SOC & CSIRT Response to Attacks & Threats based on attack defence graphs Evaluation Systems

D6.1
Initial version of the SOCCRATES Platform Orchestration, Reconfiguration and Front-end

Deliverable type:	Demonstrator
Contributing work packages:	WP6
Due date of deliverable:	30/11/2020
Submission date:	30/11/2020
Dissemination level:	PU

Responsible organisation:	ATOS
Editor:	Ruben Trapero
Revision:	1.0

Abstract	This deliverable details the initial results on the development of the SOCCRATES Orchestrator and Integration Engine and the plans for the development of the Reconfiguration capabilities and the SOCCRATES Web Front-end.
Keywords:	Security Automation, Orchestrator & Integration Engine, Web Front End, Integration

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement no. 833481
Call H2020-SU-ICT-2018 • Innovation Action • Start date: September 1st, 2019

Management summary

This deliverable provides an overview of the initial results of Task 6.1, focused on the development of an Orchestrator & Integration Engine for the SOCCRATES platform as well as the creation of reconfiguration mechanisms, and the development of an integrated web front end. All supported by the integration activities that are also covered by this task.

This deliverable D6.1, is the first report resulting from Task 6.1, which is essentially focused on the initial results related to the SOCCRATES orchestrator, including aspects such as the Security Orchestration context analysis, the technological and design decisions taken and initial proofs of concepts already developed. The document also describes the initial considerations related to the development of the reconfiguration mechanisms, the integrated Web Front End and the overall strategy to follow for the SOCCRATES integration.

With respect to the SOCCRATES orchestrator, the objective is the development of a framework that allows SOCCRATES tools to interoperate in a seamless way, concatenating requests and responses to individual tools according to a logic that depends on the type of incident detected or depending on the component that triggers the orchestrator. At the same time, the SOCCRATES orchestrator keeps track of the status of the incident, indicating the stage where it is, and information related to results obtained by the execution of the SOCCRATES tools attached to the orchestrator.

The results presented in this deliverable describe a study of the current solutions for security orchestration, as well as a study of the requirements defined for the SOCCRATES Orchestrator & Integration Engine. The approach followed by SOCCRATES is the development of a workflow-based orchestrator, that uses a BPMN engine (based on Activiti), that allows to design, deploy and execute the logic associated to the use cases considered. Additionally, the Orchestrator & Integration Engine is combined with an abstraction layer based on the Cortex framework, that unifies the way that the workflow engine uses to interact with every tool individually. To this end, an initial proof of concept has been created that demonstrates the feasibility of the approach selected.

The second pillar of T6.1 corresponds to the definition of mechanisms for the reconfiguration of the infrastructure, representing the response to the incident that has been processed by the Orchestrator & Integration Engine. The initial results are based on the use of Cortex responders that trigger external components that carry out the response, such as opening tickets in an incident management solution or invoke openC2 commands for enforcing response activities on systems and cyber security defences.

The third pillar of T6.1 corresponds to the creation of an integrated Web Front-End for the visualization and management of the SOCCRATES operation. This activity is currently at an early stage. Therefore, this document presents the initial insights which are mostly focused on requirements and challenges to address.

Finally, the integration approach is presented in this deliverable. SOCCRATES deploys an incremental integration methodology, with components integrated remotely, progressively moved towards a full on-premises deployment. To this end, the usage of Cortex as an abstraction layer for the communications between tools through the orchestration allows to carry out this type of integration. Additionally, a testbed is created that simulates a real network, which is available for initial integration activities or unit tests.

Contributor(s)	Ruben Trapero (ATOS) Martin Eian (MNM) Frank Fransen (TNO) Ewa Piatkowska (AIT) Christophe Kiennert (IMT) Maciej Kosz (VTF) Widel Wojciech (KTH) Erick Ringdahl (FRS)

Reviewer(s)	Paul Smith (AIT)

TABLE OF CONTENTS

TABLE OF CONTENTS 4
1 INTRODUCTION – RATIONALE OF THIS DOCUMENT 7
1.1 The SOCCRATES project 7
1.2 This deliverable 8
1.3 Structure of this deliverable 8
2 ORCHESTRATOR & INTEGRATION ENGINE 9
2.1 Overview 9
2.2 State of the art on Security orchestration and related technologies 9
2.3 Orchestrator & Integration Engine in SOCCRATES 12
2.3.1 Activiti 13
2.3.2 Cortex 18
2.4 Initial deployment 26
2.5 Proof of concept for the SOCCRATES orchestration 27
2.6 Use case activiti workflows 30
2.6.1 Workflow for UC1 30
2.6.2 Workflow for UC2, UC3, UC4 and UC5 32
3 RECONFIGURATION 33
4 WEB FRONT END 37
5 INITIAL INTEGRATION PLAN 40
5.1 Integration methodology 40
5.2 Individual components functions 41
5.2.1 Orchestrator triggering 41
5.2.1.1 SIEM solution triggering the Orchestrator 41
5.2.1.2 Threat Intelligence Platform triggering the Orchestrator 44
5.2.1.3 Infrastructure Modelling Component triggering the Orchestrator 44
5.2.2 SOCCRATES Analyzers 45
5.2.2.1 Infrastructure Modelling Component 45
5.2.2.2 Threat Intelligence Platform 46
5.2.2.3 ADG Analyzer 47
5.2.2.4 CoA Generator 48
5.2.2.5 Impact Analyzer 51
5.2.2.6 Response Planner 53
5.2.3 SOCCRATES Responders 54
5.2.3.1 AAD responder 55
5.3 Testbed 55
6 CONCLUSIONS 58
6.1 Next steps 58
7 REFERENCES 60
8 ABBREVIATIONS 61

TABLE OF FIGURES

Figure 1-1. The SOCCRATES platform 7
Figure 2-1. Incident management stages 10
Figure 2-2. Architecture of the SOCCRATES Orchestrator 13
Figure 2-3. Interactions between Activiti components 14
Figure 2-4. GUI for activiti-app 14
Figure 2-5. GUI of activiti admin 15
Figure 2-6. Test of activiti-rest configuration at activiti-admin app 15
Figure 2-7. Activiti GUI for creating workflows 16
Figure 2-8. Docker containers for Cortex 19
Figure 2-9. Default Cortex analyzers and responders 19
Figure 2-10. Main window of the Cortex control panel 19
Figure 2-11. Panel for configuring Cortex users 20
Figure 2-12. Cortex panel for enabling analyzers 20
Figure 2-13. Cortex panel for configuring an analyzer 21
Figure 2-14. Descriptor of a custom Cortex analyzer 22
Figure 2-15. Code for a custom Cortex analyzer 22
Figure 2-16. Cortex panel with a new analyzer 23
Figure 2-17. Configuration required to invoke two different analyzers from the control panel 23
Figure 2-18. Request to Cortex Analyzer using REST-API 25
Figure 2-19. Result of a request to a Cortex analyzer 25
Figure 2-20. Orchestrator testbed deployed at Atos 26
Figure 2-21. Activiti workflow for the proof of concept 27
Figure 2-22. Interaction between components in the proof of concept 27
Figure 2-23. Inputs and output of the proof of concept Web Service 28
Figure 2-24. Request to trigger the workflow of the proof of concept 28
Figure 2-25. Java class executed with the Service Task of the proof of concept 29
Figure 2-26. Excerpt of code that invokes the Cortex Analyzer 29
Figure 2-27. Result of execution of the workflow of the proof of concept 30
Figure 2-28. Initial incident analysis for UC1 31
Figure 2-29. Deeper incident analysis for UC1 31
Figure 2-30. Determine response in UC1 32
Figure 2-31. Workflow diagram for UC2, UC3, UC4 and UC5 32
Figure 3-1. Link between the orchestrator workflow and the reconfiguration 33
Figure 3-2. Subprocess workflow for the reconfiguration task 34
Figure 3-3. Process for triggering reconfiguration actions with Cortex 35
Figure 4-1. Concept for the Web Front-End 37
Figure 4-2. Mock-up of the Web Front End 38
Figure 4-3. Approach for unifying identity management technologies at SOCCRATES tools GUIs 38
Figure 5-4. Network topology of the testbed 56
Figure 5-5. Zebra configuration for the Router1 of the testbed 56
Figure 5-6. OSPF service configuration for Router 1 57
Figure 5-7. F-Secure's RADAR app running in the testbed 57

TABLE OF TABLES

Table 2-1. Comparison of different security orchestration solutions 10
Table 2-2. Tools interacting with the SOCCRATES Orchestrator 17
Table 3-1. CoA - OpenC2 example for containment action 36
Table 3-2. CoA - OpenC2 example for patching software on host 36
Table 5-1. Details of SIEMs triggering the orchestrator 41
Table 5-2. Details of TIP triggering the Orchestrator 44
Table 5-3. Details of IMC triggering the orchestrator 44
Table 5-4. Details of the Cortex analyzer for the IMC 45
Table 5-5. Details of the Cortex analyzer for the TIP 46
Table 5-6. Details of the Cortex analyzer for the ADG Analyzer 47
Table 5-7. Details of the Cortex analyzer for the CoA Generator 48
Table 5-8. Details of the Cortex analyzer for the Impact Analyzer 51
Table 5-9. Details of the Cortex analyzer for the Response Planner 53
Table 5-10. Details of the Cortex responder for the AAD 55

1 Introduction – Rationale of this document

This section introduces the SOCCRATES project and defines the goals of this deliverable.

1.1 The SOCCRATES project

SOCCRATES (SOC & CSIRT Response to Attacks & Threats based on attack defence graphs Evaluation Systems) is an EU funded project under the Horizon2020 programme that has the following main challenge:

How can SOC and CSIRT operations effectively improve their capability in detecting and managing response to complex cyber-attacks and emerging threats, in complex and continuously evolving ICT infrastructures while there is a shortage of qualified cybersecurity talent?

The main objective of SOCCRATES is to develop and implement a security automation and decision support platform (‘**the SOCCRATES platform’)** that will significantly improve an organisation’s capability (usually implemented by a SOC and/or CSIRT) to quickly and effectively detect and respond to new cyber threats and ongoing attacks.

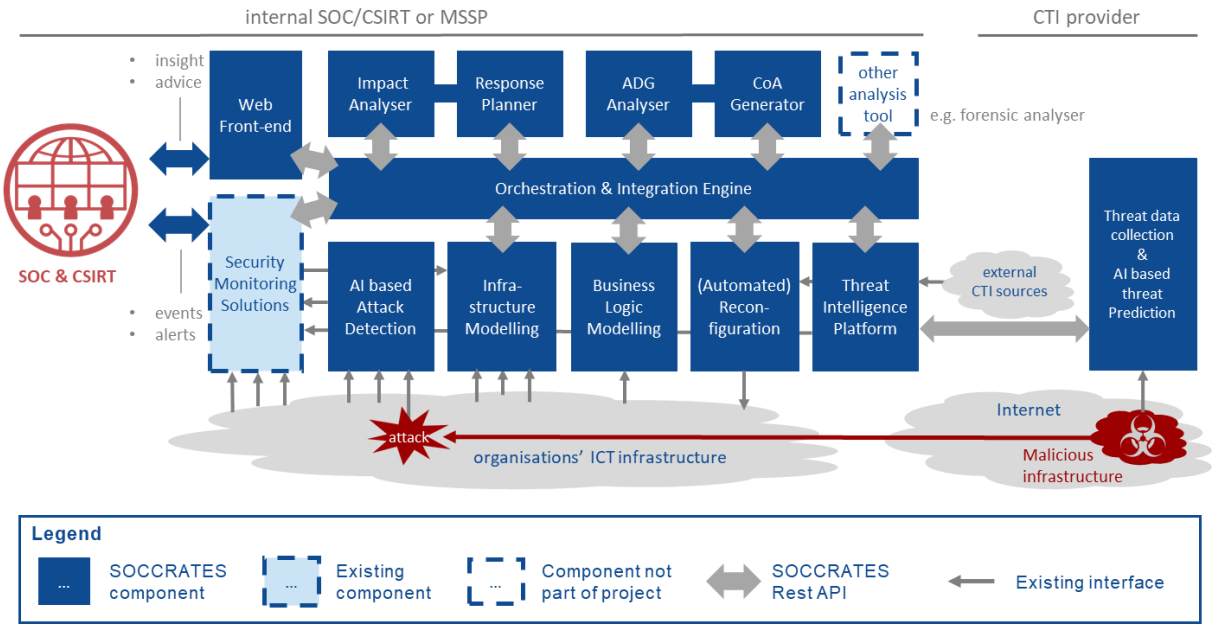

Figure 1-1. The SOCCRATES platform

The SOCCRATES platform (see Figure 1-1) consists of an orchestrating function and a set of innovative components for automated infrastructure modelling, attack detection, cyber threat intelligence utilization, threat trend prediction, and automated analysis using attack defence graphs and business impact modelling to aid human analysis and decision making on response actions, and enable the execution of defensive actions at machine-speed.

SOCCRATES has the following concrete project objectives:

1. Deliver the SOCCRATES platform consisting of an orchestration function and a unique integration of innovative background solutions that seamlessly work together.
2. Show that the SOCCRATES platform can improve SOC operations by evaluating the SOCCRATES platform in two diverse real-life pilot environments.
3. Examine and illustrate the benefits of automation for selected SOC activities to help manage the cyber security skills gap in organizations.
4. Prepare for successful exploitation by the SOCCRATES partners of the individual innovated components and the integrated SOCCRATES platform in commercial products that are offered to the market and are available for the European (business) community.

Please visit www.soccrates.eu for more information on the SOCCRATES project.

1.2 This deliverable

This deliverable is the first report of WP6, framed mostly on results of Task 6.1 (Orchestration, reconfiguration and Web Front-End of the SOCCRATES Platform). The goal of this document is to describe the development of three components that are transversal to the rest of the components of the SOCCRATES platform, this is, they have to be integrated with them (such as the Web Front-End) and interact with them (such as Orchestrator). Additionally, this deliverable includes the initial integration plan that will be followed in SOCCRATES, which is directly related to the development of the Orchestrator & Integration Engine.

1.3 Structure of this deliverable

This deliverable is structured, as follows. Section 2 describes the initial results related to the development of the Orchestrator & Integration Engine. Section 3 and Section 4 focus on the approach to follow for the development of reconfiguration mechanisms and Web Front End, respectively. Section 5 describes the integration plan, giving details about the integration between SOCCRATES tools and the Orchestrator & Integration Engine. A testbed is introduced that has been created for the initial testing of individual components that are provided by some of the partners of the SOCCRATES consortium. Finally, Section 6 concludes the document.

2 Orchestrator & Integration Engine

2.1 Overview

The Orchestrator & Integration Engine is at the core of the SOCCRATES platform. It is in charge of correctly managing the interoperation between the rest of the components of the platform. In SOCCRATES the Orchestrator & Integration Engine is a component to be developed from scratch departing from existing technologies but combined to provide the Orchestration capability. It will be adapted to the needs of the SOCCRATES requirements but generic enough to be used in different domains. This section provides a comprehensive analysis on the current state of the art on security orchestration solutions and technologies of interest for its usage in the SOCCRATES orchestrator. Then, the requirements to cover are evaluated and, finally, the complete details about the SOCCRATES Orchestrator are given.

2.2 State of the art on Security orchestration and related technologies

The concept of Security Orchestrators has recently emerged as the integration of three different cybersecurity related technologies that have traditionally worked independently:

- SIRP (Security Incident Response Platform), have the knowledge related to incidents and are in charge of managing incidents and cases, triggering workflows and responses. Some of them can be found here[1].
- TIPs (Threat Intelligence Platforms), in charge of managing information about threats, such as aggregation, visualization or distribution. Some of them can be found here[2].
- SOA (Security Orchestration and Automation), responsible for managing the workflows related to the automation of security related processes. Some of them can be found here[3].

In general, workflows at the SOA are created with the information obtained from SIRPs. On the other side, SOAs workflows run in the context of TIPs, which use them for detecting incidents and triggering alerts. These alerts are used by SIRPs for designing responses for those incidents, which at the same time are used to fine tune, modify or create new workflows at the SOA.

The cybersecurity community has recently realized the need for optimizing processes by integrating these three technologies, plus additional ones that allow to enrich the cybersecurity management. Such is the case of Endpoint Detection and Response (EDR), Network Detection or email security to name just a few. Also, traditional SIEMs are being enhanced with Security Orchestration capabilities, with the aim of providing a holistic solution that cover all aspects of the cybersecurity management.

It is difficult to determine exactly the characteristics that a Security Orchestrator must have, especially due to the amount of possible technologies that can be integrated with a Security Orchestrator. However, a few common criteria can be used at least to initially compare solutions:

- *Interoperability with other products*. This is the number and type of external tools that can be integrated in the security orchestration. This include incident detectors, business impact analyzers, vulnerability scanners, threat intelligence platforms, response planners, etc., either proprietary or open source. This is probably the most important characteristic of a security orchestrator. The higher the number and variety of products the better for a security

[1] https://www.trustradius.com/incident-response
[2] https://www.esecurityplanet.com/products/top-threat-intelligence-companies.html
[3] https://www.trustradius.com/security-orchestration-automation-and-response-soar

orchestrator to be adapted to the characteristics and requirements of the infrastructure where it is operating.

- *Security Operations covered*. This is the phases of the incident management stages (Figure 2-1) that a security orchestrator can cover: detection, analysis, containment, eradication, recovery. For example, some orchestration would provide just incident detection while other would also cover the response to the incident.
- *Included Technologies*. This denotes the capabilities of the security orchestration. To this end it is worth noticing their novelty (for instance, whether machine learning technologies are included) and flexibility of adding additional technologies to incorporate additional features.
- *Workflow creation*. This is related to the capability of create workflows within the security orchestrator, as easy as possible, preferably without the need of coding and flexible enough to be adapted to changing conditions of the infrastructure.
- *Involvement of analysts*. This is related to the capability of involving human analysts in some of the decisions to be taken during the incident management. This includes the capability of notifying specific personnel, the usage of roles, etc.
- *Deployment type*. This is the way the orchestrator works in the infrastructure: installed on premises, on cloud or hybrid.

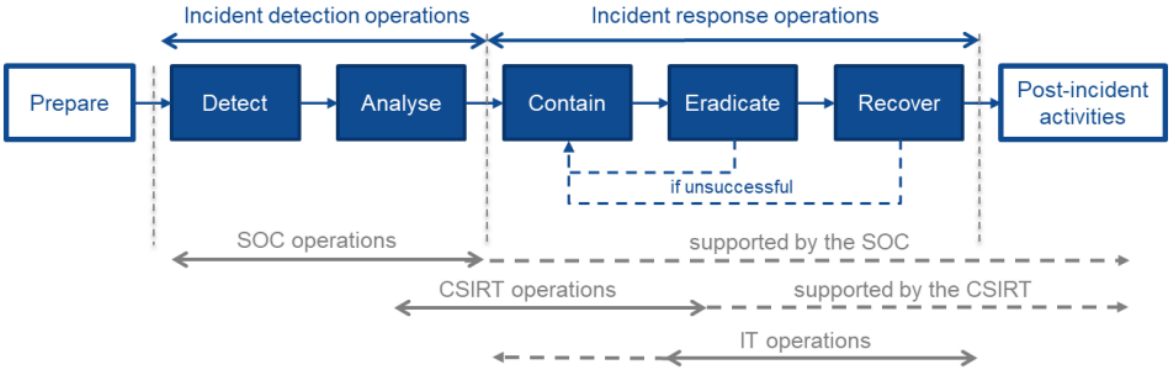

Figure 2-1. Incident management stages

Using the list of criteria describe above, Table 2-1 compares several security orchestration solutions described by [Gartner2020]. All those solutions are commercial solutions and provide different capabilities. With the information publicly available, there is no solution that fully covers all aspects in terms of openness, flexibility, interoperability or ease of use in terms of workflow creation.

