D5.2

SME Pilot Design and Integration

WP5 – SME Pilot

C3ISP

Collaborative and Confidential Information Sharing and Analysis for Cyber Protection

Due date of deliverable: 30/09/2017
Actual submission date: 30/09/2017

30/09/2017

Version 3.0

Responsible partner: BT
Editor: Ali Sajjad
E-mail address: ali.sajjad@bt.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: Ali Sajjad (BT), Wenjun Fan (UNIKENT), Rogerio de Lemos (UNIKENT), David Chadwick (UNIKENT), Mark Shackleton (BT), Paul Galwas (DIGICAT), Jozef Dobos (3DRepo), Jovan Stevovic (CHINO)

Approved by: Gianpiero Costantino (CNR), Jonas Boehler (SAP)

Revision History

Version	Date	Name	Partner	Sections Affected / Comments
1.0	15/08/2017	Ali Sajjad	BT	ToC
1.1	24/08/2017	Ali Sajjad	BT	Sections 2, 3 and 4
1.2	22/09/2017	Ali Sajjad, David Chadwick, Rogério de Lamos, Wenjun Fan	BT, UNIKENT	All sections updated.
1.3	25/09/2017	Ali Sajjad, David Chadwick, Rogério de Lamos, Wenjun Fan	BT, UNIKENT	All sections updated.
1.4	26/09/2017	Ali Sajjad	BT	Ready for review
1.5	27/09/2017	Rogério de Lemos, Joanna Ziembicka	UNIKENT	Reviewed and added comments.
1.6	28/09/2017	Gianpiero Costantino	CNR	Review and editing of whole document
3.0	20/09/2017	Ali Sajjad	BT	Final

Executive Summary

This document presents the detailed design and architecture of the SME Pilot, as well as its integration with the C3ISP Framework. The detailed design of the SME Pilot has been constructed with the help of the functional requirements. These requirements were elicited from the stakeholders of the SME Pilot and described more formally in D5.1 in form of User Stories and comprehensive UML Use Case diagram. In this deliverable, we have described the design of the SME Pilot architecture with the help of FMC Block Diagrams. Each block corresponds to a high-level subsystem and the connections between them identify the roles and responsibilities undertaken by them and how these subsystems collaborate with each other in order to achieve the desired functionality. We have taken into account the constraints being imposed by the C3ISP Framework, as described in WP7 and WP8, and these constraints are reflected in the component architecture of the SME Pilot. After giving the general understanding of the SME Pilot architecture through the individual subsystems in order to gain a general understanding of the Pilot scenario, we described what each component of a subsystem does in a more systematic way. This was done by discussing the interfaces and relationships between the individual components through the use of UML Component diagrams. This deliverable also explains how the major data or system entities are stored, processed and organized in the SME Pilot, as well as how the security model for the architecture is designed, especially with respect to authentication and authorization responsibilities of different subsystems. Finally, the hardware and software deployment requirements for the C3ISP testbed environment are presented, along with the mappings between the requirements described in D5.1 and various components of the SME Pilot architecture. Most importantly, we describe how these components are going to interact with the C3ISP Framework, specifically with its ISI and IAI sub-systems.

Table of contents

1. Introduction

1.1. Purpose

The main purpose of this deliverable is to document the work and contributions towards the task T5.2 of the Work Package 5 (WP5) of the C3ISP project. The primary aim of this task is to provide the architecture of the SME Pilot, starting with a high-level detail of the architecture's subsystems and identifying specific software components fine-tuned for the specific pilot functionalities. The SME Pilot architecture was derived from the reference pilot scenario, described in deliverable D5.1, and takes into account the requirements of the C3ISP Framework's reference architecture, described in D7.2. Moreover, the task deals with the necessary integration activities needed for the introduction of the C3ISP Framework to the SME Pilot scenario. This includes the interaction with the Managed Security Services and security software, and the data analysis supported by the C3ISP Framework. The SME Pilot architecture was analysed in the context of various scenarios that capture the level of trust placed by the SMEs on the information sharing network.

1.2. Scope

The overall scope of this document is to produce a high level block design and component-level design of the SME Pilot architecture, based on the four use cases identified in the deliverable D5.1 [1].

Based on the first use case (SME-UC-1), the integration and interaction with Managed Security Services (MSS), that will source all the CTI data to be shared with the C3ISP Framework, has been realised in the SME Pilot architecture.

Based on the other three use cases (SME-UC-2 to SME-UC-4), the SME Pilot architecture incorporated the necessary interactions and activities needed for establishing and managing the integration between the C3ISP Framework and the SME Pilot's components. These use cases cover the scenarios about selecting Data Sharing Agreement (DSA), carrying out different Data Manipulation Operations (DMO) based on the level of trust placed by the SMEs on the C3ISP Framework, and the distribution of the analysis results produced by the C3ISP Framework.

1.3. Overview

The remainder of this deliverable is organized as follows:

Section 2 gives a general description of the functionality, context and design of the SME Pilot scenario.

Section 3 reports the high level design and architecture for the SME Pilot. This section also identifies, for each subsystem, the roles and responsibilities assigned to it.

Section 4 details the subsystems, identified in the previous section, in terms of their key components.

Section 5 describes the integration plan of the SME Pilot architecture, and deployment models with the corresponding deployment models of the C3ISP Framework, as provided in D7.2.

Section 6 describes how the major data or system entities are stored, processed and organized.

Section 7 reports on the details of the authentication and authorisation processes established between different components of the SME Pilot architecture.

Section 8 gives out the detailed hardware and software requirements that are needed for the C3ISP SME Pilot testbed. This testbed will implement and run the different components of the SME Pilot, as well as the C3ISP Framework.

Section 9 provides a cross-reference to show which system components satisfy each of the functional requirements listed in the deliverable D5.1.

Finally, Section 10 presents our conclusions.

1.4. Definitions and Abbreviations

Acronym	Definition
AES	Advanced Encryption Standard
API	Application Programming Interface
AuthN	Authentication
AuthZ	Authorization
BD	Block Diagram
BT	British Telecom
C3ISP	Collaborative and Confidential Information Sharing and Analysis for Cyber Protection
CD	Component Diagram
CEF	Common Event Format
CERT	Computer Emergency Response Team
CSP	Cloud Service Provider
CSS	Common Security Services
CSV	Comma Separated Values
CTI	Cyber Threat Information is any information that can help an organization identify, assess, monitor, and respond to cyber threats
CVE	Common Vulnerability and Exposure
DBMS	Database Management System
DDoS	Distributed Denial of Service
DMO	Data Manipulation Operations
DoS	Denial of Service

DPO	Data Protected Object
DPOS	Data Protected Object Storage
DSA	Data Sharing Agreement
ENT	Enterprise
FHE	Full Homomorphic Encryption
FMC	Fundamental Modelling Concepts
GDPR	General Data Protection Regulation
IAI	Information Analytics Infrastructure
IDE	Integrated Development Environment
IdP	Identity Provider
IDS	Intrusion Detection System
IoT	Internet of Things
IP	Internet Protocol
IPS	Intelligent Protection Service (The MSS used in WP5)
ISI	Information Sharing Infrastructure
ISP	Internet Service Provider
LEEF	Log Event Extended Format
MoSCoW	Must have, Should have, Could have, and Won't have
MSS	Managed Security Service
NFR	Non Functional Requirement
OASIS	Organization for the Advancement of Structured Information Standards
OAuth	Open Authorization
OWASP	Open Web Application Security Project
sid	Session Identifier
SME	Small and Medium Enterprise
SSH	Secure Shell
SSS	Security Scan Software
STIX	Structured Threat Information Expression

UC	Use Case
UML	Unified Modelling Language
US	User Story
VM	Virtual Machine
WP	Work Package

2. System Overview

The aim of the SME Pilot is to design a solution that enables SMEs to collect relevant CTI data to be shared with the C3ISP Framework in order to be analysed. The CTI data is governed by the policies defined in the DSA, which is in effect between an SME and the C3ISP Framework. The aggregated CTI from different SMEs, and other sources, is analysed by the C3ISP Framework, with the results to be shared with the SMEs.

Some of the main benefits expected as the outcome of the SME Pilot are:

- The SMEs are able to choose the type of confidentiality controls that are appropriate for safeguarding their CTI data on the C3ISP Framework, e.g., to go for either open access, or data anonymization techniques, or even use homomorphic encryption based techniques.
- Due to the availability of different data access, confidentiality and privacy options, the SMEs can confidently share all or portions of their CTI data with the C3ISP Framework, or even with other non-trusted third parties.
- The C3ISP Framework can incorporate diverse techniques for analysing the CTI data being shared, without the SMEs worrying about issues like information leakage, as this process is transparent for the SMEs.

3. System Architecture

The SME Pilot scenario involves three main actors. First is the SME(s) that wants to take part in this collaboration effort. Second is the multi-tenant, cloud-based, Managed Security Service (MSS) that enables its tenants (the SMEs) to assess the security threats and vulnerabilities of the data and applications they run in VMs hosted on multiple cloud platforms. Third and last is the C3ISP Framework, which can collect and aggregate CTI from different sources and perform threat and vulnerability analysis on the combined data to produce useful results and reports.

The MSS can be deployed and configured on the either public or private cloud environments. The SMEs can subscribe to all of the security services offered by the MSS or even a subset of them, depending on their needs and requirements. The Intelligent Protection Service [2] is an MSS developed by BT, aimed towards SME customers, which provides services such as:

- Anti-Malware: Protects computers from viruses, Trojans, spyware and other harmful software.
- Web Reputation (Black listing): Protects against web threats by blocking access to malicious URLs.
- Firewall: A bidirectional, state-full firewall that is responsible for making sure that packets originating from unauthorized sources do not reach their targets.
- Intrusion Detection/Prevention: Protects computers from being exploited by attacks against known vulnerabilities as well as against SQL injections attacks, cross-site scripting attacks and other web application vulnerabilities.
- Integrity Monitoring: Allows monitoring specific areas on a computer for changes. It has the ability to monitor installed software, running services, processes, files, directories, open ports, registry keys, and registry values.
- Log Inspection: Allows monitoring the logs and events generated by the operating systems and applications running on the computers.

The MSS can be managed by the SMEs themselves, if they have sufficient capability and skills, or could be outsourced to a trusted third party, e.g., BT.

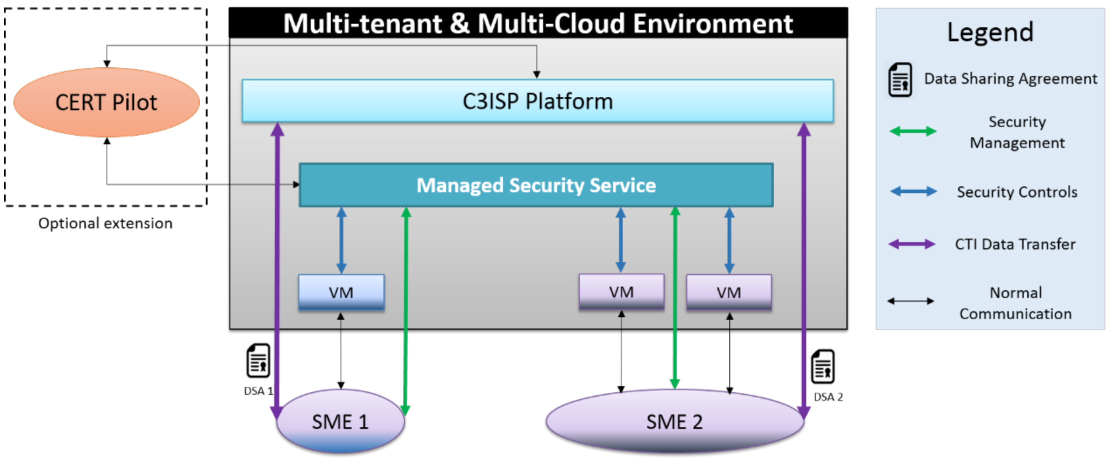


Figure 1. The original SME Pilot scenario

Figure 1 shows the proposed SME Pilot architecture based on the original high-level architecture of the SME Pilot described in D5.1 [1]. In this architecture, the SMEs implement the functionality to collect and process the CTI data and then share the data with the C3ISP Framework directly. The SMEs are also responsible for negotiating the DSA with the C3ISP Framework and enforcing the resulting policies at their end.

