D7.1

Requirements for C3ISP Architecture

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

C3ISP

Collaborative and Confidential Information Sharing and Analysis for Cyber Protection

Due date of deliverable: 31/03/2017
Actual submission date: 31/03/2017

31/03/2017

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: C. Gambardella, M. Manea, M. Belloni (HPE), T.H. Nguyen, V. Herbert (CEA), M. Shackleton (BT), G. Costantino, A. Saracino (CNR)

Approved by: List of reviewers Stefano Tranquillini, (CHINO), Francesco Di Cerbo (SAP)

Revision History

Version	Date	Name	Partner	Sections Affected / Comments
0.1	10-Jan-2017	M. Manea	HPE	Initial ToC
0.2	03-Mar-2017	M. Manea C. Gambardella	HPE	Initial draft version
0.3	04-Mar-2017	T.H. Nguyen, V. Herbert	CEA	Contribution to 2.1.2, 2.2.1 and 2.2.3 sections
0.4	05-Mar-2017	A. Sajjad	UNIKENT	Contribution to 2.2
0.5	06-Mar-2017	M. Shackleton	BT	Contribution to section 2
0.6	07-Mar-2017	G. Costantino A. Saracino	CNR	Contribution to 2.1.1, 2.2.2, 3.1.1 and 3.2.1 sections
0.7	15-Mar-2017	M. Manea C. Gambardella M. Belloni	HPE	Ready for internal review
0.8	28-Mar-2017	S. Tranquillini F. Di Cerbo	CHINO SAP	Internal Review by CHINO and SAP
1.0	31-Mar-2017	C. Gambardella M. Manea	HPE	Comments from internal review have been addressed

Executive Summary

The aim of deliverable 7.1 (D7.1) is to define the set of requirements for the C3ISP framework driven by the four pilots' scenario needs.

The deliverable provides an overview of how the C3ISP framework will work and analyses the functional requirements, declined in data sharing and data analytics requirements, and non-functional requirements, as security, the operational needs, the performance and the system usability.

D7.1 aims to collect the common requirements that will be considered in the core architecture design and implementation of the framework in order to propose a C3ISP reference architecture (first edition at M12), followed by a C3ISP reference implementation (first version at M24).

Furthermore, the deliverable discusses the requirements for the development and test bed environments that will host the C3ISP reference implementation which will be used by the pilots. In particular, it evaluates the partners' needs for developing the framework, like the setup of a continuous integration engine that will be used to test and deploy the C3ISP software artefacts. Further, it proposes procedures and means to foster the creation a high quality and secure system, like tools for discovering security issues, to enable standard code style guidelines and to support testing, in order to supply the pilots with a common and robust platform that will advance the project prototyping activities.

Table of contents

1. Introduction

The successful delivery of the C3ISP framework requires to carefully design and maintain a sound architecture throughout the project that is able to cope with different user requirements coming from the four pilots (ISP, CERT, ENT and SME Pilots[1]) proposed by the consortium. This document is the first milestone in this effort. In particular, based on the functional and non-functional (security, operational, performance and usability) requirements (NFRs) of the different business use cases, we define a *common set of design requirements* (at M6) to drive the definition of a coherent, generic and extensible architecture. The resulting *C3ISP reference architecture* (due at M12) will include the common requirements from the pilots while at the same time will decouple those that are pilots-specific instantiations (which will be implemented at pilot-level) from the core architecture.

This section introduces the C3ISP overall framework by describing a high-level overview of its objectives, what we mean by C3ISP framework and the strategy we used for requirements elicitation.

1.1. Overview

The C3ISP project wants to address the need of different stakeholders (called prosumers, i.e. producer and consumer at the same time) by developing a collaborative and confidential information sharing, analysis and protection framework for cyber security management.

The core C3ISP idea is that only by enabling the setup of a prosumers' federation for cyber security related data exchange it will be possible to improve the overall security posture of a participant. In particular, this federation develops through different enablers that interoperate between them:

- A data sharing infrastructure, to support the data sharing among the prosumers (called *Information Sharing Infrastructure* – ISI);
- A data analytics infrastructure, to enable the analysis on the shared data and the visualization of the analytics services results (called *Information Analytics Infrastructure* – IAI).

In this document we describe the C3ISP requirements that will be used to design the platform architecture for data sharing and analytics, considering both functional and non-functional requirements. In particular, special attention is given to security requirements (e.g. confidentiality of data sharing), operational requirements (e.g. extensibility and interoperability), performance (e.g. due to the planned usage of computational intensive homomorphic encryption mechanisms) and usability (e.g. of the tools that will present the analytics results).

These requirements will support the achievement of the C3ISP goals by allowing the consortium to design and then implement the C3ISP framework described next, as the project develops through its timeline.

1.2. C3ISP Framework

One of the main outcome of the project will be the *C3ISP framework*. This framework is the implementation of the *C3ISP reference architecture*, whose requirements are described in this document. The C3ISP reference architecture is the set of designed subsystems, components, and modules, as well as their interaction within the boundaries of the overall system and with

[1] Refer to D6.1 and specific pilot's deliverables: D2.1, D3.1, D4.1 and D5.1

the external systems it needs to interact with, but that are not part of the architecture (e.g. a data feed provider, or an authentication system).

In fact the C3ISP framework helps the pilot business cases, in particular the prosumers (i.e. the system actors), to *share cyber-security related data* in a way that, on one hand, is both confidential and privacy-preserving, but on the other, thanks to the established collaborative federation, makes use of *security analytics services* that help fighting (i.e. discover and react) against cyber-security threats.

This document defines what kind of cyber-security data the framework handles (i.e. the *resources*), by considering the information the pilots deliver (e.g. security logs) and the security and privacy constraints under which the data has to be used, especially considering the open yet controlled nature of a data sharing federation.

It is worth saying that the generation, collection and submission of the prosumer's data is a pilot-specific process that is outside of the C3ISP framework, but capabilities to handle the submission will be provided. While the cyber-security data is one of the inputs to the framework, another important input is the data sharing policies, agreed within the prosumers' federation to regulate their exchange. We plan to encode these policies in *multi-lateral data sharing agreements* (DSAs) to express the security and privacy requirements for the cyber-security data sharing. A DSA also encodes rules regarding the analytics services that can be run on the shared data and defines constraints about *data manipulation operations* (DMOs) performed on the data either before the computation, or after the computation on the resulting output, in particular anonymisation or homomorphic encryption operations which shall be used to find a balance between privacy requirements of the prosumer and accuracy on the analytics result. The DSA can also contain sharing policies on the analytics services output, thus controlling the dissemination of the generated insight much the same way as if it were input data.

The idea of the prosumers federation is that only by combining data from different entities or organisations it is possible to discover security threats or attacks that would go otherwise unnoticed. For example, Advanced Persistent Threats [24] is a class of attack that might go unnoticed on a single prosumer data, but might be revealed as a coordinated attack on a bigger data set from the federation.

For the analytics services, the deployment of the C3ISP framework will provide a (Big) Data Lake reference deployment, to accommodate the different pilots' requirement that might or not have it, according to their needs; for example small organizations might use the Data Lake provided by the framework, instead others bigger may prefer to leverage on their own. This is a component external to the framework that is however necessary to make the system useful. In particular, we aim at using a Data Lake based on standard technologies (e.g. Apache Hadoop) to allow already existent analytical tools to operate on the data sets with minimal to no changes (in particular the Enterprise Pilot has specific requirements about that): to address this requirement, still allowing a controlled data sharing, we are thinking about defining a virtual/sanitised Data Lake to be used by third-party out-of-the-box analytical applications, probably with some limitations on the rules we can setup in the DSAs in these cases.

Also the C3ISP framework will allow the definition of advanced Visual Analytics services that will render the data under the constraints of the DSA. The service will enable users with (security) domain knowledge to perform data analytics via interactive data exploration and visualisation. An artificial intelligence layer will allow structured and non-structured data to be analysed in order to discover patterns and to generate new levels of insight and knowledge for existing data. Users will be able to interactively filter the data, based on temporal, spatial, or logical clusters, in order to explore and drill down into the data to find patterns, anomalies, or other items of interest. The Visual Analytics capability will be combined with C3ISP preservation and transformation components, thereby integrating various data sources such as from the homomorphic encryption module, the managed security services and other entities such as CERT or external analytics tools.

1.3. Requirements Naming Convention

This document reports each C3ISP framework requirement indexed with an identifier to make it easy to trace its fulfilment in the next phases of the project, as well as with a priority that follows the MoSCoW [1] scale. In fact, the architecture will be designed to address the requirements by considering the different priorities assigned. Further, each requirement is also tagged, where possible, with the identifier(s) of pilot's requirement it originates from (as from D6.1), such in a way that the whole requirements chain can be rebuilt when we will discuss the architecture in the next D7.1.

The requirements naming convention follows this format:

C3ISP-[ReqClass]-[Id]

Where [ReqClass] = **Fun** (Functional, further split in DS=Data Sharing and DA=Data Analytics, due to their importance), **Sec** (Security), **Ope** (Operational), **Per** (Performance), **Usa** (Usability), **Dev** (Development environment), **Tst** (Test Bed environment).

E.g.:

- C3ISP-Fun-DS-001 for C3ISP Functional requirement no. 001 for Data Sharing;
- C3ISP-Sec-001 for C3ISP Non-Functional requirement no. 001 for Security.

In case subsections are present for each class of requirement, we adjust the numbering. E.g. Information Security Requirements→C3ISP-Sec-001; Regulatory Requirements→C3ISP-Sec-101, because both are part of the Security Requirement class and so we can accommodate new requirements during the on-going work of requirement elicitation and refinement.

The requirements naming convention also allows us to trace the requirements across different deliverables should this be necessary.

1.4. Deliverable Structure

The document is structured as follows:

- After this introduction, Chapter 2 describes the requirements for the C3ISP framework, by considering both functional and non-functional ones;
- Chapter 3 illustrates the requirements for the C3ISP development and test bed environment that will be created starting from M6;
- We conclude this deliverable with Chapter 4 that describes the activities and the steps for the next period.

1.5. 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
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
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 Sharing Infrastructure
IDE	Integrated Development Environment
IDS	Intrusion Detection System
IP	Internet Protocol
ISI	Information Analytics 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
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 services
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. Framework Requirements

This section describes the C3ISP framework requirements elicited from the four pilots. They are split between functional and non-functional ones. Since C3ISP is a security platform, it is obvious that the NFRs are highly important and to some extent they represent not only properties of the system, but also important functionalities it has to provide, so there is an overlap between the two requirements set. For this reason, we strived to keep on the first class what is strictly functional and move to the non-functional the security features of the system. For example, the concept of homomorphic encryption has both functional and non-functional impacts, and even if we introduce it in the security requirements section (NFR), some constrains are also present in the data analytics functional requirements section.

Each subsection includes the objective of the requirements set it deals with and provides a table with the requirements list that follows the format described in 1.3.

2.1. Functional Requirements

We split the functional requirements in two main areas related to the core C3ISP functionalities: **data sharing** and **data analytics**. For data sharing we mean the exchange of security-related information provided by the C3ISP prosumers that participate to the federation setup by the Data Sharing Agreements. This includes also the capabilities to manage the DSA lifecycle (e.g. editing, termination, etc.). With data analytics we intend tools and techniques that will make use of the prosumers' shared data to infer knowledge useful to identify and possibly mitigate cyber-attacks.

2.1.1. Data Sharing Requirements

In a data sharing scenario, it is of outmost importance to correctly identify what are the resources into play. From each specific Pilot use case, C3ISP must pinpoint what kind of data to be shared and protected and with what level of granularity (e.g. a record in a database or a file).

Broadly speaking, C3ISP aims at ensuring the security and privacy of any kind of cybersecurity-related data shared by prosumers. Such kind of information could take the form of data files, multimedia streams, database entries and textual logs. However, this raw information can and shall be encoded in a standard structured format, which enables an easier and organized representation of the information in the C3ISP framework. In C3ISP, any piece of data is related and bound to a specific DSA which specifies the security requirements for that particular data piece.

