assessment, exemplified with RAE

In addition, the format of the vulnerability reports is indicated in Figure 14:



```
{
"target": <STRING target web application URL>,
"target_id": <INTEGER target ID>,
"task_status": <STRING scanning task status (“FINISHED” or “FAILED”)>,
"report": <JSON array of vulnerability info objects**>,
"reason": <STRING details about the error***>,
"log": <STRING vulnerability scanner tools logs>
}
```

Figure 14. Format of the vulnerability report by RAE

Regarding the communication mechanisms the REST API employed in the architecture of RAE guarantees security, given that HTTPS is chosen to secure all requests. OAuth2 is used with the purpose of authenticating requests so everyone will contain an **access token**. According to good security practices, access tokens also need to be renewed after a certain amount of time, so it is necessary to use a refresh token to be granted a new, valid access token.

Finally, the following use cases define different approaches for the vulnerability assessment performed:

Assessment of web page vulnerabilities	
Description	The Vulnerability Assessment Tool performs checks for exploitable vulnerabilities of a web page using W3AF – Web Application Attack and Audit Framework [11] and OWASP Zed Attack Proxy [12].
Actors	Vulnerability Assessment, Impact Assessment, Threat/Attack repository.
Inputs	Inputs are received from the Vulnerability Assessment Tool.
Outputs	The processed data is passed to RAE for further analysis and/or the Threat/Attack repository for storage.

Infrastructure scans	
Description	The Vulnerability Assessment Tool performs NMAP scans of a target machine or network and produces a report about server availability and open ports.
Actors	Vulnerability Assessment, Impact Assessment, Threat/Attack repository.
Inputs	Inputs are received from the Vulnerability Assessment Tool.
Outputs	The processed data is passed to RAE for further analysis and/or the Threat/Attack repository for storage.

Infrastructure monitoring	
Description	Wazuh [13] collects monitoring reports and generates alerts based on input gathered from agents running on the target infrastructure.
Actors	Vulnerability Assessment, Impact Assessment, Threat/Attack repository.
Inputs	Inputs are received from Wazuh.
Outputs	The processed data is passed to RAE for further analysis and/or the Threat/Attack repository for storage.

Target infrastructure scan	
Description	RAE executes a vulnerability scan on the monitored platform: it is aimed at gathering information about the whole infrastructure.
Actors	Vulnerability Assessment, Threat/Attack repository.
Inputs	Inputs are provided by sensors and scanners deployed on the monitored platform.
Outputs	Outputs are correlated with information of the platform provided by the user (with the help of a questionnaire). This is part of the information needed to estimate how vulnerabilities could impact data on the monitored platform.

Risk assessment	
Description	RAE performs a Risk Assessment of the monitored infrastructure.
Actors	Vulnerability Assessment, impact assessment, mitigation.
Inputs	RAE receives information of vulnerabilities found. Besides, the user can provide some data about infrastructure components.
Outputs	Once the Risk Assessment is performed and based on the results obtained, RAE proposes some mitigation measures.

3.2.2 Incident Detection

Incident detection is a challenging task. It demands on information, ability to recognize patterns and new threats as well as skills to deal with different scenarios and situations. In a market environment, companies struggle to identify and recognize incidents and often this leads to a delay on the response to these incidents. Part of the key to be successful comes from gathering, filtering, and obtaining proper information from raw security data. However, the key to be successful regarding incident detection may lay on adopting an adequate security posture, something that can be associated with good security practices such as continuous monitoring. In this sense, the ability of being pro-active in terms of detecting and mitigating threats and adapting to continuous environment changes are key factors.

In that scenario, monitoring tools such as the Security Incident Event Management (SIEM) tools can help to achieve a holistic approach in cyber incidents detection. The SIEM just normally needs to integrate with the sensors, that is, to receive information sent by them. There are several kinds of sensors designed to monitor all sorts of components and detect security incidents.

Finally, we consider that the following use cases provide an insight into incident detection related to FISHY platform:

Infrastructure supervision and event detection	
Description	XL-SIEM deploys agents on the target infrastructure: they are responsible for the gathering of information and processing of security events.
Actors	Incident detection, Threat/Attack repository.
Inputs	Information provided by agents and sensors gathering data provide valuable incident input.
Outputs	Once an incident is recognised, data is sent to the threat and attack repository where it can be stored for future use.

Incident mitigation	
Description	XL-SIEM detects a security incident based on information provided by agents and sensors.
Actors	Incident detection, mitigation.
Inputs	XL-SIEM provides correlated and refined data regarding the input detected. Information of previous attacks and threats which is stored in the threat/attack repository can be used as well.

Outputs	The mitigation component can deal effectively with the incident given that information has been previously prepared and processed. Active reconfiguration of the infrastructure is managed by means of accurate mitigation strategies provided by FISHY.

3.2.2.1 Trust Monitor

The *Trust Monitor (TMon)* is in charge of verifying the integrity of the software running on both infrastructural nodes and containers. To assess the software's posture, the TMon leverages periodic *Remote Attestation (RA)* and a hardware-based *Root-of-Trust (RoT)*, which in this case is the *Trusted Platform Module (TPM)*. During the attestation process, a *Verifier* inside the TM queries an *Attester* (Trust Agent) running on the infrastructural node. The latter is in charge of providing the *Verifier* with measures related to the software we aim to attest. Those measures are certified by the RoT of the target node. Figure 15 represents a high-level sequence diagram of the attestation process integrated within the FISHY architecture.

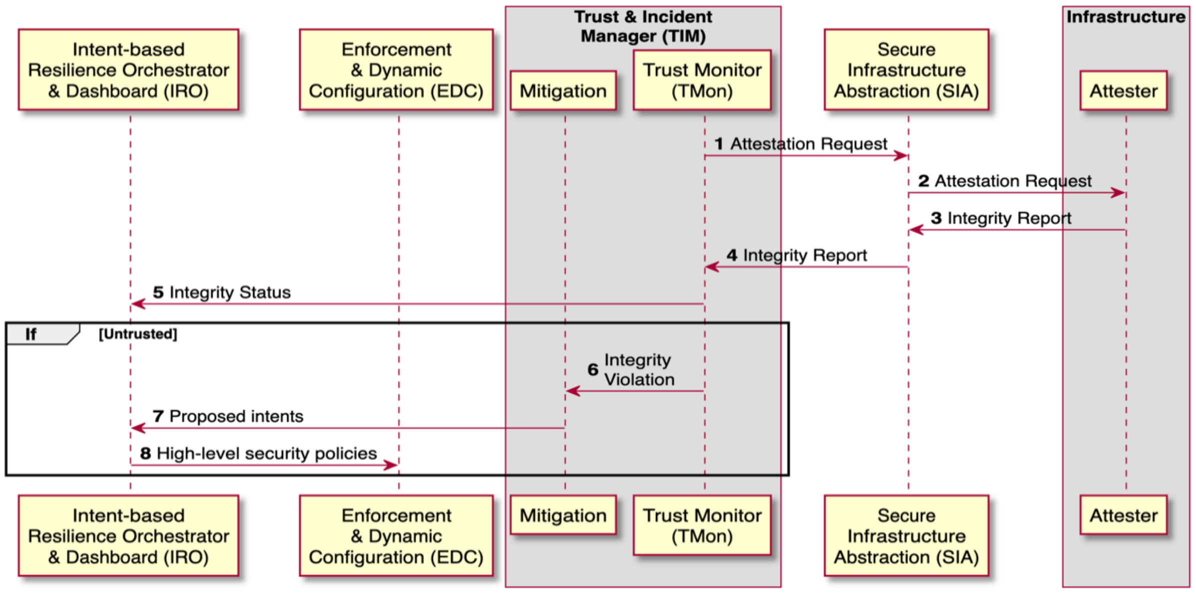


Figure 15. Remote attestation workflow

The TMon periodically sends an Attestation Request to the Attester within the infrastructural node, receiving as response an Integrity Report (IR). This report contains the measures of the binaries running on the target node certified by the RoT. TMon verifies those measures against a whitelist of golden values and reports the Integrity Status of the node to the IRO. When an integrity violation occurs, TMon also sends a report regarding the violation to the Mitigation module within the TIM. This latter must produce specific intents as remediation for the event. Those intents are forwarded to the IRO, which is in charge of transmitting them to the Enforcement & Dynamic Configuration (EDC) module upon a system administrator approval in the form of high-level security policies. This step triggers the refinement and the enforcement processes to apply the mitigation.

Eventually, the attestation process could also be triggered on-demand by the IRO. This allows for the asynchronous check of the integrity status of the platform in critical moments of its lifecycle, i.e. when a new service must be deployed.