However, it should be as simple as possible for the SMEs to participate in the C3ISP eco-system. This means offloading bulk of the management and operational processes from the SMEs. Also, it is very important to make the integration process with the C3ISP Framework as simplified and seamless as possible. A common logical way to achieve this is to use a middleware that acts as a bridge between the SMEs and the C3ISP Framework.

3.1. C3ISP Gateway

In SME Pilot, we introduce a new subsystem, called the C3ISP Gateway, which acts as the middleware between the SMEs and C3ISP Framework. The main goal of the C3ISP Gateway is to manage and process the SMEs CTI data, before it is shared with the C3ISP Framework, as well as act as a proxy for the SMEs when dealing with the interactions pertaining to the DSA and information analysis subsystems of the C3ISP Framework.

The C3ISP Gateway also gives SMEs the flexibility of deployment depending on the trust-level between the SMEs and their cloud service providers. That is, if the SMEs fully trust their cloud service providers, they can host the C3ISP Gateway on the cloud service provider's infrastructure, but if that's not the case, then the SMEs can host the C3ISP Gateway using their own IT infrastructure. In the former case, the SMEs can outsource the management and operational costs of the C3ISP Gateway to their service provider, whereas in the latter case, the SMEs will be completely responsible for the administration of the C3ISP Gateway and it will be essentially act as a local client of the C3ISP Framework. Nevertheless, logically the C3ISP Gateway will still be a middleware between the SMEs and C3ISP Framework and so in the SME Pilot it will be treated as a component that is orthogonal to the SMEs and C3ISP Framework components.

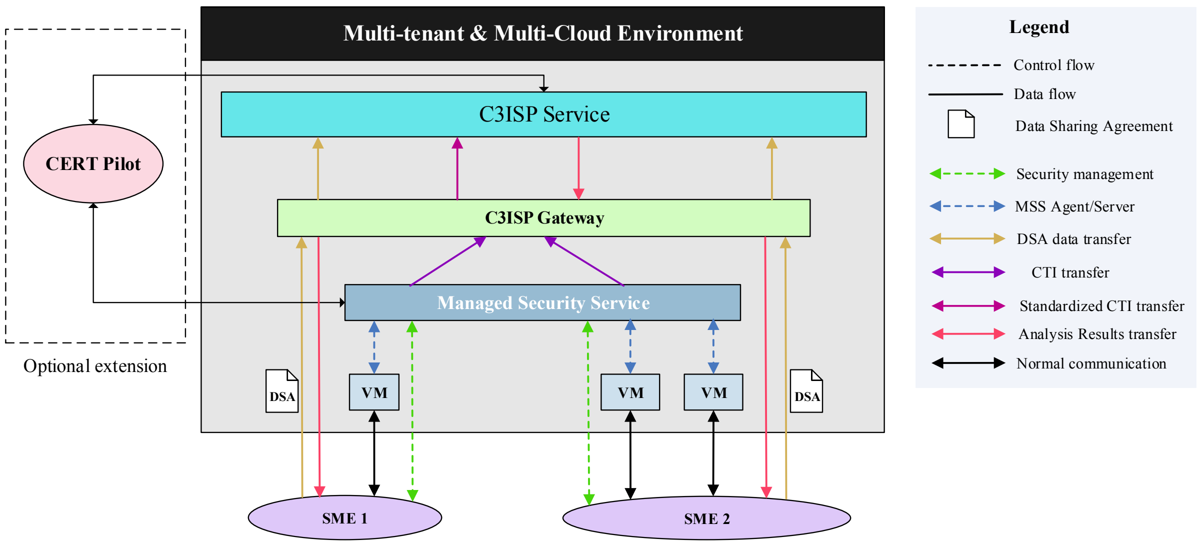


Figure 2. The new SME Pilot scenario

The new and updated high-level overview of the SME Pilot scenario is shown in **Figure 2**. The SMEs communicate with the MSS to manage the security of applications and services running on their VMs, which may be deployed on different cloud platforms. The MSS enforces the security policies and rules directly on the VMs, through an MSS Agent installed in the VMs. The SMEs delegate the tasks of collecting and processing the CTI to the C3ISP Gateway, which has the capability of collecting, processing and sending the CTI data in STIX[1] (Structured Threat Information Expression) format to the C3ISP Framework. The SMEs also accomplish the task of enforcing the Data Sharing Agreement (DSA) through the C3ISP Gateway, which processes the CTI according the DSA, before sending the data to the C3ISP Framework.

[1] STIX is an open-source language and serialization format used to exchange CTI: https://stixproject.github.io

4. Component Architecture

In this section, we detail the main components of the proposed high-level architecture shown in the previous section.

We first describe these components with the help of block diagrams, which are used to provide a high-level view of a system. The components are represented by blocks connected by lines that show the relationships of the blocks. We make use of the Fundamental Modelling Concepts (FMC) [3] notation to construct the block diagrams, which is an easy to understand rigorous graphical notation heavily used in the software industry. The FMC block diagrams depict the static structure of a system, in terms of its components and relationships between system components. Active system components are represented as rectangles, storage components as rounded rectangles and communication channels as lines. For instance, Figure 3 shows the new SME Pilot scenario (depicted in **Figure 2**) in FMC notation.

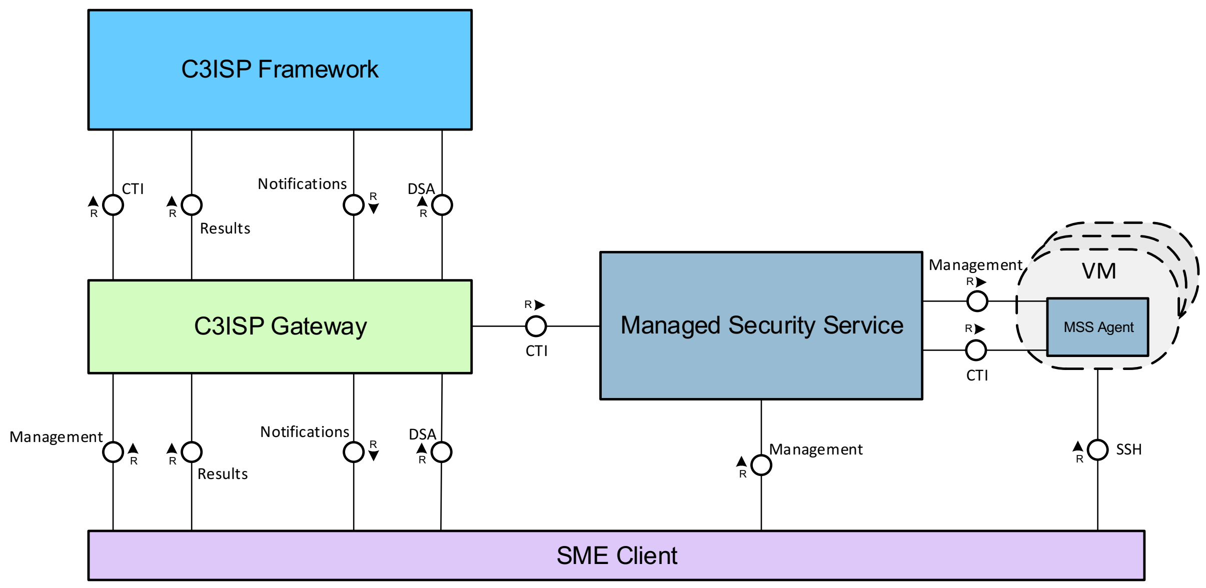


Figure 3. FMC Block diagram of the SME Pilot architecture

However, due to their high-level perspective, block diagrams may not offer the level of detail required for more comprehensive planning, integration or implementation. In such cases, we model the architecture using UML Component diagrams [4], which aim to highlight the system components and relationships between them, as well as their provided and required interfaces and ports.

4.1. Block Design

In this section, we expand upon the Use Cases (SME-UC-1 to SME-UC-4) described in detail in the C3ISP project deliverable D5.1 [1].

4.1.1. SME-BD-1: Block Design of SME-UC-1

The main purpose of this Use Case is that SMEs should be able to subscribe to an MSS. This enables application and host protection on SMEs VMs, and would also start the process of MSS collecting and logging all the security events it receives from the MSS Agents. The FMC block diagram of this Use Case is shown in Figure 4.

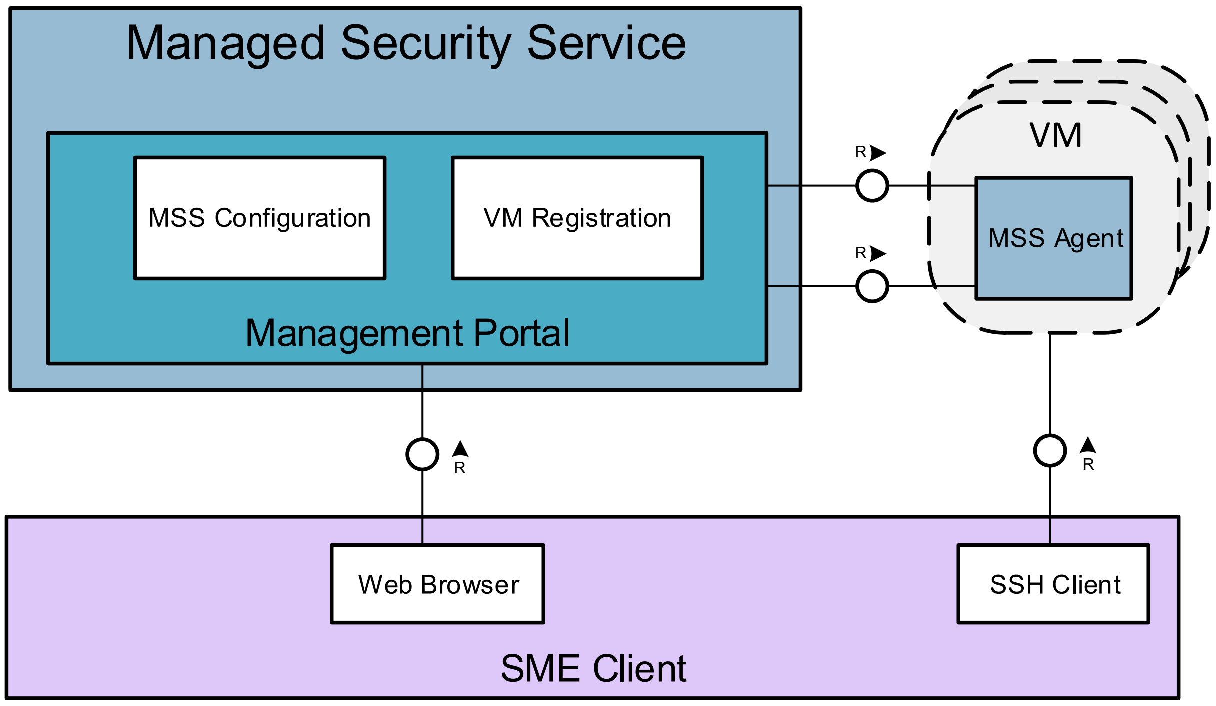


Figure 4. FMC Block diagram of the MSS subscription Use Case (SME-UC-1)

The SME client communicates securely with the web-based Management Portal of the MSS. In the context of this Use Case, the main responsibility of the MSS Management Portal is to subscribe the SME Clients to the MSS. Once the subscription process is completed, SMEs are able to login into the MSS Management Portal, where two components provide further functionalities. The VM Registration component allows the SMEs to register new VMs into the MSS or customise the registration of existing VMs, whereas the MSS Configuration component enables the SMEs to activate/enable and configure the types of Security Services they require.

The MSS Management Portal deploys the Security Services subscribed by the SMEs by installing an MSS Agent on the SMEs VMs. The MSS Agent enforces the Security Services policies on the VMs, and relay back all the security event logs back to the MSS. The Security Services process these event logs, and store the CTI discovered from these events in a database.

4.1.2. SME-BD-2: Block Design of SME-UC-2

The SME-UC-2 has been updated due to the changes in the SME Pilot's high-level architecture. The updated Use Case has been described in Appendix 1. The main purpose of this Use Case is that the SMEs should be able to select a Data Sharing Agreement (DSA) that is offered by the C3ISP Framework. This DSA will be the basis for all the processing carried out on the CTI by the C3ISP Gateway, and for enforcing of all the data sharing operations between the C3ISP Gateway and the C3ISP Framework. The FMC block diagram of this Use Case is shown in Figure 5.

