



A coordinated framework for cyber resilient supply chain systems over complex ICT infrastructures

D4.3 Security and Certification Manager components design and implementation (IT-2)

Document Identification			
Status	Final	Due Date	31/10/2022
Version	1.0	Submission Date	16/11/2022

Related WP	WP4	Document Reference	D4.3
Related Deliverable(s)	D3.1, D3.2, D4.1, D4.2	Dissemination Level (*)	PU
Lead Participant	STS	Lead Author	Grigorios Kalogiannis
Contributors	POLITO	Reviewers	ATOS
			UMINHO

Keywords:

EDC, SACM, SCM, integration, policy, assurance, certification, IT-2

This document is issued within the frame and for the purpose of the FISHY project. This project has received funding from the European Union's Horizon 2020 research and innovation programme under Grant Agreement No. 952644. The opinions expressed and arguments employed herein do not necessarily reflect the official views of the European Commission.

This document and its content are the property of the FISHY Consortium. All rights relevant to this document are determined by the applicable laws. Access to this document does not grant any right or license on the document or its contents. This document or its contents are not to be used or treated in any manner inconsistent with the rights or interests of the FISHY Consortium or the Partners detriment and are not to be disclosed externally without prior written consent from the FISHY Partners.

Each FISHY Partner may use this document in conformity with the FISHY Consortium Grant Agreement provisions.

(*) Dissemination level: PU: Public, fully open, e.g. web; CO: Confidential, restricted under conditions set out in Model Grant Agreement; CI: Classified, Int = Internal Working Document, information as referred to in Commission Decision 2001/844/EC.

Document Information

List of Contributors	
Name	Partner
Antonio Álvarez	ATOS
Cataldo Basile	POLITO
Grigoris Kalogiannis	STS
Kostas Poullos	STS

Document History			
Version	Date	Change editors	Changes
0.1	26/08/2022	Grigoris Kalogiannis (STS)	Table of Contents/work allocation.
0.2	16/09/2022	Grigoris Kalogianis (STS), Cataldo Basile (POLITO), Antonio Álvarez (ATOS)	Individual contributions provided and merged
0.3	30/09/2022	Grigoris Kalogianis (STS), Cataldo Basile (POLITO), Antonio Álvarez (ATOS)	Content added to sections 2-4, internal review of the technical parts
0.4	16/10/2022	Grigoris Kalogiannis (STS), Cataldo Basile (POLITO)	Technical contents stable and revised
0.5	24/10/2022	Antonio Álvarez (Atos)	Input to sections 1.1, 1.2, and 1.3, internal review
0.5.1	27/10/2022	Grigoris Kalogiannis (STS), Cataldo Basile (POLITO)	Merging and polishing
0.6	11/11/2022	Grigoris Kalogiannis (STS)	Added executive summary
0.7	14/11/2022	Antonio Álvarez, Juan Alonso (Atos)	Quality assessment, pending references in section 4.1.1.3
0.8	15/11/2022	Grigoris Kalogiannis (STS)	Section 4.1.1.3 updated
1.0	16/11/2022	Antonio Álvarez, Juan Alonso (Atos)	Quality assessment and final version to be submitted.

Document name:	Security and Certification Manager components design and implementation (IT-2)			Page:	2 of 48
Reference:	D4.3	Dissemination:	PU	Version:	1.0
				Status:	Final

Quality Control		
Role	Who (Partner short name)	Approval Date
Deliverable leader	Kalogiannis Grigorios (STS)	15/11/2022
Quality manager	Juan Andrés Alonso (ATOS)	16/11/2022
Project Coordinator	Antonio Álvarez (ATOS)	16/11/2022

Document name:	Security and Certification Manager components design and implementation (IT-2)				Page:	3 of 48
Reference:	D4.3	Dissemination:	PU	Version:	1.0	Status: Final

Table of Contents

Document Information	2
Table of Contents	4
List of Figures	6
List of Acronyms	7
Executive Summary	8
1 Introduction	9
1.1 Purpose of document	9
1.2 Relation to other project work	9
1.3 Structure of document	9
2 Supply chains	10
2.1 The EDC into a supply chain	10
2.2 SACM into a supply chain	10
3 EDC integration	12
3.1 Delta from IT-1 validation	12
3.1.1 Controller	12
3.1.2 Enforcer	12
3.1.3 Register and Planner, and Security Capability Model (SeCM)	12
3.1.4 Remediation Module	13
3.1.5 Interfaces with other components and subcomponents	16
3.2 IT-2 version of the EDC (deltas)	18
3.2.1 Planning of the changes until the end of IT-2	18
3.2.2 Already implemented changes to meet IT-2	18
3.2.3 Authorization and authentication	19
3.3 EDC integration-IT-2	19
4 SACM integration	20
4.1 Deltas from IT-1 validation	20
4.1.1 Evidence auditing module -IT2	20
4.1.1.1 Event Calculus	20
4.1.1.2 Evidence auditing mechanism	21
4.1.1.3 Drools	22
4.1.1.4 Event Calculus in Drools Syntax	25
4.1.2 Evidence Collection Engine-IT2	25
4.1.3 Asset Loader	25
4.2 IT-2 version of the SACM	26
4.2.1 Planning of the changes until the end of IT-2	26
4.3 STS Security Assurance solution –IT2	26
4.3.1 Security Assurance platform-IT2	26
4.4 Certification	26
4.5 Backend – SACM reasoning capabilities	27