Table 2-1. Comparison of different security orchestration solutions

	Interoperability	Security Operations	Technologies	Workflow creation	Analysts involvement	Deployment type
Anomali[4]	With commercial sources, CERTs, ISACs, websites	Detection, analysis	Incident detection based on ML	yes	Not publicly specified	Not publicly specified
Cyware[5]	Connectors to DevOps tools	Threat Intelligence,	Incident detection	yes	yes	Cloud and on-premise

[4] https://www.anomali.com/

[5] https://cyware.com/

		Response, automation	based on ML			
D3 Security[6]	+300 connectors	TIP capabilities, response, analysis	Not publicly specified	Yes, codeless	Yes, role based	Cloud and on-premise
DFLabs[7]	Noncyber incidents (OT, SCADA, IoT, physical security)	Incident analysis, containment, notification	Incident detection based on ML	Not publicly specified	Not publicly specified	Not publicly specified
FireEye[8]	150 third party tools	Detection, response, remediation	Not publicly specified	Not publicly specified	Not publicly specified	Not publicly specified
Honeycomb[9]	Not publicly specified	Detection, response, automation	Incident detection based on ML	Not publicly specified	Not publicly specified	Cloud, on-premise, hybrid
LogicHub[10]	Integration with communication tools, slack, open source and commercial threat intelligence	Automation, threat analysis, detection, automated responses	Not publicly specified	Not publicly specified	Not publicly specified	Cloud and on premises
Cortex XSOAR[11]	Includes a marketplace of use cases that integrate third party tools	Automation, response, collaborative incident investigation	Remote management assignment based on ML	Not publicly specified	Yes	Cloud, on-premise, hybrid
Rapid7[12]	+300 plugins	Automation, simple responses (patching)	Not publicly specified	Works with workflows templates and wizards. Integrates Slack and Microsoft Teams for creating workflows	Not publicly specified	Cloud

[6] https://d3security.com/
[7] https://www.dflabs.com/
[8] https://www.fireeye.com/
[9] http://honeycomb.co.uk/
[10] https://www.logichub.com/
[11] https://www.paloaltonetworks.com/
[12] https://www.rapid7.com/

ServiceNow[13]	Not publicly specified	Incident and vulnerability response, analysis, reporting	ML, for automated response, and creation of workflows	Not publicly specified	Not publicly specified	Cloud
Siemplify[14]	Not publicly specified	Incident analysis	Incident detection based on ML	Workflow marketplace for user cases	Yes, supported by ML for assigning analysts	Cloud
Splunk[15]	+319 products, +2048 actions automatic	Automation, incident management, collaboration, reporting	ML for supporting analysts	Not publicly specified	Yes, analysts collaboration	On-premises
Swimlane[16]	+1600 APIs	Automation, incident management, collaboration	Not publicly specified	Yes, drag-and-drop based	Not publicly specified	Not publicly specified
ThreatConnect[17]	Not publicly specified	Automation, analysis	Not publicly specified	Template based workflows	Not publicly specified	Cloud and on-premises

Not having a solution that completely covers all these aspects SOCCRATES will implement its own Orchestrator & Integration Engine, covering these aspects and combining different technologies. To this end, there are two main aspects to be covered by the Orchestrator & Integration Engine: workflow execution and capability to interconnect to external tools.

Deliverable D2.2 (Section 11) [D22] provides a description of possible open-source solutions to be incorporated in the SOCCRATES orchestrator, comparing them and justifying the choice. The following sections describe in detail the status of the SOCCRATES orchestrator, early implementation results, and an initial proof of concept that validates the technology used.

2.3 Orchestrator & Integration Engine in SOCCRATES

The main objective of the Orchestrator & Integration Engine is to act as broker between the different tools involved in processing a security incident. As a part of managing the exchange of information between these tools, the Orchestrator & Integration Engine also includes logic that allows it to know what tool to invoke depending on the result obtained from the execution of other tools that are part of the orchestration. Figure 2-2 represents the architecture of the SOCCRATES Orchestrator & Integration Engine.

[13] https://www.servicenow.com/
[14] https://www.siemplify.co/
[15] https://www.splunk.com/
[16] https://swimlane.com/
[17] https://threatconnect.com/

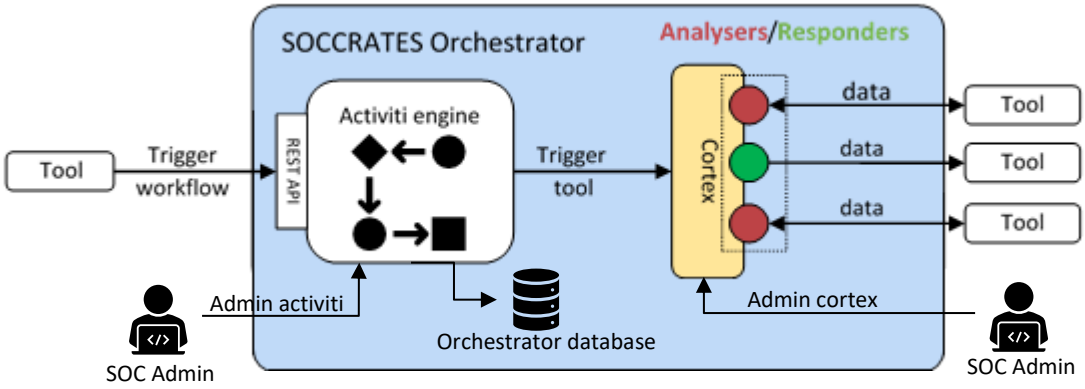

Figure 2-2. Architecture of the SOCCRATES Orchestrator

It relies on the combination of two main technologies:

- Activiti[18], which is a workflow engine and provides the logic that rules the execution of the Orchestrator.
- Cortex[19], which provides a common interface to the orchestrator for invoking external partners tools through connectors called analyzers and responders.

The following subsections provide details about these two technologies. Additionally, a dedicated database for the Orchestrator provides persistent storage capabilities for storing relevant information about the incidents processed or information that is required to be sent to some tool when requesting an action as part of the execution of the workflow.

2.3.1 Activiti

Activiti is the workflow engine selected in SOCCRATES for the core of the Orchestrator & Integration Engine. Activiti is an open source platform that offers a light (Business Process Model and Notation) BPMN engine, licensed under Apache License 2.0 and created by Alfresco Software. BPMN is a well-known standard for defining business processes using a common notation, which allows the automation of several tasks such as the creation of new processes, or their automatic execution. Activiti implements the BPMN 2.0 standard and it can be instantiated either as an Apache Tomcat Web application or embedded as a Spring boot app.

Activiti is composed of three main parts:

- Activiti-app. This is the core of the platform. It includes the execution engine and can be configured to use different types of database for the storage of persistent data. For the version that runs on Apache Tomcat it includes a graphical user interface for the creation of workflows, trigger execution of workflows, manage interactive forms, decision tables, etc.
- Activiti-admin. This is a supporting application that keeps track and controls the workflows deployed at a given time. Through the activiti-admin app, users can deploy and withdraw workflows, check the version of the workflows deployed, active workflows, etc.
- Activiti-rest is the app that manages the REST-API provided by Activiti, which allows to interact with the workflow engine in a remote way. This includes triggering workflows, check execution status, manage users, etc.

Figure 2-3 represents the relationship between these three components of Activiti.

[18] https://www.activiti.org/

[19] https://github.com/TheHive-Project/Cortex

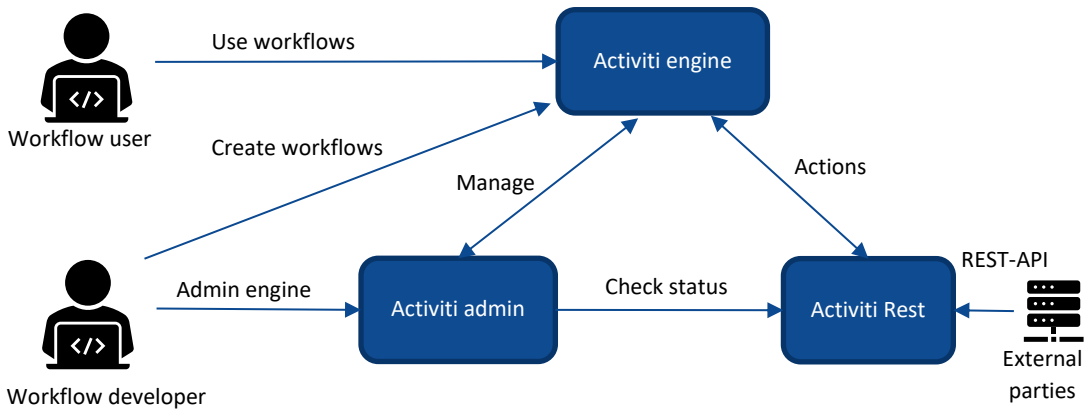

Figure 2-3. Interactions between Activiti components

The initial work carried out in SOCCRATES uses the Apache Tomcat version of Activiti. Therefore, all the descriptions included in this document are based on that type of installation.

Activiti installation and configuration

As a Tomcat Web Application, Activiti is composed of three war files, one for each of the three components mentioned above. The installation is limited to copying and pasting the war files to the webapps folder of tomcat. For the current deployment in SOCCRATES, Apache Tomcat version 8.5.55 has been used and Activiti 6. Once Tomcat is restarted, the apps are automatically installed:

- Activiti-app web front-end (Figure 2-4) will be accessible through http://<serverIP>:8080/acvititi-app

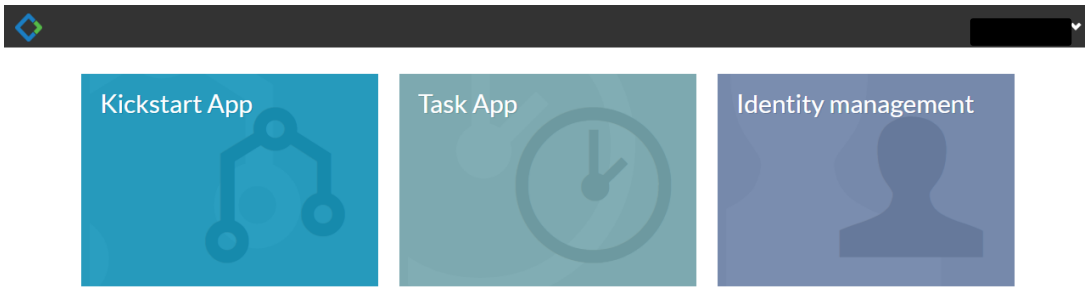

Figure 2-4. GUI for activiti-app

- Activiti-admin web front-end (Figure 2-5) will be accessible through http://<serverIP>:8080/activiti-admin

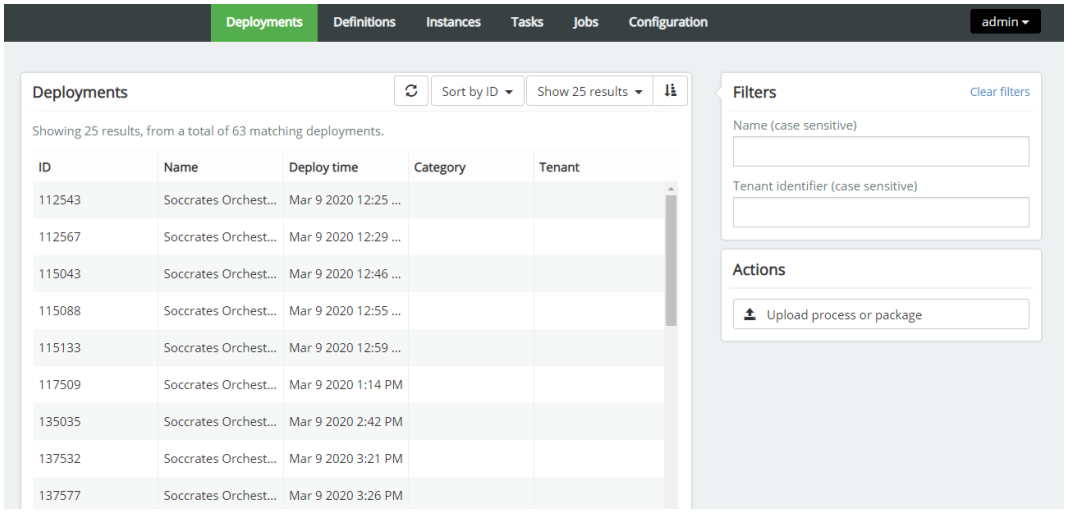

Figure 2-5. GUI of activiti admin

- Activiti-rest. It can be tested if activiti-rest is working properly by using activiti-admin as shown in Figure 2-6, configuring the access to the REST-API server with the following data:
  - Server address:
  - Server port: the port where the engine is running (in this case, 8080)
  - Context root: activiti-rest
  - REST root: service
  - Username: the username at the activiti app engine
  - Password: the username at the activiti app engine

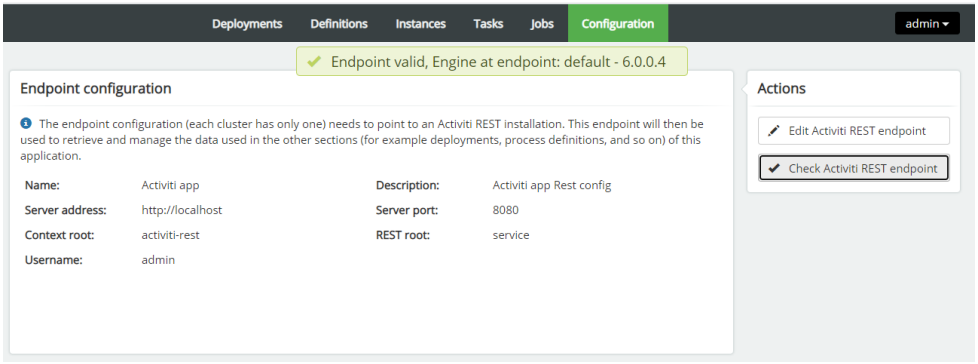

Figure 2-6. Test of activiti-rest configuration at activiti-admin app

By default, the actitivi-app includes a non-persistent database for testing. Activiti includes the capability to use external databases for the storage of persistent data, such as workflow descriptions or users. A MySQL-based database has been chosen for the SOCCRATES Orchestrator. To support this, a few configurations are required:

- Install the MySQL database and configure the required users
- Import Activiti tables. For this it is required to import the table schemas available at the database/ folder of the Activiti installation package.
- Download a Java driver for connecting to the database. For the case of MySQL, the driver to download is mysql-connector-java-8.0.18.jar. This file needs to be included in the /lib folder of Tomcat.
- Edit the activiti-app.properties located at /webapps/activiti-app/WEB-INF/classes/META-INF/activiti-app, modifying it to include the following data:

```
datasource.driver=com.mysql.jdbc.Driver
datasource.url=jdbc:mysql://127.0.0.1:3306/activiti?characterEncoding=UTF-8
datasource.username=<mysql_user>
datasource.password=<mysql_password>
hibernate.dialect=org.hibernate.dialect.MySQLInnoDBDialect
```

- For activiti-admin to use the mySQL database the file activiti-admin.properties located in /webapps/activiti-admin/WEB-INF/classes/META-INF/activiti-admin needs to be modified to add the following content:

```
datasource.driver=com.mysql.jdbc.Driver
datasource.url=jdbc:mysql://127.0.0.1:3306/activiti?characterEncoding=UTF-8
datasource.username=<mysql_user>
datasource.password=<mysql_password>
hibernate.dialect=org.hibernate.dialect.MySQLInnoDBDialect
```

- For activiti-rest, the file db.properties located in /webapps/activiti-rest/WEB-INF/classes needs to be updated with the following content:

```
db=mysql
jdbc.driver=com.mysql.jdbc.Driver
jdbc.url=jdbc:mysql://localhost:3306/activiti?autoReconnect=true
jdbc.username=<usuario>
jdbc.password=<password>
```

Once Tomcat is restarted the three Activiti apps will be attached to the persistent database.

First steps with Activiti

Activiti allows to execute workflows defined using the BPMN 2.0 standard. This means that any workflow defined with the BPMN 2.0 notation can be loaded and executed by the Activiti engine. BPMN is integrated in a wide variety of IDEs that include graphical user interfaces to create workflows. One of them is the one integrated in Eclipse, which allows not just to graphically create workflows compatible with Activiti but also to include the rest of the components (such as java classes) required.
A GUI for visually creating workflows is also included in the activiti-app, as depicted in Figure 2-7.

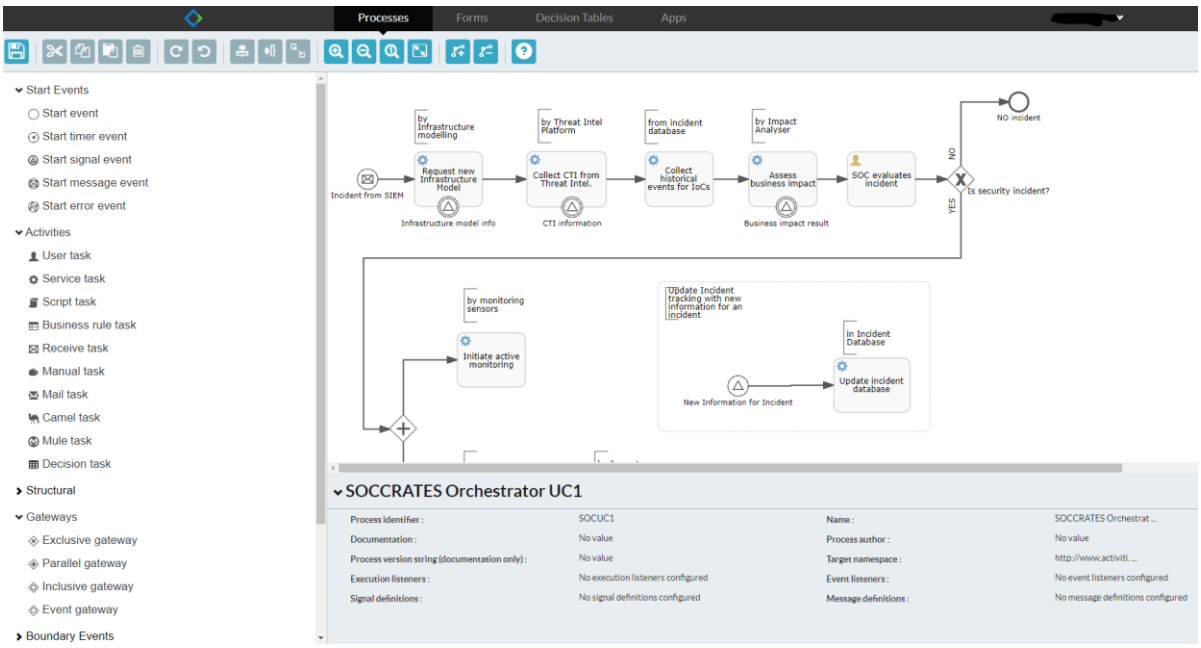

Figure 2-7. Activiti GUI for creating workflows

A workflow is composed of an execution starting event, several tasks, execution branches and one or more execution end events:

- Execution starting event. This is where the workflow starts executing. There are several types of Starting events; the type used depends on how the workflow is triggered. For example, there can be workflows started at a specific time (by using the Start timer starting event), or workflows that are directly started from the activiti-app GUI. Workflows can also be triggered by using the Activiti REST-API, which allows external third-parties to initiate the execution of a workflow. This is especially useful for SOCCRATES, where workflows can be triggered, for example, by Security Information and Event Management (SIEMs) when an incident is detected.
- Tasks represent the intermediate activities that are combined in a workflow. There are different types of tasks available. The most relevant are:
    - User task. Activiti allows tasks to be completed by users that are registered with the platform. This type of task allows to specify an activity that must be completed by a concrete user or by a group of users (for example, by specifying a role).
    - Script task. This task executes a script that is directly embedded in the BPMN workflow. It is convenient for the execution of simple activities that can be directly defined scripts that follows the JSR233[20] specification.
    - Service task. This is probably the most important task in the context of SOCCRATES. This task triggers the execution of a Java class. The Activiti specification provides with a list of Java interfaces that allows to interact with the workflow and its execution environment (variables, users, signals, events, etc). This is especially useful for the SOCCRATES orchestrator, as it allows to interact with the different tools, provide the required logic, etc.
    - Mail task. This task allows to send emails. This is useful in SOCCRATES for the notification about the progress of the incidents or other important issues during the execution of the workflow.
- Gateways. These elements allow to control the flow of the execution. There are exclusive gateways that, depending on a condition, the execution of the workflow can go in one direction or another, or parallels gateways that allows the execution of two branches of the workflow in parallel.