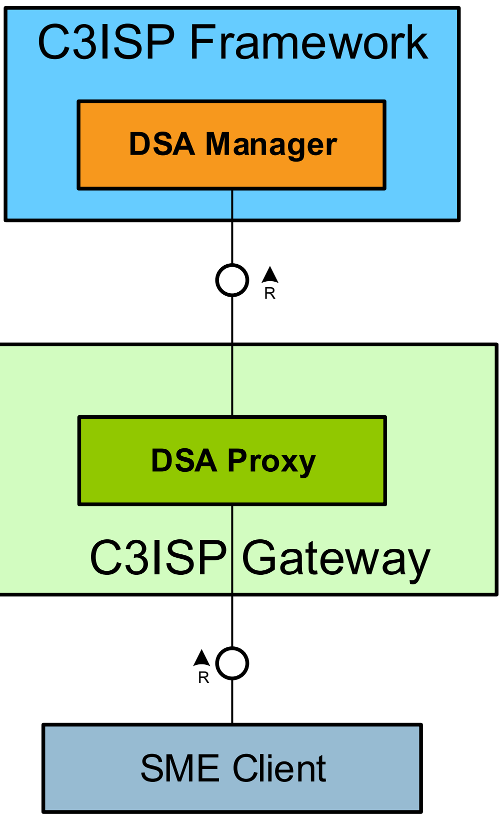


Figure 5. FMC Block diagram of the DSA selection Use Case (SME-UC-2)

The SME Client can establish a secure communication channel with the DSA Proxy, which is a component of the C3ISP Gateway. The DSA Proxy acts as a forward proxy to the DSA Manager, a component of C3ISP Framework, which allows the SMEs to select a DSA from a set of pre-existing policy templates. The Data Sharing Agreement (DSA) will include rules regarding CTI processing, CTI sharing and authorisation & access control to CTI.

4.1.3. SME-BD-3: Block Design of SME-UC-3

The main purpose of this Use Case is that the C3ISP Gateway should be able to collect the CTI data from the MSS on behalf of the SMEs and should be able to process and share the CTI with the C3ISP Framework in accordance with the DSA. The FMC block diagram of this Use Case is shown in Figure 6.

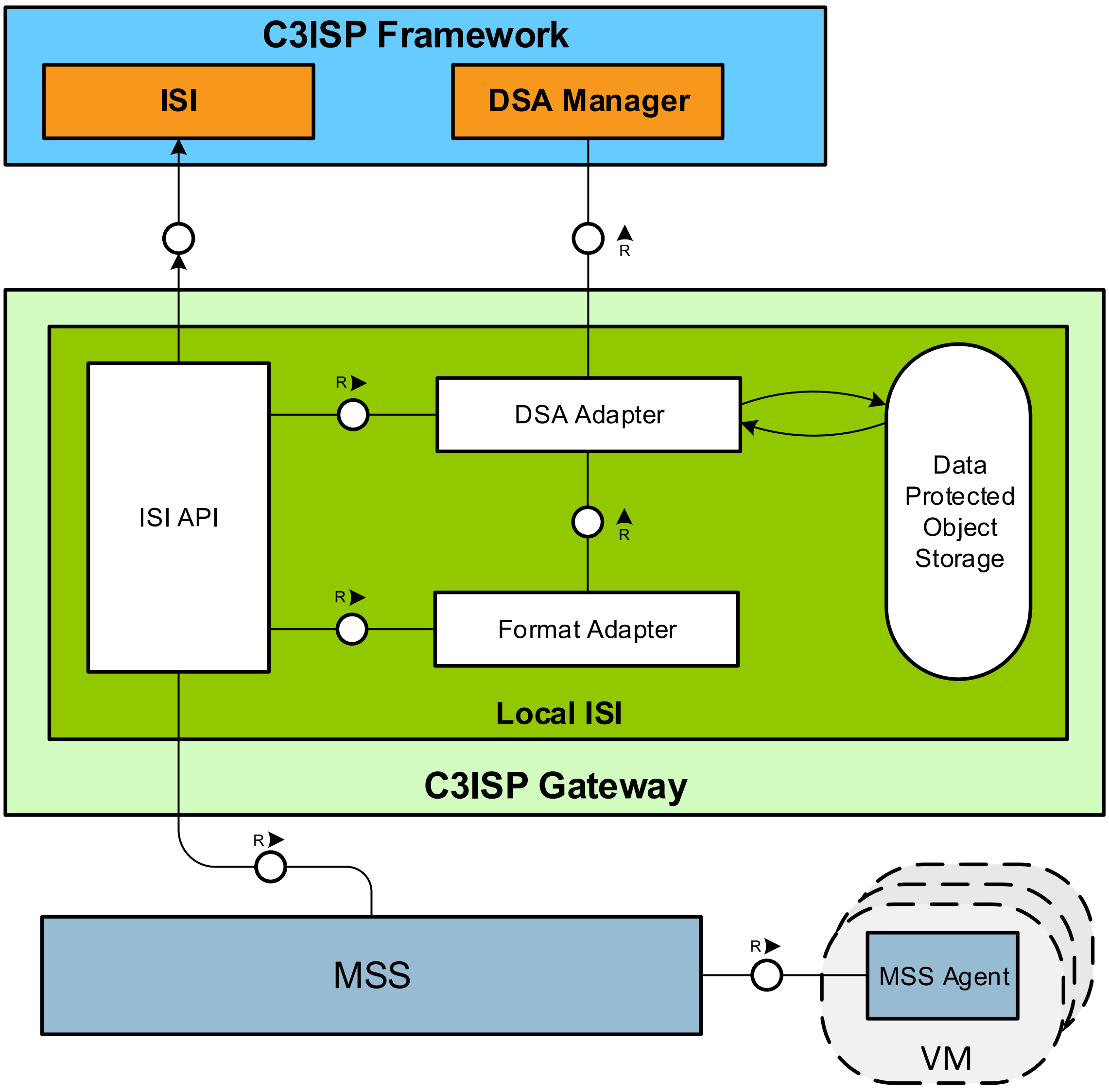


Figure 6. FMC Block diagram of the CTI collection and processing Use Case (SME-UC-3)

The MSS is the central component that stores all the CTI originating from the MSS Agents deployed in the SMEs VMs. The ISI API component in the C3ISP Gateway collects and filters the CTI from the MSS according to the DSA. The CTI is then converted into a standardized format (STIX) by the Format Adapter component, and forwarded to the ISI subsystem of the C3ISP Framework. In case the DSA calls for further processing on the CTI, in terms of confidentiality and privacy requirements, the CTI is forwarded to the DSA Adapter that can modify CTI data before it is uploaded to the C3ISP Framework.

4.1.4. SME-BD-4: Block Design of SME-UC-4

The main purpose of this Use Case is that the C3ISP Gateway should be able to retrieve the results, from the C3ISP Framework, of the analysis performed on the shared CTI data. The results can be in different forms, e.g., actions, recommendations, notifications etc. The FMC block diagram of this Use Case is shown in Figure 7.

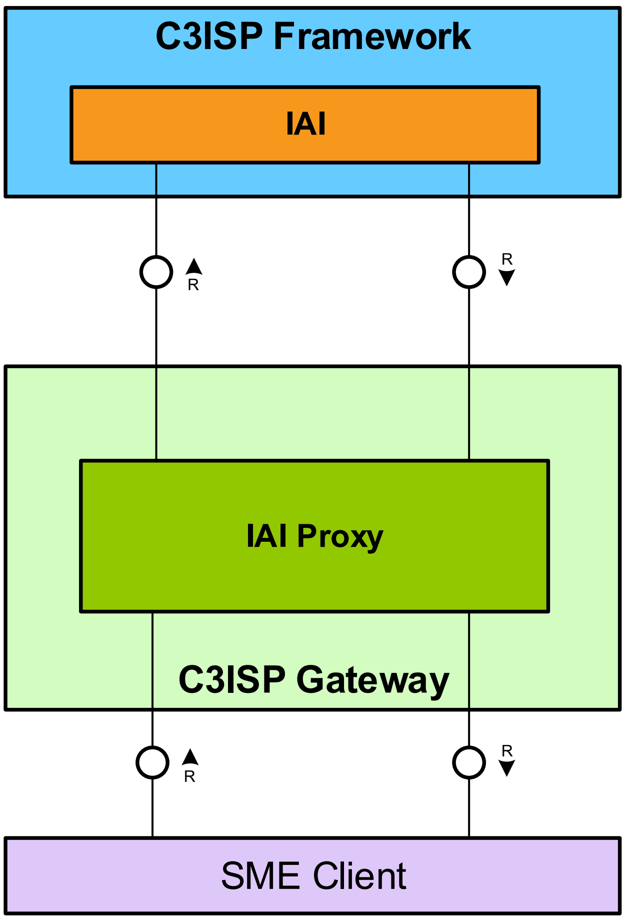


Figure 7. FMC Block diagram of the CTI analysis results Use Case (SME-UC-4)

The IAI Proxy component on the C3ISP Gateway is responsible for dealing with all the operations related to the retrieval, processing and notification of analysis results from the C3ISP Framework. The IAI Proxy component gets the analysis results from the IAI subsystem of the C3ISP Framework by issuing on-demand or periodic requests. The IAI subsystem is also able to filter out unwanted and non-relevant results, as per the SMEs' policies stated in the DSA. In case the C3ISP Framework detects a high-priority event (e.g., an on-going attack, or it is under attack itself), it can send an urgent alert to the IAI Proxy component on the C3ISP Gateway, which can relay this notification to the relevant SMEs so that they can take remedial actions.

4.2. Component Design

In this section, we break down the FMC block designs of the SME Pilot into their constituent components in a more systematic way, and discuss the interfaces and relationships between them using UML Component diagrams.

4.2.1. SME-CD-1: Component Design of SME-BD-1

The UML Component design of the SME-BD-1, which deals with the subscription of the SMEs virtual machines with the MSS, is presented in Figure 8. This diagram, which is a decomposition of the FMC block design of Figure 4, represents the SME-BD-1 in terms of its components, and provided and required interfaces connecting the components.

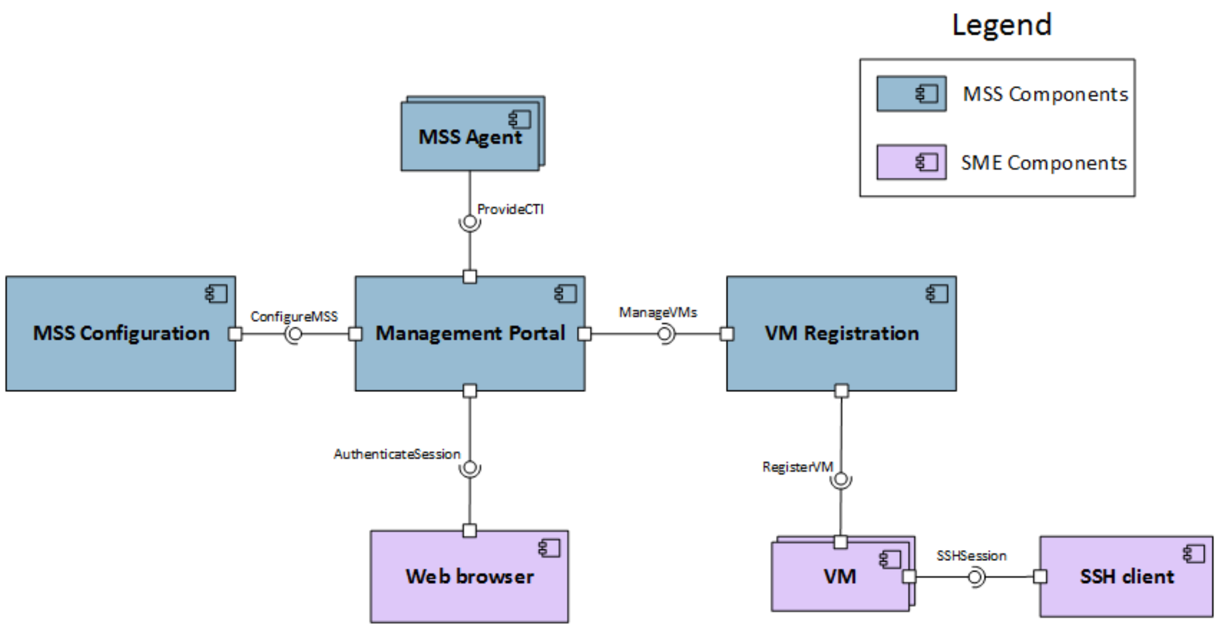


Figure 8. Component diagram for the SME-BD-1

The **MSS Agent**, which is the component responsible for collecting the CTI data from the VMs, shares CTI data with the Management Portal through the *ProvideCTI* interface. The **Management Portal** component provides three interfaces, *ManageVMs*, *ConfigureMSS*, and *AuthenticateSession*, which are consumed by the **VM Registration**, **MSS Configuration** and **Web browser** components, respectively. *ManageVMs* and *ConfigureMSS* are utilized by the SME through the *AuthenticateSession* interface. The **VM Registration** provides the *RegisterVM* interface through which individual VMs are registered and monitored by the MSS. All the VMs are running a SSH service so that the SMEs can connect to them remotely.