Document name:	Security and Certification Manager components design and implementation (IT-2)				Page:	4 of 48
Reference:	D4.3	Dissemination:	PU	Version:	1.0	Status: Final

4.5.1	Confidentiality Property Criterion – The users access a service from a set of white-listed IPs	28
4.5.2	Confidentiality Property & Privacy Criteria – A system resource (e.g., file or service call) is accessed only by a list of authorized users	28
4.5.3	Integrity Property Criterion – For every request on a specified service S2, there must have been called the service S1 first	29
4.5.4	Integrity Property Criterion – There is only one active login session for each user on a service	29
4.5.5	Availability property SLA – For every request on a specified service, there is a response within a specified time window	30
4.5.6	Availability property SLA – A service must be available and must not be down for more than a predefined threshold	30
4.5.7	Integrity and Availability property – Observe potential ransomware activity on system resources	31
4.5.8	Metrics – Compute usage metric	31
5	Conclusions	34
6	References	35
Annex A. The GUI of the SACM		36
Annex B. Event Calculus Axioms		45
Annex C. Drools Rule Syntax example		47

Document name:	Security and Certification Manager components design and implementation (IT-2)				Page:	5 of 48
Reference:	D4.3	Dissemination:	PU	Version:	1.0	Status: Final

List of Figures

Figure 1: Example recipe of the ReM.....	15
Figure 2: The ReM tool workflow, phase 1: recommendation	15
Figure 3: Architecture of the ReM tool.....	17
Figure 4: How the Event Calculus Functions.....	20
Figure 5: Expanding Figure 4 concepts.....	21
Figure 6: High-Level Drools Functionality.....	22
Figure 7: 4Drools Semantics.....	24
Figure 8: Drools Logic	24
Figure 9: Basic Rule Syntax in Drools.....	24
Figure 10: 0The SACM GUI – Assurance Platform – Start the creation of a New Project	36
Figure 11: 0The SACM GUI – Assurance Platform – Create and store Project.....	37
Figure 12: The SACM GUI – Assurance Platform – Project’s empty view.....	37
Figure 13: The SACM GUI – Assurance Platform – Start the creation of a New Asset.....	38
Figure 14: The SACM GUI – Assurance Platform – Provide the type of the asset	38
Figure 15: The SACM GUI – Assurance Platform – Provide details for a software asset	39
Figure 16: The SACM GUI – Assurance Platform – Create and store asset.....	39
Figure 17: The SACM GUI – Assurance Platform – Created assets in the project’s view	40
Figure 18: The SACM GUI – Assurance Platform – Initiate a monitoring assessment	40
Figure 19: The SACM GUI – Assurance Platform – Select an availability profile.....	41
Figure 20: The SACM GUI – Assurance Platform – Select the asset and the model execution type of the assessment.....	41
Figure 21: The SACM GUI – Assurance Platform – Created assessment profiles in the project’s view	42
Figure 22: The SACM GUI – Assurance Platform – Assessment Result overview	42
Figure 23: The SACM GUI – Assurance Platform – Availability of successful event details	43
Figure 24: The SACM GUI – Assurance Platform – Assessment Results overview	43
Figure 25: The SACM GUI – Assurance Platform – Availability violation event details.....	44
Figure 26: The four axioms of the basic structure of Event Calculus	45

Document name:	Security and Certification Manager components design and implementation (IT-2)				Page:	6 of 48
Reference:	D4.3	Dissemination:	PU	Version:	1.0	Status: Final

List of Acronyms

Abbreviation / acronym	Description
API	Application Programming Interface
DMN	Decision Model and Notation
EC	European Commission
EDC	Enforcer & Dynamic Configuration
EvC	Event Calculus
HSPL	High-Level Security Policies Language
ICT	Information and Communication Technologies
IRO	Intent-based Resilience Orchestrator & Dashboard
KB	Knowledge Base
KPI	Key Performance Indicator
NFV	Network Function Virtualization
NSF	Network Security Function
ReM	Remediation Module
RII	Recipe Instruction Interpreter
SACM	Security Assurance and Certification Manager
SCM	Security and Certification Manager
SeCM	Security Capability Model
SIA	Security Infrastructure Abstraction
TIM	Trust & Incident Manager
TM	Trust Manager
UI	User Interface
UX	User eXperience
WP	Work Package

Document name:	Security and Certification Manager components design and implementation (IT-2)				Page:	7 of 48
Reference:	D4.3	Dissemination:	PU	Version:	1.0	Status: Final

Executive Summary

The document describes the implementation plan, approach, and techniques for the Security Certification Manager (SCM) module of the FiSHY project. This means the integration of the various components of the SCM module, such as the Enforcer and Dynamic Configuration (EDC) and the Security Assurance Certification Manager (SACM).

The reader will find details on the role of the EDC and SACM within a supply chain as well as any new updates regarding the latter since the release of deliverable D4.2. In particular, the document includes deltas from IT-1 validation of the EDC and SACM integration. Additionally, the document includes a detailed planning in terms of developing and integration of the components until the end of IT-2.

Regarding EDC, the present deliverable contains a detailed description of the EDC controller and enforcer advancements, along with the Remediation module recipes and EDC interfaces with other components of FiSHY platform.

Regarding SACM, the present deliverable contains a detailed description of the STS tool (which is the basis for the SACM integration) back-end reasoning capabilities and their implementation, how SACM will contribute in the certification procedure and its front-end advancements.