Activiti in SOCCRATES

In SOCCRATES, these elements are used for the development of the SOCCRATES Orchestrator. The tools triggering the Activiti workflows and the ones interacting with the Orchestrator are listed in the following table. Tools interacting with the Orchestrator do it through service tasks that interact with the corresponding Cortex analyzers for every tool (see Section 2.4.2).

Table 2-2. Tools interacting with the SOCCRATES Orchestrator

Tools triggering Activiti workflows	Tools interacting with the Orchestrator
Pilot SIEMs, after a security alert is detected (UC1 [D2.1])	Infrastructure Modelling
Threat Intelligence Platform, when a new threat is noticed (UC2 [D2.1])	Threat Intelligence Platform

[20] List of script languages compatible with the JSR233 specification: https://en.wikipedia.org/wiki/List_of_JVM_languages

Infrastructure Modelling, when new vulnerabilities are found (UC3 [D2.1])	Course of Action
Infrastructure Modelling, when a configuration change is detected (UC4 [D2.1])	Impact Analyzer
Infrastructure Modelling, when a new system is detected in the infrastructure (UC5 [D2.1])	Response Planner

An initial draft of the Activiti workflows that correspond to SOCCRATES is described in Section 2.7. Additionally, an initial working workflow, which combines the key components of the Activiti engine, is described in Section 2.6.

2.3.2 Cortex

Cortex is an open source project created by The Hive Project[21]. It was created as a way to trigger the analysis of observables (IP, URLs, files, email addressed, etc) by different tools without the need to interact with them individually. Instead, Cortex acts as a broker that interacts with those tools, providing a common interface for any consumer entity that needs to interact with them. It allows to abstract consumer entities from the need to deal with complex APIs from different tools, that might use different technologies. Instead, Cortex provides a REST-API that can be used to interact with Cortex and invoke any of the tools that are interconnected with it.

Cortex allows to interact with external tools through two types of connectors: Analyzers and Responders:

- Analyzers allows to invoke external tools by requesting some analysis or action and providing some data in the request. In general, these analyzers respond in a format that depends on the type of tool. Examples of analyzers include requesting an antivirus scan for a file or the evaluation of risk for a given incident.
- Responders allow to trigger some action in an external tool, without the need of receiving any observable in the response. This connector is useful to trigger some response for an incident, such as sending emails to supervisors with relevant information, opening tickets in some response platform or trigger the enforcement of counter measures.

The Cortex project already provides a wide set of predefined Cortex Analyzers and Responders[22], both free and license-based. In SOCCRATES, the tools interacting with the SOCCRATES Orchestrator will do it through Cortex analyzers that will be developed by the tool vendors.

Cortex installation and configuration

Cortex provides several ways to be installed. Individual installation packages can be downloaded in DEB and RPM format. It can also be installed directly from binaries or compiled from the source files. It is also provided as Docker images, which is the way that it has been used in SOCCRATES. The dockerfile available for its installation instantiates two containers: an Elasticsearch engine, for persistent storage, and the Cortex container itself. Figure 2-8 represents the two containers required for running Cortex.

[21] https://github.com/TheHive-Project/Cortex

[22] https://github.com/TheHive-Project/CortexDocs/blob/master/analyzer_requirements.md

```
root@soccrates:/home/xlsiem# docker container list
CONTAINER ID        IMAGE                                                 COMMAND                  CREATED             STATUS
   PORTS                              NAMES
f7edecb9c871        thehiveproject/cortex:latest                          "/opt/cortex/entrypo…"   6 months ago        Up 3 weeks
   9000/tcp, 0.0.0.0:9001->9001/tcp   cortex_cortex_1
6914c7c2713b        docker.elastic.co/elasticsearch/elasticsearch:5.6.16  "/bin/bash bin/es-do…"   6 months ago        Up 3 weeks
   9200/tcp, 9300/tcp                 cortex_elasticsearch_1
```

Figure 2-8. Docker containers for Cortex

Once running, the analyzers and responders are available in the following routes of the Cortex container, as shown in Figure 2-9.

```
root@f7edecb9c871:/opt/Cortex-Analyzers# ls analyzers/
Abuse_Finder        DNSDB              GoogleSafebrowsing  MISP              Patrowl           SinkDB          VirusTotal
C1fApp              DShield            GreyNoise           MISPWarningLists  PayloadSecurity   StaxxSearch     Virusshare
CERTatPassiveDNS    DomainTools        HADES               Malpedia          PhishTank         StopForumSpam   WOT
CIRCLPassiveDNS     EmergingThreats    HIBP                Malwares          PhishingInitiative Threatcrowd    Yara
CIRCLPassiveSSL     EmlParser          Hashdd              MaxMind           ProofPoint        TorBlutmagie    Yeti
Censys              FileInfo           Hippocampe          MnemonicPDNS      Pulsedive         TorProject
Crtsh               File_Info          Hunterio            MsgParser         RecordedFuture    URLhaus
CuckooSandbox       FireEyeiSight      HybridAnalysis      Nessus            Robtex            Umbrella
CyberCrime-Tracker  FireHOLBlocklists  IBMXForce           OTXQuery          SOCCRATESImpact   UnshortenLink
Cyberprotect        Fortiguard         Investigate         Onyphe            SecurityTrails    Urlscan.io
Cymon               GoogleDNS          JoeSandbox          PassiveTotal      Shodan            VMRay
root@f7edecb9c871:/opt/Cortex-Analyzers# ls responders/
Mailer  UmbrellaBlacklister
```

Figure 2-9. Default Cortex analyzers and responders

Cortex provides a control panel that is available at port 9001. Figure 2-10 shows the main window of the Cortex control panel.

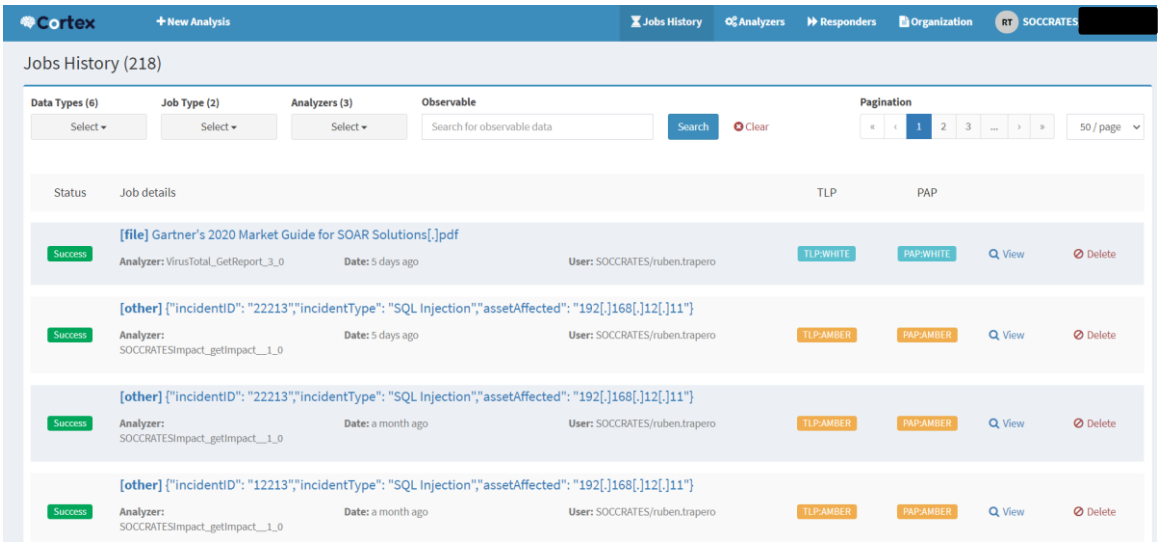

Figure 2-10. Main window of the Cortex control panel

Roles and organizations need to be configured in Cortex. For this, a Cortex Super Administrator needs to be configured in the platform. This super administrator will not be able to enable, configure or invoke cortex analyzers and responders. This super administrator will be able to create organizations, create users, assign roles to users (for example, assign a user as administrator of an organization). The screenshot of Figure 2-11 shows the user 'soccrates' as super administrator, and other user as administrator of the SOCCRATES organization with permissions to read and analyse.

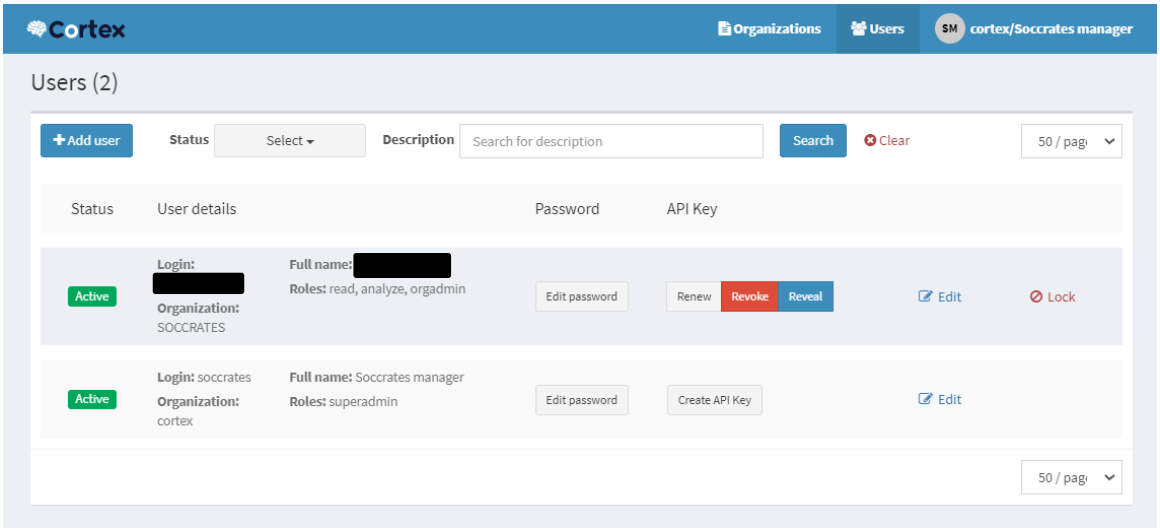

Figure 2-11. Panel for configuring Cortex users

Cortex usage

For its usage, analyzers and responders need to be enabled and configured. They can be enabled from the Organization menu for a user with permissions to manage analyzers (Figure 2-12).

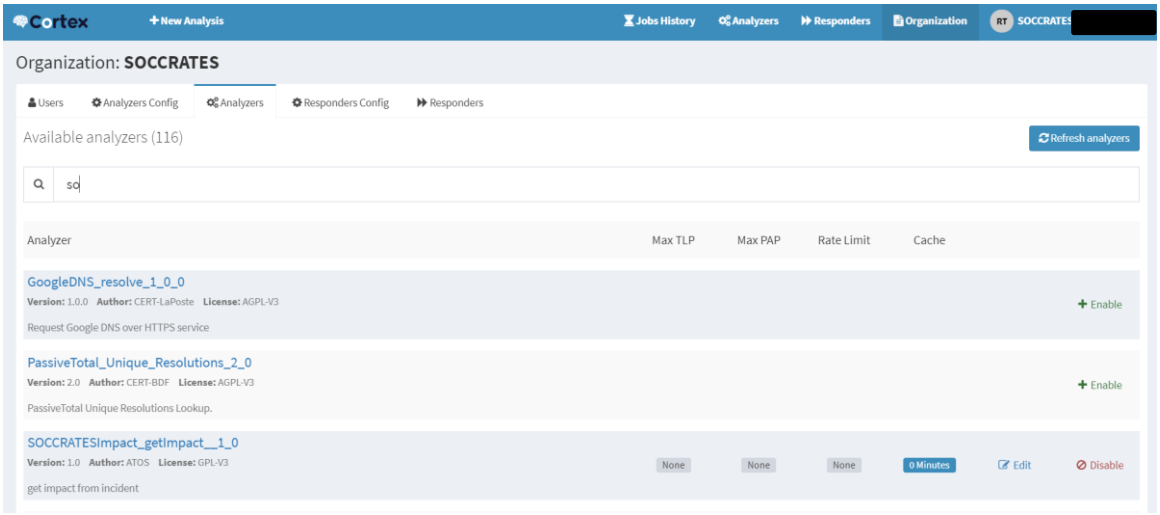

Figure 2-12. Cortex panel for enabling analyzers

Apart from enabling analyzers or responders, it is required to configure them. Figure 2-13 represents the panel for configuring an analyzer. It has three sections:

- Basic details: contains the name of the analyzer. It comes from the analyzer itself and cannot be changed.
- Configuration: every analyzer has a set of data that is required for its configuration. This includes usernames, keys for accessing the tool, intervals for invoking the analyse, the URL of the endpoint to invoke and any other configuration that the tool to invoke requires.
- Options: These fields are used to manage several additional aspects that are common to all analyses. Two important options are the PAP (Permissible Action Protocol) and TLP (Traffic Light Protocol) levels, represented as a colour code (being white as the less critical and red the most critical), which defines the minimal level of the request and its observables to trigger the analyzer.

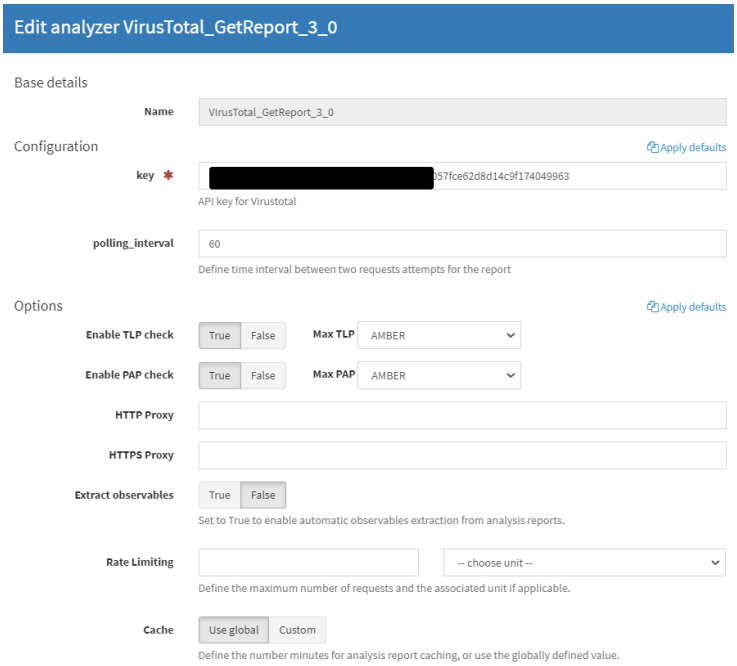

Figure 2-13. Cortex panel for configuring an analyzer

Creating an analyzer (or responder)

Analyzers are developed using python. The process for creating an analyzer requires three parts: the descriptor of the analyzer, the python script of the analyzer and, optionally, the description of the required python libraries. The following example describes the process for creating an analyzer that interacts with a Web Service that calculate an impact level for a given incident. Figure 2-14 depicts the content of the descriptor file for a custom analyzer. The important fields are:

- The name of the Analyzer as it will be shown in the Cortex panel.
- The Data Type List, which contains the format of the data to be exchange between the Cortex analyzer and the tools. There is a list of predefined types, including domains, files, ip addresses, urls or 'other' for a data type not listed (such as JSON).
- The 'config' field contains the default configuration options for this analyzer, such as the min or max TLP and PAP, and the name of services included in the Cortex analyzer. This is important because a Cortex analyzer might contain more than function (i.e., getImpact, setImpact, deleteIncident could be possible functions for the example here described).
- Command: contains the location and name of the file that contains the code of the analyzer.
- ConfigurationItems contains the description of the required specific configuration. For example, the URL of the function to invoke is included here.

```
{
    "name": "SOCCRATESImpact_getImpact_",
    "version": "1.0",
    "author": "ATOS",
    "url": "https://github.com/TheHive-Project/Cortex-Analyzers",
    "license": "GPL-V3",
    "description": "get impact from incident",
    "dataTypeList": ["other"],
    "baseConfig": "SOCCRATESImpact",
    "config": {
        "check_tlp": false,
        "service": "getImpact",
        "max_tlp": 1
    },
    "command": "SOCCRATESImpact/soccratesImpact.py",
    "configurationItems": [
        {
            "name": "url",
            "description": "url of the impact service",
            "type": "string",
            "multi": false,
            "required": true
        }
    ]
}
```

Figure 2-14. Descriptor of a custom Cortex analyzer

Figure 2-15 represents the code of a simple analyzer that invokes a Web Service through a REST API. The most important parts have been highlighted:

- Obtain required variables from the configuration (at the def __init__ method)
- Determine the function or functions to invoke (at the 'run' method)
- Extract the data to be included in the request
- Perform the actual request

```
class SOCCRATESImpactAnalyzer(Analyzer):

    def __init__(self):
        Analyzer.__init__(self)
        self.service = self.get_param('config.service', None, 'Service parameter is missing')
        self.businessImpact_url = self.get_param('config.url', None, 'Missing URL of the Business Impact tool')

    def check_response(self, response):
        status=response.status_code
        if status == 400:
            logprint("Response Error",str(status) + " - " + str(response.content))
            self.error('Status: ' + str(status) + " - " + str(response.content))
        return response

    def run(self):
        if self.service == 'getImpact':
            #Get input
            if self.data_type == 'other':
                inputData = self.get_param('data')
                logprint("input", inputData)
                json_data = json.loads(inputData)
                response = requests.post(self.businessImpact_url,json_data)
                self.check_response(response)
                self.report(json.loads(response.content))
                logprint("output",response.content)
            else:
                self.error('Invalid data type')

        else:
            self.error('Invalid service')


if __name__ == '__main__':
    SOCCRATESImpactAnalyzer().run()
```


Figure 2-15. Code for a custom Cortex analyzer

Once done, if there are no errors detected by the Cortex engine, the new analyzer will appear at the Cortex panel and will be ready to be used, as shown in Figure 2-16:

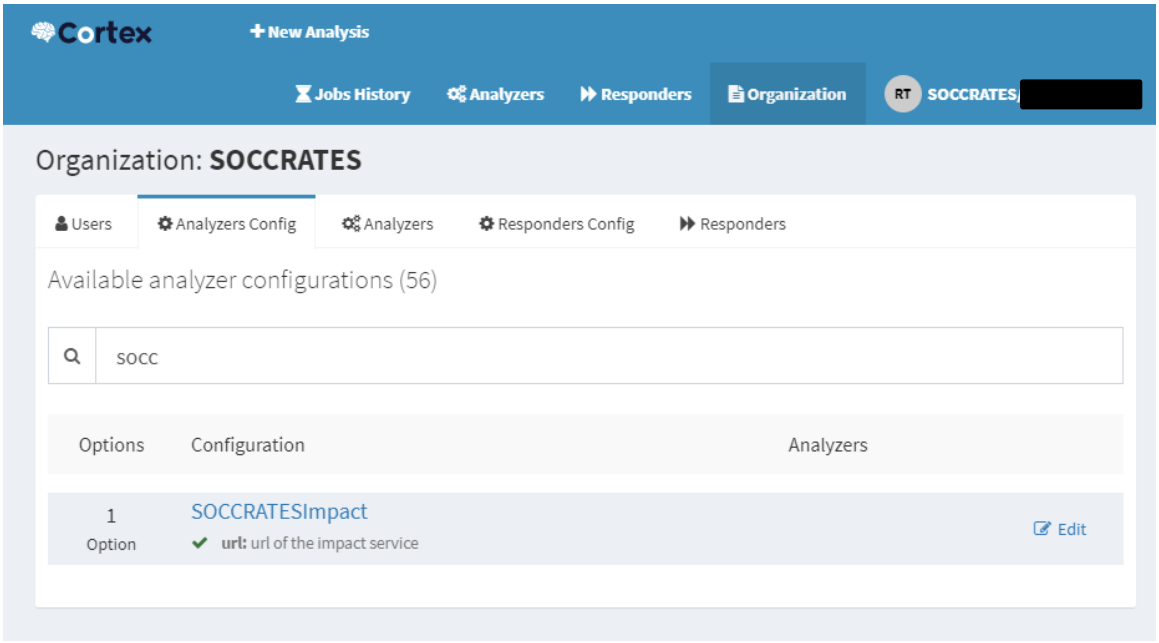

Figure 2-16. Cortex panel with a new analyzer

Invoking Cortex analyzers (or responders)

There are two ways to invoke Cortex analyzers and responders: directly from the Cortex panel or through its REST-API.

