D7.4

Final Reference Architecture

WP7 – C3ISP platform: Requirements / Architecture / Implementation and integration

C3ISP

Collaborative and Confidential Information Sharing and Analysis for Cyber Protection

Due date of deliverable: 31/07/2019
Actual submission date: 31/07/2019

31/07/2019

Version 1.0

Responsible partner: HPE
Editor: Mirko Manea
E-mail address: mirko.manea at hpe.com

Project co-funded by the European Commission within the Horizon 2020 Framework Programme		
Dissemination Level		
PU	Public	**X**
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

The C3ISP Project is supported by funding under the Horizon 2020 Framework Program of the European Commission DS 2015-1, GA #700294

Authors: M. Manea, P. Ciampoli (HPE), T. Nguyen, V. Herbert (CEA), I. Herwono (BT), R. de Lemos, D. Chadwick (UNIKENT), F. Di Cerbo, J. Boehler (SAP), P. Mori, A. Saracino, G. Costantino, I. Matteucci (CNR), J. Dobos, C. Wong (3DREPO), R. Ben Hamouda (GPS)

Approved by: BT, 3DREPO

Revision History

Version	Date	Name	Partner	Sections Affected / Comments
0.1	29-Apr-2019	M. Manea	HPE	Initial ToC
0.2	2-May-2019	M. Manea	HPE	Appointed BT and 3DREPO as internal reviewers
0.3	17-Jun-2019	M. Manea, P. Ciampoli	HPE	Added HPE contribution
0.4	27-Jun-2019	A. Arighi, I. Herwono, R. Ben Hamouda, W. Fan, D. Chadwick	CHINO, BT, GPS, UNIKENT	Merged contribution from partners
0.5	28-Jun-2019	F. Di Cerbo	SAP	Added SAP contribution
0.6	30-Jun-2019	C. Wong, G. Costantino, A. Saracino, I. Matteucci, P. Mori	3DREPO, CNR	Added 3DREPO contribution Added CNR contribution
0.7	12-Jul-2019	T. Nguyen, V. Herbert, M. Manea, P. Ciampoli	CEA, HPE	Merged CEA sections Revised all sections
0.8	19-Jul-2019	M. Manea	HPE	Ready for Internal Review
0.9	23-Jul-2019	I. Herwono, C. Wong	BT, 3DREPO	Internal review
1.0	31-Jul-2019	F. Martinelli	CNR	Final version

Executive Summary

This document describes the final reference architecture of the C3ISP Framework. First version of the architecture was released at M12 (milestone MS2, deliverable D7.2) and refined up to M24 when the first version of the C3ISP platform was implemented (milestone MS4, deliverable D7.3). In particular, D7.4 described an updated and refined C3ISP Framework which was implemented and used for evaluation by the Pilots. Confirming the sound design reported in those documents, the overall architecture has not seen major changes, but still there have been refinements in some of the most complex workflows (e.g. analytics execution and application of DMO), as well as many implementation enhancements, software upgrades and operational and performance tuning.

Table of contents

1. Introduction

1.1. Overview

This document presents the final version of the reference architecture implementation for the C3ISP Framework, as it has been originally designed in D7.2 and enhanced in D7.3. It also describes how the architecture evolved in the last period to address specific Pilots' evaluation outcomes and observations for improvements by the EC (e.g. the security assessment activities). Finally, it provides updated data flow diagrams for the most important use cases provided by the C3ISP Framework (if changed with respect to last deliverable) and updates on the development and test bed environments.

1.2. Deliverable Structure

The document is structured as follows:

- Section 2 contains the final high-level architecture based on micro-services;
- Section 3 describes how we setup the C3ISP deployment models with micro-services and containers for Local Information Sharing Infrastructure instances;
- Sections 4, 5, 6 and 7 describe respectively the Information Sharing Infrastructure, Information Analytics Infrastructure, DSA Manager, Common Security Services finalised components;
- Section 8 illustrates most significant updated workflows for the C3ISP Framework;
- Section 9 reports a view on the requirements that were not met in previous milestones and which are now addressed;
- Section 10 contains the final status of the Development, Test Bed and Production environments, including the security tests performed against the C3ISP Framework.

1.3. Definitions and Abbreviations

Term	Meaning
AES	Advanced Encryption Standard
C&C	Command and Control
C3ISP	Collaborative and Confidential Information Sharing and Analysis for Cyber Protection
CybOX	Cyber Observable eXpression
CI	Continuous Integration
CPE	Common Platform Enumeration
CSP	Cloud Service Provider
CSS	Common Security Services
CTI	Cyber Threat Information
CVE	Common Vulnerability and Exposure
CWE	Common Weakness Enumeration
DAST	Dynamic Application Security Testing
DDoS	Distributed Denial of Service

DMO	Data Manipulation Operations
DoW	Description of Work for Grant Agreement: 700294 — Collaborative and Confidential Information Sharing and Analysis for Cyber Protection (C3ISP)
DPOS	Data Protected Object Storage
DPO	Data Protected Object
DSA	Data Sharing Agreement
FHE	Full Homomorphic Encryption
GDPR	General Data Protection Regulation (EU 2016/679), http://eur-lex.europa.eu/eli/reg/2016/679/oj
IAI	Information Analytics Infrastructure
IDE	Integrated Development Environment
IDS	Intrusion Detection System
IP	Internet Protocol
ISI	Information Sharing Infrastructure
LTS	Long-Term Support
LOWMC	Low Multiplicative Complexity (a family of block ciphers)
MITRE	The MITRE Corporation, https://www.mitre.org/
NFR	Non Functional Requirement
NVD	National Vulnerability Database
OASIS	Organization for the Advancement of Structured Information Standards
OWASP	Open Web Application Security Project
OpenC2	Open Command and Control
MoSCoW	Must have, Should have, Could have, and Won't have but would like
Multiplicative depth	Multiplicative depth is the maximum number of multiplicative gates between an input and an output of the circuit
PKI	Public Key Infrastructure
PRINCE	64-bit block cipher with a 128-bit key optimized for low latency in hardware
Prosumer	An entity which is both a producer and a consumer of information, in particular of Cyber Threat Information
REST	REpresentational State Transfer, a type of web service
RFI	Remote File Inclusion attack
SaaS	Software as a Service
SQLi	SQL injection attack
STIX	Structured Threat Information eXpression

TAXII	Trusted Automated eXchange of Indicator Information
TTP	Techniques, Tactics and Procedures
VCG	VisualCodeGrepper
VM	Virtual Machine
WAVSEP	Web Application Vulnerability Scanner Evaluation Project
XSS	Cross-Site Scripting attack

2. High-Level Architecture

The high-level architecture of the C3ISP Framework had minor changes since last deliverable and is reported in Figure 1.

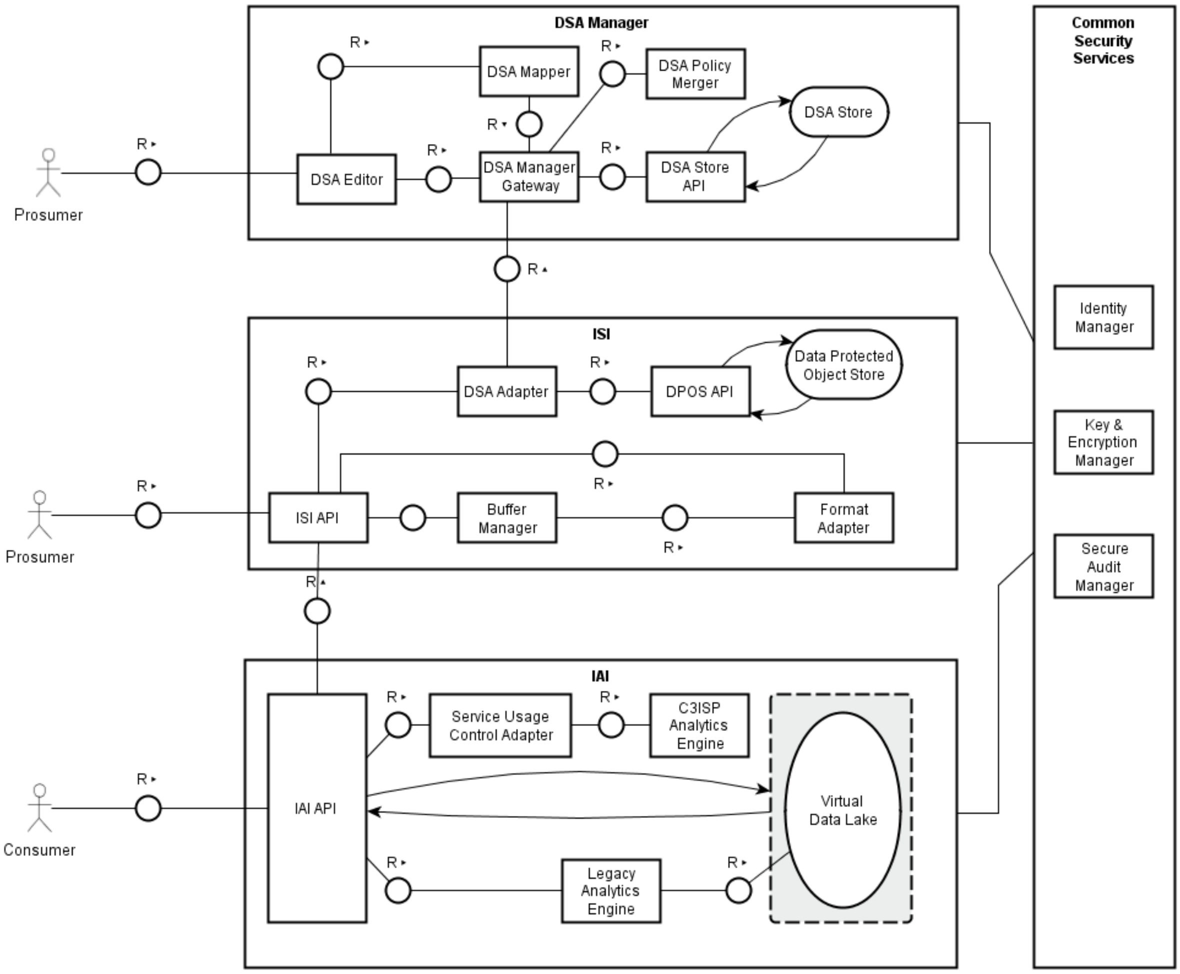


Figure 1: C3ISP high-level architecture – Final (Month 34)

We recall here that the C3ISP Framework architecture is built on 4 major subsystems, whose role has not changed since the last deliverable, and are described as follows:

- **Information Sharing Infrastructure – ISI**, devoted to providing sharing capabilities of CTI data regulated through sophisticated policies embedded in Data Sharing Agreements (DSAs);
- **Information Analytics Infrastructure – IAI**, responsible for regulating the execution of cybersecurity-based analytics services on the data shared through the ISI, hence governed by the DSAs;
- **DSA Manager**, for authoring the DSAs and handling their lifecycle, from initial writing in high-level language (CNL – Controlled Natural Language) to mapping in its enforceable representation (UPOL – Usage Policy Language[1]). With respect to D7.3,

[1] UPOL is a low level directly enforceable XACML-based policy language, developed in the Coco Cloud FP7 project (grant no. 610853)

we added the *DSA Policy Merger* component (see 6.3), necessary to create the DSA used when the C3ISP-aware analytics produce their outcome (i.e. analytics result);

- **Common Security Services – CSS**, useful to provide services that operate across the other subsystems, like handling of identities and their authentication/authorisation (also based on OIDC, see 7.1.2), encryption-related activities, and system auditing.

2.1. Micro-services architecture

The C3ISP Framework has been designed and implemented as a micro-services based architecture. Each component communicates to the others via RESTful APIs: for the sake of standardisation, APIs are defined via OpenAPI standard, which describes all the exposed methods and data structures via an open specification that allows easy creation[2] of client services that will need to interact with it.

To ease the deployment of the framework for different execution environments (like Test Bed, Production and different deployment models, including Local ISI deployment), we added to the architecture a Configuration Server. This component gives support for externalised configuration in the C3ISP distributed system, with a centralised place to manage configuration properties for the different C3ISP components across environments. This means that each software component artefact does not need to be bundled with specific environment settings (e.g. IP addresses, credentials, filesystem paths, etc.), but the component will download its configuration at boot time, depending on the environment it is running on (e.g. Test Bed vs Production) or the deployment model it has to be configured for (e.g. Local ISI vs Central ISI).

The Configuration Server is built using the Spring Cloud Config project (ref. [76]) and works as a micro-service. It has a RESTful interface queried by the C3ISP components. The Configuration Server stores the settings of all the components on a revision control system (a GIT repository), which allows keeping track of each change and labelling of configuration: this permits easy inspection and approval of changes, and performing configuration rollback if necessary. Figure 2 describes the Configuration Server setup (ConfigServer for short) for the C3ISP Fully Centralised deployment model:

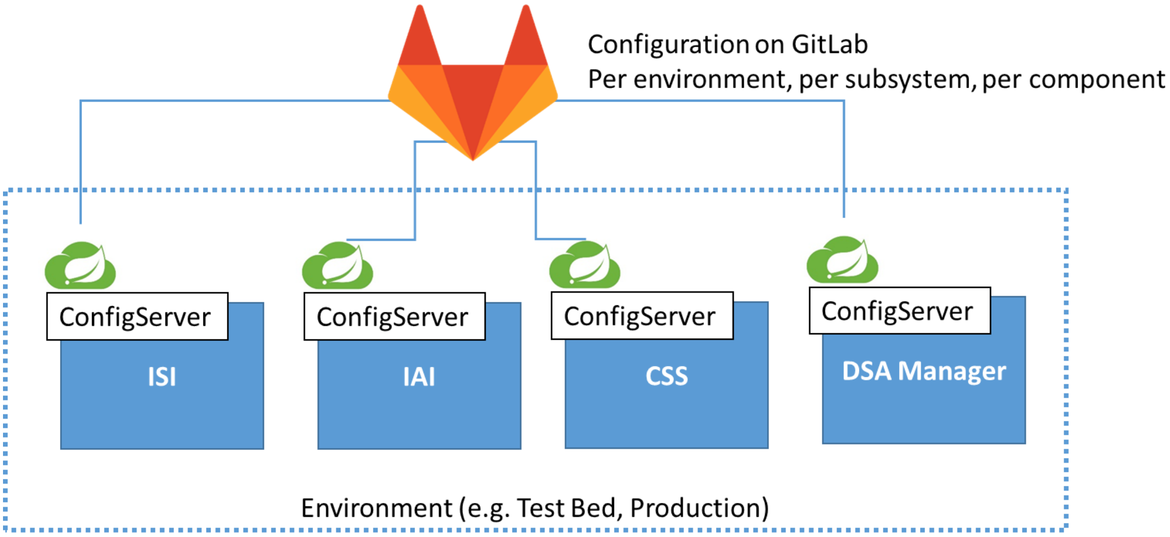


Figure 2: C3ISP Configuration Server on a Fully Centralised deployment model

[2] In fact there are also tools to automatically build source code stubs starting from the OpenAPI specification descriptor (in JSON format), see Swagger Codegen at https://github.com/swagger-api/swagger-codegen

In Figure 2, each C3ISP subsystem has its own Configuration Server instance, which points to the common centralised GIT repository. In particular, in this scenario, all the components of a specific subsystem (e.g. ISI components) is considered to run on the same computing instance (e.g. a virtual machine): when a component boots, it contacts the dedicated Configuration Server subsystem instance. This setup permits having no code or configuration changes on the component in order to run on different execution environments.

The detailed workflow is as follows:

a) The component bootstraps;

b) It discovers the Configuration Server at a well-known URI (e.g. http://localhost:8888), by authenticating to it and providing its own identity (e.g. ISI API);

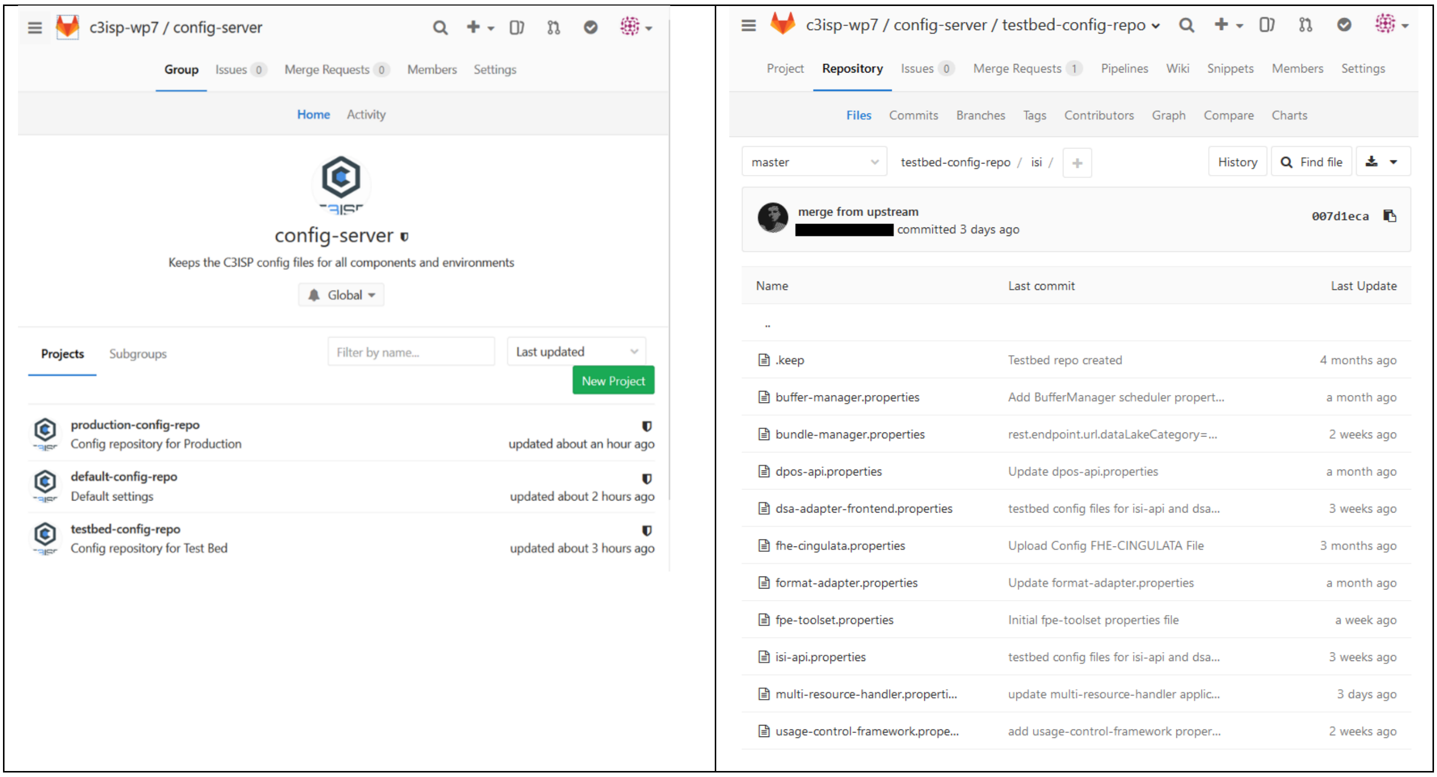
c) The Configuration Server retrieves the settings for the identified component (e.g. ISI API) from the central GIT repository;

d) The component (e.g. ISI API) configures itself with the received settings. For example, ISI API will know that the DSA Adapter Frontend it needs to communicate with is located at the provided URI Test Bed configuration; however, if ISI API is running in Production, the DSA Adapter Frontend URI will be different. The only required configuration will be a global setting defined at computing instance level (e.g. at the application server running on Test Bed or Production machines).

This allows great flexibility and supports creating standardised deployment models with almost zero-configuration needs.

In case the Configuration Server is not available for any reason, the component can fall-back to its default configuration settings, or fail entirely to start-up.

Figure 3 shows a part of the GIT structure used to host the configuration settings per environment:



Keeps the C3ISP config files for all components and environments

production-config-repo
Config repository for Production
updated about an hour ago

default-config-repo
Default settings
updated about 2 hours ago

testbed-config-repo
Config repository for Test Bed
updated about 3 hours ago

merge from upstream
[redacted] committed 3 days ago
007d1eca

Name	Last commit	Last Update
..		
.keep	Testbed repo created	4 months ago
buffer-manager.properties	Add BufferManager scheduler propert...	a month ago
bundle-manager.properties	rest.endpoint.url.dataLakeCategory=...	2 weeks ago
dpos-api.properties	Update dpos-api.properties	a month ago
dsa-adapter-frontend.properties	testbed config files for isi-api and dsa...	3 weeks ago
fhe-cingulata.properties	Upload Config FHE-CINGULATA File	3 months ago
format-adapter.properties	Update format-adapter.properties	a month ago
fpe-toolset.properties	Initial fpe-toolset properties file	a week ago
isi-api.properties	testbed config files for isi-api and dsa...	3 weeks ago
multi-resource-handler.properti...	update multi-resource-handler applic...	3 days ago
usage-control-framework.prope...	add usage-control-framework proper...	2 weeks ago

Figure 3: Configuration Server structure on internal C3ISP GIT (GitLab). On the left, the environments tree; on the right the specific components settings for the ISI subsystem on Test Bed

3. Deployment Models

In D7.2, we introduced the concept of C3ISP Deployment Models as a way to compose differently the C3ISP subsystems to match specific use case scenarios, including different trust assumptions (e.g. should I anonymise IP addresses in my local premises or do I trust central infrastructure for that?).

We identified four deployment models to support a vast range of requirements:

- **Fully centralised**: both ISI and IAI are centralised;
- **Hybrid**: ISI is both on-premises and centralised, with a centralised IAI;
- **Distributed ISI**: ISI is on-premises only and IAI is centralised;
- **Fully distributed**: both ISI and IAI are on-premises.

Among them, pilots selected both **Hybrid** and **Fully centralised** deployment models to support their use cases:

Table 1 – Deployment models in the Pilots from D7.2

	Hybrid	Fully centralised
ISP Pilot	✓	
CERT Pilot	✓	
Enterprise Pilot		✓
SME Pilot	✓	

In particular, the Hybrid deployment model allows for one or more Local ISI to run on the premises of each ISP or SMs using the C3ISP Framework. For this reason, we considered building an easily deployable **Local ISI kit**, available along with a few installation steps, in order to make its setup straightforward for users that do not have in-depth knowledge about all the inner details of the C3ISP Framework.

Leveraging on the micro-services architecture of C3ISP and the externalised configuration management provided by the Configuration Server described in Section 2.1, we setup a Local ISI instance based on the *container technology*, a fast growing lightweight virtualisation technique that is becoming very popular in recent years.

Containers provide the following benefits specifically for an easy C3ISP deployment:

1. Each micro-service is mapped to a container, i.e. a confined execution environment based on a Java application server (Apache Tomcat) that runs a single C3ISP component (the micro-service, e.g. ISI API);
2. The containers communicate between them via an internal dedicated virtual network, visible only to the containers and provided by the container software daemon, which serves also as virtual DNS resolver. In fact, the containers can talk to each other simply by using the container name: this simplifies the configuration and makes it independent from the environment (e.g. Test Bed or Production can internally use the same configuration settings with respect to addresses of the components, since we are using container's name instead of fixed IP addresses);

3. The ISI subsystem’s architecture has a single point of communication with external Prosumers, which is the ISI API. The ISI API container exposes a single HTTP port that Prosumers can use to interact with Local ISI: this minimises the attacks surface and makes Local ISI deployment more secure and less intrusive than a traditional installation without containers.

In particular, we employed Docker Community Edition (ref. [77]) as a container product. We then orchestrate the composition of all the Local ISI micro-services with Docker Compose (ref. [78]), a tool for defining and running multi-container Docker applications. Docker Compose describes the configuration settings of all the involved containers in a YAML[3] descriptor file (called *docker-compose.yml*).

Among the many features of Docker Compose, we found the following particularly convenient for our deployment strategy:

- Ability to define relationships between containers, i.e. which container has to start before the others and guarantee the correct bootstrap order;
- Persistence of data, i.e. possibility to associate data volumes to container that persists across containers restart, which for example enables Local ISI to store CTI data provided by the Prosumers.

The following graph represents the relationships between containers (e.g. format-adapter), the persistence of data in volumes (e.g. /opt/c3isp/Docker/ISI/dpos-api/data) and the shared virtual network that permits container internal communication (e.g. net: c3isp.local):

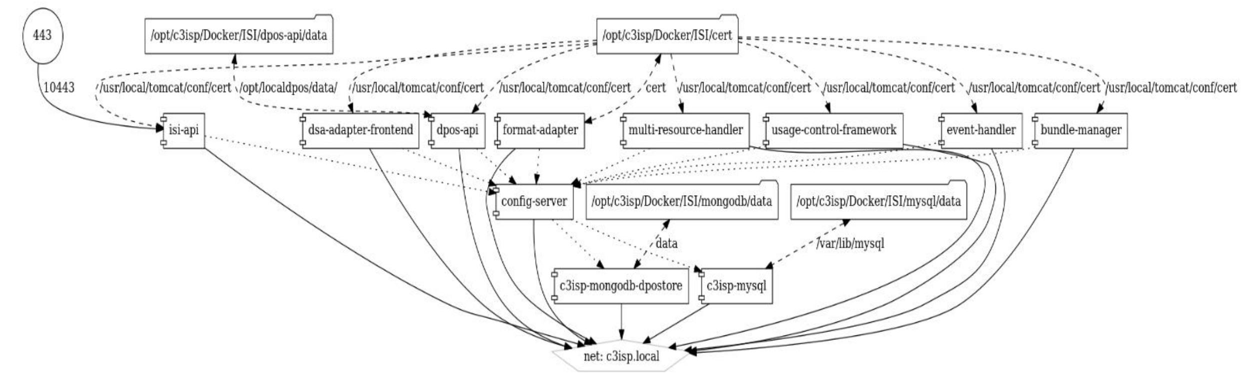

443
10443
/opt/c3isp/Docker/ISI/dpos-api/data
/opt/c3isp/Docker/ISI/cert
/usr/local/tomcat/conf/cert
/opt/localdpos/data/
/usr/local/tomcat/conf/cert
/usr/local/tomcat/conf/cert
/cert
/usr/local/tomcat/conf/cert
/usr/local/tomcat/conf/cert
/usr/local/tomcat/conf/cert
/usr/local/tomcat/conf/cert
isi-api
dsa-adapter-frontend
dpos-api
format-adapter
multi-resource-handler
usage-control-framework
event-handler
bundle-manager
config-server
/opt/c3isp/Docker/ISI/mongodb/data
/opt/c3isp/Docker/ISI/mysql/data
/data
/var/lib/mysql
c3isp-mongodb-dpostore
c3isp-mysql
net: c3isp.local

Figure 4: Docker-compose.yml graph representation[4]

Among the relationships, it is worth noting that all the C3ISP components depend on the “config-server”, which provides configuration settings. Furthermore, containers use a digital certificate for secure communication (hosted at /opt/c3isp/Docker/isi/cert). Finally, “isi-api” component exposes secure port https/443 as the entry point to the containerised Local ISI deployment.

3.1. Local ISI installation

Local ISI installation procedure is quite simple. It is necessary to match the following software requirements:

[3] YAML format description, https://yaml.org/

[4] Made with docker-compose-viz tool, https://github.com/pmsipilot/docker-compose-viz

- Docker must be installed (tested with Docker Community Edition, version 18.09);
- Docker Compose must be installed (tested with version 1.24.0);
- Obtain digital certificates for TLS (tested with Let’s Encrypt free certificates).

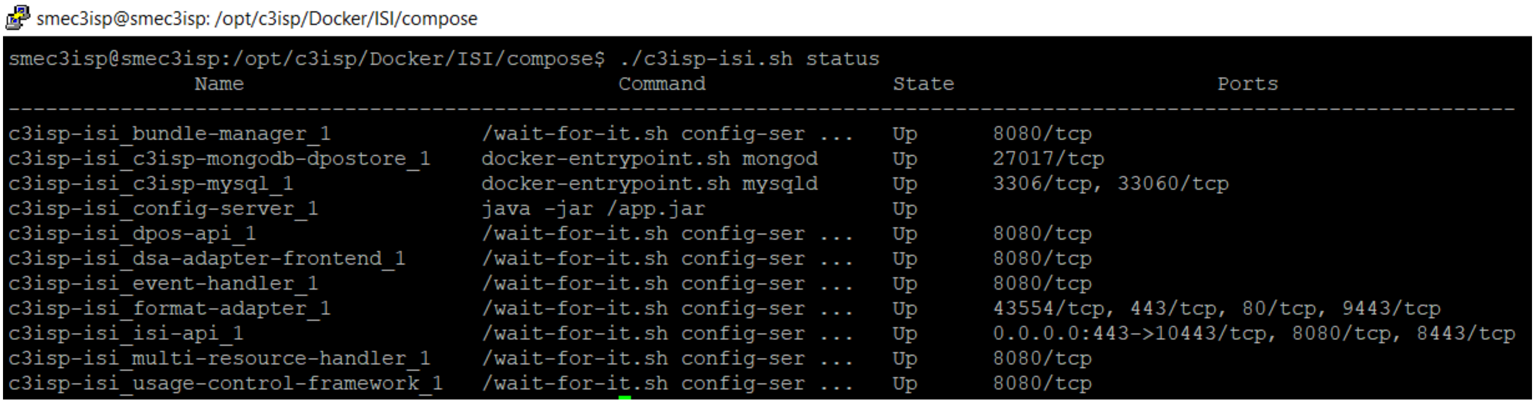
We created a tarball archive that contains the required directory tree layout, necessary to create the directories for data persistence of MySQL (used by DSA Adapter component), MongoDB and DPOS (used by Data Protected Object Storage component). For example, the tarball can be distributed for Local ISI setup at SMEs and ISPs premises.

Once extracted, there is a configuration file (/opt/c3isp/Docker/ISI/compose/.env) containing some specific settings the user can customise, e.g. credentials for the databases, C3ISP Framework version and Configuration Server profile to use (Test Bed, Production), etc.