4.2.2. SME-CD-2: Component Design of SME-BD-2

The UML Component design of the SME-BD-2, which deals with the selection of the DSA by the SMEs, is presented in Figure 9. This diagram, which is a decomposition of the FMC block design of Figure 5, represents the SME-BD-2 in terms of its components, and provided and required interfaces connecting the components.

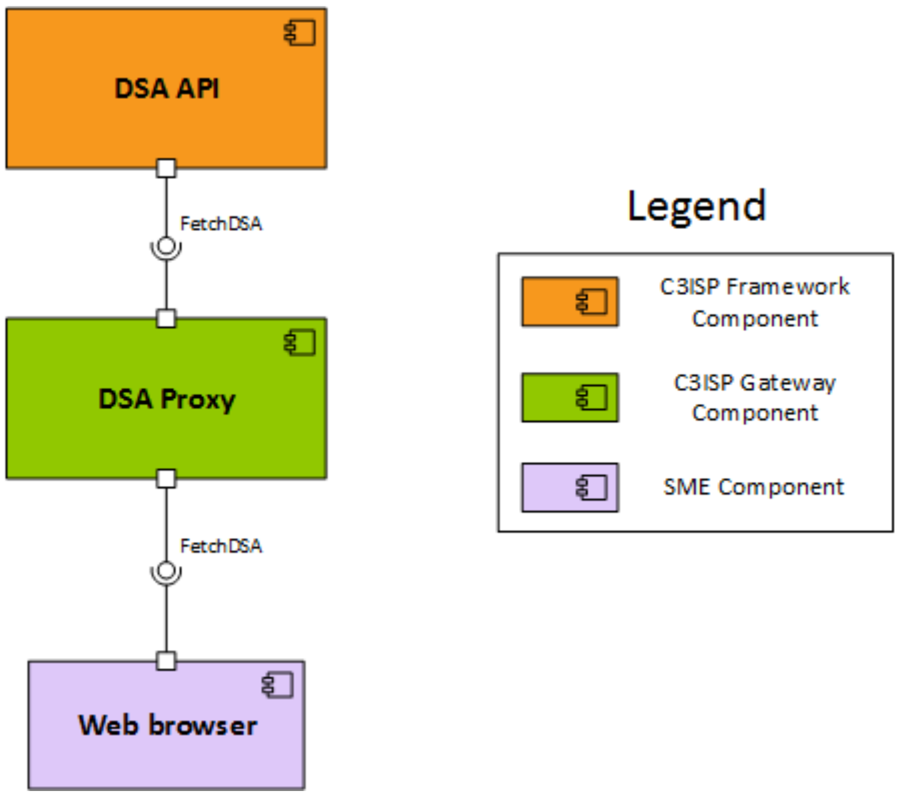


Figure 9. Component diagram for the SME-BD-2

The **DSA API** is a component of the DSA Manager subsystem of the C3ISP Framework, which is responsible for the creation of the DSAs between the C3ISP Framework and its Prosumers. As shown in the diagram of Figure 9, both the **DSA API** and the **DSA Proxy** component provides the same interface, *FetchDSA*, as the later component acts as a forward proxy for the former component.

4.2.3. SME-CD-3: Component Design of SME-BD-3

The UML Component design of the SME-BD-3, which deals with the collection and processing of CTI data from the MSS before sharing it with the C3ISP Framework, is presented in Figure 10. This diagram, which is a decomposition of the FMC block design of Figure 6, represents the SME-BD-3 in terms of its components, and provided and required interfaces connecting the components.

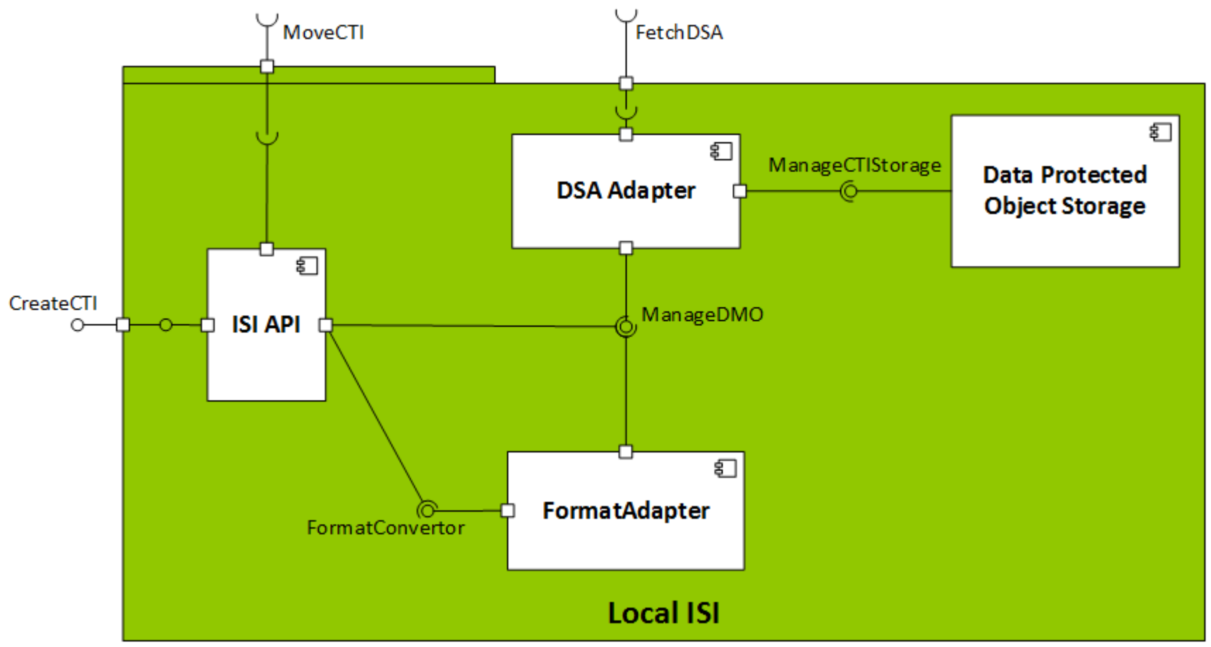


Figure 10. Component diagram for the SME-UC-3

The **ISI API** is the front-end component providing services to the C3ISP Prosumers. It orchestrates the processing flow with the other Local ISI components, interacting with the **DSA Adapter** and the **Format Adapter** All the operations of the ISI API are subject to the DSA policies associated to each CTI data object. In context of SME Pilot, the **ISI API** provides the *createCTI* interface used to submit a raw CTI data that will be paired with a DSA. Depending on the DSA, obtained through the *FetchDSA* interface of the **DSA API** component of the C3SIP Framework, the *FormatConvertor* or *ManageDMO* interfaces can be used to further process the CTI data before its storage in the **Data Protection Object Storage**. The data is stored in the form of C3ISP data bundle, i.e., the CTI data and the corresponding DSA, through its *PersistantCTIStorage* interface. The ISI API also consumes the *MoveCTI* interface of the C3ISP Framework's ISI subsystem, which is used to move a C3ISP data bundle from an ISI node to another ISI node.

4.2.4. SME-CD-4: Component Design of SME-BD-4

The UML Component design of the SME-BD-4, which deals with the collection of the analysis results and notifications from the C3ISP Framework, is presented in Figure 11. This diagram, which is a decomposition of the FMC block design of Figure 7, represents the SME-BD-4 in terms of its components, and provided and required interfaces connecting the components.

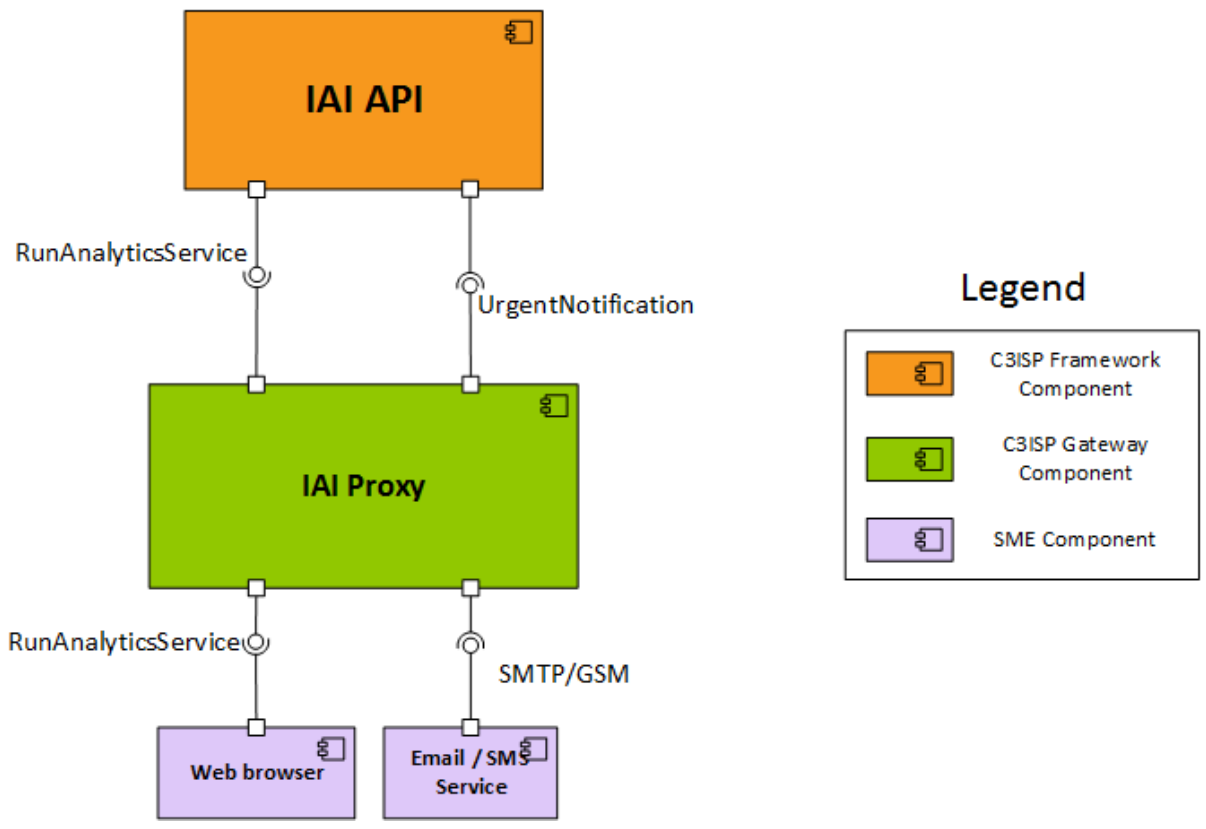


Figure 11. Component diagram for the SME-UC-4

The **IAI API** is a component of the IAI subsystem of the C3ISP Framework which offers the interface *RunAnalyticsService* for invoking analytics services on CTI data shared and provided by the SMEs. As shown in the diagram of Figure 11, both the **IAI API** and the **IAI Proxy** components provide the same interface, *RunAnalyticsService*, as the latter component acts as a forward proxy for the former component. Additionally, the **IAI Proxy** provides the *UrgentNotification* interface to the **IAI API** in order to share critical incidents like data breaches with the SMEs.