From the Cortex panel it is required to access to the Analyzers menu and under the active Analyzer click on "Run". A menu will be prompted to insert the required data to invoke the analyzer (i.e., the incident information for an Impact Analyzer or a file for a Virus Scanner). The menu will be automatically adapted to the type of input required, as it can been seen in Figure 2-17.

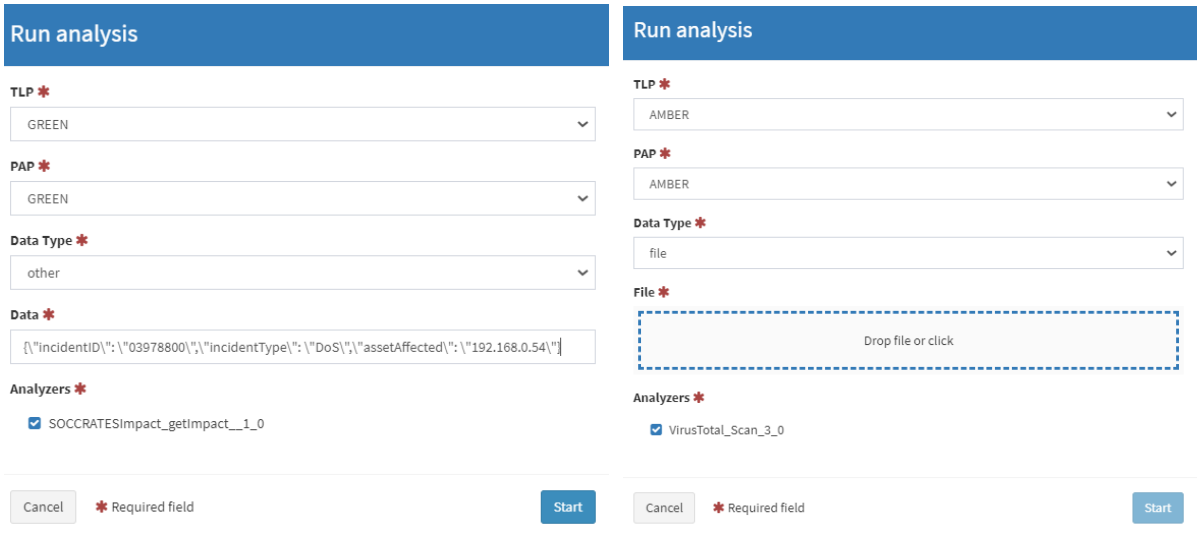

Figure 2-17. Configuration required to invoke two different analyzers from the control panel

From the REST-API it is required to know the URL of the analyzer to invoke and have an authorization key. The authorization key is a string that can be obtained from the user settings. The URL of the analyzer can be obtained through the REST-API. There is a complete set of functions available to interact

with Cortex with its REST-API[23]. For example, the following listing is the result of the request to the REST-API for obtaining information about the active analyzers. It includes the id field that can be used to invoke the Cortex analyzer in the following address:
http://<server_address>:9001/api/analyzer/a87935516f80286e3c3cead4decf29bd/run


```
[
   {
      "dataTypeList":[
         "other",
         "hash"
      ],
      "type":"analyzer",
      "description":"get impact from incident",
      "name":"SOCCRATESImpact_getImpact__1_0",
      "createdBy":"XXX.XXXX",
      "createdAt":1591103787504,
      "workerDefinitionId":"SOCCRATESImpact_getImpact__1_0",
      "jobCache":0,
      "updatedBy":"XXXX.XXXX",
      "updatedAt":1598004646127,
      "_type":"worker",
      "_routing":"SOCCRATES",
      "_parent":"SOCCRATES",
      "_id":"a87935516f80286e3c3cead4decf29bd",
      "_version":22,
      "id":"a87935516f80286e3c3cead4decf29bd",
      "maxTlp":2,
      "maxPap":2,
      "version":"1.0",
      "author":"ATOS",
      "url":"https://github.com/TheHive-Project/Cortex-Analyzers",
      "license":"GPL-V3",
      "baseConfig":"SOCCRATESImpact",
      "analyzerDefinitionId":"SOCCRATESImpact_getImpact__1_0",
      "configuration":{
         "url":"http://127.0.0.1:5000/analyze/",
         "auto_extract_artifacts":false,
         "proxy_http":null,
         "proxy_https":null,
         "check_tlp":false,
         "max_tlp":2,
         "check_pap":false,
         "max_pap":2
      }
   },
   {
      "dataTypeList":[
         "file",
         "hash",
         "domain",
         "ip",
         "url"
      ],
      }
   }
]
```


[23] https://github.com/TheHive-Project/CortexDocs/blob/master/api/api-guide.md

Figure 2-18 represents the POST request to the Cortex analyzer. It includes the authorization key as a header while the body includes a JSON document with the data expected by the Cortex analyzer. The field “data” from this JSON will be used by the python file of the Cortex analyzer to invoke the service.

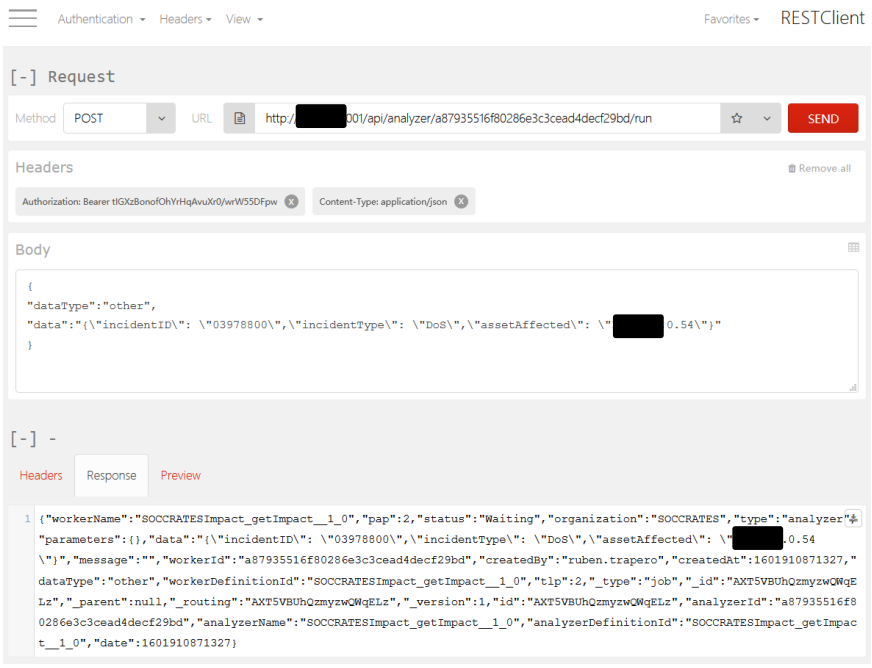

Figure 2-18. Request to Cortex Analyzer using REST-API

This request can be checked at the Cortex panel in Figure 2-19, where the successful response is given, and the data obtained.

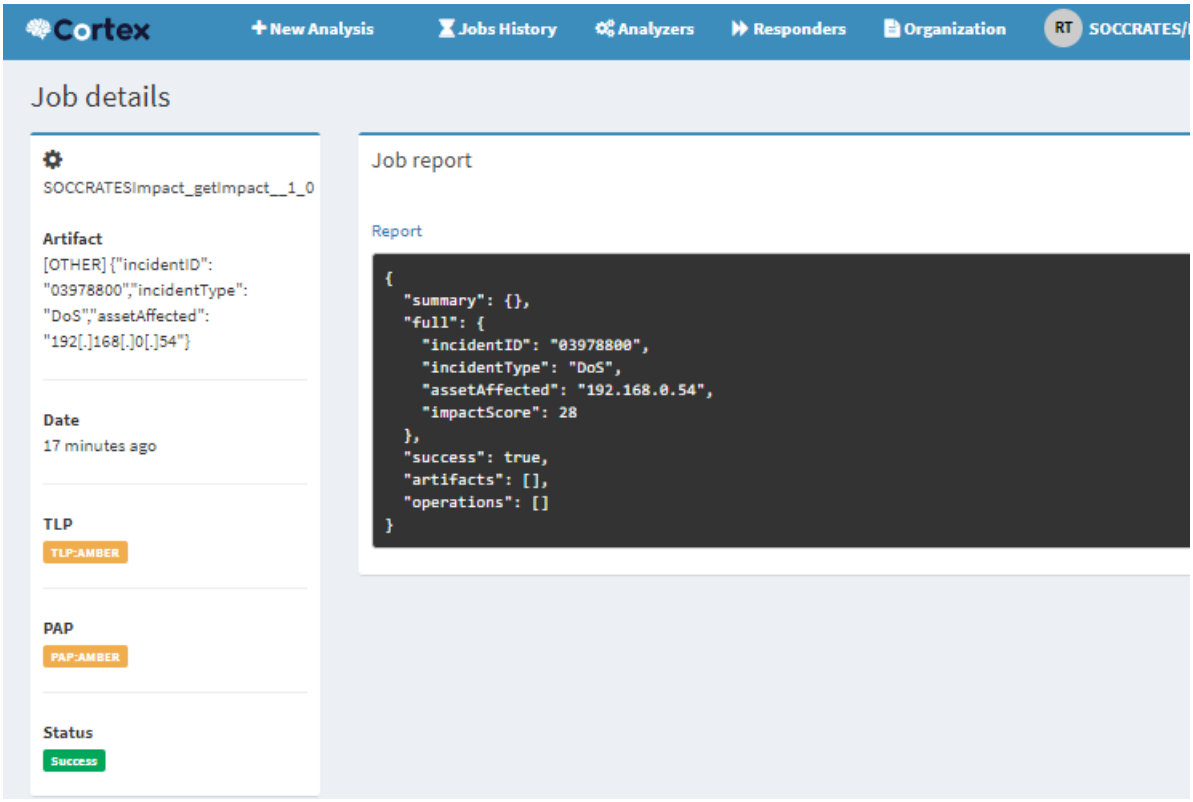

Figure 2-19. Result of a request to a Cortex analyzer

2.4 Initial deployment

An initial deployment of an Activiti engine and Cortex has been deployed at the premises of Atos. This deployment has been used to validate the usage of both technologies and ensure the correct interoperation for the development of the SOCCRATES Orchestrator. Figure 2-20 represents the deployment of these components.

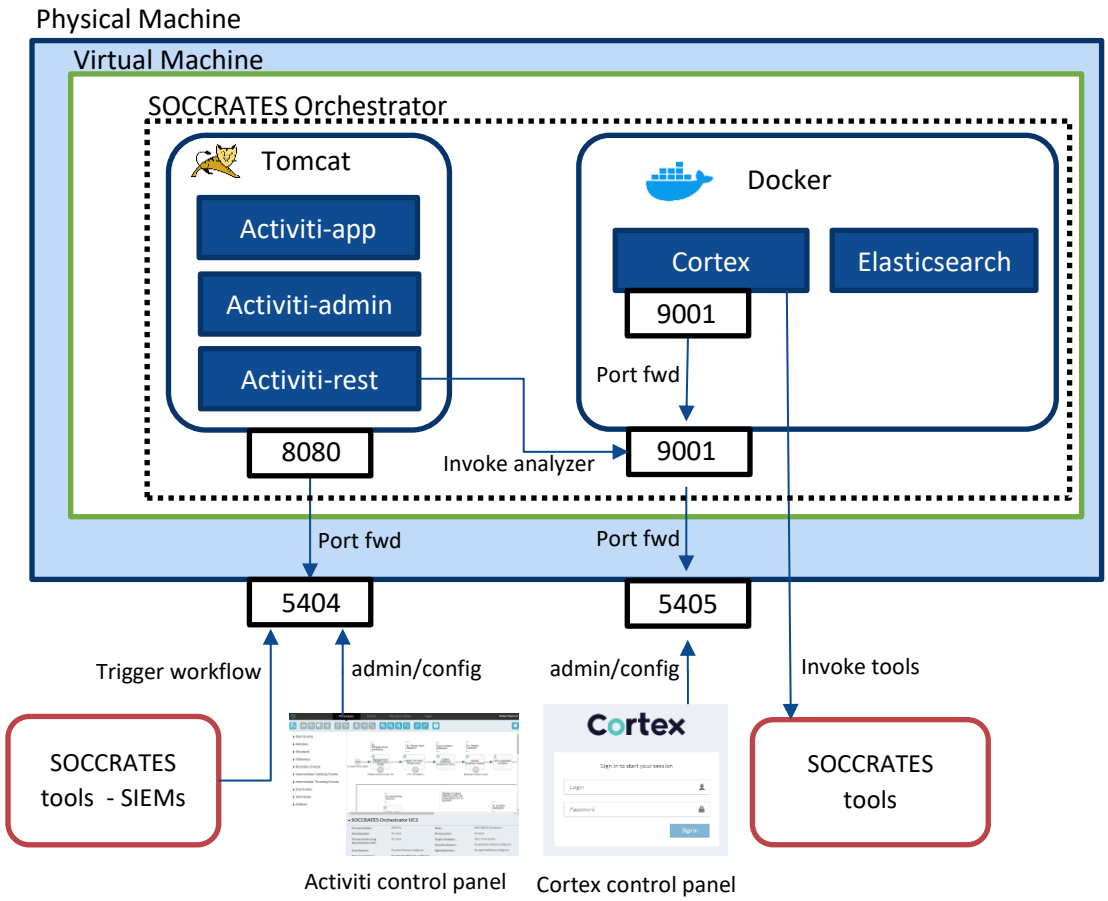

Figure 2-20. Orchestrator testbed deployed at Atos

The components of the Orchestrator are deployed in a virtual machine that is running in a shared physical server. In this virtual machine an Apache Tomcat server has been deployed, where the Activiti application has been installed. This server is accessible to the public internet through port 5404, which has been forwarded at the physical machine hypervisor from the default port 8080 at the virtual machine. This is the port used by the SOCCRATES tools that will trigger workflows. This port is also used to access the graphical user interfaces of Activiti.

A Docker instance is running in the virtual machine, which contains the two containers required by Cortex: Cortex itself and Elasticsearch. The Cortex container uses port 9001, which is linked to the port 9001 at the virtual machine by the Docker service. This port 9001 is forwarded to port 5405 at the physical machine hypervisor, which can be used to access the Cortex control panel for managing configuration, analyzers, etc. Cortex will access external SOCCRATES tools directly from the container. No port forwarding is required as this is an outbound connection from the container to the public Internet. Internally, at the virtual machine, the activiti-rest app access to Cortex REST-API through the port 9001 to invoke analyzers.

2.5 Proof of concept for the SOCCRATES orchestration

A proof of concept has been developed that shows the suitability of the solution for the SOCCRATES Orchestrator. Figure 2-21 represents the workflow that has been created and published in Activiti.

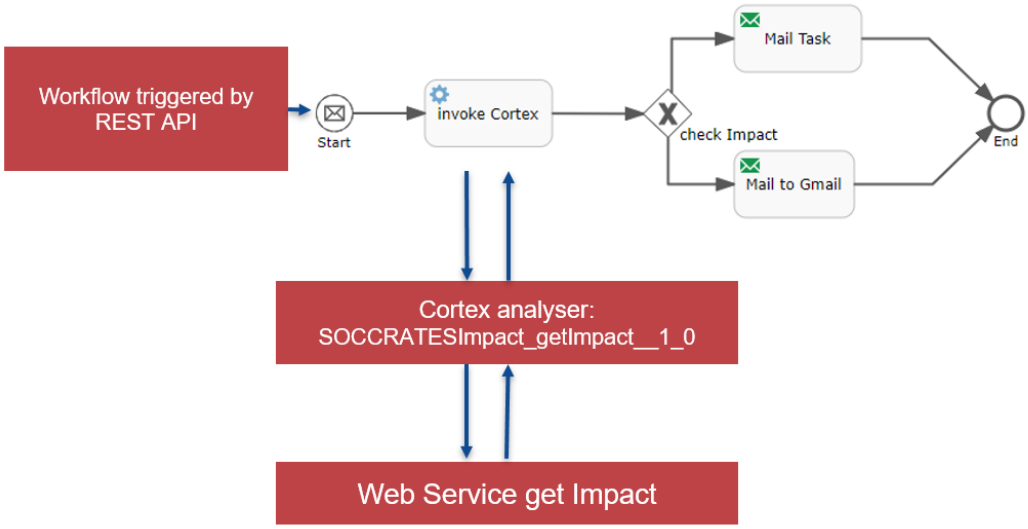

Figure 2-21. Activiti workflow for the proof of concept

This workflow is triggered with a message that is included in the request carried out through the Activiti REST API. Next is a service task invokes Cortex. This task invokes a Java class that interacts with Cortex to invoke an analyzer. The Cortex analyzer invoked is the custom Analyzer described in Section 2.4.2, which interacts with a simple Web service that calculates an impact score for a given incident. The diagram at Figure 2-22 shows the flow of interactions. The workflow, through a service task, invokes the Cortex analyzer that interacts with the Web Service that calculates the impact.

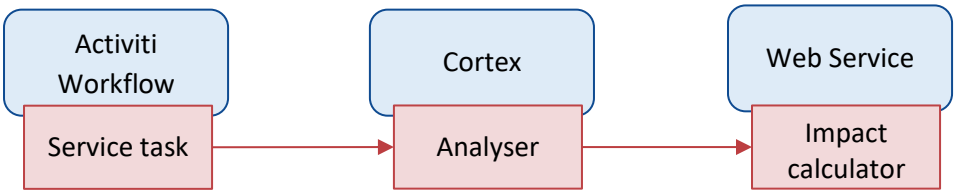

Figure 2-22. Interaction between components in the proof of concept

The Web Service created is a simple one, as shown in Figure 2-23. It exposes a REST API that requires a JSON with information about an incident (incidentID, incidentTypes, assetsAffected), and return the same JSON adding the impactScore. The impact Score is obtained as a random value between 0 and 100.

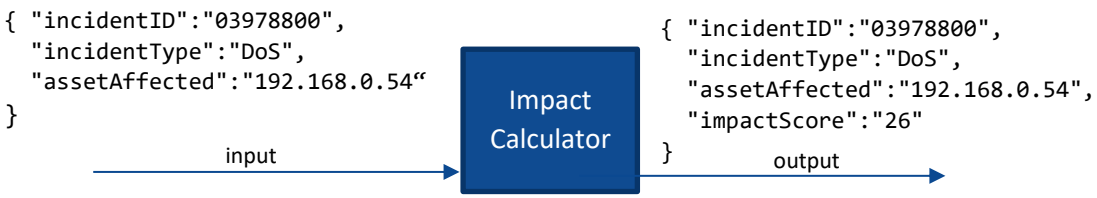

Figure 2-23. Inputs and output of the proof of concept Web Service

Following the workflow of the proof of concept, once the Cortex analyzer is invoked, an exclusive gateway compares the value of the impact. If the impact is above 50, a mail task sends an email to a recipient notifying about a high impact incident. If the impact is below 50 the email is sent to a different email address.