A simple script allows starting and stopping the Local ISI as a whole, inspecting its logs and checking the running status. At start-up, the script connects to a private Docker Registry (deployed on our C3ISP infrastructure) to download the required C3ISP container images.

Once started, Local ISI entry point is the ISI API, which can be accessed at a fixed URL: https://<host_fqdn>:443/isi-api/swagger-ui.html, where <host_fqdn> must match the name of the digital certificate in use.

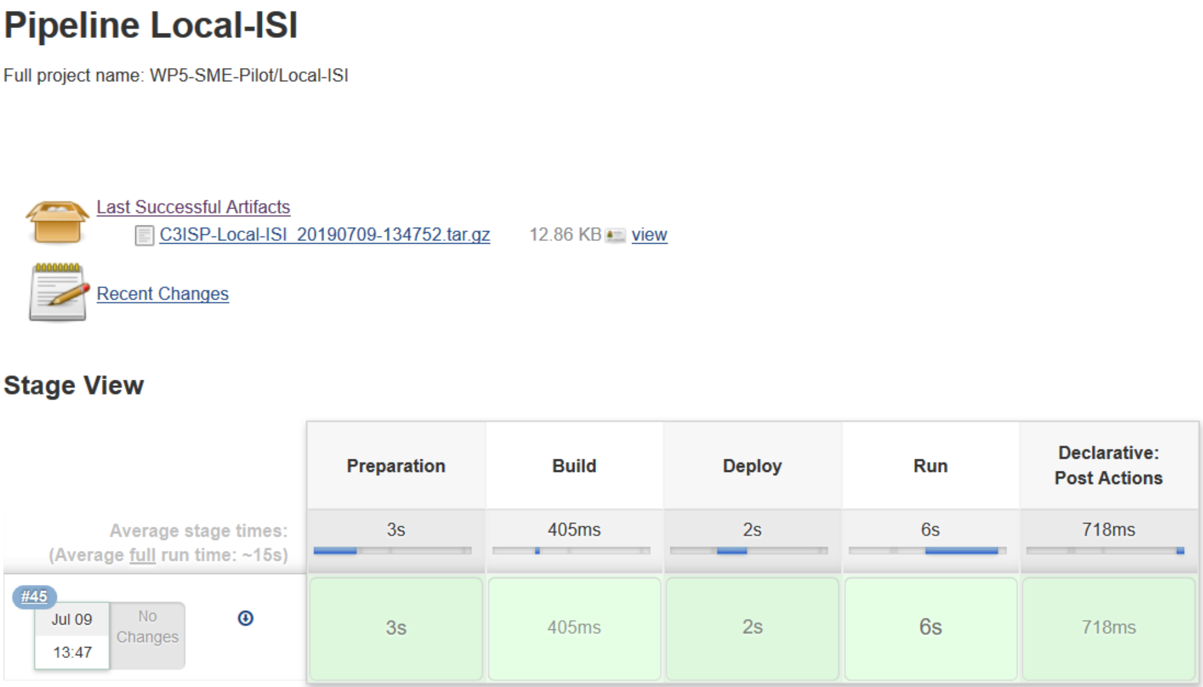
In Figure 5, a sample screen showing the containers status for Local ISI deployed for the SME Pilot:



```
smec3isp@smec3isp:/opt/c3isp/Docker/ISI/compose$ ./c3isp-isi.sh status
               Name                              Command               State                     Ports
-------------------------------------------------------------------------------------------------------------------------
c3isp-isi_bundle-manager_1            /wait-for-it.sh config-ser ...   Up      8080/tcp
c3isp-isi_c3isp-mongodb-dpostore_1    docker-entrypoint.sh mongod      Up      27017/tcp
c3isp-isi_c3isp-mysql_1               docker-entrypoint.sh mysqld      Up      3306/tcp, 33060/tcp
c3isp-isi_config-server_1             java -jar /app.jar               Up
c3isp-isi_dpos-api_1                  /wait-for-it.sh config-ser ...   Up      8080/tcp
c3isp-isi_dsa-adapter-frontend_1      /wait-for-it.sh config-ser ...   Up      8080/tcp
c3isp-isi_event-handler_1             /wait-for-it.sh config-ser ...   Up      8080/tcp
c3isp-isi_format-adapter_1            /wait-for-it.sh config-ser ...   Up      43554/tcp, 443/tcp, 80/tcp, 9443/tcp
c3isp-isi_isi-api_1                   /wait-for-it.sh config-ser ...   Up      0.0.0.0:443->10443/tcp, 8080/tcp, 8443/tcp
c3isp-isi_multi-resource-handler_1    /wait-for-it.sh config-ser ...   Up      8080/tcp
c3isp-isi_usage-control-framework_1   /wait-for-it.sh config-ser ...   Up      8080/tcp
```

Figure 5: Running status of Local ISI at SME Pilot

To ease the setup of Local ISI reference installation, in particular for SME, ISP and CERT Pilots installations, we also setup a Jenkins pipeline job. This automatically fetches the latest Docker Compose configuration from the internal Git repository (phase: Preparation), creates the tarball (phase: Build), installs the artefacts by transferring them to the target Local ISI machine (phase: Deploy), starts all the micro-services (phase: Run) and finally archives the tarball for historic reference into C3ISP internal Nexus repository (phase: Post Actions). This flow is depicted in Figure 6:



Pipeline Local-ISI

Full project name: WP5-SME-Pilot/Local-ISI

Last Successful Artifacts

C3ISP-Local-ISI_20190709-134752.tar.gz 12.86 KB view

Recent Changes

Stage View

	Preparation	Build	Deploy	Run	Declarative: Post Actions
Average stage times: (Average full run time: ~15s)	3s	405ms	2s	6s	718ms
#45 Jul 09 13:47 No Changes	3s	405ms	2s	6s	718ms

Figure 6: Automatic installation of Local ISI on WP5 SME Pilot via Jenkins Pipeline

4. Subsystem: ISI – Information Sharing Infrastructure

4.1. DSA Adapter

The DSA Adapter is the component of the C3ISP architecture, which is in charge of evaluating the DSA policy paired with the CTI data and enforcing it when the execution of some operation on the data is requested (e.g. create, read, move, analytics execution, etc.).

Figure 7 illustrates the final version of this crucial component of the C3ISP Framework:

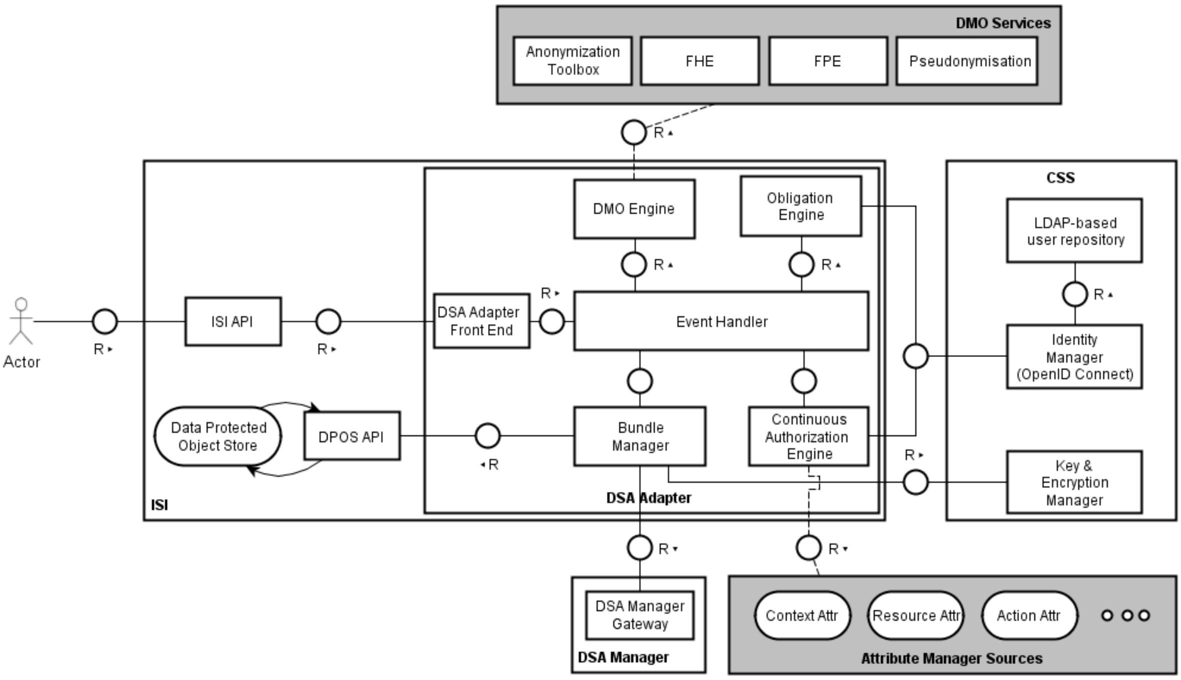


Figure 7: Components of the DSA Adapter (final)

With respect to the previous version, the set of DMO services have been extended with Format Preserving Encryption (FPE) and Pseudonymisation capabilities (their description is reported in D8.3, Section 8). Also, we introduced OpenID Connect as a standard authentication service, as discussed in Section 7.1.2.

The next sub-sections illustrates specific updates to the DSA Manager components.

4.1.1. DSA Adapter Front End

The **DSA Adapter Front End** is in charge of receiving incoming requests (from the ISI API) and to return the related answers to the requestor. The DSA Adapter Front End implements each request it receives by means of an in-house computation followed by calls to other ISI components. Such orchestration is achieved by means of the **Event Handler** that allows for message exchanges from and to the rest of the ISI components, to implement the ISI functionalities. The architecture of this component has not changed since last deliverable as is reported next for the sake of completeness:

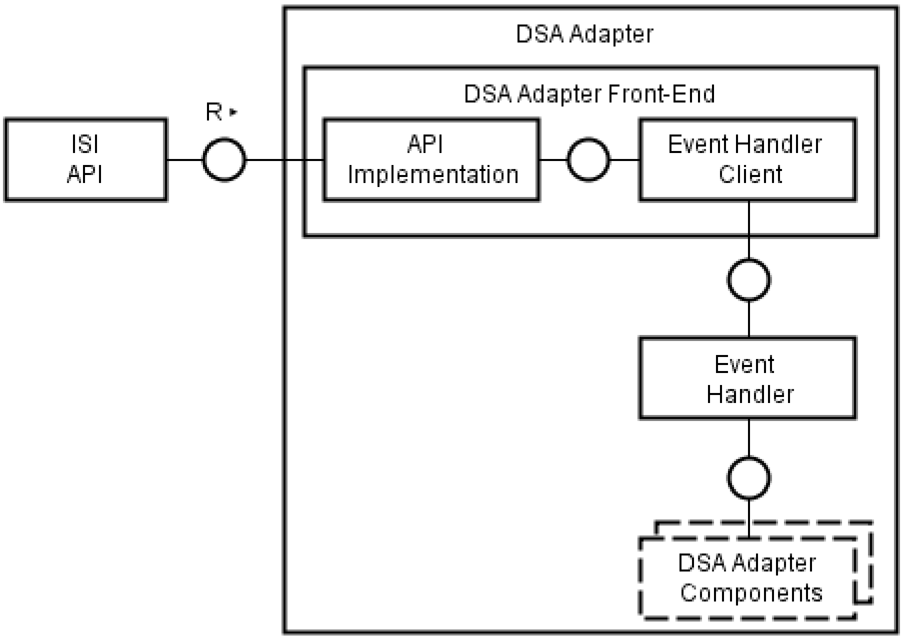


Figure 8: DSA Adapter Front End architecture

The DSA Adapter Front End exposes a number of methods to implement all ISI API functionalities as public API. Namely, they are:

- **Create DPO**: it creates a new DPO and returns a DPO_ID;
- **Read DPO**: it returns the content of a DPO;
- **Delete DPO**: it deletes a DPO, identified by the DPO_ID;
- **Move DPO**: implementation of the Move operation, with two methods (for supporting the two cases: DPO sender – DPO receiver) (see 8.2);
- **PrepareData**: a necessary functionality for analytics operations. It allows to generate a VDL/DLB for a set of DPO_IDs if the call is appropriately authorised, by calling the respective Buffer Manager operation (see 4.3).

4.1.2. Event Handler

The **Event Handler** is a component that allows an exchange of messages among the components of the ISI. It is in charge of collecting and distributing the messages, also called events, resulting from the computation done by the modules of the DSA Adapter. It allows other components to register for events, and it notifies them when such events occur. The architecture of this component has not changed since last reporting period and is outlined in Figure 9:

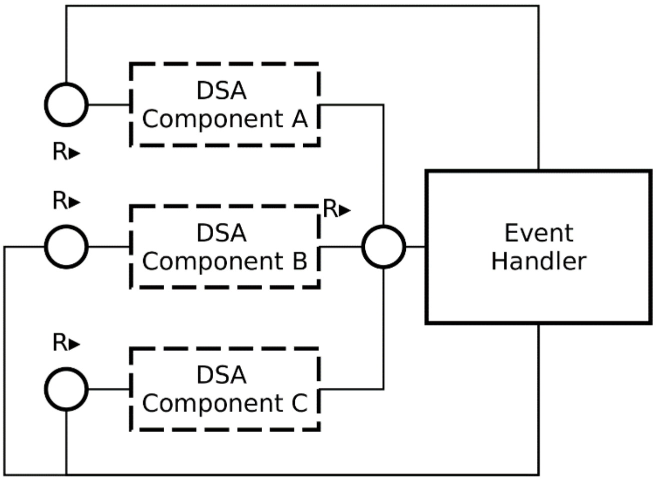


Figure 9: Event Handler architecture with interacting DSA Manager's components

The Event Handler exposes a simple API:

- **Subscribe/Unsubscribe**: a module can register/unregister an own endpoint to receive messages of a specific type;
- **NotifyEvent**: it allows to submit a new message that will be routed to all subscribed modules. Importantly, all components that communicate with the Event Handler must expose a method with a similar signature, in order to receive in real time new messages from the Event Handler.

4.1.3. Continuous Authorization Engine

An instance of the **Continuous Authorization Engine** (CAE) is integrated within the ISI subsystem in order to control the usage of the set of CTIs involved in each analytic request (another instance is integrated in the IAI subsystem, i.e., at service level, see 5.2). This component is responsible for enforcing the usage control policy paired with a set of CTIs both at access request time (to determine whether each CTI in the request can be exploited for executing a given action, e.g., a read action or the execution of an analytics) and continuously while the usage of the CTIs is in progress, following the UCON model, to ensure that the right to use each of the CTI in the set still holds. The internal architecture of this component (shown in Figure 10) has been defined at M24, and no changes has been done at M34. The features and functionality of the internal components of the Continuous Authorization Engine, instead, have been maturated at M34 to address the C3ISP requirements, and a detailed description can be found in Deliverable D8.3.

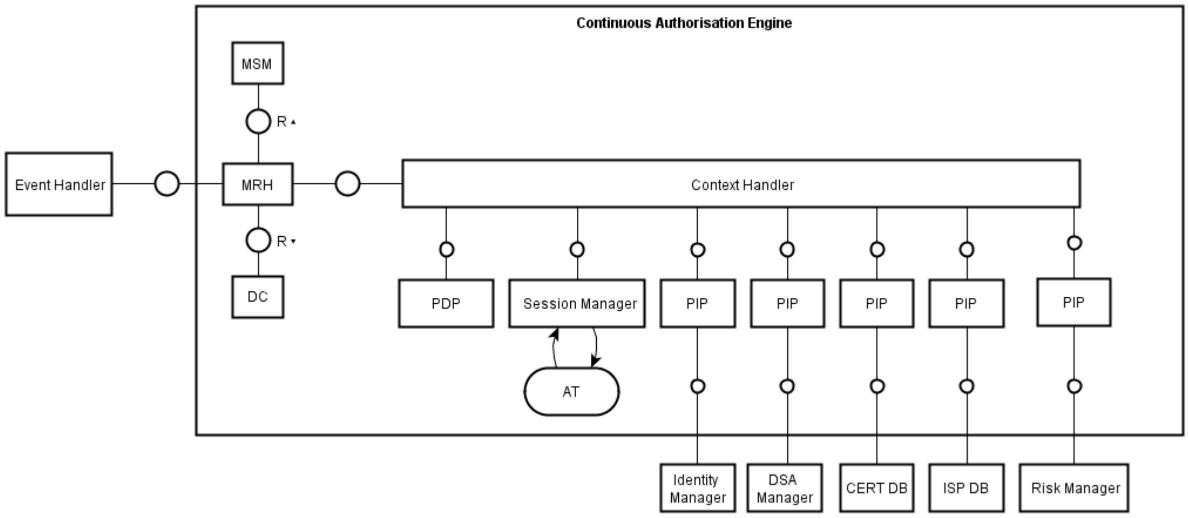


Figure 10: Internal Architecture of the Continuous Authorisation Engine (MRH=Multi-Resource Handler; DC=Decision Combiner; MSM=Multi Session Manager; AT=Access Table; PDP=Policy Decision Point; PIP=Policy Information Point)

We report here the extensions to D7.3 we did to finalise the implementation and the fulfilment of the requirements. These are related to the following CAE modules:

- **Attribute Managers** (AMs), the modules which manage the attributes required to evaluate the usage control policies (e.g. Identity Manager and MySQL);
- **Policy Information Points** (PIPs), the interfaces for interacting with the Attribute Managers to perform operations on attributes: *retrieve* (its value), *subscribe/unsubscribe* (for value changes) and *update* (to possibly alter the value).

A new Attribute Manager has been introduced called **Risk Manager** (to address requirement C3ISP-Fun-DS-012), which is described next:

- **Risk Manager** (AM): this AM provides an estimation of the risk related to a given analytics request taking into account all of its attributes. Further details concerning this AM can be found in D8.3 Section 5.7.4.7.

Consequently, a new PIP has been introduced to interact with the above-described Risk Manager AM:

- **Risk Manager** (PIP): is exploited to retrieve the current risk value of performing a given analytic or action over a given set of CTIs.

4.1.4. Obligation Engine

The **Obligation Engine** is a module that is responsible for the execution of specific operations when certain conditions take place. Such operations are *Usage Control Obligations* that are prescribed by the DSA policy (i.e. the sticky policy) associated to a specific data.

The architecture of the Obligation Engine has not changed since D7.3 released and we report in the next diagram its current status:

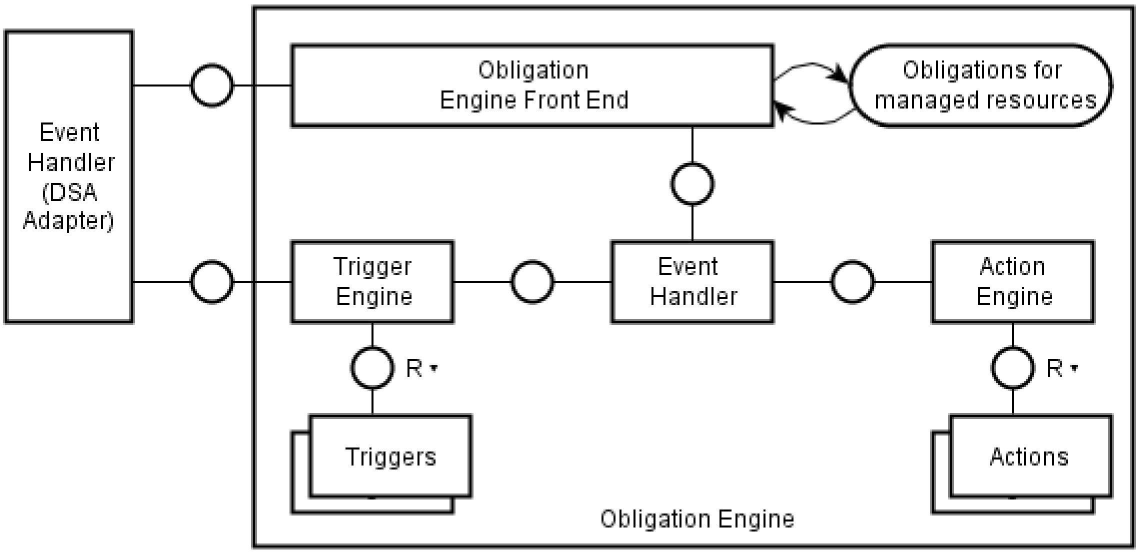


Figure 11: Obligation Engine architecture

After an analysis of the requirements expressed by the Pilots with respect to the formalisation of their DSAs, it results that the initial set of triggers and actions planned is sufficient to serve the Pilots' use cases, with one notable exception. A number of Pilots expressed an interest for a mechanism that notifies data owners when a new analytics result is created out of their DPOs (see C3ISP-Fun-DS-009 in Section 9). This leads to the definition of a new Trigger, called *AnalyticsCompleted*, which can be used in DSA statements to express obligations connected to such event. A new action, *NotifySubject* was also defined, to allow for requesting a notification to be sent (see also notifications in 6.1).

4.1.5. DMO Engine

The **Data Manipulation Operation (DMO) Engine** is the component in charge of executing the Data Manipulation Operation prescribed as part of the decision process on the data and/or by any obligation as prescribed by the data's DSA. In fact, besides determining whether the data can be accessed or not by the requestor, the decision process also determines a set of operations that must be executed on such data before being released to the requestor. As an example, a DSA paired to a system log could require that all the IP addresses present in such log must be anonymised before releasing this log to a third party. A similar action may be mandated also in case a retention period for the log is expired, irrespective of any access request. The DMO Engine is the component of the DSA Adapter devoted to performing such anonymisation operation on the log.

The architecture of the DMO Engine is plugin-based, in order to support an arbitrary number of manipulation operations, and it is depicted in the following Figure 12 as it has not changed since last reporting period (see D7.3 for further details):

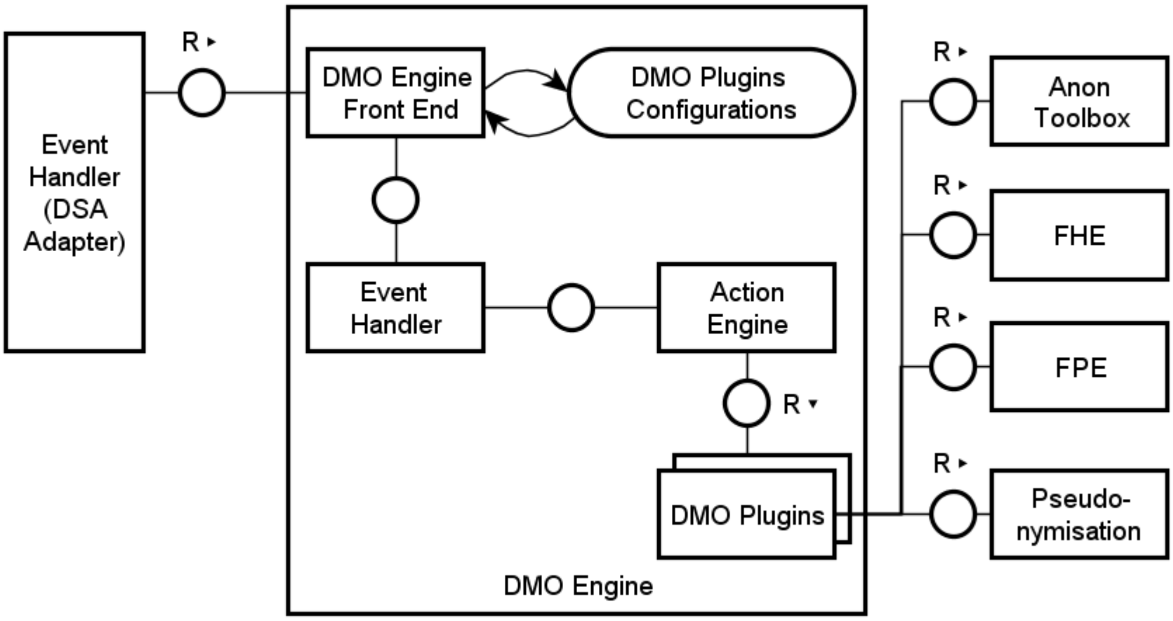


Figure 12: DMO Engine Architecture

As mentioned, the DMO Engine implements a set of operations which cover the requirements of the C3ISP Pilots. In particular, we developed the following list of DMOs:

- Anonymisation based on:
    - Simple attribute suppression/replacement;
    - Differential privacy techniques (e.g. to achieve geo-indistinguishability, see D7.2 – Appendix 1);
- Symmetric homomorphic-friendly encryption, used to pre-process data for applying homomorphic computations (see the *transcryption* process in D7.2 – Appendix 2);
- Format Preserving Encryption, to allow analytics on structured information to be performed flawlessly despite the data contents being encrypted (see corresponding section in D8.3);
- Key-based Pseudonymisation, to anonymise data content maintaining some data properties (e.g. field length), in a non-reversible way (see corresponding section in D8.3).

4.1.6. Bundle Manager

The **Bundle Manager** is the component that handles the *C3ISP Data Bundle*, i.e., the container of the Data Protected Object (DPO). It is used for both packing and unpacking operations. In the packing phase, the Bundler Manager is used for creating a bundle by:

(i) Retrieving a DSA;

(ii) Pairing the DSA with the provided CTI data.

In the unpacking phase, instead, it is used:

(i) To extract the paired DSA from the bundle, which will be sent to the Event Handler (4.1.2) for policy evaluation, and

(ii) To retrieve the CTI data (if allowed by the DSA policies evaluation outcome).

At M34, the **Bundle Manager (BM)** aims at optimising the performance for creating, deleting and retrieving packets of data by invoking specific functions in **DPOS API** component (*Data Protected Object Store API*). About this optimised version of BM, we added some supplement features:

1. BM supports working with binary files. This feature is only concerned for supporting the storage of FHE files (see 8.3.1);
2. BM allows working with a reference (*link*) to a file instead of the file content. This version allows decreasing the communication bandwidth.

Hence, we have two possibilities for creating and reading CTI bundles, the first one is described in the previous deliverable D7.3 and works directly with the file data object, the second one is the version working with the link to a location of a data object, in particular:

a. The CTI data in which the content is the *link* to the CTI location;

b. The metadata file, which is a description of the bundle, in which the content is the *link* to the metadata location.

The workflow is not changed since the previous deliverable. For more details about this feature, please refer to D8.3 section 5.10. Below is the architecture of the Bundle Manager taken from D7.3:

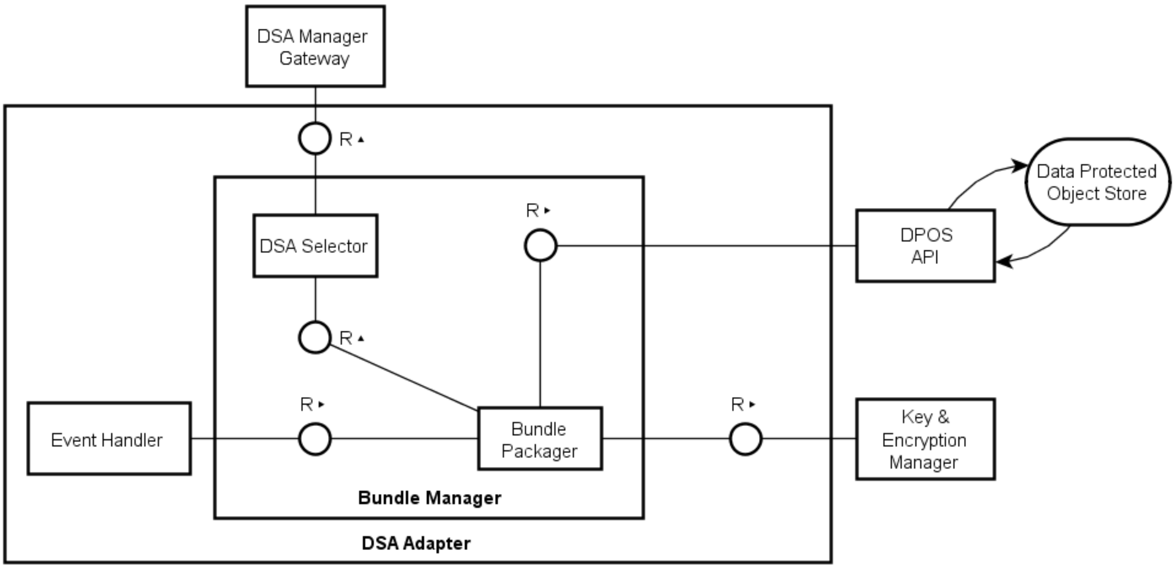


Figure 13: Bundle Manager architecture

4.2. *Format Adapter*

The **Format Adapter** is the component of the C3ISP Framework in charge of the following functions:

- **Convert (/convert API)**: adapts the format of input CTI data to a standard format (STIX) to be easily processed by the various C3ISP components. To accomplish this, the Format Adapter is called by the ISI API;
- **Convert for Data Lake (/convertDL API)**: formats the CTI data appropriately for a given analytics service before executing it. This is done when the Buffer Manager calls the Format Adapter in order to have the data prepared in the right format for the analytics.

The component has been extended to integrate with all the planned data types as shown next:

Table 2 – Integrated CTI data types

CTI data	Integration Status
Monitoring of connections to malicious hosts	✓
Monitoring of Domain Generation Algorithm DNS-request	✓
Email Analysis	✓
Firewall Schema	✓
Anti-Malware Schema	✓
Security Report Sharing	✓
Enterprise Pilot (Intrusion Detection Events, Malware Events, Network Traffic Events, Web Events)	✓
IDEA to STIX	✓

The Format Adapter has also been extended in order to be able to “flatten” a STIX with CEF file format into plain CSV (/**convertDL API**). This was required to allow the integration of the Legacy Analytics Service (BT Saturn) with HDFS Data Lake based on Apache Impala technology (ref. [63]) via the Buffer Manager (see 4.3).

The Format Adapter is now able to automatically detect the incoming format and converts appropriately to STIX for C3ISP usage.