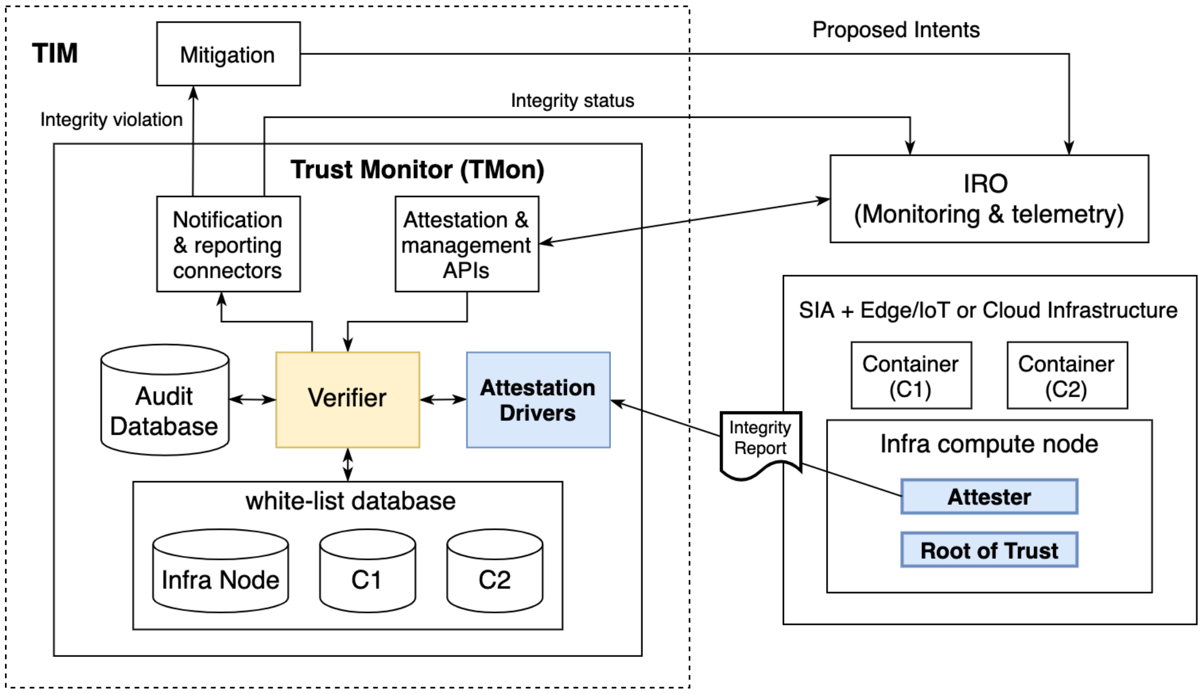


Figure 16. High-level Trust Monitor architecture

Figure 16 depicts a high-level architectural view of the Trust Monitor adapted to the scope of the project. In particular, the TM is composed of six different submodules:

- *Attestation Drivers*: a collection of attestation drivers is available to provide integrity verification of heterogeneous hosts (e.g., compute nodes) and virtual instances (e.g., containers). The main idea is to leverage several frameworks for the remote attestation process depending on the host to be attested and the virtualisation technologies adopted. Specific attestation drivers could be developed to interact with attestation frameworks following a specific logic imposed by the Verifier in order to validate the Integrity Reports.
- *Verifier*: this is the main module that receives the integrity reports, validates them, and compares their values with golden measures stored in the white-list database.
- *Audit Database*: this database contains data regarding the results of the attestation processes. External components could use these data to collect historical data about the trustworthiness of the infrastructure.
- *white-list database*: compute nodes and virtual instances within the infrastructure shall own a related white-list of golden measures. The white-list database is in charge of storing those white-lists.
- *Notification & reporting connectors*: this is an abstraction of the interfaces that should be in place to report special events related to the integrity verification process. In particular when an event occurs (i.e., a node becomes untrusted), the Verifier could notify external components of this event using this submodule.
- *Attestation & management APIs*: this represents a set of APIs built for requiring the attestation of specific nodes or instances. In addition, an API for configuration and management is in place.

TMon is agnostic to the underlying technologies used for the RA. In particular, TMon could be extended with specific Attestation Drivers that abstract the low-level operations depending on the type of host, the virtualisation technology, and the choice of operating system. So far, TMon has been extended to support Docker container attestation. This allows TMon to establish on a bare-metal infrastructural node which one of the containers has been compromised. Leveraging this information, it is possible to address mitigation towards a single container without requiring the isolation of the whole infrastructural node.

A general use case scenario is represented by the integrity violation of an infrastructural node by modifying or adding software for malicious purposes. In our system, the TMon periodically queries the Attesters asking for IRs. When a binary related to a specific software is modified, the next round of the attestation process must detect it. In particular, the TMon starts the RA process and collect the IRs from the Attesters. At this point, the Verifier within the TMon can compare the measures of the binaries with the whitelists that it stores within its database. When it validates the measure of the compromised binary, it detects the violation and informs the mitigation module. This latter is in charge of providing specific intents to the IRO to mitigate the attack. Ultimately, the IRO, upon approval from a system administrator, could then send the high-level security policies to the EDC, starting the refinement process and the enforcement of the mitigation policies.

Possible mitigations in this scenario could be the isolation of the compromised node or container. Two real use case scenarios could be the following ones:

- Botnet use case: attackers find a vulnerability in nodes, and they use it to install malware to take remote control of the nodes themselves. The controlled nodes are now part of a botnet. At the moment, they are still quiet, but they could be used at any time to launch a large-scale attack, for example, a denial of service (DoS), against unknown targets. At the next programmed nodes attestation, the anomaly is detected and reported through the Fishy dashboard (IRO) to the system administrator as a series of intents that he could apply as mitigation. Possible remediation could be shutting down the vulnerable nodes and restart them using an image patched to fix the vulnerability.
- Cryptominer use case: attackers find nodes vulnerable to malware installation and take advantage of the situation using them for cryptomining. After the next attestation round, the system administrator receives a report about the failure of the RA process. The report he receives back notifies him about compromised nodes and possible mitigation in the form of intents. He decides to apply them shutting down the compromised nodes to analyze the attack.

To summarize the role of TMon, it periodically attests the infrastructure and could also be driven by the IRO module for on-demand attestation. It starts the attestation process, taking in input the IRs coming from the Attesters and provides in output a report in JSON format about the status of the infrastructure. In case of Integrity Violation, it sends a notification to the Mitigation module which in turn provides intents to mitigate the breach.

3.2.3 Impact Assessment

The general idea of the impact assessment is the quantification on how risks and, more specifically, changes can influence a system.

Given that the FISHY project is aimed at helping heterogeneous supply chain infrastructures, there are plenty of variables which can sway it. While the objective is to make the ICT infrastructure resilient to external factors, various tools and approaches could help achieving this goal. One strategy could be to have support of automated-assessment tools that can provide cybersecurity and threat information such as risks, vulnerabilities, changes, and any other external driver affecting the monitored infrastructure.

The impact assessment is a key process to be performed on the FISHY project. This activity is accomplished in the Trusted Incident Manager of the Trusted Manager (or TM). It should draw from:

- A previous vulnerability assessment of the. As it has been described on the vulnerability assessment section, Risk Assessment Engine (RAE) can effectively perform this vulnerability analysis of the target system.

- A provided list of incidents.
- It could be also a good idea to check the threat/attack repository (depending on the scenario).

Based on our knowing of its capacities, on a simple approach, we suggest the use of the RAE. However, there are multiple tools that can be used for the purpose of dealing with impact assessment. On a scenario where RAE is the chosen tool to be used, we propose the workflow depicted in Figure 17:

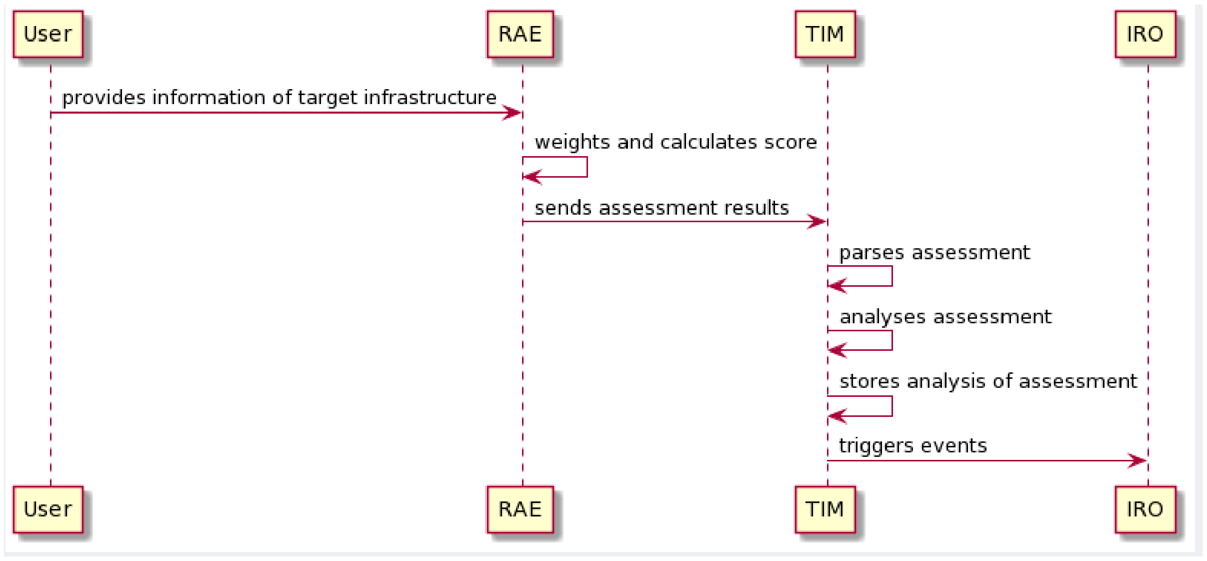

User
RAE
TIM
IRO
provides information of target infrastructure
weights and calculates score
sends assessment results
parses assessment
analyses assessment
stores analysis of assessment
triggers events
User
RAE
TIM
IRO

Figure 17. Impact assessment, exemplified with RAE

3.2.4 Mitigation

3.2.4.1 ML-Based Solutions

These solutions focus on the detection of anomalous behaviour by the network and IoT systems based on the analysis of monitoring data through Machine Learning algorithms. Mitigation mechanisms based on ML algorithms use to work in two different ways: online mode and offline mode.

Offline mode: A Database/Repository of the net entries is read. Based on these data, an expected behaviour pattern is established for the devices to be protected.