5. Integration Plan with C3ISP Architecture

It is anticipated that the SME Pilot will use the 'Hybrid' C3ISP deployment model [5]. In the Hybrid deployment model, the ISI subsystem is present both on-premises (locally) and the centralised C3ISP Framework and the 'Prosumers' interact with the 'local ISI'. The IAI is only present on the centralised C3ISP Framework and the 'Prosumers' interact with the 'remote IAI'. Note that for the SME Pilot, the C3ISP Gateway fulfils the role of a Prosumer (a producer and consumer of CTI). The Hybrid deployment model is shown in FMC notation in Figure 12, where the *green* colour is used to delimit the trusted zones (where the Prosumer has more control), and *orange* colour for the untrusted ones (where the Prosumer has less control).

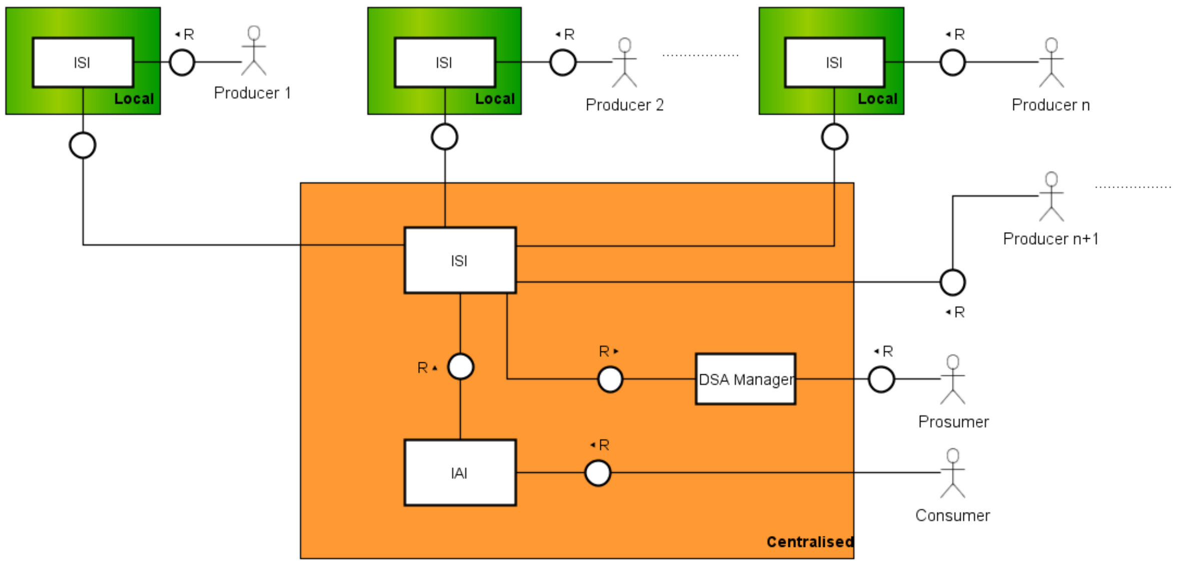


Figure 12. Hybrid deployment model of the C3ISP Framework (Local ISI with Centralised ISI and IAI)

As noted above, it should be as simple as possible for SMEs to participate in the C3ISP eco-system, including offloading the bulk of the management and operational processes from the SMEs. The Hybrid deployment model supports this ethos, since the data processing load is taken by the C3ISP Gateway and analysis efforts are primarily borne by the C3ISP Framework itself. An SME need only fulfil the role of a Prosumer if it decides to become a direct Producer of the CTI data, as depicted by the *Producer n+1* in Figure 12.

Note that it is important for the C3ISP Framework to support and maintain logical consistency between **Figure 2** (high-level SME Pilot architecture) and Figure 12, as the C3ISP Framework develops. In this context, the C3ISP Framework needs to support the SME Pilot's key Use Cases identified in deliverable D5.1, namely:

- **SME-UC-2:** SMEs shall select a Data Sharing Agreement (DSA) that is offered by the C3ISP Framework. This DSA shall be the basis for all the processing carried out on the CTI data by the C3ISP Gateway, and for enforcing all the data sharing operations between the C3ISP Gateway and the C3ISP Framework.
- **SME-UC-3:** The C3ISP Gateway shall collect the CTI data from the MSS on behalf of the SMEs, and shall process and share the CTI with the C3ISP Framework in accordance with the DSA. This feature shall be carried out by the 'local ISI' subsystem.

- **SME-UC-4:** The C3ISP Gateway shall retrieve the results of the analysis performed on the shared CTI data from the C3ISP Framework. The results can be in different forms, e.g., actions, recommendations, notifications etc.

Figure 13 provides a graphical view of all the relevant components of the SME Pilot in context of the Hybrid deployment model.

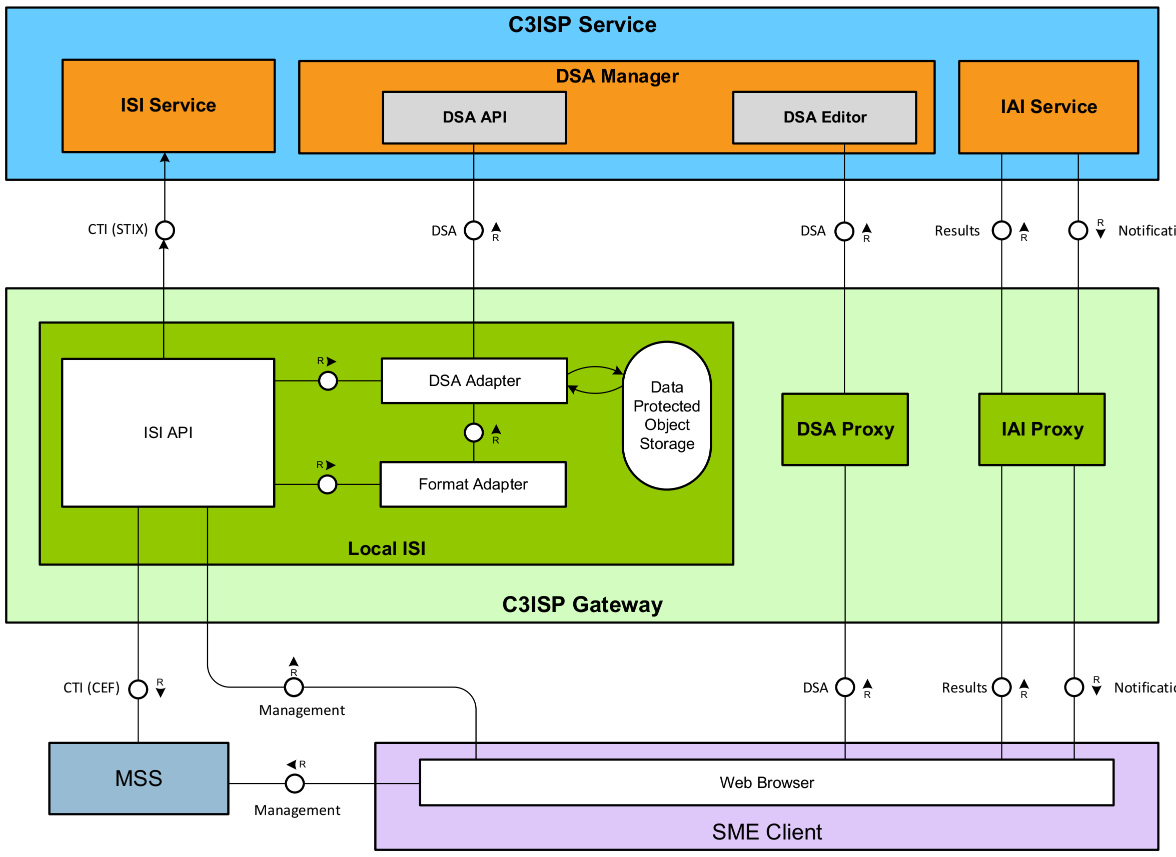


Figure 13. Overall block design of the SME Pilot in accordance with the Hybrid deployment model

As shown in the above FMC block diagram, C3ISP Gateway should be able host the *local ISI* component of the C3ISP Framework, which is able to communicate with the centralized ISI subsystem as well as the DSA Manager. The *ISI API* component in the *local ISI* collects and filters the CTI from the MSS according to the DSA in place between the SMEs and C3ISP Framework. The CTI data is then converted into a standardized format (STIX) by the *Format Adapter,* and uploaded to the remote ISI subsystem of the C3ISP Framework. In case the DSA calls for further processing on the CTI, in terms of confidentiality and privacy requirements, the CTI is forwarded to the *DSA Adapter* before uploading it to the remote ISI subsystem of the C3ISP Framework.

6. Data Model

This section introduces the schemas that the MSS uses to share the CTI data with the SMEs. Each security service offered by the MSS (firewall, anti-malware etc.) uses a slightly different schema to present the CTI data. This is due to the fact that the CTI data is classified by the type of security events that the security services of the MSS can monitor and detect.

6.1. Data Source

The original source of the CTI data are the computers (physical or virtual) on which the MSS agents have been installed, registered and configured. The MSS agents monitor the computers according to the security policies configured by the SMEs on the MSS. When some malicious or anomalous behaviour occurs in the protected computers that matches an existing policy, the MSS agents detect it, generate the security event, and send the event data to the MSS.

Once the MSS collects the CTI data, it stores it in a relational DBMS (Database Management System) e.g. MS SQL Server or Oracle. Each security service has a corresponding table in the DBMS for storing the CTI data belong to it.

6.2. Data Retrieval

The MSS offers the SMEs three ways to retrieve the CTI data of their interest from its DBMS.

- Export CTI data in CSV format through a Web interface
- Export CTI data to a Syslog server in CEF or LEEF format
- Query-based retrieval of CTI data through a Web Service

Out of these, the Web Service based approach is the most appropriate with respect to automation and integration in the SME Pilot. The MSS Web Service provides a SOAP and REST-based API for querying and retrieve the CTI data. The API also allows setting of basic filters (with *greater than*, *less than*, *equal to* operations) as the parameters in the API calls. The filters can be metrics like event IDs, host names (the protected computer's name), and time ranges etc.

At present, the MSS offers six security services, as mentioned earlier. The XML schema used by the MSS SOAP API requests is the same for all the six security services, and is given in Table 1.

Table 1. XML Schema of the SOAP Requests

	Element Name	Element Type	Description
XXXEventRetrieve	sID	xsd:string	Authentication session identifier
	eventIdFilter	impl:IDFilterTransport	The IDFilterTransport to filter by
	hostFilter	impl:HostFilterTransport	The HostFilterTransport to filter by
	timeFilter	impl:TimeFilterTransport	The TimeFilterTransport to filter by

In the first column, the "XXX" should be replaced by the actual security service's name. The filters' detail descriptions can be described as follows:

- EventIdFilter: Used as a search criteria to limit the scope of objects returned by event transport object ID. Each event transport object, such as IntegrityEventTransport, includes an ID property that is assigned as the primary key of an event when it is generated by a computer agent. Using IDFilterTransport, it is possible to filter event retrieval by this event ID in order to retrieve a specific event by ID, or events that are greater or less than a specified ID. For example, a utility that is designed to retrieve all new events on an interval can use the event ID property to uniquely identify which events have already been retrieved. This way retrieval of duplicate events can be avoided.

- HostFilterTransport: Used as search criteria to limit the scope of objects returned by computer-related attributes, such as by a Group, a Security Profile, or a specific computer. The event retrievalrelated methods will require a HostFilterTransport that is empty to search for all events, or with specific properties populated to limit the scope of the search. For example, setting the HostFilterTransport securityProfileID property to the ID of a Security Profile will limit any event retrieval method calls to events that pertain to computers with the specific Security Profile assigned.
- TimeFilterTransport: Used as search criteria to limit the scope of objects returned by time related attributes, such as from, to, or a specific time. If the type is set to EnumTimeFilterType CUSTOM_RANGE, then the rangeFrom and rangeTo property will be required. If the EnumTimeFilterType SPECIFIC_TIME type is set, then the specificTime property will be required.

The XML schema used by the SOAP API responses of the individual security services of the MSS, as well as their filters' datatypes, can be found in Appendix 2.

7. Security Model

Table 2 summarizes the SME pilot's security model which includes authentication, authorization and communication encryption.