Document name:	Security and Certification Manager components design and implementation (IT-2)				Page:	8 of 48
Reference:	D4.3	Dissemination:	PU	Version:	1.0	Status: Final

1 Introduction

1.1 Purpose of document

Deliverable D4.3 documents the second iteration (IT-2) of the design and implementation of the Security and Certification Manager (SCM) component of the FISHY Framework, as foreseen in the Description of Action. This final delivery of the design and implementation activities T4.1 and T4.2 within WP4 is due on M26 (October 2022). The document takes as starting point the respective one for IT-1, D4.1, which was delivered in M9 (May 2021) and presents the activities aiming at evolving the different components as well as the needed adjustments and refinements following the feedback obtained from the piloting activities in WP6.

1.2 Relation to other project work

Deliverable D4.3 builds on top of D4.1 (M9, May 2021), also considering the integration insights of D4.2 (M13, September 2021). It is the final delivery of T4.1 and T4.2, which focus on the two main components of SCM: SACM and EDC. Then, T4.3, which is about the local integration of SACM and EDC into SCM, will run until M30 (February 2023) and deliver D4.4 to conclude WP4.

WP4 runs fully in parallel and with a symmetric internal structure to WP3, which is about delivering TIM. WP4 receives relevant input from WP2 to set the grounds for the actual design and implementation activities carried out within the WP. The output of WP4 is smoothly integrated with the rest of the FISHY Framework within WP5, particularly T5.3, and is eventually piloted within WP6, which sends feedback considered in WP2, therefore, closing the loop. Obviously, the different technical achievements of WP4 also influence the activities in WP7 about impact generation.

1.3 Structure of document

This document is structured into four major chapters.

Chapter 2 presents some general considerations that are relevant to the supply chain scenarios developed along the document

Chapter 3 presents the highlights during IT-2 about EDC, considering design and implementation; it also sheds some light on its integration within SCM.

Chapter 4 presents the highlights during IT-2 about SACM, considering design and implementation; it also sheds some light on its integration within SCM.

Document name:	Security and Certification Manager components design and implementation (IT-2)				Page:	9 of 48
Reference:	D4.3	Dissemination:	PU	Version:	1.0	Status: Final

2 Supply chains

When preparing IT-2, new requirements emerged from a careful analysis of the supply chain use cases (WP2). At the WP4 level, an analysis has been performed to assess the impact (and the needed adaptation) to allow the use of EDC and SACM in a supply chain. The results can be summarized as follows:

- EDC is almost not impacted by the supply chain requirements;
- The supply chain requirements significantly impact the SACM as they drive the types of assessments/reasoning that the SACM will perform.

2.1 The EDC into a supply chain

The EDC enforces security policies obtained from the intents and stored in the repository as HSPL (High-Level Security Policies Language) statements. The policies are enforced thanks to a refinement process, which transforms HSPL into low-level configurations. This refinement process needs precise information about the network landscape where policies will be enforced thanks to the network security controls (which own security capabilities). Moreover, it needs full control of the software network infrastructure to resolve non-enforceability issues and react to security incidents. Therefore, the scope of the EDC is an individual organization.

The supply chain sees the involvement of several organizations that may have different levels of relative trust. Therefore, using one instance of EDC for each organization appears to be the most practical approach to use the EDC. Hence, an instance of the EDC in one organization will not be accessible to users not belonging to that organization, at least in the scenarios we are addressing in the FISHY project. The current EDC implementation applies to this scenario.

The EDC can also be seen as a set of services available to a set of organizations, provided proper isolation is guaranteed to avoid information leakage between organizations. Implementing this isolation is not an objective of the FISHY project, as it is just an implementation requirement. If it receives the proper inputs, it produces outputs that can be stored in any repository. However, implementing this separation in real scenarios is not straightforward and will not be implemented during the project.

2.2 SACM into a supply chain

The SACM is responsible for assessing and validating a series of predefined security metrics for FISHY pilots/use cases. These predefined metrics cover the triangle of Confidentiality, Integrity, and Availability (CIA); however, several other security metrics tailored to the needs of FISHY pilots/use cases are evaluated through SACM.

Since a supply chain may include several assets (e.g., hardware, software, persons), a detailed and relational description must be provided to the SACM tool to perform its validation concretely. Therefore, SACM includes in its functionalities the asset model loader, a real-time component that parses information regarding a pilot's assets and creates the asset model of the supply chain. Based on this asset model for each pilot, SACM collects all the necessary information through its evidence collection engine while the evidence auditing mechanism performs the additional reasoning. The security metrics/rules that are evaluated are written in the Event Calculus logic language, which will be explained in the next sections, while the end user may create any custom rule through the GUI of the SACM.

Document name:	Security and Certification Manager components design and implementation (IT-2)				Page:	10 of 48
Reference:	D4.3	Dissemination:	PU	Version:	1.0	Status: Final

SACM also uses the multitenancy architecture, meaning that it can support, in one instance, several use cases as the assessments/reasoning of the security metrics are presented per organization/project. Therefore, in one instance, it can support multiple views of metrics assessments of the same supply chain of one organization or several assessments of different security metrics from different supply chains.