Three main components are defined:

- The first component aims to perform a binary classification to detect anomalies (system entries are benign or anomalies).
- The second component identifies the type of anomaly that has occurred, to be able to issue a warning message, so that corrective actions can be carried out.
- If the type of anomaly was not classified, there is a third component which, through unsupervised machine learning techniques, establishes a possible categorization of the anomaly. Figure 18 shows these components.

The models used for detecting anomalies and categorizing are adjusted offline. Once the models have been trained, the online mode can be used.

Online mode: A user interface monitors system data to detect attacks. This interface uses the above mentioned selected fitted model that has been trained offline.

Figure 18 also shows the interdependence among the components and the two modes.

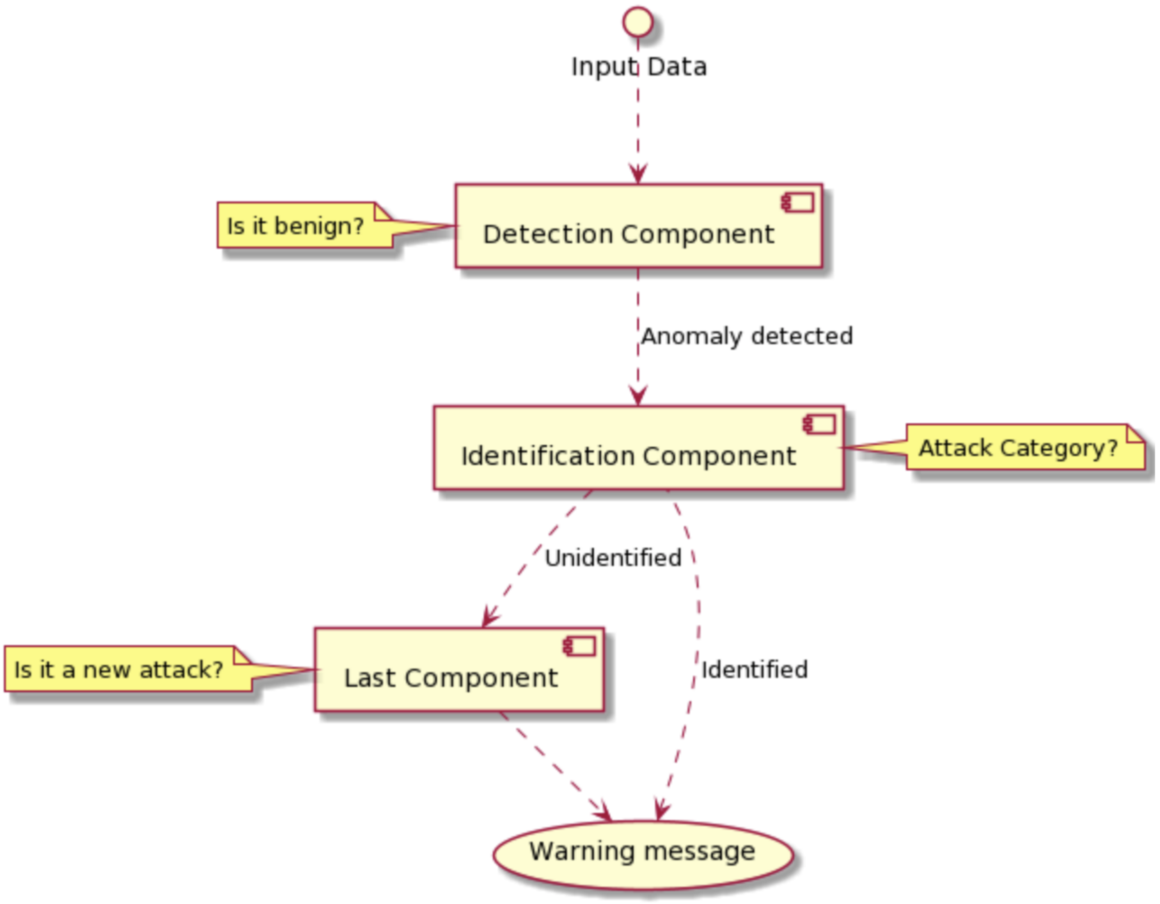


Figure 18. ML-based mitigation workflow

3.2.5 Threat/Attack Repository

FISHY will also provide a threat/attack repository facilitating data interoperability among the different stakeholders in the supply chain aimed at improving both estimation and mitigation actions. The Threat/Attack Repository will store the outcome of the Trust & Incident Management layer whenever the analysis leads to a threat or attack. Based on the immutability principle, the Repository will store the result, so that other stakeholders can be timely informed, and that information can be used for evidence.

A practical example of the Threat/Attack repository interacting with other components is presented in Figure 19.

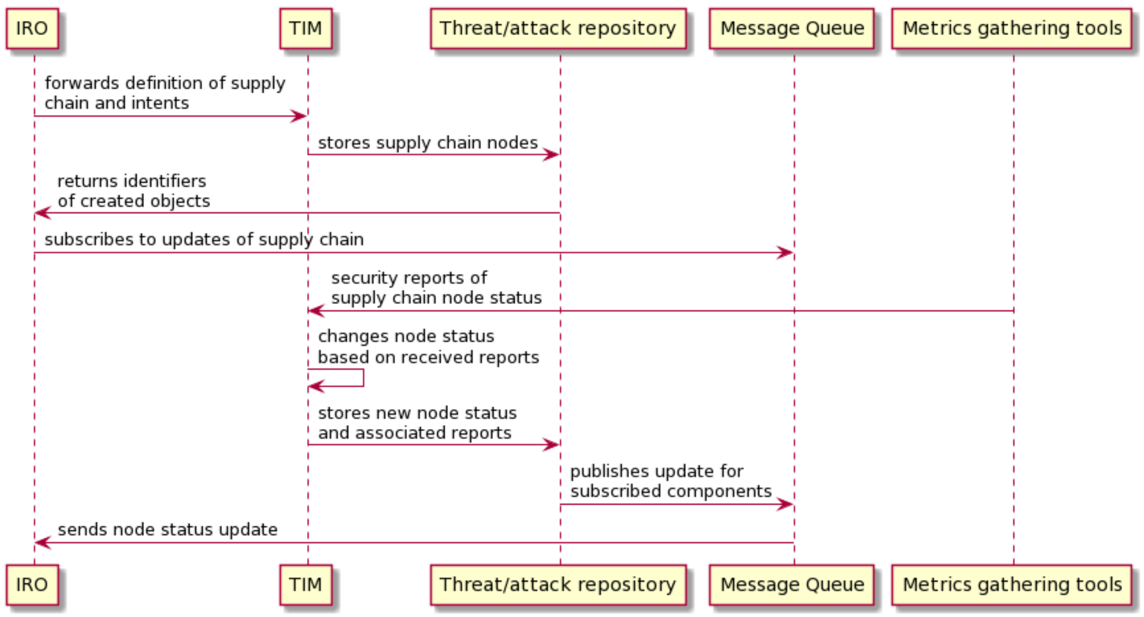

IRO
TIM
Threat/attack repository
Message Queue
Metrics gathering tools
forwards definition of supply chain and intents
stores supply chain nodes
returns identifiers of created objects
subscribes to updates of supply chain
security reports of supply chain node status
changes node status based on received reports
stores new node status and associated reports
publishes update for subscribed components
sends node status update

Figure 19. High-level workflow involving the Threat/Attack repository

The Threat/Attack repository should be implemented as a database or document store with a REST API wrapper layer that exposes the ability to perform CRUD operations without direct access to the underlying storage service. Additionally, the REST wrapper should also implement a pub/sub mechanism, allowing components that interact with it to receive near real-time updates when changes to entities occur. The pub/sub mechanism can be implemented by using existing solutions, most likely a combination of a message queue service like RabbitMQ [14] or Redis [15] and websockets that push updates and notifications to the user interface.

3.2.6 Smart Contracts

The main goal of the Smart Contract (SC) module is to offer an immutable mechanism within the FISHY platform for storing and retrieving information, in a secure manner, related to threats and attacks, including logs and events with specific information that can be used across the supply chain. This component, based on Distributed Ledger (DL) technologies, will be properly adapted to the specific needs of the supply chain use cases that FISHY deals with and beyond.

The SC component includes three main (groups of) components namely the Relay Server, the Event Server and a collection of generic and FISHY specific smart contracts acting as distributed applications (DAPP) on top of the supporting DL system.

The entry point of the SC will be the Relay Server, a web service serving RESTful HTTP requests from the SC clients and interacting for their interest with the DL infrastructure through the web service in charge of guaranteeing the temporal succession of events, the Event Server. This interaction may be either in the form of interacting with DAPPs or mediating simple requests against the basic features of the underlying DL. For each of the two modalities, a separate RESTful API category will be exposed.

In the case of simple ledger requests, functionalities would include querying about the current block number, the list of transactions included in a block, the list of logs associated to a transaction, etc.

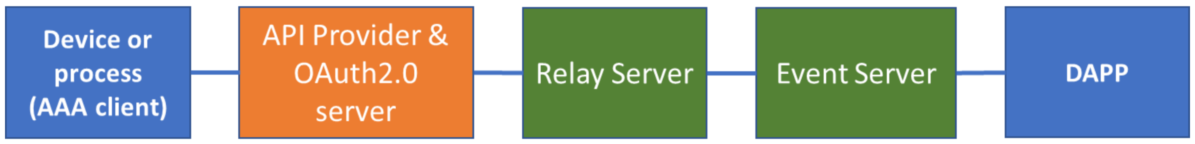

Device or process (AAA client)
API Provider & OAuth2.0 server
Relay Server
Event Server
DAPP

Figure 20. FISHY Smart Contract sub-modules

3.3 Metric Gathering

In the white box mode described in section 3.1 the capacity of generating metrics under the control of the FISHY framework was introduce. Usually, metrics are described as quantifiable measurements of any specific characteristic of a system, or a product and they can only be established if there is a well-defined objective. While metrics can be seen as quantifiable measures, the measurements provide specific information since they are generated nominally [16].