In particular, the Format Adapter now includes a conversion from IDEA to STIX, thanks to the collaboration with the European Project PROTECTIVE[5]. PROTECTIVE gathers and exchanges communication between automated detection systems and information aggregators of various types. This type of data manifests wide variability, so its representation should be extensible enough to be prepared for unexpected or new types of security events. The format should be simple, because complexity and intricacy bring ambiguity. The main propose of the IDEA format is an attempt to define nowadays requirements and propose foundations for viable solution for security event model, taking into consideration existing formats, their benefits and drawbacks. Even though there exists a variety of models for communication between honeypots, agents, detection probes, none of them was really used because of various limitations for general usage. *IDEA* stands for *Intrusion Detection Extensible Alert* and is the format PROTECTIVE uses for security event exchange. With this integration, PROTECTIVE is able to share its data with the C3ISP Framework.

This is an example of a JSON-serialised IDEA synthetic security event:

```
{
```

[5] PROTECTIVE – Proactive Risk Management, https://protective-h2020.eu/

```
    "Format": "IDEA0",
    "ID": "4390fc3f-c753-4a3e-bc83-1b44f24baf75",
    "CreateTime": "2012-11-03T10:00:02Z",
    "DetectTime": "2012-11-03T10:00:07Z",
    "WinStartTime": "2012-11-03T05:00:00Z",
    "WinEndTime": "2012-11-03T10:00:00Z",
    "EventTime": "2012-11-03T07:36:00Z",
    "CeaseTime": "2012-11-03T09:55:22Z",
    "Category": ["Fraud.Phishing"],
    "Ref": ["cve:CVE-1234-5678"],
    "Confidence": 1,
    "Note": "Synthetic example",
    "ConnCount": 20,
    "Source": [
       {
          "Type": ["Phishing"],
          "IP4": ["192.168.0.2-192.168.0.5", "192.168.0.10/25"],
          "IP6": ["2001:0db8:0000:0000:0000:ff00:0042::/112"],
          "Hostname": ["example.com"],
          "URL": ["http://example.com/cgi-bin/killemall"],
          "Proto": ["tcp", "http"],
          "AttachHand": ["att1"],
          "Netname": ["ripe:IANA-CBLK-RESERVED1"]
       }
    ],
    "Target": [
       {
          "Type": ["Backscatter", "OriginSpam"],
          "Email": ["innocent@example.com"],
          "Spoofed": true
       },
       {
          "IP4": ["10.2.2.0/24"],
          "Anonymised": true
       }
    ],
    "Attach": [
       {
          "Handle": "att1",
          "FileName": ["killemall"],
          "Type": ["Malware"],
          "ContentType": "application/octet-stream",
          "Hash": ["sha1:0c4a38c3569f0cc632e74f4c"],
          "Size": 46,
          "Ref": ["Trojan-Spy:W32/FinSpy.A"],
          "ContentEncoding": "base64",
          "Content": "TVpqdXN0a2lkZGluZwo="
       }
    ],
    "Node": [
       {
          "Name": "cz.cesnet.kippo-honey",
          "Type": ["Protocol", "Honeypot"],
          "SW": ["Kippo"],
          "AggrWin": "00:05:00"
       }
    ]
 }
```

The Format Adapter converts this JSON in a STIX 2.0 format, which can be used internally by the C3ISP Framework.

Example of transformed output is the following:

```
{
  "type": "bundle",
  "id": "bundle--741b64fc-bdf8-4ae8-b4bb-865c10137540",
  "spec_version": "2.0",
  "objects": [
    {
      "type": "identity",
      "id": "identity—-00f3fe54-dd02-4307-88d2-fdc0ef9ecc8a",
      "created": "2012-11-03T10:00:02Z",
      "modified": "2012-11-03T10:00:02Z",
      "name": "cz.cesnet.kippo-honey",
      "labels": [
        "Protocol",
        "Honeypot"
      ],
      "description": [
        "Kippo"
      ],
      "identity_class": "group"
    },
    {
      "type": "observed-data",
      "id": "observed-data--4afddef3-dec1-4e2f-8121-8df4ada16aa9",
      "created_by_ref": "identity—-00f3fe54-dd02-4307-88d2-fdc0ef9ecc8a",
      "created": "2012-11-03T10:00:07Z",
      "modified": "2012-11-03T10:00:07Z",
      "first_observed": "2012-11-03T07:36:00Z",
      "last_observed": "2012-11-03T09:55:22Z",
      "labels": [
        "Fraud.Phishing"
      ],
      "number-observed": 20,
      "objects": {
        "0": {
          "type": "ipv4-addr",
          "value": "192.168.0.2-192.168.0.5,192.168.0.10/25"
        },
        "1": {
          "type": "ipv6-addr",
          "value": "2001:0db8:0000:0000:0000:ff00:0042::/112"
        },
        "2": {
          "type": "ipv4-addr",
          "value": "10.2.2.0/24"
        },
        "3": {
          "type": "network-traffic",
          "src_ref": "0",
          "protocols": [
            "ipv6"
          ]
        },
        "5": {
          "type": "network-traffic",
          "dst_ref": "2",
          "protocols": [
            "ipv4"
          ]
        }
      }
    },
    {
      "type": "sighting",
      "id": "sighting--27bb0d2d-3a13-4631-8bc2-7dfb63a02e8d",
      "created_by_ref": "identity—-00f3fe54-dd02-4307-88d2-fdc0ef9ecc8a",
      "created": "2012-11-03T10:00:07Z",
      "modified": "2012-11-03T10:00:07Z",
      "first_seen": "2012-11-03T07:36:00Z",
      "last_seen": "2012-11-03T09:55:22Z",
      "count": 20,
      "sighting_of_ref": "observed-data--4afddef3-dec1-4e2f-8121-8df4ada16aa9",
      "where_sighted_refs": "identity—-00f3fe54-dd02-4307-88d2-fdc0ef9ecc8a"
    }
  ]
}
```

IDEA messages can be converted to STIX thanks to the **/convert API**: there is no need to specify the format type since the Format Adapter detects it automatically like every other format it supports.

4.3. Buffer Manager

The **Buffer Manager** is the component of the C3ISP Framework that manages Data Lakes.

A Data Lake is a storage area that is used to store data before and after analytics tasks executed by C3ISP-aware analytics services or Legacy Analytics services. Data Lakes are of two types: **Data Lake Buffer** (DLB) and **Virtual Data Lake** (VDL) that are different in terms of how they access and read the data. They are transient, meaning that they are created and reserved only for the execution of a specific Analytic service; then, they get removed.

The design specifications of the Buffer Manager have not varied significantly from M24, but important internal implementation improvements have been made.

The Buffer Manager is fully integrated with the Format Adapter component to perform the conversion between CEF and STIX formats. The integration has been intensively used during the last year.

The implementation of the new **HDFS Data Lake** has been designed and executed. The Buffer Manager writes data on the HDFS and makes them available to a Legacy Analytic service (provided by BT) which reads them from a linked **Apache Impala** [63] table. The Legacy Analytic service requires data in CSV format and a new feature of the Format Adapter has been implemented (see 4.2).

The next figures show the designed and implemented flow of interaction between the Buffer Manager and the Apache Impala running on a Cloudera [63] Virtual Machine (VM) distribution. In particular, Figure 14 shows that at analytics runtime, upon receiving a *prepareData* from ISI API, the Buffer Manager creates a Virtual Data Lake on the HDFS running in the Cloudera VM. The dataset is then loaded by the Buffer Manager via a JDBC driver for Apache Impala that ultimately writes the data on the ad-hoc prepared Virtual Data Lake (HDFS).

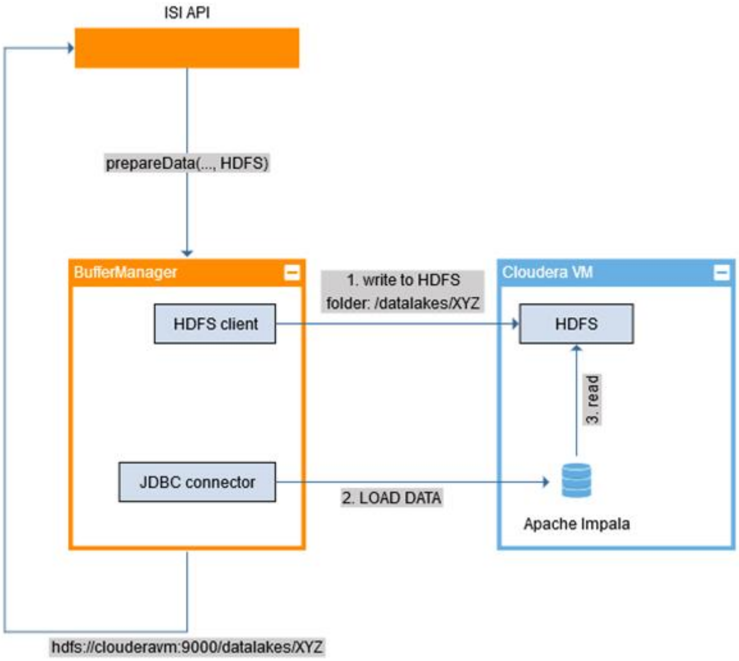


Figure 14: The workflow to write data on a HDFS Data Lake

Then the Legacy Analytics engine (BT Saturn) can access the Virtual Data Lake by using its native support for Apache Impala (in a SQL-like fashion):

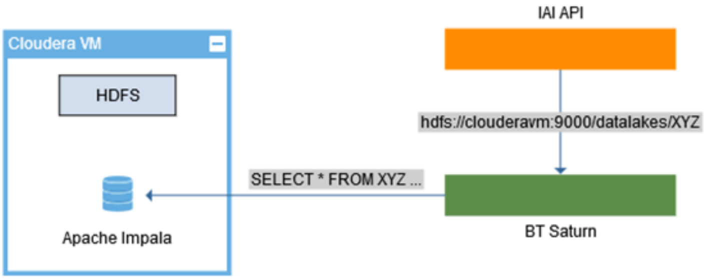


Figure 15: The BT Legacy Analytic Saturn reads data from Apache Impala

Finally, a new **configuration** system is now in use for the Buffer Manager and all the Data Lakes, since it has integrated with the Configuration Server that can provide different configurations for different environments.

4.4. *Data Protected Object Store API*

The **Data Protected Object (DPO) Store API** is the external interface of the DPO Store (DPOS, see 4.5), and it provides functions for managing the storage of DPOs. The DPO Store API is used by the ISI components to manage the storage of Data Protected Objects. It is also indirectly used by the IAI subsystem to search and retrieve DPOs for use in collaborative analytics, via the ISI API. The interface is stable and has not undergone major changes since last reporting period. In particular, some optimisation about handling big files has been done.

The DPO Store API supports the following functionalities:

- **CreateDPO**: Given a CTI file, and the associated DSA file, hash code and a DPO metadata header, creates a DPO in the DPO Store;
- **ReadDPO**: Retrieves the four components of a DPO from the DPO Store repository, given its DPO_ID;
- **DeleteDPO**: Given a DPO_ID, deletes the corresponding DPO from the repository;
- **SearchDPO**: Given a JSON-based search string (described in detail in D8.2), query the DPO metadata repository, and returns a set of metadata entries corresponding to the matching DPOs. This method returns either a set of DPO_IDs or a set of full DPO metadata entries (when the flag *longResultFlag* is set).

The DPO Store API micro-service has been integrated with the Configuration Server described in Section 2.1.

4.5. *Data Protected Object Store*

The **Data Protected Object Store (DPOS)** persistently stores the CTI data provided by the Prosumers in the form of the C3ISP data bundle (or DPO – Data Protected Object), i.e. the CTI data, the corresponding DSA, hash code, and DPO metadata. It functions as the backend to the DPO Store API.

For Local ISI, which is typically used by a single Prosumer and has lower computational resources than Central ISI, the DPOS is implemented as a protected filesystem, i.e. operating system access is restricted to the user impersonating the application server that runs the C3ISP Framework components. For the containerised Local ISI, the DPOS is mapped to a Docker Volume[6], which easily permits advanced functionalities like storing data on remote hosts or cloud providers, and to encrypt its content.

For Central ISI, which is where several Prosumers store their data, i.e. a possibly huge data repository, the DPOS is implemented as an Apache Hadoop Filesystem (HDFS), a highly available distributed filesystem suitable for big data processing (cf. [54]).

4.6. ISI API

The **ISI API** is the front-end component providing services to the C3ISP actors (Prosumers or Prosumers applications, or the IAI subsystem). It authenticates incoming requests and orchestrates the processing flow with the other ISI components, interacting with the DSA Adapter or the Format Adapter (see the data flow diagrams in D7.3 Section 9). All the APIs are actions that are subject to the (DSA) policies associated to each DPO data object.

We report next the final list of the operations of the ISI API:

- **Create DPO**: used to submit a raw CTI data that will be paired with a DSA to create the C3ISP Data Protected Object bundle (DPO). It returns a DPO_ID used for other operations. The Create DPO requires the identifier of the DSA to associate the data with, and the DSA policies are evaluated to authorise the "Create" action (DMOs can also be specified, e.g. to anonymise the data before storing it). In particular, unless an explicit policy granting the "Create" is defined, the action is denied;
- **Read DPO**: used to retrieve in raw (and clear-text) format a previously stored CTI data from the ISI once the paired DSA policies have been met, including prescribed authorisation rules and Data Manipulation Operations;
- **Move DPO**: used to move a C3ISP DPO from an ISI node to another (e.g. from a Local ISI to a Central ISI), if the DSA policy allows it. Two sub-methods (**Move** and **Transfer**) permit the materialisation of this functionality, as described in Section 8.2. The "Move" action must be explicitly authorised in a DSA policy, otherwise it is forbidden.
- **Delete DPO**: used to remove a DPO that have been previously stored (if DSA policies allow that action).
- **Prepare Data**: used for preparing data for performing analytics service; it accepts a number of DPO_IDs and ensures that all DPOs are retrieved (through the input DPO_ID), formatted and provisioned to a DLB/VDL; the method returns a reference to the prepared (virtual) data lake in the requested format. The "Prepare Data" requires DSA policies to allow reading the data and the execution of the requested analytics service.

The Create DPO is also used to submit to the C3ISP Framework the result of the analytics service executed by the IAI subsystem. As we described in D7.2 section 6 (Policies for Derived Data), a DSA can contain policies to be used for analytics result (see Policy Merger in Section 6.3).

[6] Docker Volumes, https://docs.docker.com/storage/volumes/

5. Subsystem: IAI – Information Analytics Infrastructure

5.1. C3ISP Analytics Engine

The **C3ISP Analytics Engine** architecture has not changed since last reporting period. However, it has been extended to support the addition of custom analytics such in a way that expert users can program their own analytics to be compatible with the C3ISP Framework. Section 5.5 reports about the C3ISP Analytics SDK for extending the IAI API. The architecture of this component is reported in Figure 16. It includes *computational intelligence functions*, in particular clustering and classification algorithms, data aggregation and correlation functions, statistical analysis tools, and data visualisation primitives. These functions are either implemented through open source libraries for Machine Learning (ML) and statistical analysis, in particular WEKA[7] and Scikit-Learn[8] libraries, and tools for Big Data analysis.

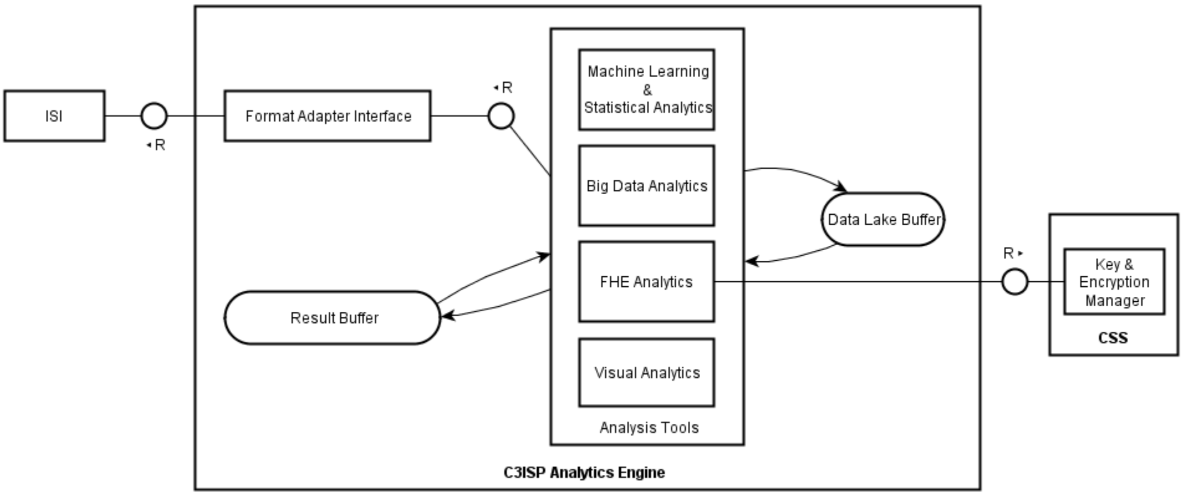


Figure 16: C3ISP Analytics Engine

The **Analysis Tools** represents the core computational services, which are supported by the following three functional modules to handle the data flow in the engine.

The **Format Adapter Interface**, is an interface to the Format Adapter component (via the ISI API), which converts the structured CTI information (i.e. in STIX format), to the one needed by the required analysis algorithm.

The **Data Lake Buffer** (DLB) is a temporary storage created by the Buffer Manager (see 4.3) that keeps formatted CTI data used by the analytics service. The DLB format depends on the analytics services requirements: the Buffer Manager has been defined to support the set of buffer structures needed by the C3ISP analytics services, including standard filesystem, relational database (MySQL) and Hadoop filesystem (via Apache Impala), as described in Section 4.3 and D7.3 (Section 5.3).

The **Result Buffer** contains temporary and final results, acting thus both as a complimentary component to the Data Lake Buffer, and to store the final results before they are sent to the ISI via the Format Adapter Interface.

Analytics functions performed via the C3ISP Analytics Engine obey to the DSA policies and in particular generates a CTI result which is published into the Information Sharing Infrastructure (ISI), with its own DSA policies (see DSA Policy Merger component in Section

[7] http://www.cs.waikato.ac.nz/ml/weka/

[8] http://scikit-learn.org/stable/

6.3). Analytics operations are compatible with the DMOs, like the FHE encryption which is required by the FHE Analytics described next.

5.1.1. FHE Analytics

The **FHE Analytics** module is in charge of performing *homomorphic computation functions* (HE computation or FHE analytics – for short) on homomorphic encrypted data (D7.3 Section 6.1.2 describes it in detail). The homomorphic encryption process is handled by the Key & Encryption Manager component services, reported in Section 7.2, so please refer there for all the details, in particular the FHE Analytics uses the *Homomorphic Encryption* technology.

At M34 the FHE Analytics are used to make computations directly on encrypted data and focuses on a **blacklist algorithm** using short (32-bits) and fixed-sized words (i.e. IPv4 addresses). The performance of this feature depends on the chosen security analysis level, meaning it is possible to use a mechanism of partial encryption. FHE encryption for IPv4 can be partial with respect to the IPv4 octets: a level of BLACK_LIST_LOW (last octet only) is faster than BLACK_LIST_FULL (all the IPv4 octets), but its security is much lower than BLACK_LIST_FULL.

FHE Analytics offer tools which permit to detect if an encrypted IPv4 belongs to a set of encrypted IPv4s. There are 4 security levels when encrypting IPv4:

- BLACK_LIST_FULL (all IPv4 octets);
- BLACK_LIST_HIGH (last 3,2,1 octets);
- BLACK_LIST_MEDIUM (last 2,1 octets);
- BLACK_LIST_LOW (last octet).

More details are available in D8.3 Section 8.2.2.

5.1.2. Interactive 3D Visualisation

The analytics results can be presented in an Interactive 3D Visualisation that can be accessed via a web browser or virtual reality (VR) headset. The visualisation utilises multiple technologies for processing and rendering data. Rhinoceros 3D[9] (Rhino), a computer graphics and computer-aided design (CAD) software, and Grasshopper[10], a visual scripting language supported by Rhino, are used to create a 3D visual representation of the data. As shown in Figure 17, three different representations are used to describe results: (a) nodes marked as *susceptible* are modelled as yellow spheres and *infected* nodes modelled as red diamonds can be used by the ISP Pilot to show the impact of a malware attack; (b) connections between nodes modelled as arcs can be used to show the source and destination of firewall events from the SME Pilot; and (c) anonymised results are modelled as a heat map, which uses the vertical axis to indicate occurrences of events, can be used in the Enterprise Pilot to visualise malware distribution.

[9] https://www.rhino3d.com/

[10] https://www.grasshopper3d.com/

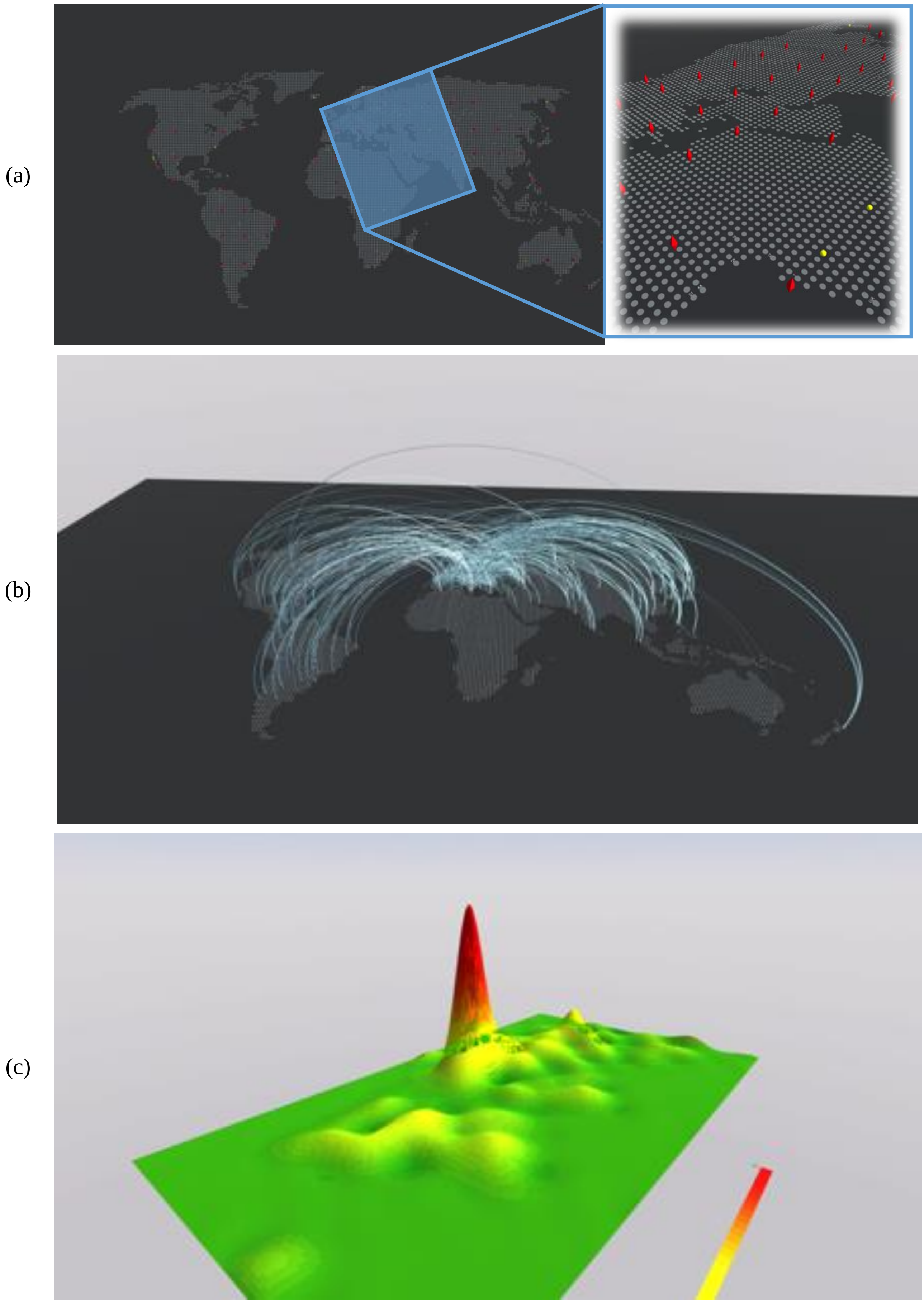


Figure 17: Interactive 3D Visualisation

3D Repo is an open source cloud-based collaboration platform for 3D data. The platform supports 3D visualisation on modern web browsers, such as Google Chrome and Mozilla Firefox, and VR headsets, such as HTC Vive[11] and Oculus Rift[12]. The 3D model file generated by Rhino is uploaded to 3D Repo, which serves as a publishing application for the distribution of the results. Automation of tasks such as grouping of nodes by day or region and improved filtering of the analytics results is achieved through applications created in Node-RED[13], a flow-based development tool, which connects to 3D Repo's API.

5.2. Service Usage Control Adapter

The Service Usage Control Adapter is an instance of the **Continuous Authorization Engine** (described in Section 4.1.3) which is integrated within the IAI subsystem in order to control the usage of the analytics services exposed by the C3ISP Framework. This is responsible for enforcing the usage control policy paired with the C3ISP analytics services both at access request time (to determine whether the user can exploit the analytics service) and continuously, following the UCON model, to ensure that the right to use such service holds while the service is in use. It is invoked by the IAI API when a Prosumer requests the execution of an analytics. The IAI API does not execute the analytics if the Service Usage Control Adapter returns DENY as result of the policy evaluation process. If the result is PERMIT, the IAI API starts the execution of the analytics. The execution of C3ISP-aware analytics could be interrupted because of a policy violation detected by the Service Usage Control Adapter. In this case, the Service Usage Control Adapter interacts with the IAI API which is aware of the procedure required to deal with (e.g. interrupt) the running C3ISP-aware analytics.

The internal architecture of the component, as well as the features and functionality of the internal components, are the same as described in Section 4.1.3. There are only two differences:

- The first difference is that the access request involves one service only, i.e., the service to be invoked;
- The second difference is that the (Service) Usage Control policy that is enforced by the Service Usage Control Adapter is stored in a Policy Administration Point (PAP), which interacts with the Context Handler (CH), and which is installed in the same machine as the Service Usage Control Adapter.

[11] https://www.vive.com/eu/

[12] https://www.oculus.com/rift/

[13] https://github.com/node-red/node-red

The next figure contains the Service Usage Control Adapter architecture, with the Policy Administration Point:

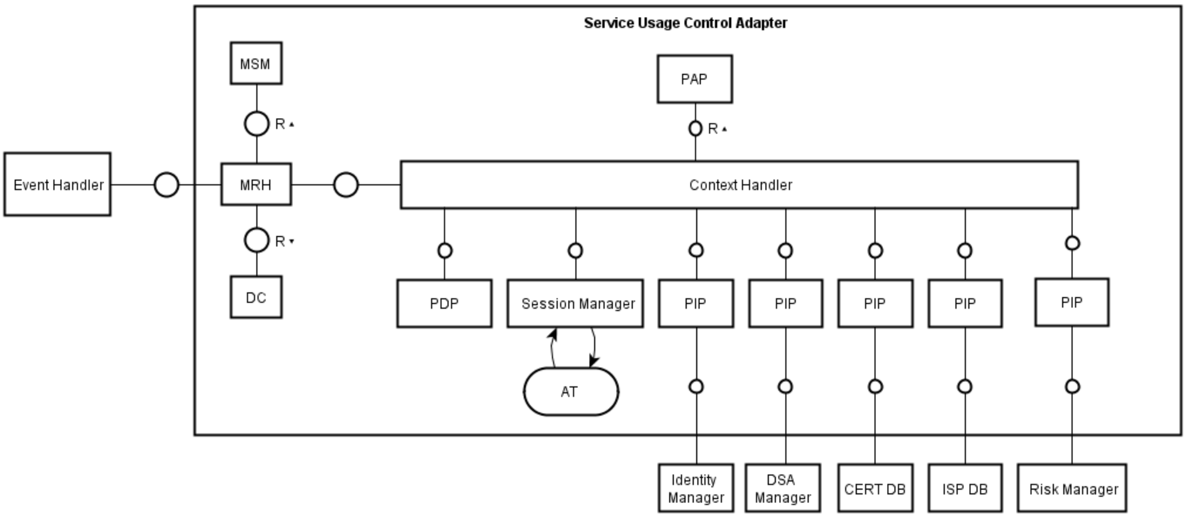


Figure 18: Service Usage Control Adapter architecture

The Service Usage Control policies are written by the C3ISP platform administrators, and they are updated every time a new analytics is installed in the framework in order to regulate who can use that analytics.

There is a UPOL template that can be personalised for each analytics to detail with the specific usage control requirements of the Pilots. Please see Pilots' Deliverables for the specific logic configured in these policies and D8.3 for further details on the implementation of this component.

5.3. Interface to Legacy Analytics Engines

In order to extend its overall analytics capability the C3ISP Framework offers an option to integrate (existing) Legacy Analytics Engines by providing them with access to CTI data that have been shared within the Framework. However access is strictly limited to data that have been transferred to a transient Virtual Data Lake (VDL) instance. Each time a C3ISP Prosumer invokes a Legacy Analytics service a new VDL instance with a unique URI is created and populated with the requested data.

Legacy Analytics Engines come in different forms and thus provide different types of interface. One may only be accessed and consumed through its graphical user interface (e.g. BT Saturn Visual Analytics tool) while the other may provide RESTful or Java API to invoke its analytics functions (e.g. Apache Spark). The IAI provides a RESTful API to invoke the service of a Legacy Analytics Engine (see Section 8.4). In case the Legacy Analytics Engine does not support any API its service invocation will result in data population into a new VDL instance only (see Section 4.3). Otherwise its invocation is similar as in the case of generic C3ISP analytics service except that a VDL instance is used to temporarily store the data.

5.4. Virtual Data Lake

The **Virtual Data Lake (VDL)** is a dedicated service instance for storing CTI data in standard tabular format. A VDL instance is created temporarily/on demand in order to allow a *Legacy Analytics Engine* consume the CTI data or analytics results that have been shared in the C3ISP Framework. Any relevant DSA obligation rules to sanitise the data such as IP address or usernames anonymisation has to be enforced before the data are transferred to the VDL. Once the VDL is made available to a Legacy Analytics Engine the usage and distribution of the stored data cannot be controlled by C3ISP Framework any longer. The VDL instance will be terminated after a certain amount of time or if the data are not further in use.

