D2.4
CS-AWARE Framework

Grant Agreement number:	740723
Project acronym:	CS-AWARE
Project title:	A cybersecurity situational awareness and information sharing solution for local public administrations based on advanced big data analysis

Principal author:	Veronika Kupfersberger, University of Vienna, veronika.kupfersberger@univie.ac.at

Document version:	2

Table of Contents

Document Revision History

Revision #	Document Version	Revision	Pages	Recommendation
1	2	Addition to Introduction – Paragraph on Section 4	6	Recommendation 1 regarding period covered - ... '*more information about the APIs*' ...
2	2	Update of the Framework (Diagram 1) – Addition of 2 relationships	7	Recommendation 1 regarding period covered - ... '*updated version of the architecture of the framework* ' ...
3	2	Inclusion of table with subprocess and respective features (Tables 3 - 9)	11 – 18	Recommendation 1 regarding period covered - ... '*internal architecture and functionality of each of the components* ' ...
2	2	Update of the Framework (Diagram 3, Diagram 4) – Addition of relationships between layers	14, 16	Recommendation 1 regarding period covered - ... '*updated version of the architecture of the framework* ' ...
2	2	Update of the Framework (Diagram 1) – Addition of relevant input / output fields in Interface Specification tables (11, 14, 15, 16)	20, 21, 24 – 30	Recommendation 1 regarding period covered - ... '*updated version of the architecture of the framework* ' ...
4	2	Inclusion of functionalities of subprocesses in the respective interface specification tables (Tables 11 - 18)	20 – 33	Recommendation 1 regarding period covered - ... '*internal architecture and functionality of each of the components* ' ...
5	2	API Documentation Includes: Sequence Diagrams, Endpoints specifications, exemplary requests and responses	34 – 48	Recommendation 1 regarding period covered - ... '*more information about the APIs*' ...
6	2	Mapping of functionalities and internal architecture (subprocesses) of components to requirements see: Deliverable 2.2 - Table 4, p. 13	13 (D2.2)	Recommendation 1 regarding period covered - ... *'functionality/ internal architecture matches the selected requirements* ' ...

Executive Summary

This deliverable describes the CS-AWARE Framework by defining the building blocks, the interfaces between them and the relevant information flows. By outlining the high-level relations between the modules and how they will be designed on a basis for further development and more detailed specifications in the upcoming work packages is provided, all in-line with the agile software development methods applied to this project.

The first step in developing the framework was to assess the current technical specifications and degrees of flexibility of the technologies to be used. Since all tools are developed by consortium partners there were no limitations detected. Nevertheless, it was essential to define general requirements for the CS-AWARE solution and how information is to be shared among the components and which data transformations might be required. This is described in the framework by using diagrams of various abstraction levels – beginning with a high-level overview of the main technology concepts and providing more detail in thematically differentiated levels. A three-layered division based on the modules' functions was chosen to more optimally visualise the framework and the respective relations: Data Extraction, Data Transformation and Data Provisioning. A complete overview of all the layers and their respective components can be found in Annex 1.

To ensure all partners share a common understanding of the interactions between their technologies, the interface definitions were specified on a high-level basis, defining their subcomponents and data formats to be used. These details are described in the framework based on the I/P/O model, input – process – output, for each component individually.

The CS-AWARE Framework is to be considered a basis for all further project related software architecture and implementation efforts. It describes the main components

System Dependency Analysis, Data Collection, Data Pre-Processing, Data Analysis and Pattern Recognition, Multi-Language Support, Visualisation, Self-Healing and *Information Sharing*

and their relations

data flow, control flow and *guidance*.

One of the first aspects analysed for this framework were the relations between the components and what defines them. It became clear that while all can be summarized under the term 'information flow' they differ in content. The differentiation was made between data and control flow, where the first describe actual data transfer for analysis purposes and the latter conceptual and logical information which has influence on the execution of the receiving component. The last relation, that of guidance, represents a relation where one component has an extensive effect on another and its setup.

As with any IT related project, the chosen formats and guidelines represent the current state-of-the-art and may be subject to change. Should this be the case, these alterations will be reflected in the deliverables of WP3 or WP4.

1 Introduction

The CS-AWARE Framework is the fundament of the CS-AWARE solution and is based largely on the piloting workshops in the local public administrations (LPA) of Larissa and Rome and the resulting deliverable D2.1 (D2.1 System and dependency analysis (first iteration) – Cybersecurity requirements for local public administrations). The aim of the framework is to provide a unified understanding of which components interact with each other and in what way this interaction is made possible. The framework provides a high-level overview of the main components, most of which are represented by one of the consortium partners, as well as a more detailed view of the main subcomponents or processes each of them consist of. Additionally, the relations between these components are defined as well as, in the case of data flows, the data format in which the exchange takes place. The high-level nature of the framework was crucial, since some technical details will only be specifiable during the projects implementation phase reflected in the upcoming work packages and will be substantiated in D4.1.

The CS-AWARE framework consists of an information flow model which is described in detail in the following chapter, as well as individual interface definitions for each of the components. The information flow model is explained on two levels of detail in this deliverable. The first is a high-level, abstract view on how each of the separate technology components cooperates with the others, which can be found in Section 2, and in what relation they stand to each other. This might be data flows or also logical control flows between the modules.

To facilitate further analysis, the detailed investigation into the appropriate connections was based on the ETL structured diagram. ETL stands for Extraction (Section 2.1), Transformation (Section 2.2) and Load and is a process most commonly used for database warehouses. Extract stands for the gathering of the data from various sources, transform for cleaning and manipulation of data to ensure integrity and completeness, load for transferring the data into its target space (Bansal & Kagemann, 2015). Since the CS-AWARE solution is evidently not a database warehouse, the final layer load was adjusted to better suit the frameworks nature and renamed Data Provisioning Layer (Section 2.3). In our case the division into layers will be mainly applied to facilitate the structuring of the following, more detailed diagrams of the subcomponents, processes and their interrelations.

The Data Extraction layer covers all components responsible for defining relevant data and extracting it, as well as the sources themselves. The System Dependency Analysis (Section 2.1.1 – 2.1.4) is where the analyst defines relevant sources and data necessary for monitoring the LPAs systems. This information is fed into the Data Collection module (Section 2.1.5) via a control flow, which then extracts the data accordingly.

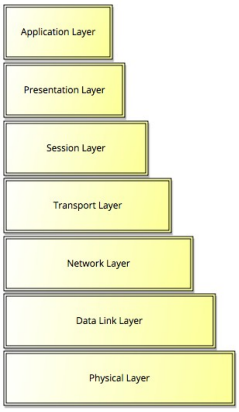


Figure 1 - ISO/OSI model

As will become clear in the detailed description of the LPA System Specific Sources (Section 2.1.3), the focus of the current design of CS-AWARE lies on layers 3,4 and 7, namely the Network, Transport as well as Application layers of the LPAs systems.

The Data Transformation layer summates all components tasked with transforming and analysing the data in some way. The first step is to filter and adapt the data as required before it can be, if necessary, run through the Natural Language Processing Information Extraction component (Section 2.2.1). The

Data Analysis and Pattern Recognition (Section 2.2.3) and the Multi-Language Support module (Section 2.2.2) further process the data.

For visualising and sharing the detected incidents and data patterns, the Data Provisioning layer was defined. This is where all collected information is either visually presented to the end user (Section 2.3.1), shared with selected Information Sharing communities (Section 2.3.2) or used for Self-Healing Rule definition (Section 2.3.3). A complete overview of the three layers and all subcomponents can be found in Annex 1.

The Interface Definitions also remain high-level, defining data formats and schemas where possible, but leaving a degree of flexibility for allowing decisions to be made in the software development phase of this project. While the interfaces are fixed, the detailed data schema can in some cases only be defined during the development of the detailed software architecture. The approach chosen to present the CS-AWARE framework interface specifications is based on the classical I/P/O - Input, Process, Output – model, where each component consists of as many input, process and output entities as is required.

For each component, all other building blocks providing data or control flows are summarized as inputs, including which data format they use and if applicable, which data schema as well as the type of flow they send out. Additionally, each component has one or multiple processes or sub components that execute the respective logic of the module and are described in detail as well. Each sub process has inputting and outputting components, data format specifications for each and underlying functionalities. Finally, the output components are defined by the same information as the inputs; data format, schema and which type of information flow they use.

Finally, the API documentation (Section 4) details how the individual interfaces between the components were implemented. An overview is given by sequence diagram outlining the communication flows between components and the respective endpoints. These endpoints are subsequently described in more detail and exemplary requests and responses are given to demonstrate a potential use case.

In preparation of conceptualising the framework, various models and approaches were researched. In the end the CS-AWARE Framework was based on the information flows between the components. Nevertheless, it is in line with the NIST Cybersecurity Framework (NIST National Institute of Standards and Technology, 2015), which identifies five functions as its core: Identify, Protect, Detect, Respond and Recover. All five steps are covered by the CS-AWARE Framework, making it also compliant to the Italian Cyber Security Report, which is based on the NIST Framework (CINI Cyber Security National Laboratory, 2016).

2 Information Flow Model

The information flow model described in this deliverable is one of the key components in the final CS-AWARE framework. It describes how each component is connected to the relevant other components as well as which type of relation they share. Based on this high-level overview of the CS-AWARE solution, the data formats and interface integrations used in the framework were further specified.

The high-level version of the information flow model in Diagram 1 shows the interactions between the major technology modules of the CS-AWARE solution. Additionally, the type of the relations has been included, being either data or control flows, automated or manual. This visualisation also shows each of the layers, differentiating between the Data Extraction, the Transformation and the Provisioning components of CS-AWARE.

The first step in any implementation of the CS-AWARE solution is the analysis of the given LPA. For the pilot use cases this of course includes the primary evaluation of all Public Sources as well, which will be investigated thoroughly and selected for all LPA implementations homogenously. Information on these public sources will be kept up-to-date by CS-AWARE and new settings are distributed if required. During the pilot workshops, as described in D2.1, a set of LPA System Specific Sources which are generalizable, were determined.

After these sources are determined, this information is inserted into the System Dependency Analysis tool, which collects and represents interdependencies between entities. This information is then distributed among the other components, providing information on how and what to proceed - represented via control flows.

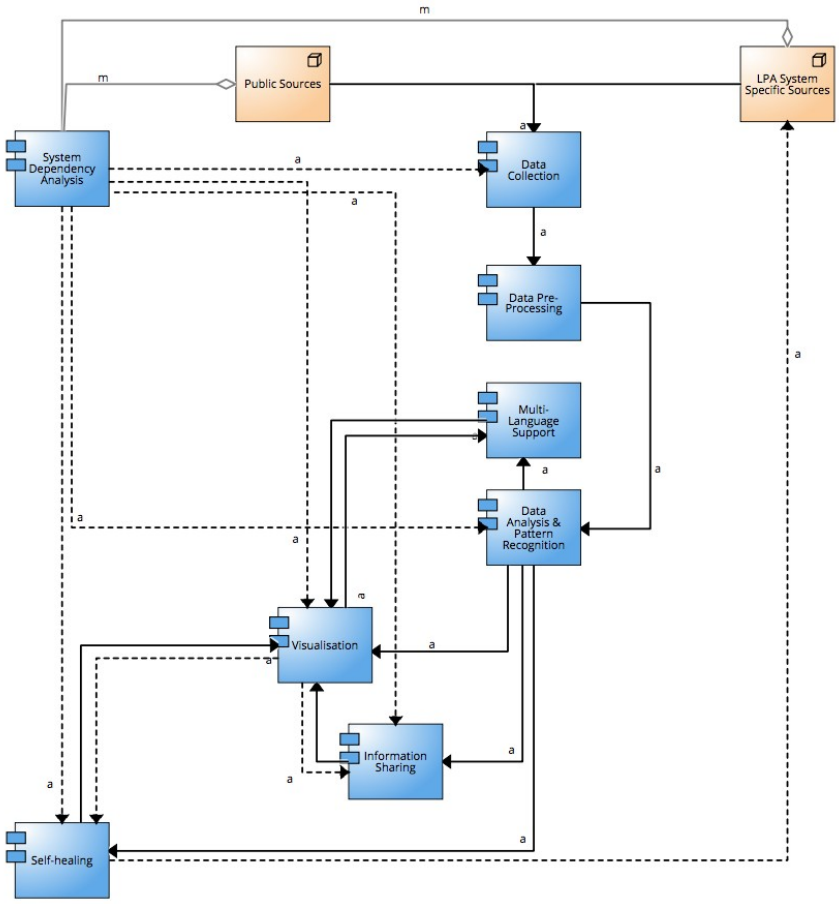


Diagram 1 - Overview Information Flow Model

Each of the tasks and relations that occur in any of the diagrams is explained in Table 1.