The **definition of security and privacy metrics** is essential to assist the decision-making process related to aspects of data security and privacy. Metrics are also important in the design phase of the security architecture till the development of controls to the effectiveness and efficiency of security operations. These can be used throughout the entire software development lifecycle (SDLC) to recognize and eliminate vulnerabilities that may exist. In addition, metrics can help to identify, analyse, and fix security flaws and, therefore, increasing the effectiveness of security and privacy controls.

One simple classification is to consider metrics that represent the maturity level of the most valuable processes that secure the system [17]. Since the Software Engineering Institute has launched the Capability Maturity Model (CMM) many maturity models have been suggested by specialists across multiple domains to diagnose and eliminate inadequate capabilities. Maturity models usually include a sequence of levels that together form an accurate and logical path from an initial state to maturity that in the end will indicate an organization's current or desirable capabilities. In practice, maturity models are expected to expose current maturity levels and to include respective improvement measures following application-specific purposes which may be for example:

- Descriptive purpose - "if it is applied for as-is assessments where the current capabilities of the entity under investigation are assessed with respect to given criteria".
- Prescriptive purpose - "if it indicates how to identify desirable maturity levels and provides guidelines on improvement measures".
- Comparative purpose - "if it allows for internal or external benchmarking. Given sufficient historical data from a large number of assessment participants, the maturity levels of similar business units and organizations can be compared" [18].

Develop efforts related to security metrics and measurements with a high level of abstraction and formalism are often difficult to analyse and apply in organizations [17].

In order to improve the understanding and acceptance of all defined metrics, organizations should fund them into process improvement frameworks that should take into account the objective of the metrics application. They should also identify metrics as develop strategies to generate metrics and define how these will be reported and ultimately create an action plan and establish a review cycle [16].

Currently, information is one of the most valuable assets for companies so, its preservation has become critical. For that purpose, it is important to use frameworks capable of guaranteeing the security and privacy of data since this is obtained through processes and procedures created to strengthen the objectives of organizations.

A couple of examples of these frameworks are NIST Special Publication 800-53 [19] and [20]. NIST SP800-53 reports security and privacy controls to facilitate the system implementation and protect organizational operations from a diverse set of threats, such as cyberattacks, structural failures, and human error. Thus, security and privacy controls are positioned "to support the integration of information security and privacy into organizational processes including enterprise architecture, systems engineering, system development life cycle, and acquisition/procurement". If this integration is successful, "it will demonstrate greater maturity of security and privacy programs and provide tighter coupling of security and privacy investments for core organizational missions and business functions". The Center of Internet Security (CIS) is an entity that promotes Internet security. To help preventing dangerous attacks and supporting the structures implemented, it proposes a set of cybersecurity practices and defensive actions. CIS, for each measure, encourages that metrics are always defined. It is important to be aware that many of the metrics defined require a high maturity level.

4 Implementation

4.1 Security & Privacy Data Space Infrastructure

4.1.1 Identity Manager and Access Policy

As referred in section 3, the IdM and AC functions will be supported by a Keycloak server [21], an implementation of the OpenID Connect standard. Nowadays, it is usual to deploy a Federated solution to allow a more flexible architecture and a centralization strategy. Keycloak supports any type of architecture. In the FISHY prototyping and development phases it may be better to deploy a separate server in each domain (user case provider). They all should be configured in the same way, making it easy to move to an alternative architecture organization (like high-availability cluster) at any time.

Keycloak demands for very low requirements:

- Runs on any operating executing Java.
- Java 8 JDK.
- zip or gzip and tar utilities.
- At least 512 MB of RAM.
- At least 1G of disk space.
- Shared external database, such as PostgreSQL, MySQL, Oracle, etc. To run in a cluster, a DB is required.

Within FISHY, Keycloak will run in a Docker container. To proceed with the installation via Docker, the following command is used:

```
$docker run -p 8080:8080 -e KEYCLOAK_USER=admin -e KEYCLOAK_PASSWORD=admin quay.io/keycloak/keycloak:13.0.0
```

When Keycloak is started for the first time, Keycloak itself creates a predefined domain, called Master, as it is the domain with full permissions on everything. Administrator accounts in this domain have permissions to view and manage any other domains created on the server instance. When defining an initial administrator account, it is created in the master domain. The initial login to the administration console will also be through the master domain. Thinking of good identity management practices, it is recommended not to use the master domain to manage users and applications. According to Keycloak, there are two types of realms, as illustrated in Figure 21:

- **Master realm -** Created at the beginning of the Keycloak. It contains the administrator account that was created at the first login. This realm should only be used to create other realms with administrative permissions.
- **Other realms C**reated by the administrator in the master realm. In these realms, administrators create users and applications. Applications are owned by users.

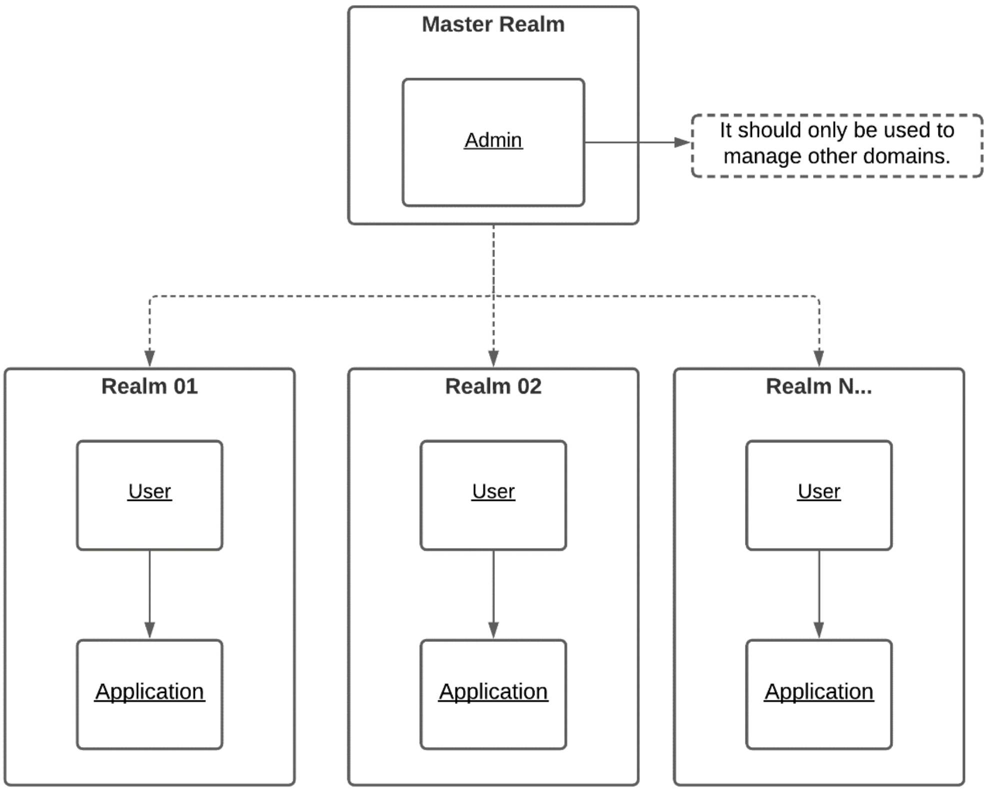


Figure 21. Realm management architecture

Keycloak provides some pre-defined functions to facilitate the management, such as Admin, User, Manager, and Employee. To manage these functions and accesses, Keycloak provides us with a series of roles and the possibility of creating more granularly the rules for these functions, because managing users individually is not a good practice. There is also the possibility to enable the SSL/HTTPS Mode for the domains, thus defining that to interact with that region, the requirements established by the SSL/HTTPS Mode must be met.

For user management, Keycloak offers several options that can be analysed according to the requirements of each scenario covered. The creation of users can be carried out mainly through Realm Admin, User Storage Federation, or by self-registration.

- **Realm Admin:** In this case, the domain administrator, will have to create each user through the administrative console or code.
- **User storage federation:** Keycloak can federate databases of existing external users. You can also use the protocols: LDAP and Active Directory. Keycloak also provides support for encoding extensions in any database, using the SPI of Keycloak itself.
- **Self-registration:** This function, when activated, creates a link on the registration page so that the user can make his registration. The fields that the user will see, can be changed by the admin.

Keycloak also allows us to implement some security standards, such as the mode of actions required, tasks that a user must complete before being allowed to log in. According to Keycloak, these are the necessary actions:

- **Update password:** When defined, a user must change their password.
- **Configure OTP:** When defined, a user must set up a one-time password generator on their mobile device using the free OTP application or Google Authenticator.

- **Check email:** When defined, a user must verify that they have a valid email account. An email will be sent to the user with a link that they must click on. Once this workflow is completed, they will be allowed to log in.
- **Update profile:** This required action asks the user to update their profile information, that is, their name, address, e-mail, and/or phone number.

For customer management, it is important to understand how the AS sees a customer. Clients are entities that can request authentication from a user. The creation and configuration of clients are carried out in advance through privileged access, an administrator account. Keycloak creates customers in two ways:

- The first type of customer is an application that wants to participate in a single sign-on. These customers just want Keycloak to provide security for them.
- The other type of client is requesting an access token to be able to call other services on behalf of the authenticated user.