Document name:	Security and Certification Manager components design and implementation (IT-2)				Page:	11 of 48
Reference:	D4.3	Dissemination:	PU	Version:	1.0	Status: Final

3 EDC integration

This section is structured as follows. Section 3.1 presents the updates since the release of deliverable D4.2. During this period, the three main EDC components underwent incremental improvements. Furthermore, a new component has been developed, which uses the threat intelligence information produced in WP3. Section 3.2 then presents the summary of changes that are expected to meet the IT-2 requirements.

3.1 Delta from IT-1 validation

3.1.1 Controller

During the updates required to support the F2F use case, some limitations of this component emerged. In particular, the CLIPS forward reasoning rules were structured, so it was not easy to support new scenarios. Moreover, the potential updates given by the new components described in the security capability mode made support of new categories of security policy types more complex than expected. Moreover, the integration with the dashboard showed minor issues in terms of the way the web service was structured. In particular, the API methods requiring interactions with users (e.g., to select among alternatives ways to enforce HSPL policies).

Therefore, the core features of the forward reasoning engine used by this component have been completely re-engineered to be more extensible, flexible, and stable. Overall, the design of the component remained the same; the internal reasoning engine has been completely rewritten. That is, the templates and the rules used in ClipsPY (the forward reasoner used by the Controller, as detailed in D4.2) have been refactored based on the experience gained in the project.

3.1.2 Enforcer

During the last months, this component did not require significant modifications. Only the translator from medium-level to low-level policies was slightly updated to support the new high-level changes in the Security Capability Model. These changes mainly focus on supporting new categories of conditions used by layer 7 filters, which use regular expressions.

An additional, module was developed to interact with the RabbitMQ message handler used in the F2F use case. This module is obsolete by the integration performed at project level as the Central Repository will incorporate its features. A new format was defined to support the wrapping of the low-level configurations generated that was usable by the deployment techniques used in that scenario.

3.1.3 Register and Planner, and Security Capability Model (SeCM)

The Register and Planner is a web service front end for accessing data about NFVs represented according to the Security Capability model. The core of the service remained stable. Only the GUI underwent some modifications to be compliant with other dashboard components. Moreover, some manual operations were made accessible on the GUI, including fully supporting the Security Capability Model lifecycle.

On the other hand, the Security Capability Model required effort to fully support the case of application layer filters. Indeed, the application layer filters also accept conditions whose values are expressed as regular expressions. The existing model required a few adjustments as regular expressions showed peculiarities that were not considered in previous versions of the SeCM. For instance, given the value

Document name:	Security and Certification Manager components design and implementation (IT-2)				Page:	12 of 48
Reference:	D4.3	Dissemination:	PU	Version:	1.0	Status: Final

of a condition, i.e., a string of characters (e.g., on the values of a URL), that condition can be evaluated differently. For instance, string match only checks for existing characters without applying any expansion. On the other hand, if strings are interpreted as regular expressions, several expansions are performed to generate the regular automaton that will perform the matching (e.g., the star operator '*'). Even more complex, there are different profiles for regular expressions (*regex*), like the POSIX, ISO 8601 and Perl Regex versions¹.

The Squid Proxy² was the best application-layer filter candidate as the configuration language it uses is simple and expressive. Squid supports the string match and full regex match, which corresponds more or less to grep vs. egrep checks. All the security policy conditions used by Squid that were not already in SeCM have been added to the current version of the model.

3.1.4 Remediation Module

The Remediation Module (ReM) of the EDC recommends actions to mitigate the increased network-centered threats that other FiSHY tools (e.g., PMEM, TIM) have identified.

The information about the risks to mitigate is reported in a Threat Intelligence Report (TIR) from the WP3 components, which are in charge of identifying anomalies. Threat Intelligence Reports that have an impact for the ReM will be selected by the IRO.

The solutions recommended by ReM are named Remediation Recipes (or simply Recipes), which are sequences of actions represented in an abstract format.

The application of a Recipe selected by the FiSHY user proposes a list of changes in the networked environment. Following the FiSHY policy on the controls, changes are not automatically enforced by ReM. Rather, changes are shown in a dashboard to the FiSHY roles that are in charge of approving them. A fully automatic reaction can be selected via ReM configuration.

The changes proposed are:

- modifications to the landscape (e.g., adding new security controls); and
- changes to the (network) High-level Security policies, i.e., HSPL statements inserted as the refinement of new intents added by the ReM. For maximum coherence of the framework, the second option should be preferred.

Changes to the higher-level policies will then translate into changes in the configuration of the security controls in the landscape, including the ones that the ReM tool proposes to add. Configurations are obtained by refinement and translation performed by the other EDC tools, Controller and Enforcer.

3.1.4.1 Remediation Recipes

All the Remediation Recipes are characterised by:

- a set of *labels* that indicate the threat scenario they address;
- a set of *enabling constraints* that allow understanding when a Recipe is applicable. For instance, Recipes report all the information necessary for their correct deployment and the security capabilities that need to be enforced (e.g., a layer 7 filter), which may not be available in the network.
- the set of remediation *deployment instructions*, written in an abstract language, that programmatically state all the steps that need to be performed to remediate the identified risks.

¹ <https://github.com/PCRE2Project/pcre2>

² <http://www.squid-cache.org/>

Document name:	Security and Certification Manager components design and implementation (IT-2)				Page:	13 of 48
Reference:	D4.3	Dissemination:	PU	Version:	1.0	Status: Final

Figure 1 presents an example of deployment instructions for a ReM Recipe. This Recipe remediates an ongoing attack against a specific host in the network. It proposes inserting a control to drop a specific payload in the path between the attacker and the target host. For example, this can be useful if the impacted host has become part of a botnet, and the Command and Control messages between them exhibit a specific string that can be filtered to disrupt the communication between the bot and the botnet master.