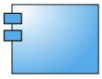
	External component
	Manual tasks
	External Information Sharing Communities
	Subcomponents / Modules
	Static data
	Guides conceptualization of one component is guided by composition of another
	Data flow data is transferred from one component to another
	Control flow logic and/or rules are initiated or defined by one component for another

Table 1 - Legend

The solid line represents the data flow that is being controlled by the outgoing component and the dotted line stands for information transmission from outgoing to the receiving component relevant to logical and or conceptual issues. This means that based on the information provided by the sending component, the receiving one adapts its settings and calibrations. The third relation represents that one component essentially guides the composition of another. Additionally, the manual ('**m**') as well as the automated ('**a**') connections are marked, each of which will be described in detail in the following chapters. One connection is described as semi-automated ('**a/m**') since the data being transmitted might require manual adaptations before sending.

The simple nature of the diagram offers an intuitive overview of the interaction between each of the components. Table 2 lists all high-level components which were identified to be the most relevant:

Component	Description
System Dependency Analysis	The System Dependency Analysis is performed by combining the Soft Systems Methodology and the GraphingWiki, resulting in a strategic implementation process for any Local Public Administration. Based on the pilot analyses, guidelines for future System Dependency Analyses will be developed.
Data Collection	The Data Collection will be undertaken by technology developed by 3rdPlace, one of the consortium partners. This tool allows the specification and collection of data sources relevant for future analyses. Additionally, this component will be responsible for developing the data collectors for LPA System specific data and storage.
Cybersecurity Information Exchange	This module is responsible for sharing information on detected attacks with authorities, according to the NIS regulations. The Cybersecurity Information Exchange provided by InnoSec will allow the user to individually authorize any transmission before it occurs.
Visualisation	The Visualisation component covers the final data manipulations required for graphically representing the collected information as well as the construction of the user interface.
Self-Healing	The Self-Healing component, also by InnoSec, will receive information from the Data Analysis module and compose Security Rules based on the detected incident. These

	rules can then be applied to the LPA specific systems by the respective IT departments.
Data Pre-Processing	One of the pre-processing strategies is the Natural Language Processing for Information Extraction, which is a simplification process of textual information. This feature has been developed by the University of Passau and can be used to simplify collected data. Other possibilities would include the simple filtering of known irrelevant data or the transformation of data formats.
Multi-Language Support	Due to the European context of CS-AWARE, the final UI should include not only easily understandable visualisations but all text in either the native language of the end user or English.

Table 2 - Component Description

The following Subsections outline each components individual architecture and their respective functionalities, all implemented to achieve the overall gaols of the holistic awareness generating solution the consortium has set out to develop.

2.1 Data Extraction Layer

For each layer, diagrams were devised to further elaborate on the connections between the modules. As seen in Diagram 2, the extraction level is divided into the building blocks.

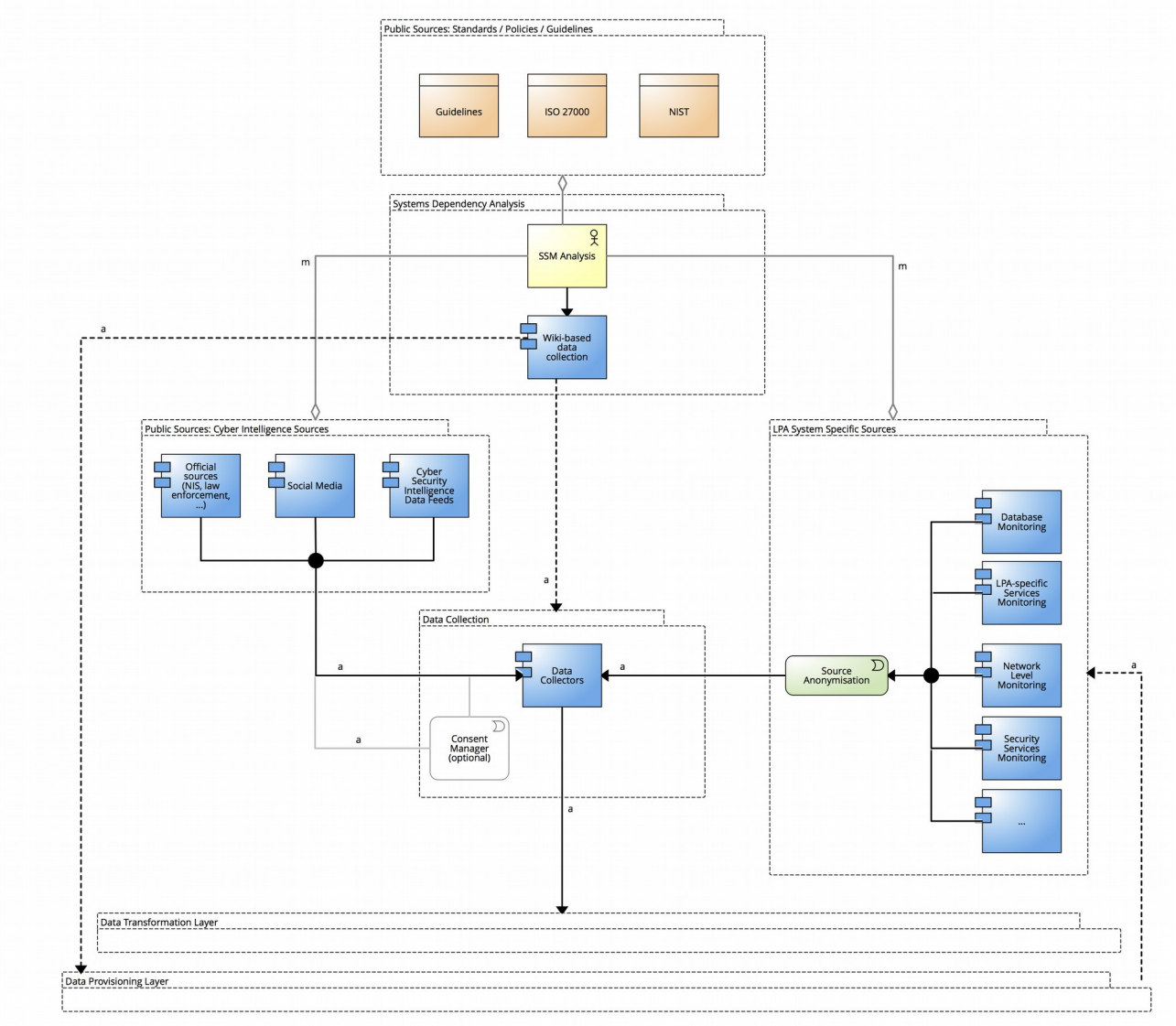


Diagram 2 - Data Extraction Layer

The Extraction layer combines the System Dependency Analysis, LPA System Specific and Public Sources, Guidelines and Policies as well as the Data Collection module. Their respective relationships to the other layers are only roughly represented in this diagram.

2.1.1 System Dependency Analysis

This system component is the first step when implementing CS-AWARE. Using the Soft Systems Methodology, a method which aims to identify software issues by questioning their users directly and letting them visualise their views of the system using rich pictures (Checkland & Soles, 1990), the local public administrations' systems are analysed.

During the pilot first round of workshops in Rome and Larissa, the following questions proved to be most efficient to identify the most vulnerable components:

Define the critical applications Without which data/ applications would the LPA not function?

- What are the backup capabilities of these applications?
- What are the existing monitoring systems for these applications?

Which monitoring systems are currently in place?

The System Dependency Analysis is strongly reliant on the analyst, in future either provided by the CS-AWARE team or an independent contractor. Not only the internal systems and the results of the workshop itself play a significant role but also public sources which will be selected during the pilot implementation of CS-AWARE and will be continually updated by the CS-AWARE team. The information collected by the analyst is inserted into the GraphingWiki, which allows a Dependency Graph representation, as well as a textual representation in JSON format, which will be used in multiple other components.

Table 3 shows the functionalities necessary to fulfil the functional requirements of the subprocess of the component.

Subprocesses	Functionalities
Wiki-based data collection	F1.1 CompilingAndGroupingData F1.2 GraphicalRepresentation F1.3 Repository

Table 3: Subprocesses and Functionalities of System Dependency Analysis

2.1.1.1 Standards, Policies and Guidelines

This component covers several different standards, policies and guidelines relevant to the Analysis phase as guiding principles for the analyst. The individual documents covered by this component may vary and will be regularly revised by the CS-AWARE team. The Guidelines mentioned are strategies for the analysis workshops identified during the pilot workshops, a summary of lines of questioning for directing the users in the correct direction as well as general principles of analysis. Policies and Standards will mostly cover judicial regulations or technical standards which need to be adhered to during the analysis process, such as the upcoming GDPR or NIS directive (General Data Protection Regulation, 2016) (Council, 2016).

It is assumed that the LPA in question has previously conducted a risk analysis and is aware of its main assets. One guideline to be used for such an analysis would be the BSI Catalogue (Federal Office for Information Security, 2017) should it not already have been conducted thus ensuring the compliance to Article 32 regarding the ‘Security of Processing’ of the GDPR (General Data Protection Regulation, 2016).

2.1.1.1.1 LPA System Specific Sources

As identified in the piloting Soft Systems Methodology workshops conducted in Larissa and Rome and described in deliverable D2.1, the most important LPA system specific sources are likely to be results of existing monitoring systems.

Database level

Depending on the LPA, their database structure will vary in size and complexity. Nevertheless, there are a few main database providers, which are most likely to be present in the LPAs. Such databases maintain logs on which users access what information and what is done to the data to guarantee access restriction and data integrity. These logs can be accessed by CS-AWARE and after anonymizing according to guidelines defined in deliverable 7.3 which is based on the GDPR standards, the data can be used and analysed for suspicious activities.

LPA-specific services level

This category covers the mission critical tools and services identified during the SSM workshops. While the responsibilities of an LPA are in general quite similar, each might have back-up strategies for different situations and/or services which another LPA might not. Identifying these critical services and implementing or activating monitoring functions are essential and will differ between the LPAs. Some of these systems might be handling citizen data and financial information or coordinate citizen or employee data, if either proves to be mission critical, it would be considered appropriate for constant monitoring. Many of these tools, as we discovered in the pilot workshops, already allow for monitoring of access and user interactions.

Network level

After identifying a single or at least main point of contact of all relevant and critical traffic, a possibility for monitoring said traffic must be identified. Usually routers offer such monitoring capabilities via built-in monitoring systems.

Security Services level

This level covers logs provided by any security appliance such as existing SIEM (Security Information and Event Management) systems, firewalls, Intrusion Protection Systems or Intrusion Detection Systems implemented by the LPA.

An additional potential source was identified but not included in the standard LPA system specific sources, namely the Client level, related to fleet management. The client level might not always be feasible to monitor. Nevertheless, it should be at least mentioned in the framework since it constitutes a major potential point of failure. Most successful attacks to an information system occur, as is commonly known, due to improper behaviour of users when using IT systems. Additionally, this component could potentially include the update status of the OS and any other relevant fleet management information that could be collected in the LPA.

The only requirement for the data format of these LPA system specific sources is that it must be textual, any other specifications are not yet determined and most likely will remain flexible. The most relevant aspect of internal data collection in LPAs is the anonymization of the data, which must occur at source. This is a process that is required by the EU for data protection purposes and will be implemented by the LPA individually or with support by the CS-AWARE team and is specified in the CS-AWARE deliverable D7.3 and D7.4. This ensures the compliancy with the Article 25 of the EU GDPR (General Data Protection Regulation, 2016) that requires ‘Data Protection by Design and by Default’. This means that systems must be designed to protect data they handle as well as set all default settings to the most secure versions.

Most fundamental LPA operations are similar. Therefore their systems can be expected to provide similar log results. 3rd Place will develop generalized data collectors, capable of collecting all relevant data after anonymization, which can be individually adapted according to the LPAs requirements and data structure.

While the actual information collected by the data collectors from the internal systems logs must be independently defined by the analyst before each individual CS-AWARE deployment, information on who accesses data at what time and for which purpose is likely to be essential. Therefore, it is advisable to consider the 5W1H model for questioning: who, what, where, when, why, and how? The sources selected during the System Dependency Analysis will provide data aimed at providing the answers to most of these questions.

2.1.1.2 Public Sources

As part of the conceptual phase of this project an analysis of potential information sources was conducted, results of which can be found in deliverable D2.1. Potential information sources in the following categories were identified.

NIS competent Authorities and Law Enforcement Agencies

Official sources include CERTs, CSIRTs, Europol or other entities offering valuable information on cyber threats. These institutions offer selective information on cyber

threats and will not only act as a source of information but also receive identified threats from CS-AWARE, thus complying to the NIS directive of the EU (Council, 2016), by the Information Sharing component. It is most likely that these sources will provide strategies, policies and guidelines for threat detection and prevention rather than raw data feeds. Nevertheless, these documents will provide meaningful guidelines for identifying and mitigating incidents and attacks.

Social Media and Cybersecurity visualisations

This information source is expected to result in early warnings on potential threats, albeit mostly unofficial. At the current point of development, it is most likely that CS-AWARE will commence with extracting data from two different Social Media platforms; Twitter and Reddit. Pre-defined keywords and/or users will limit the collected data and aim at ensuring high quality information.