Figure 2-24 represents the POST message that triggers the workflow. The body of the request includes the message format required by Activiti to trigger the workflow. It includes the name of the message ("incidentinput") and a set of variables described as a tuple "name" and "value". In this case there is a single variable "incident" whose value contains the information about the incident, as required by the Cortex analyzer to invoke the Web Service. Once the workflow is triggered those variables included in the message are part of the execution environment of the workflow and are available from any element of the workflow (tasks, gateways, etc).

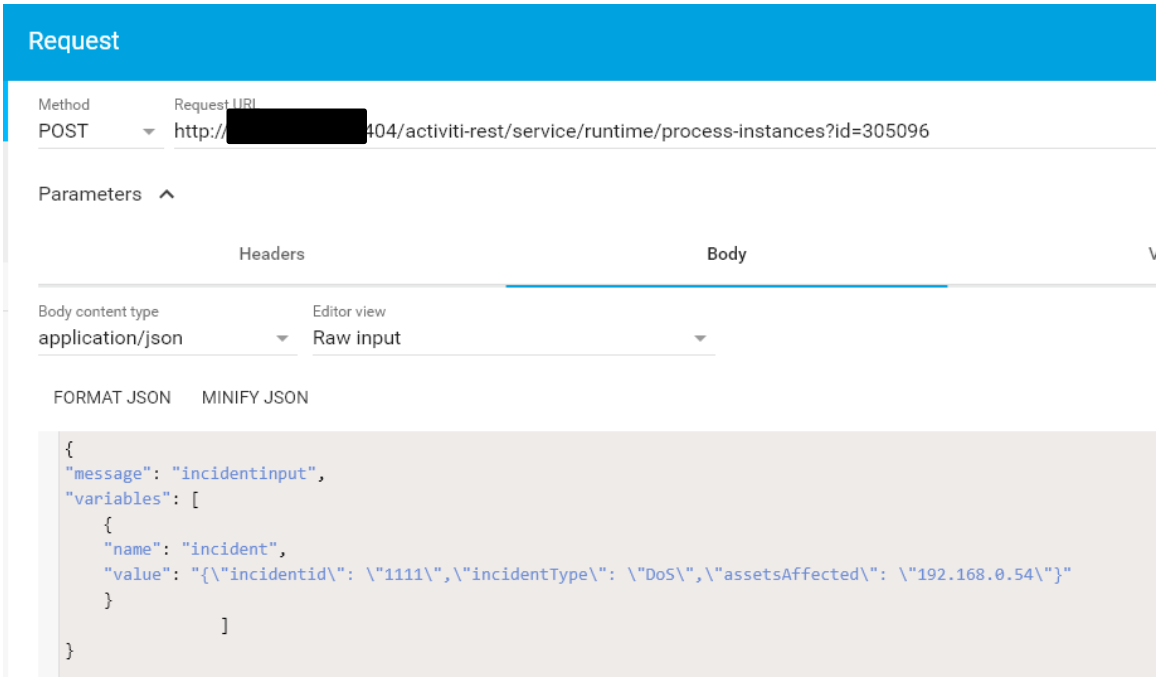

Figure 2-24. Request to trigger the workflow of the proof of concept

The class that is triggered by the service task of the workflow extracts the value "incident" from the execution environment of the workflow and create the request to the Cortex analyzer that will invoke the Web Service. Figure 2-25 represents the code this class.

```
package cortextest;
import org.activiti.engine.delegate.DelegateExecution;

public class CortexTest implements JavaDelegate {
    public void execute(DelegateExecution execution) {

        String key = "[redacted]ow";
        String input = (String) execution.getVariable("incident");
        System.out.println(input);
        //convert input to json;
        try {
            String escapedString = StringEscapeUtils.unescapeJava(input);
            JSONObject input_json = new JSONObject(escapedString);
            //extract incidentID from input json
            String incidentid = input_json.getString("incidentid");
            String incidentType= input_json.getString("incidentType");
            String assetsAffected=input_json.getString("assetsAffected");
            String inputdata = "{\"incidentID\": \"" + incidentid + "\",\"incidentType\": \""
                    + "" + incidentType + "\",\"assetAffected\": \"" + assetsAffected + "\"}";
            String result = GetImpact.invokeCortex(key,inputdata);
            JSONObject obj = new JSONObject(result);
            String status = obj.getString("status");
            if (status.equals("Failure")) {
                execution.setVariable("executionStatus", status);}
            else if (status.contentEquals("Success")) {
                JSONObject report = obj.getJSONObject("report");
                execution.setVariable("executionStatus", status);

                String data = report.getString("full");
                String impactScore = report.getJSONObject("full").getString("impactScore");
                execution.setVariable("bi_output", data);
                execution.setVariable("impact", impactScore);}
        } catch (JSONException e) {
            e.printStackTrace();}
    }
}
```

Extracts variable incident from the workflow execution environment

Call to class that invokes cortex

Obtains the result of the POST request to Cortex

Set the result from Cortex as variables that become part of the workflow execution environment

Figure 2-25. Java class executed with the Service Task of the proof of concept

In this code it is shown that an additional class is invoked (GetImpact.invokeCortex) that receives the information required to invoke the REST API of the Cortex analyzer. Figure 2-26 shows the piece of code that invokes the analyzer. It is a simple HTTP POST with the required headers, the URL of the Cortex analyzer and the data in the body of the request.

```
url = new URL("http://10.0.2.3:9001/api/analyzer/a87935516f80286e3c3cead4decf29bd/run");
HttpURLConnection con = (HttpURLConnection) url.openConnection();
con.setRequestMethod("POST");
con.setRequestProperty("Content-Type", "application/json");
con.setRequestProperty("Authorization", "Bearer "+ key);
con.setDoOutput(true);
JSONObject json = new JSONObject();
json.put("dataType", "other");
json.put("tlp", "0");
json.put("data", datainput);
String data = json.toString();
try(OutputStream os = con.getOutputStream()){
    byte[] input = data.getBytes("utf-8");
    os.write(input, 0, input.length);}
```

Figure 2-26. Excerpt of code that invokes the Cortex Analyzer

Figure 2-27 shows the email sent by the Activiti workflow after an execution.

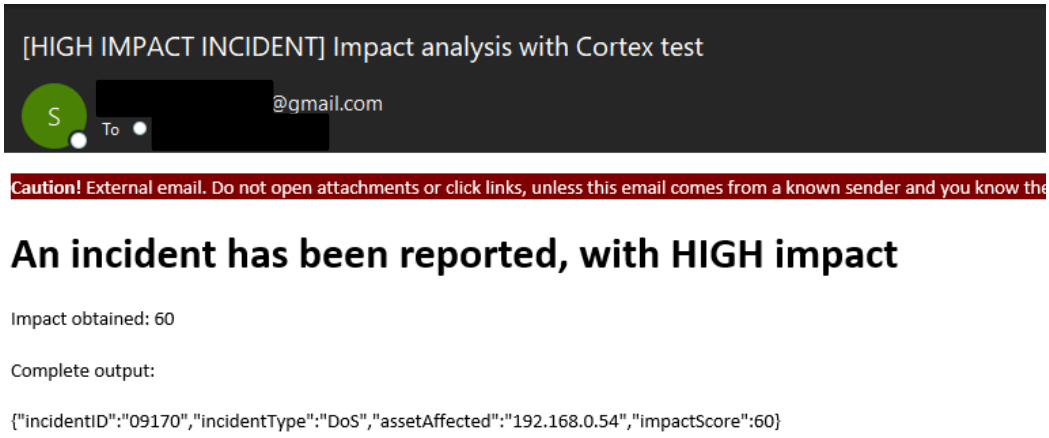

Figure 2-27. Result of execution of the workflow of the proof of concept

As a result, there is a successful integration of Activiti with Cortex, demonstrating that it is perfectly viable for its use as SOCCRATES Orchestrator. A full operating SOCCRATES Orchestrator will have Cortex analyzers for every tool that needs to be invoked from it. The SOCCRATES orchestrator admins will create workflows adapted to the needs of the infrastructure. In SOCCRATES, there will be a workflow for every Use Case to implement. The following section provides an initial draft of the Activiti workflows for the SOCCRATES use cases.

2.6 Use case activiti workflows

SOCCRATES is considering five Use Cases, as described in deliverable D2.1:

- Use Case 1: Response on Detected Ongoing Attack
- Use Case 2: Response on Newly Received Cyber Threat Intelligence
- Use Case 3: Response on Newly Discovered New Vulnerable Assets
- Use Case 4: Response on Discovered System Configuration Change
- Use Case 5: Response on Deployment of New Systems in Infrastructure

This section provides with the initial workflows depicted in Activiti. These workflows represent an initial draft of the workflows to implement. These workflows count with Service Tasks that will be linked to Java classes, which will implement the interconnection to the Cortex Analyzers developed for every tool.

For the reasoning behind these use cases the reader is referred to deliverable D2.1. Note that in the coming months these use cases will be updated and specified in more detail in SOCCRATES deliverable D2.3.

2.6.1 Workflow for UC1

UC1 is, by far, the most complex of the workflows of all use cases defined in SOCCRATES. For a better reading it has been separated into three parts, as depicted in the following figures.

The first part is depicted in Figure 2-28. It represents the initial analysis of an incident. This workflow is triggered by SIEMs that detect an incident. The message included in the trigger request will contain information about the incident: type of incident, timestamp, asset or assets affected, etc. The information related to this incident is passed through different tools, that provide additional analysis and add information to the incident. This initial analysis finished with the SOC admin evaluating this analysis and deciding whether it can be considered an incident worth to be properly handled.

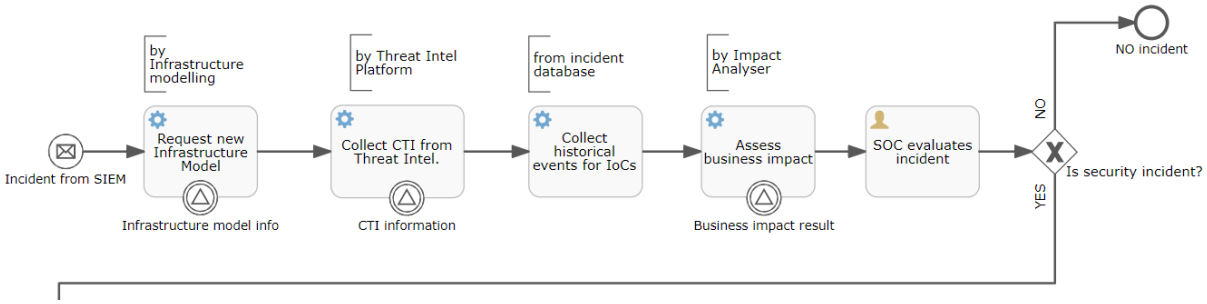

Figure 2-28. Initial incident analysis for UC1

The second part of the workflow is depicted in Figure 2-29. It is executed when the SOC analyst considers that the incident is relevant enough to perform a deeper analysis. In this case two activities are done in parallel. On the one side an intensive monitoring, deploying additional detectors focused on the type of incident detected, is performed, while, on the other side, an ADG analysis is requested to determine the assets affected; the result from this latter analysis is used to perform an impact analysis on the affected assets. After this deeper analysis, the SOC selects the containment and the workflow continues. It is worth noticing that in this figure a subprocess is shown for updating an incident database. It is expected that, after every task is finished, the information about the incident, resulting from the execution of every task, is updated at a persistent database of the Orchestrator.

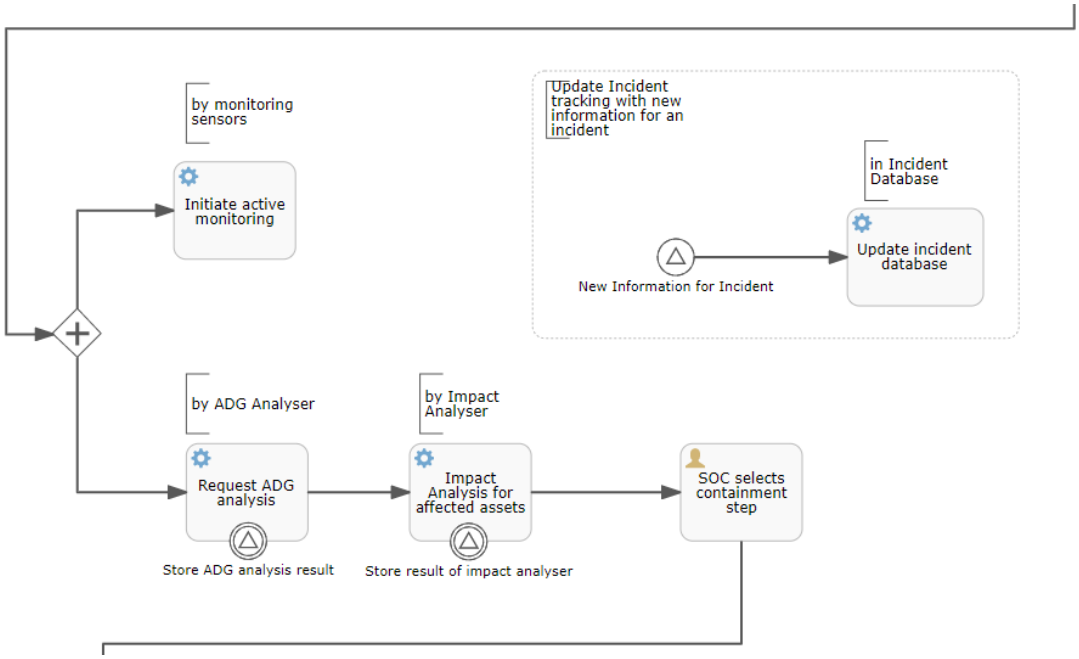

Figure 2-29. Deeper incident analysis for UC1

The third part of the workflow, depicted in Figure 2-30, determines the appropriate response for the incident. It consists of three branches that includes activities such as determining a preliminary response to stop the impact, while, in parallel, a containment strategy that involves the CoA generator, Impact Analyser and Threat Intelligence Platforms tools among others. The flow can go towards any of the three branches, which depends on the selection made by the SOC at the end of Figure 2-29, which depends on the expert knowledge based on the results obtained from the ADG and Impact Analyser in the deeper incident analysis.

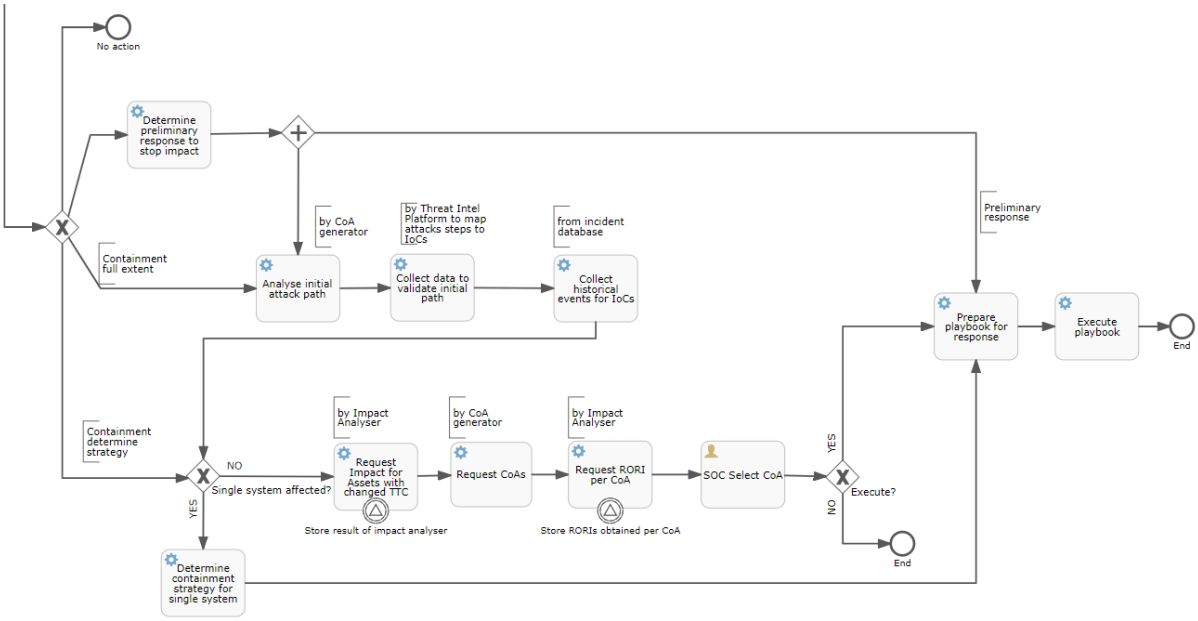

Figure 2-30. Determine response in UC1

2.6.2 Workflow for UC2, UC3, UC4 and UC5

The workflow for UC2, UC3, UC4 and UC5 is mostly the same, with the difference on the type input received:

- For UC2 it is received information about the threat identified
- For UC3 it is received information about the new vulnerability discovered
- For UC4 it is received information about the detected configuration change
- For UC5 it is received information about the new asset inserted in the infrastructure.

The workflow is depicted in Figure 2-31. For all the UCs, the tools to invoke are the same. The process starts with the request of an infrastructure model. The input to this infrastructure model will depend on the type of message received. The ADG is then invoked, inserting in the request the infrastructure model before and after the event the triggered the use case (for example, before and after an asset is inserted). The ADG returns a simulation of the effects of these changes, which are forwarded to the Impact Analyzer for determining the impact. This impact is evaluated by the SOC who decides whether to continue towards the response or not. In the positive case, a response is determined in the last part of the workflow.

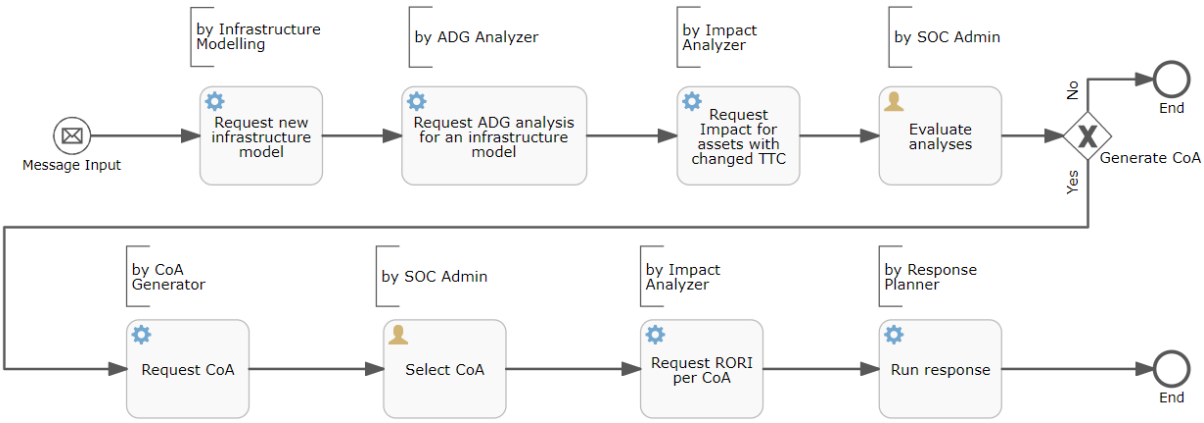

Figure 2-31. Workflow diagram for UC2, UC3, UC4 and UC5

3 Reconfiguration

Typically, CoAs are enforced on IT systems that are not necessarily under the control of the SOC or CSIRTs, especially in large organizations or MSSPs. Therefore, the execution of those Course of Actions depends a lot on the type of organization, the internal procedures related to IT systems and on the characteristics of the infrastructure, i.e., multiple administrative domains, whose complexity and size can be an obstacle to the automatic enforcement of countermeasures. To deal with this, SOCCRATES relies on Course of Actions to guide IT system administrators on the actions to deploy in the infrastructure, to react to the incident or the issues that have been discovered (Figure 3-1). The Reconfiguration component in SOCCRATES acts as link between the CoA produced after the execution of the workflow at the Orchestrator engine and the IT systems of the infrastructure.