The language describes different concepts, reported here using different colors.

- The keyword representing operations are reported in green. These operations are available as they are exposed either from the FISHY framework or any of its components
 - adding security controls (e.g., `add_firewall`),
 - modifying the configuration of specific security controls (e.g., `add_filtering_rule`),
 - modifying the network layout or flows,
- The keywords that represent language-specific concepts are reported in red. They are introduced to satisfy the language-required features, e.g.:
 - results from past computations (e.g., `found_node`),
 - placeholders for predefined concepts (e.g., `new_node`),
- inputs from FISHY threat intelligence are reported in orange (e.g., `impacted_host_ip`);
- information related to the security capability model is reported in blue (e.g., `security_capability=UrlRegexCapability`).

The actions proposed are tailored to the actual landscape of the target network. In this case, if a security control with payload filtering capability is already present in the path from the impacted host to the attacker, the existing control is reconfigured to filter the target payload; otherwise, a new security control is placed in front of the impacted host, and is appropriately configured with the needed filtering rules.

The target landscape is described using a Landscape Description Language. As explained in past deliverables, this is a graph-based representation of the network layout, which describes both nodes (their attributes, e.g., capabilities) and edges. This representation is prone to be imported with graph libraries available for the main programming languages (e.g., iGraph on Python). Currently, it is represented as a simple text file that follows the specification defined during a past EC-funded project (SECURED [8]). However, the final format for the landscape description may be updated as soon as other aspects of the software network are explored. Furthermore, we are investigating the possibility of avoiding using network graphs if the network flows (e.g., the SDN ones) are expressive enough for our needs.

Document name:	Security and Certification Manager components design and implementation (IT-2)				Page:	14 of 48
Reference:	D4.3	Dissemination:	PU	Version:	1.0	Status: Final

```

list_paths from impacted_host_ip to 'attacker'
iterate_on path_list
find_node of type 'l7filter' in iteration_element
if not present
add_firewall      behind      impacted_host_ip      in      iteration_element      with
security_capability=UrlRegexCapability
found_node=new_node

add_HSPL (subject="attacker", object=" impacted_host_ip", action="is not authorized to access")
to found_node
endif
end iteration

```

Figure 1: Example recipe of the ReM

ReM will implement the following workflow (see Figure 2), which can be divided into two phases:

- recommendation of Recipes;
- deployment of the selected Recipe.

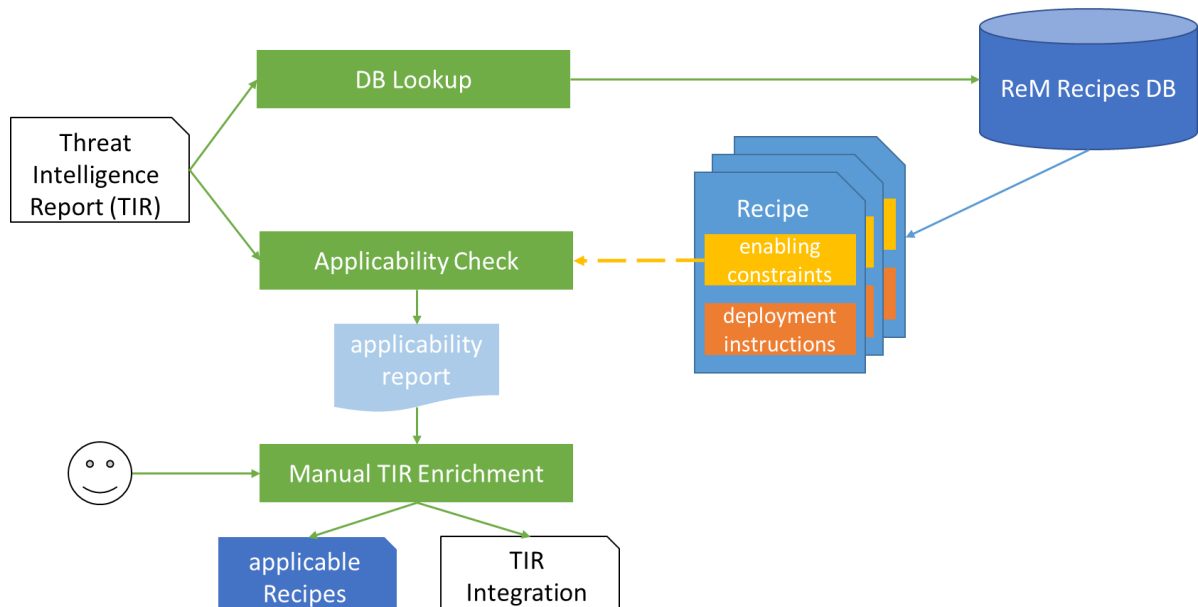


Figure 2: The ReM tool workflow, phase 1: recommendation

When the ReM tool receives the notification of a risk to mitigate in the form of a TIR, it uses the information in the TIR to look up into a database where all the remediation recipes are stored (*DB lookup*). All the remediation recipes are labelled according to a standard set of categories to allow finding an easier match with threat intelligence information. For instance, the Recipe to mitigate the risks from a malware infection is labelled as 'malware' and further refined with more specific data, such as 'botnet' or 'ransomware.' If needed, custom recipes against specific malware strains may be added to increase the efficacy of the proposed mitigations. It is expected that the TIR will include the

Document name:	Security and Certification Manager components design and implementation (IT-2)				Page:	15 of 48	
Reference:	D4.3	Dissemination:	PU	Version:	1.0	Status:	Final

same set of labels (or information that will allow mapping to these labels easily), allowing very fast filtering of the recipes.