The VDL provides a standard SQL-like interface for querying the data table. If there are multiple sets of independent CTI data (i.e. DPOs) their contents will be merged together into a single data table. It is assumed that all the CTI data are of the same (event) type, e.g. Malware event data. Depending on the data volume and the database supported by the Legacy Analytics Engine, a C3ISP Consumer can choose one of the following systems to instantiate a VDL:

- *MySQL*: this is a conventional relational database system that employs SQL as its query language.
- *Apache Hadoop Distributed File System (HDFS)*: this is a scalable distributed Java-based file system that can be used to store huge amount of data on a cluster of servers. Apache Impala [63] is used to provide the SQL-like interface to query the data.

5.5. IAI API

The **IAI API** is the C3ISP entry point to execute analytics services. The analytics services can be C3ISP analytics services (full integrated with the C3ISP Framework and its DSA policies, see Section 5.1 and 8.3) and Legacy analytics services (that access C3ISP-sanitised data – via DMOs – after authorised DSA policies, see 5.3 and 8.4).

It exposes methods for invoking analytics services on a *single DPO* data or a *set of DPOs* shared and provided by the Prosumer(s) via ISI. For this reason, the API includes:

- A method for invoking an analytics service (*runAnalytics*), which will receive as input the service analytics name (in the form of a unique identifier) and the DPO identifier (DPO_ID) to be involved in the analytics. The analytics name is an identifier specifying an analytics service available in the analytics engine;
- A method for processing multiple DPOs, i.e. two or more DPOs can be involved as input to an analytics service. The method is analogous to the previous one, but it receives as input a list of DPO_ID.

When dealing with multiple DPOs, we assume that the DSA associated to each CTI is the same: this is the reasonable scenario where several Prosumers share CTI data, by using their agreed DSA, and that want to perform analytics on them. However, the DSA Policy Merger component introduced in Section 6.3 may support different scenarios.

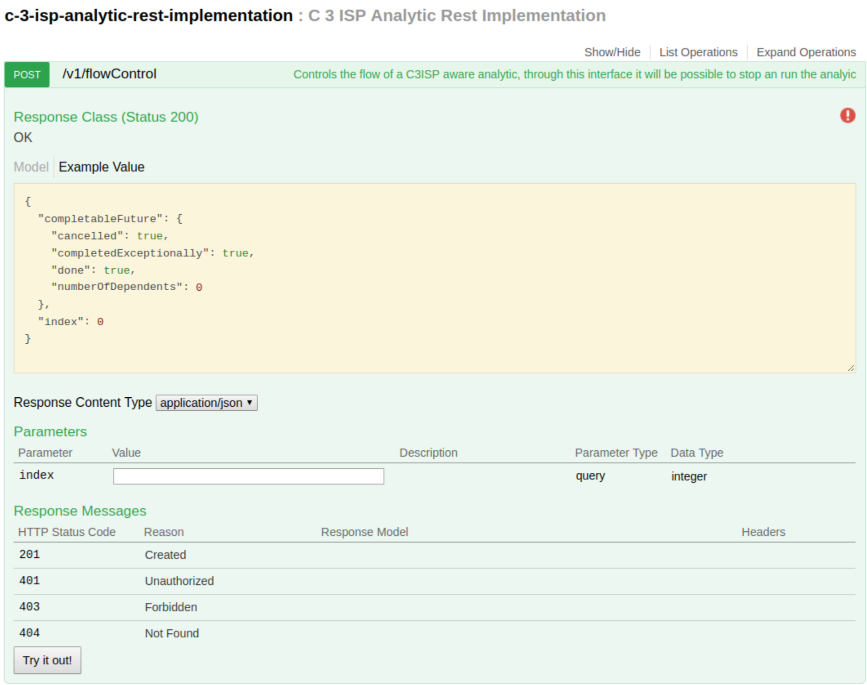
Depending on the kind of analytics service, the IAI API creates a DLB (for C3ISP-aware analytics) or a VDL (for the Legacy Analytics Service). The Data Lake creation process obey to the DSA policies associated to the CTI data, by performing *Read DPO* operations against the ISI and so evaluating the DSA policies (e.g. for authorisation of CTI data access), including the execution of defined DMOs that can sanitise the data before the analytics run.

At M34 the IAI API offers a set of analytics aimed at addressing analytics requests from the four pilots. In addition to the already available **runAnalytics** service, a **stopAnalytics** function has been added to be able to revoke the execution of a specific analytics at any time. The APIs accept as input the *service name* and all the parameters that are specific for the invoked analytics. The list of available analytics services from the IAI API is the following:

- **detectDGA**: takes domain-name logs and checks if they are DGA (Domain Generated Algorithm) using a third-party algorithm;

- **matchDGA**: takes domain-name logs and checks if they are DGA using a public source of known DGAs;
- **detectBruteForce**: takes SSH logs and highlights brute-force attacks considering the number of failed logins and the fraction of failed and accepted logins on the SSH protocol;
- **detectBlackMarket:** this analytics has been designed to detect URL of App markets distributing malicious Android apps. The analytics takes as input a web URL, then a set of attributes are extracted to be analysed by a classifier. The output is the probability that the URL is related to a malicious market;
- **findAttackingHosts**: takes CTI data (firewall logs etc.) and counts the number of source IP addresses in the log. The administrator then determines the threshold above which the IP addresses are thought to be malicious;
- **spamEmailClassify**: analytics that takes as input a set of spam emails in *eml* format and returns the class assigned to each email, based on the spammer goal (e.g. phishing, advertisement, etc.);
- **spamEmailClusterer**: analytics that takes as input a set of spam emails and divides them in clusters based on structural similarity;
- **spamEmailDetect**: takes as input one or more *eml* files and separates them in genuine emails (Ham) and unsolicited ones (Spam) sets;
- **correlateVulnerabilitytoAttack**: takes as input a software product name (i.e. app, OS version, Firmware, etc.), extracts the related CPE and correlates this information with different databases (e.g., CPE, CWE, CVE, CAPEC, etc.) to extract a full record of vulnerabilities, attack-patterns, specific attacks and recommendations for attack mitigation or their prevention. See D8.3 for further details;
- **findMaliciousHost**: analytics that takes as input a set of security logs such as IDS/IPS alerts, Malware alerts or Web Proxy logs in *csv* format and returns a prioritised list of malicious hosts based on specified criteria (e.g. frequencies of occurrence, commonness, threat level, etc.);
- **analyseDNSTraffic**: analytics that takes as input a set of DNS request logs in *csv* format and returns a list of domain names with suspicious network traffic behaviours (e.g. high frequency of requests);
- **findVulnerability**: analytics that searches in an archive of CVEs, stored in the DPOS, the keyword passed as a parameter;
- **findMalware**: analytics that searches, in an archive stored in the DPOS, information about the malware related to the keyword;
- **connToMaliciousHost**: this is the FHE analytics described in Section 5.1.1;
- **legacyAnalytics**: this is the analytics described in Section 5.3.

More analytics can be added at any time in the C3ISP IAI. To do so, the IAI API offers an SDK for C3ISP-aware analytics developers. The SDK is provided as a project which can be imported in major IDEs (such as Eclipse[14]) with the template of functions to be included in any new analytics developed for the C3ISP framework. In particular, each analytic should present the following functions:



c-3-isp-analytic-rest-implementation : C 3 ISP Analytic Rest Implementation

/v1/flowControl

Controls the flow of a C3ISP aware analytic, through this interface it will be possible to stop an run the analyic

Response Class (Status 200)

OK

```
{
  "completableFuture": {
    "cancelled": true,
    "completedExceptionally": true,
    "done": true,
    "numberOfDependents": 0
  },
  "index": 0
}
```

Response Content Type application/json ▾

Parameters

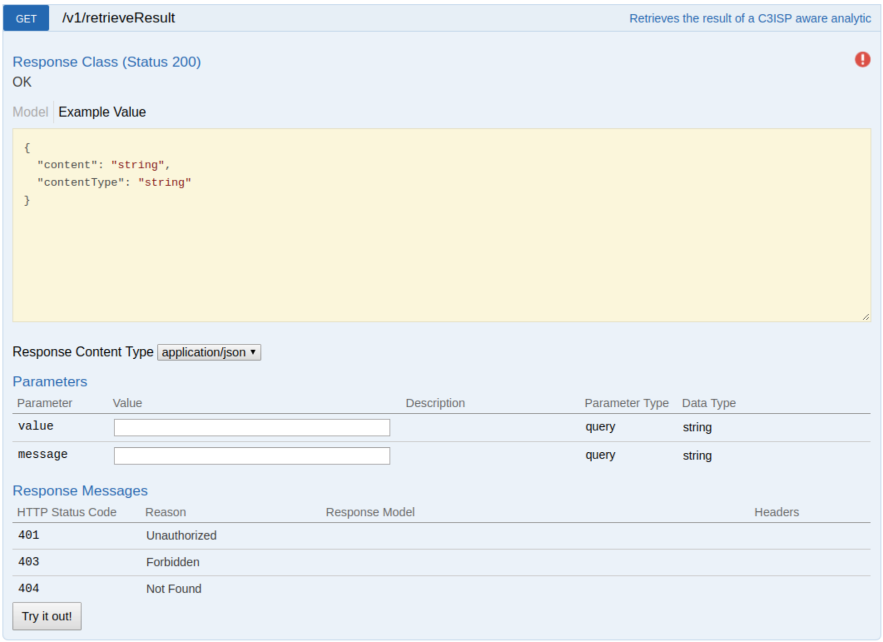
Parameter	Value	Description	Parameter Type	Data Type
index			query	integer

Response Messages

HTTP Status Code	Reason	Response Model	Headers
201	Created		
401	Unauthorized		
403	Forbidden		
404	Not Found		

Figure 19: C3ISP Analytics SDK – flowControl

The **flowControl** API allows the developer to define the behaviour to correctly stop the analytics on request of the Service Usage Control Adapter (see 5.2).



/v1/retrieveResult

Retrieves the result of a C3ISP aware analytic

Response Class (Status 200)

OK

```
{
  "content": "string",
  "contentType": "string"
}
```

Response Content Type application/json ▾

Parameters

Parameter	Value	Description	Parameter Type	Data Type
value			query	string
message			query	string

Response Messages

HTTP Status Code	Reason	Response Model	Headers
401	Unauthorized		
403	Forbidden		
404	Not Found		

Figure 20: C3ISP Analytics SDK – retrieveResult

[14] https://www.eclipse.org/

The **retrieveResult** allows the developer to define the correct way to collect the results of the analytics. This process changes depending on the data type and file system on which the analytics operates (Filesystem, MySQL, HDFS).

6. Subsystem: DSA Manager

The DSA Manager is an autonomous subsystem of the C3ISP Framework appointed to provide services for the definition of policies in the DSAs, creation, storage and management of DSAs to the Prosumers.

This section describes the design changes made to the DSA Manager between M24 and M34.

The DSA Manager is made up of the following components, described in details in the next sections:

- DSA Editor: to write the sharing rules that can be understood by humans;
- DSA Mapper: to translate the sharing rules to an enforceable language that can be processed by a machine;
- DSA Policy Merger: to create the DSA used when the C3ISP-aware analytics produce their outcome (i.e. analytics result);
- DSA Store API: to manage the DSAs from the other components;
- DSA Manager Gateway: to provide DSA services to other C3ISP subsystems;
- DSA Store: to persist the DSAs on a storage area.

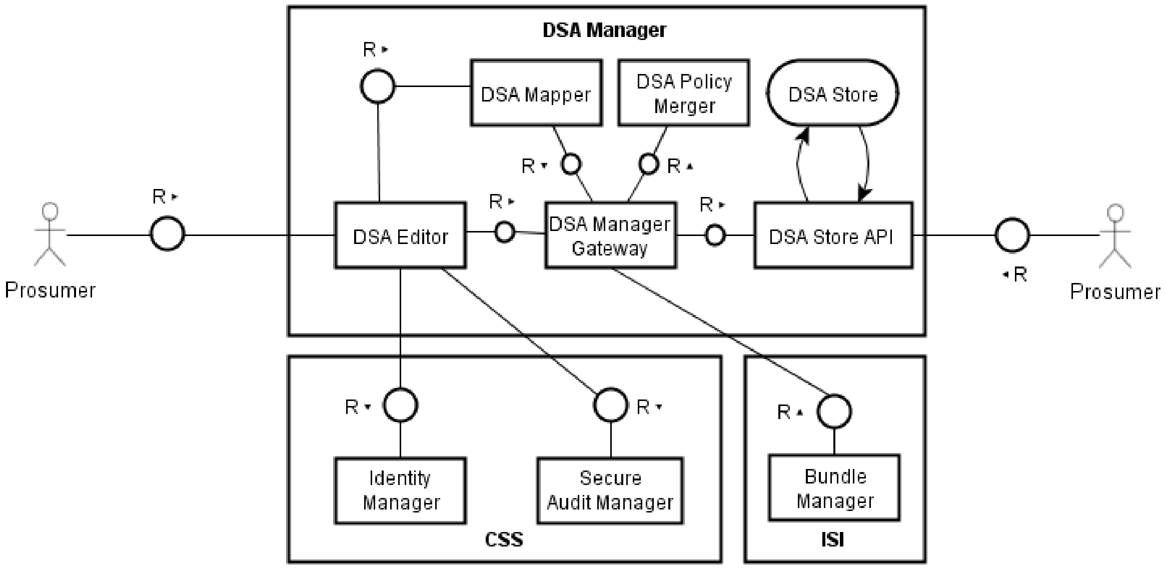


Figure 21: DSA Manager

The DSA Manager interacts with external clients:

- the Prosumer, which would use the services exposed by the DSA Editor and (optionally, if needed) DSA search capabilities exposed by the DSA Store API;
- the Information Sharing Infrastructure (ISI), that needs to use the DSA Manager Gateway for supporting the data sharing among the Prosumers using the C3ISP Framework;
- the Common Security Services (CSS) for secure auditing of its activities and for identity management.

6.1. DSA Editor

The design of the DSA Editor has not been changed in the last period, even if there were several small changes and bug fixes to the software artefacts. The DSA Editor is reported below for completeness, as documented in D7.3:

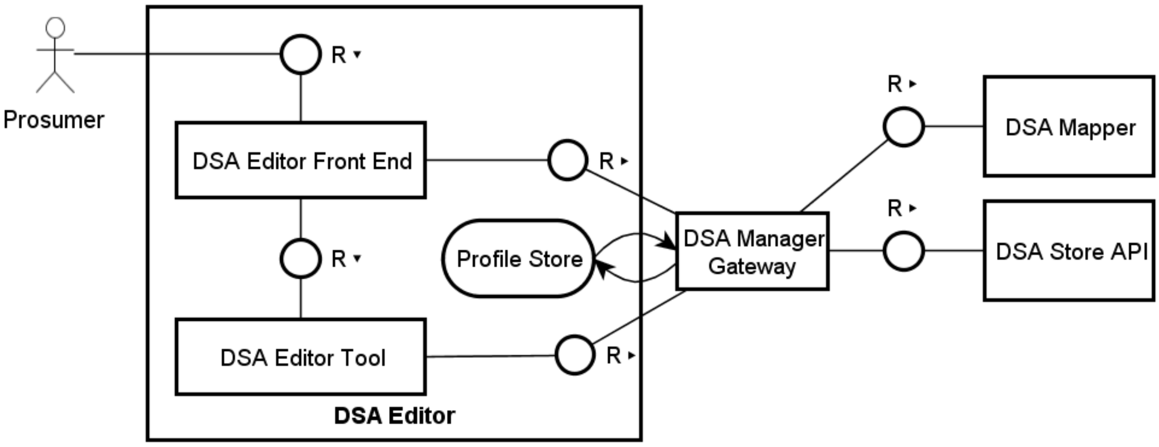


Figure 22: DSA Editor

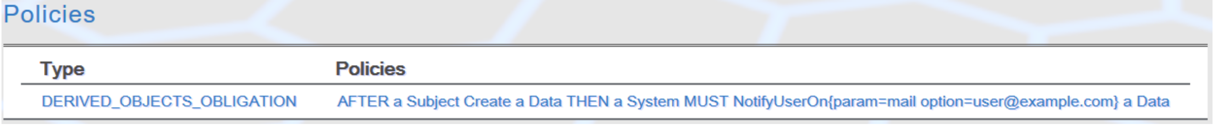
The component is made of the following modules:

- DSA Editor Front End: this is the entry point that allows Prosumers to access the available DSAs via a role-based access control system, where each user can work on his/her own DSAs without impacting other users. It interfaces with the DSA Editor Tool to allow the creation of DSAs and with the DSA Manager Gateway to manage the users' Profile Store, including some DSAs functions (in particular the "revoke DSA" and "delete DSA" operations);
- DSA Editor Tool: a user-friendly and easy to use web application that allows creating both DSA Templates (a generic DSA with predefined useful policies for a certain context that must be instanced to be used) and of DSA instances (a DSA specifically created from a DSA Template by adding rules for refinement that can be directly used for enforcement): see D8.1 for more on DSA and DSA Templates;
- Profile Store: this is a repository that hosts the users that can access the other modules with their roles and a reference to the DSAs they have written.

The DSA Editor Front End and the DSA Editor Tool have been integrated with the Configuration Server (see 2.1) to make it easy to switch between different environments (e.g. Test Bed or Production). Settings like the list of vocabularies and the parties of the agreement are now easily configurable, as well as some other technical configuration (e.g. IP addresses of database, etc.).

The DSA XML schema has been extended to include a new field used by the DSA Editor called 'application-domain'. This field must be specified when creating a new DSA Template and support multi-tenancy scenarios of the DSA Adapter, in particular the lookup of attributes used in policy evaluation from different data sources. This means that depending on DSA attached to the data, the DSA policy evaluation engine will fetch the required attributes (e.g. group membership) from an 'application-domain' specific source: e.g. for ISP Pilot (hence 'application-domain=ISP') it will contact a relational database, while for SME Pilot it will contact an LDAP service.

The DSA Editor now supports specifying a policy for expressing a notification by email. The following screenshot shows a sample DSA policy for this:



Policies

Type	Policies
DERIVED_OBJECTS_OBLIGATION	AFTER a Subject Create a Data THEN a System MUST NotifyUserOn{param=mail option=user@example.com} a Data

Figure 23: DSA notification policy

Finally several improvements to the GUI has been made, which are described in D8.3.

A detailed description about the usage and appearance of the DSA Editor is contained in D8.2.

6.2. DSA Mapper

The **DSA Mapper** is the component in charge of taking in input the DSA XML file returned by the DSA Editor component (i.e. the file containing all the DSA policies) and both modifying it by injecting XACML translation of each rule directly into the XML file and producing a UPOL policy that can be directly managed by the enforcer engine (i.e. DSA Adapter).

The current status of the DSA Mapper functionalities maturation and how they work are described in D8.3.

The improvements of the DSA Mapper within the third year of the project are mainly related to the management of **obligation rules**, with a particular eye to *Data Manipulation Operations (DMOs)* and the *notify* operation, and to the refined functionalities of the enforcement engine to manage policies on *derived data* and on *other data*.

Hence, the architecture of the DSA Mapper component is still the same as that reported in D7.3, but its translating capabilities are enhanced in terms of accepted vocabularies and kind of policies that can be mapped from CNL to UPOL/XACML.

As described more in detail in D8.3 Section 4.2, the DSA Mapper has been improved to be able to manage the new capabilities of the DSA Editor. In particular,

- *Support definition of Data Manipulation Operations* (DMOs) *(pre/post-processing rules)*: the DSA Mapper is able to recognise DMO policies and to translate them as obligations policies. When the mapper processes an Obligation rule, it inspects in the XML tags of the obligation rule the attribute *statementInfo* in which, in case in the rule a DMO is present, there are parameters and options of the DMO (see D8.2 Section 4.2.2.2). If it is present, the mapper stores in a Hashset the name of the DMO. To avoid misunderstanding about which one is the target action and the DMO in the expression and to avoid having a “hardcoded rule”, the DSA Mapper searches for all sets for bracket in the expression, extrapolates the inner part and then checks if the extrapolate string is present in the Hashset or not. If it is not present it must be the *Action* (e.g. ‘create’), if it is present then it is a *DMO Action*. In this way the action is targeted correctly, and it is flexible enough to be able to support multiple DMOs since the Hashset can store multiple values.

- *Translate policies on Analytics Results*: policies related to data derived from analytics functions (i.e. the analytics results) are in a separate section of the DSA XML and the DSA Mapper translates them in a separate section too. Note that the implementation of the DSA Mapper manages policies on first order derived data, i.e., on data generated as results of the application of analytics on original data, the one the DSA is related to. Furthermore, the *buildUPOL* function (see D7.3) can be called by the DSA Policy Merger component (see 6.3) to create the UPOL policy that has to be enforced on the derived data.

6.3. DSA Policy Merger

The result obtained from the execution of a C3ISP-aware analytics is stored on the ISI subsystem, in order to be subsequently read by the C3ISP Prosumer and possibly exploited in further analytics. This means that to create the DPO representing the analytics result it is necessary to choose a DSA to regulate its subsequent usage. Obviously, all the subjects whose data have been used to produce the result should be enabled to express their preferences concerning the usage of such results. Consequently, the DSA paired with the result should be *derived* from the DSAs of the DPOs which have been used to execute the analytics. To this aim, we introduced in the DSA Manager Subsystem the **DSA Policy Merger**, which is the component that combines the DSA policies coming from different DPOs in order to create a new DSA which "merges" the rules of all DSAs, in a so called *Analytics Result Policy*, that will be used when creating the analytics result. This component will be invoked when, after having performed an analytics on different DPOs, there is the need to define the DSA policy that the result DPO should have. To allow to define the *derived policy*, the DSA of each DPO includes a dedicated section containing DSA Policies for the result (called "Analytics Result Policies", see DSA Editor description in D7.3). Since there could be many DPOs involved in an analytics, a component which is in charge of merging all the related policies to produce the resulting one is needed. The strategy exploited to define the resulting policy could depend on the specific Pilot. One possible strategy could produce the Analytics Result Policy by operating a conjunction of all the policies of the original DPOs. In its simplest implementation, the DSA Policy Merger takes the list of DSAs, assumes that all DSAs are the same (simplistic assumption) and creates a *new* DSA where:

- The DSA policies contained in the "Analytics Result Policies" section (if any) are set into the "Parties Policies" (i.e. the main DSA policies section);
- If there are no "Analytics Result Policies", the "Parties Policies" are set in the new DSA in the same section.

Other merger algorithms could be defined and the component can be easily replaced because it is implemented as a RESTful micro-service with a simple interface.

6.4. DSA Store API

The **DSA Store API** is the interface to the DSA Store component (in such a way to abstract from its underlying implementation technology), and it provides functions for managing the storage of DSAs. The DSA Store is used by the ISI to retrieve DSAs or by the DSA Editor when it needs to create or update DSAs.

The DSA Store API supports the following interface:

- **Create DSA**: used to persist a DSA in the DSA Store. The API returns a unique DSA identifier;
- **Read DSA**: used to retrieve a DSA by its DSA identifier;
- **Update DSA**: used to modify the content of an already existing DSA, by its identifier;
- **Delete DSA**: used to delete a DSA, by its identifier;
- **Search DSA**: given a JSON-based search string (see specification in D5.3), query the DSA metadata repository, and returns a set of metadata entries corresponding to the matching DSAs (alternatively, a set of DSA IDs, depending on an input flag). DSA metadata is stored in a JSON document, and contains a set of DSA fields extracted from the DSA XML files.

The DSA Store API has been integrated with the Configuration Server described in Section 2.1.

6.5. DSA Manager Gateway

The DSA Manager Gateway provides a set of APIs that other components can use to interact with the DSA Mapper and the DSA Store API at more logical/use case level. In particular, it provides an interface that contains more business logic than the DSA Store API, which is a direct CRUD interface to the DSA Store (e.g. the DSA Manager Gateway can return parts of the DSA, instead of the whole DSA returned by the DSA Store API).

The interface is stable, but has been enhanced to include a new method:

- **Create Merged DSA**, given several DSAs identifiers, it provides them to the DSA Policy Merger component (see 6.3) which combines and returns a single DSA. This is related to the use case of analytics running over several DPOs shared under possibly different DSAs. See 6.3 for more information.

Further, the DSA Manager Gateway has been integrated with the Configuration Server (see 2.1) for easing deployment to different execution environment (e.g. Test Bed and Production).

6.6. DSA Store

The **DSA Store** is a database that acts as a repository where the DSAs are stored and consumed by the DSA Store API. It is a stable component of the architecture that has not gone under change in the last period. It is implemented with an instance of MongoDB [56], a NoSQL database. D7.3 reported the implementation details.

7. Subsystem: CSS – Common Security Services

7.1. Identity Manager

The **Identity Manager** aims at identifying entities and storing authorisation information within C3ISP to be used by Pilots and C3ISP consortium members. It has been split into two services: (a) LDAP for keeping the user repository, and (b) OIDC for providing a standard authentication system, as shown in the following diagram:

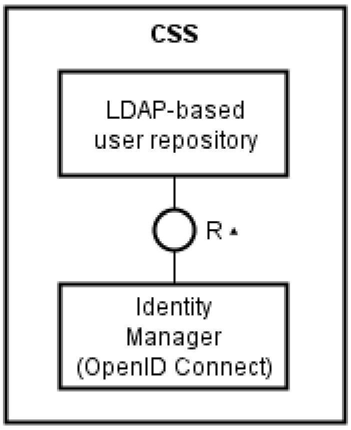


Figure 24: Identity Manager services

The next sections describe in more detail the LDAP and OIDC services.

7.1.1. LDAP

The LDAP service provided by C3ISP is very stable and only few small updates are worth mentioning.

OpenLDAP was used as user repository for the OIDC server (see next section).

LDAP is also used as a PIP (Policy Information Point) to collect user data for DSA policy evaluation: this includes attributes for organisation specification and group membership. These are used in C3ISP Pilots DSA policies.

Finally, the C3ISP private Docker Registry described in Section 2.1 was integrated with the LDAP system for authentication. Only authenticated LDAP users can access the C3ISP Docker images.

7.1.2. OIDC

OpenID Connect (OIDC [79]) is a federated login system that allows a user to access a set of federated resources by using the same username and password for all the resources, as well as providing single sign on (SSO) to the user. With SSO, once the user logs into the first federated resource, he or she does not need to login to any of the subsequent ones, as the OIDC server takes care of this. In order to add OIDC to C3ISP, the following changes have been made to the C3ISP infrastructure:

1. The addition of an OIDC server. This is the service that the user is redirected to by all the C3ISP components, in order to login, prior to accessing the C3ISP component;
2. Connecting the OIDC server to the LDAP server described above, so that the user can use the same username and password combination after federated login is added;
3. The modification of the authentication mechanism supported by each of the C3ISP components from LDAP login to OIDC federated login;

4. Upgrading all the C3ISP infrastructure components to use Springboot 2.0.*, Springfox 2.9.2 and Oauth2 Autoconfigure in order to support federated login.

We employed MITREid Connect (OpenID-Connect-Java-Spring-Server[15]) as an open source instantiation of an OIDC server. Also, we used its LDAP overlay[16] to support authenticating with the OIDC server by using accounts registered in an LDAP server. Based on that, we have developed an OIDC server supporting LDAP authentication.

A user guide was written informing all the consortium partners how to add OIDC login to their particular C3ISP component (DSA Manager, central ISI, local ISI, Gateway, IAI and portal). It describes the libraries to add to the component, their configuration to work as a resource server (component that asks for authentication) or a client (component that accesses a resource server), as well as how to get information about the requestor (e.g. access information of the authenticated user via the OIDC protocol).

7.2. Key and Encryption Manager

The Key and Encryption Manager (K&E Manager) is responsible for key management and encryption services needed for ensuring the confidentiality data throughout the C3ISP Framework. In D7.3 we selected HashiCorp Vault[17] as an open source solution to use for basing the development the K&E Manager services.

With respect to the C3ISP high-level architecture reported in Section 2, the Key & Encryption Manager interacts with the C3ISP subsystems and components as shown in the following figure:

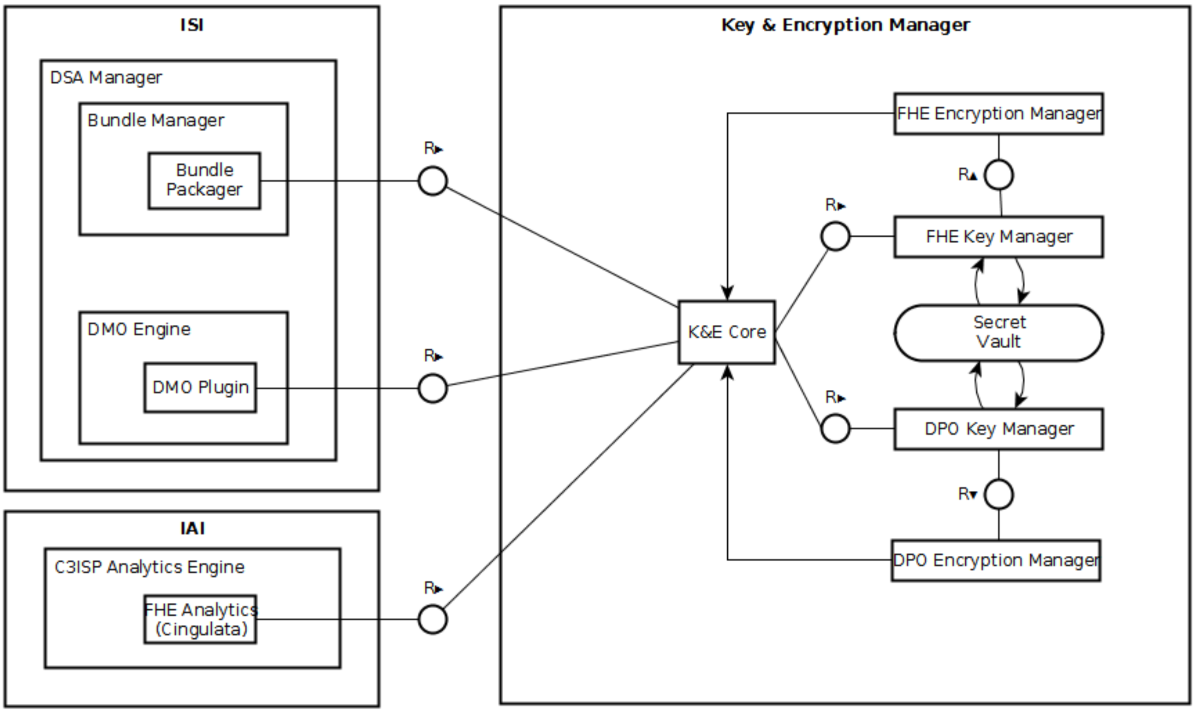


Figure 25: Key & Encryption manager architecture

[15] https://github.com/mitreid-connect/OpenID-Connect-Java-Spring-Server

[16] https://github.com/mitreid-connect/ldap-openid-connect-server

[17] HashiCorp Vault, https://www.vaultproject.io/

7.2.1. K&E Core

K&E Core is the frontend for the services of K&E Manager which allows dispatching the requests from the ISI (specifically the Bundle Packager or the DMO Plugin) and from the IAI (specifically the FHE Analytics). At M34 the K&E Core is completed; with respect to the previous deliverable we completed also the following task:

Specification K&E Core API	Description
Integration with ISI – DMO Plugin for FHE encryption	Selection of the correct keys and parameters for using FHE encryption service to encrypt data.