We outline the following resource specifications:

- As anticipated, C3ISP aims at protecting virtually any kind of information, independently from the format and the specific content. Most of information handled by the C3ISP framework falls in the category of CTI (Cyber Threat Information [3][2]), still any piece of data which can directly or indirectly help in describing a vulnerability, attack or countermeasure, is considered as an asset for analysis.
- Protection of shared information can be ensured by use of standard cryptography techniques, for secure data storage, sanitization techniques, such as anonymization and generalization to preserve privacy, and homomorphic encryption when it is a

[2] *(Cyber) "Threat information is any information related to a threat that might help an organization protect itself against a threat or detect the activities of an actor"*

requirement to maintain data secrecy also for the information analysis infrastructure. The specific security requirements for a piece of data are described in the DSA, together with the specification of the prosumers which might receive the data itself, or can receive the analysis results performed on the data piece.

According to the use cases defined in D6.1, resources to be shared and protected by C3ISP framework are of different types. The following table summarises the resource types involved in the specific pilot use cases:

Resource Type	Pilot Use Case Id
Security Log File contains security evaluation (in Common Event Format [4])	ISP-US-1
Network and systems, security appliances, software monitoring	ENT-US-1
Security event data	ENT-US-2
Security Threats, attacks or vulnerabilities	CERT-US-1, CERT-US-6

In a generalized approach, we can conclude that all the resources involved are CTI[3] with different peculiarities depending on the pilots and the contexts in which they operate.

To enable the management of different kind of data on C3ISP, we evaluated a common format for describing CTI. The STIX [2] (Structured Threat Information eXpression) standard, sponsored by the OASIS Cyber Threat Intelligence Technical Committee, has been considered suitable for describing the identified resource types, since the formats supported by the standard cover the information involved in all the pilots scenarios. STIX mission is to "enable organizations to share CTI with one another in a consistent and machine readable manner, allowing security communities to better understand what computer-based attacks they are most likely to see and to anticipate and/or respond to those attacks faster and more effectively". More on STIX architecture is reported in 2.2.2.2.

Also referring to D6.1, we have extrapolated and consolidated a set of requirements concerning data sharing, how pilots expect to define security policies that regulate the data access and usage, the need to have evidence of the effective application of the defined policies and a set of collateral data sharing requirements (pre-processing of data before sharing operations, post sharing notifications, etc.).

The requirements are summarized in the following table, in which is indicated the correspondent pilot's use case from which it was derived. The MoSCoW notation is used to prioritise them, according to the pilot's classification.

Table 1 – Data Sharing Requirements

ID	Goal	Priority	Requirement
C3ISP-Fun-DS-001	ISP-US-2 CERT-US-2 ENT-UC-2 SME-US-4	MUST	C3ISP allows defining Data Sharing Agreements between parties that want to exchange CTI data

[3] As detailed in the "Classification of Requirements" in D6.1

C3ISP-Fun-DS-002	ISP-US-1 CERT-US-2 ENT-US-2 SME-US-4 SME-US-11	MUST	C3ISP allows the sharing of files (including log data, threat intelligence data, analysis reports)
C3ISP-Fun-DS-003	ISP-UC-5 CERT-US-5 ENT-US-2 SME-US-2	MUST	C3ISP grants prosumers the control over the sharing of data (i.e. prosumers have both tools and functionalities to specify “constraints” that regulate the data sharing process)
C3ISP-Fun-DS-004	ISP-US-5 SME-US-2	MUST	C3ISP allows controlling the process of data sharing at file level
C3ISP-Fun-DS-005	ISP-US-5 ENT-US-1 CERT-US-2 SME-US-2	MUST	C3ISP allows defining policies (i.e. a set of rules) that regulate the data sharing process
C3ISP-Fun-DS-006	ISP-US-5 ENT-US-1 SME-US-4 CERT-US-5	MUST	C3ISP policies allow access control to the shared data (i.e. define conditions to be verified before accessing the data)
C3ISP-Fun-DS-007	ISP-US-5 ISP-US-2 CERT-US-3 ENT-US-2 SME-US-4	MUST	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)
C3ISP-Fun-DS-008	ENT-US-1 SME-US-4	SHOULD	C3ISP policies allow defining rules that can evaluate contextual information (i.e. information from the environment/use case)
C3ISP-Fun-DS-009	SME-US-10 SME-US-11	MUST	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.
C3ISP-Fun-DS-010	ENT-US-2 CERT-US-2	MUST	C3ISP provides evidences (e.g. audit logs) of the compliance to the sharing policies enforcement
C3ISP-Fun-DS-011	ENT-US-2 SME-US-5	SHOULD	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 2.1.3) for minimising

			sensitive data exchange; (ii) encryption mechanisms (see 2.2.1).
C3ISP-Fun-DS-012	ENT-US-2 SME-US-7	COULD	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)
C3ISP-Fun-DS-013	SME-US-2	COULD	C3ISP could use an open and/or standard policy description language for data sharing (DSA/XACML)
C3ISP-Fun-DS-014	CERT-US-5	COULD	C3ISP allows defining multi-lateral Data Sharing Agreements, i.e. DSA between multiple prosumers (two or more)

2.1.2. Data Analytics Requirements

The C3ISP framework will provide **data analytics as a service** to support the pilots in detecting ongoing attacks and deriving intelligence to prevent against potential (future) attacks. As more and more relevant information is shared between the prosumers, the analytics service gains higher importance as attacks can be detected more accurately and predicted earlier. Some examples of such analytics services are briefly described as follows:

- **Domain hijacking**: Domain hijacking is an impersonation of a domain owner with the aim of stealing a domain name and related services. The C3ISP service analyses the communication between the attacker and a registrar in order to reveal traffic patterns that can be used to identify and prevent further attacks (at different registrar);
- **Distributed Denial of Service (DDoS)**: Multiple websites that are targeted together by DDoS attacks are likely registered and hosted by different registrars. The C3ISP service analyses the DDoS traffic and provides intelligence to help registrars better differentiate legitimate Web traffic from requests that are part of the DDoS attack;
- **Malware spreading**: Malware commonly spreads as email attachments. Relying on log analysis, the C3ISP service creates profiles of the malicious emails (e.g. sender, email body) and their attachments (e.g. document name) in order to support mail servers block them and prevent further spreading;
- **Malicious port scanning**: Port scanning is used by attackers to discover open ports and vulnerable services to exploit at a target machine. The C3ISP service detects new malicious sources from the information shared in C3ISP Data Lake (e.g. blacklisted IPs) and sends notification alerts to interested stakeholders/prosumers;
- **Periodic beaconing**: A beacon is traffic leaving the network at regular intervals. It can be used to communicate with a Command and Control (C&C) server. The C3ISP service analyses the traffic patterns and identifies the potential C&C server hosts that are commonly shared in the information provided by prosumers and sends notification alerts;

- **Drive-by-Download**: It is a technique to inject malware when a visitor navigates to a malicious website. The C3ISP service analyses the outbound web traffic data shared by different prosumers and identifies websites (i.e. URLs) that have potentially caused malware infection;
- **Stealthy attack pattern**: A stealthy attack normally remains undetected by conventional security systems such as IDS (Intrusion Detection System) or Firewall. It is using stealthy penetration and observation techniques to discover the victim's vulnerabilities and determine the locations of sensitive data on victim's network. Nevertheless the perpetrator may use the same methods on different targets and follow a specific traffic pattern. The C3ISP service analyses the aggregated prosumers' shared data to detect such stealthy attack patterns and notifies the affected prosumers.

In general the C3ISP framework defines two types of data operation that can be performed on information shared between the prosumers: (1) **data manipulation operations (DMOs)**, and (2) **analytics operations**. Data manipulation operations are mainly used to pre-process the information before or after its usage in order to make it usable for further processing, or to comply with the associated sharing policy. Examples are data anonymization, homomorphic encryption, data conversion, etc. Analytics operations are used to analyse the (aggregated) data and extract intelligence related to security attacks, threats and vulnerabilities. Examples includes the analytics services described at the beginning of this sections, which uses techniques for anomaly detection, data correlation, data visualisation, etc.

The data analytics requirements presented in this section apply to both types of operation. We also use the term *privacy-preserving operation* to describe such data manipulation operation that specifically aims at removing or protecting sensitive information, e.g. homomorphic encryption. Table 2 summarises the requirements derived from the correspondent pilot's user stories and use cases; they are prioritised using the MoSCoW notation. Main requirement is to enable prosumers define DSA policies for controlling how and which analytics operations should be allowed on their data. The pilots also set requirements on which capability the operations should provide (e.g. threat classification), as well as how the analytics results should be represented and communicated back to the prosumers.

Table 2 – Data Analytics Requirements

ID	Goal	Priority	Requirement
C3ISP-Fun-DA-001	ISP-US-2 ISP-US-5 CERT-US-3 ENT-US-1	MUST	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
C3ISP-Fun-DA-002	ENT-US-1 ENT-US-2	MUST	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 2.1.3)

C3ISP-Fun-DA-003	ISP-US-1 ISP-US-2 ENT-US-4 SME-US-5 SME-US-6	MUST	C3ISP provides a programming interface for executing DMOs, such as privacy-preserving operations on data (e.g. data sanitisation or encryption)
C3ISP-Fun-DA-004	SME-US-5	MUST	C3ISP allows executing privacy-preserving DMOs on all or part of the data
C3ISP-Fun-DA-005	ISP-US-2 CERT-US-3 ENT-US-1 ENT-US-3	MUST	C3ISP provides a programming interface for supporting the analysis on the data stored in C3ISP data lake, in compliance with the associated DSA policies
C3ISP-Fun-DA-006	CERT-US-1 CERT-US-2 CERT-US-3 CERT-US-4 CERT-US-5 ENT-US-3 ENT-US-4 SME-US-9	MUST	C3ISP provides a function to query data and analytics operation results that are stored in C3ISP data lake in compliance with the DSA policies
C3ISP-Fun-DA-007	ENT-US-4	MUST	C3ISP supports standard query language (e.g. SQL) for querying data and analytics operation results from C3ISP data lake
C3ISP-Fun-DA-008	CERT-US-6 SME-US-9	MUST	C3ISP provides a function for automatic threat classification of analytics operation results
C3ISP-Fun-DA-009	CERT-US-6 SME-US-4	MUST	C3ISP provides a function for automated mapping of analytics operation results to interested stakeholders/prosumers that are specified in the DSA
C3ISP-Fun-DA-010	ENT-US-1 SME-US-3 SME-US-4 SME-US-9	MUST	C3ISP provides a function to convert analytics operation results to standardised and machine-readable formats (e.g. STIX) in compliance with the DSA
C3ISP-Fun-DA-011	ENT-US-3 ENT-US-4	MUST	C3ISP provides an 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)

C3ISP-Fun-DA-012	SME-US-4	MUST	C3ISP provides near real-time notifications of analytics operation results (see also C3ISP-Fun-DS-009)
C3ISP-Fun-DA-013	SME-US-9 SME-UC-4	SHOULD	C3ISP provides a function to query analytics operation results of specific categories (e.g. malware analysis, attack on cloud platform)
C3ISP-Fun-DA-014	SME-US-9	SHOULD	C3ISP supports different categories for analytics operations results, i.e. threat types, threat risks, threat origins, threat costs, regulatory and compliance concerns
C3ISP-Fun-DA-015	SME-UC-4 SME-US-10	COULD	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 2.2.2.
C3ISP-Fun-DA-016	SME-UC-4	COULD	C3ISP provides a dashboard showing status and results of the analysis
C3ISP-Fun-DA-017	SME-US-10	SHOULD	C3ISP allows scheduling of the provisioning of analytics operation results (e.g. on demand, periodical, etc.)
C3ISP-Fun-DA-018	SME-US-5 SME-US-6 SME-UC-3	MUST	When using homomorphic encryption (see 2.2.1.1), before data analytics execution, data is represented as bits or integers.
C3ISP-Fun-DA-019	SME-US-5 SME-US-6 SME-UC-3	MUST	When using homomorphic encryption (see 2.2.1.1), 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.
C3ISP-Fun-DA-020	SME-US-5 SME-US-6 SME-UC-3	MUST	When using homomorphic encryption (see 2.2.1.1), analytics operands (cipher-texts) are encrypted bits (most current case) or encrypted integers with considered homomorphic cryptosystems.
C3ISP-Fun-DA-021	SME-US-5 SME-US-6 SME-UC-3	MUST	When using homomorphic encryption (see 2.2.1.1), 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.
C3ISP-Fun-DA-022	SME-US-5 SME-US-6 SME-UC-3	MUST	When using homomorphic encryption (see 2.2.1.1), analytics computation on encrypted bits is represented as a Boolean circuit with multiplicative depth[4] roughly 20 or 30.
C3ISP-Fun-DA-023	SME-US-5 SME-US-6 SME-UC-3	SHOULD	When using homomorphic encryption (see 2.2.1.1), the number of multiplicative gates should be minimized to decrease latency of homomorphic evaluation.