The recipes produced by the DB lookup phase are all Recipes that can mitigate a specific set of threats. Nonetheless, depending on the target landscape, these recipes may not be deployable in the current threat scenario. For this purpose, an additional step is performed, named Applicability Check, where all the enabling constraints are evaluated. The applicability report lists all the directly applicable recipes. When a Recipe is not applicable, the tools reports the constraints that were not satisfied. Therefore, a user analyzing the report can provide additional information enabling additional recipes (e.g., missing IP/URL information or missing information about honey pot networks).

The additional data provided by the users are saved in a data structure developed for the purpose, named TIR integration (e.g., within the ReM subcomponent). In this way, these data are neither forgotten nor merged with official information coming from the tool. It is a future task to understand how this information can be integrated into future versions of the FiSHY framework.

The next phase of the ReM tool workflow starts when the user decides on the remediation recipe to enforce. At this point, the Recipe Deployment Engine reads and starts interpreting the deployment instructions.

In the first phase, all the generic concepts are made concrete with the TIR and the TIR Integration information. In the Recipe in Figure 1, the generic concepts "*impacted_host_ip*" and "*attacker*" are associated with their actual IP addresses, and this information is made available to the rest of the EDC components.

Then, a Remediation Recipe Interpreter executes the deployment instructions and generates as output, depending on the selected Recipe:

- the HSPL policies to enforce;
- (optionally, if the networked scenario where the remediation takes place needs it) a set of suggested changes to the landscape. Examples of these changes are:
 - moving network nodes to a different position in the network
 - removing nodes from the network
 - changing the network connectivity (either by connecting a node to a different network or by redefining the flows by changing the routing information)
 - adding nodes (e.g., an NSF).

3.1.5 Interfaces with other components and subcomponents

At the current stage of design and development, the ReM tool interacts (or will interact in future iterations) with the following FiSHY components:

- (WP3) TIM will provide this subcomponent with the information discovered by the threat intelligence;
- (WP5) NED will process the instructions provided as output by this subcomponent to remediate the TIM-identified threat scenario.

Document name:	Security and Certification Manager components design and implementation (IT-2)					Page:	16 of 48
Reference:	D4.3	Dissemination:	PU	Version:	1.0	Status:	Final

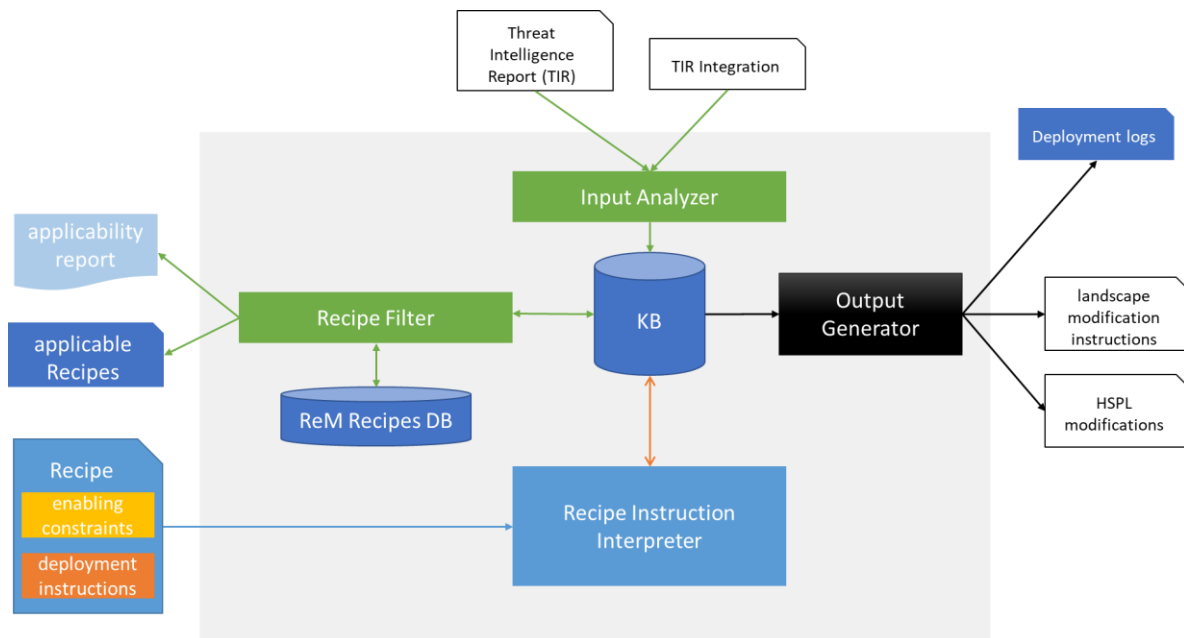


Figure 3: Architecture of the ReM tool

This section provides insights into the Recommendation and Remediation tool, particularly detailing the sub-modules constituting the tool and the exchanged communications (see Figure 3)

The Input Analyzer is tasked with the interpretation of the Threat Intelligence Report. It extrapolates the information needed to enrich the recipe instructions with concrete information from the TIR. This information includes the labels that allow filtering recipes and the IP addresses (and possibly the TCP/UDP ports) of the impacted hosts and the attacker.