7.2.2. Key Management & Encryption/Decryption

The key management for both DPO and FHE was stable in the past period and was already completed when we released D7.3.

At M34, we added a new feature for data encryption, and now there are two options:

- The data is directly encrypted in FHE format;
- The data is encrypted with Kreyvium which is used for multiple encryption schemes.

In the BLACK_LIST analytics functionality (see 5.1.1), we did not choose the multiple encryption schemes for CTI data. The reason is due to its performance issue, but still the option for using this functionality is available. In fact, when directly using FHE encryption, the multiplicative depth in the Boolean circuit is clearly decreased from 12 to 5. Hence, it is very fast as FHE technology. In the benchmark we did with a *CPU W3520 @ 2.67GHz with 8 cores*, when using the multiple encryption schemes with Kreyvium, checking one IPv4 in a list of 100 IPv4 takes around 420s (6 minutes), whereas in the scenarios of encrypting IPv4 directly with FHE, the analysis time is 178s (less than 3 minutes). Note that the FHE technologies are not used for real-time or online analysis, but for offline analysis, so the improvement is very important.

7.3. Secure Audit Manager

The **Secure Audit Manager** tracks all the critical operations performed by the C3ISP Framework (e.g. the policies evaluation and their enforcement results, analytics execution, etc.) in order to achieve the accountability requirement for the C3ISP Framework.

This component supports two distinct features for the auditing process:

- Inter-component auditing: traces the REST calls between the modules, components and subsystems. This helps assuring the right workflow is in place and seeing at each moment who is calling what;
- Internal-component auditing: audits critical events raised by a module or a component (e.g. policy evaluation, user attribute fetch, DPO bundle encryption/decryption, etc.). This assures accountability of actions for relevant software artefacts.

By combining both features, we can obtain a complete picture of what the C3ISP Framework is doing and ultimately provide an accountability track that can be useful to demonstrate that the Data Sharing Agreement is being respected.

The designed architecture has not been updated and has been implemented. Next figure reports the design presented in D7.3:

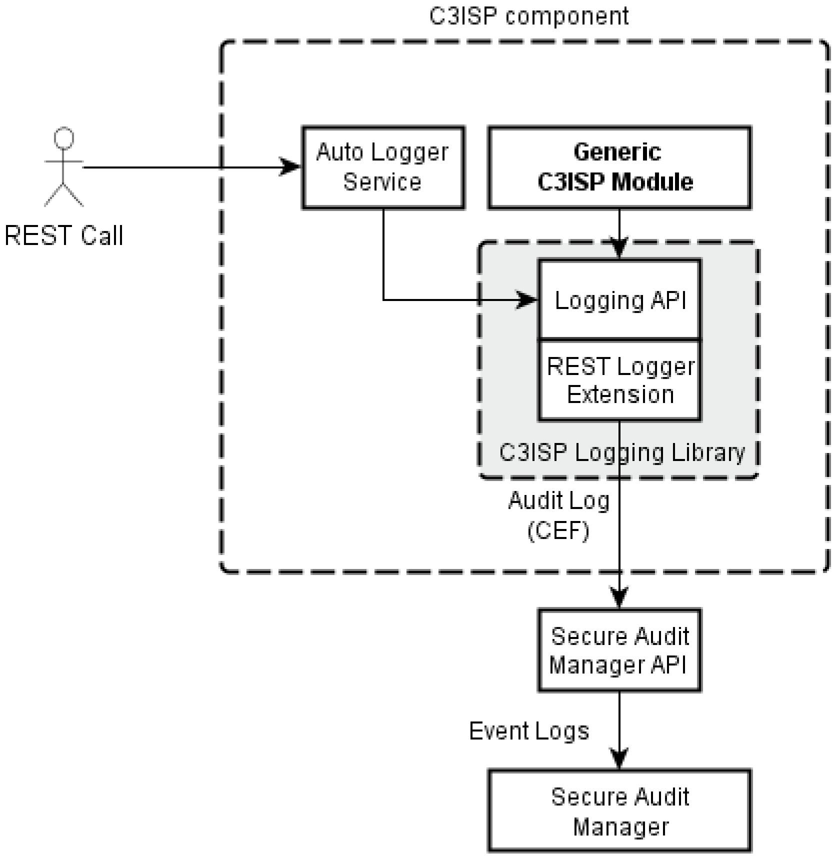
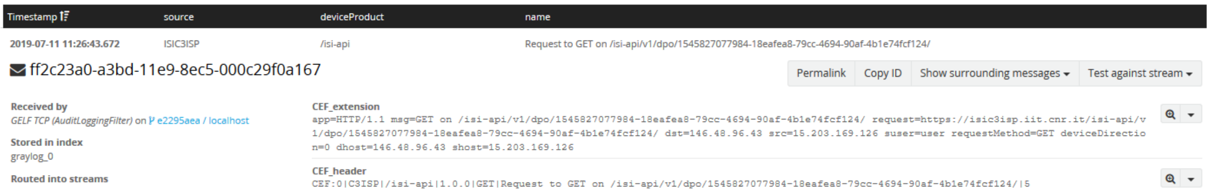


Figure 26: Secure Audit Manager architecture

The "REST call" is any request to a C3ISP component (developed with the micro-services paradigm).

The "Auto Logger Service" is a component (implemented with a Servlet filter running along the micro-service's application server) that automatically traces received REST requests and traces them via a "C3ISP Logging Library".

The Secure Audit Manager API abstracts the interface between the C3ISP Logging Library and the Audit Manager to allow using different Log Management products. The Auto Logger Service traces some basic logging information, with common attributes and values for all the components. Below an example from the Graylog[18] product that has been integrated:



Timestamp	source	deviceProduct	name
2019-07-11 11:26:43.672	ISIC3ISP	/isi-api	Request to GET on /isi-api/v1/dpo/1545827077984-18eafea8-79cc-4694-90af-4b1e74fcf124/

ff2c23a0-a3bd-11e9-8ec5-000c29f0a167

Received by
GELF TCP (AuditLoggingFilter) on e2295aea / localhost

Stored in index
graylog_0

Routed into streams

CEF_extension

```
app=HTTP/1.1 msg=GET on /isi-api/v1/dpo/1545827077984-18eafea8-79cc-4694-90af-4b1e74fcf124/ request=https://isic3isp.iit.cnr.it/isi-api/v
1/dpo/1545827077984-18eafea8-79cc-4694-90af-4b1e74fcf124/ dst=146.48.96.43 src=15.203.169.126 suser=user requestMethod=GET deviceDirectio
n=0 dhost=146.48.96.43 shost=15.203.169.126
```

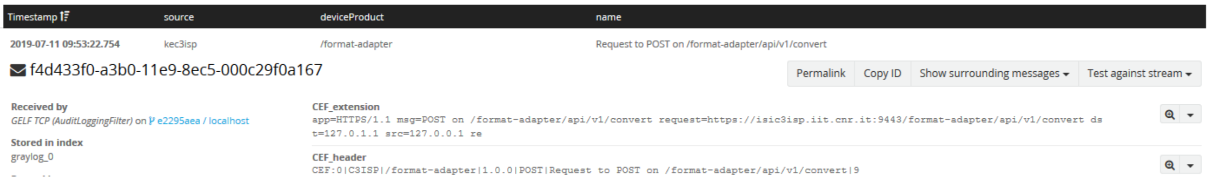
CEF_header

```
CEF:0|C3ISP|/isi-api|1.0.0|GET|Request to GET on /isi-api/v1/dpo/1545827077984-18eafea8-79cc-4694-90af-4b1e74fcf124/|5
```

Figure 27: Example of a logged ISI API readDPO request (Auto Logger Service)

[18] Graylog, https://www.graylog.org/

The “Generic C3ISP Module” is any of the C3ISP modules in the C3ISP architecture and supports internal-component auditing to trace their specific activity. For example, the Format Adapter, which is written in NodeJS (and so does not support the Auto Logger Service is implemented as a Java Servlet Filter), has been integrated in this way.



Timestamp	source	deviceProduct	name
2019-07-11 09:53:22.754	kec3isp	/format-adapter	Request to POST on /format-adapter/api/v1/convert

f4d433f0-a3b0-11e9-8ec5-000c29f0a167

Received by
GELF TCP (AuditLoggingFilter) on e2295aea / localhost

Stored in index
graylog_0

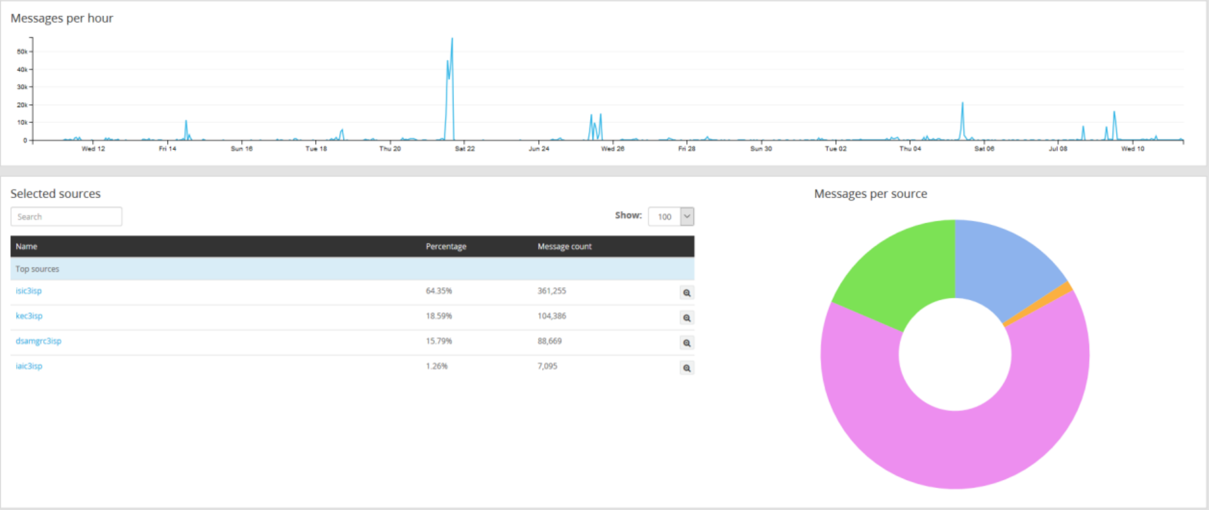
CEF_extension
```
app=HTTPS/1.1 msg=POST on /format-adapter/api/v1/convert request=https://isic3isp.iit.cnr.it:9443/format-adapter/api/v1/convert ds
t=127.0.1.1 src=127.0.0.1 re
```

CEF_header
```
CEF:0|C3ISP|/format-adapter|1.0.0|POST|Request to POST on /format-adapter/api/v1/convert|9
```

Figure 28: Example of a logged Format Adapter request (Generic C3ISP Module)

As cited above, the selected Secure Audit Manager product is Graylog, an open source log management solution. The main reason for its selection is that we found it to be the only OSS software in this field to support LDAP central authentication. In fact, we configured Graylog to use the user base we defined on the C3ISP LDAP service (see 7.1.1).

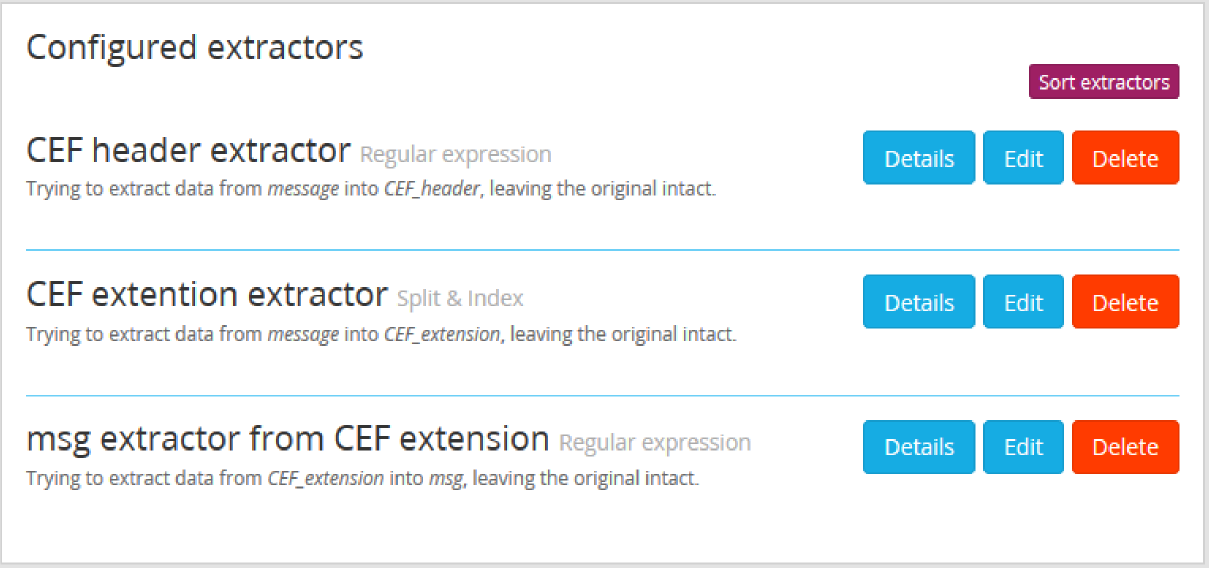
Figure below shows a summary screenshot presenting the log messages received from the different C3ISP subsystems in that last month:



Name	Percentage	Message count
Top sources		
isic3isp	64.35%	361,255
kec3isp	18.59%	104,386
dsamgrc3isp	15.79%	88,669
iaic3isp	1.26%	7,095

Figure 29: Statistics of C3ISP Framework usage per subsystem (colours represent logs from different components)

We also extended Graylog capabilities to support CEF message. All the C3ISP components sends CEF-compliant log messages, but Graylog did not support such standard. For this reason we developed a CEF parser for Graylog (called "Extractors") able to interpret it:


Configured extractors

CEF header extractor Regular expression
Trying to extract data from *message* into *CEF_header*, leaving the original intact.

CEF extention extractor Split & Index
Trying to extract data from *message* into *CEF_extension*, leaving the original intact.

msg extractor from CEF extension Regular expression
Trying to extract data from *CEF_extension* into *msg*, leaving the original intact.

Figure 30: CEF Extractors for Graylog

8. Data Flow Diagrams

This section describes updated data flows at M34, in particular those related to the Move DPO operation and the analytics service invocation.

In particular, the integration with the OpenID Connect (OIDC, Section 7.1.2) added a preliminary interaction before using all the APIs with respect to authentication.

8.1. Create, Read, Delete DPO

The functionalities to create a DPO starting from a CTI, reading it in the clear (read DPO) and removing it (delete DPO) have already been designed (and also implemented) in their final form in D7.3. It is worth recalling here, that all these functions evaluate the associated DSA policies and perform the actions of specified DMOs. Basically these are the functions used for CTI data sharing.

8.2. Move DPO

The Move DPO operation involves two ISIs, for example, a Local ISI where data are sanitised on-premises thanks to the application of a DMO, and a Central ISI connected to an IAI to compute analytics on the sanitised data.

Each ISI exposes two methods:

- The **Move DPO operation**, to be called at the sender endpoint (in our example, the Local ISI). It receives as input a DPO_ID and a destination address. Its implementation checks whether the request can be authorised (by using the DSA Policy on the "*Move*" action), then retrieves the contents of the DPO (as a standard Read operation of the ISI API) and calls the Transfer DPO method on the destination ISI (see next);
- The **Transfer DPO operation**, called during the Move DPO operation automatically by the sender ISI (e.g. Local ISI) to the recipient ISI (e.g. Central ISI). Its implementation checks if the transfer operation can be authorised, then behaves like a Create DPO, returning a *new* DPO_ID for the submitted content.

The following sequence diagram depicts the Move DPO behaviour:

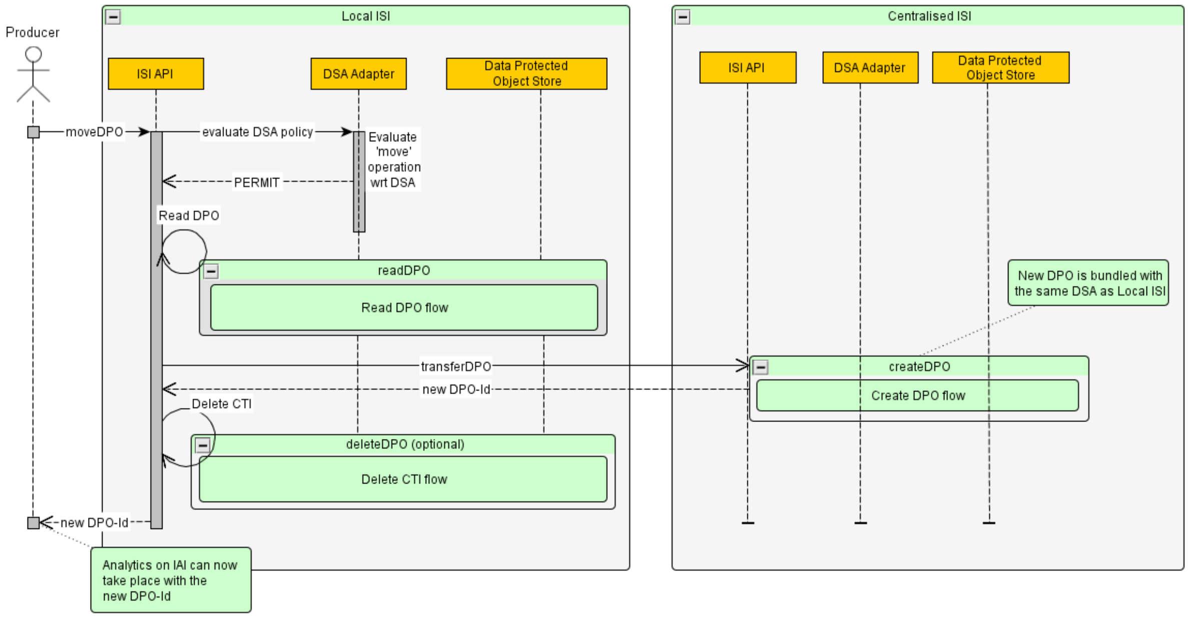


Figure 31: Move DPO

8.3. *Invoke C3ISP analytics service*

This flow reports the sequence diagram to detail the invocation of a generic C3ISP-aware analytics service on a DPO (or list of DPOs), given the service name and the DPO_ID (or DPO_IDs).

With respect to what was reported in D7.3, it includes the role of the DSA Policy Merger (see 6.3). It is only reported in the comments, just not to make the diagram more complex, but the DSA Policy Merger belongs to the DSA Manager subsystem, with which the ISI interacts with.

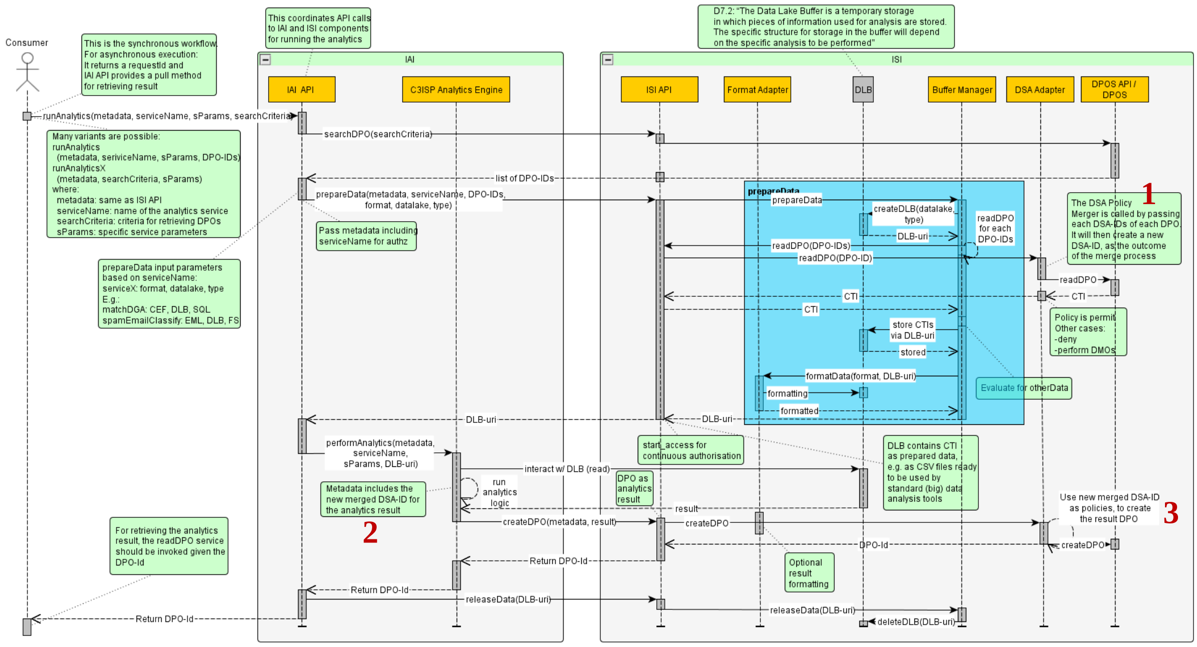

Consumer
This is the synchronous workflow.
For asynchronous execution:
It returns a requestId and
IAI API provides a pull method
for retrieving result
This coordinates API calls
to IAI and ISI components
for running the analytics
IAI
IAI API
C3ISP Analytics Engine
D7.2: "The Data Lake Buffer is a temporary storage
in which pieces of information used for analysis are stored.
The specific structure for storage in the buffer will depend
on the specific analysis to be performed"
ISI
ISI API
Format Adapter
DLB
Buffer Manager
DSA Adapter
DPOS API / DPOS
runAnalytics(metadata, serviceName, sParams, searchCriteria)
Many variants are possible:
runAnalytics
(metadata, seriviceName, sParams, DPO-IDs)
runAnalyticsX
(metadata, searchCriteria, sParams)
where:
metadata: same as ISI API
serviceName: name of the analytics service
searchCriteria: criteria for retrieving DPOs
sParams: specific service parameters
searchDPO(searchCriteria)
list of DPO-IDs
prepareData(metadata, serviceName, DPO-IDs, format, datalake, type)
Pass metadata including serviceName for authz
prepareData input parameters
based on serviceName:
serviceX: format, datalake, type
E.g.:
matchDGA: CEF, DLB, SQL
spamEmailClassify: EML, DLB, FS
prepareData
prepareData
createDLB(datalake, type)
DLB-uri
readDPO for each DPO-IDs
readDPO(DPO-IDs)
readDPO(DPO-ID)
1
The DSA Policy
Merger is called by passing
each DSA-IDs of each DPO.
It will then create a new
DSA-ID, as the outcome
of the merge process
readDPO
CTI
CTI
CTI
Policy is permit
Other cases:
-deny
-perform DMOs
store CTIs via DLB-uri
stored
formatData(format, DLB-uri)
formatting
formatted
Evaluate for otherData
DLB-uri
DLB-uri
start_access for
continuous authorisation
DLB contains CTI
as prepared data,
e.g. as CSV files ready
to be used by
standard (big) data
analysis tools
performAnalytics(metadata, serviceName, sParams, DLB-uri)
interact w/ DLB (read)
DPO as analytics result
run analytics logic
Metadata includes the
new merged DSA-ID for
the analytics result
2
result
createDPO(metadata, result)
createDPO
Use new merged DSA-ID
as policies, to create
the result DPO
3
createDPO
DPO-Id
Optional result formatting
For retrieving the analytics
result, the readDPO service
should be invoked given the
DPO-Id
Return DPO-Id
Return DPO-Id
releaseData(DLB-uri)
releaseData(DLB-uri)
deleteDLB(DLB-uri)
Return DPO-Id

Figure 32: Invoke C3ISP Analytics Service sequence diagram

In particular, after reading all the DPOs (if authorised to do so) needed to create the DLB, the DSA Adapter calls the DSA Policy Merger (**1**) passing the list of DSA-IDs associated to the DPOs. The Policy Merger creates a new DSA that contains the policies that will be used when creating the DPO representing the analytics result (see 6.3 for the merge logic used). The new DSA-ID is then transferred back to the IAI API, as metadata (**2**): once the analytics service has completed its job, the result is sent back to the C3ISP Analytics Engine which creates the result DPO by using the "new merged" DSA-ID (**3**). The analytics result becomes like any other DPO and can be used further (e.g. to be read).

8.3.1. Invoke FHE analytics

This flow describes the feature provided by the C3ISP Framework for invoking the FHE analytics for **identifying the presence of malicious connections on a set of shared CTI**. The check is based on a list of DPOs each containing a "malicious" IPv4 address encrypted in FHE format (see *blacklist algorithm* in 5.1.1).

Invoking FHE analytics service works in the same way as invoking a C3ISP analytics service (see previous section). The requested CTI data are transferred to a dedicated Data Lake Buffer (DLB) instance created by the prepareData API (see Buffer Manager in Section 4.3) which offers to the FHE analytics engine the available location (URI) for the data to process.

Before invoking FHE analytics, the Prosumers must encrypt the IPv4s in the CTI data, which for this use case are considered to be in CEF format. In this way, the C3ISP Framework allows working with CEF log files describing log connections generated by a standard firewall device. A specific CEF field contains the IPv4 address to check (called *destinationAddress* or *dst*; also possible on *sourceAddress* or *src*). In the BLACK LIST application, the field name used for extracting the IPv4 information is “*dst*”.

Here below it is the workflow to encrypt in FHE format the IPv4s contained in the shared CTI data:

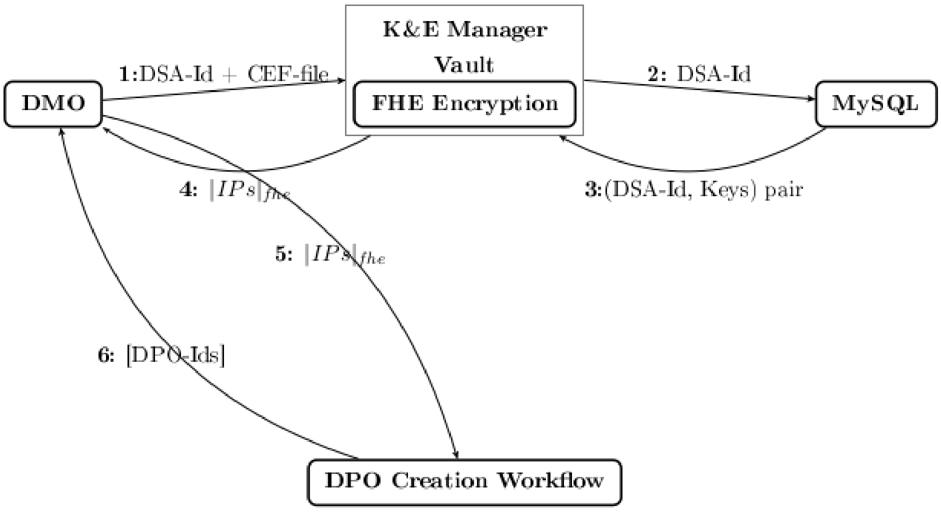
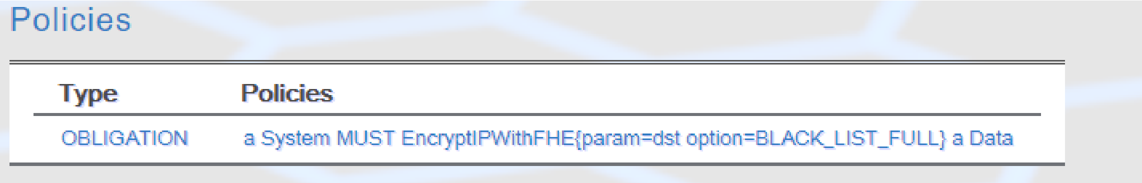


Figure 33: Workflow for encryption of a CEF file with FHE. [.] notation indicates an array of objects

When the Prosumer sends a request to share the CEF file via **ISI**, a specifically defined DSA policy triggers the DMO Engine (1) to handle this request by redirecting it to FHE Encryption service. Encryption key stored on the MySQL DB is used by the Vault service (2, 3) for this process. When the encryption is done, FHE Encryption service sends back to the DMO Engine several *zip files*, each containing 32 cipher-texts files which describe a FHE-encrypted IPv4 (4). Hence, the DMO Engine allows continuing the ISI’s DPO creation workflow to store all of them into the DPOS (5) and sends back to the Prosumer an array of DPO_IDs (6).