Table 2. Security Model of the SME Pilot

	Authentication	Communication Encryption	Authorization
SME to MSS	HTTP Basic Auth (see 7.1.1)	HTTPS	RBAC (see 7.1.4)
Gateway to MSS	Token Auth (see 7.1.2)	SOAP over TLS	RBAC (see 7.1.4)
SME to Gateway	HTTP Basic Auth (see 7.1.5)	HTTPS	See 7.1.5
Gateway to C3ISP	OpenID Connect (see 7.1.3)	TLS	see D7.2 Section 7.1

7.1. Authentication and Authorization

In this subsection, the authentication mechanisms between different components of the SME pilot as well as the gateway to the C3ISP platform is presented.

7.1.1. Authentication from SME to MSS

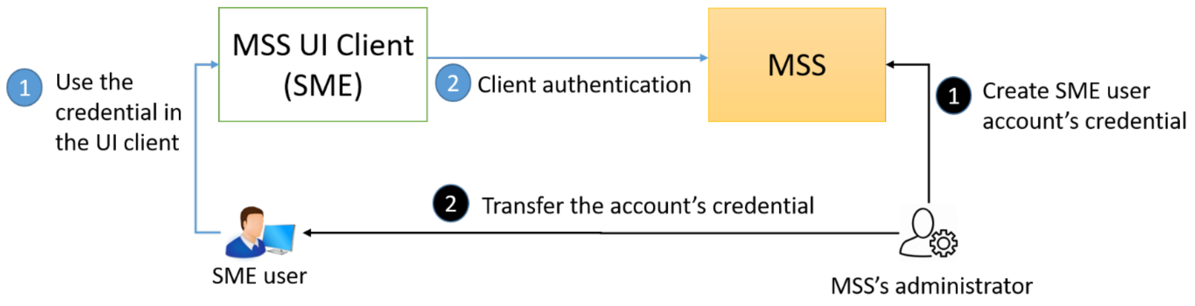

1 Use the credential in the UI client
MSS UI Client (SME)
2 Client authentication
MSS
1 Create SME user account's credential
2 Transfer the account's credential
SME user
MSS's administrator

Figure 14. HTTP Basic Authentication from SME to MSS: the black lines refer to configuration, and the blue lines refer to authentication

Before the session authentication can take place, the MSS administrator needs to do the following (shown in Figure 14 with black lines):

1) The MSS administrator creates a user account in the MSS along with the user's credential (username and password).
2) The MSS administrator transfers the credentials to the SME user.

Session authentication uses HTTP Basic Auth. This allows the MSS to authenticate (identify) a MSS UI client (shown in Figure 14 with blue lines):

1) The MSS UI Client (SME) (typically a web browser) uses HTTPS to connect to the MSS. The MSS's PKI certificate is used to authenticate the MSS to the SME user.
2) The SME user enters his username and password in the MSS UI client. The MSS UI client transfers these credentials to the MSS. The received credential allows the MSS server to authenticate the MSS UI client.

After the authentication, the SME user's commands can be transferred from the MSS UI client to the MSS in the same HTTPS session without further authentication.

7.1.2. Authentication from Gateway to MSS

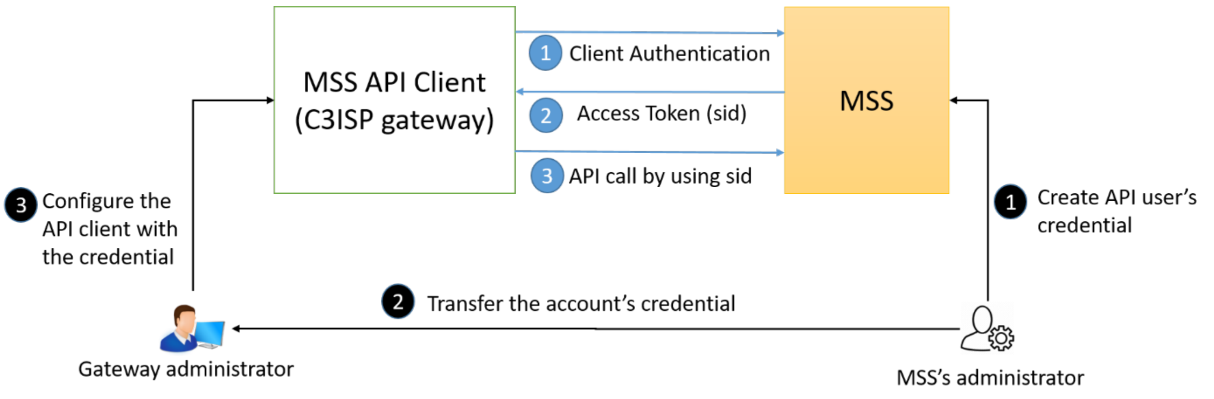

MSS API Client (C3ISP gateway)
1 Client Authentication
2 Access Token (sid)
3 API call by using sid
MSS
3 Configure the API client with the credential
1 Create API user's credential
2 Transfer the account's credential
Gateway administrator
MSS's administrator

Figure 15. Token Authentication from C3ISP Gateway to MSS: the black lines refer to configuration, and the blue lines refer to authentication

Before the authentication, the MSS system administrator needs to configuration the MSS as follows (shown in Figure 15 with black lines):

1) The MSS's administrator creates an API user's credential (username and password).
2) The MSS's administrator transfers the credential to the Gateway administrator.
3) The C3ISP Gateway administrator configures the API client with the credential.
4) The C3ISP Gateway administrator configures the API client with the X.509 certificate of the MSS.

The Token Authentication method used for authenticating the API client can be described as follows (shown in Figure 15 with blue lines):

1) The API client established a HTTPS connection to the MSS and identifies the MSS from its X.509 certificate. The API client proves its identity to the MSS by passing its credential across the secure TLS connection.
2) The MSS authenticates the API client and issues an Access Token to it. Typically, the Access Token is a session identifier (sid).
3) Subsequently, the API client can use the token (sid) to access the MSS Web Service. Once the application completes, the session should be ended and the sid discarded.

7.1.3. Authentication from C3ISP Gateway to C3ISP Framework

C3ISP Framework plans to implement Federated Identity Management for all the communications between its API services and its users (see D7.2, Section 7.1 [5]). It is anticipated that the OpenID Connect protocol [6] will be used, which is based on OAuthv2 [7]. In this mode of communication, the client is first redirected from the C3ISP API to an Identity Provider (IdP) which holds the user's login credentials. The user presents the credentials to the IdP and is authenticated. The user is then redirected back to the C3ISP API along with an Authz Code that is validated by the C3ISP API. If the code is valid the user is granted access. More details about this can be found in D7.2, Section 2.1 and 7.1.

7.1.4. Authorization of MSS users

The MSS uses the Role-Based Access Control (RBAC) approach for facilitating authorization. Each MSS user is assigned one or more roles by the MSS administrator. Once the SME client is authenticated by the MSS, its roles are determined. The MSS administrator defines the operations and rights that are allowed for each role. These are selected from the following:

- Computer Rights

The MSS administrator can define the computer rights for the various roles:

1) Allow users to: view, edit, delete, dismiss alerts for, tag items for all computers or selected computers.
2) Allow computer file imports.
3) Allow Directories to be added, removed and synchronized.
4) Allow VMware vCenters to be added, removed and synchronized.
5) Allow Cloud Accounts to be added, removed and synchronized.

- Policy Rights

The MSS administrator can also define the policy rights for the various roles:

1) Allow users to: view, edit, and delete all policies or selected policies.
2) Allow new Policies to be created.
3) Allow Policy imports.

- User Rights

The MSS administrator can allow users to have the following user rights:

1) Change own password and contact information only;
2) Create and manage SME users with equal or less access;
3) Have full control over all Roles and Users; (a user who has full control is another Administrator)
4) Custom Rights:
   a. User: can view, create, edit and delete users.
   b. Roles: can view, create, edit and delete roles.
   c. Delegate Authority: Can only manipulate Users with equal or lesser rights.

The above lists the rights that can be configured to the various user roles. A user who is assigned to a certain role will own the corresponding rights.

7.1.5. Authentication and Authorisation of SME Client to C3ISP Gateway and C3ISP Service

The C3ISP Gateway comprises three functional components: the local ISI, the DSA proxy, and the IAI proxy. The same Federated Identity Management system will be implemented by the local ISI component of the C3ISP Gateway as the ISI component of the C3ISP Service. Consequently, the SME Client will be able to use the same (IdP issued) credentials for accessing the local ISI of the C3ISP Gateway, as for accessing the C3ISP Service (either directly or indirectly via one of the proxies in the C3ISP Gateway).

Since the local ISI is a (partial) copy of the C3ISP ISI, the authorization components of the C3ISP ISI are fully contained within the local ISI. Consequently, the same mechanisms for user authorization can be used by the local ISI of the C3ISP Gateway as are used by the C3ISP ISI. For full details of ISI Authorization, please refer to D7.2 Section 4 [5].

The DSA and IAI proxies will appear to the DSA Manager and IAI Service respectively to be a C3ISP SME client, and so will be redirected to the IdP for authentication. The proxies will pass the redirect back to the SME client, so that the SME client will connect to the IdP, allowing the user to authenticate directly with the IdP. In this way, the C3ISP Gateway never learns the login credentials of the SME Client. When the Authz Code is passed to the SME client by the IdP, the SME client will present this to the gateway, which will forward it to the appropriate C3ISP Service, thereby granting the user/SME client access to the C3ISP service.

8. Requirements for C3ISP Testbed Environment

8.1. Hardware Requirements

8.1.1. Hardware Requirements for the MSS

A hardware pre-requisite guideline for an SME scale deployment of MSS is given below:

- Processors: 2 Intel/AMD 64-bit
- Minimum RAM: 8 GB
- Hard Disk: 300 GB

8.1.2. Hardware Requirements for the SME

No dedicated hardware resources are required for the SME clients.

8.1.3. Hardware Requirements for the C3ISP Gateway

The hardware requirements for the C3ISP Gateway will be same as the Local ISI component of the C3ISP platform.

8.2. Software Requirements

8.2.1. Software Requirements for the MSS

The software required to install and configure an instance of the MSS are given below:

- Operating system: Windows Server 2012 or 2012 R2 (64-bit)
- Database: Microsoft SQL Server 2014

8.2.2. Software Requirements for the SME

The main software required by the SME clients will be a web browser. Some specific examples of supported web browsers are:

- Firefox 46.0.1+
- Microsoft Internet Explorer 11+ or Edge
- Google Chrome 50+
- Apple Safari 9+ (for Mac)

The SMEs are also required to install the MSS agents on their VMs and assets that need to be protected by the MSS. These agents will be provided by BT and can be installed on most current flavours of Windows and Linux based VMs.

8.2.3. Software Requirements for the C3ISP Gateway

The software requirements for the C3ISP Gateway will be same as those for the Local ISI component of the C3ISP platform. Additionally, to assist in deployment and integration into customer and partner environments, MSS has a SOAP [8] Web Service API, which allows for an easy, language-neutral method to externally access data and programming configurations.

Using a programming language that supports SOAP over HTTPS standard, a client application can be developed to make SOAP requests to the MSS. The language should be selected on the basis of familiarity, suitability for the task at hand, and language compatibility for the intended integration. Apache Axis works well and is the native implementation of the Web Service itself. The Microsoft .Net Framework's support for Web Services through Visual Studio is a very robust choice. A list of potential SOAP Web Service implementations that can be used is given in Appendix 2. Therefore, any programming language that supports XML or JSON encoding and the HTTP and HTTPS protocols can be used to develop a MSS API client application. In particular, taking into account the Java and Apache Axis API for interacting with the MSS SOAP API, the detailed information about the required libraries can be found in Appendix 4.