Cyber Intelligence Sources and Information Sharing Tools (Open Source and Commercial), Cybersecurity Intelligence Data Feeds

This subcomponent covers not only the open source threat intelligence sources, but also commercial cyber security companies that collect relevant information. Some of the latter have been identified as using such data for their products and will be contacted by the consortium inquiring if a partnership and cooperation with this EU H2020 project would be of interest to them. While we aim at only utilizing open source data, this additional data would surely provide useful information, particularly during the development phase of CS-AWARE. The open source cyber intelligence data feeds that we came upon during the research for D2.1 all offer, as their name suggests, free and open data feeds.

Some sources offer their information as RSS feed, clear text files, while others provide APIs. The last group, the open source CTI sources, usually provide their information in STIX format via the TAXII mechanism. A more detailed description of the latter can be found in the chapter on data formats and transformations.

2.1.2 Data collection

After the data sources have been identified and where necessary anonymized at source, the data collectors gather the resulting data before transferring it to the storage. For the data collected from public sources, the framework has foreseen the optional implementation of a consent manager. This would allow citizens to contact CS-AWARE via the website and opt-out of their data being used for data analysis. An example for such a solution can be found at the SSIX project: **https://ssix-project.eu.** The necessity of such a consent manager will be evaluated in a later project phase.

The Data Collection component is also responsible for keeping the database up to date and ensuring the most current file formats are supported.

2.2 Data Transformation Layer

This layer includes all components responsible for transforming the data in some way. All data used in this layer is provided by the Data Collection component in the Data Extraction layer.

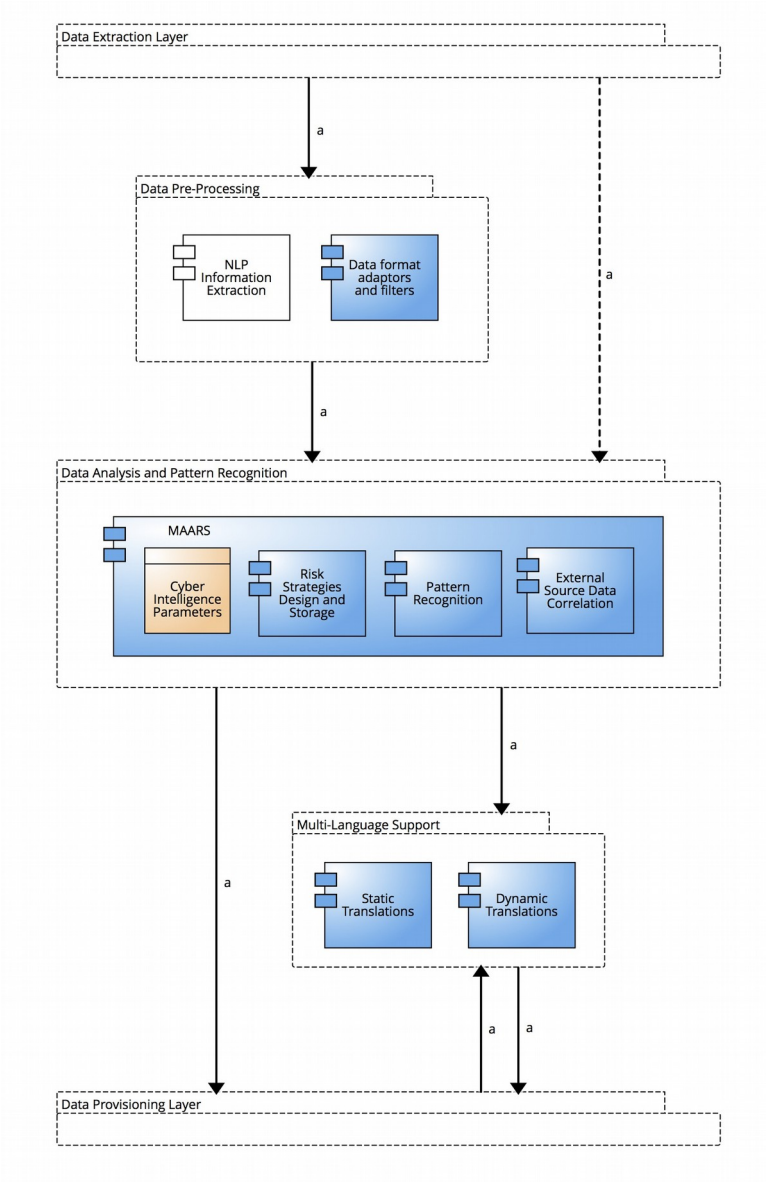


Diagram 3 - Data Transformation Layer

2.2.1 Data Pre-Processing

The Data Pre-processing step covers various methods which aim at structuring the data to a suitable format or excluding irrelevant information. This, of course, is only applicable for data that requires some sort of adaptation, all other data sets bypass the sub processes of this component. Additionally, these actions are required to optimize performance by constructing a more efficient data flow and limiting the amount of data passed on to the Data Analysis component. Currently, two main subcomponents have been identified: NLP Information Extraction and Data Filtering and Formatting, all of which are applied to the previously anonymized data. Natural Language Processing, provided by the University of Passau, will in this case be used to extract relevant information from a large data set. Due to the immense size of monitoring logs, information extraction and the consequential reduction of data will be advantageous. Another advantage is that this NLP information extraction process allows for information extraction from text in various languages, supporting the multi-language supporting characteristic of CS-AWARE. As semantic interpretation requires large amounts of computational power, it is important to identify the exact use cases in which this module proves beneficial. The Filtering subcomponent covers the limitation of the original data set to the most useful information by simple filtering rules. It aims to ensure not all data is used by the data analysis component, thereby increasing cost and computation time.

Data adaptors will be used to transform any suboptimal data structure into a more preferential format for the data analysis component. Regarding data from cybersecurity related data feeds, this might be

the adaptation of the actual data format or the transformation of STIX v1.x to its most recent version, currently STIX v2.0.

Table 4 shows the functionalities necessary to fulfil the functional requirements of the subprocesses of the component.

Subprocesses	Functionalities
NLP Information Extraction	F3.1 InformationExtraction
Data format adaptors and filters	F3.2 DataFormatter

Table 4: Subprocesses and Functionalities of Data Pre-Processing

2.2.2 Multi-Language Support

This component, also provided by the University of Passau, will translate processed and analysed data before it is represented visually to the end user. This is mostly for usability purposes and will be applied to data depending on the type of visualisation deemed most optimal. An exemplary scenario would be to apply the automatic translation function to information from official governmental institutions, such as CERTs, and their recommendations for mitigating threats. The extent to which this function will be included in the final CS-AWARE solution will be tested and evaluated during the implementation phase in WP4.

Additionally, this component covers the provisioning of any standard translations required for the GUI, the graphical user interface.

Table 5 shows the functionalities necessary to fulfil the functional requirements of the subprocesses of the component.

Subprocesses	Functionalities
Static Translations	F3.5.StaticGUITranslation
Dynamic Translations	F3.5.DynamicTranslation

Table 5: Subprocesses and Functionalities of Multi-Language Support

2.2.3 Data Analysis and Pattern Recognition

The data analysis is a key feature of the final CS-AWARE solution and uses the MAARS technology by Peracton Ltd for automated cyber incident detection. The first step is to identify cyber incident parameters with which new information received by the component can be compared to. For this purpose, the MAARS algorithm, currently used for financial decision making support, will be adapted and applied to the cyber threat domain. In the end the appropriate risk handling recommendation, either added by CS-AWARE experts or collected automatically from official institutions such as CERTs, is selected and offered to the user.

Table 6 shows the functionalities necessary to fulfil the functional requirements of the subprocesses of the component.

Subprocesses	Functionalities
Cyber Intelligence Parameters	F4.1 CyberIntelligenceParameters
Risk Strategies Design and Storage	F4.2 RiskPatternsStrategiesDesign
Pattern Recognition	F4.3 PatternRecognition
External Source Data Correlation	F4.4 ExternalSourcesDataCorrelation

Table 6: Subprocesses and Functionalities of Data Analysis

2.3 Data Provisioning Layer

The data provisioning layer covers all components in which the transformed data is provided to an end user.

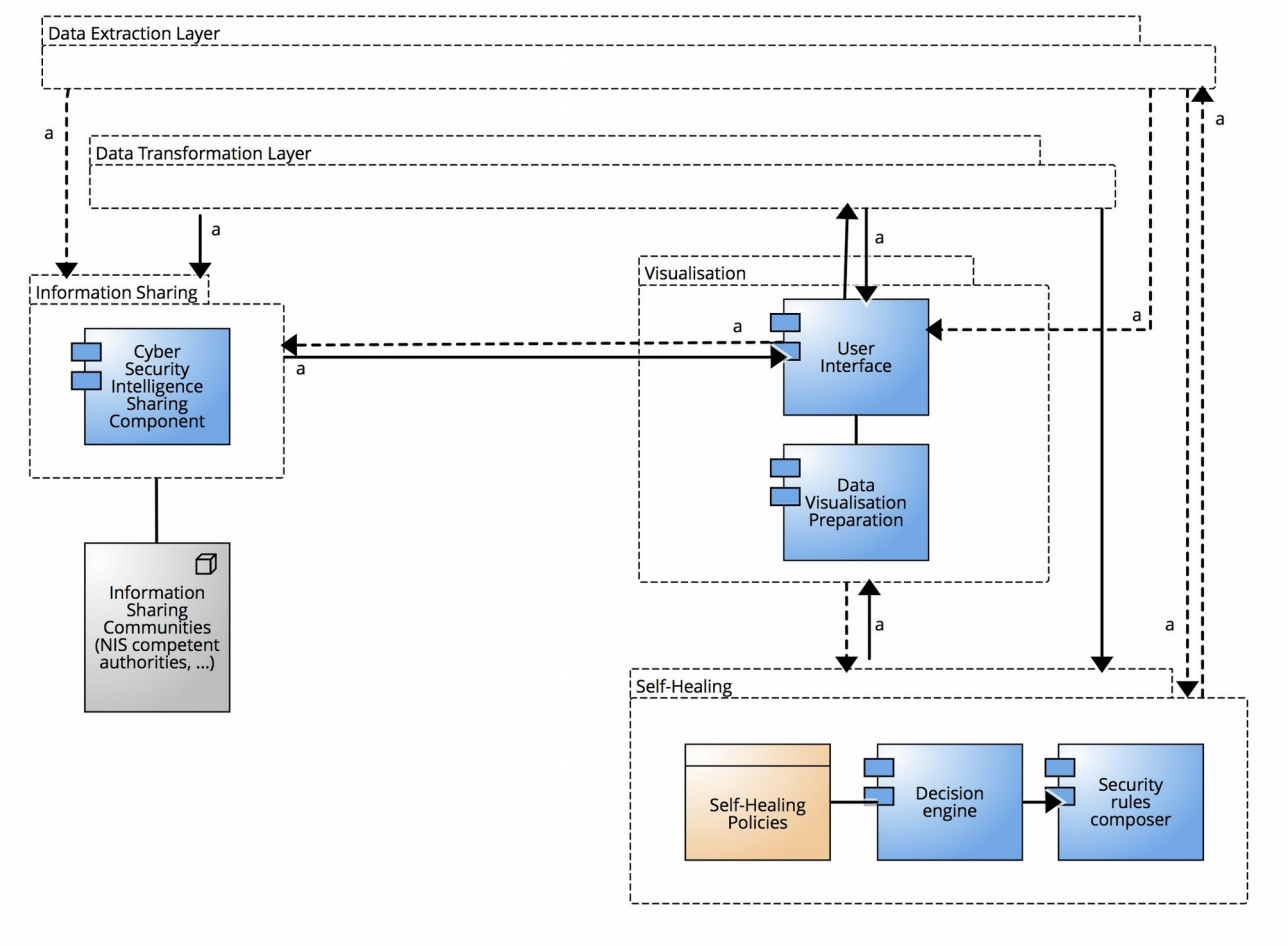


Diagram 4 - Data Provisioning Layer

2.3.1 Visualisation

This component is divided into two subcategories: the user interface and the data visualisation, both of which will be developed by CloudPartners.

The latter is responsible for formatting and preparing the data received from the Multi-Language Support and the Data Analysis and Pattern Recognition components for it to be presented to the end user. Next to these adaptations for presentation purposes, the main task of the Visualisation component is to, as the name suggests, visualise all output of CS-AWARE for the user. This does not only cover the analysis results, but also any administrative and reporting requirements. Part of the features available to administrative users is the opportunity to manage users and which rights they have. This will depend on their position in the company, employees from the IT department will require different function made available than those from the management level. Additionally, the administrative users will be able to import any information gathered in the GraphingWiki.

For LPA internal users the Visualisation component will make all relevant data available in a user-friendly and visually representable way. Also, it will provide updates on what can or should be shared with NIS competent authorities via the Information Sharing module as well as what steps should be taken to prevent attacks and mitigate incidents in the LPA's systems via the Self-Healing component. Both of which are described in more detail in the following subsections. Another vital part of this component is the triggering or authorization of above mentioned self-healing procedures and information sharing actions.