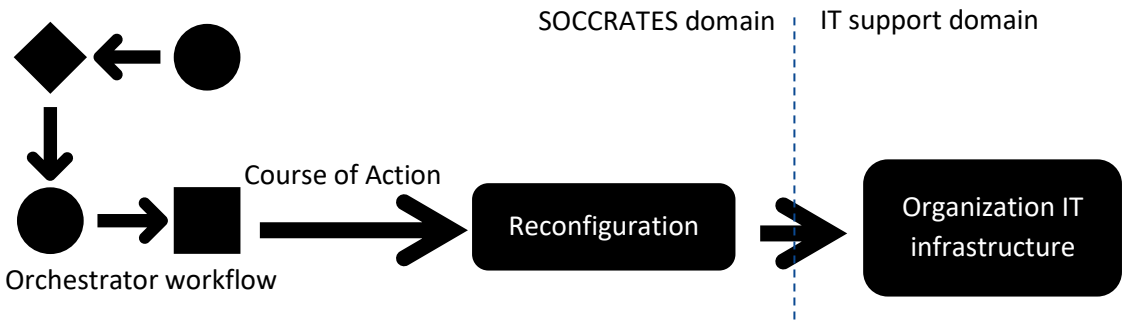

Figure 3-1. Link between the orchestrator workflow and the reconfiguration

To this end, the Reconfiguration component allows to transform a CoA to direct actions that might be directly enforced or used by IT staff to manually enforce in an organization's infrastructure. In both cases, a human in the loop is required to authorize changes in the IT infrastructure, especially when those changes are critical and might involve turning off certain systems and consequently impacting business continuity of the organization.

In both cases, according to Deliverable D2.2, the Reconfiguration component can be interconnected with different types of infrastructure systems:

- Direct communication with infrastructure systems. This aspect considers aspects such as launching scripts directly on a specific machine (for example, force the update of a vulnerable software or to reboot the machine).
- Use OpenC2, a standardized language for the command and control of technologies that provide or support cyber defences[24], for triggering actions on compatible systems. This happens in case the infrastructure where the SOCCRATES platform is operating has OpenC2 compatible systems. This compatibility with OpenC2 can be either direct (although currently the list of profiles available is very limited in the OpenC2 specification) or through a proxy that transforms OpenC2 commands to direct actions to infrastructure systems.

[24] https://openc2.org/

- Additionally, there is the option of using additional proxies to communicate using different mechanisms (proprietary or not) that transform these mechanisms to direct actions to infrastructure systems.

The above actions (either custom scripts or OpenC2 commands) will be received by IT service management to be enforced in the organization infrastructure. To this end, the Reconfiguration component can carry out additional activities. For example, it might be possible that the IT service management counts with a tracking system where actions are registered and assigns them to a specific supervisor before being enforced. Figure 3-2 depicts an example of the internal processes of the reconfiguration action. It can include the following actions:

- Open ticket, to keep track of the reconfiguration response
- Analyst action, to be validated, in case it is required, by a human analyst.
- Trigger reconfiguration, to enforce the reconfiguration actions in the infrastructure
- Close ticket, to close the opened ticket once the reconfiguration has been triggered or once the IT system manager has considered the incident or the issue has been solved.

In order to give the possibility to IT system administrators to automate this process, as much as possible, the SOCCRATES platform allows to develop them as a subprocess in the Activiti engine. As it has been mentioned, SOCCRATES gives this possibility, although its use depends on the internal procedures of the organization regarding IT systems administration. The solution chosen for implementing the Orchestrator and Integration engine is flexible enough to be adapted to the Reconfiguration activities. For example, the action of involving a human actor (system administrator action) can employ the User tasks available at Activiti. This would allow to automatically assign this action to a specific group of people according to their role.

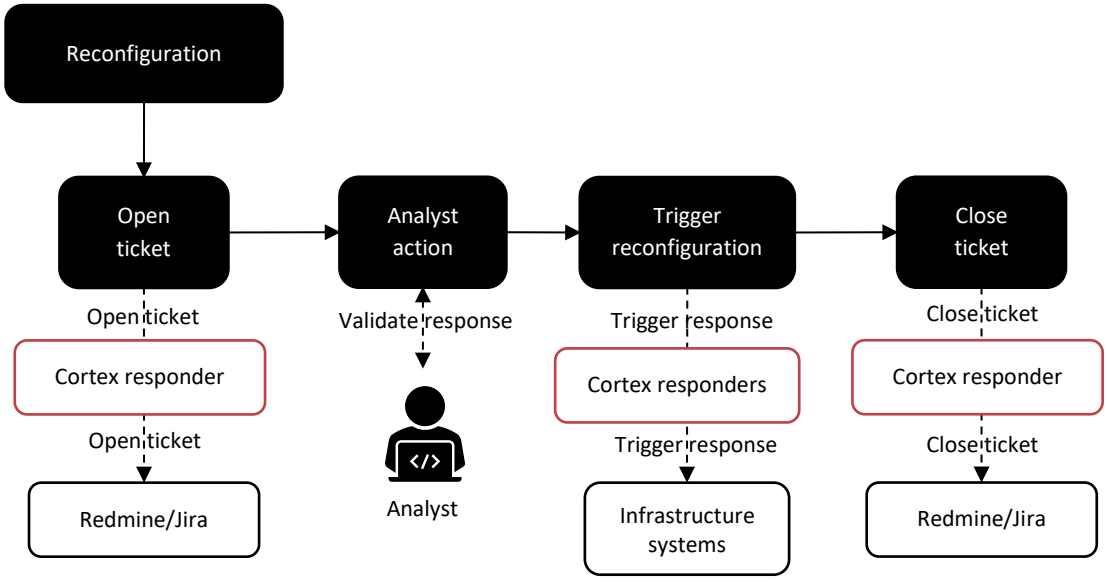

Figure 3-2. Subprocess workflow for the reconfiguration task

The actions of interacting with the ticketing system or to trigger the reconfiguration requires the interaction with external components. Cortex responders can be used to interact with these elements. Cortex responders, contrary to Cortex analyzers, are conceived to trigger actions in external

components that do not require to return data in the response. A typical example of a Cortex responders is the one used to send notification emails. The activities for opening tickets in a ticketing system would require developing or reusing currently available Cortex responders for that purpose (i.e., a Redmine responder that is currently available[25]).

With respect to the task in charge of triggering the reconfiguration, as mentioned above, it can be triggered manually by the IT system administrator (once it has been considered that it is safe to enforce it). However, with the SOCCRATES Reconfiguration component and in case the IT system management of the organization supports it, it is possible to automatically enforce it. To this end, the Activiti workflow and Cortex responders can also be used. For instance, one or more Cortex responders will have to be developed, depending on the way to enforce the reconfiguration. Figure 3-3 depicts an example of the procedure of using Cortex responders for triggering reconfiguration actions.

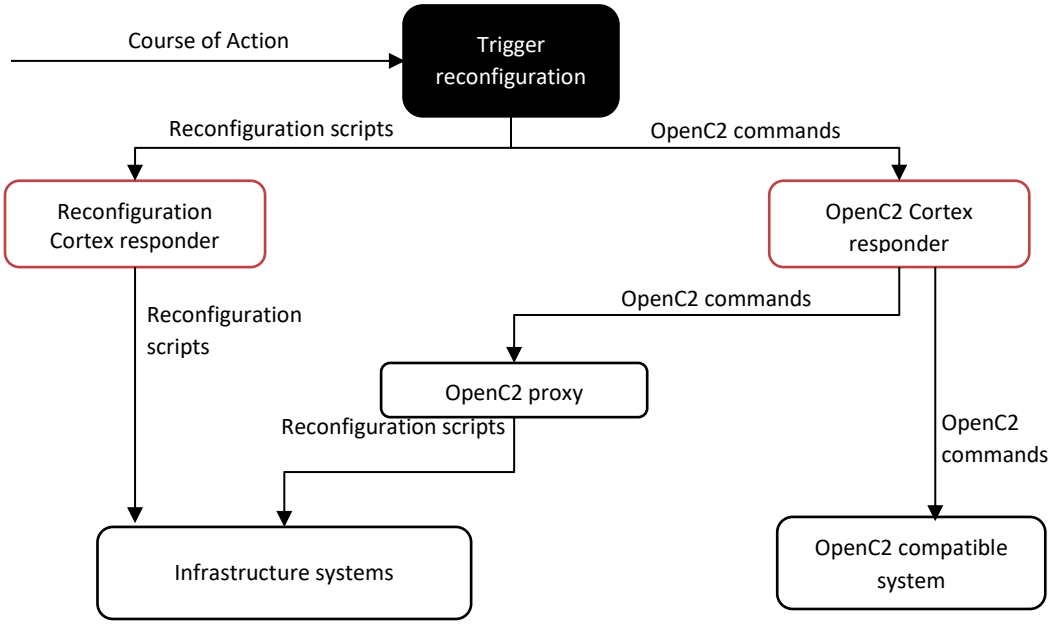

Figure 3-3. Process for triggering reconfiguration actions with Cortex

Depending on the characteristics of the infrastructure, several Cortex responders could be required:

- Reconfiguration Cortex responder, which directly receives the scripts to be launched at the affected infrastructure systems. This will happen when these affected infrastructure systems are proprietary or not compatible with standard management commands such as OpenC2. In this case, the triggered reconfiguration task at the Activiti engine will transform the Course of Action received from the Orchestrator workflow to these executable scripts.
- OpenC2 Cortex responder, which allows to interact with OpenC2 compatible systems. To this end, the triggered configuration task at the Activiti engine will transform the Course of Action received from the Orchestrator workflow into specific OpenC2 commands. Additionally, conventional infrastructure systems can also be triggered through this responder, if an OpenC2 proxy is available at the infrastructure to transform OpenC2 commands to reconfiguration scripts runnable at the infrastructure systems.

[25] https://github.com/TheHive-Project/Cortex-Analyzers/tree/master/responders/Redmine

- As it has been mentioned, the Trigger reconfiguration activity needs to transform Course of Action results to OpenC2 commands. In the table below, we have included two examples for a translation to OpenC2.

Table 3-1. CoA - OpenC2 example for containment action

<table>
<tr><th colspan="2">Example: Containment – contain device</th><th>UC</th></tr>
<tr><td colspan="2">Host that are compromised must be contained. This can be done by, e.g. placing the host in a so called quarantine network or disconnecting the host from the network. In the example below a laptop has been compromised and has to be contained. We asume that the ICT infrastructure contains a SDN or Network Access Control system that can change the network configuration such that an endpoint is place in a quarantine network.</td><td rowspan="2">UC1</td></tr>
<tr><th>Course of Action</th><th>OpenC2</th></tr>
<tr><td>Contain laptop4</td><td><pre>{
  "action": "contain",
  "target": {
      "device": {
          "hostname": "laptop4.exam-
ple.org"
      }
   }
}</pre></td></tr>
</table>

Table 3-2. CoA - OpenC2 example for patching software on host

<table>
<tr><th colspan="2">Example: Patch software on host</th><th>UC</th></tr>
<tr><td colspan="2">The COA Generator produces a set of defences that securiCAD uses. This includes, for example, patching a particular sotware application on a host. The COA Generator provides a reference “ref” that relates to an asset in the infrastructure model, e.g. the software application “application” and host FDQN “server1.example.org”.
Comment: the OpenC2 specification do not yet include an actuator profile for software update. It has been proposed as a use case for OpenC2[26]. We therefore asume that a software management system is the actuator responsible for updating software applications on endpoints within the organisations ICT infrastructure. The OpenC2 command is thus directed to the actuator “SoftwareManagementSystem”.</td><td rowspan="2">UC2
UC3
UC4
UC5</td></tr>
<tr><th>Course of Action from COA Generator</th><th>OpenC2</th></tr>
<tr><td><pre>{
   "efficency": 99123,
   "defenses": [
       {
          "ref": "112233",
          "defenseName": "Patched",
           "defenseInfo": "Patch ap-
plication",
           "mitreRef": "M1040"
       },
   ]
}</pre></td><td><pre>{
  "action": "update",
  "target": {
     "software": {
         "name": "application",
         “SWID”: “…”
     }
  },
  "actuator": {
     "SoftwareManagementSystem": {
        "hostname": "server1.exam-
ple.org"
     }
  }
}</pre></td></tr>
</table>

[26] https://github.com/oasis-tcs/openc2-usecases/blob/master/Symantec/endpoint_update_software.md and https://github.com/oasis-tcs/openc2-usecases/blob/master/sFractalConsulting/19.resp_update_sw.md

4 Web Front End

This section introduces the initial insights about the development of a Web Front End. The idea behind the Web Front End is to provide a unified control panel where all the aspects related to SOCCRATES can be managed (Figure 4-1). This includes aspects such as the visualization of the incidents processed, information related to them and their status. This information is directly taken from the Orchestrator database.

Additionally, the Web Front End will integrate the control panel of tools that are already available from other partners. This will allow the SOC admin to check more detailed information about some specific aspect of the platform. For instance, from the integrated Web Front End, the SOC admin will be able to check the ADG simulations for a given incident, and analyse in detail more information in order to, for instance, decide to advance on the incident response.

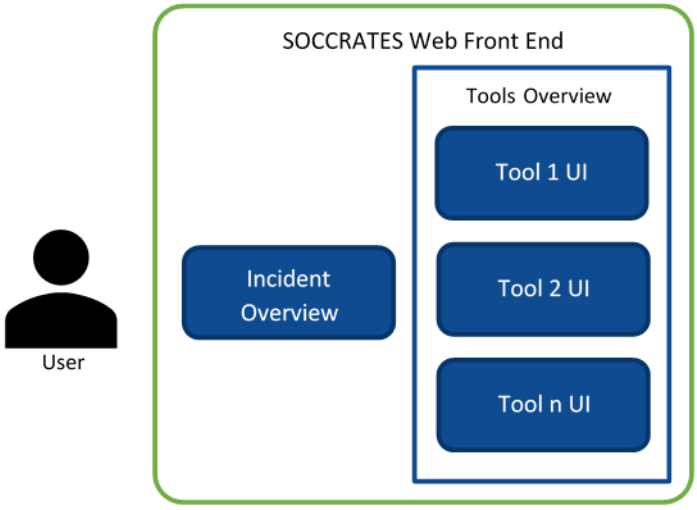

Figure 4-1. Concept for the Web Front-End

Most of the activities related to the development of the Web Front End will happen during the second half of the project (reported in deliverable D6.4 “Final version of the SOCCRATES platform”). This includes not just the integration of currently available user interfaces from partners’ tools, but also the development of a new control panel for the visualization of general information related to the infrastructure, the progress of the incident linked to the activities of the orchestrator and the framework that allows to access to the rest of the individual tools.

Figure 4-2 represents a mock-up of the Web Front End panel to be developed from scratch. This screen represents the concept behind the Web Front End, in which the incident being processed is displayed, showing information about the results of the analysis done so far and the phase within the incident lifecycle where it is now. From there, in the left panel, there are links to every individual tool in case the SOC operator needs more information about the incident, the platform or about some of the results already obtained.

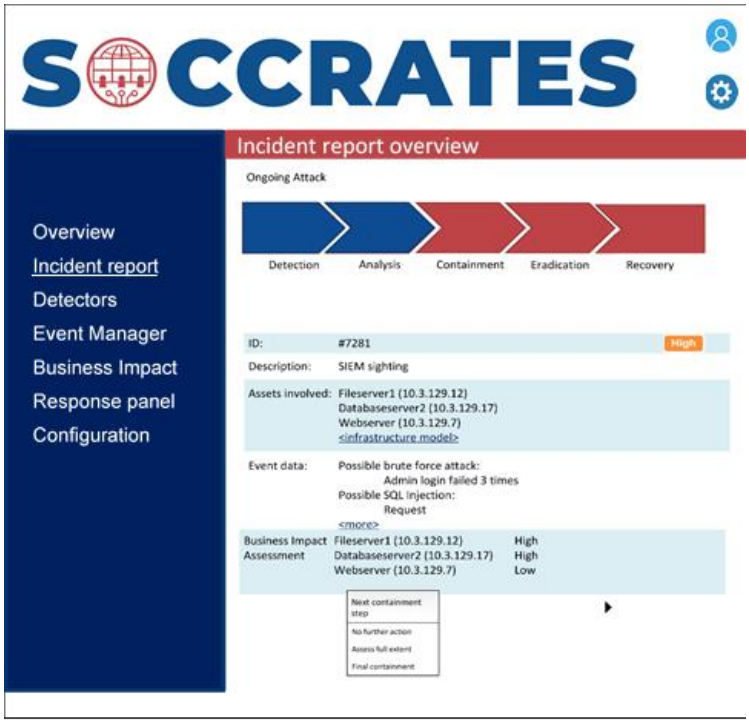

Figure 4-2. Mock-up of the Web Front End

The access to these partners tools will be seamless. This is precisely one of the biggest challenges to address, in order to provide with a seamless user experience. Several issues are required to be solved, not just related to a uniform look and feel, but, also and more important, related to the automatic access to every tool without the need to log in again. This is because all currently graphic interfaces from partners already have their own identity management solution, with their own database and their own authentication mechanisms. Therefore, in order to automate the access to those tools in an automatic and seamless way it is required to provide with a single sign solution. A potential candidate to solve this issue is the usage of a solution based on OAuth2, which will require an Authorization server as depicted in Figure 4-3.

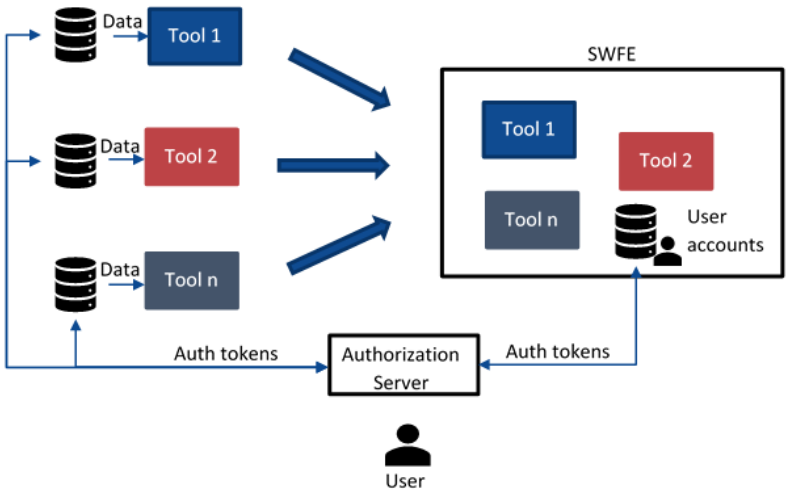

Figure 4-3. Approach for unifying identity management technologies at SOCCRATES tools GUIs

In this case, the tools rely on authentication tokens exchange between tools GUIs and the authorization server, which links the identities at the different tools to the tokens, allowing tools to know the identity of the user to be logged it.

5 Initial integration plan

This section describes an initial integration plan that has been designed as a roadmap for the integration activities that will happen in SOCCRATES during the following months. This section is composed of three subsections:

- The integration methodology that dictates the integration plan to be followed
- The Individual components functions, which define the functions that will be available at the Cortex analyzers and data structure defined both for the request and the response.
- A testbed that has been created for initial testing of some of the components of the SOCCRATES platform.