Keycloak provides two protocols to protect the applications, OpenID Connect and SAML [22]. In FiSHY, the protocol that will be used at first will be OpenID Connect, however, if cases arise, it is possible to create clients that use the SMAL protocol.

4.1.2 Data Infrastructure

The core component of the FISHY Data Infrastructure is the Fabric Node, and its implementation will be based on the evolution of the Semantic Data Aggregator (SDA) being implemented to support the processing of experimental telemetry data within the 5Growth project [23], augmented to support access control and provenance metadata, and to incorporate specific privacy preservation mechanisms. The architecture of this SDA is depicted in the following figure.

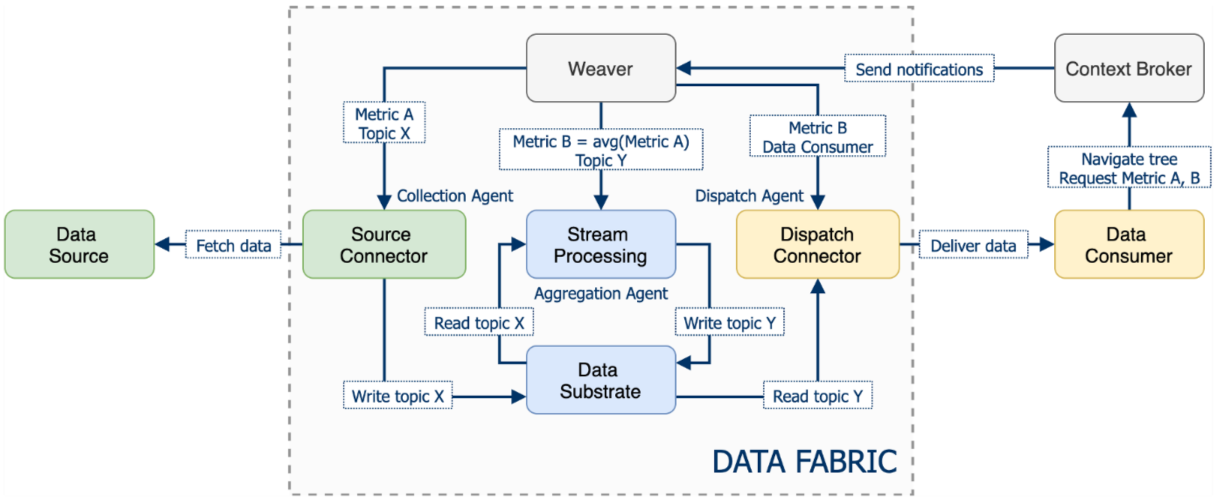


Figure 22. Initial Data Fabric Node architecture

Sources are Consumers are registered using their metadata, and these metadata are used to drive the collection of data from Sources, their adaptation according to the required results, and the delivery to the registered Consumers. The main elements in the current SDA architecture include:

- The **Context Broker**, in charge of receiving and processing the metadata descriptors for the attached Sources and Consumers. Metadata are based on the ETSI NGSI-LD standard [24] and kept in a **Context Registry** to be used by the other SDA components.
- The **Weaver**, in charge of orchestrating the data collection, adaptation and transformation process. It uses the Context Registry to identify Sources and Consumers, instantiating and configuring the required connectors and processing modules.

- **Source Connectors**, in charge of retrieving data from Sources. Current connectors are based on Apache NiFi [25] and implement different access methods and protocols, and support for consuming a number of data modelling languages.
- **Stream Processors**, in charge of performing processing and adaptation tasks that can be derived from the analysis of Source and Consumer metadata by the Weaver. Current processors are based on Apache Flink [26], with a number of modules specific for generally required tasks, plus support for the definition of specific ones.
- **Dispatch Connectors**, in charge of delivering data to Consumers. Current connectors are based on Apache NiFi as well, implementing different delivery methods and protocols, and support for a number of output data modelling languages.
- All these elements are supported by a **Data Substrate** based on Apache Kafka [27], conforming a generalized data bus on which elements are plugged in. The use of Kafka allows as well for a direct integration of those Sources and Consumers able to directly interface through a data bus of these characteristics.

4.2 Trust & Incident Management

4.2.1 Vulnerability Assessment

Regarding vulnerability assessment, ATOS is providing the Risk Assessment Engine (RAE). The tool can perform a vulnerability assessment of the target infrastructure by means of its own vulnerability scanner. RAE's Testing Module is responsible of running the scanner.

On a **high level**, RAE works in the following way:

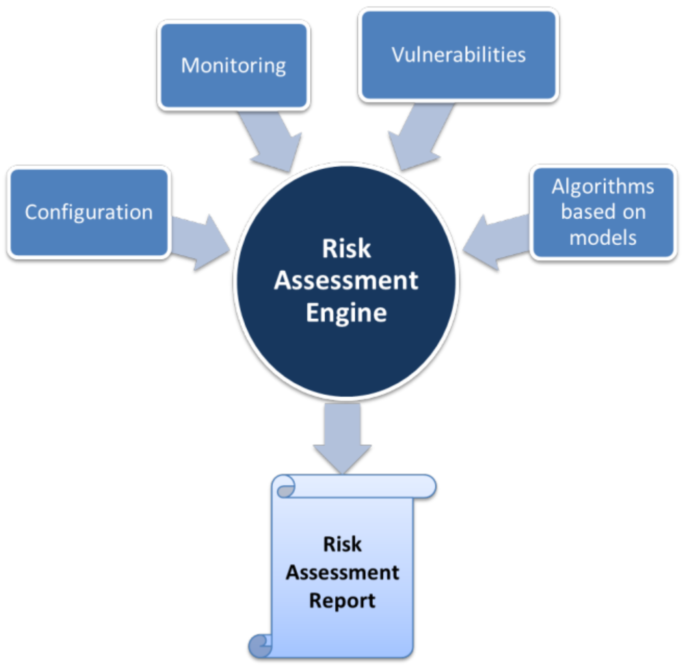


Figure 23. Inputs and outputs of RAE

Firstly, the user is asked to complete a questionnaire. This is done with the purpose of gathering infrastructure or company-related information which RAE needs to perform the assessment. Any information gathered from the user is stored in the Datawarehouse of the tool and employed by RAE to outline a profile. There are two algorithms in RAE:

- The first one (**Algorithm 1**) gathers the answers provided by the user and gives them a weight and a score.

- The second one (**Algorithm 2**) considers how the answers relate to the security environment, that is, how relevant is the answer in terms of confidentiality, integrity, and availability of the information.

Besides, data gathered from the user is combined with information related to vulnerabilities. RAE contains a Testing Module. This component **executes a vulnerability scanner** that can detect vulnerabilities in the target infrastructure. Once the scan is completed, the information is sent to Algorithm 2.

Therefore, Algorithm 2 weighs and links the vulnerabilities found with the impact they may suppose to the business with reference to the confidentiality, integrity, and availability of the information. When the processing and computation is performed, the algorithm sends the results to the Report production component or “Aggregator”, which is the one responsible for elaborating the report and sending it to the Dashboard where it will be shown to the user.

In addition, RAE also contains:

- A modelling component: starting from the inputs described the modelling component defines an instance of a model used to be used by the algorithm.
- A triggering detector, which detects if any input of the model has been modified.

RAE works in **real time**, but this does not mean that is constantly assessing the risk level. It is triggered according to the following scenarios:

1. On-demand, that is, when asked by the user. This encompasses two possibilities:
    a. The user completes a questionnaire about the company.
    b. A new vulnerability scan is launched.
2. “Semi-automatic” way, which considers four scenarios:
    a. The user changes any of the answers provided in the questionnaire: that would mean that there is a modification in the business indicators.
    b. The user chooses a different model to work with. This situation is detected by RAE and then a new risk assessment is launched.
    c. There is a modification of the vulnerability indicators, that is noticed when the vulnerability scan is completed, and results are interpreted.
    d. If there is a variation in events and alarms, which produce the algorithm to be launched again.

As commented before, RAE needs two inputs to operate:

1. Information of vulnerabilities of the monitored platform.
2. The user provides data about the monitored infrastructure.

Then, all the data is combined and a RAE’s algorithm can weight and assess how vulnerabilities found could influence the monitored platform, especially in terms of guaranteeing the confidentiality, integrity, and availability of the information.

As a general-purpose cyber-risk assessment solution, RAE offers a pre-built framework which enables the development and evaluation of specific use case models.

Regarding the FISHY project, the two main features to be considered are vulnerability- and impact assessment, for which the following specific components of RAE may be adapted on demand to meet the FISHY specifications:

- Company info: Both basic entity data (such as company description, industry, or managing staff) and set of more specific questions about the company (expectedly related to the assessment taking place) can be defined. Partners, external contractors, or any kind of third-party companies may be covered as well.
- Company assets: Whether these be physical servers, equipment, software, or any other kind of valuable company good involved in a risk scenario to be evaluated, the RAE covers a wide

range of variables describing the asset loss costs in case of a breach, and associated coefficients for its availability, confidentiality and integrity to be modelled.