2.1.3. Data Manipulation Operations

In the context of the DMOs described for C3ISP-Fun-DS-011 and C3ISP-Fun-DA-002, possible anonymization techniques that find application in the pilots' use cases are:

- Suppression of identifiers (e.g. names);
- Generalization of values in certain finite domains (e.g. subnet masking);
- Randomization methods that anonymize individual values (and thereby one's membership in the data set) in such a way that accurate aggregates for certain functions (e.g. mean) can be produced when enough data is provided.

DMOs includes also homomorphic encryption described in 2.2.1.

2.2. Non-Functional Requirements

NFRs address the goal of building "quality" into the system. Being C3ISP a security platform, this kind of requirements is particularly important. The requirements are split in:

- **Security**: in addition to IT security requirements, this class includes needs from the regulatory space that the pilots have to obey;
- **Operational**: they deal with constraints or necessities under which the pilots have to operate;
- **Performance**: since C3ISP uses analytical platform services, this class is important as well, especially considering the advanced encryption techniques we are using that have high memory and computation needs;
- **Usability**: the C3ISP framework services requirements for addressing effectiveness and efficiency of use by the C3ISP users.

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

2.2.1. Security Requirements

The standard security requirements of **confidentiality**, **integrity** and **availability** (CIA-triad) are at the core of C3ISP data-centric protection vision. In particular, in the context of C3ISP, the shared data shall:

- Maintain confidentiality, the property of protecting the secrecy of data and disclose only to authorised parties under the policies specified in their DSAs. Specifically, encryption has been traditionally used for preserving confidentiality, and just lately homomorphic encryption has started emerging as a novel approach for this goal;
- Integrity, the fact of being able to preserve the data such that it does not get altered fraudulently (conversely, C3ISP-protected data gets altered based on the DMOs regulated by the DSAs policies);
- Availability, the capability to have the data, including the C3ISP analytics results, available when they are needed or requested (this is important since results could drive the react phase to mitigate a discovered attack).

In addition to these traditional security requirements, also **non-repudiation**, **authentication**, **authorisation**, and **accountability** are among the top C3ISP priorities because of the data sharing. In particular:

- Non-repudiation: the fact that one party cannot deny of having submitted data to the C3ISP federation, can help as a deterrent countermeasure to limit a malicious party to submit bad data;
- Authentication and authorisation: both the data sharing and the consumption of analytics results have to be protected by access control mechanisms and conditions. DSAs regulate how access and usage control protect the cyber threat information;
- Accountability: especially to address compliance mandatory requirements or to help internal investigations, assess the correctness of system processing, etc., C3ISP has to be able to trace and identify the right entities or people that participate in the DSA-regulated federation and be able to understand that the policies stated have been correctly and effectively enforced.

The next section introduces homomorphic encryption.

2.2.1.1. Fully Homomorphic Encryption (FHE)

In this section, we provide an overview of what FHE is as well as of its benefits and limitations. The FHE addresses data privacy issues and regulatory requirements; for this reason we consider it as an innovative security requirement.

In a nutshell, Fully Homomorphic Encryption is a new kind of cryptographic techniques, which on top of allowing the scrambling of data in order to protect their confidentiality, also provides the necessary mathematical building blocks for the execution of general algorithms directly on encrypted data (for example, we can make addition, multiplication, division for comparison purpose, and subtraction). As such, FHE is a unique ground breaking software-only technology allowing to enforce the confidentiality of data when they are manipulated by untrusted servers without decryption and without disclosing any secret to those servers.

The ability to compute directly over encrypted data results in the ability for a computer (depicted below as Charlie) to do something useful with the data of an end user (depicted below as Alice) using additional data from one or more providers (depicted below as Bob). In doing so, both Alice's and Bob's data remain confidential with respect to Charlie which manipulates them only in encrypted form and, thus, neither has access to these data in clear form nor is provided with any decryption capability. In the setting below, any by-product of Charlie's calculations (be it intermediate or final calculation results) remains sealed under Alice's homomorphic cryptosystem who, as the owner of that cryptosystem secret key, is the only party able to retrieve the intelligibility of Charlie's outputs.

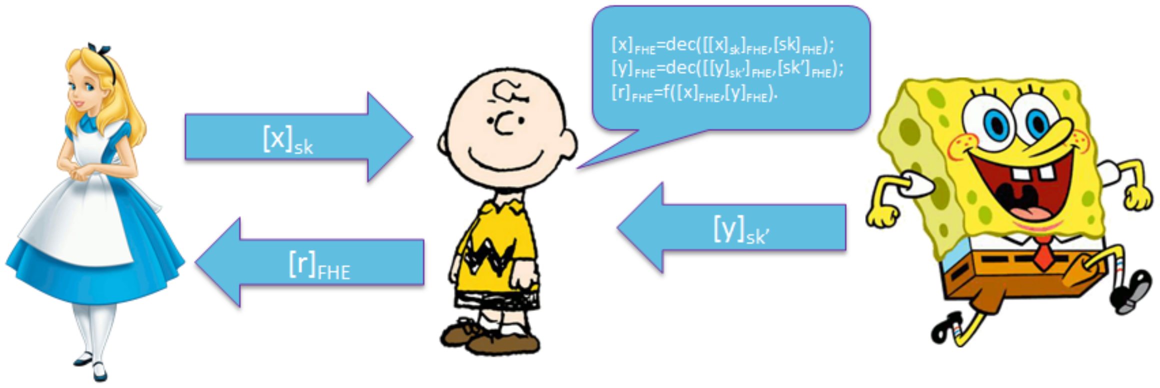
$[x]_{FHE}=dec([[x]_{sk}]_{FHE},[sk]_{FHE})$;
$[y]_{FHE}=dec([[y]_{sk'}]_{FHE},[sk']_{FHE})$;
$[r]_{FHE}=f([x]_{FHE},[y]_{FHE})$.
$[x]_{sk}$
$[r]_{FHE}$
$[y]_{sk'}$

Figure 1: Alice's (left) and Bob's (right) data remain confidential with respect to Charlie (middle) data manipulation operations.

This capability allows to imagine a number of settings where users can benefit from services taking into account their privacy-critical data, still without effectively giving them away.

Among these are:

- Undisclosed cross-valorisation of data (and algorithms): where it becomes possible for an algorithm to interact with some data with this interaction implying neither the disclosure of the algorithm to the data owner, nor the disclosure of the data to the algorithm owner.
- Intrinsic data protection on vulnerable platforms: where it becomes possible to store sensitive data (e.g. medical or biometric data) on e.g. a computing platforms connected to the Internet (hence intrinsically more vulnerable) while keeping an intrinsic protection layer on their confidentiality.
- Privacy-preserving outsourcing: where it becomes possible to store data on an untrusted server (with respect to confidentiality, i.e. in the honest-but-curious threat model [10]) while still preserving an ability to do more than just retrieving them.

Using FHE technology, we can clearly protect data and confidentiality with very high security level. But there are a number of issue with respect to transmitting FHE-encrypted data, mainly:

- FHE-encryption is a computationally heavy operation;
- FHE-encrypted data are much larger than their associated plaintexts. As an example, let us compare two encrypted IPv4 addresses (32 bits). First we encrypt them bitwise, that is we encrypt 2*32 bits; the size of each cipher text is 82 KiB. Be conscious, this figure depends on many parameters such as the employed cryptosystem, the desired security level and the computations we want to do with cipher texts.

However, FHE is quite powerful and allows to perform a "trick" known as trans-ciphering, by means of which it is possible to switch from data encrypted with some cryptosystem (e.g. a "classic" overhead-free symmetric cryptosystem) to the same data encrypted under FHE, *without this data to ever be in clear form*. The following figure illustrates this principle assuming that the initial cryptosystem is the AES:

$$\text{If } AES^{-1}([x]_{key}, key) = x$$

$$\text{then}$$

$$AES^{-1}([[x]_{key}]_{FHE}, [key]_{FHE}) = [x]_{FHE}$$

Figure 2: Trans-ciphering AES to FHE

However, homomorphically executing an AES decryption still takes 18 mins with our best implementation [11]. And this is intrinsic as the algorithm has a *multiplicative depth* of 40, which is quite large. Hence, more homomorphically-friendly symmetric systems are required and this is precisely on this issue that we have focused on the first year of the project.

In particular, we have studied how stream ciphers could help. Indeed, when a block-cipher usually is a relatively low degree function iterated a significant number of times (e.g. 10 times for AES-128 or more to the notable exception of PRINCE [12] and the more recent LOWMC [13]), a design which is intrinsically not FHE-friendly, stream ciphers (when not based on block-ciphers) follow different design patterns, some of them "friendlier" for efficient FHE execution.

So what we need is a stream cipher where *keystream bits* must be multiplicatively bounded. This is the case if keystream bits are independent by chunks (which is good for parallelism and batching). Also, when using a stream cipher, keystream bits can be homomorphically «mined» independently of the data. Hence, trans-ciphering induces almost no latency (it is just a homomorphic XOR) as long as keystream mining has been done in advance. So, we turned to the basic pattern of using an IV-based (FHE-friendly) stream cipher in «counter mode». Usually, in cryptography counter mode turns a block cipher into a stream cipher. In our context, counter mode is used with stream ciphers during keystream generation (refer to [14], page 5 for detailed information).

With that respect, we did an analysis [14] of all the finalists of the recent ESTREAM [15] stream cipher design competition and found that the TRIVIUM [16] algorithm was a very good candidate as a respected 80-bits key lightweight stream cipher. Still, in order to increase the overall key-length to a larger 128-bits, we contributed to the design of a 128-bits key extension of TRIVIUM, KREYVIUM, which also retains the FHE-friendliness.

2.2.1.2. *Information Security Requirements*

We outline below the requirements that are specific to maintain the security properties of the data that C3ISP will manage.