The input Analyzer stores the interpreted information in a Knowledge Base (KB), which stores all the data produced by all the ReM tool components.

The Input Analyzer is also the module that parses and stores the additional information the user provides as TIR Integration into the KB.

The Recipe Filter is the module that reads from the KB the TIR and TIR integration. It extracts the labels and the additional information that allows for categorizing the threats. At the current development and integration stage with threat intelligence, the enabling constraints are limited to checking a set of labels. The Consortium has discussed Further extensions (e.g., use of ML techniques); however, they have been discarded as they do not add enough value to the ReM technology.

From the extracted information, this module selects the Recipe that applies. More precisely, this module queries the ReM Recipe DB and:

- extracts the applicable Recipes;
- checks the satisfaction of the enabling constraints associated with selected Recipes;
- produces the Applicability Report;
- produces the Applicable Recipe list.

The Applicability Report is then presented to the user. The Applicability report is currently a list of Recipes and the information that the Recipe Filter was unable to collect or verify the Enabling Constraints. Examples of Enabling constraints are:

- check for the presence of specific attributes (e.g., the IP addresses of the Command and control for malware infections);
- need for specific security capabilities (e.g., if the Security Capability Catalogue contains an element able to filter by URLs).

Document name:	Security and Certification Manager components design and implementation (IT-2)				Page:	17 of 48
Reference:	D4.3	Dissemination:	PU	Version:	1.0	Status: Final

The recipes in the Applicability Report for satisfying all the constraints are listed in the Applicable Recipe List.

This module is in charge of deploying the Recipe that it receives as input. The Recipe to deploy is received as user input.

When the user selects a recipe among the applicable ones presented by the Recipe Filter, the Recipe Instruction Interpreter (RII) interprets the deployment instructions in the Recipe. This module concretizes them using the information contained in the Knowledge Base.

The result of the Recipe execution is also stored in the KB. It consists of the following:

- changes to the landscape, such as:
 - adding new nodes, including adding new security controls/NSFs;
 - deleting edges and disconnecting nodes from the network;
 - moving nodes to different networks/subnetworks.
- changes to the HSPL policies.

When the interpretation of all the deployment instructions is complete, the Output Generator reads the internal KB and provides the output in a format that is usable for other FISHY components and to the FISHY users; in particular, it produces:

- the needed modifications to the landscape, including, for example, new NSFs that must be deployed and the necessary alterations to the connections among network nodes, which will be sent to the SIA for their deployment;
- the HSPL policies to enforce;
- a set of deployment logs describing all the actions taken to remediate the risk.

This module is currently a simple interface to the KB. The integration with the FISHY repositories is not completed in the currently implemented version, and the HSPL policies are not yet stored in the central repository. This activity is not considered complex and will be performed in the following weeks.

3.2 IT-2 version of the EDC (deltas)

3.2.1 Planning of the changes until the end of IT-2

There are no changes to the EDC architecture and its relations with other FISHY components to note for Enforcer, Controller, and Register and Planner. Only the ReM will be connected to threat intelligence components developed in WP3. These changes will be refined during the next months, within the tasks expected for the IT-2 integration. Currently, we have investigated the integration with PMEM and TIM that will probably happen through the use of the Central Repository, where data will be exchanged, and the IRO dashboard, where the users will explicitly authorize these decisions. The relations between EDC and the IRO Dashboard have been clarified in the past deliverable D4.2 and do not require updates.

3.2.2 Already implemented changes to meet IT-2

The updates to the EDC components expected for IT-2 at the design level have already been implemented for all the components but ReM, for which we expected minor updates when it is fully integrated into the FISHY framework.

Moreover, an analysis of the use cases has been performed that seems to indicate that no major updates will be needed to the components. Nonetheless, changes are possible to comply with the

Document name:	Security and Certification Manager components design and implementation (IT-2)				Page:	18 of 48	
Reference:	D4.3	Dissemination:	PU	Version:	1.0	Status:	Final

potential extensions to the security capability model to support additional security controls that may be used in these scenarios.

Moreover, a significant effort has been invested by WP4 partners to model the features of software networks formally. We are focusing on NFV and SDN features, working on the FISHY sandbox and all the Kubernetes features and packages that are used in FISHY. The idea behind this approach is to avoid the hard coding of operations on the landscape. With a formal model explaining precisely the entities involved, the relations among them, and the consequences at the security policy enforcement level, we expect to build a reasoning system that can significantly improve the incident response and the remediation actions taking advantage of the comprehension of the semantics of the operations on the landscape. This approach will be documented in the next WP4 Deliverable.

3.2.3 Authorization and authentication

For the correct use of the EDC and its subcomponents, the following roles have been highlighted, which may be played by one or more users:

- *FISHY (local) Policy administrator*, who starts the EDC services and makes them available to other FISHY components. It may also have to ensure proper reachability by also configuring filtering devices
- *Policy designer*, who executes the Controller to refine HSPL policies, discusses enforceability considerations, and identifies proper measures to correct non-enforceability issues;
- *Policy implementer*, who executes the Enforcer to translate the medium-level policies generated by the Controller into low-level configurations. Moreover, he can decide to deploy the generated configurations by using the SIA's features;
- *Incident response team member*, who can use ReM (and the additional services required by this module to Controller and Enforcer) to react to incidents or other security-relevant events detected by the threat intelligence.