- Data processing activities (DPA): Any critical operation involving company assets (such as daily supply chain facility processing) is conceived as a DPA, which defines specific processing threats, risks, and counter measures as model selection assistance. As stated before, a DPA may relate different companies incurring on a common processing or sharing situation.
- Risk models: This last element offers a set of 10 predefined well-known risk models to carry out the main risk assessment. This set may be possibly extended with new models to fit the specific supply chain end cases. More specifically, models evaluate risks within a predefined data processing activity under the following considerations:
    - Model methodology: Risk models may yield both qualitative and quantitative (economic losses) estimations, being the former suitable for impact assessment modelling as a partial goal of FISHY.
    - Indicators: These are the internal system state variables which take the role of inputs to the model assessment. Vulnerabilities scanned and detected by FISHY can in turn trigger specific indicators of the models defined, launching a re-evaluation or update of the previously computed risk models.
    - Mitigations: The report generated after the assessment if performed also features a list of suggested mitigations to secure or further reduce the risk of the target entity system. These, together with the previous methodological estimations, constitute an enriched source of maintenance data for system administrators or technical operators.

The current design of RAE allows not just for custom risk model implementation, but also for vulnerability alerting via the built-in message broker. Support for report re-evaluation is provided out of the box, while a whole model needs to be defined first.

Furthermore, as part of the RAE end-user interface, REST endpoint or integration APIs may be served by the web application, so as any kind of read or write access to the internal data model.

4.2.2 Incident Detection

Monitoring capabilities of XL-SIEM require of agents to be deployed on the target infrastructure. These agents gather all sort of data related to security incidents and send it to the SIEM. Later, this information will be correlated and processed by the XL-SIEM engine resulting in refined data, more suitable to be harnessed by FISHY. Incident detection is an essential process as part of the FISHY Trust Manager and, more specifically, of the Trust & Incident Manager.

XL-SIEM is made up of three components:

1. XL-SIEM **agent(s)** gather and normalize events. The agent is designed to send information of events to the engine where they are processed. Agents need to be deployed on the target infrastructure. Apart from gathering information, the agents deal with the translation of data to security events. According to the context, some events could possibly be processed by agents prior to their shipping to the engine. One example of this behaviour would be the anonymization of data.
   Agents can be customized to gather data from different explicit sources, always depending on the use case.
2. XL-SIEM **engine** is aimed at analysing and processing events collected by the agents. The engine raises alarms according to the correlation rules and security directives. Event Processing Language (EPL) is the way security rules are expressed in XL-SIEM. EPL allows for declaring security directives, which may have complicated patterns, in a simple and straightforward way. When defining security directives, XL-SIEM can work in two ways:

a. Rules customized by the user, who has the possibility to determine and select his own rules.
b. Several pre-established categories of rules: in this scenario, the user should choose amongst categories such as malware detection, brute-force attacks, network attacks… which are different categories for which security directives have already been defined.

The engine runs on an Apache Storm framework [28] and produces alarms expressed in a standard format such as JSON. This allows for an easy sharing of the information.

3. XL-SIEM **dashboard** provides data in a friendly way by means of a graphical UI. It displays different kind information such as events, charts, graphics, etc and helps users defining the setup of the tool in a friendly way. In addition, the dashboard is a fully configurable view where various widgets provide information on KPIs, security trends and more. One example of the KPIs displayed by the dashboard includes the events and alarms by hour managed by the XL-SIEM.

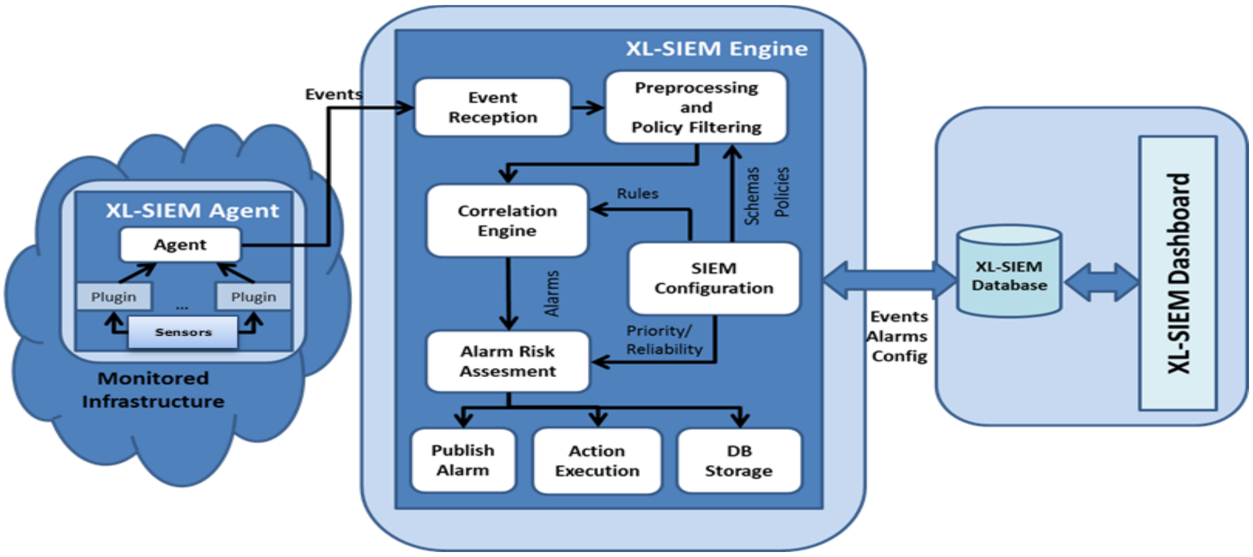


Figure 24. Architecture of the XL-SIEM

In addition, XL-SIEM integrates quite efficiently with different sensors, including DNS traffic sensors (for DoS detection), IDS (both NIDS and HIDS), firewalls or even honeypots and honeynets. This non-intrusive interaction makes possible that XL-SIEM can help with incident detection without requiring neither a great amount of time nor expertise.

Besides, some fine tuning can be performed on XL-SIEM based on FISHY's requirements:

- Develop new ad-hoc sensors to gather data and ease detection events according to specific features of the monitored infrastructure. This could be complemented with the refinement of event parsing rules, as needed.
- Improvement of the internal logic of the correlation engine. This means providing new correlation rules for creating event alarms. These could even potentially feed other tools such as the RAE.
- Tailor alarms to the specific features of the monitored infrastructure.
- Integration of the dashboard with normalization software, depending whether data is required to be manipulated before being displayed.

A new functionality for filtering and separating the relevant information for different endpoints will be implemented.

4.2.2.1 Trust Monitor

A preliminary implementation of the Trust Monitor is available at https://github.com/shield-h2020/trust-monitor. In particular, it is a cloud-native application that could be deployed using Docker technology. A mechanism for the automation of such a deployment is provided within the same repository and leverages Docker Compose.

The current version of the TM supports several attestation drivers, including one for interacting with the Open Attestation Framework (OAT). This framework has been customised to support Docker container attestation and leverages TPM 1.2. We foresee an improvement of the aforementioned solution towards the development of a new attestation driver for a more recent attestation framework, Keylime[1]. This is a CNCF-backed project which allows performing remote attestation leveraging TPM 2.0. Setting aside the development of the attestation driver, several changes to Keylime are required in order to verify the integrity of the software running within Docker containers.

4.2.3 Impact Assessment

The purpose of the Risk Assessment Engine (RAE) is to assess cyber risk. As it has been described for the vulnerability assessment, RAE works in **real time** and executes a risk-model based algorithm. According to the results obtained in the assessment, RAE suggests some mitigation measures, which are represented to the user by the Dashboard or, to be more precise, by the Decision Support System, in a friendly way.

Although RAE was mainly developed to assess the risk level, it can perform other functions as well, including the estimation how serious can be any incident for the infrastructure or providing valuable insight about the impact on the business processes. It is vital not only to know that information but also what measures can be adopted to mitigate or, at least, reduce the risk level as much as possible. As an added value of the RAE tool, it can suggest mitigation measures that help to reduce the impact of risks on the monitored infrastructure.

4.2.4 Mitigation

4.2.4.1 PMEM

PMEM is an R-based application usable as an ML-base mitigation tool.

PMEM incorporates a threat/attack detection model generated with supervised machine learning. The model detects different types of attacks that can affect the system. When an attack is detected, the program triggers an alarm and suggests an action to take.

Attacks detected by PMEM will be stored in the threat/attack repository and used to train future models that eventually will replace the current model that contains the application.

The online mode is under development currently, and it will be completely functional for IT-2 of the FISHY project.

4.2.5 Threat/Attack Repository

The Threat/Attack Repository will handle multiple types of data, necessitating a careful consideration of the selection of storage technology. Relational databases, such as MySQL [29] or Postgres [30], should be well suited to definitions of supply chains, as nodes that comprise the supply chains and the organizations that are owners or otherwise involved form traditional relations that can be mapped

[1] https://keylime.dev/

efficiently. The metrics gathered from the monitored infrastructure on the other hand would be better served by a time-series database, such as InfluxDB [31], allowing easier identification of trends in the data. The volume of results of continuous metrics analysis would again be more suited to a different storage technology, namely document stores like Cassandra [32] or MongoDB [33]. As the Threat/Attack Repository component will expose a REST API, if the requirements and specifics of the different types of data strongly diverge, multiple storage technologies could be supported in a way that would remain transparent to other actors in the FISHY Platform. The vision for the Threat/Attack Repository for IT-1 is the use of a relational database, the performance when handling data from use case partners can then be measured and indicate the necessity for supporting multiple storage technologies.