Table 3 – Information Security Requirements

ID	Goal	Priority	Requirement
C3ISP-Sec-001	ISP-NFR-05 SME-US-12	MUST	There is mutual authentication carried out between the C3ISP framework and

			the Prosumers at the start of any communication.
C3ISP-Sec-002	ISP-NFR-05 SME-NFR-4	MUST	Confidentiality and Integrity of the Data Sharing Agreement communications between the Prosumers and C3ISP Service is guaranteed.
C3ISP-Sec-003	ISP-NFR-05 SME-NFR-5	MUST	The transfer of CTI from the Prosumers to the C3ISP framework is secure (w.r.t. confidentiality and integrity).
C3ISP-Sec-004	SME-NFR-6	COULD	The integrity of the CTI data stored on the C3ISP framework is maintained.
C3ISP-Sec-005	ISP-NFR-05 SME-NFR-7	MUST	The transfer of analysis results from the C3ISP framework to the Prosumers is secure (w.r.t. confidentiality and integrity).
C3ISP-Sec-006	SME-US-5 SME-US-6 SME-UC-3	SHOULD	C3ISP Service is able to process anonymised or homomorphically encrypted CTI shared with it by the Prosumers.
C3ISP-Sec-007	SME-US-5 SME-US-6 SME-UC-3	SHOULD	Minimum security level is at least 80 bits (security strength). See discussion on FHE in section 2.2.1.1.
C3ISP-Sec-008	SME-US-5 SME-US-6 SME-UC-3	MUST	Maximum security level is at most 128 bits (computational efficiency), for real world scenarios. See discussion on FHE in section 2.2.1.1.
C3ISP-Sec-009	SME-US-5 SME-US-6 SME-UC-3	MUST	The homomorphic encryption uses randomization methods (see section 2.1.3). It is required to have semantic security. That is, it should be hard to distinguish between the encryption of any two messages, even if the public key is known to the attacker and even if the two messages are chosen by the attacker (chosen plaintext attacks). (In return, cipher-text size is greater than plaintext size). See discussion on FHE in section 2.2.1.1.

2.2.1.3. Regulatory Requirements

This section specifies requirements due to compulsory law obligations or industrial standards, as determined by the pilots' needs. We chose to be specific, e.g. instead of specifying generically a GDPR compliancy requirement, we tried to set more concrete requirements that can be "easily" addressed during the project development.

More specifically, according to the reference context, pilots need for a certain regulatory compliance. In the CERT pilot the knowledge of regulation and policies on data privacy, defined by law authorities or data providers are a prerequisite for the data collection and manipulation (ref. CERT-UC-2). Also for the SME pilot, concerns about the regulatory compliance are considered in the SME-UC-4, when the scenario about the filtering of relevant information for the SME from the analysed shared CTI is described.

Moreover, in the ENT pilot, the threat intelligence feeds involved in the sharing can be proprietary and/or subject to licensing restrictions. Constraints about confidentiality and data usage referring to legal and ethical regulation need to be considered in the data sharing and analysis (ENT-US-1). Further, customers specify policies governing about how its data may be used; they need to have evidence of the enforcement of the policies and of the degree of data confidentiality and integrity.

Table 4 – Regulatory Requirements

ID	Goal	Priority	Requirement
C3ISP-Sec-101	SME-US-1	MUST	The Prosumers are given the details of their data's lifecycle at the C3ISP framework. (GDPR Requirement)
C3ISP-Sec-102	SME-US-1	MUST	The Prosumers are able to reject or cancel the terms and conditions of their data sharing agreement with the C3ISP framework at any time. (GDPR Requirement)
C3ISP-Sec-103	SME-US-11	MUST	The Prosumers are informed of any breach or compromise of the C3ISP framework within 72 hours, so that they can take remedial actions. (GDPR Requirement)
C3ISP-Sec-104	ENT-US-1	MUST	The Prosumers are able to define data access and usage policies
C3ISP-Sec-105	CERT-UC-2	MUST	Data sharing and data analysis is compliant with the law obligations and/or the industrial standard
C3ISP-Sec-106	ENT-US-1	MUST	For accountability purposes, C3ISP has an auditing subsystem that traces the enforcement results of the policies

2.2.2. Operational Requirements

2.2.2.1. Cloud Computing and Deployment Models Requirements

The cloud-based deployment model is nowadays prominent because of it technical practicality and of it business relevance (e.g. shifting costs from CapEx to OpEx, [5]). Also Gartner reports that by 2020 "*a Corporate "No-Cloud" Policy Will Be as Rare as a "No-Internet" Policy Is Today*" [25]. Hence, it is evident that C3ISP has to evaluate how it can fit into the Cloud paradigm. Further, pilots call for a cloud-based deployment model: SMEs gain benefits from a

C3ISP provided as a service (SaaS), since the Managed Security Services are provided through the cloud (ref. SME-UC-1), where C3ISP will integrate with. The Enterprise pilot[5] describes a Managed Security Service that is multi-tenant (ref. ENT-US-4), asking for C3ISP to be able to fit into this picture as well.

Table 5 – Cloud Computing and Deployment Models Requirements

ID	Goal	Priority	Requirement
C3ISP-Ope-001	SME-US-1, SME-UC-1, ENT-UC-2	MUST	C3ISP is available as a service, following the SaaS model
C3ISP-Ope-002	SME-US-1, SME-UC-1, ENT-US-4, ENT-UC-2	MUST	C3ISP is multi-tenant, where several tenants (i.e. pilots) can use the framework at the same time w/o troubles
C3ISP-Ope-003	SME-US-7	SHOULD	C3ISP is independent of the CSP where it runs (e.g. public or private)
C3ISP-Ope-004	ENT-US-2	MUST	DSA policies allow to specify different DMOs depending on the trust level the prosumer has on the CSP or on other prosumers in the federation

2.2.2.2. Extensibility and Interoperability Requirements

In this section we analyse requirements to enable the integration with the C3ISP framework and also to allow future improvements of the framework itself, in order to realize a system extensible and interoperable.

The following table summarizes this requirements category.

Table 6 – Extensibility and Interoperability Requirements

ID	Goal	Priority	Requirement
C3ISP-Ope-101	ISP-US-2, ISP-US-4, CERT-UC-1, ENT-US-2, SME-UC-3	MUST	C3ISP provides an open interface for application integration (e.g. a C3ISP API)
C3ISP-Ope-102	CERT-US-2, CERT-NFR-2, SME-US-3, SME-US-4	SHOULD	C3ISP uses a standard to represent data in order to simplify the integration with the framework
C3ISP-Ope-103	ISP-US-1 CERT-US-2 ENT-US-4 SME-US-3	SHOULD	C3ISP should be able to represent different kind of *cyber observables* (see below)

[5] Refer to D4.1

C3ISP-Ope-104	ISP-US-1 CERT-US-2 ENT-US-4 SME-US-3	MUST	C3ISP provides information related to the *cyber observables* (see below) which characterize it

In order to satisfy these requirements we evaluated the adoption of a standard format for organized representation of cyber threat information called **STIX** and proposed by MITRE. STIX is an XML or JSON-based language that has been recently standardized by OASIS, which allows to represent and contextualize any kind of CTI. In particular, it is possible to use STIX to represent an *observed behaviour*, relate it to a known security attack, represent the previous knowledge on methodology and target of the attack, eventual information on the attacker and the known countermeasures, according to the schema shown in Figure 3.

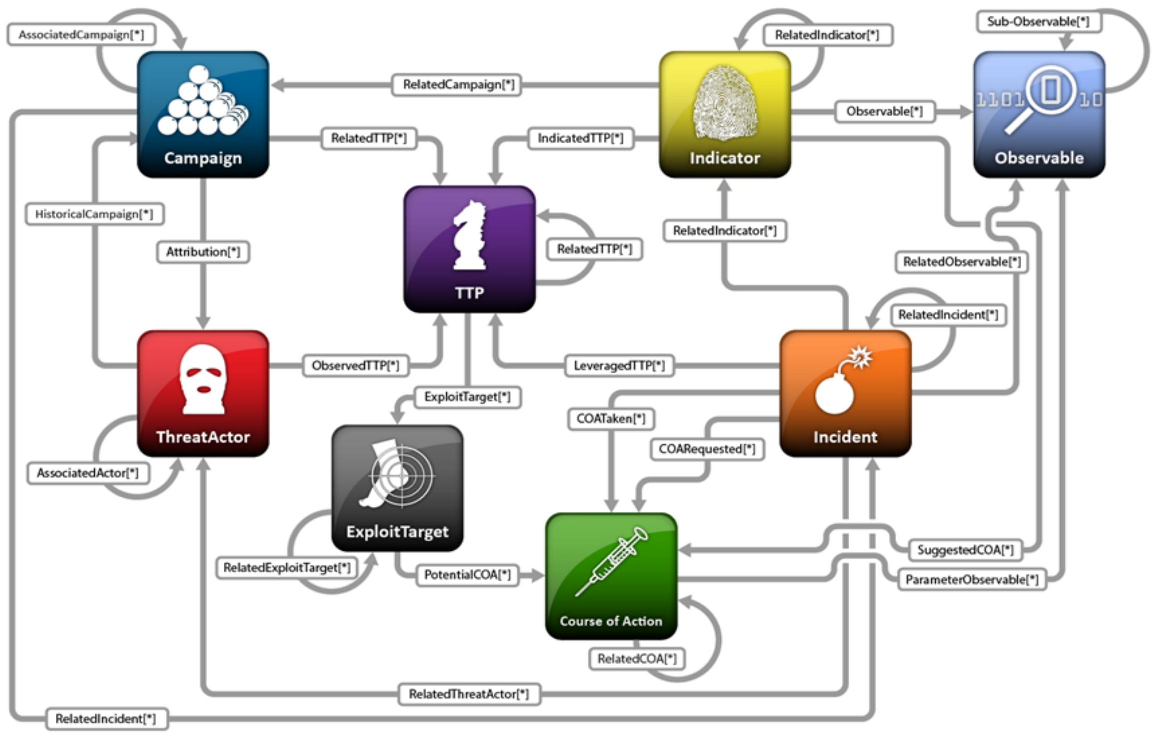

Figure 3: STIX language architecture

One of the main functional blocks of the STIX language are the **Cyber Observables** (upper right in Figure 3), which are used to represent any observed event or information, such as the amount of network traffic, a list of IP addresses, the configuration of a firewall, etc. Cyber Observables represent thus any decontextualized information, which could be related to a threat or attack. For representing different kind of cyber observables, the MITRE itself has proposed the ***CybOX*** standard [7], which easily integrate with STIX, to represent the observed events and raw information in a standard format (e.g. a network connection, an IP address, a file instance, etc.). The other blocks of STIX are:

- The **Indicator**, which contextualize the cyber observables relating them to incidents and describing their importance;
- The **Incident**, which describes in which critical situation the cyber observable and the associated indicator have been observed and could then be related;

- The **Exploit Target**, which describes the specific vulnerability of the system that has been/could have been exploited;
- The **TTP (Techniques, Tactics and Procedures)** which specifies how an attack is performed or how a vulnerability is exploited;
- The **Threat Actor**, which relates the attack to a specific malicious entity;
- The **Campaign**, which collocates the specific attack in a larger set of similar or related attacks;
- The **Course of Action**, which as a counterpart of the TTP, specifies how countermeasures should be taken to avoid, tackle or recover from a specific attack.

TAXII [8] is a transport mechanism that has been defined to provide a protocol for sharing STIX records among different entities, implementing different sharing models. It is designed to integrate within existing data sharing agreements, including the possibility to specify access control conditions. TAXII supports both push and pull messages, to cater for subscription based notification paradigm and on-demand queries (based on standard protocols like HTTP or TLS over HTTP).

TAXII provides three sharing models: (i) **Hub and Spoke**, where a central organisation (the hub) coordinates the exchange between federated parties (the spokes, that can be considered prosumers in the C3ISP jargon); (ii) **Source/Subscriber**, where there is a single data source (an organisation) that share data to subscribers; (iii) **Peer to Peer**, where two or more parties exchange data directly with one another.

Alongside, another language has been proposed to describe Course of Actions, to be taken in order to tackle or mitigate noticed threats. This language is named **OpenC2** [9] (Open Command and Control) and its purpose is to define a lexicon language and semantics at a level of abstraction that will enable the coordination and execution of command and control of cyber defence components between and within networks. OpenC2 commands are vendor neutral and message fabric agnostic, thus can be incorporated in different architectures and environments.

OpenC2 was designed to have a concise set of commands and extensible in order to provide context specific details. Conciseness ensures minimal overhead to meet possible latency and overhead constraints while extensions enable greater utility and flexibility.

The specific implementation of the single commands will be application specific, hence OpenC2 only provides the main instrument to express in a structured way the workflow of a Course of Action. The actual definition of the various steps are left to the designer of the enforcement mechanism. OpenC2 well integrates for structure, paradigm and functionalities with both STIX and TAXII.