9. Requirements Matrix

The following table shows the mapping of the SME Pilot's User Stories and Use Cases with the relevant C3ISP objectives:

Table 3. Mapping the SME Pilot requirements with the corresponding C3ISP components

Use Cases	User Stories	Description	Coverage
SME-UC-1	SME-US-1	The SMEs should be able to utilise the services of a managed security service provider.	Using the Management Portal, via a Web Browser, it is possible to configure the managed security services so that SMEs are able to access the MSS. (**SME-CD-1**).
SME-UC-2	SME-US-2	The SMEs should be able to ~~negotiate~~ establish[2] a data sharing agreement with the C3ISP Framework.	The SME Client can access the DSA Editor via the DSA Proxy in the C3ISP Gateway. (**SME-CD-2**)
SME-UC-3	SME-US-3	The MSS should allow the SMEs to collect and process its CTI data	This is facilitated by the local ISI in the C3ISP Gateway collecting and processing the CTI data on behalf of the user. (**SME-BD-3).**
	SME-US-4	The SMEs should be able to share their CTI data with the C3ISP Framework in a standardised format	This is facilitated by the Format Adapter in the C3ISP Gateway. **(SME-CD-3)**
	SME-US-5	The C3ISP Framework needs to provide confidentiality and integrity according to the SMEs needs	This is facilitated by the DSA Adapter in the C3ISP Gateway by retrieving the user's DSA policy and enforcing it. **(SME-CD-3)**
	SME-US-6	The SMEs should be able to anonymise some attributes of the data.	This is facilitated by DMO Engine in the DSA Adapter in the C3ISP Gateway by retrieving the user's DSA policy and enforcing it. **(SME-CD-3)**
NFR	SME-US-7	The MSS and C3ISP Framework should comply with the financial and computational costs set up by the SME	Non-functional requirement; there is no specific component in the SME architecture that address it. The relevant acceptance tests are available in D5.1 [1].
	SME-US-8	The MSS and C3ISP Framework should offer the SMEs an easy-to-integrate solution.	Non-functional requirement; there is no specific component in the SME architecture that address it. The relevant acceptance tests are

[2] The requirement has been changed from negotiate a DSA to establish a DSA.

			available in D5.1 [1].
SME-UC-4	SME-US-9	The C3ISP Framework should offer the SMEs customised results.	This requirement is satisfied by the IAI subsystem of the C3ISP Framework. **(SME-CD-4)**
	SME-US-10	The SMEs should be able to receive the C3ISP results in order to take action.	The results are received through the IAI Proxy in the C3ISP Gateway. **(SME-CD-4)**
	SME-US-11	The C3ISP Framework inform the SMEs about data breaches.	This requirement is satisfied by the IAI sending notifications via the IAI Proxy in the C3ISP Gateway to the SME Client. **(SME-CD-4)**
	SME-US-12	The C3ISP Framework should be able to handle insider threats from SMEs.	Non-functional requirement; there is no specific component in the SME architecture that address it. The relevant acceptance tests are available in D5.1 [1].

10. Conclusions and Future Work

In this deliverable, we have presented the architecture of the SME Pilot. The design caters for the 'Hybrid' deployment model proposed by the C3ISP Framework, which calls for the installation of a local ISI subsystem on the C3ISP Gateway in order to perform Data Manipulation Operations (DMOs) on the CTI data before it is shared with the C3ISP Framework. Hence the data collection, filtration and processing is done locally within the trust domain of the SMEs, whereas the data storage, access control, sharing and analytics processes are outsourced to the ISI and IAI subsystems of the C3ISP Framework.

We have also presented the Data and Security models for the SME Pilot, as well as the hardware and software requirement for the future implementation and integration of the various components of the C3ISP SME Pilot testbed. Lastly, we have provided a requirements matrix that maps the SME Pilot's requirements defined in D5.1 [1], against the components detailed in this deliverable.

The SME Pilot architecture described in this deliverable should act as a reference for the future implementation, testing and validation that has to be carried out under task T5.3 of the project, and has to be completed by month 26 of the C3ISP project.

11. References

[1] A. Sajjad, et. al., “Requirements for the SME Pilot,” C3ISP Deliverable D5.1, Ipswich, 2017.

[2] J. Daniel et. al., “Integrating Security Services in Cloud Service Stores,” in *Trust Management IX, Springer International Publishing*, Hamburg, Germany, 2015.

[3] R. Alpfelbacher, “FMC Visualization Guidelines,” 2017. [Online]. Available: http://www.fmc-modeling.org/notation_reference/. [Accessed 30 September 2017].

[4] UML, “Component Diagrams, Unified Modeling Language,” [Online]. Available: http://www.uml-diagrams.org/component-diagrams.html.

[5] M. Manea, “First Version of C3ISP Architecture,” C3ISP Deliverable D7.2, 2017.

[6] N. Sakimura et. al., “OpenID Connect Core 1.0 incorporating errata set 1,” The OpenID Foundation, 2014.

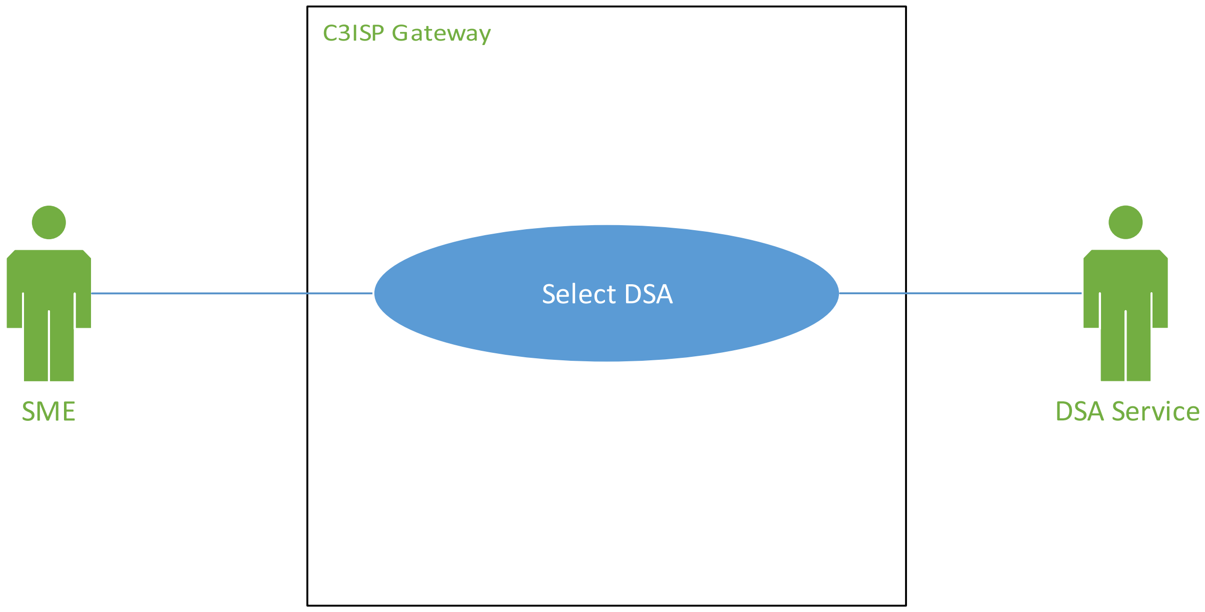
[7] D. Hardt, “The OAuth 2.0 Authorization Framework,” Internet Engineering Task Force (IETF), 2012.

[8] S. Weerawarana et. al., Web services platform architecture: SOAP, WSDL, WS-policy, WS-addressing, WS-BPEL, WS-reliable messaging and more, Prentice Hall, 2005.

Appendix 1. Updated SME-UC-2

The SME-UC-2 has been updated due to the changes in the Pilot's high-level scenario. The updated Use Case has been described in detail below:

SME-UC-2: Select the Data Sharing Agreement



Use Case Name	Select the DSA
Participating actors	• SME • DSA Service
Purpose	SMEs and DSA Service establish an agreement on the policies governing the data sharing process.
Priority	The DSA Service: • **must** be able to define and create data sharing agreements (a set of policies) • **must** offer a pre-defined set of data sharing policies to the SMEs to choose from • **should** be responsible of maintaining and managing the DSA repository • **could** use an open and standardised policy description language or schema, which ○ the data policy **must** include details about**:** ▪ all the parties participating in CTI sharing ▪ all the parties participating in CTI processing

	▪ rules concerning authorisation and access to the CTI data The SME: • **must** be able to select a data sharing agreement from a set of pre-defined policies provided by the DSA Service
Flow of events: ***Normal flow***	1. DSA Service create a set of data sharing agreements 2. The DSAs are stored in the C3ISP DSA repository and are made available to the SME 3. SME uses the DSA web app via the C3ISP Gateway to choose a data sharing policy from the DSA repository that is suitable for it 4. SME uses the C3ISP Gateway to notify the DSA Service about the policy it has chosen as the DSA
Pre-condition	• The C3ISP Gateway should have access to the DSA Service • SMEs are registered with C3ISP Framework to use the DSA Service • C3ISP Framework and the C3ISP Gateway should be using the same format, template or schema for their data sharing policies
Post-condition	• A Data Sharing Agreement exists between the SMEs and the C3ISP Framework • C3ISP Framework has started enforcing the DSA on the SMEs CTI data • SMEs can start consuming the C3ISP Framework

Appendix 2. XML Schema of the Security Services of the MSS

The XML schema used by the SOAP API responses of the individual security services of the MSS, as well as their filters' datatypes, are given below:

- **Firewall CTI data schema:**

The Deep Security firewall is a bidirectional, stateful firewall that is responsible for making sure that packets originating from unauthorized sources do not reach the applications on its host. The following table shows all the elements that can profile a firewall event:

	Element Name	Element Type	Description
FirewallEventTransport	firewallEventID	xsd:long	FirewallEventTransport ID
	action	xsd:string	Resulting action of the triggered event, e.g. log or deny
	data	xsd:base64Binary	Any capture packet data in Base64 encoded format
	dataFlags	xsd:int	A binary indication xor'd flags from the network engine that are used to indicate conditions of the engine and data capture, e.g. TRUNCATED 0x01, OVERFLOW 0x02, SUPRESSED 0x04, HAVE DATA 0x08, REF DATA 0x01
	dataIndex	xsd:int	Index of the final character in the data that triggered the event
	destinationIP	xsd:string	Destination IP address
	destinationMAC	xsd:string	Destination MAC address
	destinationPort	xsd:string	Destination Port
	direction	xsd:string	Direction of the event, e.g. incoming or outgoing
	driverTime	xsd:long	Epoch time the Agent driver recorded at the time of the event
	endTime	xsd:dateTime	End time of the event if repeated multiple times, e.g. Internet browsers will resend a request multiple times if the connection is dropped and the exact same event would be repeated multiple times.
	eventOrigin	Impl:EnumEventOrigin	Origin of the event, e.g., AGENT, GUESTAGENT, APPLICATION
	flags	xsd:string	Data packet flags, e.g. ACK FIN

	flow	xsd:string	Flow of the packet the log was recorded for in relation to the connection direction, e.g., 0=FORWARD, 1=BACKWARD
	frameType	xsd:string	Connection frame type, e.g., IP, ARP
	hostID	xsd:int	Host ID of the computer where the event was triggered
	hostName	xsd:string	Host name of the computer where the event was triggered
	iface	xsd:string	Name of the physical network interface where the event was triggered
	note	xsd:string	Internal note property that the engine may set for use by the Manager, e.g., Drop_data
	packetSize	xsd:int	Size of the packet which triggered the event
	protocol	xsd:string	Protocol of the connection
	rank	xsd:int	Calculated Rank Value (Computer Asset Value * IPS Filter Ranking)
	reason	xsd:string	Name of the firewall rule which triggered the event
	repeatCount	xsd:int	Repeat count of the event if repeated multiple times.
	sourceIP	xsd:string	Source IP Address
	sourceMAC	xsd:string	Source MAC Address
	sourcePort	xsd:string	Source Port
	startTime	xsd:dateTime	Start time of the event if repeated multiple times.
	status	xsd:int	Error status code which will be 0 if no abnormal conditions were found
	tags	xsd:string	Name of any event tags assigned to this event.