The other important aspect of the Threat/Attack Repository is the inclusion of a pub/sub (publish, subscribe) layer, that will allow components that expect certain data as their input to be notified immediately when new data is available. This system can facilitate the immediate deployment of cybersecurity tools to the nodes of a newly registered supply chain, analysis of infrastructure metrics as soon as they are available, and the generation of new intents and policies based on analysis results of the TIM components. The most likely candidate for the implementation of the pub/sub layer of the Threat/Attack Repository is RabbitMQ [14], an AMQP [34] based message broker. It has a system of different types of exchanges that allow easy and fine-grained subscriptions to topics of interest, such as fan-out exchanges where all messages are delivered to all subscribers, or topic exchanges, where each message has a routing key, enabling subscribers to only receive a subset of all the messages published to the exchange. Subscribers can consume messages by defining queues that bind to combinations of exchanges and routing keys, thus defining the data sources the subscribers are interested in. Queues also enable the scalability of subscribers, with support for round-robin delivery of messages in case of multiple consumers.

4.2.6 Smart Contracts

The SC component includes three main (groups of) components namely the Relay Server, the Event Server and a collection of generic and FISHY specific smart contracts acting like DAPPs on top of the Ethereum Blockchain (ETH BC).

For a complete list of possible information that could be eventually included in the specification of the FISHY SC, the interested reader is requested to have a look at [35]. Note, however, that for the present prototype, a subset of all these capabilities will be implemented, based on the project needs, as will be described in the relevant part of section 4 of this document.

The next table provides a tabulated overview of the DAPPs that are foreseen to be provided by the SC context.

Table 2: List of DAPPs provided by the SC module

DAPP name	Description
EntityManager	Handles the UE and IoT entity types, such that their identities are categorised
EntityDataManager	Handles data logging operations of the registered identities (UE and IoT).

Indeed, as an extended measure of security, whenever a FISHY compliant device first enters the blockchain, it needs to also register its entity type, as well as the set of devices and platforms supported by the FISHY supply chain entities. Based on its type, every entity (UE or IoT) can interact with the blockchain by invoking a set of smart contracts.

Notably, all the DAPPs above are protected by the Identity Manager of FISHY (see above for a detailed description). In this sense, it is not possible to invoke them directly but, rather, only through the IdM (connected to Relay Server, through the relevant RESTful API). It is worth highlighting that the FISHY consortium will provide a pre-compiled and ready to use python client so that the UE can interact with the blockchain without knowing anything related to the blockchain per se.

Every time a DAPP completes a designated and well-defined set of steps, an ETH event [36] will be emitted, getting handled by the Event Server. The Event Server will feature a mechanism allowing third party applications and contexts to connect to it and get information in a real-time or online manner. This functionality will be backed up at least either by a publish-subscribe mechanism (e.g. based on the well-known AMQP protocol [34] or via a web-socket-based toolkit, facilitating real-time web-applications integration. Alternatively, any third party will be able to get this information posted to a pre-defined service API endpoint so that it may be further processed.

As regards the interaction of SC with the rest of the core components of FISHY, these are mostly because the SC is a gateway to the ETH BC infrastructure, whose security part severely lies on IdM functionality and the relevant smart contracts.

Since IdM and SC are very tightly connected, being utterly intertwined, the core architecture of SC has been already described in the IdM section. The SC core architecture is composed of two components (Relay Server and Event Server) together with the DAPPs.

The SC Relay Server acts as a proxy between the UE (or other modules integrating with the blockchain e.g. to store data) and the blockchain, particularly the Relay SC. Essentially, it exposes part of the Relay SC in the form of a RESTful interface so that it might be exploited without necessitating mining or operation of exotic blockchain technologies at the UE or IoT side. When the request is not related to storing information or identities in the blockchain but, rather, to retrieving data from the blockchain, the Relay Server seeks the data from the SC Event Server.

The SC Event Server is a component that integrates with the SC DAPPs acting in the form of a cache, speeding up the data retrieval from the blockchain; whenever a transaction against one of the SC DAPPs gets accepted by the blockchain and enters a valid block, an Ethereum event gets emitted and written in the transaction log of the blockchain. The SC Event Server handles such events and stores them locally, always holding a local copy of the transaction logs of the blockchain. Periodically, the event server performs a self-assessment, refreshing the database to ensure integrity with the blockchain data. Similarly, the SC Event Server has the capability to re-build its internal database once this has been destroyed or upon initialization.

4.3 Metrics Gathering Tools

To facilitate the identification and monitoring of metrics, we have developed a list of possible tools capable of meeting these requirements.

4.3.1 Nagios

Nagios is a monitoring system that allows supervising and controlling the entire IT infrastructure ensuring that systems, applications, services, and business processes work properly. If any process fails, the technical team intervenes almost immediately to bridge the failure before it affects business processes, end-users, or customers [37].

Nagios provides agentless and agent-based solutions to monitor Windows, Linux, and Unix systems, as well as network equipment (include operating system metrics). Furthermore, agentless technologies are used to monitor solutions without the need to install agent software on each monitored system.

Nagios incorporates a set of plugins for monitoring different types of system metrics, such as:

- Number of available cores of a CPU
- Maximum time to a file being inside a folder
- Percentage available physical memory on a Linux system

- Delays and inconsistencies in connection
- Average time between request and response (“Black box”)
- Number of sensors alive (“Black box”)
- Number of UDP packet loss (“Black box”)
- Check if specified process name can be found listening on a specified TCP port (“Black box”)

4.3.2 OSSEC

Open Source HIDS SECurity (OSSEC) is a robust open-source **intrusion detection system** that performs log analysis, integration of logs, file integrity monitoring, Windows registry monitoring, centralized policy enforcement, rootkit detection, real-time alerting, and active response from multiple devices and formats running on most operating systems. This tool has a centralized, cross-platform architecture, allowing multiple systems to be monitored, managed, as well as analyzing firewalls, IDSs, web servers, and authentication logs [38]. In OSSEC we define metrics based on features such as the following:

- Number of applications installed on your client box
- Checking if exist some changes in a rule in a firewall
- Number of non-public rootkits
- Number of hidden ports
- Number of TCP and UDP ports on the system
- Minimum time to block an IP
- Number of service unavailable

4.3.3 XL-SIEM

As it has been described, XL-SIEM (Cross-Layer SIEM) is a tool developed by ATOS with the purpose of dealing with huge volumes of security information. After analysing and correlating the data, and depending on the context, the XL-SIEM could raise security alerts. Besides, being able to process and provide security and event information to FISHY could make the different when dealing with security incidents on the ICT supply chain of the monitored infrastructure. When answering to incidents, the feature of *active reconfiguration* of the monitored platform proposed by FISHY, can be easily performed if tools such as XL-SIEM feed with appropriate, reliable, and refined security information. Faster response times will be achieved in case data is available at the very moment of need.

Components such as incident detection can greatly benefit from relying on XL-SIEM as a source of information while mitigation of threats and security incidents will be more effective as long as the platform relies on updated security information.

Scalability is other of the various advantages of XL-SIEM, since the tool can distribute the information of security events to be processed by several nodes. Besides, XL-SIEM can obtain information from diverse kind of sensors including:

- DNS traffic sensor of ATOS, which aims at detecting botnets, DoS attacks or even brute force attacks.
- Network Intrusion Detection Systems or NIDS such as Suricata.
- Hosted IDS such as OSSEC.
- Information provided by firewalls.
- Tools that collect log data from the OS. One example is Snare (in a Windows environment).
- Tools that detect and prevent attacks on MAC / IP address such as Arpwatch.
- Information provided by Honeypots.

Finally, XL-SIEM can provide reports on PDF format which sum up all the activity performed by the tool in a specific timeline. Alarms, assets, information of events as well as several metrics can also be included in the report generated by XL-SIEM.

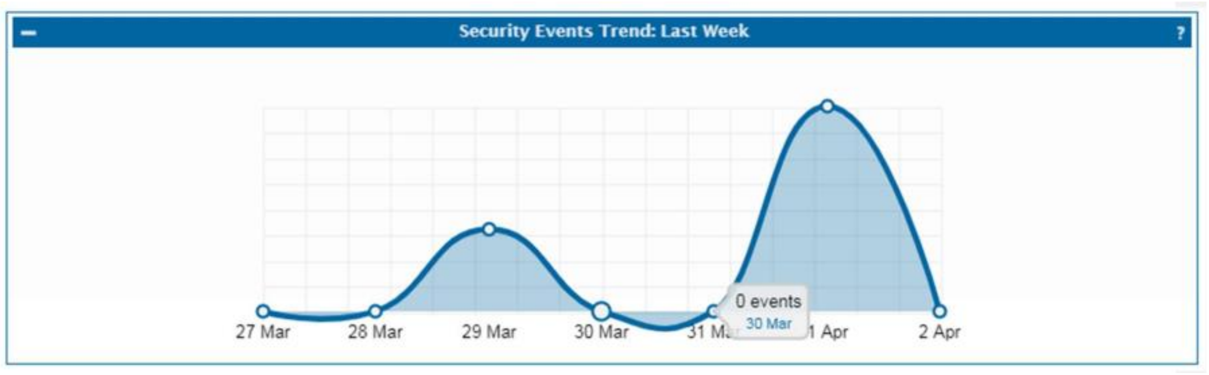


Figure 25. Example of security events graphic provided by XL-SIEM

To sum up, FISHY response against an incident cannot but improve when it is provided with reliable security and event information.

4.3.4 Wazuh

Wazuh helps organizations to detect intrusions, threats, and behavioral anomalies by collecting, aggregating, indexing, and analyzing security data. Fast threat detection and remediation is possible because the lightweight agent provides the necessary monitoring and response capabilities, and the server component supplies the security intelligence and performs data analysis. It can be deployed on-premises or in hybrid and cloud environments [13]. The Wazuh agents can run on many different platforms, such as Windows, Linux, Mac OS X, AIX, Solaris, and HP-UX.