Policies

Type	Policies
OBLIGATION	a System MUST EncryptIPWithFHE{param=dst option=BLACK_LIST_FULL} a Data

Figure 34: A sample DSA Policy for applying the FHE-based DMO

When the Prosumer invokes the FHE analytics service for checking malicious connection via **IAI** (1), as shown in Figure 35, the array of DPO_IDs (2) is also sent to ISI in order to retrieve all the DPOs (3, 4) and prepare (5, 6) a Data Lake Buffer (DLB) in which the DPO objects for encrypted IPv4s are unzipped (but still FHE-encrypted) by the Buffer Manager. The FHE analytics service receives a DLB URI (7) to start its processing logic over it.

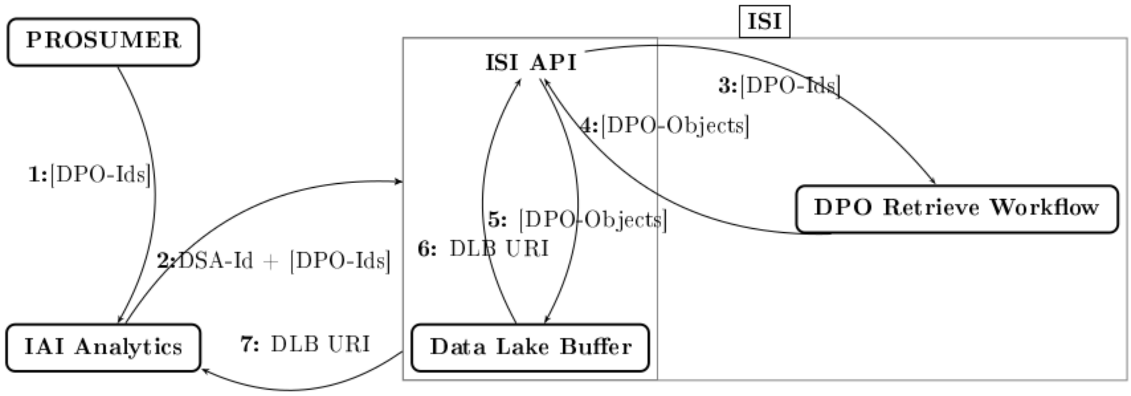


Figure 35: Workflow for preparing the DLB for FHE analytics. [.] notation indicates an array of objects

The URI path to the DLB is an input parameter (7) for the FHE malicious connection service to invoke the FHE API for this analysis purpose. The FHE results (8) of this analysis is stored as a DPO by invoking the DPO creation workflow of ISI (9, 10). The DPO_ID associated to this result is finally sent back to Prosumer (11). The workflow is described in the Figure below:

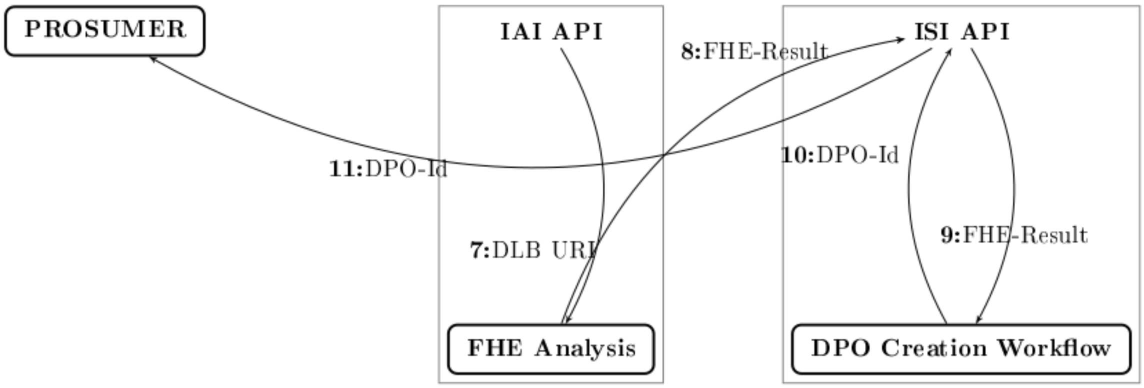


Figure 36: Workflow for running the FHE analytics

8.4. *Invoke Legacy analytics service*

This flow describes the feature provided by the C3ISP Framework for invoking the service of a Legacy Analytics Engine on shared CTI data (i.e. DPOs) based on the provided search criteria. Alternatively a list of DPO_IDs can also be provided as input parameter. Invoking legacy analytics service works much the same as invoking a C3ISP analytics service (see previous section). The main difference is that the requested CTI data are transferred to a dedicated Virtual Data Lake (VDL) instance and once the VDL is made available to the Legacy Analytics Engine the usage and distribution of those data may not be fully controlled by the C3ISP Framework, e.g. no continuous authorisation check is possible anymore. Furthermore, in case that the Legacy Analytics Engine does not support any API to invoke its functions programmatically, e.g. in case of a Visual Analytics tool with its own graphical user interface (BT Saturn), only a reference to the VDL instance (i.e. VDL-URI) is returned by the C3ISP Framework to the C3ISP Consumer, who in turn uses the URI to configure the Legacy Analytics Engine to access the data stored in the VDL. A C3ISP Consumer can be a software component of a Pilot. Otherwise the C3ISP Framework will pass on the results as produced by the Legacy Analytics Engine. Figure 37 shows the corresponding sequence diagram.

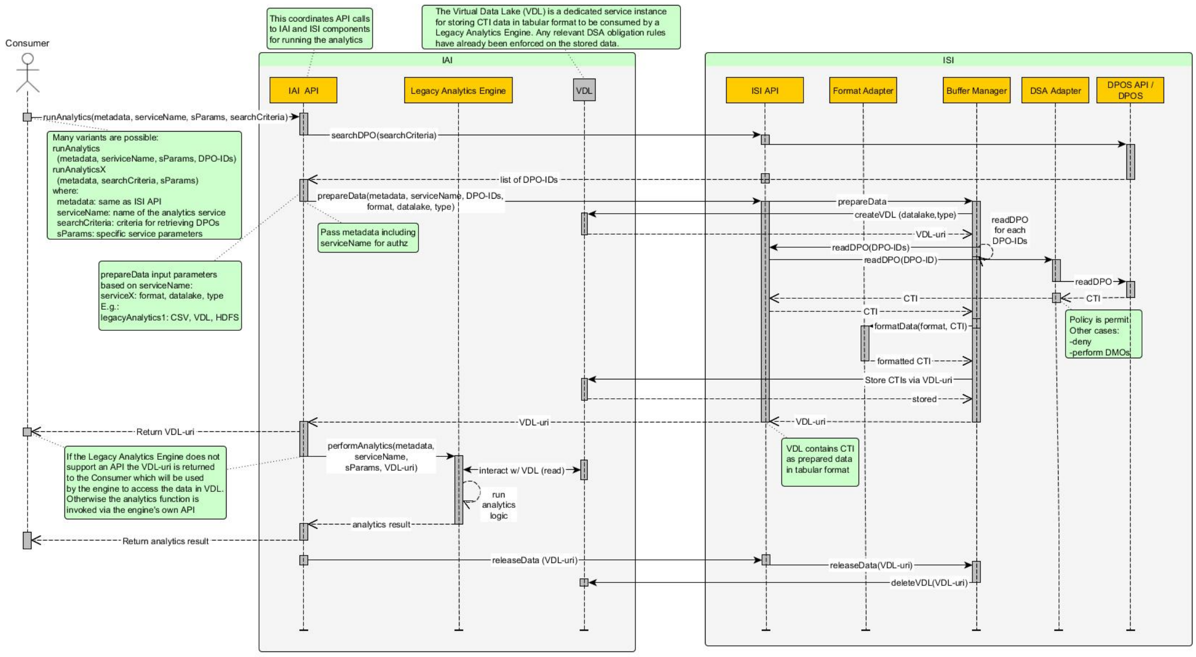

This coordinates API calls to IAI and ISI components for running the analytics
The Virtual Data Lake (VDL) is a dedicated service instance for storing CTI data in tabular format to be consumed by a Legacy Analytics Engine. Any relevant DSA obligation rules have already been enforced on the stored data.
Consumer
IAI
IAI API
Legacy Analytics Engine
VDL
ISI
ISI API
Format Adapter
Buffer Manager
DSA Adapter
DPOS API / DPOS
runAnalytics(metadata, serviceName, sParams, searchCriteria)
Many variants are possible:
runAnalytics
(metadata, seriviceName, sParams, DPO-IDs)
runAnalyticsX
(metadata, searchCriteria, sParams)
where:
metadata: same as ISI API
serviceName: name of the analytics service
searchCriteria: criteria for retrieving DPOs
sParams: specific service parameters
searchDPO(searchCriteria)
list of DPO-IDs
prepareData(metadata, serviceName, DPO-IDs, format, datalake, type)
Pass metadata including serviceName for authz
prepareData input parameters
based on serviceName:
serviceX: format, datalake, type
E.g.:
legacyAnalytics1: CSV, VDL, HDFS
prepareData
createVDL (datalake,type)
VDL-uri
readDPO for each DPO-IDs
readDPO(DPO-IDs)
readDPO(DPO-ID)
readDPO
CTI
CTI
CTI
Policy is permit
Other cases:
-deny
-perform DMOs
formatData(format, CTI)
formatted CTI
Store CTIs via VDL-uri
stored
VDL-uri
VDL-uri
VDL contains CTI as prepared data in tabular format
Return VDL-uri
If the Legacy Analytics Engine does not support an API the VDL-uri is returned to the Consumer which will be used by the engine to access the data in VDL. Otherwise the analytics function is invoked via the engine's own API
performAnalytics(metadata, serviceName, sParams, VDL-uri)
interact w/ VDL (read)
run analytics logic
analytics result
Return analytics result
releaseData (VDL-uri)
releaseData(VDL-uri)
deleteVDL(VDL-uri)

Figure 37: Invoke Legacy Analytics Service sequence diagram

9. Requirements mapping from D7.1

This section is dedicated to mapping the C3ISP reference architecture components with the framework requirements drawn in the D7.1 in order to check the coverage with them. We report *only* those requirements that were not satisfied when we released D7.2 and which are now addressed.

9.1. Data Sharing Requirements

Table 3 – Coverage of Data Sharing Requirements

ID	Requirements	Coverage
C3ISP-Fun-DS-007	C3ISP policies allow usage control of the shared data (i.e. define conditions to be continuously verify while the data is being consumed and after it has been accessed)	The DSA Adapter is the component of the ISI subsystem that enforces the usage control policies specified in the DSA on the shared data. The DSA Adapter checks that the DSA is satisfied for the duration of the analytics execution.
C3ISP-Fun-DS-009	C3ISP allows defining notifications (i.e. email, SNMP, etc.) that are triggered once the analytic service result is available (i.e. be able to encode this requirement in a policy rule). A notification mechanism could be email.	Notifications are encoded in DSA policies as obligations when the analysis finishes (the result is created). The Obligations Engine is a component of the DSA Adapter that enforces these obligations. The DSA Editor supports expressing notifications (see 6.1).
C3ISP-Fun-DS-010	C3ISP provides evidences (e.g. audit logs) of the compliance to the sharing policies enforcement	The Secure Audit Manager from Common Security Service subsystem is responsible for storing auditing records. The DSA Adapter traces to the Secure Audit Manager the DSA policies execution and outcomes.
C3ISP-Fun-DS-011	C3ISP policies allow writing “pre-processing rules” on the data to be shared, which are data manipulation operations performed before the data is shared with the other party(ies). These operations should include: (i) sanitisation operations (see 9.3) for minimising sensitive data exchange; (ii) encryption mechanisms	The pre-processing rules (also known as DMOs) are encoded in the DSA policies as obligations to be carried out before data access is granted. These rules, which are enforced by the DSA Adapter component of the ISI subsystem, modifies the data according to the specified algorithm before allowing the execution of the analytics. C3ISP supports the following DMOs: different types of anonymisation, FHE encryption, FPE encryption, pseudonymisation (see Section 4.1.5).
C3ISP-Fun-DS-012	C3ISP allows specifying policy rules to control the risk of data sharing (i.e. if a metrics is over a certain threshold, data can’t be shared or additional sanitisation measures must be applied before sharing)	The risk of data sharing can be encoded in the DSA policies. A specific Attribute Manager has been developed and is described in Section 4.1.3. This attribute represents the risk of storing a DPO in a given Local ISI system, and it is meant to regulate the data move operation.

9.2. *Data Analytics Requirements*

Table 4 – Coverage of Data Analytics Requirements

ID	Requirements	Coverage
C3ISP-Fun-DA-001	C3ISP allows defining policies (i.e. a set of rules) for data analytics operations to control what analysis can be performed on the Prosumer’s data	DSA Editor allows specifying policies on analytics services to control their access and execution. This is documented in D8.3, Section 4.1.2.3.
C3ISP-Fun-DA-002	C3ISP policies allows writing “post-processing rules” on analytics operation result, which are data manipulation operations performed before returning it to the Prosumer(s). DMOs should include data sanitisation and (de)encryption (see also C3ISP-Fun-DS-011 and 9.3)	The post-processing rules are encoded in the DSA policies as obligations to be carried out after that the data access has been granted, and before that the result that has been produced is stored on the ISI system. These rules, which are enforced by the DSA Adapter component of the ISI subsystem, modifies the data according to the specified algorithm before allowing the execution of the analytics. C3ISP supports the following DMOs: different types of anonymisation, FHE encryption, FPE encryption, pseudonymisation (see Section 4.1.5).
C3ISP-Fun-DA-003	C3ISP provides an application programming interface for executing DMOs, such as privacy-preserving operations on data (e.g. data sanitisation or encryption)	A specific DMO Engine API allows to implement this requirement, DMO execution is triggered automatically by the DSA Adapter if prescribed in the DSA of a resource, when that resource is being processed.
C3ISP-Fun-DA-004	C3ISP allows executing privacy-preserving DMOs on all or part of the data	The DSA policy allows the specification of data manipulation operations (DMOs) to be applied on specific data fields (or all the fields for which a privacy preserving technique is available), and the DMO Engine executes these when prescribed.
C3ISP-Fun-DA-007	C3ISP supports standard query language (e.g. SQL) for querying data and analytics operation results from C3ISP data lake	C3ISP Analytics Engine supporting creating Virtual Data Lake that can be queried via Apache Impala (which is based on SQL dialect). See Section 4.3.
C3ISP-Fun-DA-008	C3ISP provides a function for automatic threat classification of analytics operation results	The DSA Policy Merger (6.3) merges DSA policies, which also include DSA metadata for the analytics result. In the new DSA created for the analytics result, a new data classification is set.
C3ISP-Fun-DA-009	C3ISP provides a function for automated mapping of analytics operation results to interested	The DSA policies allow defining specific conditions on analytics results for triggering notifications to interested Prosumers (see 6.1).

	stakeholders/Prosumers that are specified in the DSA	
C3ISP-Fun-DA-011	C3ISP provides an application programming interface to integrate external analytics tools while preserving the policy compliance (i.e., extract data from C3ISP data lake and feed it into analytics tool)	The IAI API defines a standard way to integrate new analytics. Also it provides access for Legacy Analytics Engines via Virtual Data Lakes and SQL-like dialects thanks to the HDFS-based Apache Impala. (See also C3ISP-Fun-DA-007)
C3ISP-Fun-DA-012	C3ISP provides near real-time notifications of analytics operation results (see also C3ISP-Fun-DS-009)	The Obligations Engine provides this functionality.
C3ISP-Fun-DA-013	C3ISP provides a function to query analytics operation results of specific categories (e.g. malware analysis, attack on cloud platform)	The DPOS contains metadata associated to each stored DPO. These metadata contains, for example, the type of data like antivirus, malware, etc. The ISI API provides a Search API that can query by metadata all the DPOs, including the DPOs created as analytics result.
C3ISP-Fun-DA-014	C3ISP supports different categories for analytics operations results, i.e. threat types, threat risks, threat origins, threat costs, regulatory and compliance concerns	This is accomplished by the same mechanism explained in C3ISP-Fun-DA-013, as long as the required metadata is provided when the DPOs are initially stored.
C3ISP-Fun-DA-015	C3ISP supports the provisioning of analytics operation results in form of actionable items (e.g. security patches, recommended configurations, fixes, etc.). See also the OpenC2 description in 9.4.3.	This strictly depends on the implemented analytics service. C3ISP supports STIX, which can be augmented with actionable items. For example, the SPAM detection analytics returns augmented STIX CTI as their execution result.
C3ISP-Fun-DA-016	C3ISP provides a dashboard showing status and results of the analysis	It is not part of the C3ISP architecture, but the Pilots portals provide a dashboard to track the progress of analytics and to pick up the results by quoting the ticket.
C3ISP-Fun-DA-017	C3ISP allows scheduling of the provisioning of analytics operation results (e.g. on demand, periodical, etc.)	It is not part of the core C3ISP services, but C3ISP provides an asynchronous interface to run the analytics and to retrieve the result (based on a ticket). The Pilots portals can periodically poll C3ISP for getting the result. Analogously, the Pilots portals can schedule the execution of analytics and then retrieve the results.
C3ISP-Fun-DA-018	When using homomorphic encryption, before data analytics execution, data is represented as bits or integers.	FHE Encryption Service opens an API with CEF file as an input parameter. This API consists of extracting IPs from a given pattern (src or dst). Each IPv4 like x.x.x.x is formatted as an integer of 32 bits and which

		is encrypted and generated to 32 cipher-texts.
C3ISP-Fun-DA-019	When using homomorphic encryption, data is of constant length (in real world scenarios). If not, a possible solution is to compute a hash function (not necessarily a cryptographic one) on data.	When developing FHE service for performing IPv4 with FHE analysis, the IP is always in constant format and constant length. Hence, only a method (not using hash function) to convert them to a corresponding integer of 32 bits is enough.
C3ISP-Fun-DA-020	When using homomorphic encryption, analytics operands (cipher-texts) are encrypted bits (most current case) or encrypted integers with considered homomorphic cryptosystems.	Since each IPv4 is encrypted in 32 cipher-texts, only this operand for encrypted bit is considered.
C3ISP-Fun-DA-021	When using homomorphic encryption, analytics operations are expressible in terms of two elementary operations: (homomorphic) addition and (homomorphic) multiplication with considered homomorphic cryptosystems, Homomorphic sum (resp. product) of two cipher-texts is a cipher-text of the sum (resp. the product) of two associated plaintexts.	In C3ISP Framework, the data is considered in 32 bits, all the elementary operation (FHE addition and multiplication) are developed and tested by CEA. The optimised version 2.0 and latest version of Cingulata is now available on github (https://github.com/CEA-LIST/Cingulata/wiki)
C3ISP-Fun-DA-022	When using homomorphic encryption, analytics computation on encrypted bits is represented as a Boolean circuit with multiplicative depth[19] roughly 20 or 30.	C3ISP Framework provides the algorithm allowing working with multiple encryption schemes (transciphering for short). This mechanism is helpful for optimising bandwidth communication. But the performance is very bad because the multiplicative depth is roughly 20. To optimise it, the FHE Analysis Engine offers the process for encrypting data in FHE format, and does not combine Kreyvium with FHE encryption process. Hence, the performance is improved because the multiplicative depth is around 5.
C3ISP-Fun-DA-023	When using homomorphic encryption, the number of multiplicative gates should be minimised to decrease latency of homomorphic evaluation.	The optimisation has been done and fully integrated in the latest version of open source Cingulata (https://github.com/CEA-LIST/Cingulata/wiki). The principle is to replace the multiplicative gates by a combination of non-multiplicative gates. For

[19] Multiplicative depth is the maximum number of multiplicative gates between an input and an output of the circuit.

		more details refer to the paper "*Faster homomorphic encryption is not enough: improved heuristics for multiplicative depth minimization of Boolean circuits",* which has been submitted to Asiacrypt conference 2019.

9.3. Data Manipulation Operations

In the context of the DMOs described for C3ISP-Fun-DS-011 and C3ISP-Fun-DA-002, possible privacy preserving techniques that find application in the Pilots' use cases are discussed in D8.3, Section 8 (Anonymisation Toolbox, FHE, FPE and Pseudonymisation).

9.4. Non-Functional Requirements

9.4.1. Information Security Requirements

Table 5 – Coverage of Information Security Requirements

ID	Requirements	Coverage
C3ISP-Sec-001	There is mutual authentication carried out between the C3ISP Framework and the Prosumers at the start of any communication.	Prosumers are authenticated and servers are identified using digital certificates and HTTP over TLS (https)
C3ISP-Sec-002	Confidentiality and Integrity of the DSA communications between the Prosumers and C3ISP Service is guaranteed.	We use secure communication with the DSA Manager based on HTTP over TLS and prosumers authentication to guarantee both confidentiality and integrity of data transmission
C3ISP-Sec-003	The transfer of CTI from the Prosumers to the C3ISP Framework is secure (w.r.t. confidentiality and integrity).	Same as C3ISP-Sec-002. In addition, the Key & Encryption Manager of CSS provides security to the DPO bundle, which is encrypted (for confidentiality) and stored with a hash code (for integrity). See Bundle Manager in D7.3, Section 5.1.6.
C3ISP-Sec-004	The integrity of the CTI data stored on the C3ISP Framework is maintained.	The ISI Data Protected Object Storage supports this. The integrity of data is ensured by Bundle Manager, the function HMAC HASH with secret key is used to compute and compare the hash of data.
C3ISP-Sec-005	The transfer of analysis results from the C3ISP Framework to the Prosumers is secure (w.r.t. confidentiality and integrity).	Since the analysis result is a DPO, the same applies as requirement C3ISP-Sec-003/004. FHE Encryption provides a further security measure, when used.
C3ISP-Sec-006	C3ISP Framework is able to process anonymised or homomorphically encrypted CTI shared with it by the Prosumers.	The C3ISP Analytics Service supports processing of anonymised CTI. In particular, FPE has been added to C3ISP to guarantee that computation can be applied without it

		should know that data is protected (FPE encrypted via the specific DMO). The FHE DMO supports the processing of homomorphically encrypted CTI. For a high performance with trade-off between the security levels and keep FHE analysis in practical usage, C3ISP Framework proposes four configurations named: • BLACK_LIST_FULL • BLACK_LIST_HIGH • BLACK_LIST_MEDIUM • BLACK_LIST_LOW They permit to establish a tradeoff between security and efficiency by encrypting respectively the last i=4,3,2,1 byte(s) of each IPv4.

9.4.2. Regulatory Requirements

Table 6 – Coverage of Regulatory Requirements

ID	Requirements	Coverage
C3ISP-Sec-102	The Prosumers are able to reject or cancel the terms and conditions of their DSA with the C3ISP Framework at any time. (GDPR Requirement)	The DSA Editor allows Prosumers to revoke a DSA at any time (or even edit): the DSA Adapter react to this change to the DSA policies in near real-time, by forbidding further access to the shared data.
C3ISP-Sec-106	For accountability purposes, C3ISP has an auditing subsystem that traces the enforcement results of the policies	The Secure Audit Manager provides this functionality.

9.4.3. Operational Requirements

9.4.3.1. Cloud Computing and Deployment Models Requirements

Table 7 – Coverage of Cloud Computing and Deployment Models Requirements

ID	Requirements	Coverage
C3ISP-Ope-002	C3ISP is multi-tenant, where several tenants (i.e. pilots) can use the framework at the same time w/o troubles	This is to be covered by the implementation of the micro-service architecture (e.g., see Section 2.1). In addition, the DSA Adapter is able to cope with different Pilots' Prosumers at the same time (see also the introduction of the application-domain field in the DSA in Section 6.1).

9.4.3.2. Extensibility and Interoperability Requirements

Table 8 – Coverage of Extensibility and Interoperability Requirements

ID	Requirements	Coverage
C3ISP-Ope-102	C3ISP uses a standard to represent data in order to simplify the integration with the framework	We selected to adopt STIX standard as the native C3ISP format and CEF as log file format. CEF can be embedded in STIX. Furthermore, the Format Adapter supports the conversion of different input formats such as CSV, JSON, etc., to the C3ISP standard format (STIX).
C3ISP-Ope-103	C3ISP should be able to represent different kind of *cyber observables*	This is supported because CybOX [7] are specific STIX objects.
C3ISP-Ope-104	C3ISP provides information related to the *cyber observables* which characterize it	This is supported because CybOX are specific STIX objects.

9.4.4. Performance Requirements

The performance of the C3ISP proof of concept implementation is not of primary concern.

Table 9 – Coverage of Performance Requirements

ID	Requirements	Coverage
C3ISP-Per-001	C3ISP does not introduce significant delay when enforcing policies for sharing analytics operation results	We assessed the C3ISP performances and worked to improve them (see D8.3 Section 5.7.2.7).
C3ISP-Per-002	FHE services are deployed into a physical server, not into a Virtual Machine, because of using parallelism method for optimising the Boolean circuits.	In any configuration, the FHE analysis service can adapt to because C3ISP can configure it in function of the number of available CPUs dedicated to FHE analysis. The higher the number of available core it is, the better performance it will be.
C3ISP-Per-003	With each pilot’s use case, C3ISP defines an interval of tolerant response delay, in order to obtain a compromise resource availability for other requests on FHE services (*response delay requirement*). This requirement applies only if FHE is used.	When running FHE analysis, the majority of API working with this service are in asynchronous requests.

9.4.5. Usability Requirements

Table 10 – Coverage of Usability Requirements

ID	Requirements	Coverage
C3ISP-Usa-001	C3ISP provides response information (results or errors) about requests (analytics query) to the C3ISP data lake (e.g. why the requested data cannot be provided to Prosumers)	When trying to retrieve the ticket for the result DPO, C3ISP notifies the Prosumer of errors, if any.
C3ISP-Usa-004	C3ISP's representation of analytics results is effective and efficient for the end user.	It is not part of the C3ISP Framework, but the Pilot portals provide UIs to present the analytics results. There is also the *Interactive 3D Visualisation* capability described in Section 5.1.2. For the effectiveness and efficiency consideration, please see the corresponding Pilots deliverables.

10. Update on the Development, Test Bed and Prod Environments

This section includes the updates done to the operating environments of the C3ISP Framework. In addition to Development and Test Bed which we tuned during the last period, we introduced a new Production due to the need of having a more stable environment (updated periodically) than the Test Bed which contains the unstable, but always updated C3ISP Framework implementation (thanks to the continuous integration server we setup, Jenkins). In addition, Local ISI instances (made with Docker containers, as discussed in 3.1) have been deployed for the Pilots integration and validation activities (specifically for ISP, CERT, and SME Pilots).

The Configuration Server described in Section 2.1 has been set up for Test Bed and Production environments, as well as it is part of the Local ISIs instances (as a Docker container).

We also describe the security testing assessment carried out on the C3ISP Framework and the planned remediation activities. This assessment is the first phase of the remediation plan; in D7.5 we will report the final outcomes of this plan.

Please consider that all the machines hosted in those environments are continually patched for security issues (at operating system level) and when necessary also with feature updates.

10.1. Development Environment

10.1.1. Base configuration

There are no changes to the configuration of the VM that hosts the Development environment.

10.1.2. Software configuration

As described in D7.1, we installed and used **OWASP ZAP** as a DAST tool. For more details, please see description in Section 10.4.2.

Docker was installed to create the containers used for Local ISI deployment, as discussed in Section 3.1.

10.1.3. End-to-end functional integration tests

We had several necessities to assess whether the deployed C3ISP Framework is stable and can be used without issues, including:

- Partners are presenting C3ISP at conferences and customers;
- C3ISP DevOps administrators should update Production environment with latest features from Test Bed;
- Pilots should be able to mature their implementation and integration with C3ISP.

For these reasons we developed a C3ISP end-to-end integration test suites. It is a list of test cases that verify the most common C3ISP use cases and functionalities, like creating and reading DPOs or running an analytics and retrieving its result.