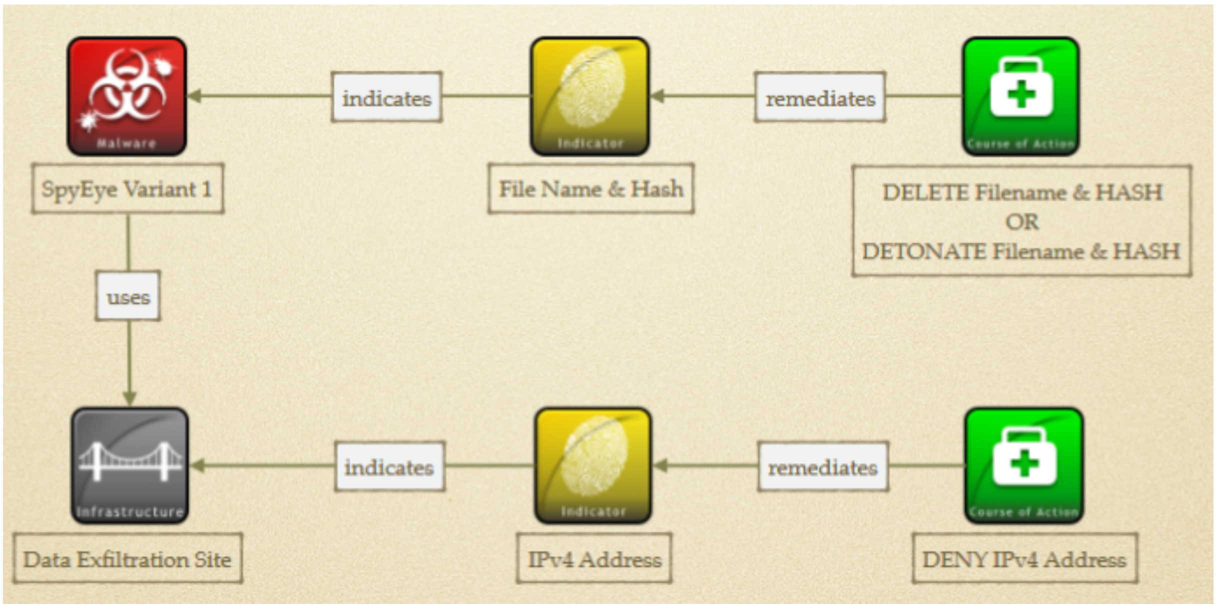

Figure 4: In the picture[6], the Course of Actions contain OpenC2 commands to DENY access at the Firewall and DELETE the malware.

2.2.3. Performance Requirements

From our experience [17], the performance requirements come from the ones of each industrial user-cases providers. Indeed, from each user-cases provider, in function of his need and his algorithm or generic algorithm need, we will provide a corresponding algorithm allowing working on data in HE format. The algorithm performance depends on his Boolean circuit – in fact, each algorithm working with data in HE format can be represented by one Boolean circuit. So optimizing this Boolean circuit, i.e., reducing the Boolean circuits' multiplicative depth, will provide a good performance for the application. Moreover, there is a huge amount of work needed for by-hand optimization of non-straightforward Boolean circuits' multiplicative depth. As one can expect this will be the case for many applications.

Let us define the *direct multiplicative depth* of a node in a Boolean circuit as the length of the longest path starting from circuit inputs to this node. Equivalently, the *reverse multiplicative depth* is the length of the longest path from this node to circuit outputs. The nodes for which these two values coincide are called *critical nodes*. The *critical circuit* contains all the critical nodes of a Boolean circuit. It is straightforward to see that optimizing critical circuit paths allows to minimize the overall multiplicative depth of a Boolean circuit. From this point of view, CEA provides a runtime environment (see 3.2.6).

Table 7 – Performance Requirements

ID	Goal	Priority	Requirement
C3ISP-Per-001	ENT-NFR-2	MUST	C3ISP does not introduce significant delay when enforcing policies for sharing analytics operation results
C3ISP-Per-002	SME-US-5 SME-US-6 SME-UC-3	MUST	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

[6] From: https://www.oasis-open.org/committees/download.php/59483/OpenC2.key.pdf

			on FHE services (*response delay requirement*). This requirement applies only if FHE is used.

2.2.4. Usability Requirements

In this section specific requirements of usability are considered; usability means not only effectiveness and efficiency but also easiness to be used and learned. We considered usability in the presentation aspects and also in the simplicity of processes and tools. The following table summarises the usability requirements collected from the pilots and also implicitly needed for the C3ISP framework.

Table 8 – Usability Requirements

ID	Goal	Priority	Requirement
C3ISP-Usa-001	ENT-NFR-1	SHOULD	C3ISP provides response information about requests (analytics query) to the C3ISP data lake (e.g. why the requested data cannot be provided to prosumers)
C3ISP-Usa-002	SME-US-2 and common to pilots	MUST	C3ISP provides a tool to guide and support the end user in the definition of DSA policies (authorisations, prohibitions, obligations)
C3ISP-Usa-003	SME-US-8 and common to pilots	MUST	C3ISP's processes are seamless and transparent in order to not interfere with the core operations
C3ISP-Usa-004	Common to pilots	MUST	C3ISP's representation of analytics results is effective and efficient for the end user.
C3ISP-Usa-005	SME-US-5 SME-US-6 SME-UC-3	MUST	To use the FHE technology, the decryption service (library) is installed or integrated in client applications.
C3ISP-Usa-006	Common to pilots	MUST	C3ISP provides an intuitive graphical user interface for exploring and visualising the analytics results, e.g. potential cyber-attacks, cyber intelligence, anomalous behaviour.

3. Requirements for Development and Test Bed Environment

This section describes the requirements for setting up tools, processes and activities for delivering a testable system implementing the reference architecture of the C3ISP framework to be used by the Pilots. To gather this information, we interviewed all the partners and came out with a list of preferences or constraints that are reported below.

We foresee two distinct environments that will support this goal:

- **Development Environment**: provides tools to partners for coding the C3ISP framework, like versioning, continuous integration, bug tracking, etc. The artefacts created through this environment will feed the Test Bed where they will be integrated and tested;
- **Test Bed**: provides an instantiation of the C3ISP reference architecture implementation. The Pilots shall use the Test Bed for testing activities by integrating their specific tools and services with the C3ISP reference architecture installed there.

3.1. Development Environment Requirements

3.1.1. Development Environment Infrastructure Setup

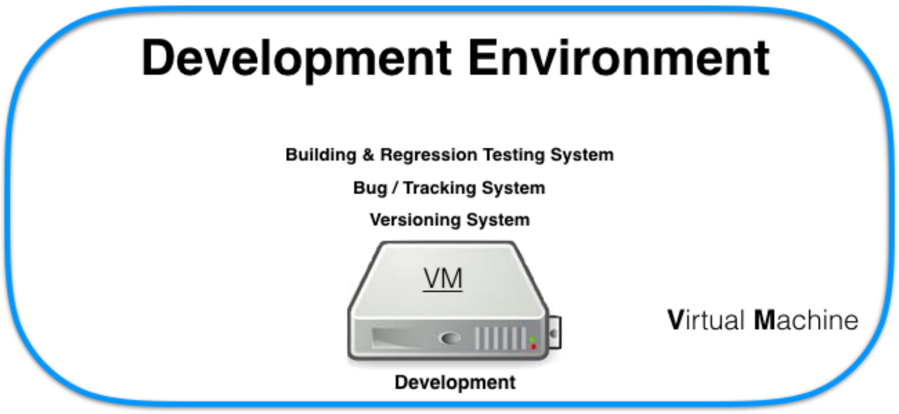

Figure 5: Development Environment

The *Development Environment* will be deployed as a Virtual Machine that contains the tools to store and share the artefacts, for the bug tracking system, continuous integration, versioning and so on. At month 6 of the project, we have not fixed yet the hardware and software specification for this environment. However, based on prior experiences, we plan to set up at VM with at least 8 cores, 8 GB of RAM and 200 GB of storage. The Operating System will be Ubuntu 16.04 LTS[7] and the VM will be deployed in the CNR data-elaboration centre, where it will be reachable with a public IP to all the consortium's partners.

3.1.2. Development Tools

With the aim of creating a computing environment that supports the development of the C3ISP framework, by being available to all partners and to satisfy pilots' constraints, we interviewed the consortium members and presented an Agile-based style of working [21] to be able to reach the relevant goals and milestones set for C3ISP within the expected deadlines but at the same time without scarifying the overall quality of the software artefacts.

[7] Ubuntu LTS (Long-Term Support) versions have five years support.

The assessment resulted in the requirements summarised in the following table, which can be considered common to all the pilots and stakeholders**.**

Table 9 – Development Environment Tools Requirements

ID	Goal	Priority	Requirement
C3ISP-Dev-001	All artefacts should be stored in the same repository for ease of sharing	MUST	Have a Revision Control System for storing the source code of the project
C3ISP-Dev-002	Speed up the build process	MUST	Automate the build
C3ISP-Dev-003	Run test suite automatically	SHOULD	Make the build self-testing
C3ISP-Dev-004	Speed up the test process	SHOULD	Have a centralized bugs tracking system
C3ISP-Dev-005	Deploy	SHOULD	Automate deployment
C3ISP-Dev-006	User free tools	MUST	Free or open source software
C3ISP-Dev-007	Commercially friendliness tools	SHOULD	The license is commercially friendly (i.e. BSD or Apache-like) and not copy-left (i.e. GPL-like)
C3ISP-Dev-008	Java support	MUST	Support the Java programming language, used for the C3ISP core framework (i.e. DSA tools and enforcement)
C3ISP-Dev-009	C/C++ support	MUST	Support the C/C++ programming language, mainly used for FHE components
C3ISP-Dev-010	Python support	MUST	Support the Python programming language, mainly used for scripting and also to use some standard reference implementation (e.g. TAXII)

In this section we concentrate on the tools that can assist the objective of setting up a continuous integration service [22]. In particular, in the survey we carried out among the consortium, we asked to raise preferences about the following classes of solutions:

- **Version control system**: it is the hearth of a disciplined software development process, used to store the source code of the project along with all the required artefacts (libraries, configuration files, etc.), with multiple parties (developers) that contribute concurrently, merge their code, and are assisted in solving code conflicts (when two or more people update the same piece of code);
- **Build Automation system**: it is in charge of automating the process of compiling the source code into executable code. It typically handles software dependencies, i.e. can

automate the management (finding, retrieval, etc.) of the correct libraries required to build the software component;

- **Artefact repository**: it is a system that keeps binary software components that are on one hand needed to support the Build Automation systems for dependencies, on the other used to store the software artefacts generated during the build step;
- **Continuous Integration software**: it is the orchestrator of the whole build process that integrates with all the other solutions and enables automation of the cycle that includes the fetching the code from the Revision Control system, compiling it through the Build Automation, testing the generated software, store it on the Artefact repository, evaluate the solution for quality and security issues, including the automatic deployment of the binaries and configuration into the running environment;
- **Bug Tracking system**: it is a tool that supports the tracking of software defects for different systems components and/or deployment environments, including enhancement or change requests, issue prioritization and assignment, roadmaps, etc., by providing a centralised dashboard of the project development requests;
- **Unit Test system**: it supports the automation of testing “units of source codes” (methods, classes, modules, etc., depending on the programming language), by defining “unit test cases” which are code fragments that verify the intended behaviour of the code unit. Typically, Unit Test frameworks allow the developers to use mock-up services (e.g. stubs) to verify the component without being impacted by other depending piece of codes or systems;

The following table summarises the evaluated software components:

C3ISP - Development Software components	
Version Control System	Concurrent Version Systems (CVS), http://savannah.nongnu.org/projects/cvs
	Apache Subversion (SVN), https://subversion.apache.org
	GIT, https://git-scm.com
Build Automation System	Apache Ant, http://ant.apache.org
	Apache Maven, https://maven.apache.org
	Gradle, https://gradle.org
Artefact repository	Nexus Repository OSS, https://www.sonatype.com/nexus-repository-oss
	Artifactory, https://www.jfrog.com/open-source
Continuous Integration Software	Hudson, http://hudson-ci.org
	Jenkins, https://jenkins.io

	Cruise Control, http://cruisecontrol.sourceforge.net
Bug Tracking System	MantisBT, https://www.mantisbt.org
	Bugzilla, https://www.bugzilla.org
	Trac, https://trac.edgewall.org
Unit Test System	Junit, http://junit.org
	TestNG, http://testng.org
	Spock, http://spockframework.org

For evaluating tools, we took into account the criteria and requirements in Table 9 and tried to find a balance in order to cover most (if not all) of them. The comparison is summarized in the following table:

	C3ISP -Dev-001	**C3ISP -Dev-002**	**C3ISP -Dev-003**	**C3ISP -Dev-004**	**C3ISP -Dev-005**	**C3ISP -Dev-006**	**C3ISP -Dev-007**	**C3ISP -Dev-008**	**C3ISP -Dev-009**	**C3ISP -Dev-010**
CVS	X						X	X	X	X
SVN	X						X	X	X	X
GIT	X					X		X	X	X
ANT		X	X		X		X	X	X	X*
MAVEN		X	X		X			X	X	X*
Gradle		X	X		X		X	X	X	X*
NEXUS		X	X			X		X	X	X
Artifactory		X	X			X		X	X	X
Hudson		X	X				X	X	X	X
Jenkins		X	X			X		X	X	X
Cruise Control		X	X				X	X	X	X
MantisBT				X	X					
Bugzilla				X		X				
Trac				X	X					
Junit			X			X		X		X*
TestNG			X			X		X		
Spock			X			X		X		

*the support is enabled by specific library/module to be included

According to the evaluation and the feedback collected the selected tools are:

- **GIT** as the revision control system;
- **Maven** as build automation system for Java code; make and cmake for C/C++ code;
- **Jenkins** as Continuous Integration software;
- **Nexus** as artefact repository;
- **Trac** as bug tracking system;
- **Junit** as Java unit test system (possibly it can be used also for Python [23]).

Since we do not plan to modify the code of the selected tools (so, the tools' licenses do not have strong impact), C3ISP-Dev-007 requirement was relaxed in the evaluation process, accepting also GPL-like licensed tools.

3.1.3. Quality and Assurance Strategy

Mechanisms to evaluate the software quality will be available in the development environment in order to provide tools to measure the software in terms of functionality, reliability, security, performance efficiency and maintainability, according to ISO/IEC 25010:2011 standard[8]. We investigated about several tools to be used for evaluating quality of the code. In particular, our analysis involves free or open source tools for static and dynamic analysis of the code that can be used to assess the quality of the code in terms of presence of design flaws or code bugs that can be considered security issues or can represent vulnerabilities. We focused on tools supporting Java, Python, C and C++ since they are the programming languages used for implementing the pilots.

These tools should be integrated in a Continuous Integration system for versioning, build and deploy of the code to meet high level of assurance software requirement, which is common to all the pilots. For comparing the tools, we have also consulted the online community Black Duck Open Hub[9] that offers analytics for open source code and projects, such as licenses used and project wellness (how many developers, when was last code commit, etc.).

We consider of outmost importance that security software is also secure and so we put great attention on both software quality and assurance practices. For this reason, the following paragraphs provide an analysis for each evaluated tool (split between static code analysers and dynamic analysers) and from this selection we then choose a subset that better addresses the development requirements provided by partners and pilots as reported in the following table:

Table 10 – Development Environment Quality & Assurance Requirements

ID	Description	Pilot Requirement	ISO25010:2011 Recommendation
C3ISP-Dev-101	Support for Continuous Integration	X	
C3ISP-Dev-102	Open source or free software (type of license)	X	
C3ISP-Dev-103	Support for C/C++/Java/Python programming languages	X	
C3ISP-Dev-104	IDE integration (to be used in development environment)	X	
C3ISP-Dev-105	Stability of the code		X
C3ISP-Dev-106	Absence of known vulnerabilities		X
C3ISP-Dev-107	Healthy of the open source community		X
C3ISP-Dev-108	Compliance with standard		X
C3ISP-Dev-109	Usability (i.e. analysis results)		X

[8] http://www.iso.org/iso/home/store/catalogue_ics/catalogue_detail_ics.htm?csnumber=35733

[9] https://www.openhub.net

C3ISP-Dev-110	Source code availability needed for analysis		X
C3ISP-Dev-111	Scan for vulnerabilities		X

3.1.3.1. Static Analysis Tools Overview

In this section we offer a roundup of tools used for static analysis of the code, i.e. the practice of analysing the source code of a system without having it running. In particular such tools are dedicated to Python, C, C ++ and Java programming languages, according to the analysis provided by the open software security community OWASP[10] and by the National Institute of Standard and Technologies (NIST)[11].

CheckStyle

CheckStyle[12] is a syntactical checking tool used to verify if the developed code is compliant with standard good programming rules to foster maintainability and reuse. It is configurable to write rules to support any code standard and it is able to check code in terms of style constraints (naming conventions, comments, limit of number of parameter for functions, duplicated sections, etc.). It can be useful to define levels of check according to the accountability needed for a program. CheckStyle supports **build automation tools** (i.e. Maven, Ant, Jenkins/Hudson) and Java programming language, including latest **Java v1.8**. It also integrates with many IDEs, like Eclipse and NetBeans, and with **continuous integration** systems. CheckStyle is an open source **LGPL-licensed** software. According to Black Duck Open Hub[13], CheckStyle is a mature project maintained by a very large community of developers; this is one of the largest open-source teams in the world, and it is in the top 2% of all project teams on Open Hub. Finally, no vulnerabilities are currently reported for this project.

FindBugs and FindSecBugs

FindBugs[14] is a static analysis tool supporting **Java**, Groovy and Scala programming languages. It operates at bytecode level; it means that the source code is not needed to perform the analysis. For this reason, FindBugs is very useful to analyse third-parties libraries and external modules. The analysis is based on the discovery of "bug patterns", which are code instances that are likely to be errors. FindBugs requires **JRE (or JDK) 1.7.0** or later to run. However, programs in any version of Java, from 1.0 to 1.8 can be analysed. The set of rules can be extended using additional plug-ins. For example, to increase the security bugs analysis, FindBugs can be extended with FindSecBugs[15] plug-in, which is dedicated to the security audit. FindSecBugs provides OWASP Top 10 and CWE coverage and it supports popular frameworks (i.e. Spring-MVC, Struts, Tapestry). Both FindBugs and FindSecBugs are open source **LGPL-licensed** software and they can be used in a **continuous integration** environment (using Jenkins, for example), can be integrated within IDEs (Eclipse, NetBeans, IntelliJ, etc.) and support **build automation tools** (Ant, Maven). FindBugs is born in 2003 and has registered recent activities

[10] https://www.owasp.org/index.php/Source_Code_Analysis_Tools

[11] https://samate.nist.gov/index.php/Source_Code_Security_Analyzers.html

[12] http://checkstyle.sourceforge.net/

[13] https://www.openhub.net/p/checkstyle

[14] http://findbugs.sourceforge.net/

[15] https://find-sec-bugs.github.io/

in terms of contribution of the developers; according to Black Duck Open Hub[16], projects with recent activities and a code base more than five years old are likely solving vital problems and delivering consistent value, then it comes out reliable.

PMD

PMD[17] is an analyser using a syntactic approach to discover potential flaws in the source code. It supports several languages, including **Java** (also version 1.8), JavaScript and it is also able to analyse **C, C++**, C# code. It leverages on the CPD (Copy-Paste-Detector), which finds duplicated code in Java, C, C++, C#, PHP, Ruby, FORTRAN, and JavaScript. PMD is integrated with many IDEs (Eclipse, and IntelliJ IDEA among the others) and it integrates with **build automation tools** (i.e. Maven, Ant and Jenkins/Hudson). In particular, Jenkins provides a dashboard to report graphically the evolving trends of the discovered bugs with specific views (through the Jenkins' DRY plug-in) for the Copy-Paste-Detector (CPD). PMD is an open source **BSD-like licensed** software. About PMD[18], Black Duck Open Hub reports a vital community, born in 2002 but currently strongly active, which indicates a mature and relatively bug-free code base.

VisualCodeGrepper (VCG)

VisualCodeGrepper[19] scans **C/C++**, C#, VB, PHP, **Java**, and PL/SQL for security issues and for comments that may indicate defective code. The tool tries to identify potential risks as buffer overflows and signed/unsigned comparison in C code, violations of OWASP recommendations in Java code, etc. The configuration files can be used to carry out additional checks for banned functions or functions, which commonly cause security issues. The output of the analysis is shown in a pie chart (for the entire codebase and for individual files) showing relative proportions of code, whitespace, comments, "ToDo" style comments and bad code. The software is a "desktop" application that means it cannot be integrated into a Continuous Integration system automatically. VCG is **GPL-licensed**. Black Duck Open Hub does not provide report for VCG.

Splint

Splint[20] is a free tool for statically checking **C** programs for security vulnerabilities and programming mistakes. Using annotation in the code, it is possible to increase the checks providing additional information used at analysis time. Problems detected by Splint include dereferencing a possibly null pointer, memory management errors, buffer overflow vulnerabilities, dangerous macro implementations or invocations, violations of customized naming conventions and so on. Splint is available as source code and binary executables for several platform under **GPL-3.0+** a licensed. Black Duck Open Hub[21] reports no changes in over a year on this project and no vulnerabilities are known for this software. Since comments are very few in the source code, it puts Splint among the lowest one-third of all C projects on Open Hub.

[16] https://www.openhub.net/p/findbugs

[17] http://pmd.sourceforge.net/

[18] https://www.openhub.net/p/pmd

[19] https://sourceforge.net/projects/visualcodegrepp

[20] http://www.splint.org

[21] https://www.openhub.net/p/splint

Cppcheck

Cppcheck[22] is a free (**GPLv3 license**) static analysis tool for **C/C++** code. The tool provides checks for several potential risks (for example pointer to a variable that goes out of scope, bounds, classes with missing constructors or unused private functions, exception safety, memory leaks, invalid STL usage, overlapping data in sprintf, division by zero, null pointer dereference, unused struct member, passing parameter by value, etc.) with the goal of no false positives. The tool can be invoked by a command line and can be integrated in a Jenkins job for implementing a **Continuous Integration** system. Black Duck Open Hub evaluates Cppcheck[23] a versatile tool since it is able to discover bugs that the C/C++ compiler does not find and also it supports non-standard code such as various compiler extensions, inline assembly code, etc. Nevertheless, since the number of lines of comment in Cppcheck project is very low respect to the all C++ projects on Open Hub (11% of all source code lines are comments against the 22% across all the C++ project) Black Duck Open Hub puts Cppcheck among the lowest one-third of all C++ projects on Open Hub. The presence of comments indicates a very well documented code source and a disciplined development team that are significant characteristics for evaluating a project.

UNO

UNO[24] is an acronym and stands for:

- Use of uninitialized variable,
- Nil-pointer references, and
- Out-of-bounds array indexing.

These are the three most common types of software defects on which the free tool for static **C** code analysis in focused on. The idea of UNO is avoid producing a huge amount of results that can probably contain false positive results and be more precise concentrating on the most common issues. The tool also allows to define several user-defined properties to be used to extend the checks used by the tool. UNO is available as a command line tool. Black Duck Open Hub does not provide report for UNO.