5.1 Integration methodology

SOCCRATES will follow a progressive approach for integrating the different components that will be part of the SOCCRATES platform. This approach has been designed with the intention of facilitating integration activities and optimizing the resources that are devoted to the integration. It is progressive because it is organized in several phases. The final objective of the integration plan is to have all tools and developed components running on the pilots' premises, considering the possibility that there might not be Internet connectivity due to security restrictions. In order to avoid continuous interaction with pilots, which increases the logistics to deploy everything on site from the beginning, in SOCCRATES the initial integration is carried out remotely while progressively certain components are moved on site. The following diagrams represents the paradigm to follow for the three phases of the SOCCRATES integration plan:

- Phase 1: remote integration. In this phase all SOCCRATES tools are running on every partner premises. These components will provide pubic interfaces that can be consumed by individual components. For example, the Orchestrator & Integration Engine will be running at Atos premises. In this scenario, the Cortex framework will interact remotely with the individual tools through their respective Cortex analyzers. In this scenario, however, there are two activities that will not be able to be done:
    - Trigger the workflow from pilots SIEMs. In this phase, if the infrastructure of any of the pilots is working in offline mode, that is, disconnected from the Internet, the SIEMs triggering the workflow for UC1 will not be interconnected to the public Orchestrator. In this case, the entry message to trigger UC1 workflow will be simulated following the data format and structure defined by UC1 providers. For the rest of the Use Cases, the workflow can be triggered remotely, for example, from the Threat Intelligence platform.
    - Provide real results from the infrastructure. Having components running remotely and disconnected from a pilot's premises, the data exchanged during the execution of the workflow will not be linked to real data from the infrastructure. To this end, the purpose of the phase is to ensure the correct interoperability between components, namely between the Orchestrator and the tools through Cortex analyzers.

  This phase is expected to be completed by M20. In this phase for tools interacting with the Orchestrator & Integration engine Cortex Analyzers will be built, which will be tested remotely with the instance of the Orchestrator available at ATOS.
- Phase 2: In this phase, some components will already be deployed at pilot premises, while others will still be running on partners premises. The formers can be security monitoring solutions such as AI based detectors, that will use data directly from the infrastructure to detect incidents in real time. The latter can include components such as the Orchestrator, which will

still be triggered remotely through the Internet. The main drawback of this approach that is not possible to be used for pilot premises with strict security restrictions that limit the connection to the public Internet. In such cases, Phase2 will be skipped, moving directly to Phase 3.
During this phase, incident detectors will be deployed on pilot premises (i.e., the AAD), which will report events to pilots SIEMs. Additionally, some components (such as ADG or IMC) can also be deployed on pilot premises in this phase, interacting remotely with the Orchestrator at ATOS through the Cortex analyzers implemented in Phase 1. This phase is expected to be completed by M26.

- Phase 3: In this phase, all components are already running on premises. The integration should be seamless if phase 1 has been carried out successfully. It will just be required to change endpoints IPs and ports. Detectors can read data directly from the pilot's network, detect incidents in real time and report them to the SIEM, which will be able to directly trigger the workflows of the Orchestrator.
  This phase, to be completed by M34, will include all tools running locally on pilots premises, including the Orchestrator. In this phase also the Web Front End will be integrated.

5.2 Individual components functions

Table 2-2 shows the SOCCRATES components that interact with the Orchestrator and Integration engine. This section details the information that is exchanged by these components with the Orchestrator and Integration Engine, either to trigger the Orchestrator (described in Section 5.2.1) or to participate in the workflow through analyzers and responders (Sections 5.2.2 and 5.2.3 respectively). It is worth noticing that the details here described correspond to a static view of the data specification that is subject to certain changes as long as the work continues in WP6.

5.2.1 Orchestrator triggering

This section details the information received by the Orchestrator and Integration Engine by every component, which includes the data required to trigger the appropriate workflow. This includes the SIEMs, IMT, and TIP.

5.2.1.1 SIEM solution triggering the Orchestrator

The detection of cyber security incidents carried out in UC1 relies on incident detectors that monitor the infrastructure of the organization being protected. These detectors can include NIDS, HIDS or AI-based incident detectors like that AAD developed in WP4. In all cases, these detectors send security events to the SIEM solution(s) of the respective organizations (or pilots in the case of SOCCRATES), which, according to an internal logic that is typically based on correlation rules, trigger security alerts containing information about the incident detected. The information included in these security alerts are used by the SOCCRATES Orchestrator and Integration engine to trigger the UC1 workflow.

Table 5-1. Details of SIEMs triggering the orchestrator

Component triggering	Data included	UC
SIEM from VTF	**Time** string Timestamp when the incident has been detected **Title** string Description of the incident detected **Category** string The type of incident that has been detected	UC1

	Urgency string Importance of the incident detected established by the SIEM. **Source** string (when available) IP or fully qualified domain name of the host that has caused the incident. **Destination** string (when available) IP of fully qualified domain name of the host targeted by the attack.	

SIEM from MNM	MNM's Argus platform does not support push notifications on new cases (incidents) by default. In order to trigger the orchestrator, a worker that polls for new cases could be implemented. Alternatively, mnemonic could develop triggers that call the orchestrator API when a new case is created. Each case (incident) can contain one or more events (alerts). The events contain relevant indicators. MNM's SIEM allows to retrieve information from a specific API, both for incidents[27] and alerts[28]. These APIs are using swagger[29].
	
	Common fields for all alerts:
	
	Signature string
	
	Timestamp string
	ISO 8601 date and time
	
	Severity string
	Possible values: "low", "medium", "high", "critical"
	
	Source IP address dotted decimal notation
	
	Source port integer
	
	Destination IP address dotted decimal notation
	
	Destination port integer
	
	Attack category integer
	
	Attack category description string
	
	The following JSON message represents an example of a security alerts generated by the MNM's SIEM:
	`{`
	`"responseCode": 200,`
	`"limit": 0,`
	`"offset": 0,`
	`"count": 0,`
	`"metaData": {},`
		UC1

[27] https://docs.mnemonic.no/display/public/API/Case+Integration+Guide
[28] https://docs.mnemonic.no/display/public/API/Event+Integration+Guide
[29] https://portal.mnemonic.no/spa/swagger?module=events

```
  "messages": [],
  "data": {
    "customerInfo": {
      "id": 2171,
      "shortName": "demo",
      "name": "Demo"
    },
    "properties": {
      "http_protocol": "HTTP/1.1",
      "event.original.signature": "SNORT-1:41336",
      "agent.parseTimestamp": "1569185281851",
      ...
    },
    "associatedCase": null,
    "location": {
      "shortName": null,
      "name": null,
      "timeZone": "Europe/Oslo",
      "id": 0
    },
    "attackInfo": {
      "alarmID": 58129,
      "alarmDescription": "MALWARE - Andr.Trojan.Sysch variant
activity",
      "attackCategoryID": 363,
      "attackCategoryName": "Callback traffic (Check-in) (in-
fected client/server)",
      "signature": "SNORT-1:41336"
    },
    "domain": null,
    "uri": null,
    "count": 1,
    "source": {
      "port": 12342,
      "geoLocation": null,
      "networkAddress": {
        "host": true,
        "ipv6": false,
        "maskBits": 32,
        "multicast": false,
        "public": false,
        "address": "10.5.97.201"
      }
    },
    "destination": {
      "port": 80,
      "geoLocation": {
        "countryCode": "NO",
        "countryName": null,
        "locationName": "",
        "latitude": 52.3824,
        "longitude": 4.8995
      },
      "networkAddress": {
        "host": true,
        "ipv6": false,
        "maskBits": 32,
        "multicast": false,
        "public": true,
        "address": "94.127.56.1"
      }
    },
    "protocol": "0",
    "timestamp": 1569185283911,
    "startTimestamp": 1569185283910,
    "endTimestamp": 1569185283910,
    "lastUpdatedTimestamp": 1569186046339,
    "flags": [
```

```
        "BLOCKED",
        "CREATED_BY_ANALYSIS_FILTER",
        "FINALIZED",
        "CUSTOM_SOURCE_AGGREGATION",
        "CUSTOM_DESTINATION_AGGREGATION",
        "SOURCE_IS_CUSTOMERNET"
      ],
      "severity": "high",
      "detailedEventIDS": [],
      "id": "AGGR/1569185283911/2171/6c36deb6-d5e3-461c-bab4-
fca397006cb9"
    },
    "size": 0
}
```


5.2.1.2 Threat Intelligence Platform triggering the Orchestrator

The threat intelligence platform continuously collects, enriches and analyzes threat data and informations. Rules in the TIP can be configured to trigger the orchestrator when a notable event is detected. The rules are being developed in WP4, and the types and values are subject to change.

Table 5-2. Details of TIP triggering the Orchestrator

Component triggering	Data included	UC
Threat Intelligence Platform, (new threat is noticed)	**Type** string Type of CTI. Example values: vulnerability, IoC, technique **Value** string Example values corresponding to the types above: CVE identifier, fqdn/hash/IP address, MITRE ATT&CK technique name Supported data types can be retrieved dynamically from the ACT REST API endpoints: GET /v1/objectType The endpoints are documented here: https://act-eu1.mnemonic.no/swagger/	UC2

5.2.1.3 Infrastructure Modelling Component triggering the Orchestrator

The infrastructure modelling component continuously monitors the infrastructure and records any changes. Rules in the IMC can be configured to trigger the Orchestrator when a new vulnerability is discovered (UC3), when a new configuration is detected (UC4), and when a new system is detected (UC5).

Table 5-3. Details of IMC triggering the orchestrator

Component triggering	Data included	UC
Infrastructure Modelling (new vulnerabilities)	**Host** string The fully qualified domain name of the system running the vulnerable software, e.g. computer01.soccrates.eu **Software** string The software impacted by the vulnerability, represented as a Common Platform Enumeration (CPE) identifier	UC3

	Vulnerability string The vulnerability detected, represented as a Common Vulnerabilities and Exposures (CVE) identifierThe vulnerability detected, represented as a CVE identifier	
Infrastructure Modelling (new configuration is detected)	**Host** string The fully qualified domain name of the system with a new configuration, e.g. computer01.soccrates.eu **Type** string route (routing) OR rule (firewall) **Value** string For route: interface:ipv4 address nextHop:ipv4 address metric:integer routesTo: CIDR For rule: sourceNet:CIDR OR “any” destinationNet:CIDR OR “any” action:”allow” OR “deny” protocol (optional):”tcp” OR “udp” OR “icmp” lowSourcePort (optional):integer highSourcePort (optional):integer lowDestinationPort (optional):integer highDestinationPort (optional):integer protocolState (optional):”established” OR “any”	UC4
Infrastructure Modelling (new system is detected)	**Host** string The fully qualified domain name of the new system, e.g. computer01.soccrates.eu **IP address** string The IPv4 address of the new system, e.g. 10.0.0.1	UC5

5.2.2 SOCCRATES Analyzers

5.2.2.1 Infrastructure Modelling Component

This section describes the information exchanged by the IMC analyzer with the Orchestrator and Integration Engine, detailing the functions that will be implemented in the corresponding Cortex analyzer.

Table 5-4. Details of the Cortex analyzer for the IMC

Infrastructure Modelling Component	
Function in the cortex analyzer	infrastructureModel
Request Data	
before ISO 8601 date and time The response data will contain all information that was added to the IMC before (earlier than) this timestamp.	

after ISO 8601 date and time The response data will contain all information that was added to the IMC after (later than) this timestamp. To retrieve infrastructure data for a time frame, both parameters must be defined. The response data will then contain all infrastructure data that was added to the IMC within this time frame. What time frame to use depends on the frequency of adapter updates to the IMC: if adapters run once per day then a 24-hour time interval is appropriate.
Response Data
infrastructureModel array An array of triplets ([source object, relation, destination object] for relations and [source object, property, null] for properties) representing the elements of the infrastructure model for the time frame specified in the request data. The triplets represent a graph of the infrastructure. The IMC meta-model is under development, so it is currently not possible to specify the exact data types contained in this array. The data types can be retrieved dynamically from the IMC REST API end-points: GET /v1/factType GET /v1/objectType The endpoints are documented here: https://act-eu1.mnemonic.no/swagger/

5.2.2.2 Threat Intelligence Platform

This section describes the information exchanged by the TIP analyzer with the Orchestrator and Integration Engine, detailing the functions that will be implemented in the corresponding Cortex analyzer.

Table 5-5. Details of the Cortex analyzer for the TIP

Threat Intelligence Platform	
Function in the cortex analyzer	emulationPlan
Request Data	
threatActor string The name of the threat actor for which to retrieve an adversary emulation plan (case sensitive).	
Response Data	
results array An array of attack steps with pre- and post-conditions. Each element of the array will contain a JSON object containing the MITRE ATT&CK ID, a string with the MITRE ATT&CK technique name, and array of strings with the MITRE ATT&CK tactic name(s), an array of strings with pre-conditions and an array of strings with post-conditions. Example attack step (element in the results array). Note that this is work in progress and might change:	

```
    "T1005": {
        "name": "Data From Local System",
        "provides": [
            "target_information"
        ],
        "requires": [
            "access_host",
            "execute_code"
        ],
        "tactic": [
            "Collection"
        ]
    }
```

5.2.2.3 ADG Analyzer

This section describes the information exchanged by the ADG analyzer with the Orchestrator and Integration Engine, detailing the functions that will be implemented in the corresponding Cortex analyzer.

Table 5-6. Details of the Cortex analyzer for the ADG Analyzer

ADG Analyzer	
Function in the cortex analyzer	simulation
Request Data	
date (optional) string The date from which securiCAD will fetch a snapshot from the IMC. Default is latest. Example: "2020-10-20" **threatActor** (optional) string The name or alias of the threat actor from the TIP to simulate with. The ADG will fetch the necessary data to customize the simulation configuration. Default is a generic attacker. **entryPoints** (optional) array An array of objects representing attacker entry points given current e.g. IOCs, with a reference "ref" to an asset ID in the IMC and type of attack. Default is external attacker. Example: `[ { "ref": "112233", "type": "Phishing" } ]`	
Response Data	
simulationId string A unique ID for the simulation Example: "123456789123" **reportUrl** string URL to the report in securiCAD Example: "https://10.0.0.2/org/project/123456789123"	

results array
An array of objects with a reference "ref" to an asset ID in the IMC and the TTC and probability of C/I/A compromise.
Example:

```
[
    {
        "ref": "112233"
        "confidentiality": {
            "ttc": 1,
            "probability": 1.0
        },
        "integrity": {
            "ttc": 10,
            "probability": 0.5
        },
        "availability": {
            "ttc": 3,
            "probability": 0.8
        }
    }
]
```

resultsDelta array
An array of objects with a reference "ref" to an asset ID in the IMC and the delta TTC and probability of C/I/A compromise compared to the previous simulation.
Example:

```
[
    {
        "ref": "112233"
        "confidentiality": {
            "ttc": -5,
            "probability": 0
        },
        "integrity": {
            "ttc": 0,
            "probability": -0.3
        },
        "availability": {
            "ttc": 1,
            "probability": 0
        }
    }
]
```

5.2.2.4 CoA Generator

This section describes the information exchanged by the CoA Generator with the Orchestrator and Integration Engine, detailing the functions that will be implemented in the corresponding Cortex analyzer.

Table 5-7. Details of the Cortex analyzer for the CoA Generator

CoA Generator	
Function in the cortex analyzer	coa_strategy
Request Data	
ip_address. String Valid IP address of a machine on which securiCAD is installed. `{"ip_address": "value",` **username**. String	

A name of an existing user of securiCAD. It is preferred that the user has admin rights in securiCAD, so that intermediate results of simulations can be deleted on the fly.

```
"username": "value",
```

password. String
Password corresponding to the username. Format: String.

```
"password": "value",
```

organization. (optional) String
Users in securiCAD can be assigned organizations. If an organization is assigned to the user, provide it here. If not, you can omit this field or provide any value here. Format: String.

```
"organization": "value",
```

securiCAD's model id of the model for which CoAs are to be generated. Format: String.

```
"mid": "value",
```

pid. String
securiCAD's project id. The model whose id is provided in the previous field has to be a part of the project whose id is provided in this field. Format: String.

```
"pid": "value",
```

assets. String
List of securiCAD's ids of objects. The goal of the generated CoAs will be to increase the TTC value for these objects (in securiCAD's terms: to increase the TTC value of the 'compromise' steps on these objects). Format: List of strings.

```
"assets": ["value1",
          ...
                  "valuen"],
```

monetary_budgets. (optional) Dictionary.
Values specifying available (monetary) resources, e.g., budget for capital expenditures and budget for operating expenditures. It is a dictionary with keys being consecutive numbers starting with 1, as strings, and with values being floats.

```
"monetary_budgets": {
         "1": value,
            …
         "n": value
}
```

time_budget. (optional) Dictionary
Float values specifying available (time-like) resources. In practice, one value is expected, expressing a limit on time available for implementation of a CoA. It is a dictionary with keys being consecutive numbers starting with 1, as strings, and with values being floats.

```
"time_budgets":  {
         "1": value,
            …
         "n": value
}
```

monetary_costs. (optional) Base64 String

```
"monetary_costs": "value",
```

(optional) time**_costs**. Base64 String

```
"time_costs": "value",
```

defense_step. (optional) Dictionary
A defense step defined as a string of the form objectId.defenseName, where objectId is a securiCAD's id of an object present in the model whose id has been provided in the request data (under "mid"), and defenseName is a securiCAD's name of a defense step that can be implemented on the object. It is represented as a dictionary with keys being defense steps (strings) and values being lists of defense steps (lists of strings).

```
"defense_step": [
            "defense_step",
              …
            "defense_step"
  ]
}
```

prerequisites. (optional) Dictionary
All of the defense steps in prerequisites["def_step"] need to be implemented in the system before "def_step" itself can be implemented. It is a dictionary with keys being defense steps (strings) and values being lists of defense steps (lists of strings).

```
"prerequisites": {
        "defense_step": [
            "defense_step",
              …
            "defense_step"
  ]
```

exclusivity. (optional) Dictionary
The defense steps "def_step" cannot be implemented in the system in parallel with any of the defense steps defined in the exclusivity["def_step"]. It is a dictionary with keys being defense steps (strings) and values being lists of defense steps (lists of strings).

```
"exclusivity":  {
        "defense_step": [
            "defense_step",
              …
            "defense_step"
  ]
```

Response Data

coas. array
List of course of actions (coa). Each coa contains a value for quantifying the quality of the coa "efficency" and a list of "defenses" to be applied. Each defense has a reference "ref" to an asset ID in the IMC, a defense in securiCAD "defenseName", a description of the defense "defenseInfo" and a reference to a mitigation in mitre attack "mitreRef".
Example:

```
[
    {
        "efficency": 123,
        "defenses": [
            {
                "ref": "112233",
                "defenseName": "Patched",
                "defenseInfo": "Patch application",
                "mitreRef": "M1040"
```

```
            },
            {
                "ref": "112233",
                "defenseName": "Patched",
                "defenseInfo": "Patch application",
                "mitreRef": "M1040"
            }
        ]
    }
]
```

5.2.2.5 Impact Analyzer

This section describes the information exchanged by the Impact analyzer with the Orchestrator and Integration Engine, detailing the functions that will be implemented in the corresponding Cortex analyzer.

Table 5-8. Details of the Cortex analyzer for the Impact Analyzer

Business Impact Analyzer	
Function in the cortex analyzer	*bia_analysis*
Request Data	

Impacted assets Array

In order to compute the business impact, it is necessary to know which assets are impacted, based on the output of the ADG analyser. For each impacted asset, it is also necessary to know the level of compromise in terms of confidentiality, integrity and availability, based on the output of the ADG analyzer.