Test cases are written in Java using Spring Boot and the REST-assured[20] framework that makes extremely easy and intuitive writing tests for REST services of the C3ISP micro-services based

[20] http://rest-assured.io/

architecture. In the next table, we report a simplified example just to have a grasp how the effectiveness and simplicity of these tests:

Table 11 – Sample test case for createDPO (ISI API)

LOGGER.info(">Invoking createDPO");	
JsonPath response =	
given()	Preconditions for the REST call, like input parameters, etc.
.log().all()	
.accept(ContentType.*JSON*)	
.queryParam("norm", "false")	
.multiPart(formInputMetadata, createDpoInputMetadata)	
.multiPart("fileToSubmit", ResourceUtils .*getFile*("classpath: DNS_Bind_StixFormat.json")).	
when()	Execution of the REST call, in this case a POST request
.post(createDpoPath).	
then()	Validation of REST call response by doing some assertions on the result content (e.g. DPO ID must not be null, etc.)
.log().all()	
.statusCode(HttpStatus.***OK***.value())	
.assertThat().body("additionalProperties.dposId", *not*(*isEmptyOrNullString*()))	
.assertThat().body("additionalProperties.dposId", *not*(*is*("FAILURE"))).	
extract().jsonPath();	

The test suite runs daily and reports are available from a dedicated Jenkins job, as shown in the next figure:

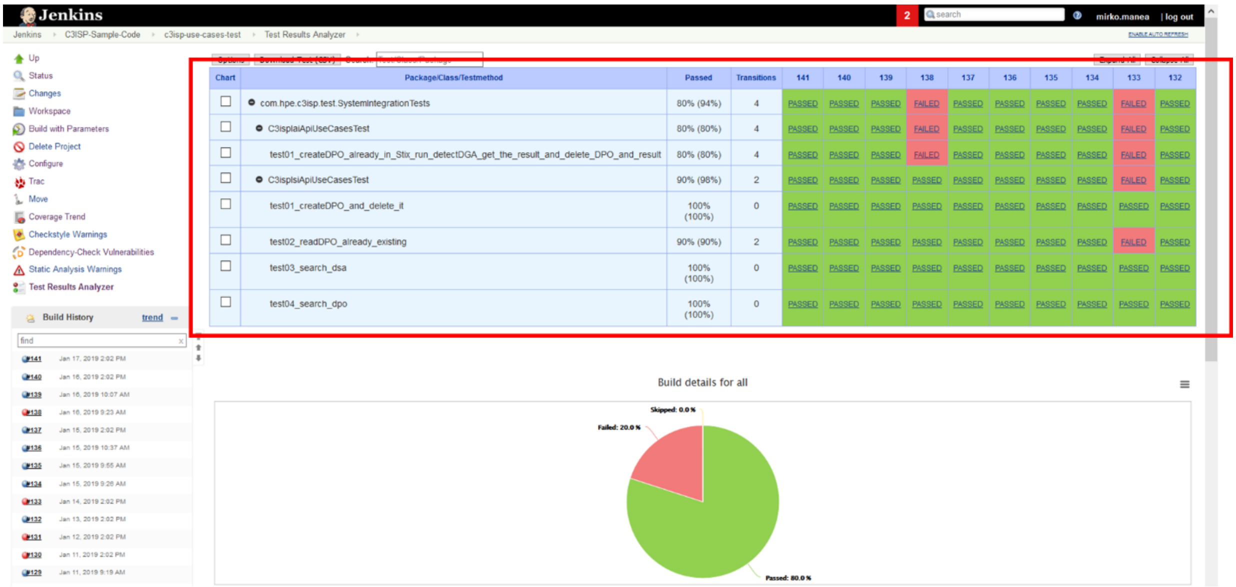


Chart	Package/Class/Testmethod	Passed	Transitions	141	140	139	138	137	136	135	134	133	132
□	● com.hpe.c3isp.test.SystemIntegrationTests	80% (94%)	4	PASSED	PASSED	PASSED	FAILED	PASSED	PASSED	PASSED	PASSED	FAILED	PASSED
□	● C3ispIaiApiUseCasesTest	80% (80%)	4	PASSED	PASSED	PASSED	FAILED	PASSED	PASSED	PASSED	PASSED	FAILED	PASSED
□	test01_createDPO_already_in_Stix_run_detectDGA_get_the_result_and_delete_DPO_and_result	80% (80%)	4	PASSED	PASSED	PASSED	FAILED	PASSED	PASSED	PASSED	PASSED	FAILED	PASSED
□	● C3ispIsiApiUseCasesTest	90% (98%)	2	PASSED	PASSED	PASSED	PASSED	PASSED	PASSED	PASSED	PASSED	FAILED	PASSED
□	test01_createDPO_and_delete_it	100% (100%)	0	PASSED	PASSED	PASSED	PASSED	PASSED	PASSED	PASSED	PASSED	PASSED	PASSED
□	test02_readDPO_already_existing	90% (90%)	2	PASSED	PASSED	PASSED	PASSED	PASSED	PASSED	PASSED	PASSED	FAILED	PASSED
□	test03_search_dsa	100% (100%)	0	PASSED	PASSED	PASSED	PASSED	PASSED	PASSED	PASSED	PASSED	PASSED	PASSED
□	test04_search_dpo	100% (100%)	0	PASSED	PASSED	PASSED	PASSED	PASSED	PASSED	PASSED	PASSED	PASSED	PASSED

#141	Jan 17, 2019 2:02 PM
#140	Jan 16, 2019 2:02 PM
#139	Jan 16, 2019 10:07 AM
#138	Jan 16, 2019 9:23 AM
#137	Jan 15, 2019 2:02 PM
#136	Jan 15, 2019 10:37 AM
#135	Jan 15, 2019 9:55 AM
#134	Jan 15, 2019 9:26 AM
#133	Jan 14, 2019 2:02 PM
#132	Jan 13, 2019 2:02 PM
#131	Jan 12, 2019 2:02 PM
#130	Jan 11, 2019 2:02 PM
#129	Jan 11, 2019 9:19 AM

There is also an analogous test suite for assessing the Production environment in order to make sure it is always stable. Failures of tests executions are reported also by email to C3ISP developers.

10.2. Test Bed Environment

10.2.1. Base configuration

The configuration for each machine of the Test Bed environment is illustrated on Table 12 under the *Specifications* column. In addition, all machines are equipped with Ubuntu 16.04 LTS as operating system.

Table 12 – C3ISP subsystems on Test Bed Environment

Subsystem	Virtual Machine	Specifications	Notes

DSA Manager	dsamgrc3isp.iit.cnr.it	vCPU = 2
		Ram = 4GB
		Storage = 40GB
			This virtual machine hosts the DSA Manager subsystem.

ISI	isic3isp.iit.cnr.it	vCPU = 4
		Ram = 12GB
		Storage = 100GB
			This virtual machine hosts the Information Sharing Infrastructure subsystem.

IAI	iaic3isp.iit.cnr.it	Cores = 8
		Ram = 16GB
		Storage = 200GB
			This server hosts the Information Analytics Infrastructure subsystem.

IAI	impala.iit.cnr.it	vCPU = 4
		Ram = 16GB
		Storage = 200GB
			This virtual machine hosts Impala to support HDFS access for Saturn.

CSS	kec3isp.iit.cnr.it	vCPU = 8
		Ram = 11GB
		Storage = 20GB
			This virtual machine hosts the Key and Encryption Manager components, part of the CSS subsystem.

CSS	auditmgrc3isp.iit.cnr.it	vCPU = 1
		Ram = 2GB
		Storage = 20GB
			This VM runs the Secure Audit Manager component (off-the-shelf product based on Graylog). See 7.3.

CSS	oidcc3isp.iit.cnr.it	vCPU = 2
		Ram = 4GB
		Storage = 20GB
			This virtual machine hosts the OIDC Server (cf. 7.1.2).

ISP Pilot	ispc3isp.iit.cnr.it	vCPU = 6
		Ram = 8GB
		Storage = 60GB
			This virtual machine hosts all services and components for the ISP Pilot that are used to interact with the C3ISP Framework.
			It also hosts the Local ISI (with Docker) due to the

			C3ISP Hybrid deployment model used by the pilot.

CERT Pilot	90.147.82.10	vCPU = 4
		Ram = 8GB
		Storage = 200GB
			This virtual machine hosts all services and components for the CERT Pilot that are used to interact with the C3ISP Framework.
			It also hosts the Local ISI (with Docker) due to the C3ISP Hybrid deployment model used by the pilot.

Enterprise Pilot	entc3isp.iit.cnr.it	vCPU = 4
		Ram = 4GB
		Storage = 60GB
			This virtual machine hosts all services and components for the Enterprise Pilot that are used to interact with the C3ISP Framework.

SME Pilot	smec3isp.iit.cnr.it	vCPU = 8
		Ram = 16GB
		Storage = 200GB
			This virtual machine hosts all services and components for the SME Pilot that are used to interact with the C3ISP Framework.
			It also hosts the Local ISI (with Docker) due to the C3ISP Hybrid deployment model used by the pilot.

Note: even if Pilots servers are installed at CNR, they point to Production environment (hosted at BT), via a VPN connection. This is because Pilots has the need to interact with the C3ISP stable environment for the C3ISP Framework evaluation and validation.

10.2.2. Software configuration

In addition to the off-the-shelf software reported in D7.3 (MySQL, MongoDB, Apache HDFS), we installed and configured **Spring Cloud Config** ([76]) as the basis for the implementation of the Configuration Server (see 2.1).

10.3. Prod Environment

With the aim of having a frozen environment useful for Demos at events and also a stable environment for Pilots' evaluation/validation, the consortium built, on BT premises, a Production Environment (Prod) starting from a clone of Test Bed at M26 (Validation state). On these systems, partners made all suitable configurations to have a complete Prod Environment disjoined from Test Bed. Prod Environment can be accessed by setting up a Virtual Private Network (VPN) connection to BT data centre, only.

We also setup a Jenkins pipeline job to regularly synchronise the Prod Environment by promoting the components verified (see end-to-end integration tests 10.1.3) in the Test Bed.

The rest of this section provides a short summary of the Prod Environment specification.

10.3.1. Base configuration

The following table maps the C3ISP subsystems to the Prod Environment:

Table 13 – C3ISP subsystems on Prod Environment

Subsystem	Virtual Machine	Specifications	Notes

DSA Manager	dsamgrc3isp.rp.bt.com	vCPU = 2
		Ram = 4 GB
		Storage = 30 GB
			This virtual machine hosts the DSA Manager subsystem.

ISI	isic3isp.rp.bt.com	vCPU = 4
		Ram = 8 GB
		Storage = 100 GB
			This virtual machine hosts the Information Sharing Infrastructure subsystem.

IAI	iaic3isp.rp.bt.com	vCPU = 4
		Ram = 8 GB
		Storage = 60 GB
			This virtual machine hosts the Information Analytics Infrastructure subsystem.

CSS	kec3isp.rp.bt.com	vCPU = 8
		Ram = 16 GB
		Storage = 50 GB
			This virtual machine hosts the Key and Encryption Manager components, part of the CSS subsystem.

CSS	auditmgrc3isp.rp.bt.com	vCPU = 1
		Ram = 2 GB
		Storage = 20 GB
			This VM runs the Secure Audit Manager component (off-the-shelf product based on Graylog). See 7.3.

All systems runs Ubuntu 16.04 LTS on a BT managed data centre with Xen[21] virtualisation software.

10.3.2. Software configuration

Because systems are built as clone of Test Bed, software releases are the same for that environment.

C3ISP Framework components are built for Test Bed Environment and promoted to Prod Environment when new features are ready and successfully tested from an end-to-end perspective. Thanks to the Configuration Server (see 2.1), software components are exactly the

[21] Xen Project, https://xenproject.org/

same and upon start they download the specific settings for Prod Environment (e.g. the different IPs used for Prod environment).

10.4. Security Tests

In the "recommendation concerning for future work" section (see Table 14) of the C3ISP Project Review at M26, EC suggested to perform Security Testing against the C3ISP Framework components in a specific observation.

Table 14 – EC Observation about Security Testing

EC Observation	C3ISP Consortium response already provided to EC
O: *"Security testing (and possibly certification) should be included in the plan for all systems providing publicly accessible portal (e.g. ISP pilot portal, C3ISP Gateway portal, APIs), as well as the system as a whole.*	**R:** "we already have setup tools for security analysis at build time for all C3ISP software components. In particular, we employ static code analysis to spot security defects (e.g. FindBugs with FindSecurityBugs), as well as to find used opensource libraries that have known CVE - security defects (see OWASP Top 10, item A9). We plan to extend security analysis to DAST (Dynamic Application Security Testing) for Pilots portals, C3ISP Gateway and front-end C3ISP APIs, by employing opensource scanners (e.g. OWASP ZAP) that could spot security bugs at runtime, like SQL Injection, Cross-Site Scripting, etc."

We exercised the C3ISP Framework in three different, but complementary, types of Security Testing activities:

- Static Application Security Testing (SAST);
- Dynamic Application Security Testing (DAST);
- Infrastructure Security Testing.

The process followed to run all tests is the same and is made by these steps:

1. Security Assessment of components in scope;
2. Discussion with components owners to present the outcomes;
3. Definition of remediation plan;
4. Actuation of remediation plan;
5. New assessment phase to check remediation activities effectiveness.

Regarding assessment step (1), each kind of test had a specific tool used and different target components as described in next paragraphs. In particular, before running DAST activity on live portals, we asked for explicit authorisation to involved partners (authorisation letters will be included in D7.5 since they are considered confidential material).

Assessment results were shared with components' owners (step 2) to evaluate vulnerabilities (e.g. true or false positive) and prepare a remediation plan (step 3); finally each owner was asked to execute the plan (step 4). Remediation plan and actions were focused on High/Medium vulnerabilities scores.

After remediation phase, we will run again the security assessment (step 5) to check how much effective were the measures put in place (e.g. source code fixes, etc.) and will publish the results in deliverable D7.5.

10.4.1. Static Analysis Security Testing

Static Analysis Security Testing (SAST) is a set of technologies designed to analyse application source code, byte code and binaries for coding and design conditions that are indicative of security vulnerabilities. SAST solutions analyse an application from the "inside out" in a no running state. We used these tests to analyse static code for Front End Components of C3ISP Framework:

- ISI API
- IAI API
- DSA Editor
  - DSA Editor Front End
  - DSA Editor Tool
- SME Pilot Portal
- SME Gateway
- ISP Pilot Portal
- CERT Pilot Portal

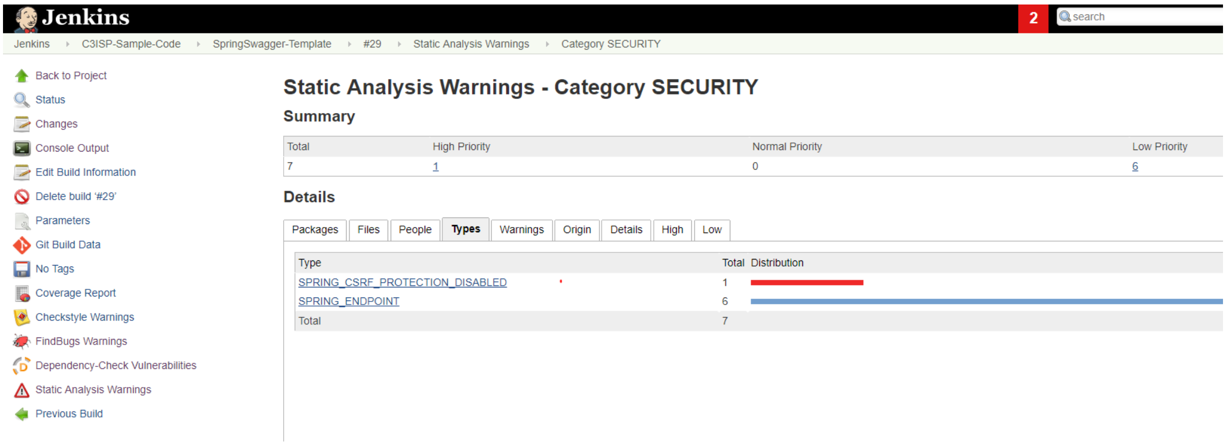
Static Analysis of code was executed in steps:

- Analysis of component source code: we used Find Bugs, augmented with the Find Security Bugs extension;
- Analysis of third-party libraries: we used OWASP Dependency Check to spot known vulnerabilities among the used open source libraries.

Results were analysed thanks to a Jenkins Utilities Plugin who gives two outputs:

- Static Analysis Warnings (Category: Security);
- Dependency-Check Vulnerabilities.

First output, narrowed on Security Category, gives a list of code that could be affected by a potential vulnerability (e.g. External method exposed, Hard-coded password, Possible Log manipulation, Potential path traversal vulnerabilities, etc.). For these Warnings, it presents suggested actions for remediation, generally describing a better (more secure) way of writing code.



Static Analysis Warnings - Category SECURITY

Summary

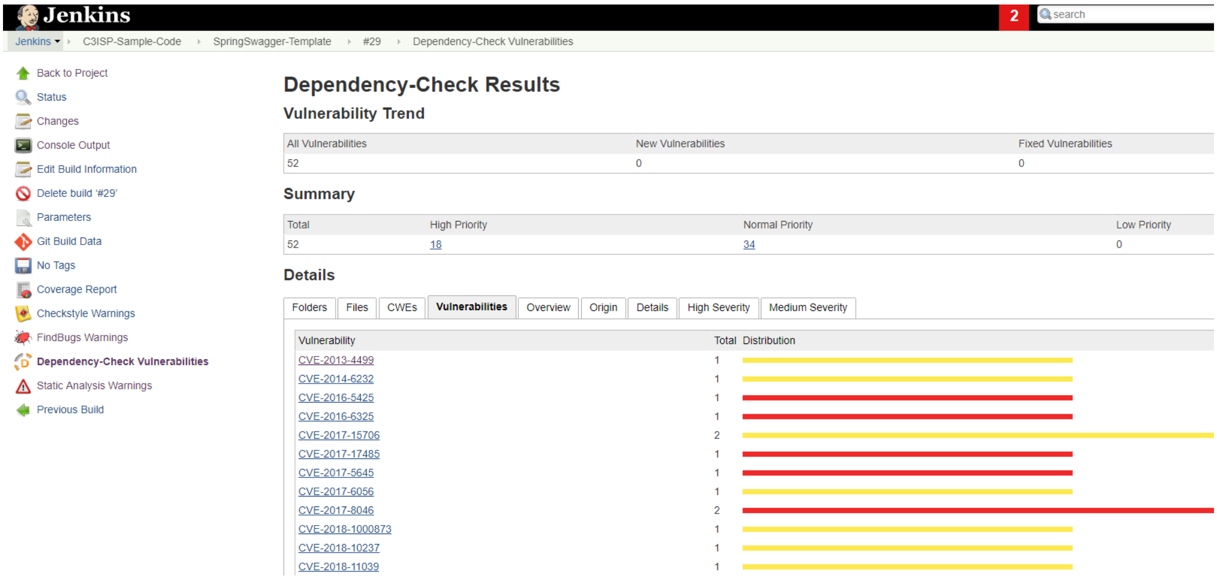
Total	High Priority	Normal Priority	Low Priority
7	1	0	6

Details

Type	Total	Distribution
SPRING_CSRF_PROTECTION_DISABLED	1	🟥
SPRING_ENDPOINT	6	🟦
Total	7	

Figure 38: Find Security Bugs warning list by Types

Second output reports the list from OWASP Dependency Check showing the software library components that have known vulnerabilities (i.e. a well-known CVE). In this case, the possible remediation is usually to upgrade the software releases impacted.



Dependency-Check Results

Vulnerability Trend

All Vulnerabilities	New Vulnerabilities	Fixed Vulnerabilities
52	0	0

Summary

Total	High Priority	Normal Priority	Low Priority
52	18	34	0

Details

Vulnerability	Total	Distribution
CVE-2013-4499	1	🟨
CVE-2014-6232	1	🟨
CVE-2016-5425	1	🟥
CVE-2016-6325	1	🟥
CVE-2017-15706	2	🟨
CVE-2017-17485	1	🟥
CVE-2017-5645	1	🟥
CVE-2017-6056	1	🟨
CVE-2017-8046	2	🟥
CVE-2018-1000873	1	🟨
CVE-2018-10237	1	🟨
CVE-2018-11039	1	🟨

Figure 39: OWASP Dependency Check list of well-known CVE for used 3rd-parties libraries

10.4.2. Dynamic Application Security Testing

Dynamic Application Security Testing (DAST) technologies are designed to detect conditions indicative of a security vulnerability in an application in its running state.

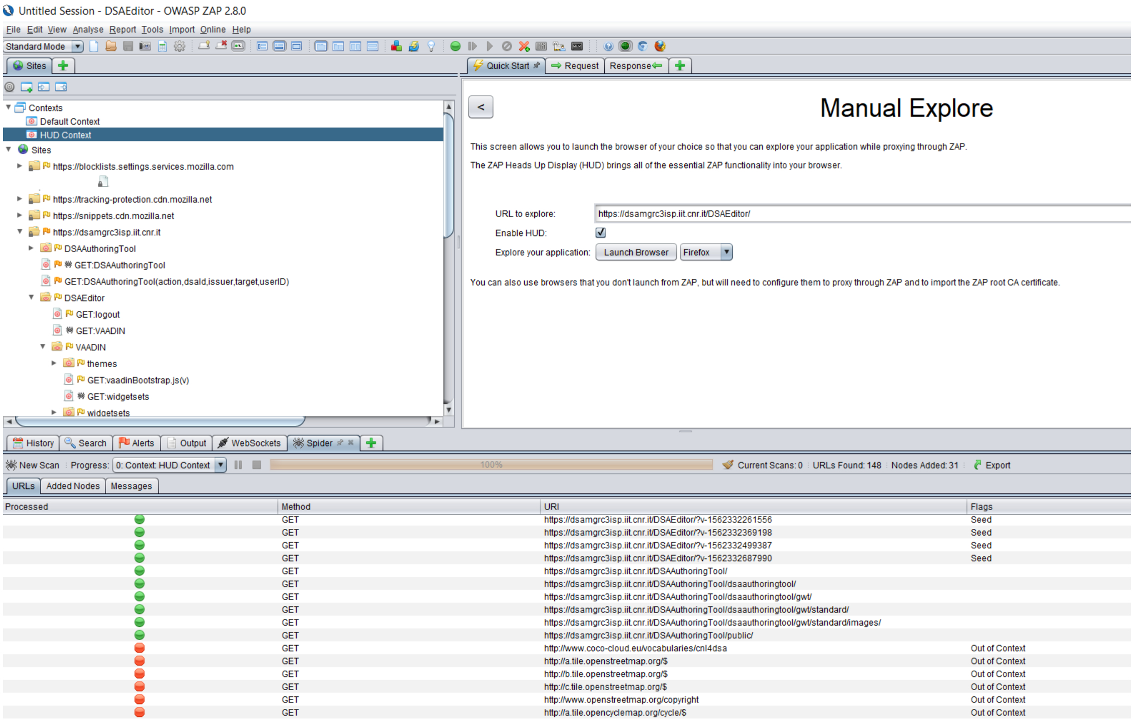
For this test we used OWASP Zed Attack Proxy (ZAP[22]), an open-source web application security scanner sponsored by the Open Web Application Security Project foundation (OWASP[23]).

We used the tool to assess C3ISP Pilot Portals (ISP, CERT, SME) and DSA Editor Tool, and to generate a report for each of them.

[22] https://www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project

[23] https://www.owasp.org

The next screenshot shows an attack session taken during the testing activities:



Untitled Session - DSAEditor - OWASP ZAP 2.8.0
File Edit View Analyse Report Tools Import Online Help
Standard Mode
Sites
Quick Start
Request
Response
Contexts
Default Context
HUD Context
Sites
https://blocklists.settings.services.mozilla.com
https://tracking-protection.cdn.mozilla.net
https://snippets.cdn.mozilla.net
https://dsamgrc3isp.iit.cnr.it
DSAAuthoringTool
GET:DSAAuthoringTool
GET:DSAAuthoringTool(action,dsaId,issuer,target,userID)
DSAEditor
GET:logout
GET:VAADIN
VAADIN
themes
GET:vaadinBootstrap.js(v)
GET:widgetsets
widgetsets
<
History
Search
Alerts
Output
WebSockets
Spider
New Scan
URLs
Added Nodes
Messages
Export

Manual Explore

This screen allows you to launch the browser of your choice so that you can explore your application while proxying through ZAP.

The ZAP Heads Up Display (HUD) brings all of the essential ZAP functionality into your browser.

URL to explore: https://dsamgrc3isp.iit.cnr.it/DSAEditor/

Enable HUD: ☑

Explore your application: Launch Browser Firefox

You can also use browsers that you don't launch from ZAP, but will need to configure them to proxy through ZAP and to import the ZAP root CA certificate.

Progress: 0: Context: HUD Context 100% Current Scans: 0 URLs Found: 148 Nodes Added: 31

Processed	Method	URI	Flags
🟢	GET	https://dsamgrc3isp.iit.cnr.it/DSAEditor/?v-1562332261556	Seed
🟢	GET	https://dsamgrc3isp.iit.cnr.it/DSAEditor/?v-1562332369198	Seed
🟢	GET	https://dsamgrc3isp.iit.cnr.it/DSAEditor/?v-1562332499387	Seed
🟢	GET	https://dsamgrc3isp.iit.cnr.it/DSAEditor/?v-1562332687990	Seed
🟢	GET	https://dsamgrc3isp.iit.cnr.it/DSAAuthoringTool/	
🟢	GET	https://dsamgrc3isp.iit.cnr.it/DSAAuthoringTool/dsaauthoringtool/	
🟢	GET	https://dsamgrc3isp.iit.cnr.it/DSAAuthoringTool/dsaauthoringtool/gwt/	
🟢	GET	https://dsamgrc3isp.iit.cnr.it/DSAAuthoringTool/dsaauthoringtool/gwt/standard/	
🟢	GET	https://dsamgrc3isp.iit.cnr.it/DSAAuthoringTool/dsaauthoringtool/gwt/standard/images/	
🟢	GET	https://dsamgrc3isp.iit.cnr.it/DSAAuthoringTool/public/	
🔴	GET	http://www.coco-cloud.eu/vocabularies/cnl4dsa	Out of Context
🔴	GET	http://a.tile.openstreetmap.org/$	Out of Context
🔴	GET	http://b.tile.openstreetmap.org/$	Out of Context
🔴	GET	http://c.tile.openstreetmap.org/$	Out of Context
🔴	GET	http://www.openstreetmap.org/copyright	Out of Context
🔴	GET	http://a.tile.opencyclemap.org/cycle/$	Out of Context

Figure 40: OWASP ZAP during an attack session of the DSA Editor

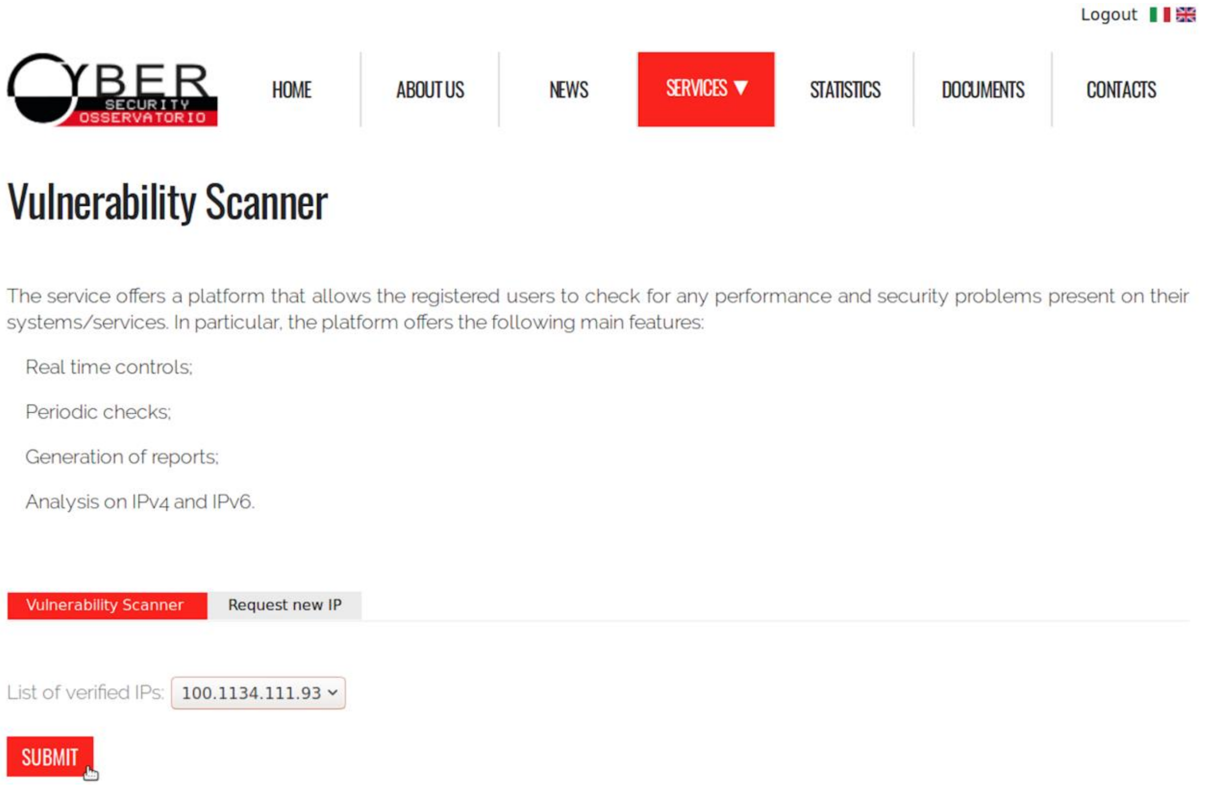
The vulnerability scan was done in several steps:

1. **Setup of entry point conditions**: we instructed the tool on how to login automatically and how to detect when it is not logged in. This is important because during the site scan, the tool can follow the 'logout' link and it must be aware of the authentication process to log back in again;
2. **Spidering of the web site**: the tool can navigate automatically the site as a web crawler, inspecting all the links found on each visited page. This is necessary to improve the 'attack surface', i.e. the list of pages/functions to be tested for vulnerabilities. However, since some portals used the Single Page Application (SPA) pattern[24], we had to use a more advanced web crawler known as Ajax crawler: this crawler can navigate the site by inspecting the changes to the page Document Object Model (DOM) and so can work with SPA web sites;
3. **Use cases recording**: the most significant functionalities of the site in scope were navigated manually letting OWASP ZAP recording the paths and data used for passing from one page to the other. This is fundamental, because even the better web crawler cannot successfully provide the required content to work with the business functionalities (e.g. selecting a specific DSA, filling a valid date for CTI searching, etc.). The aim of this step is to improve the site coverage and finally the attack surface to be exercised;
4. **Active Scan (attack)**: on the collected list of resources forming the attack surface, we instruct the tool to test for various types of attack patterns it is aware of (e.g. different types of injections like XSS, SQL Injection, etc.);
5. **Reporting**: after the session, we created reports with found vulnerabilities.

[24] http://www.computepatterns.com/1048/single-page-application-a-brief/

10.4.3. Infrastructure Security Testing

To assess the vulnerabilities of the servers involved in the project, we leveraged a service available from the https://www.cybersecurityosservatorio.it portal. Its Vulnerability Scanner service[25] allowed us to scan the virtual machines involved within the C3ISP Framework. Figure 41 illustrates the web page of the service used for the infrastructure security testing.



Vulnerability Scanner

The service offers a platform that allows the registered users to check for any performance and security problems present on their systems/services. In particular, the platform offers the following main features:

Real time controls;

Periodic checks;

Generation of reports;

Analysis on IPv4 and IPv6.

List of verified IPs: 100.1134.111.93

Figure 41: Vulnerability Scanner service from www.cybersecurityosservatorio.it

In particular, this service uses OpenVAS[26] as engine to scan the selected server. OpenVAS has been developed as vulnerability assessment system and includes unauthenticated, authenticated testing, and more than 50,000 Network Vulnerability Tests (NVTs).

The detailed report of vulnerabilities of vulnerabilities found at M34 is available internally to the Consortium members. However, section 10.4.4 reports aggregated data that can be safely published.