Table 7 shows the functionalities necessary to fulfil the functional requirements of the subprocesses of the component.

Subprocesses	Functionalities
User Interface	F6.5 VisualiseLPAInfo
Data Visualisation Preparation	F6.2 InfoSupplyVis_SH F6.1 CombineDA_SH F6.3 VisualiseSDA F6.4 ManageInfoIS

Table 7: Subprocesses and Functionalities of Visualisation

2.3.2 Information Sharing

This component covers the required sharing of detected cyber intelligence threat information with national institutions which are NIS competent authorities, so-called Computer Security Incident Response Teams (CSIRTs), in line with the information sharing requirements of the NIS directive (Council, 2016). In today's cyber security threat intelligence community, the importance of sharing information is becoming more apparent. In accord to this trend, the sharing component of CS-AWARE will not only benefit each LPA individually but also the Data Analysis component and therefore the CS-AWARE community. Naturally, CS-AWARE will comply with appropriate source protection mechanisms, like anonymization of shared content according to the NIS and GDPR guidelines.

All data handled by the Information Sharing component will be exclusively in JSON format, following the STIX schema as closely as possible, both of which are described in more detail in the following chapter. This will ensure that receiving institutions and organizations can understand the information sent by CS-AWARE, as it follows a commonly used schema. The STIX-based objects will be sent via the TAXII protocol, details on which can also be found in the upcoming sections. As can be seen in the diagram, the information sharing component currently consists of one sub process, which is responsible for sharing the accumulated information.

Table 8 shows the functionalities necessary to fulfil the functional requirements of the subprocess of the component.

Subprocesses	Functionalities
Cyber Intelligence Sharing Platform	F7.1 Authorisation F7.2 CTI_Sharing

Table 8: Subprocesses and Functionalities of Information Sharing

2.3.3 Self-Healing

The self-healing component is responsible for matching the results of the Data Analysis module with data stored in the Self-Healing Policy subcomponent. This static index lists potential threats or incidents that might be detected by the Analysis module and the respective handling strategies. This is done in the decision engine, which initiates the composition of a rule if a match is found. These rules will be composed in a Security rules language comprehensible for the LPA to ensure they can be directly implemented. An example might be that based on the detected incident, an addition to the security appliance configurations, such as firewall policies is required. This alteration will be directly offered by the Self-Healing component to the user who can copy it to the settings with as little alterations required as possible.

Table 9 shows the functionalities necessary to fulfil the functional requirements of the subprocess of the component.

Subprocesses	**Functionalities**
Self-healing Policies	F8.4 SelfHealingPolicies
Decision Engine	F8.1 IdentifyMitigationRule F8.3 ApplyMitigation
Security Rules Composer	F8.2 ComposeMitigation

Table 9: Subprocesses and Functionalities of Self-Healing

2.4 Data Storage

The diagrams of this framework focus on the dynamic flow of data and information between the components, excluding the static state of the data, the storage. Clearly, storage for each component is required both combined and individually, but the architecture to be applied has not yet been defined. One strategy is to centralize the data storage, allowing each component the read/write access it requires. In this case, each of the technologies is provided their logically separated space and access rights for the other components can be individually allocated, therefore ensuring every following component receives the data it requires.
The second strategy would be to de-centralize the data storage, where all technology providers are responsible for implementing and maintaining their own independent storage.
Both options will be evaluated based on their pros and cons and the final decision will be made during the software architecture development phase.

3 Interface Specifications and Data Transformations

3.1 Interface Specifications

The definition of the interfaces will facilitate future implementations of the CS-AWARE solution, and will therefore be required to follow a specified schema as shown in Table 10.

Input	**Name of Component**
Source Module	Name of Component
Data Format	Data format used by component
Description	must cover the following information: which type of incoming flow Schema in which the data must be structured

Output	**Name of Component**
Destination Module	Name of Component
Data Format	Data format used by component
Description	must cover the following information: which type of outgoing flow

Process	**Name of Process**
Module/s A	Name of input components
Module/s B	Name of output components
Process	Name of Process
Definition	Description of what happens in this subcomponent
Data Input Format	Data format used by input component/s
Data Output Format	Data format used by output component/s

Table 10 - Interface Definition Schema

The interface definition schema was based on the classical I/P/O model, which is commonly used to describe processes by defining input, process and output. It was considered appropriate for this framework, since it allows to modularize inputs, processes and output and combine them flexibly to define one interface. In the following, each components' interfaces from the high-level overview framework diagram are specified by defining all input components, each subcomponent or process as described in the chapter before including a list of their functionalities and finally all output components. Each building block of CS-AWARE can be described using 1..n inputs, 1…n outputs and 1…n processes. Additionally, this model directly and implicitly describes the data transformations necessary from the input to the appropriate outputs of each component.

3.1.1 System Dependency Analysis Interface Specification

Input	**Public Sources**
Source Module	Public Sources
Data Format	unstructured text
Description	manual analysis manual data flow

Input	**LPA System Specific Sources**
Source Module	LPA System Specific Sources
Data Format	unstructured text, anonymised
Description	manual analysis manual data flow

Process	**SSM Analysis**
Module/s A	Input Public Sources Input LPA System Specific Sources
Module/s B	Process Wiki-based data collection
Process	SSM Analysis
Definition	LPA specific structures and systems as well as public sources are analyzed and information on relevant data sources collected
Data Input Format	unstructured text
Data Output Format	unstructured text

Process	**Wiki-based data collection**
Module/s A	Process SSM Analysis
Module/s B	Output Self-Healing Output Information Sharing Output Visualisation Output Data Analysis Output Data Collection
Process	Wiki-based data collection
Definition	The information collected by the manual analysis is inserted, grouped and manipulated in the GraphingWiki. Graphical layout and state-fulness are provided. **Functionalities:** F1.1 CompilingAndGroupingData F1.2 GraphicalRepresentation F1.3 Repository
Data Input Format	unstructured text
Data Output Format	JSON

Output	Visualisation
Destination Module	Visualisation
Data Format	JSON
Description	Information on what data to display System data automated control flow

Output	Self-Healing
Destination Module	Self-Healing
Data Format	JSON
Description	Definition of what can be healed and which authorization is required automated control flow

Output	Data Analysis
Destination Module	Data Analysis
Data Format	JSON
Description	Definition of system automated control flow

Output	Information Sharing
Destination Module	Information Sharing
Data Format	JSON
Description	Definition of system automated control flow

Output	Data Collection
Destination Module	Data Collection
Data Format	JSON
Description	Definition of system automated control flow

Table 11 - System Dependency Analysis Interface Specification

As can be seen in Table 11, the System Dependency Analysis has two input components: Public Sources and LPA System Specific Sources. These two components guide the SSM Analysis process that is an essential part of the System Dependency Analysis. The analyst conducts a thorough analysis of the LPA specific systems and their properties as well as reviews the settings for the Public Sources data extraction. In the case of the System Dependency Analysis, one sub process directly triggers the next, where the analyst inserts all collected information into the GraphingWiki. This tool produces a visualisation of the dependencies of the analysed sources and the respective information they provide and is the basis for the LPA specific CS-AWARE implementation. As can be seen in Table 4, the output format of the dependency graph is a JSON file, which will be imported into the output components. The relationship to each of the output components is a semi-automated control flow, since the main purpose of the transferred JSON file is not to transmit data but control the logic of the receiving component and must be manually triggered in the GraphingWiki.

3.1.2 Data Collection Interface Specification

Input	**Public Sources**
Source Module	Public Sources
Data Format	flexible format – no specification
Description	automated data flow

Input	**LPA System Specific Sources**
Source Module	LPA System Specific Sources
Data Format	flexible format – no specification
Description	automated data flow

Input	**System Dependency Analysis**
Destination Module	System Dependency Analysis
Data Format	JSON
Description	information on which sources to collect from and what data to collect automated control flow

Process	**Data Collection**
Module/s A	Input System Dependency Analysis Input Public Sources Input LPA System Specific Sources
Module/s B	Process Data Storing
Process	Data Collection
Definition	Data collected from public and internal sources as directed by results from System Dependency Analysis is saved in repository. System, application, database and network devices logs are collected and anonymisation is provided. Also, installed packages within the device in STIX2 format (software/hardware installed on specific machines/servers), threat intelligence reports from multiple agencies and specialised cybrcecurity websites and relevant information from various cybersecurity specific accounts is collected. **Functionalities**: F2.1 Repository F2.2 LogCollection F2.3 PackagesCollection F2.4 ThreatCollection F2.5 SocialCollection
Data Input Format	flexible format – no specification
Data Output Format	flexible format – no specification

Output	**Data Pre-Processing**
Destination Module	Data Pre-Processing
Data Format	

	flexible format – no specification
Description	all source-anonymized data automated data flow

Table 12 - Data Collection Interface Specification

The Data Collection component is responsible for retrieving data as is described in Table 12. The components' extraction is defined by the information imported from the System Dependency Analysis. Various data collectors will gather information from multiple sources before it is pre-processed by the subsequent component. Since the sources and their formats may vary, it was essential to remain flexible at this point in the data manipulation process. Therefore, there was no specification made on which data format must be used for the Data Collection. Based on the analysis conducted for deliverable D2.1 we can assume that the most commonly provided data formats will be JSON, CSV or unstructured text.

3.1.3 Data Pre-Processing Interface Specification

Input	**Data Collection**
Source Module	Data Collection
Data Format	flexible format – no specification
Description	all anonymized data automated data flow

Process	**NLP Information Extraction**
Module/s A	Input Data Collection
Module/s B	Output Data Analysis and Pattern Recognition
Process	NLP Information Extraction
Definition	Extracting knowledge graphs from texts (n-ary relations and rhetorical structures extracted from complex factoid discourse). Given a sentence or a text, it outputs a semantic representation of the text which is a labelled directed graph (a knowledge graph). **Functionalities:** F3.1 InformationExtraction
Data Input Format	JSON
Data Output Format	JSON

Process	**Data Filtering and Formatting**
Module/s A	Input Data Collection
Module/s B	Output Data Analysis and Pattern Recognition
Process	Filtering
Definition	Data from external sources needs to be formatted in the standard data format used in the framework. Currently the standard is the STIX2 format.

	Functionalities: F3.2 DataFormatter
Data Input Format	flexible format – no specification
Data Output Format	JSON

Output	**Data Analysis and Pattern Recognition**
Destination Module	Data Analysis and Pattern Recognition
Data Format	JSON
Description	automated data flow

Table 13 - Data Pre-Processing Interface Specification

As can be seen in Table 13, the Data Pre-Processing component receives the data collected and transforms it as required. While the Natural Language Processing process requires JSON as input format, there are no such requirements for any other subcomponent. To enable the NLP Information Extraction, the relevant data is transformed into JSON and only then used for NLP purposes. The rest of the data can be filtered and formatted as required, making sure that no irrelevant or unreadable information is passed on to the Data Analysis component, to avoid unnecessary waste of time and costs.

3.1.4 Data Analysis and Pattern Recognition Interface Specification

Input	**Cyber Intelligence Parameters**
Source Module	Cyber Intelligence Parameters
Data Format	JSON
Description	Individual cyber intelligence parameters are defined and instantiated within MAARS decision engine to be used in various cybersecurity patterns. **Functionalities:** F4.1 CyberIntelligenceParameters automated data flow

Input	**System Dependency Analysis**
Destination Module	System Dependency Analysis
Data Format	JSON
Description	information on system dependencies automated control flow

Input	**Data Pre-Processing**
Source Module	Data Pre-Processing
Data Format	JSON
Description	automated data flow

Process	**Risk Strategies and Design**

Module/s A	Input Cyber Intelligence Parameters Input Data-Preprocessing Input System Dependency Analysis
Module/s B	Output Information Sharing Output Self-Healing Output Multi-Language Support Output Visualisation
Process	Risk Strategies and Design
Definition	Unique risk patterns are created and saved based upon the existing individual cyber intelligence parameters. **Functionalities:** F4.2 RiskPatternsStrategiesDesign
Data Input Format	JSON
Data Output Format	JSON

Process	**Pattern Recognition**
Module/s A	Input Cyber Intelligence Parameters Input System Dependency Analysis Process Risk Strategies and Design Input Data-Preprocessing
Module/s B	Process ExternalSourcesDataCorrelation Output Information Sharing Output Self-Healing Output Multi-Language Support Output Visualisation
Process	Pattern Recognition
Definition	The provided data is run against the pre-defined cybersecurity patterns and then MAARS returns exact matches or closest fit of suspicious activity/ potential attacks. **Functionalities:** F4.3 PatternRecognition
Data Input Format	JSON
Data Output Format	JSON

Process	**ExternalSourcesDataCorrelation**
Module/s A	Process PatternRecognition Process Risk Strategies and Design Input System Dependency Analysis Input Data-Preprocessing
Module/s B	Output Information Sharing Output Self-Healing Output Multi-Language Support Output Visualisation
Process	Pattern Recognition

Definition	Data from external sources it is extracted and added so when detecting possible threats to also see if similar patterns have been already flagged out by external data sources. **Functionalities**: F4.4 ExternalSourcesDataCorrelation
Data Input Format	JSON
Data Output Format	JSON

Output	**Multi-Language Support**
Destination Module	Multi-Language Support
Data Format	JSON
Description	automated data flow

Output	**Visualisation**
Destination Module	Visualisation
Data Format	JSON
Description	automated data flow

Output	**Information Sharing**
Destination Module	Information Sharing
Data Format	JSON
Description	automated data flow

Output	**Self-Healing**
Destination Module	Self-Healing
Data Format	JSON
Description	automated data flow

Table 14 - Data Analysis and Pattern Recognition Interface Specification

As shown in Table 14, the Data Analysis and Pattern Recognition component is responsible for comparing the data with predefined parameters of cyber incidents and detect any suspicious occurrences. This is done in the MAARS component by using the Risk Strategies Design and Storage subcomponent. Risk prevention and mitigation strategies is where the detected incidents are matched and the resulting actions listed. This way the Data Analysis module triggers the correct steps and alerts the right components in the CS-AWARE solution.