Flawfinder

Flawfinder[25] is lexical source code static analyser used to scan **C** and **C++** code in order to identify any flaw, sorted by risk level, in a Common Weakness Enumeration[26] compatible approach. The tool requires Python 2 to run; it provides a command line and requires the source code availability. It is a very simple tool that does not even know about the data types of function parameters, and it does not perform control flow or data flow analysis. It just makes a comparison between the application code and its built-in database of well-known problems (buffer-overflow risks, format string problems, race conditions, etc.). Flawfinder is released under the **GPL version 2** or later, and thus is open source and free software. Black Duck Open Hub evaluates Flawfinder[27] as a very useful tool for quickly finding and removing some security problems before a program is widely released. It appears to be a very young project, since the

[22] https://sourceforge.net/projects/cppcheck

[23] https://www.openhub.net/p/cppcheck

[24] http://spinroot.com/uno/

[25] http://www.dwheeler.com/flawfinder

[26] http://cwe.mitre.org

[27] https://www.openhub.net/p/flawfinder

source code repository for Flawfinder has less than a year of continuous activity, however it is known to be one of the very first security static tool (around 2001).

OWASP Dependency Check

OWASP Dependency Check[28] is used to analyse any project dependencies with external module or third-parties libraries which are publicly known to be vulnerable. It does not require source code and it is in line with the OWASP Top 10 2013 (in particular A9[29] check). **Java** and .NET are currently supported; other programming languages, like **C** and **C++**, are partially supported (with *autoconf* and *cmake* build systems) and under experimentation. OWASP Dependency Check is integrated in **build automatic tools** (Maven, Ant, Jenkins/Hudson), but it is not integrated as IDE plug-in. This utility scans the dependencies used in the application getting "evidences" from it and then uses these evidences to identify the Common Platform Enumeration[30] for each dependency. Once a CPE is identified, it gathers the associated Common Vulnerability and Exposure (CVE) from the US NIST's National Vulnerability Database[31] (NVD) in order to document the founded issue. OWASP Dependency Check is a **GPL-licensed** software. Black Duck Open Hub evaluates OWASP Dependency Check Jenkins Plugin[32]: it is considered a very stable project (no recent activities) and with the 45% of line of comment, it is among the highest one-third of all Java projects on Open Hub.

Tox

tox[33] is an automation tool providing packaging, testing and deployment of **Python** software. It is available as test command line tool, but also it integrates with **continuous integration** servers (like Jenkins). It provides the following features:

- Checking that packages install correctly with different Python versions and interpreters;
- Configuring and running tests;
- Acting as a front-end to CI servers.

tox is a **GPL-licensed** software. According to Black Duck Open Hub34, tox has a mature, well established codebase, it is well documented and developed by a large team.

PyChecker

PyChecker[35] is a very simple static analysis tool for **Python**, which is able to discover bugs similar to those that a compiler finds on languages such as C and C++. Some false positives can occur because of the dynamic nature of the Python, according to its developers. It is available as a command line tool and it seems to be a stable project. Lack of comments indicates that it is not well documented. PyChecker is a **BSD-licensed** software.

[28] https://www.owasp.org/index.php/OWASP_Dependency_Check

[29] https://www.owasp.org/index.php/Top_10_2013-A9-Using_Components_with_Known_Vulnerabilities

[30] http://nvd.nist.gov/cpe.cfm

[31] http://nvd.nist.gov

[32] https://www.openhub.net/p/dependency-check-plugin

[33] https://tox.readthedocs.io/en/latest/

[34] https://www.openhub.net/p/python-tox

[35] http://pychecker.sourceforge.net/

Pylint

Pylint[36] is a highly configurable, customizable tool used for checking compliance to a coding standard for **Python**. It provide a wide variety of feature like checking line-code's length, checking if variable names are well-formed according to a coding standard, or checking if declared interfaces are truly implemented, and much more. It can be used in a **continuous integration** environment working on a custom configuration (i.e. Jenkins) and it is also available for IDEs. Pylint is a **GPL-licensed** software. Black Duck Open Hub provides a quite good review of the project, since it is considered mature and no vulnerabilities are known about it even if the lacks of comments puts Pylint among the lowest one-third of all Python projects on Open Hub.

3.1.3.2. Static Analysis Tools Evaluation

The following table summarises the tools evaluation by following the evaluation criteria and requirements described in Table 10 – Development Environment Quality & Assurance Requirements.

	C3ISP-Dev-101	**C3ISP-Dev-102**	**C3ISP-Dev-103**	**C3ISP-Dev-104**	**C3ISP-Dev-105**	**C3ISP-Dev-106**	**C3ISP-Dev-107**	**C3ISP-Dev-108**	**C3ISP-Dev-109**	**C3ISP-Dev-110**	**C3ISP-Dev-111**
CheckStyle	X	X (LGPL)	X (Java)	X	X	X	X	X	X	X	-
FindBugs / FindSecBugs	X	X (LGPL)	X (Java)	X	X	X	X	-	X	-	X
PMD	X	X (BSD)	X (Java; C; C++)	X	X	X	X	-	X	X	X
VCG	-	X	X (Java; C; C++)	-	-	X	-	X	-	X	-
Splint	-	X (GPL-3.0+)	X (C)	X	X	X	-	-	X	X	X
Cppchecker	X	X (GPL-3)	X (C/C++)	X	X	X	-	-	-	X	X
UNO	-	X	X (C)	X	-	X	-	-	-	X	-
Flawfinder	-	X (GPL-2)	X (C/C++)	X	-	X	-	X	-	X	X
OWASP Dependency Check	X	X (GPL)	X (Java; C and C++ partially)	X	X	X	X	X	X	X	X
tox	X	X (GPL-2.0+)	X (Pytho	X	X	X	X	-	X	X	-

[36] https://www.pylint.org/

			n only)								
PyChecker	-	X (BSD-3-Clause)	X (Python only)	X	X	X	X	-	X	X	-
Pylint	X	X (GPL-2.0+)	X (Python only)	X	X	X	X	-	X	X	-

According to the evaluation criteria and the pilots' requirements, we can conclude that the tools which satisfy our needs are:

- **CheckStyle**, for syntactical checking Java code;
- **FindBugs** and **FindSecurityBugs**, to cover the (security) static analysis of Java code;
- **Cppchecker**, to cover static analysis in C/C++ code;
- **Tox** for automatize the static analysis of Python code;
- **OWASP Dependency Check**, for checking external dependency with Java libraries.

3.1.3.3. Dynamic Analysis Tools Overview

In this section, we want to provide an overview of the dynamic analysis tools to spot security issues[37] available as open source or free. For the comparative analysis, we take in account the results of the WAVSEP (Web Application Vulnerability Scanner Evaluation Project)[38] assessment, which is an evaluation platform containing a collection of vulnerable pages that can be used to help assessing the features, quality and accuracy of web application vulnerability scanners. A common way to test and compare the capabilities of today's scanners is via the WAVSEP benchmark. After the tools overview, we evaluate them with respect to the requirements set in Table 10.

OWASP Zed Attack Proxy Project (ZAP)

OWASP ZAP[39] is probably the most popular open source (**Apache 2.0 License**) tool for the dynamic analysis. A very active and mature community of volunteers supports the project and its capabilities include not only web application scanning but also penetration test. OWASP ZAP supports a wide range of scripting languages (i.e. JavaScript, Ruby, Groovy, Python, Zest, etc.); it also includes a large set of functionalities like intercepting proxy, passive scanner, forced browsing, etc.[40] and provides a REST API to interact programmatically with the tool. Using the API, ZAP can be enabled as a proxy: it means that ZAP will be positioned between the browser and the web application to intercept all the requests. Before starting running attack scenarios, ZAP crawls through the web application and record all URLs from the local domain, skipping URLs that point to other domains. The API allows the tool to be fully integrated in a

[37] Gartner calls this family of tools *DAST – Dynamic Application Security Testing*, http://www.gartner.com/it-glossary/dynamic-application-security-testing-dast

[38] https://github.com/sectooladdict/wavsep

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

[40] More details at https://www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project#tab=Functionality

continuous integration environment. According to the WAVSEP assessment[41], ZAP has very good performance in SQL injection detection and in RFI (Remote File Inclusion, which could involve XSS attacks) discovery, but it is not the optimum in the discovery of invalidated redirect URL and it does not support Web Services scanner. Overall, it is considered a very stable tool, easy to use and simple to configure.

Arachni

Arachni[42] is a web application security framework, released **free for not commercial purposes**. Its features include crawling and vulnerabilities detection modules and it is in the top of the scoreboard of the WAVSEP benchmark (96% score)[43]. It does not support Web Services crawling and scanning but it provides features for almost all the authentication, control and connection features observed by WAVSEP. Integrate Arachni in a continuous integration environment is possible since it provides a couple of different interfaces that can be used for automation: a command line interface (CLI) as well as REST and an RPC service can be triggered. Furthermore, the tool is highly customisable since it is possible (for example using the suitable command in the CLI) run just a set of test to discover a certain type of vulnerabilities (for example only SQLi) rather than different kind of possible issues.

Syntribos

Syntribos[44] is an open source (**Apache Licence**) automated API security testing tool part of the OpenStack Security Project. The tools is designed for testing OpenStack API and it is developed in Python. It allows to test Web Services given a configuration file and an example of HTTP request, and scans the application to find a large set of vulnerabilities. Some templates are available as starting point to implement custom security tests. The integration of Syntribos in a continuous integration environment is not explicitly documented, but, since it is a command line tool and it is thought to automatize the test, it should be possible to do that. WAVSEP does not assess this tool.

IronWASP

IronWASP (Iron Web application Advanced Security testing Platform)[45] is a security testing tool distributed under **General Public License**. The tool is designed to be customizable in order to allow the user to write custom security scanners for web applications (however, no Web Service scanner is provided). It means that the user should have Python/Ruby scripting languages expertise; anyway, the tool provides a set of features useful for beginners and not development expert. WAVSEP considers IronWASP a great tool for testing applications that use non-standard input delivery method, but it seems to be more specifically useful for manual testing[46]. The use of the IronWASP tool in a continuous integration environment is not officially reported.

[41] http://sectoolmarket.com/web-application-scanners/52.html

[42] http://www.arachni-scanner.com/

[43] http://sectoolmarket.com/web-application-scanners/57.html

[44] https://docs.openstack.org/developer/syntribos/

[45] http://ironwasp.org/

[46] http://sectoolmarket.com/web-application-scanners/78.html

Burp suite

Burp Suite[47] is a web application scanning tool produced by PortSwigger. It is present in the Gartner's[48] "Magic Quadrant for Application Security Testing" in its commercial version, but it has also free version, with a sub-set of functionalities. Born as intercepting proxy, in the Pro version includes an advanced Web Vulnerability Scanner for OWASP Top 10 vulnerabilities. It has a plugin system that extends its scope (e.g. SAML Editor, WSDL Wizard, etc.) and it can be controlled via open APIs. Burp Suite also enables a static scan of JavaScript code during its DAST activities and it can be installed in a continuous integration environment.

3.1.3.4. Dynamic Analysis Tools Evaluation

The tools evaluation, summarised in the following table, follows the evaluation criteria also used for the static analysis tools (Table 10); some of them (C3ISP-Dev-103, C3ISP-Dev-104 and C3ISP-Dev-110) are not applicable for a dynamic analysis tool and they are ignored.

	C3ISP-Dev-101	**C3ISP-Dev-102**	**C3ISP-Dev-105**	**C3ISP-Dev-106**	**C3ISP-Dev-107**	**C3ISP-Dev-108**	**C3ISP-Dev-109**	**C3ISP-Dev-111**
OWASP ZAP	X	X (Apache 2.0)	X	X	X	X	X	-
Arachni	X	X (free for not commercial use)	X	X	X	-	X	X
Syntribos	X	X (Apache)	X	X	X	-	X	-
IronWASP	-	X (GNU)		X	X	X	X	X
Burp Suite	X	X (free version)	X	X	-	X	X	X

According to our analysis, OWASP ZAP seems to be the more mature and reliable tool for dynamic analysis, comply with our needs.