Example in the case of a single asset:

```
[
    {
        "assetID": "112233"
        "confidentiality": {
            "ttc": 1,
            "probability": 1.0
        },
        "integrity": {
            "ttc": 10,
            "probability": 0.5
        },
        "availability": {
            "ttc": 3,
            "probability": 0.8
        }
    }
]
```

Business function/process ID (optional) Array of strings

The business impact can be computed on any existing business function or process. If no input is provided, the business impact will be computed for the highest level business process.

```
business id: ["value_1",..., "value_n"]
```

Attack time (optional)

The attack time can be useful for the computation of the impact of attacks that may occur at a certain time, possibly different from the current time. If no input is provided, the computation will be performed for the current time.

```
"attack time": value
```

Response Data

Business impacts Array

The business impact computed for each business function/process provided in the input. The business impact is expressed through three values: the impact probability, impact criticality (value between 0 and 10), and critical time expressed in hours. The impact probability represents the probability that a business process or function cannot work as intended, the impact criticality represents the importance of the business process / function for the company, and the critical time represents the time under which the business process / function will suffer no significant impact.

Example:

```
[
    {
        "business id": "43"
        "impact probability”: 0.67
        "impact criticality": 6
        "critical time": 2.4
    }
]
```

Function in the cortex analyzer	*bia_fetchBusiness*

Request Data

None

This function fetches the list of business functions and processes of the current model, therefore no input data are required.

Response Data

Business items Array

The output is an array of all the business functions and processes, with their identifier (String), name (String), level in the tree (Integer), and criticality (Float). Level 1 represents the level of the most vital business process.

Example:

```
[
    {
        "business id": "43",
        "business name": "Procurement"
        "business level": "2"
        "business criticality": "7.3"
    }
]
```

Function in the cortex analyzer	*bia_fetchCriticalAssets*

Request Data

Business function/process ID (optional) Array of strings

This function fetches the list of critical assets of the current model. Specifying one or more business function / process with their corresponding Business ID will limit the list of critical assets to the ones supporting these specific business functions / processes.

```
business id: ["value_1",..., "value_n"]
```

Response Data

Critical assets Array of strings
The output is the list of critical assets, i.e. assets that directly support business functions. The assets are identified by the IMC identifier.

```
Critical assets: ["value_1",..., "value_n"]
```

5.2.2.6 Response Planner

This section describes the information exchanged by the Response Planner with the Orchestrator and Integration Engine, detailing the functions that will be implemented in the corresponding Cortex analyzer.

Table 5-9. Details of the Cortex analyzer for the Response Planner

Response Planner	
Function in the cortex analyzer	coa_ranking
Request Data	

aiv Float
The AIV is the Annual Infrastructure Value, which can be estimated based on the costs of equipment, personnel and services. The AIV is specific to the whole infrastructure and does not depend on the attack or CoA.

```
"aiv": value
```

ale Float
The ALE is the Annual Loss Expectancy, which represents the losses for the company if no CoA is deployed to mitigate the attack. The ALE is specific to the attack but does not depend on the CoA.

```
"ale": value
```

coas Array
The RORI will be computed for each of the CoAs received in the input. To do so, on top of the description of the CoA obtained from the CoA generator, the following parameters must be provided:
- the Annual Response Cost (ARC, representing the CoA deployment cost)
- the Surface Coverage (SC, representing the percentage of the surface covered by the CoA).

These two values are of type Float.
Example:

```
[
    {
```

```
        "arc": 650,
        "sc": 46,
        "efficency": 123,
        "defenses": [
            {
                "ref": "112233",
                "defenseName": "Patched",
                "defenseInfo": "Patch application",
                "mitreRef": "M1040"
            },
            {
                "ref": "112233",
                "defenseName": "Patched",
                "defenseInfo": "Patch application",
                "mitreRef": "M1040"
            }
        ]
    }
]
```


Response Data

Ranked coas Array

The output is an array of CoAs of the same length as the one provided as input, providing the description of the CoAs as well as their RORI value, ranked by highest RORI value. RORI is of type Float.

Example with one evaluated CoA:


```
[
    {
        "RORI": 76,
        "defenses": [
            {
                "ref": "112233",
                "defenseName": "Patched",
                "defenseInfo": "Patch application",
                "mitreRef": "M1040"
            },
            {
                "ref": "112233",
                "defenseName": "Patched",
                "defenseInfo": "Patch application",
                "mitreRef": "M1040"
            }
        ]
    }
]
```


5.2.3 SOCCRATES Responders

The SOCCRATES Orchestrator and Integration engine allows to interact with external components through Cortex analyzers, invoking functions and retrieving a reponse. However, Cortex is also compatible with Responders, which trigger actions in external components without requiring any specific response. These responders are used by the AAD component, which triggers additional detection capabilities whenever new vulnerabilities are reported to the Orchestrator and Integration Engine. Next subsection details this case.

5.2.3.1 AAD responder

The AI-based Attack Detection (AAD) component is foreseen to receive and use the context information about the current system state to improve its detection capabilities. The AAD can benefit from the context information in two ways:

- First, the knowledge about a new asset in the network (infrastructure change) can help to verify anomalies caused by that change, i.e. help to distinguish whether the root cause of anomalies was malicious or benign. Consequently, a baseline detection model can be updated in response to the presence of a new asset.
- Second, the information about the detected host vulnerabilities can be used to increase confidence of events (anomalies) related to that host and certain types of attack. Given that this information is available in the infrastructure modelling component, an event with appropriate information (as listed in Table 5-10) could be used to improve AAD detection capabilities.

Table 5-10. Details of the Cortex responder for the AAD

AAD	
Function in the cortex responder	Infrastructure Modelling (new vulnerabilities)
Request Data	
{host, software (CPE), vulnerability (CVE)} Check Table 5-3. Details of IMC triggering the orchestrator	
Response Data	
Result OK/error code	
Function in the cortex responder	Infrastructure Modelling (new system is detected)
Request Data	
{host, IP address} Check Table 5-3. Details of IMC triggering the orchestrator	
Response Data	
Result OK/error code	

5.3 Testbed

A testbed has been created for some tools to be tested. This testbed has been deployed at Atos premises and simulates a real network topology using virtual machines and routers. The complete diagram is depicted in Figure 5-1.

The testbed consists of three subnets: VLAN1, VLAN2 and VLAN3. Every VLAN is connected to a router. These routers are at the same interconnected, forwarding appropriately the traffic to the rest of the VLANs. In every LAN one or more machines are running. The lines of the figure in blue represents the VLANs, specifying the IP of every machine and the name of the interface that is connected to it.

In red it is shown the links between routers. Every router has three network interfaces that are part of the testbed: two to every other router and another one towards their respective VLAN. These routers

are virtual routers based on Quagga[30] running on Ubuntu machines. Quagga includes services for defining network interfaces (Zebra service) and for providing different network routing protocols such as OSPF, BGP, etc. For this testbed Zebra and the OSPF service has been used.

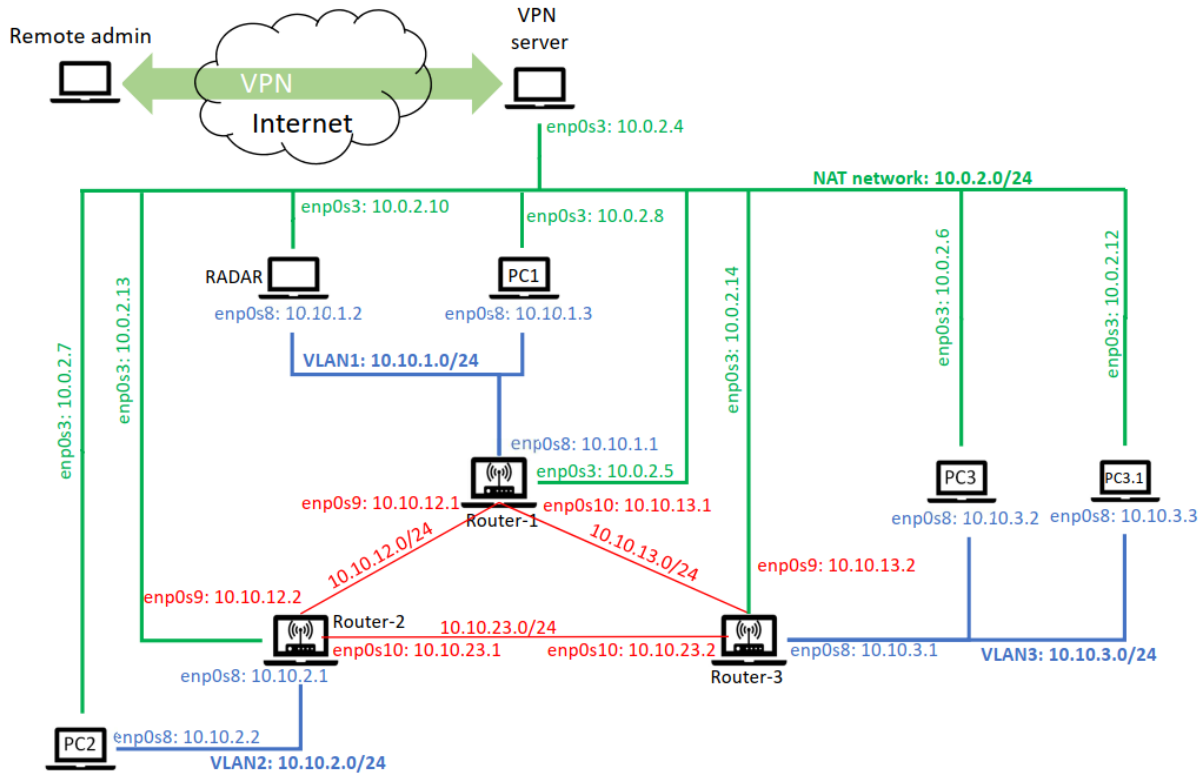

Figure 5-1. Network topology of the testbed

The configuration of Zebra allows to define the IP address of every interface. For example, Figure 5-2 represents the configuration for Router1:

```
interface enp0s8
 ip address 10.10.1.1/24
 ipv6 nd suppress-ra
interface enp0s9
 ip address 10.10.12.1/24
 ipv6 nd suppress-ra
interface enp0s10
 ip address 10.10.13.1/24
 ipv6 nd suppress-ra
interface lo
ip forwarding
```

Figure 5-2. Zebra configuration for the Router1 of the testbed

For the configuration of the OSPF service, it is required to specify the address of the subnets that are connected to this router, as well as the interfaces available. This is depicted in Figure 5-3 for the Router 1:

[30] https://www.quagga.net/

```
!router ospf
!  network 192.168.1.0/24 area 0
!
log stdout
interface enp0s8
interface enp0s9
interface enp0s10
interface lo

router ospf
!    ospf router-id 10.10.1.1
network 10.10.1.0/24 area 0
network 10.10.12.0/24 area 0
network 10.10.13.0/24 area 0
log file /var/log/quagga/ospfd.log
line vty
```

Figure 5-3. OSPF service configuration for Router 1

Overlapped to this network there is an administration network, shown in green in Figure 5-1, that is configured with Internet connectivity and accessible by using a VPN server running on another virtual machine.

Using this testbed, several tools from some partners have been able to be tested. This is the case of the RADAR tool by F-Secure, which scans the network, discovers assets and evaluates the vulnerability of the services running in those assets. The RADAR tool was used in the testbed as a way to demonstrate initial vulnerability discovery that will be used in the pilots. An example is shown in Figure 5-4, where a critical vulnerability inserted in PC1 has been discovered by RADAR.

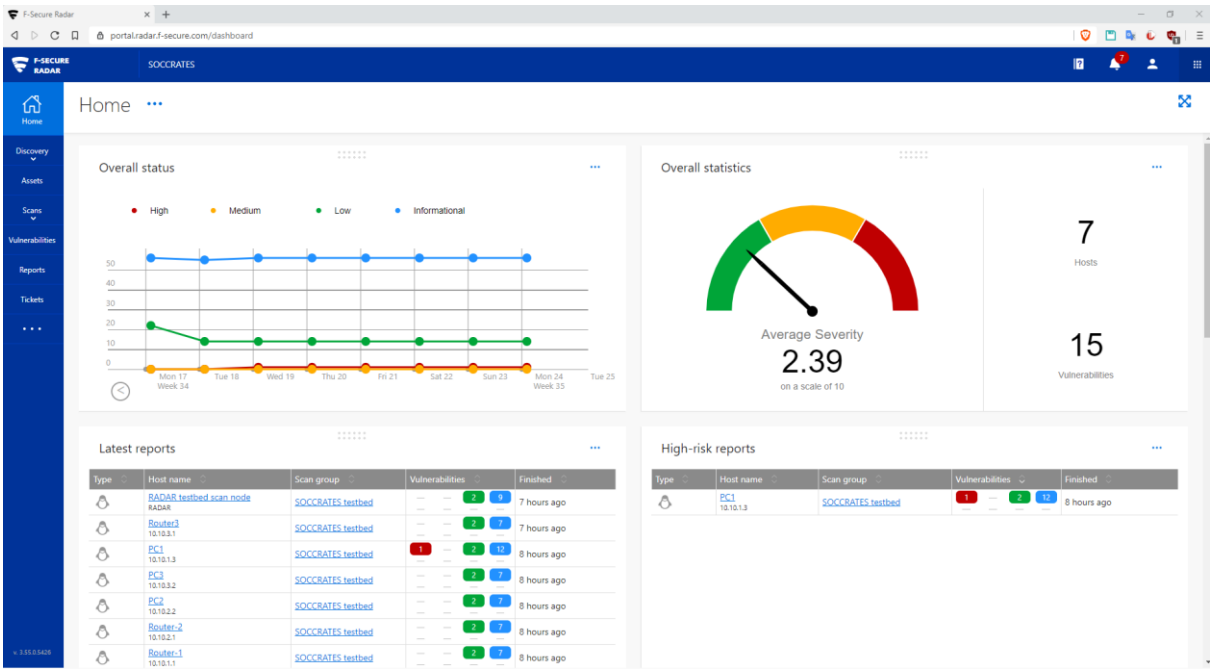

Figure 5-4. F-Secure's RADAR app running in the testbed

This testbed has been used to change the routing configuration or adding access control lists that has allowed to simulate several scenarios that can appear during a real operation environment.

6 Conclusions

This deliverable has presented the initial results of task T6.1, describing the initial results related to the development of the SOCCRATES Orchestrator & Integration Engine, reconfiguration and Web Front End. More specifically, most of the progress carried out during the first months of this task has been focused on the development of the Orchestrator & Integration Engine. The outcomes on this aspect are focused on the study of technologies to use for the Orchestrator. This has resulted on the decision to combine two main technologies: a workflow execution engine based on Activiti and a framework for aggregating the access to the different SOCCRATES tools based on Cortex. To this end, with Activiti, different workflows can be created using a graphical editor. These workflows implement the logic of the five SOCCRATEs use cases, combining elements that allow to interact with external tools, analysts, provide with conditional branches that modify the execution flow, etc. Cortex has been chosen to prevent the Orchestrator & Integration Engine to deal with a myriad of different technologies for interacting with external tools. A set of Cortex analyzers will deal with the complexity of access to external tools, while the Orchestrator & Integration Engine will trigger requests to these tools by using a simple REST-API offered with Cortex. This not only simplifies the orchestrator, but also increase flexibility to incorporate additional tools just by providing with the corresponding Cortex analyzer.

An initial deployment has also been carried out, which has been used to successfully validate the approach by implementing a proof of concept that combines an external tool for triggering the workflow, the interaction with Cortex from the workflow engine and the interaction with external tools through a Cortex analyzer.

The other two pillars of the task correspond to the development of an integrated Web Front End and the development of mechanisms to respond to incidents by reconfiguring the infrastructure. For the former we have analysed the requirements for the Web Front End and described the approach to follow for its development, which will happen during the second half of the project. Similarly, the reconfiguration capabilities will also be developed during the second half of the project, although an initial approach has been presented, which relies on the usage of Cortex responders to interconnect to external components in charge of triggering actions to respond to managed incidents.

Finally, this document also introduces the integration plan, which will guide the combination of the different tools with the SOCCRATES Orchestrator. The details for integrating partners' tools with the Orchestrator by using Cortex analyzers are also included, describing the data to be handled by the Cortex analyzers when invoking external tools and the functions available.

6.1 Next steps

Current deliverable has reported the results obtained during the first 9 months of WP6. The following months will be focused on the followng actions:

- Development of Cortex analyzers and responders for the SOCCRATES tools. It is expected that these analyzers are ready by M20.
- Development of Activiti workflows integrating the Cortex analyzers, which is expected to be completed by M26

Both activities will be reported in D6.2 ("Initial version of the SOCCRATES platform"), scheduled for M27. By M27 the Orchestrator & Integration engine will be ready for testing and validation that will be reported in D6.3 ("SOCCRATES Testing and Validation Report"), to be delivered in M29.

During the last 12 months of the project it is expected to integrate the different tools GUIs into the SOCCRATES Web Front end, and finish with the incorporation of the Orchestrator & Integration Engine in the different pilots. This will be reported in D6.4 (“Final version of the SOCCRATES Platform”), in M34.

7 References

[D2.1] SOCCRATES deliverable 2.1. “Use cases definition & pilot sites requirements”. Dec 2019.

[D2.2] SOCCRATES deliverable 2.2. “System Architecture & Interface Specifications (Initial). May 2020.

[Gartner2020] Claudio Neiva, Craig Lawson , Toby Bussa , Gorka Sadowski. “Market Guide for Security Orchestration, Automation and Response Solutions”. September 2020 Available online at https://www.gartner.com/en/documents/3990720/market-guide-for-security-orchestration-automation-and-r

8 Abbreviations

This glossary serves as inventory of abbreviations used in the document.

This is a standard glossary, used for all SOCCRATES report deliverables; it will be expanded when necessary.

Acronym	Description
AAD	AI based Attack Detection
ACT	semi-Automated Cyber Threat intelligence
ADG	Attack Defence Graph
AEF	Argus Event Format
AI	Artificial Intelligence
AIT	AIT Austrian Institute of technology
API	Application Programming Interface
APT	Advance Persistent Threat
ATOS	ATOS Spain
AV	AntiVirus
BGP	Border Gateway Protocol
BPM	Business Process Model
BPMN	Business Process Model and Notation
CC	Command and Control
CERT	Computer Emergency Response Team
CMDB	Configuration Management Database
CSIRT	Computer Security Incident Response Team
CoA	Course of Action
CTI	Cyber Threat Intelligence
DC	DataCentre
DGA	Domain Generated Algorithm
DNS	Domain Name System
EDR	Endpoint Detection and Response
ELK	Elasticsearch/Logstash/Kibana
FRS	Foreseeti
FSC	F-secure
ICT	Information and Communication Technology
IDS	Intrusion Detection System
IMC	Infrastructure Modelling Component
IMT	Institut Mines Télécom – Télécom SudParis
INTF	Interface
IoC	Indicators of Compromise
IP	Internet Protocol
IPS	Intrusion Prevention System
IRM	Incident Response and Management
ITIL	Information Technology Infrastructure Library
KTH	Kungliga Tekniska högskolan - Royal Institute of Technology
LAN	Local Area Network
LDAP	Lightweight Directory Access Protocol
M_n	Infrastructure Model (at time *n*)

ML	Machine Learning
MNM	mnemonic
MSSP	Managed Security Service Provider
MTTD	Mean Time To Detection
NOC	Network Operations Centre
OT	Operational Technology
OS	Operating System
OSPF	Open Shortest Path First
PAP	Permissible Action Protocol
RORI	Return on Response Investment
SDN	Software Defined Network
SHS	Shadowserver
SIEM	Security information and event management
SIRP	Security Incident Response Plan
SOA	Security Orchestration and Automation
SOAR	Security Orchestration, Automation and Response
SOC	Security Operation Centre
SOCCRATES	SOC & CSIRT Response to Attacks & Threats based on attack defence graph Evaluation Systems
SSL	Secure Sockets Layer
TAP	Test Access Point
TIP	Threat Intelligence platform
TLP	Traffic Light Protocol
TLS	Transport Layer Security
TNO	Nederlandse Organisatie voor toegepast natuurwetenschappelijk onderzoek
TTC	Time To Compromise
UC	Use Case
VLAN	Virtual LAN
VM	Virtual Machine
VTF	Vattenfall