3.1.5 Multi-Language Support Interface Specification

Input	**Data Analysis and Pattern Recognition**
Source Module	Data Analysis and Pattern Recognition
Data Format	

	JSON
Description	automated data flow

Input	**Visualisation**
Source Module	Visualisation
Data Format	JSON
Description	automated control flow

Process	**Dynamic Translations**
Module/s A	Input Visualisation Input Data Analysis and Pattern Recognition
Module/s B	Output Data Visualization
Process	Dynamic Translations
Definition	This component translates instructions and alerts, such as threat descriptions and self-healing suggestions, from English to the user's mother tongue (Italian or Greek). **Functionalities:** F3.5.DynamicTranslation
Data Input Format	JSON
Data Output Format	JSON

Process	**Static Translations**
Module/s A	Input Data Analysis and Pattern Recognition
Module/s B	Output Data Visualization
Process	Static Translations
Definition	This component translates GUI static label and messages to user's mother tongue (Italian or Greek). **Functionalities**: F3.5.StaticGUITranslation
Data Input Format	JSON
Data Output Format	JSON

Output	**Visualisation**
Destination Module	Visualisation
Data Format	JSON
Description	automated data flow

Table 15 - Multi-Language Support Interface Specification

The Multi-Language Support component summarized in Table 15, translates text into the LPA specific language if required and offers standard text for the graphical user interface in the appropriate language. This is a feature that will be used mostly for recommendations on mitigation and prevention strategies, which will be presented to the users depending on which incident was detected. For this project, the two national languages of the piloting countries, Italian and Greek, will be made available in addition to the English vocabulary.

3.1.6 Visualisation Interface Specification

Input	**Data Analysis and Pattern Recognition**
Source Module	Data Analysis and Pattern Recognition
Data Format	JSON
Description	automated data flow

Input	**Multi-Language Support**
Source Module	Multi-Language Support
Data Format	JSON
Description	automated data flow

Input	**System Dependency Analysis**
Source Module	System Dependency Analysis
Data Format	JSON
Description	what to display and how automated control flow

Input	**Self-Healing**
Source Module	Self-Healing
Data Format	JSON
Description	automated data flow

Input	**Information Sharing**
Source Module	Information Sharing
Data Format	JSON
Description	automated data flow

Process	**User Interface**
Module/s A	Input Data Analysis and Pattern Recognition Input Multi-Language Support Input System Dependency Analysis
Module/s B	Output Self-Healing Output Information Sharing
Process	User Interface
Definition	

	Based on data from Graphing Wikki it is possible to draw a diagram (edge - vertices) showing the relations between the different components. For now the user can select a JSON formatted file with the information. It is possible to edit the information. It also possible to edit, add or remove components. Changed information can be exported to a JSON file **Functionalities:** F6.5 VisualiseLPAInfo
Data Input Format	
	JSON
Data Output Format	
	JSON

Process	**Data Visualization**
Module/s A	
	Input System Dependency Analysis Input Multi-Language Support Input Self-Healing Input Information Sharing Input Data Analysis and Pattern Recognition
Module/s B	
	Output Self-Healing Output Information Sharing
Process	
	Data Visualization
Definition	
	Internal interface to elastic search where threats are stored. It is only used internally. Self healing and visualisation can request list of accepted and declined threats and state. In visualisation there is a number of screens that show different aspects of information. We have a screen with a dart board layout of identified threats for a given day. A screen show active threats, another show mitigated threats. Yet another screen is used to show an overview of the components in the LPA. A screen exists to manage user and access. And we have a screen for language selection. **Functionalities:** F6.1 CombineDA_SH F6.2 InfoSupplyVis_SH F6.3 VisualiseSDA F6.4 ManageInfoIS
Data Input Format	
	JSON
Data Output Format	
	JSON

Output	**Information Sharing**
Destination Module	
	Information Sharing
Data Format	
	JSON
Description	
	automated control flow

Output	**Multi-Language Support**
Destination Module	
	Multi-Language Support
Data Format	

	JSON
Description	automated data flow

Output	**Self-Healing**
Destination Module	Self-Healing
Data Format	JSON
Description	automated control flow

Table 16 - Visualisation Interface Specification

The Visualisation component displayed in Table 16 combines all the insights and data generated and analysed by the other components and visualises them for the end user. As mentioned in the chapter before, this component combines two sub processes; the User Interface and the Data Visualisation. The first will most likely be divided based on the end users' rights, offering different features to the Soft Systems Methodology analyst, the IT department of the LPA and an administrative or managerial employee. The Data Visualisation sub process will be responsible for providing useful, visually appealing and user-friendly data representations and actionable recommendations for mitigation and attack prevention purposes.

3.1.7 Self-Healing Interface Specification

Input	**System Dependency Analysis**
Source Module	System Dependency Analysis
Data Format	JSON
Description	automated control flow

Input	**Visualisation**
Source Module	Visualisation
Data Format	JSON
Description	automated control flow

Input	**Self-Healing Policies**
Source Module	Visualisation
Data Format	JSON
Description	Extend the database of self-healing rules with new rules regarding threats that don't have a respective mitigation rule. automated control flow **Functionalities:** F8.4 SelfHealingPolicies

Input	**Data Analysis and Pattern Recognition**
Source Module	Data Analysis and Pattern Recognition
Data Format	JSON
Description	automated data flow

Process	**Decision Engine**
Module/s A	Input Data Analysis and Pattern Recognition Input Self-Healing Policies Input System Dependency Analysis Input Visualisation
Module/s B	Process Security Rules Composer
Process	Process Decision Engine
Definition	Identify and compose at least one proper mitigation rule for the detected threat. Apply the mitigation automatically on the affected system. **Functionalities**: F8.1 IdentifyMitigationRule F8.3 ApplyMitigation
Data Input Format	JSON
Data Output Format	JSON

Process	**Security Rules Composer**
Module/s A	Process Decision Engine
Module/s B	Output LPA System Specific Sources Output Visualisation
Process	ComposeMitigation
Definition	Compose the identified mitigation/s rule/s in a human-readable or machine-readable format. **Functionalities:** F8.2 ComposeMitigation
Data Input Format	JSON
Data Output Format	JSON

Output	**Visualisation**
Destination Module	Visualisation
Data Format	JSON, any required technical language
Description	automated data flow

Output	**LPA System Specific Sources**
Destination Module	LPA System Specific Sources
Data Format	any required technical language

Description	automated control flow

Table 17 - Self-Healing Interface Specification

The Self-Healing component shown in Table 17 receives insights from the Data Analysis module and guidelines on what to do with which type of situation from the System Dependency Analysis component. The sub process triggered by the received data, the Decision Engine, compares the new data with the listed Self-Healing Policies and in case of a match triggers the Security Rules Composer process. This process builds a directly implementable rule in a technical language used by the LPA, which can be instantly used by the IT department. This rule is then shown to the end user via the Visualisation component and might, at some point, allow for user input to trigger the automatic implementation of said rule in the LPAs system via the Self-Healing module.

3.1.8 Information Sharing Interface Specification

Input	**Data Analysis and Pattern Recognition**
Source Module	Data Analysis and Pattern Recognition
Data Format	JSON
Description	automated data flow

Input	**Visualisation**
Source Module	Visualisation
Data Format	JSON
Description	automated control flow authorization of sharing of information

Input	**System Dependency Analysis**
Source Module	System Dependency Analysis
Data Format	JSON
Description	automated control flow Information on selected Information sharing communities

Process	**Information Sharing**
Module/s A	Input Data Analysis and Pattern Recognition Input System Dependency Analysis Input Visualisation
Module/s B	Output Information Sharing Community Output Visualisation
Process	Information Sharing
Definition	This subcomponent is responsible for interacting with Visualisation for authorisation of sharing cybersecurity information and sharing relevant information with relevant threat intelligence communities. **Functionalities:** F7.1 Authorisation F7.2 CTI_Sharing

Data Input Format	JSON
Data Output Format	JSON

Output	**Visualisation**
Destination Module	Visualisation
Data Format	JSON
Description	automated data flow

Output	**Information Sharing Community**
Destination Module	Information Sharing Community
Data Format	JSON
Description	automated or manual data flow

Table 18 - Information Sharing Interface Specification

The Information Sharing module described in Table 18 is responsible for forwarding detected threats to handpicked Information Sharing communities. These might be national NIS competent authorities such as CERTs/CSIRTs or individually selected Cyber Intelligence Sharing communities. Naturally, such a distribution of information would only occur with direct authorization from the LPA and after anonymization in accordance to the guidelines defined in D7.3 and D7.4

3.2 Data Formats and Transformations

Due to the large number of different sources the Data Pre-Processing component will be required to transform any incoming data format to JSON. The JavaScript Object Notation, JSON, is a standardized format for data-interchange and is state of the art in information system technology (Introducing JSON, 2017). One of the many benefits is that it is not only machine-readable but also humans can easily comprehend the structure and the transferred information. JSON is currently already the most commonly used notation for data exchange and the preferred data format of most CS-AWARE technical partners.

For the distribution of detected cyber incidents, the STIX schema and TAXII protocol will be used. STIX stands for Structured Threat Information Expression and is a JSON schema of rapidly growing importance for cyber intelligence threat description purposes (OASIS Open, 2017). It allows institutions to share information on detected threats more easily, due to a standardized format. STIX is structured using twelve Domain Objects:

Attack Pattern, Campaign, Course of Action, Identity, Indicator, Intrusion Set, Malware, Observed Data, Report, Threat Actor, Tool and Vulnerability

as well as two Relationship Objects:

Relationship, Sighting.

Objects and relationships can now describe any cyber threat identified by an observer in JSON format, which can be easily shared with other institutions using the TAXII standard. TAXII, Trusted Automated eXchange of Indicator Information, is an "*open transport mechanism that standardizes the automated exchange of cyber threat information*" (MITRE Corporation, 2017).

Any information generated by a CS-AWARE instance that leaves its originating location either to provide information to other CS-AWARE instances or a NIS national authority will be using the STIX and TAXII protocols. CS-AWARE ensures that all hardware and software used for any of the

components are state of the art. Communication between each of the modules will be via SFTP/SHTTP and via SSL encryption, therefore ensuring data security in transport to the CS-AWARE servers.

4 API Documentation

This Section describes in detail each of the interfaces of the components. They are all implemented using a REST API and have endpoints for specific purposes. Each relationship between components is covered in a the sequence diagram, detailing the specific steps during the communication process. In addition, for each endpoint the description and an exemplary request and response has been given. These examples have been taken from the use case implemented as a demonstrator, the vulnerability use case. In the cases where a component might not yet have been included in this process, mock data has been used to outline a possible response and/or request to an endpoint.

4.1 System Dependency Analysis

To ensure the communication of the System Dependency Analysis component with its partners, two endpoints have been defined.

Endpoint GW1

https://gwiki.cs-aware.eu/api/mapping/list

Description

GET

Lists all available graphs in the wiki

Sample Request

https://gwiki.cs-aware.eu/api/mapping/list

Sample Response

```
{"Global - Rome","Global - Larissa"}
```

CodeSnippet 1: GW1 Sample Response

Endpoint GW2

https://gwiki.cs-aware.eu/api/mapping/list

Description

GET

Lists all available graphs in the wiki

Sample Request

https://gwiki.cs-aware.eu/api/mapping/list

Sample Response

```
{
"name": "Global - Rome",
"description": "Regional/International/Global",
"resources": [
{
"_id": "5c936589fa8ad39c8d682a79",
"href": "https://gwiki.cs-aware.eu/api/asset/5c936589fa8ad39c8d682a79",
"name": "Self Service",
"description": "",
"connected_to": [],
"tags": ["CategoryMilan", "CategoryAll"]
},
{
"_id": "5c936589fa8ad3ac13682a76",
"href": "https://gwiki.cs-aware.eu/api/asset/5c936589fa8ad3ac13682a76",
"name": "Access-Manager",
"description": "",
"connected_to": [
"5c936589fa8ad39c8d682a79",
"5c936589fa8ad3b99b682a77",
"5c936589fa8ad35104682a78",
"5c936589fa8ad373ee682a7a",
"5c936589fa8ad3eb6a682a82"
],
"tags": ["CategoryLogin", "CategoryReg", "CategoryMilan", "CategoryAll"]
},
{
"_id": "5c936589fa8ad3b99b682a77",
"href": "https://gwiki.cs-aware.eu/api/asset/5c936589fa8ad3b99b682a77",
"name": "Login",
"description": "",
"connected_to": [],
"tags": ["CategoryLogin", "CategoryMilan", "CategoryAll"]
},
```


CodeSnippet 2: GW2 Sample Response

4.2 Data Collection

To ensure the communication of the Data Collection component with its partners, following endpoint has been defined. Diagram 5 shows the data flow between the Data Collection client and the various sources it collects data from, as well as the Data Analysis component that uses the collected data for further investigation.