3.2. Test Bed Requirements

The Test Bed Environment has the following objectives:

- Work as an integration environment, where artefacts produced in the Development Environment can be combined and integrated to form the C3ISP subsystems and components;
- Be a testing environment, where end-to-end scenarios can be exercised to identify integration issues and to verify the intended system behaviour;
- Act as the reference C3ISP framework installation to be used by Pilots to realise their use cases.

To pursue these objectives we identified a set of requirements mainly in terms of infrastructure setup that we think will help us to fully implement a working C3ISP prototype that will support in the appropriate way our demonstrators (i.e. the four Pilots).

[47] https://portswigger.net/burp/

[48] https://www.gartner.com/doc/3107518/magic-quadrant-application-security-testing

3.2.1. Test Bed Environment Infrastructure Setup

This environment is constituted by a set of servers, which can be both virtual and physical, to contain artefacts developed in the C3ISP project. At this time (*Month 6*), we have designed a test bed environment with the following machines:

- One Virtual Machine (VM) for the Information Sharing Infrastructure (ISI) artefacts;
- One Virtual Machine for each pilot. So, in total 4 VMs;
- One Physical Server for the Information Analytics Infrastructure (IAI). This machine should respect the requirements set on Table 20.

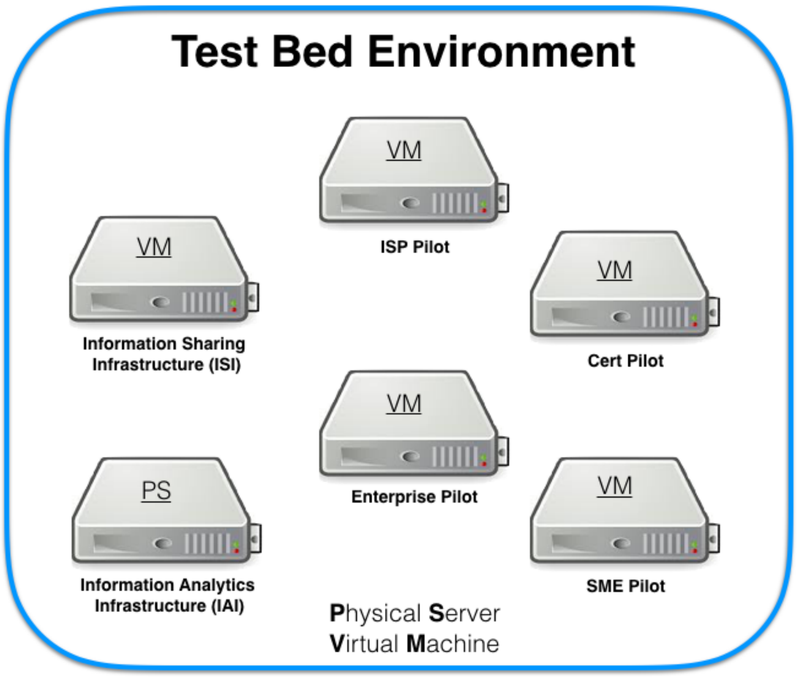

Figure 6: Test Bed environment

In the following, we list the requirements that the pilots' owners set for C3ISP. The project involves four pilots, which are: *ISP Pilot* owned by CNR, *CERT Pilot* owned by the Italian CERT, *Enterprise Pilot* owned by SAP and finally the *SME Pilot* owned by BT. Each use case is described in its Work Package and details are available in the final deliverables: *D2.1*, *D3.1*, *D4.1* and *D5.1*.

In the following tables, we define three type of requirements for each pilot related to the test bed requirements: *Test Bed* (identified as C3ISP-Tst-*), *Integration* (identified as C3ISP-TsT-Int-*) and *Software* related (identified as C3ISP-Tst-Sof-*). *Test bed* requirements will build the setup of the machines in the Test Bed environment, *Integration* will guide the phase of integration among the different artefacts and machines, and, finally, the *Software* requirements will specify additional software components that are needed by each pilot.

3.2.2. ISP Pilot

The ISP Pilot focuses on providing security benefits to Internet Service Providers (ISPs) that interact with the C3ISP framework by exploiting its analytics operations.

In the following, test bed, integration and software requirements are listed for the ISP pilot.

3.2.2.1. *Test Bed requirements*

Table 11 – ISP Pilot – Test Bed Requirements

ID	Requirement	Priority	Description
C3ISP-Tst-101	Server for remote host scanning	MUST	Two Virtual Machines (*master plus backup*) able to run the *Security Scan Software* to be used by ISPs. Each VM will need 4 virtual CPU Cores, 8 GB of RAM, at least 100 GB of storage, and two public IPs (one IPv4 and one IPv6)

3.2.2.2. *Integration Requirements*

Table 12 – ISP Pilot – Integration Requirements

ID	Requirement	Priority	Description
C3ISP-Tst-Int-101	Internet Service Providers integration with Registro.it	MUST	For a proper integration, the virtual machines, which host the Security Scan Software, must be reachable by the ISPs with a public IP
C3ISP-Tst-Int-102	Internet Service Providers integration with C3ISP	MUST	The ISPs must be able to reach the C3ISP framework to upload and download report to/from C3ISP
C3ISP-Tst-Int-103	Integration with the *Information Sharing Infrastructure (ISI)*	MUST	An ISP must be integrated with the Information Sharing Infrastructure (ISI)
C3ISP-Tst-Int-104	Integrity and confidentiality	SHOULD	Support of a protocol to have integrity and confidentiality security properties in communications

3.2.2.3. *Software Requirements*

Table 13 – ISP Pilot – Software Requirements

ID	Requirement	Priority	Description
C3ISP-Tst-Sof-101	Security Scan Software	MUST	The software used for remote scanning will be web-accessible so that ISPs can initiate scan sessions for: • DNS configuration checks • Mail configuration glitches • Operating System checks based on known vulnerabilities • Service scan based on service discovery (i.e. if FTP is found,

			FTP-based vulnerabilities will be tested)
C3ISP-Tst-Sof-102	Local Scan	SHOULD	The ISPs should be able to collect local logs, such as authentication log, and send them to C3ISP

3.2.3. CERT Pilot

This pilot allows SMEs to participate to a collaborative platform for sharing security relevant information for early discovery of security threats and attack attempts. In the following, test bed, integration and software requirements are listed for the CERT Pilot.

3.2.3.1. Test Bed Requirements

Table 14 – CERT Pilot – Test Bed Requirements

ID	Requirement	Priority	Description
C3ISP-Tst-201	Virtual Machine	MUST	A Virtual Machine with Linux OS, at least 4 cores for computation on big data for analysis, and at least 200 GB needed for the data lake.
C3ISP-Tst-202	Programming language	MUST	Java Development Kit with up-to date installation

3.2.3.2. Integration Requirements

Table 15 – CERT Pilot – Integration Requirements

ID	Requirement	Priority	Description
C3ISP-Tst-Int-201	Integration with the *Information Sharing Infrastructure (ISI)*	MUST	Integration with the Information Sharing Infrastructure (ISI)
C3ISP-Tst-Int-202	Integrity and confidentiality	SHOULD	Support of a protocol to have integrity and confidentiality security properties in communications

3.2.3.3. Software Requirements

No specific software requirements have been identified at this stage.

3.2.4. Enterprise Pilot

The deployment of this Pilot represents a next-generation, enterprise cyber-defence operations platform based on big data (Hadoop ecosystem) technology.

In the following, test bed, integration and software requirements are listed for the Enterprise pilot.

3.2.4.1. Test Bed Requirements

Table 16 – Enterprise Pilot – Test Bed Requirements

ID	Requirement	Priority	Description
C3ISP-Tst-301	Virtual Machine	MUST	A Virtual Machine with Linux OS, at least 4 cores for computation on big data for analysis, and at least 200 GB needed for the data lake.

3.2.4.2. Integration Requirements

Table 17 – Enterprise Pilot – Integration Requirements

ID	Requirement	Priority	Description
C3ISP-Tst-Int-301	Integrity and confidentiality	SHOULD	Support of a protocol to have integrity and confidentiality security properties in communications
C3ISP-Tst-Int-301	Reuse of data lake	MUST	The pilot is able to integrate with an already existing big data lake

3.2.4.3. Software Requirements

No specific software requirements have been identified at this stage.

3.2.5. SME Pilot

The SME pilot provides access to a multi-party cloud environment and a managed security service to enable application & host protection. In the following, test bed, integration and software requirements are listed for the SME pilot.

3.2.5.1. Test Bed Requirements

Table 18 – SME Pilot – Test Bed Requirements

ID	Requirement	Priority	Description
C3ISP-Tst-401	Data Lake	MUST	SMEs need at least 200 GByte of storage for the Data Lake
C3ISP-Tst-402	SME-US-5 SME-US-6 SME-UC-3	MUST	FHE services are deployed into a physical server, not into a Virtual Machine, because of using parallelism method for optimizing the Boolean circuits.

3.2.5.2. Integration Requirements

Table 19: SME Pilot - Integration Requirements

ID	Requirement	Priority	Description
C3ISP-Tst-Int-401	Integrity and confidentiality	SHOULD	Support of a protocol to have integrity and confidentiality security properties in communications

3.2.5.3. Software Requirements

No specific software requirements have been identified at this stage.

3.2.6. Other Test Bed Requirements

CEA provides a runtime environment supporting FHE for Boolean circuits' optimisation– i.e. the pre-processing step – linking either with HElib[49] or with CEA custom implementation of the Fan-Vercauteren FHE scheme. The runtime generates OpenMP[50] [19] code that is compiled using GNU g++[51] (with –fopenmp activated) where a stand-alone executable binary is produced and executed with parallelism handled by the OpenMP runtime. Alternately, the optimized Boolean circuit is interpreted by a parallel Boolean circuit interpreter [20], which allows a fine-grained dynamic optimisation of the parallelism.

In order to obtain adequate performance, the table below reports the required hardware specification.

Table 20 – Other Test Bed Requirements

ID	Goal	Priority	Requirement
C3ISP-Tst-001	SME-US-4 SME-US-6 SME-UC-3	MUST	To accommodate for FHE requirements, C3ISP uses a server with the following specifications[52]: • 40 cores; • 224 GByte RAM; • 5 TByte Storage.

[49] HElib is a C++ software library that implements homomorphic encryption (HE), https://github.com/shaih/HElib

[50] OpenMP API is a specification for parallel programming in C/C++/Fortran, http://www.openmp.org/

[51] GNU C++ compiler from the GNU Compiler Collection, https://gcc.gnu.org/

[52] A server with this setup is provided at CNR premises (Pisa, Italy). Such server satisfies C3ISP-Per-002 requirement.

4. Conclusions and Next Steps

In this document we outlined the system requirements, both functional and non-functional that will be used as the foundation to build the C3ISP reference architecture. In particular, the approach was to understand the Pilots' and partners' needs in order to drive the requirements definition. We put strong emphasis on the data sharing and data analytics features (which are the core C3ISP functionalities), as well as on the security aspects, including needs and constraints of the homomorphic encryption techniques that we plan to use to address specific scenarios.

We also defined the approach and drafted the requirements of how we will setup the development environment and the test and integration environment (test bed). The former will be used by the consortium partners as a common ground to build the C3ISP software subsystems and components, by achieving a high quality product prototype. The latter will be used by the Pilots to exploit the C3ISP framework services for realising their use cases.

Starting from M6 we will concentrate on the definition of the reference architecture: we have already sketched the overall ideas throughout the document, in the definition of the C3ISP Framework (1.2), the use of the CTI data, the leverage of cybersecurity-related standards (STIX, TAXII, CybOX, OpenC2), or the usage of homomorphic encryption and computation, just to mention the most important. The architecture definition will proceed in parallel with the activities of WP8, with which we will collaborate tightly to understand the tools and techniques that will need to be integrated and used in C3ISP.

The next major goal is to conceive a concrete and viable definition of the reference architecture at M12 ready to be worked on for later implementation activities.

5. 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