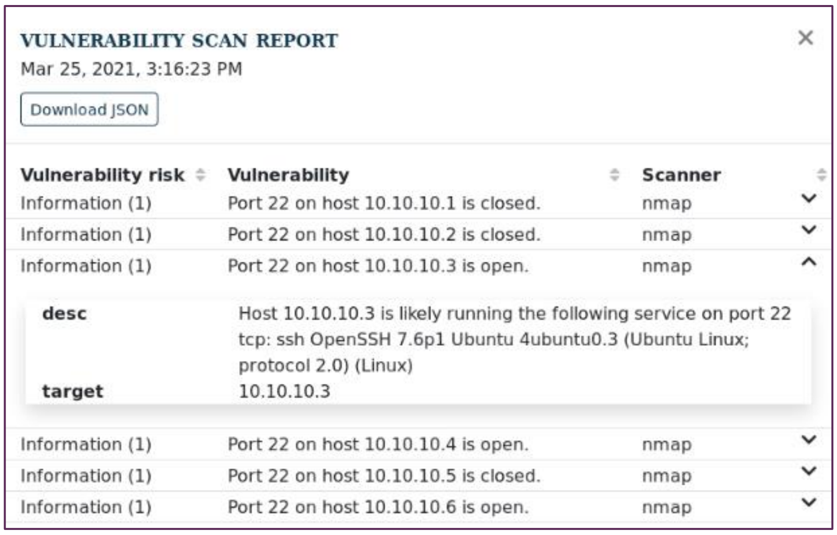
Wazuh has two methods for infrastructure monitoring: agent and agentless monitoring. "Agent monitoring can also be vendor-agnostic and uses a small client installed on servers to collect data and metrics. This typically allows for richer data and more flexibility. Agentless monitoring is relying on SNMP, WMI, SSH, NetFlow, and other protocols to retrieve metrics back to monitoring software, agentless monitoring is lightweight and is often enabled by default on your servers or devices. For specialized hardware (like routers, switches, and load balancers), this is usually your only option"[1].

This platform could protect monitor systems because providing capabilities like security analytics, intrusion detection, log data analysis, file integrity monitoring, vulnerability detection, configuration assessment, incident response, regulatory compliance, cloud security, and container security that are used for threat prevention, detection, and response. For each capability, Wazuh has some process where it is possible to define metrics, for example:

- Number of files that end with .log
- Maximum recursion level allowed
- Percent of hidden processes
- Maximum time to localhost response
- Checks that the output of the command contains a line starting by enabled, check if a registry exists
- Average time to resolve system vulnerabilities
- Number of cryptographic keys, number of publicly accessible buckets
- Number of applications installed in a container

[1] https://www.panopta.com/resources/agent-vs-agentless-monitoring/

4.3.5 VAT (XLAB)


VULNERABILITY SCAN REPORT
Mar 25, 2021, 3:16:23 PM
Download JSON

Vulnerability risk	Vulnerability	Scanner
Information (1)	Port 22 on host 10.10.10.1 is closed.	nmap
Information (1)	Port 22 on host 10.10.10.2 is closed.	nmap
Information (1)	Port 22 on host 10.10.10.3 is open.	nmap
desc	Host 10.10.10.3 is likely running the following service on port 22 tcp: ssh OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0) (Linux)	
target	10.10.10.3	
Information (1)	Port 22 on host 10.10.10.4 is open.	nmap
Information (1)	Port 22 on host 10.10.10.5 is closed.	nmap
Information (1)	Port 22 on host 10.10.10.6 is open.	nmap

Figure 26. Example of a VAT report

The VAT is composed of several components, a scheduler tasked with performing vulnerability assessments at predefined intervals, a REST API that allows communication and integration with other components (HTTP and AMQP), a docker interface that is able to start the VAT containers on-demand, a webUI frontend that gives users an overview of running scans and the VAT docker image that includes W3AF, OWASP and NMAP assessment tools

The VAT performs web server and infrastructure vulnerability scans and assessments according to an execution schedule and produces reports of its findings. The reports are in JSON format and can be viewed through the webUI (depicted in Figure 26) or passed on to other components for further analysis.

5 Conclusions

The blocks constituting the Trust Manager (TM) have been identified and described, putting them in the context of the whole FISHY framework. A modular description of these blocks has been provided, discussing the functional characteristics, and required interfaces of the individual modules, and considering the relevant workflows in which they participate. Finally, the applicable tools identified by the project team to implement the discussed functionality are described, including the related features, and the necessary adaptations to interface them within the TM environment and with the rest of the FISHY framework.

References

[1] OpenID, Welcome to OpenID Connect. https://openid.net/connect/. Retrieved 2021-06-27.

[2] M. Jones, J. Bradley, N. Sakimura, JSON Web Token (JWT). https://datatracker.ietf.org/doc/html/rfc7519. Retrieved 2021-06-27.

[3] Yuan Cao and Lin Yang, A survey of Identity Management technology. 2010 IEEE International Conference on Information Theory and Information Security, 2010, pp. 287-293, DOI: 10.1109/ICITIS.2010.5689468.

[4] Talamo, M., Ramachandran, S., Barchiesi, M.-L., Merella, D. & Schunck, C., Towards a seamless digital Europe: the SSEDIC recommendations on digital identity management. Open Identity Summit 2014, Bonn.

[5] Rountree, D., Federated identity primer. Syngress, 2013.

[6] P. Samarati and S. Capitani de Vimercati, Access control: Policies, models, and mechanisms. Foundations of Security Analysis and Design, LNCS 2171. Springer, 2000.

[7] OASIS, eXtensible Access Control Markup Language (XACML) Version 3.0. http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-cs-01-en.pdf. Retrieved 2021-06-27.

[8] Ravidas, S., Lekidis, A., Paci, F., & Zannone, N., Access control in Internet-of-Things: A survey. Journal of Network and Computer Applications 144, 79–101. 2019. DOI: 10.1016/j.jnca.2019.06.017.

[9] D. Hardt, Ed., The OAuth 2.0 Authorization Framework. https://datatracker.ietf.org/doc/html/rfc6749. Retrieved 2021-06-24.

[10] Ansible, Ansible is Simple IT Automation, https://www.ansible.com/. Retrieved 2021-06-14.

[11] W3AF, Open source web application security scanner, https://w3af.org/. Retrieved 2021-04-07.

[12] OWASP, Zed Attack Proxy, https://owasp.org/www-project-zap/. Retrieved 2021-04-07.

[13] Wazuh, The open source security platform, https://wazuh.com/. Retrieved 2021-04-07.

[14] RabbitMQ, Messaging that just works, https://www.rabbitmq.com/. Retrieved 2021-04-07.

[15] Redis, Redis, https://redis.io/. Retrieved 2021-04-07.

[16] Payne, S. C., A guide to security metrics. SANS Institute Information Security Reading Room, 2006.

[17] Jansen, W. A., Directions in security metrics research. Diane Publishing, 2009.

[18] J. Poppelbuss, M. Roglinger, What makes a useful maturity model? a framework of general design principles for maturity models and its demonstration in business process management. ECIS 2011.

[19] NIST SPSP 800-53 Rev. 5, Security and Privacy Controls for Information Systems and Organizations. 2021.

[20] CIS, Center for Internet Security. https://www.cisecurity.org. Retrieved 2021-04-12.

[21] Keycloak, Open Source Identity and Access Management. https://www.keycloak.org. Retrieved 2021-06-27.

[22] OASIS, Security Assertion Markup Language (SAML) V2.0 Technical Overview. https://www.oasis-open.org/committees/download.php/27819/sstc-saml-tech-overview-2.0-cd-02.pdf. Retrieved 2021-06-27.

[23] 5GROWTH Project, 5G-enabled Growth in Vertical Industries. https://5growth.eu. Retrieved 2021-06-27.

[24] ETSI White Paper No.31, NGSI-LD API: for Context Information Management, January 2019.

[25] Apache Foundation, Apache NiFi: An easy to use, powerful, and reliable system to process and distribute data. https://nifi.apache.org. Retrieved 2021-06-27.

[26] Apache Foundation, Apache Flink - Stateful Computations over Data Streams. https://flink.apache.org. Retrieved 2021-06-27.

[27] Apache Foundation, Apache Kafka. https://kafka.apache.org. Retrieved 2021-06-27.

[28] Apache Foundation, Apache Storm. https://storm.apache.org. Retrieved 2021-06-27.

[29] MySQL. https://www.mysql.com/. Retrieved 2021-06-14.

[30] PostgreSQL: The world's most advanced open-source database. https:/www.postgresql.org/. Retrieved 2021-06-14.

[31] InfluxData, InfluxDB: Purpose-Built Open-Source Time Series Database. https://www.influxdata.com/. Retrieved 2021-06-14.

[32] Apache Foundation, Apache Cassandra. https://cassandra.apache.org/. Retrieved 2021-06-14.

[33] MongoDB, The database for modern applications. https://www.mongodb.com/. Retrieved 2021-06-14.

[34] Advanced Message Queuing Protocol. https://www.amqp.org/. Retrieved 2021-06-14.

[35] Web3 ETH API. http://web3py.readthedocs.io/en/stable/web3.eth.html. Retrieved 2021-06-01.

[36] Solidity - Events. http://solidity.readthedocs.io/en/v0.4.21/contracts.html#events. Retrieved 2021-06-01.

[37] Nagios - The Industry Standard In IT Infrastructure Monitoring. https://www.nagios.org. Retrieved 2021-04-12.

[38] OSSEC - World's Most Widely Used Host Intrusion Detection System – HIDS. https://www.ossec.net/. Retrieved 2021-04-13.