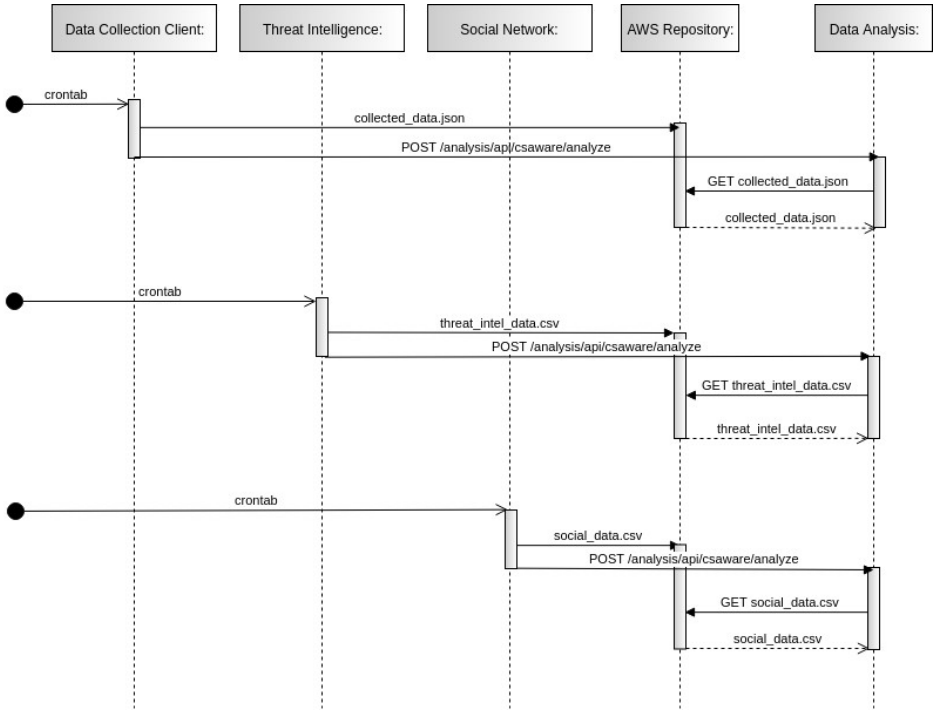


Diagram 5: Data Collection Sequence Diagram

Endpoint DC

http://<host>:8080/analysis/api/csaware/analyse

Description

POST

Included in the body is a JSON containing all information necessary to retrieve the uploaded file.

Sample Request

http://<host>:8080/analysis/api/csaware/analyse

Sample Response

```
{ "service": "s3", "bucket": "cs-aware-data-collection", "path": "2019/04/10/DEMO/16_18_51_DEMO_packages.txt" }
```

CodeSnippet 3: DC Sample Response

4.3 Data Pre-Processing

Filtering and format adaptations are mostly done in each individual component, to ensure the result is usable for further analysis. Natural language processing is a highly complex procedure and an individual subprocess, therefore requiring a dedicated endpoint. Diagram 6 shows the communication between Visualisation and the NLP information extraction tool Graphene.

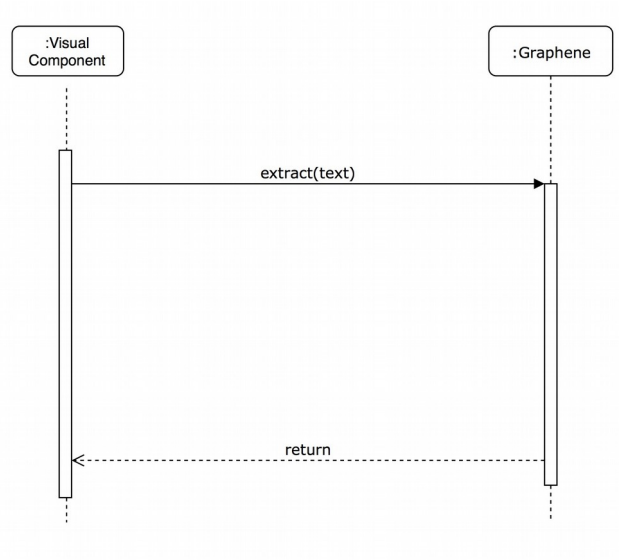


Diagram 6: NLP Information Extraction Sequence Diagram

Endpoint DPP

https://<host>:8173/extract/

Description

POST

This service extracts knowledge graphs from texts (n-ary relations and rhetorical structures extracted from complex factoid discourse). Given a sentence or a text, it outputs a semantic representation of the text which is a labelled directed graph (a knowledge graph).

text - the text from which knowledge will be extracted, JSON payload

Sample Request

https://<host>:8173/extract/

Body:

```
{ "text": "Metamorphic computer virus written in assembly language for Microsoft Windows" }
```

CodeSnippet 4: DPP Sample Request

Sample Response


```
{"name":"metamorphic computer virus written in assembly language for microsoft windows","core":
{"pureTerm":"virus","effectiveTerm":"virus","tag":"NN","specTypes":["spec","spec","written in-VBN"],
"specContents":[{"pureTerm":"computer","effectiveTerm":"computer","tag":"NN","specTypes":null,
"specContents":null,"planSpecs":[]},"pureTerm":"metamorphic","effectiveTerm":"metamorphic","tag":"JJ",
"specTypes":null,"specContents":null,"planSpecs":[]},{"pureTerm":"language","effectiveTerm":"language","tag":
"NN","specTypes":["spec","for-IN"],"specContents":[{"pureTerm":"assembly","effectiveTerm":"assembly","tag":"NN",
"specTypes":null,"specContents":null,"planSpecs":[]},{"pureTerm":"windows","effectiveTerm":"windows","tag":"NNS",
"specTypes":["spec"],"specContents":[{"pureTerm":"microsoft","effectiveTerm":"microsoft","tag":"JJ","specTypes":null,
"specContents":null,"planSpecs":[]}],"planSpecs":[["microsoft","spec"]]}],"planSpecs":[["assembly","spec"],
["microsoft","spec"],["windows","for-IN"]]}],"planSpecs":[["metamorphic","spec"],["computer","spec"],
["assembly","spec"],["microsoft","spec"],["windows","for-IN"],["language","written in-VBN"]]},"cores":["virus"]}
```


CodeSnippet 5: DPP Sample Response

4.4 Multi-Language Support

To ensure the communication of the Multi-Language Support component with its partners, two endpoints have been defined. Diagram 7 shows the communication between Visualisation and the dynamic translating unit in the Multi-Language Support component, while Diagram 8 outlines the communication for the static translations required by Visualisation.

Endpoint MLS1

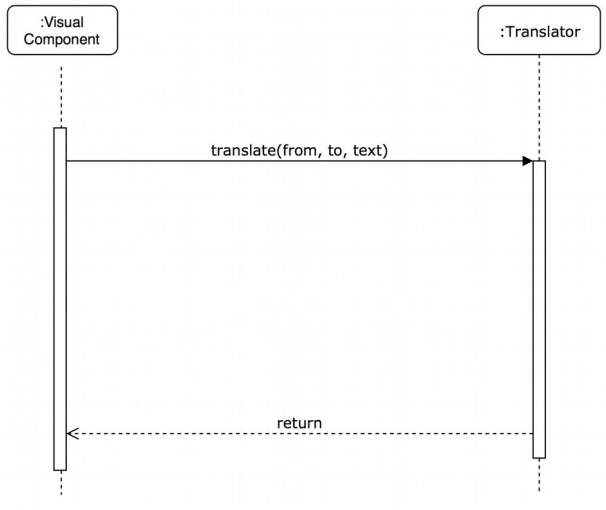


Diagram 7: Multi-Language Support Dynamic Translations Sequence Diagram

http://<host>:8080/translate/{from}/{to}

Description

POST

text - the text to be translated

{from} - the language of the given text represented by its two-letter ISO 639-1 code (e.g. en for English)

{to} - the target translation language, also represented by its ISO 639-1 code.

This service translates instructions and alerts, such as threat descriptions and self-healing suggestions, from English to the user's mother tongue (Italian or Greek).

Sample Request

http://<host>:8080/translate/en/it

Body:

```
{ "text": "Denial of service" }
```

CodeSnippet 6: MLS1 Sample Request

Sample Response

```
{ "translation": "Negazione del servizio" }
```

CodeSnippet 7: MLS1 Sample Response

Endpoint MLS2

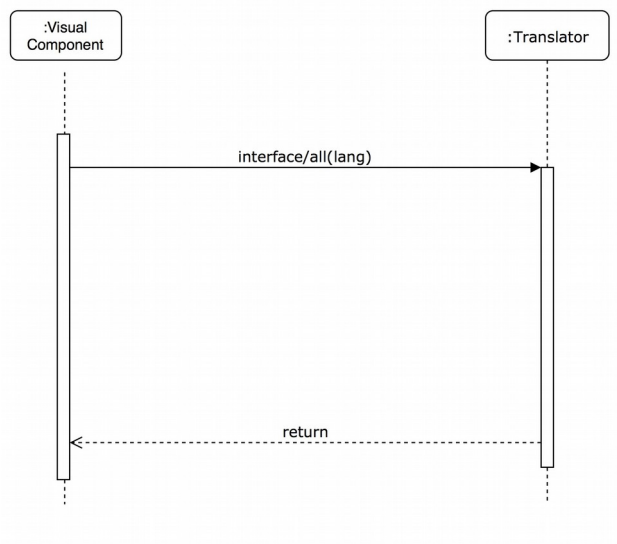


Diagram 8: Multi-Language Support Static Translations Sequence Diagram

http://<host>:8080/interface/all/{lang}

Description

GET

{lang} - The language in which the messages should be.

Sample Request

http://<host>:8080/interface/all/en

Sample Response

```
{ "labels": ["Options", "Attacks", "Self Healing", ..., "Analytics"] }
```

CodeSnippet 8: MLS2 Sample Response

4.5 Data Analysis

The sequences of the relationships of the Data Analysis component are covered by the Data Collection (Diagram 5, p. 35) and the Visualisation (Diagram 9, p. 40) component (DC, VIS1).

Endpoint DA

http://<host>:8080/analysis/api/csaware/analyse

Description

GET

With a valid STIX2 JSON document with an ObservedData object.

Sample Request

http://<host>:8080/analysis/api/csaware/analyse

Sample Response

```
{
  "first_observed": "2011-08-01T10:30:21.000Z",
  "last_observed": "2012-08-01T10:30:21.000Z",
  "number_observed": 1,
  "objects": {
    "0": {
      "type": "software",
      "name": "Word",
      "cpe": "cpe:2.3:a:microsoft:word:2000:*:*:*:*:*:*:*",
      "version": "2002",
      "vendor": "Microsoft"
    }
  },
  "created_by_ref": "identity--311b2d2d-f010-4473-83ec-1edf84858f4c",
  "created": "2016-05-12T08:17:27.000Z",
  "modified": "2017-03-20T01:27:32.000Z",
  "id": "observed-data--b67d30ff-02ac-498a-92f9-32f845f448cf",
  "type": "observed-data"
}
```

CodeSnippet 9: DA Sample Response

4.6 Data Visualisation

To ensure the communication of the Visualisation component with its partners, four endpoints have been defined. They are described in detail in this Subsection. The endpoint IS2, relating to the POST to Information Sharing with the data on approval or denial of the admin, is yet to be specified in more detail. Diagram 9 shows the communication of Visualisation with Data Analysis, Diagram 10 of Visualisation and Self-Healing, Diagram 11 of Visualisation and Information Sharing.