- **Intrusion prevention CTI data schema**

The Intrusion Prevention module protects computers from being exploited by attacks against known and zero-day vulnerability attacks as well as against SQL injections attacks, cross-site scripting attacks, and other web application vulnerabilities. Shields vulnerabilities until code fixes can be completed. It identifies malicious software accessing the network and increases visibility into, or control over, applications accessing the network. Intrusion Prevention prevents attacks by detecting malicious instructions in network traffic and dropping relevant packets. Intrusion Prevention can be used for the following functions:

The following table lists all the element need for describing an intrusion prevention event:

	Element Name	Element Type	Description
DPIEventTransport	DPIEventID	xsd:long	DPIEventTransport ID
	DPIRuleID	xsd:int	DPIRuleTransport ID that triggered this event
	action	xsd:string	Resulting action of the triggered event, e.g. log or deny
	data	xsd:base64Binary	Any capture packet data in Base64 encoded format
	dataFlags	xsd:int	A binary indication xor'd flags from the network engine that are used to indicate conditions of the engine and data capture, e.g. TRUNCATED 0x01, OVERFLOW 0x02, SUPRESSED 0x04, HAVE DATA 0x08, REF DATA 0x01
	dataIndex	xsd:int	Index of the final character in the data that triggered the event
	destinationIP	xsd:string	Destination IP address
	destinationMAC	xsd:string	Destination MAC address
	destinationPort	xsd:string	Destination Port
	direction	xsd:string	Direction of the event, e.g. incoming or outgoing
	driverTime	xsd:long	Epoch time the Agent driver recorded at the time of the event
	endTime	xsd:dateTime	End time of the event if repeated multiple times, e.g. Internet browsers will resend a request multiple times if the connection is dropped and the exact same event would be repeated multiple times.
	eventOrigin	Impl:EnumEventOrigin	Origin of the event, e.g., AGENT, GUESTAGENT, APPLICATION
	flags	xsd:string	Data packet flags, e.g. ACK FIN
	flow	xsd:string	Flow of the packet the log was recorded for in relation to the connection direction, e.g., 0=FORWARD, 1=BACKWARD
	hostID	xsd:int	Host ID of the computer where the event was triggered

	hostName	xsd:string	Host name of the computer where the event was triggered
	iface	xsd:string	Name of the physical network interface where the event was triggered
	note	xsd:string	Internal note property that the engine may set for use by the Manager, e.g., Drop_data
	packetSize	xsd:int	Size of the packet which triggered the event
	protocol	xsd:string	Protocol of the connection
	rank	xsd:int	Calculated Rank Value (Computer Asset Value * IPS Filter Ranking)
	reason	xsd:string	Name of the firewall rule which triggered the event
	repeatCount	xsd:int	Repeat count of the event if repeated multiple times.
	sourceIP	xsd:string	Source IP Address
	sourceMAC	xsd:string	Source MAC Address
	sourcePort	xsd:string	Source Port
	startTime	xsd:dateTime	Start time of the event if repeated multiple times.
	status	xsd:int	Error status code which will be 0 if no abnormal conditions were found
	tags	xsd:string	Name of any event tags assigned to this event.

- **Anti-malware CTI data schema**

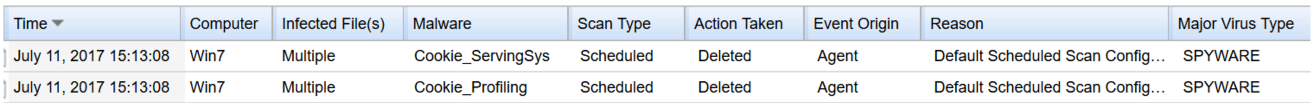
The Anti-Malware module provides both real-time and on-demand protection against file-based threats, including threats commonly referred to as malware, viruses, Trojans, and spyware. To identify threats, Anti-Malware checks files against a comprehensive threat database, portions of which are hosted on servers or kept locally as updatable patterns. Anti-Malware also checks files for certain characteristics, such as compression and known exploit code.

To address threats, Anti-Malware selectively performs actions that contain and remove the threats while minimizing system impact. Anti-Malware can clean, delete, or quarantine malicious files. It can also terminate processes and delete other system objects that are associated with identified threats. The following table lists all elements that are used to profile an anti-malware event:

	Element Name	Element Type	Description
AntiMalwareEventTransport	antiMalwareConfigID	xsd:int	The ID of the Anti-Malware configuration this event corresponds to
	antiMalwareEventID	xsd:long	The ID of the event
	endTime	xsd:dateTime	Endtime of this event if it was repeated
	errorCode	xsd:int	The VSAPI error code indicates the reason of the actions of failure
	hostID	xsd:int	The host ID this event corresponds to
	infectedFilePath	xsd:string	The infected file full path
	infectionSource	xsd:string	The source computer of the infection
	logDate	xsd:dateTime	The time this event occurred
	malwareName	xsd:string	The name of the malware
	malwareType	impl:EnumMalwareType	The type of the malware
	protocol	xsd:int	The protocols: Local Files(0), Network shared folder(1), etc. However, currently Agent only support local files.
	quarantineRecordID	xsd:int	The ID of the quarantined file, if a file was quarantined as a result of this event
	scanResultAction1	xsd:int	The result of the first scan action: represent whether the action is successful (0) or failed (Error Code)
	scanResultAction2	xsd:int	The result of the first scan action: represent whether the action is successful (0) or failed (Error Code)
	scanAction1	xsd:int	The actual first scan action being taken: e.g. Pass (1), Delete (2), Quarantined (3), Clean (4), Deny Access (5)
	scanAction2	xsd:int	The actual second scan action being taken: e.g. Pass (1), Delete (2), Quarantined (3), Clean (4), Deny Access (5)
	scanType	impl:EnumAntiMalwareScanType	Type of scan this event was captured under
	spywareItems	impl:ArrayOfAntiMalwareSpywareItemTransport	An array of spyware items associated with this event

	startTime	xsd:dateTime	Start time of this event if it was repeated multiple times (not currently used)
	tags	xsd:string	Any tags associated with this event
	summaryScanResult	xsd:string	Summary field for the Scan Result: e.g. passed, deleted, quarantined, cleaned, deny access.

The following table shows an example of anti-malware event presented by the MSS:



Time	Computer	Infected File(s)	Malware	Scan Type	Action Taken	Event Origin	Reason	Major Virus Type
July 11, 2017 15:13:08	Win7	Multiple	Cookie_ServingSys	Scheduled	Deleted	Agent	Default Scheduled Scan Config…	SPYWARE
July 11, 2017 15:13:08	Win7	Multiple	Cookie_Profiling	Scheduled	Deleted	Agent	Default Scheduled Scan Config…	SPYWARE

- **Web reputation CTI data schema:**

The Web Reputation module protects against web threats by blocking access to malicious URLs. MSS uses Trend Micro's Web security databases from Smart Protection Network sources to check the reputation of Web sites that users are attempting to access. The Web site's reputation is correlated with the specific Web reputation policy enforced on the computer. Depending on the Web Reputation Security Level being enforced, MSS will either block or allow access to the URL. The following table lists all elements that can be used to profile a web reputation event:

	Element Name	Element Type	Description
WeReputationEventTransport	hostID	xsd:long	The host ID this event corresponds to
	hostName	xsd:string	The name of the host this event corresponds to
	logTime	xsd:dateTime	The time this event occurs
	rank	xsd:int	The rank of the event
	risk	impl:EnumWebReputationEventRisk	The risk level of this event
	tags	xsd:string	Any tags associated with this event
	url	xsd:string	The URL that triggered this event.
	webReputationEventID	Xsd:int	The ID of the event

- **Integrity monitoring CTI data schema:**

Integrity Monitoring allows you to monitor specific areas on a computer for changes. MSS has the ability to monitor installed software, running services, processes, files, directories, listening ports, registry keys, and registry values. It functions by performing a baseline scan of the areas on the computer specified in the assigned rules and then periodically rescanning those areas to look for changes. The MSS ships with predefined Integrity Monitoring Rules and new Integrity Monitoring Rules are provided in Security Updates. Recommendation Scans will recommend Integrity Monitoring Rules for a computer.

The following table lists all the elements for profiling an integrity event:

	Element Name	Element Type	Description
IntegrityEventTransport	integrityEventID	xsd:long	IntegrityEventTransport ID
	integrityRuleID	xsd:int	IntegrityRUleTransport ID which triggered this event
	change	xsd:string	Change applied to the target key, e.g., Created, Updated, Deleted, Renamed
	description	xsd:string	Description of the monitored attributes and what changed
	hostID	xsd:int	Host ID of the computer where the event was triggered
	hostName	xsd:string	Host name of the computer where the event was triggered
	isEntity	impl:EntityTransport	EntityTransport of the monitored entity after the change which triggered the event
	key	xsd:string	Name of file or registry key which the Integrity rule triggered on during a scan (if available)
	logTime	xsd:dateTime	Time the triggered event was logged
	process	xsd:string	Name of process or service which the Integrity rule triggered on during a scan (if available)
	rank	xsd:int	Calculated Rank Value (Computer Asset Value * IPS Filter Ranking)
	reason	xsd:string	Name of the Integrity rule which triggered the event
	severity	impl:EnumIntegrityRuleSeverity	EnumIntegrityRuleSeverity severity level of the triggered event, e.g., CRITICAL, HIGH, MEDIUM, LOW
	tags	xsd:string	Name of any event tags assigned to this event.
	type	xsd:string	Key type, e.g., Directory, File, Group, Installed Software, Service, User

	user	xsd:string	Name of the user which the Integrity rule triggered on during a scan (if available)
	wasEntity	impl:EntityTransport	EntityTransport of the monitored entity before the change which triggered the event

- **Log inspection CTI data structure:**

The OSSEC Log Inspection Engine is integrated into MSS and gives you the ability to inspect the logs and events generated by the operating systems and applications running on the computers.

	Element Name	Element Type	Description
LogInspectionEventTransport	logInspectionEventID	xsd:long	LogInspectionEventTransport ID
	logInspectionRuleID	xsd:string	LogInspectionRuleTransport ID
	action	xsd:string	Resulting action of the triggered event
	command	xsd:string	Command associates to the event
	data	xsd:string	Source log file data type, e.g., Windows Events = Crypt32, Security, Application
	description	xsd:string	Name of the triggered LogInspectionRuleTransport sub-rule
	destinationIP	xsd:string	Destination IP address if available
	destinationUser	xsd:string	DestinationUser if available
	destinationPort	xsd:string	Destination Port if available
	fullEvent	xsd:string	Copy of the triggered full log entry
	groups	xsd:string	Groups of the LogInspectionRuleTransport triggered sub-rule
	hostID	xsd:int	Host ID of the computer where the event was triggered
	hostName	xsd:string	Host name of the computer where the event was triggered
	location	xsd:string	Location of the inspected log file
	logTime	xsd:dateTime	Time of the triggered event
	message	xsd:string	Message of the event

	programName	xsd:string	Name of the monitored log file application
	rank	xsd:int	Calculated Rank Value (Computer Asset Value * IPS Filter Ranking)
	reason	xsd:string	Name of the Inspection rule that triggered the event
	ruleID	xsd:int	LogInspectionRuleTransport sub-rule ID as defined in the rule syntax
	severity	xsd:string	Severity of the triggered sub-rule, e.g., Lowest = 1, Critical = 15
	sourceHostName	xsd:string	Source hostname if available
	sourceID	xsd:string	Source ID if available
	sourceIP	xsd:string	Source IP Address if available
	sourcePort	xsd:string	Source Port if available
	sourceUser	xsd:string	Source User if available
	status	xsd:int	Error status code which will be 0 if no abnormal conditions were found
	systemName	xsd:string	System name of the computer the event triggered on
	tags	xsd:string	Name of any event tags assigned to this event.
	url	xsd:string	URL attribute of the log event if available

Appendix 3. SOAP Web Service implementations

Programming Language/API	Resources
C#/VB.NET/Managed C++ using .NET Framework	http://msdn2.microsoft.com/en-us/netframework/default.aspx
Java using Apache Axis	http://ws.apache.org/axis/java/index.html
C++ using gSOAP	http://www.cs.fsu.edu/~engelen/soap.html
C++ using Apache Axis	http://ws.apache.org/axis/cpp/index.html
PHP using PEAR	http://pear.php.net/package/SOAP
Ruby using soap4r	http://dev.ctor.org/soap4r
Perl using SOAP:Lite	http://www.soaplite.com/
CORBA using SOAP2CORBA	http://soap2corba.sourceforge.net/
Python using Python Web Services	http://pywebsvcs.sourceforge.net/

Appendix 4. Java and Apache Axis required libraries

API	Version	Resources
axis	1.4	http://axis.apache.org/axis/java/releases.html
axis-ant	0.1	http://axis.apache.org/axis/java/releases.html
commons-discovery	0.2	http://axis.apache.org/axis/java/releases.html
commons-logging	1.0.4	http://axis.apache.org/axis/java/releases.html
jaxrpc	0.1	http://axis.apache.org/axis/java/releases.html
log4j	1.2.8	http://axis.apache.org/axis/java/releases.html
saaj	0.1	http://axis.apache.org/axis/java/releases.html
Wsdl4j	1.5.1	http://axis.apache.org/axis/java/releases.html
mail	Javamail-1.4.7	http://www.oracle.com/technetwork/java/index-138643.html
activation	Jaf-1.1.1	http://www.oracle.com/technetwork/articles/java/index-135046.html