For each scan, the report contains the number of high, medium and low vulnerabilities found. In addition, for each vulnerability the following information is given:

- *NVT*: it specifies the test done with success on the server;
- *Port*: the port affected during the test;
- *Severity*: the relevance of the test. The result can be: high, medium or low;
- *Score*: it is the score of the vulnerability found. The higher the score, the higher its impact;
- *Summary*: it is the description of the test;

[25] This service is not public available, but on-request can be used.

[26] http://www.openvas.org

- *Impact*: it refers to impact that this attack may have (if exploited) on the scanned target;
- *CVE*: it says if there is a related CVE;
- *Remediation*: it is the plan to mitigate or avoid the vulnerability.

10.4.4. Security Assessment results

Due to confidentiality of Assessment result and the fact that this deliverable is of public nature, next we report some aggregated data about Security Assessment results and some examples, to give an idea, of potential vulnerabilities found.

Figure 42 reports the number of security issues at code-level (SAST) found on each tested C3ISP component, grouped by severity. As it appears clear from the graph, the absolute number of vulnerabilities were very low (in the order of tens) and mostly of medium severity. We think this is a result of having adopted from the very beginning a Secure Development Lifecycle with automated security tests performed at build-time from the continuous integration service (Jenkins, see D7.2 as well as the different security plugins used).

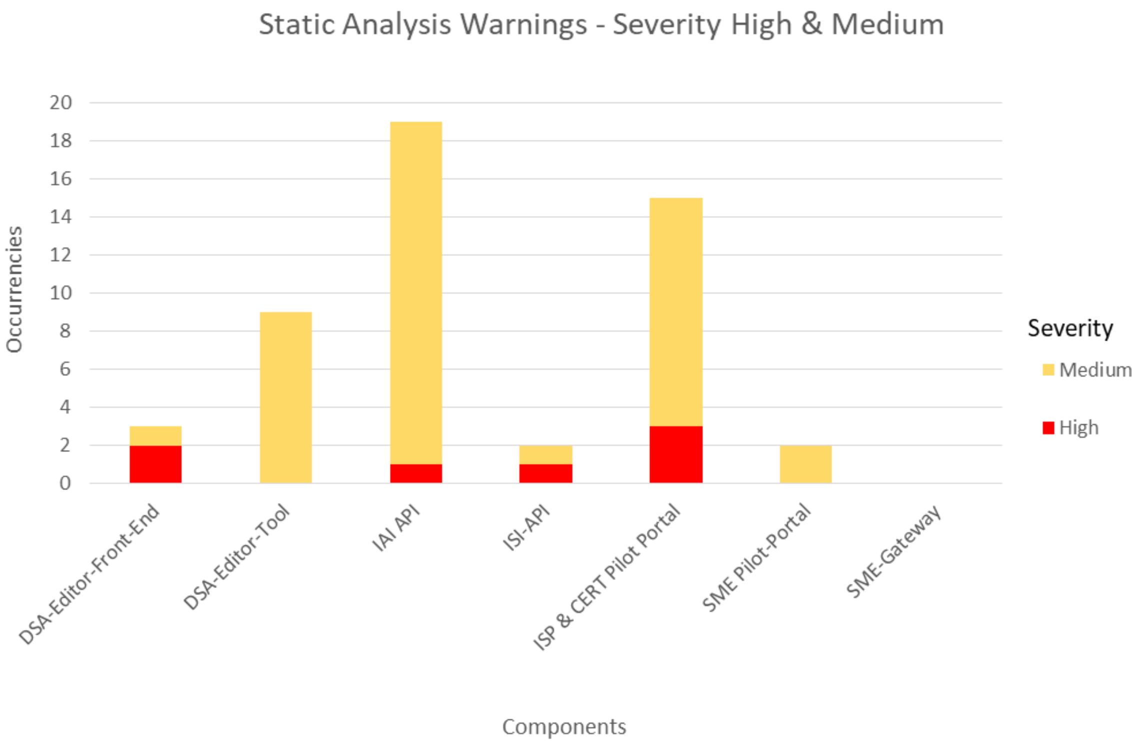


Figure 42: SAST - N. of Static Analysis Warnings by Component - Severity High & Medium

Next Figure 43 presents a different view on the SAST tests results, by presenting the type of security issues affecting the tested components. In particular three major categories of issues were found affecting different components (with a certain degree of spread): Path Traversal[27], Log Injection[28], and Cross-Site Request Forgery – CSRF (lack of) Protection[29]. We investigated the reasons for these being the most common. Path Traversals were found to be common

[27] OWASP reference for Path Traversal: https://www.owasp.org/index.php/Path_Traversal

[28] OWASP reference for Log Injection: https://www.owasp.org/index.php/Log_Injection

[29] OWASP reference for CSRF: https://www.owasp.org/index.php/Cross-Site_Request_Forgery_(CSRF)

because components use intensively files (CTIs, DPOs or DSAs) and Log Injection because components traced in the logs content coming from external sources (typically API parameters that might be abused).

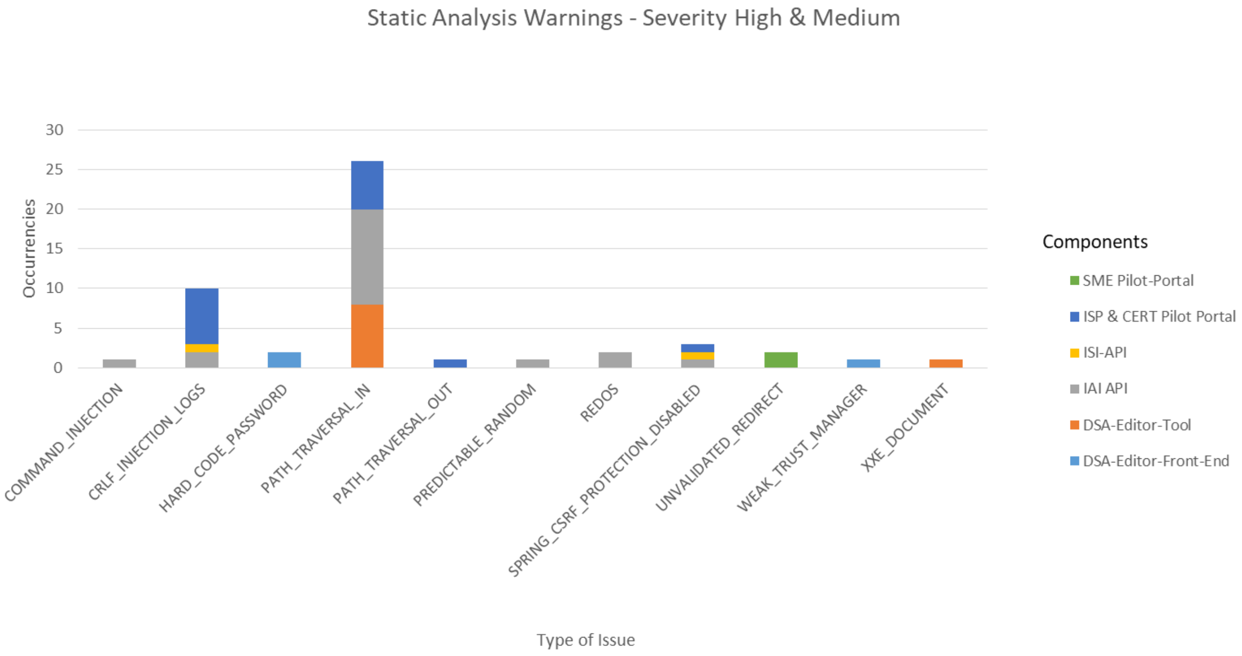


Figure 43: SAST - N. of Static Analysis Warnings by Type of Issues and C3ISP Component– Severity High & Medium

The next Figure 44 illustrates the output of the checking of used third-parties libraries (typically open-source) used by the C3ISP components in scope. We found there were several used libraries with known vulnerabilities (CVE) and most of the components shared the same set of issues. This is easy to explain, because all components are micro-services built with Spring Boot technology and they are all based on the same code template defined in earlier stages of the project. At the same time, this is quite easy to fix, because it is just a matter of upgrading the libraries specified in the Maven’s *pom.xml* configuration file. Of course upgrading is not seamless, because new libraries versions can change the API interfaces and/or API behaviours. To address a change in the interface it is just a matter of syntax and usually can be achieved quickly (e.g. a changed method signature). However, changes in the API behaviour requires in-depth non-regression testing to make sure it does not break the functionality. In our case this is facilitated because we introduced a set of end-to-end functional tests (see Section 10.1.3), in addition to component-specific unit tests, which helps assuring the C3ISP Framework does not break upon such changes.

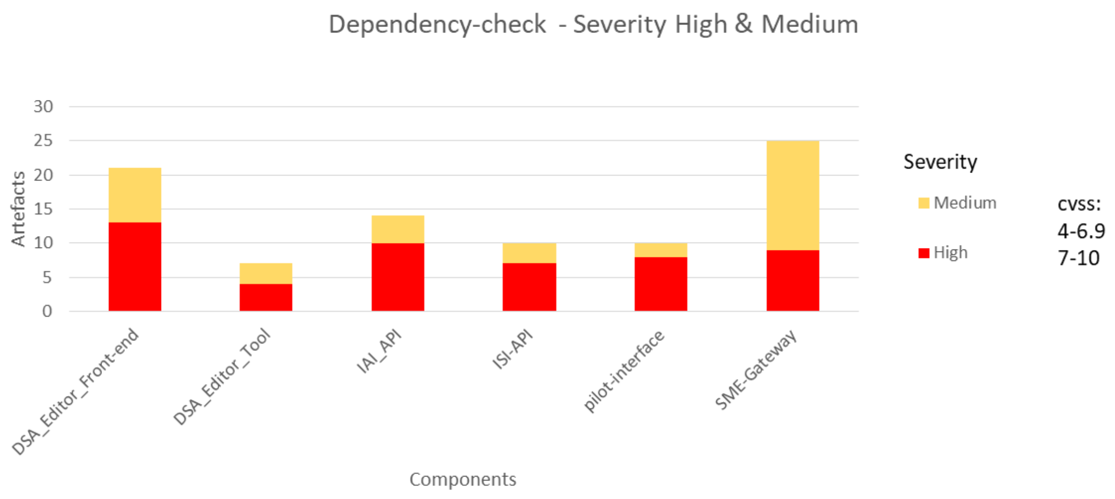


Figure 44: SAST - N. of Artefacts for C3ISP Component with Dependency-Check Vulnerabilities – Severity High & Medium and correspondent CVSS score[30]

The following Figure 45 completes the Application Testing with the Dynamic Application Security Testing (DAST), i.e. the testing of the live running application. Here we found that a single issue is very prevalent, which could lead to the possible exploitation of Clickjacking attacks[31]. These types of attacks can be mitigated by using specifically defined HTTP protection headers (i.e. X-Frame-Options), that the Internet browser use to control how content can be embedded in HTML Frames.

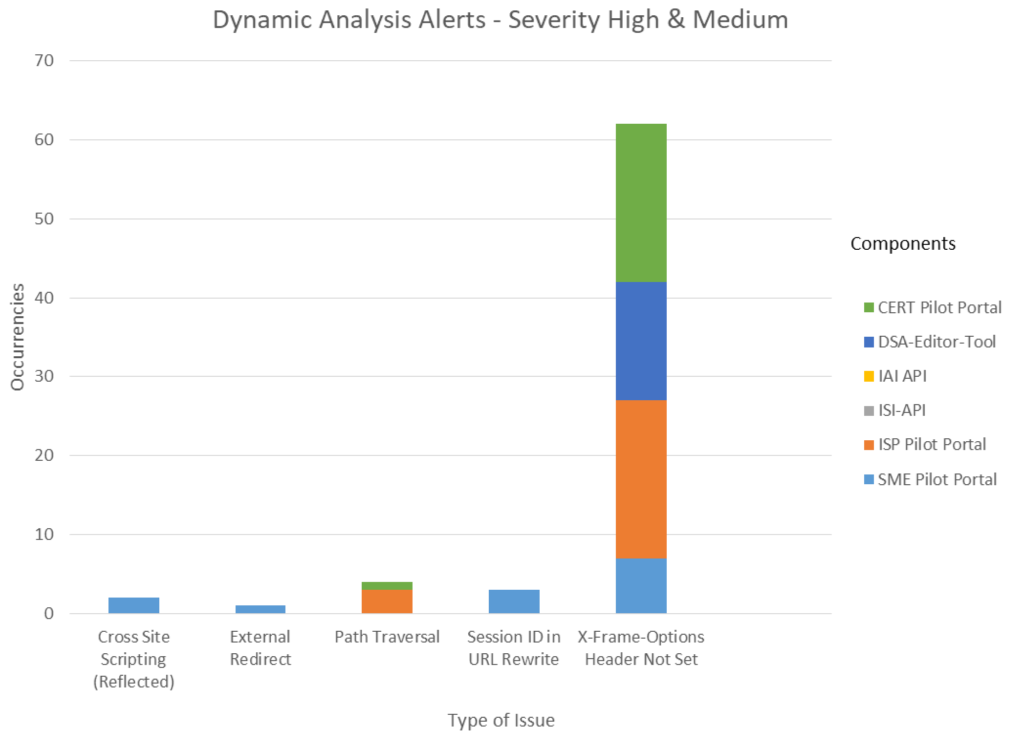


Figure 45: DAST - N. of issues by Type of Alert and C3ISP Component – Severity High & Medium

[30] Common Vulnerability Scoring System, https://www.first.org/cvss/

[31] "*The attacker is "hijacking" clicks meant for their page and routing them to another page*", for further details please see https://www.owasp.org/index.php/Clickjacking

Figure 46 provides a different view on the same DAST results as previous diagram, grouping by component and severity:

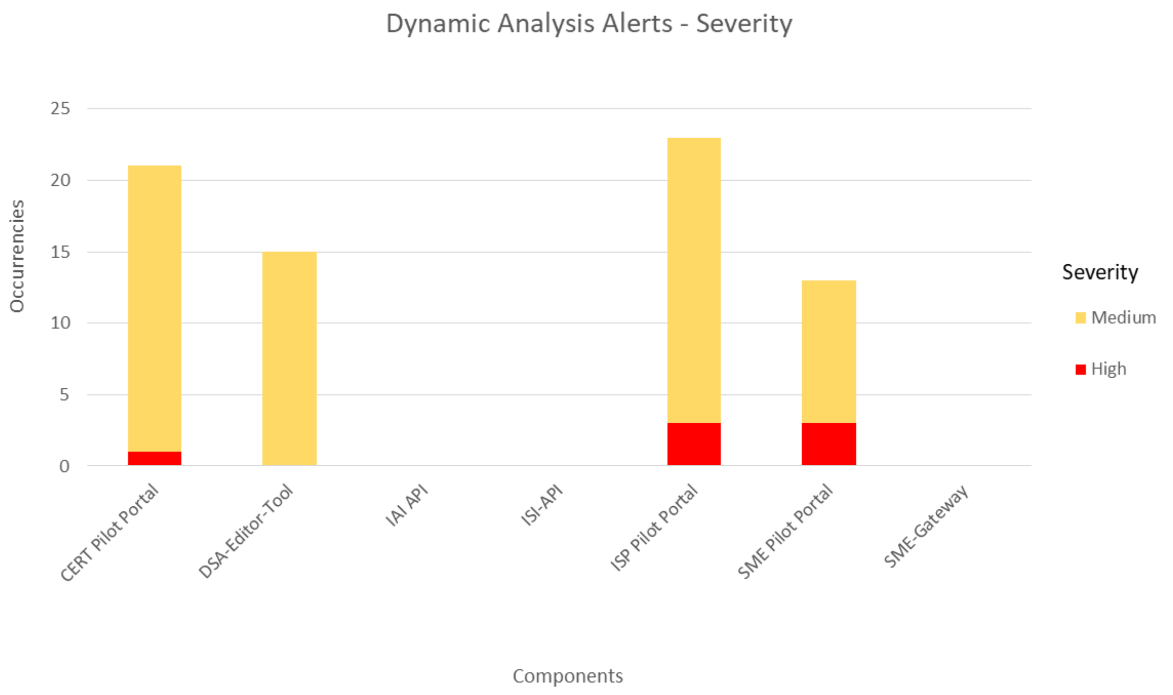


Figure 46: DAST - N. of issues by C3ISP Component – Severity High & Medium

Infrastructure Security Tests did not report major issues, thanks to our policy of continuous security patching of Operating System (OS) infrastructure (see introduction to Section 10), as shown in the next figure:

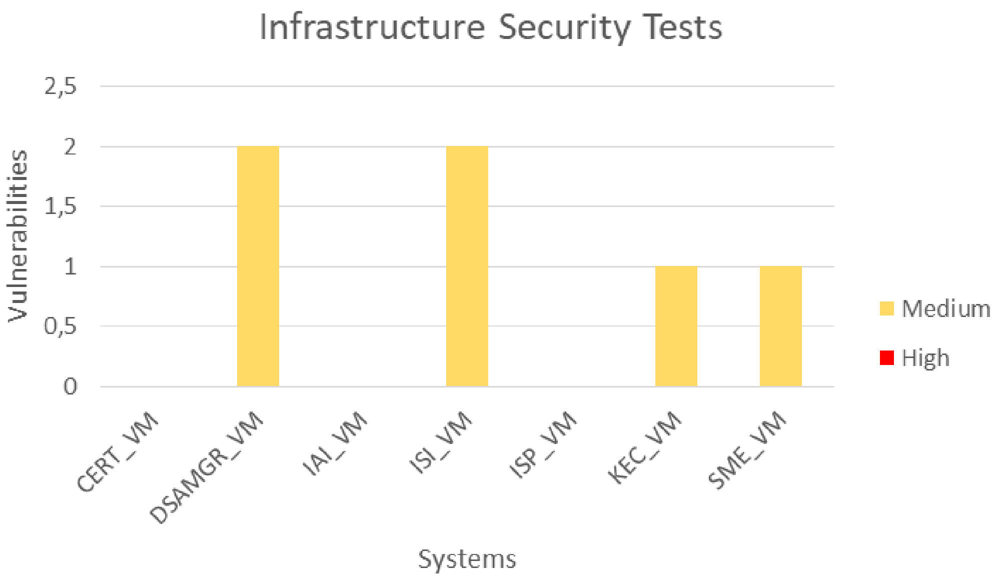


Figure 47: Infrastructure Security Assessment – N. of issues per VM

11. Conclusions

This document presented the final reference C3ISP architecture. Most of the updates were due to the feedback from the Pilots during their first validation period (M26). However, the architecture did not go under major re-design activities, but was only completed and minimally extended to support all the requirements initially stated (Section 9). As anticipated in the conclusions and future work of D7.3, we added the Service Usage Control Adapter component (Section 5.2), the Secure Audit Manager (Section 7.3), and extended the Identity Manager functionalities by introducing the OIDC Connect support (Section 7.1.2). As requested by EC we executed a security assessment for C3ISP (Section 10.4); detail results as well as the outcome of the remediation activities will be published in D7.5, which is marked as a confidential deliverable. We also extended the list of available DMOs (which are more in depth described in D8.3, including the new FPE and Pseudonymisation), thanks to the modular/plug-in based architecture of the DMO Engine. Finally we improved the packaging of C3ISP by using Docker containers, in particular for the Local ISI hybrid deployment model.

12. References

This section lists the references used throughout the document:

[1] K. Brennan, A Guide to the Business Analysis Body of Knowledge, International Institute of Business Analysis, 2009.

[2] STIX™ – Structured Threat Information Expression, https://oasis-open.github.io/cti-documentation/, https://stixproject.github.io/, fetched on March 16th, 2017

[3] Guide to Cyber Threat Information Sharing, NIST Special Publication 800-150, http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-150.pdf, fetched on March 16th, 2017

[4] CEF – Common Event Format, https://www.protect724.hpe.com/docs/DOC-1072, fetched on March 16th, 2017

[5] B. Kepes, Cloudonomics: the Economics of Cloud Computing, Rackspace Hosting, Diversity Limited, Aug. 2011

[6] Gartner, Gartner Says By 2020, a Corporate "No-Cloud" Policy Will Be as Rare as a "No-Internet" Policy Is Today, http://www.gartner.com/newsroom/id/3354117, Jun. 2016, fetched on March 16th, 2017

[7] CybOX™ – Cyber Observable eXpression, https://oasis-open.github.io/cti-documentation/, https://cyboxproject.github.io/, fetched on March 16th, 2017

[8] TAXII™ – Trusted Automated eXchange of Indicator Information, https://oasis-open.github.io/cti-documentation/, https://taxiiproject.github.io/, fetched on March 16th, 2017

[9] OpenC2 – Open Command and Control, http://openc2.org/, fetched on March 16th, 2017

[10] S, Carpov; T.H. Nguyen; R. Sirdey; G. Constantino; F. Martinelli; Practical Privacy-Preserving Medical Diagnosis Using Homomorphic Encryption

[11] S. Carpov, P. Dubrulle, R. Sirdey, "Armadillo: a compilation chain for privacy preserving applications", Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security (3rd International Workshop on Security in Cloud Computing), pp. 13-19, 2015

[12] Borghoff J. et al. (2012) PRINCE – A Low-Latency Block Cipher for Pervasive Computing Applications. In: Wang X., Sako K. (eds) Advances in Cryptology – ASIACRYPT 2012. ASIACRYPT 2012. Lecture Notes in Computer Science, vol 7658. Springer, Berlin, Heidelberg

[13] Albrecht M.R., Rechberger C., Schneider T., Tiessen T., Zohner M. (2015) Ciphers for MPC and FHE. In: Oswald E., Fischlin M. (eds) Advances in Cryptology -- EUROCRYPT 2015. EUROCRYPT 2015. Lecture Notes in Computer Science, vol 9056. Springer, Berlin, Heidelberg

[14] Canteaut A. et al. (2016) Stream Ciphers: A Practical Solution for Efficient Homomorphic-Ciphertext Compression. In: Peyrin T. (eds) Fast Software Encryption. FSE 2016. Lecture Notes in Computer Science, vol 9783. Springer, Berlin, Heidelberg

[15] ECRYPT - European Network of Excellence in Cryptology: The eSTREAM StreamCipher Project (2005).

[16] De Cannière C., Preneel B. (2008) Trivium. In: Robshaw M., Billet O. (eds) New Stream Cipher Designs. Lecture Notes in Computer Science, vol 4986. Springer, Berlin, Heidelberg

[17] N. Bouzerna, R. Sirdey, O. Stan, T.-H. Nguyen and P. Wolf, "An architecture for practical confidentiality-strengthened face authentication embedding homomorphic cryptography", Proceedings of the 8th IEEE International Conference on Cloud Computing Technology and Science, pp. 399-406, 2016.

[18] Junfeng Fan, Frederik Vercauteren: Somewhat Practical Fully Homomorphic Encryption. IACR Cryptology ePrint Archive 2012: 144 (2012)

[19] Leonardo Dagum and Ramesh Menon. 1998. OpenMP: An Industry-Standard API for Shared-Memory Programming. IEEE Comput. Sci. Eng. 5, 1 (January 1998), 46-55. DOI=http://dx.doi.org/10.1109/99.660313

[20] Berkeley Verification and Synthesis Research Center, A System for Sequential Synthesis and Verification https://bitbucket.org/alanmi/abc

[21] Stober, Thomas –Hansmann, Uwe “Agile Software Development: Best Practices for Large Software Development Projects” -Springer 2009

[22] Bass, Len Ingo Weber, Zhu Liming – Hansmann, Uwe DevOps: A Software Architect’s Perspective” –Addison-Wesley Professional 2015

[23] A Python module for creating JUnit XML test result documents, https://pypi.python.org/pypi/junit-xml, fetched on March 16th, 2017

[24] NIST Glossary of Key Information Security Terms, http://nvlpubs.nist.gov/nistpubs/ir/2013/NIST.IR.7298r2.pdf, fetched on March 16th, 2017

[25] Gartner Newsroom, http://www.gartner.com/newsroom/id/3354117, fetched on March 16th, 2017

[26] I. Matteucci, M. Petrocchi, and M. L. Sbodio. CNL4DSA: a Controlled Natural Language for Data Sharing Agreements. In SAC: Privacy on the Web Track. ACM, 2010. 21, 23, 52

[27] Grigoris Antoniou and Frank Van Harmelen. Web Ontology Language: OWL. In Handbook on Ontologies in Information Systems, pages 67–92. Springer, 2003. 18

[28] Definition of Document Oriented Database https://www.techopedia.com/definition/30329/document-oriented-database

[29] Original BSD syslog, https://tools.ietf.org/html/rfc3164

[30] RFC5424, https://tools.ietf.org/html/rfc5424

[31] “Common Event Format” , ArcSight, Inc. July 22, 2010 Revision 16, https://kc.mcafee.com/resources/sites/MCAFEE/content/live/CORP_KNOWLEDGEBASE/78000/KB78712/en_US/CEF_White_Paper_20100722.pdf

[32] Representational State Transfer (REST) http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm

[33] Unified Modeling Language - UML, http://www.uml.org/

[34] Pub, NIST FIPS. "197: Advanced encryption standard (AES)." Federal information processing standards publication 197.441 (2001): 0311.

[35] Fan, Junfeng, and Frederik Vercauteren. "Somewhat Practical Fully Homomorphic Encryption." *IACR Cryptology ePrint Archive* 2012 (2012): 144.

[36] Menezes, A. J., Van Oorschot, P. C., & Vanstone, S. A. (1996). *Handbook of applied cryptography*. CRC press.

[37] Gentry, Craig. "Fully homomorphic encryption using ideal lattices." *STOC*. Vol. 9. No. 2009. 2009.

[38] Barker, Elaine, et al. "Recommendation for key management part 1: General (revision 3)." *NIST special publication* 800.57 (2012): 1-147. Key Management Guidelines SP 800-57 Part 2, Best Practices for Key Management Organizations

[39] Key Management Guidelines SP 800-57 Part 3, Application-Specific Key Management Guidance

[40] Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM)

[41] Shirey, R. "RFC 2828–Internet security glossary, 2000." *URL: http://www.faqs.org/rfcs/rfc2828. Html* (2003).

[42] Oppliger, Rolf. *Contemporary cryptography*. Artech House, 2011.

[43] RFC 6239, Suite B Cryptographic Suites for Secure Shell (SSH)

[44] RFC 6379, Suite B Cryptographic Suites for Ipsec

[45] RFC 6460, Suite B Profile for Transport Layer Security (TLS)

[46] Lyubashevsky, Vadim, Chris Peikert, and Oded Regev. "On ideal lattices and learning with errors over rings." *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, Berlin, Heidelberg, 2010.

[47] Fontaine, Caroline, and Fabien Galand. "A survey of homomorphic encryption for non specialists." *EURASIP Journal on Information Security* 2007.1 (2007): 013801.

[48] Gentry, C., S. Halevi, and N. P. Smart. *Homomorphic evaluation of the AES circuit (updated implementation)*. IACR Cryptology ePrint Archive: Report 2012/099, https://eprint. iacr. org/2012/099. Pdf, 2015.

[49] *"FIPS PUB 140-2" (PDF). NIST. 2002-12-03. Retrieved 2017-03-31.*http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.140-2.pdf

[50] Canteaut, Anne, et al. "Stream ciphers: A practical solution for efficient homomorphic-ciphertext compression." *International Conference on Fast Software Encryption*. Springer Berlin Heidelberg, 2016.

[51] De Canniere, Christophe, and Bart Preneel. "Trivium." *New Stream Cipher Designs* (2008): 244-266.

[52] Amazon S3 Data Lake https://aws.amazon.com/big-data/data-lake-on-aws/

[53] JavaScript Object Notation official site: http://json.org/

[54] Hadoop Distributed File System (HDFS) URL: https://hadoop.apache.org/docs/r1.2.1/hdfs_user_guide.html

[55] OpenStack™ Swift URL: https://docs.openstack.org/swift/latest/

[56] MongoDB official web site: https://www.mongodb.com/

[57] RFC 6749, OAuth2: https://tools.ietf.org/html/rfc6749, http://oauth.net/2/

[58] Unity 3D: https://unity3d.com/

[59] A One-Time Password System: https://www.rfc-editor.org/rfc/rfc2289.txt

[60] Lightweight Directory Access Protocol: https://msdn.microsoft.com/en-us/library/aa367008(v=vs.85).aspx

[61] OpenID Specification: http://openid.net/developers/specs/

[62] Apache Hive, URL: https://hive.apache.org/

[63] Cloudera Impala, URL: https://www.cloudera.com/products/open-source/apache-hadoop/impala.html

[64] Jaehong Park, R. S. (2002). Towards usage control models: beyond traditional access control. SA CMA T, (pp. 57 - 64).

[65] Oppliger, Rolf. Contemporary cryptography. Artech House, 2011

[66] Fundamental modeling concepts (FMC), URL: http://www.fmc-modeling.org

[67] Sticky Policy: An Approach for Managing Privacy across Multiple Parties, URL: http://ieeexplore.ieee.org/document/5959137/

[68] PostgreSQL Database, URL: https://www.postgresql.org/

[69] Transport Layer Security, URL: https://tools.ietf.org/html/rfc5246

[70] LDAP over SSL, URL: https://social.technet.microsoft.com/wiki/contents/articles/2980.ldap-over-ssl-ldaps-certificate.aspx

[71] Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. Foundations and Trends in Theoretical Computer Science, 2014.

[72] Miguel E Andrés, Nicolás E Bordenabe, Konstantinos Chatzikokolakis, and Catuscia Palamidessi. *Geo-indistinguishability: Differential privacy for location-based systems.* In Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security, 2013

[73] MySQL Database, URL: https://www.mysql.com/it/

[74] Oracle DBMS, URL: https://www.oracle.com/database/index.html

[75] Roth, Cynthia Dwork and Aaron, The algorithmic foundations of differential privacy, Foundations and Trends in Theoretical Computer Science, 2014

[76] Spring Cloud Config, https://spring.io/projects/spring-cloud-config

[77] Docker Community Edition, https://docs.docker.com/install/

[78] Docker Compose, https://github.com/docker/compose/

[79] Open ID Connect, https://openid.net/connect/