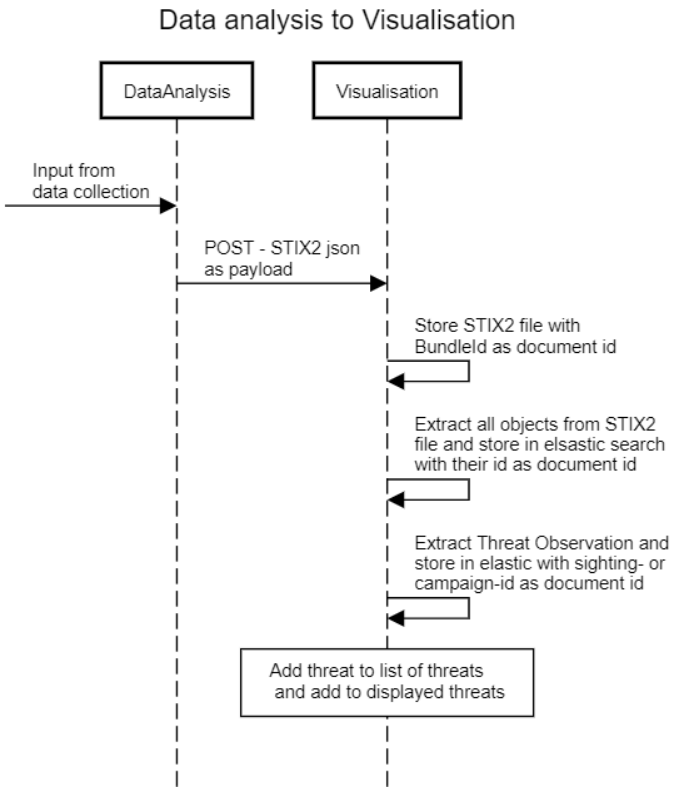


Diagram 9: Visualisation and Data Analysis Sequence Diagram

Endpoint VIS1

http://<host>:5679/bundle/consume

Description

POST

Visualisation expects to receive a STIX2 bundle in JSON format from Data Analysis with a either a Sighting or a Campaign, defining the actual threat.

After receiving the STIX2 bundle Visualisation will:

1) store the received bundle in Elastic Search with the bundle-id as document id.

2) store STIX2 Objects from Bundle - all the objects of the bundle are stored using their STIX object id as document id.

3) extract information for ThreatObservation and update/store in ElasticSearch. The "sighting" or "campaign" id is used as document id.

Information from the bundle is "denomalized" to produce a ThreatObservation holding "all" information relevant for the System Administrator.

Sample Request

http://<host>:5679/bundle/consume

Sample Response

```
{
  "type": "bundle",
  "id": "bundle--a836f05a-f235-4b4b-b523-bd87e40478a1",
  "spec_version": "2.0",
  "objects": [
    {
      "type": "identity",
      "id": "identity--023d105b-752e-4e3c-941c-7d3f3cb15e9e",
      "name": "larissa_LPA",
      "identity_class": "organization",
      "labels": ["local","public","administration"],
      "created": "2016-08-23T18:05:49.307Z",
      "modified": "2016-08-23T18:05:49.307Z"
    },
    {
      "type": "vulnerability",
      "id": "vulnerability--0c7b5b88-8ff7-4a4d-aa9d-feb398cd0061",
      "name": "jqueryui",
      "description": "Cross-site scripting (XSS) vulnerability in the default content
option in jquery.ui.tooltip.js in the Tooltip widget in jQuery UI before 1.10.0 allows remote
attackers to inject arbitrary web script or HTML via the title attribute, which is not
properly handled in the autocomplete combo box demo.",
      "x_da_threat_severity": 4,
      "x_da_threat_group": "WEB threat",
      "x_da_risk_level": 90,
      "x_da_exploitability_level": 50,
      "created": "2016-08-23T18:05:49.307Z",
      "modified": "2016-08-23T18:05:49.307Z",
      "external_references": [{"source_name": "cve","external_id": "CVE-2012-6662"}
      ]
    },
    {
      "type": "malware",
      "id": "malware--fdd60b30-b67c-11e3-b0b9-f01faf20d111",
      "created": "2014-02-20T09:16:08.989Z",
      "modified": "2014-02-20T09:16:08.989Z",
      "name": "Poison Ivy",
      "x_da_threat_severity": 3,
      "x_da_threat_group": "WEB threat",
      "x_da_risk_level": 90,
      "x_da_exploitability_level": 50,
      "labels": ["remote-access-trojan"]
    },
```


CodeSnippet 10: VIS1 Sample Response

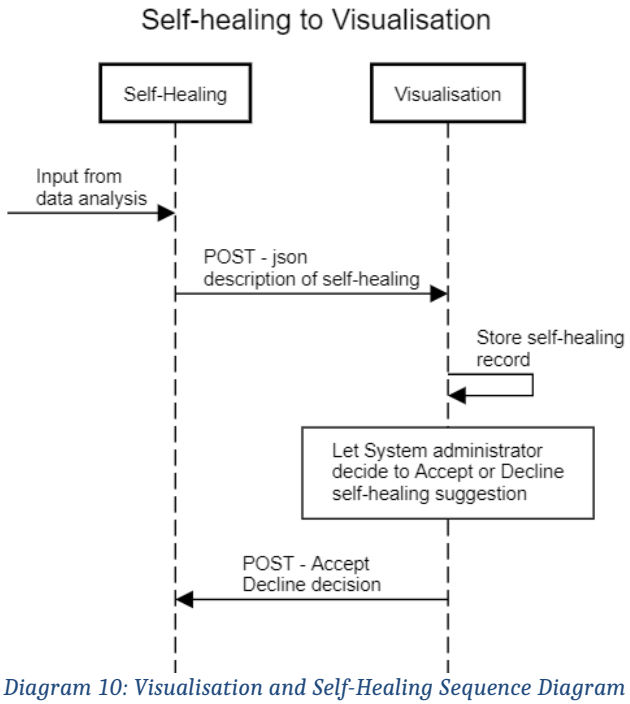


Diagram 10: Visualisation and Self-Healing Sequence Diagram

Endpoint VIS2

http://<host>:5679/selfhealing/update

Description

Visualisation expects to receive information is JSON format.

State can be one of the following:

confirm - Ask for a system administrator confirmation (may be an automatic confirm, if this is configured (not supported yet))

progress - Report that the self healing is on the way.

Healed - Report that the self healing is done.

Failed - Report that the self healing was not possible.

Sample Request

http://<host>:5679/selfhealing/update

Sample Response

```
{
   "id": "sighting--779c4ae8-e134-4180-baa4-03141095d971",
   "state": "confirm",
   "course_of_action": {
      "id": "course-of-action--8e2e2d2b-17d4-4cbf-938f-98ee46b3cd3f",
      "name": "jqueryui_mitigation",
      "description": "Download and apply the update packages found in the following link: https://access.redhat.com/errata/RHSA-2015:1462"
   }
}
```

CodeSnippet 11: VIS2 Sample Response

Endpoint VIS3

http://<self-healing_api:6000>/vis_state_input

Description

When the user (system administrator) has decided what to respond to the received Self-Healing message, a message is returned to Self-Healing sending a POST to this Endpoint.

where

"id" is the id of the threat to be healed

"state" is either confirm or decline

Sample Request

http://<self-healing_api:6000>/vis_state_input

Sample Response

```
{
   "id": "sighting--8356e820-8080-4692-aa91-ecbe94006833",
   "state": "confirm",
   "course_of_action": {
      "id": "course-of-action--8e2e2d2b-17d4-4cbf-938f-98ee46b3cd3f",
      "name": "CouchDB upgrade to 2.2.3",
      "description": "apt-get dist-upgrade -y couchdb"
   }
}
```

CodeSnippet 12: VIS3 Sample Response

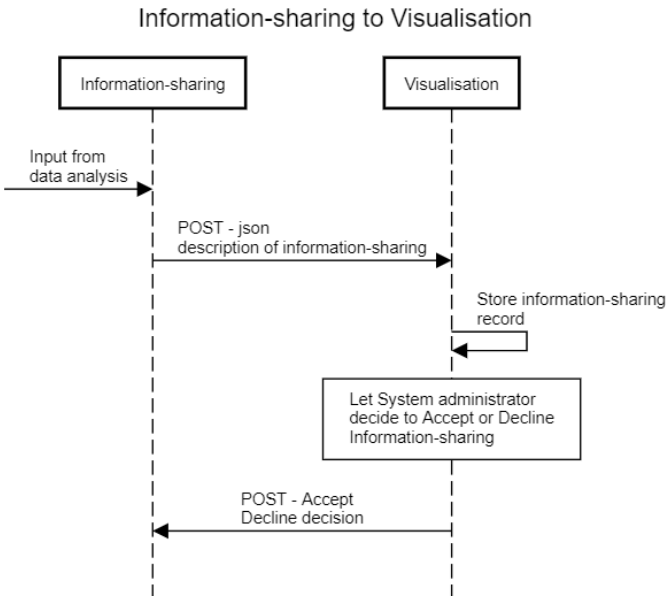


Diagram 11: Visualisation and Information Sharing Sequence Diagram

Endpoint VIS4

http://<host>:5679/informationshare/decision/update

Description

Visualisation can receive information from information sharing in a POST to this endpoint. Visualisation expects to receive information is JSON format.

where

"id" is the id is the id of the sighting that is to be shared.

"state" is confirm signifying that Information sharing requests the System Administrator to confirm if the information can be shared or not.

Sample Request

http://<host>:5679/informationshare/decision/update

Sample Response

```
{
   "id": "sighting--779c4ae8-e134-4180-baa4-03141095d971",
   "state": "confirm",
}
```

CodeSnippet 13: VIS4 Sample Response

4.7 Information Sharing

To ensure the communication of the Information Sharing component with its partners, two endpoints have been defined. Diagram 12 clearly outlines the sequence of communication between Information Sharing and their partner components.

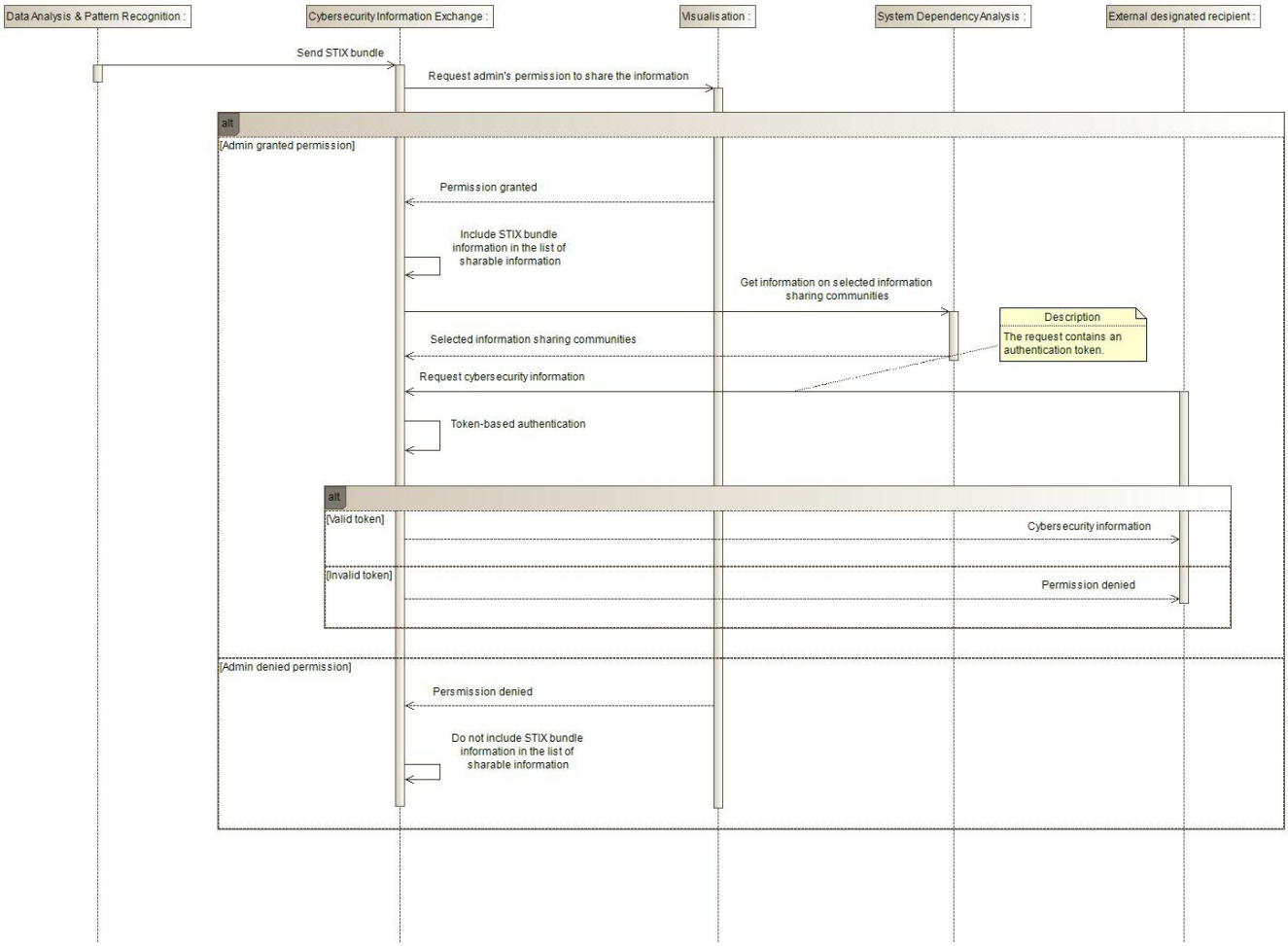


Diagram 12: Information Sharing Sequence Diagram

Endpoint IS1

http://<cie-api>:5000/ctie_input

Description

POST

Cybersecurity Information Exchange expects to receive a POST Request from Data Analysis which contains information regarding the detected threat within an LPA's system.

Sample Request

http://<cie-api>:5000/ctie_input

Sample Response


```
{
"type": "bundle",
"id": "bundle--a836f05a-f235-4b4b-b523-bd87e40478a1",
"spec_version": "2.0",
"objects": [
{
"type": "identity",
"id": "identity--023d105b-752e-4e3c-941c-7d3f3cb15e9e",
"name": "larissa_LPA",
"identity_class": "organization",
"labels": ["local","public","administration"],
"created": "2016-08-23T18:05:49.307Z",
"modified": "2016-08-23T18:05:49.307Z"
},
{
"type": "vulnerability",
"id": "vulnerability--0c7b5b88-8ff7-4a4d-aa9d-feb398cd0061",
"name": "jqueryui",
"description": "Cross-site scripting (XSS) vulnerability in the default content option in
jquery.ui.tooltip.js in the Tooltip widget in jQuery UI before 1.10.0 allows remote attackers to inject
arbitrary web script or HTML via the title attribute, which is not properly handled in the autocomplete combo box demo.",
"x_da_threat_severity": 4,
"x_da_threat_group": "WEB threat",
"x_da_risk_level": 90,
"x_da_exploitability_level": 50,
"created": "2016-08-23T18:05:49.307Z",
"modified": "2016-08-23T18:05:49.307Z",
"external_references": [{"source_name": "cve","external_id": "CVE-2012-6662"}
]
},
{
"type": "malware",
"id": "malware--fdd60b30-b67c-11e3-b0b9-f01faf20d111",
"created": "2014-02-20T09:16:08.989Z",
"modified": "2014-02-20T09:16:08.989Z",
"name": "Poison Ivy",
"x_da_threat_severity": 3,
"x_da_threat_group": "WEB threat",
"x_da_risk_level": 90,
"x_da_exploitability_level": 50,
"labels": ["remote-access-trojan"]
},
```


CodeSnippet 14: IS1 Sample Response

Endpoint IS2

TBD

Description

POST

In order to share the cybersecurity information, Cybersecurity Information Exchange ask for the administrator's permission via a POST Request to the Visualisation component.

An approach would be that the POST Request is expected to contain the ID of a bundle which references the sanitised and anonymised information regarding a detected threat in an LPA's system. All classified information related to the affected LPA's system will be cleared. Thus, the sanitised bundle will contain only the information that is relevant to the detected threat and is considered useful for the Information Sharing communities. The Visualisation component will be able to retrieve the sanitised bundle using the provided ID and display to the user the information that is going to be shared. Therefore, the user is aware of the exact information that is going to be shared with external Information Sharing communities.

The structure of the POST Request has not been defined yet.

4.8 Self-Healing

To ensure the communication of the Self-Healing component with its partners, three endpoints have been defined. They are described in detail in this Subsection and the relevant communication sequence is outlined in Diagram 13.

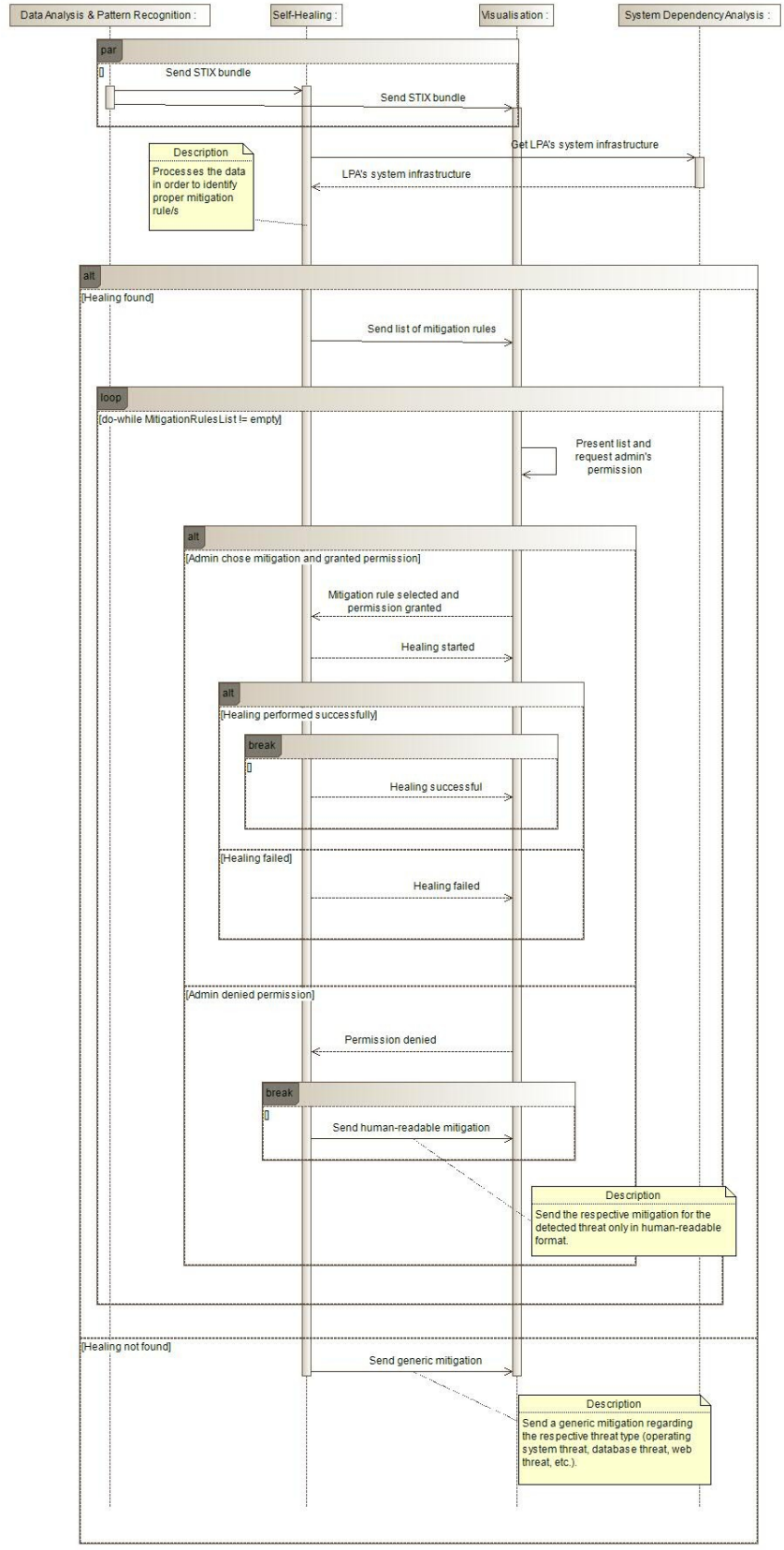


Diagram 13: Self-Healing Sequence Diagram

Endpoint SH1

http://<host>:5679/selfhealing/update

Description

POST

Self-Healing sends the identified list of mitigations to the Visualisation component via a POST Request in order to ask for the administrator's permissions to apply the mitigation rule automatically on the affected system.

State can be one of the following:

confirm: Ask for a system administrator confirmation (may be an automatic confirm, if this is configured (not supported yet)).
progress: Report that the healing process is still active.
healed: Report that the healing process has successfully performed.
failed: Report that the healing process has failed.

Sample Request

http://<host>:5679/selfhealing/update

Sample Response


```
{
   "id": "sighting--779c4ae8-e134-4180-baa4-03141095d971",
   "state": "confirm",
   "course_of_action": {
      "id": "course-of-action--8e2e2d2b-17d4-4cbf-938f-98ee46b3cd3f",
      "name": "jqueryui_mitigation",
      "description": "Download and apply the update packages found in the following link: https://access.redhat.com/errata/RHSA-2015:1462"
   }
}
```


CodeSnippet 15: SH1 Sample Response

Endpoint SH2

http://<self-healing_api>:6000/vis_state_input

Description

POST

When the administrator has decided what to respond to the received Self-Healing message, a message is returned to Self-Healing via a POST Request.

Where:

"id"is the ID of the threat to be healed.

"state" is either confirm or decline.

Sample Request

http://<self-healing_api>:6000/vis_state_input

Sample Response


```
{
   "id": "sighting--8356e820-8080-4692-aa91-ecbe94006833",
   "state": "confirm",
   "course_of_action": {
      "id": "course-of-action--8e2e2d2b-17d4-4cbf-938f-98ee46b3cd3f",
      "name": "CouchDB upgrade to 2.2.3",
      "description": "apt-get dist-upgrade -y couchdb"
   }
}
```


CodeSnippet 16: SH2 Sample Response

Endpoint SH3

http://<self-healing_api>:6000/vis_state_input

Description

POST

Cybersecurity Information Exchange expects to receive a POST Request from Data Analysis which contains information regarding the detected threat within an LPA's system.

Sample Request

http://<self-healing_api>:6000/vis_state_input

Sample Response


```
{
"type": "bundle",
"id": "bundle--a836f05a-f235-4b4b-b523-bd87e40478a1",
"spec_version": "2.0",
"objects": [
{
"type": "identity",
"id": "identity--023d105b-752e-4e3c-941c-7d3f3cb15e9e",
"name": "larissa_LPA",
"identity_class": "organization",
"labels": ["local","public","administration"],
"created": "2016-08-23T18:05:49.307Z",
"modified": "2016-08-23T18:05:49.307Z"
},
{
"type": "vulnerability",
"id": "vulnerability--0c7b5b88-8ff7-4a4d-aa9d-feb398cd0061",
"name": "jqueryui",
"description": "Cross-site scripting (XSS) vulnerability in the default content option in
jquery.ui.tooltip.js in the Tooltip widget in jQuery UI before 1.10.0 allows remote attackers to inject
arbitrary web script or HTML via the title attribute, which is not properly handled in the autocomplete combo box demo.",
"x_da_threat_severity": 4,
"x_da_threat_group": "WEB threat",
"x_da_risk_level": 90,
"x_da_exploitability_level": 50,
"created": "2016-08-23T18:05:49.307Z",
"modified": "2016-08-23T18:05:49.307Z",
"external_references": [{"source_name": "cve","external_id": "CVE-2012-6662"}
]
},
{
"type": "malware",
"id": "malware--fdd60b30-b67c-11e3-b0b9-f01faf20d111",
"created": "2014-02-20T09:16:08.989Z",
"modified": "2014-02-20T09:16:08.989Z",
"name": "Poison Ivy",
"x_da_threat_severity": 3,
"x_da_threat_group": "WEB threat",
"x_da_risk_level": 90,
"x_da_exploitability_level": 50,
"labels": ["remote-access-trojan"]
},
```


CodeSnippet 17: SH3 Sample Response

5 References

Bansal, S. K., & Kagemann, S. (2015). Integrating big data: A semantic extract-transform-load framework. *Computer, 48*(3), 42-50.

BSI. (2016, January). *IT-Grundschutz*. Retrieved May 28, 2017, from Bundesamt für Sicherheit in der Informationstechnik: https://www.bsi.bund.de/DE/Themen/ITGrundschutz/ITGrundschutzKataloge/itgrundschutzkataloge_node.html

Checkland, P., & Soles, J. (1990). *Soft Systems Methodology in Action.* Chichester: John Wiley & Sons Ltd.

CINI Cyber Security National Laboratory. (2016). *Italian Cyber Security Report 2015 – A National Cyber Security Framework.*

Council, E. P. (2016). Directive (EU) 2016/1148 of the European Parliament and of the Council. *Official Journal of the European Union*, 1-30.

ENISA. (2014). *EP3R 2010-2013 - Four Years of Pan-European Public Private Cooperation.* Athens: ENISA.

Federal Office for Information Security. (2017). *The BSI*. Retrieved January 2, 2018, from https://www.bsi.bund.de/EN/TheBSI/thebsi_node.html

General Data Protection Regulation. (2016). Regulation, General Data Protection. "Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46. *Official Journal of the European Union, 59*, 1-88.

Introducing JSON. (2017). Retrieved January 29, 2018, from json.org: http://json.org

MITRE Corporation. (2017). *TAXII Project*. Retrieved January 3, 2018, from http://taxiiproject.github.io/about/

NIST National Institute of Standards and Technology. (2015). *Framework for Improving Critical Infrastructure Cybersecurity.* NIST.

OASIS Open. (2017). *STIX*. Retrieved January 3, 2018, from cti-documentation: https://oasis-open.github.io/cti-documentation/

Zimmermann, H. (1980). OSI reference model -- The ISO model of architecture for open systems interconnection. *IEEE Transactions on communications*, 425-432.