3.3 EDC integration-IT-2

The integration of the EDC with the rest of the infrastructure is in line with the expectations. The integration of all the EDC components with the Central Repository has started and does not pose significant challenges.

Only the integration with SIA is not complete. However, updates are expected before M28, when the modelling of software networks will reach a stable point. We also expect that the effort for properly modelling the software network features will be documented in D4.4 and properly framed in the WP2 design.

The integration of the ReM with WP3 threat intelligence, which emerged only recently, needs additional effort to be completed. No major risks that may require significant changes to the architecture have emerged during our analyses.

Document name:	Security and Certification Manager components design and implementation (IT-2)				Page:	19 of 48	
Reference:	D4.3	Dissemination:	PU	Version:	1.0	Status:	Final

4 SACM integration

This section is structured as follows. Section 4.1 presents the updates since the release of deliverable D4.2. During this period, the SACM component was configured and deployed in the FISHY dashboard, including the auditing mechanism based on Event Calculus and Drools implementation. In addition, new tailored security metrics were created while the evidence collection engine was fully integrated into the F2F use case. Furthermore, the asset loader module, as the core component of the SACM, was introduced, and the development of the F2F asset model was utilized. Section 4.2 presents the SACM's further development until the end of IT-2, while section 4.3 presents the multitenancy GUI of the SACM that has been deployed in the FISHY dashboard. Section 4.4 encapsulates the basic concepts of the certification used by the SACM tool, while section 4.5 presents the SACM reasoning capabilities and integrated security metrics that can be used within it.

4.1 Deltas from IT-1 validation

4.1.1 Evidence auditing module -IT2

4.1.1.1 Event Calculus

Event Calculus (EvC) is a logical language for representing and reasoning about actions and their effects as time progresses. It is a formalism for reasoning about action and change. The core Event Calculus concepts contain the following definitions:

- *actions* – which are called events and indicate changes in the environment
- *fluents* – which are time-varying properties (predicates/functions)
- *timepoint sort* – which implements a linear time structure on which actual events occur.

It is based on first-order predicate calculus and can simulate various phenomena, such as actions with indirect effects, actions with non-deterministic effects, compound actions, concurrent actions, and continuous change. The EvC defines predicates for expressing, among others, which fluents hold and when (HoldsAt), what events happen (Happens), and what their effects are (Initiates, Terminates). It adopts a straightforward solution to the frame problem, which is robust and works in the presence of each of these phenomena.

In a generalized view, event calculus is working using a Logic Machinery, as shown in Figure 4, to provide evaluations regarding actions that are happening or happened in a past timeline. The Logic machinery can also provide representations of predicates, values that specific actions are meant to alter, to extract evaluations of aggregated actions on demand.

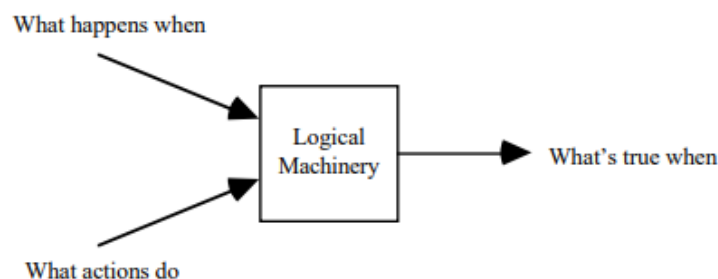


Figure 4: How the Event Calculus Functions

Document name:	Security and Certification Manager components design and implementation (IT-2)				Page:	20 of 48	
Reference:	D4.3	Dissemination:	PU	Version:	1.0	Status:	Final

Expanding this generalised approach, we can add the above properties into the figure and enlarge in detail. Subsequently, the Initially, Happens and the temporal ordering formulae, as well as the Initiates and Terminates formulae, are passed through the Event Calculus Axioms, producing the outcome represented as a fluent in Figure 5, that holds the value produced, either to re-enter a rule logic machinery, either to produce the outcome fluent (True/False) that evaluates the sequences of actions performed.

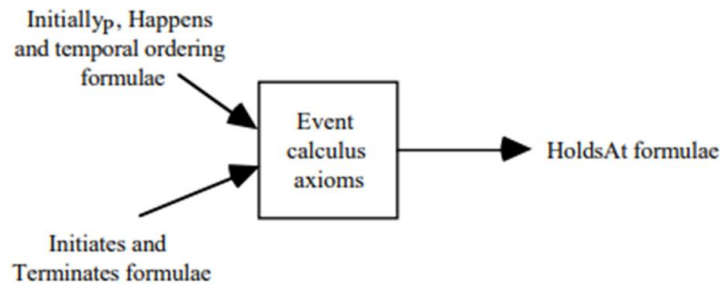


Figure 5: Expanding Figure 4 concepts

EvC Axioms are explained thoroughly in Annex A of this document. In a more generalized view, the axioms can be seen as the concatenation that keeps these blocks interconnected since we view the predicates as the building blocks of this logical representation. In EvC, specific values of the fluents describe a situation. An event that changes the value of one or more fluents has as a consequence the change of the situation. An evolution of the world is a sequence of actions and situations.

We formally define events and fluents. Events occur sequentially or in parallel on denoted time points. Events can change the state of fluents and trigger new events. Rules model these transactions. The rules have preconditions. If the preconditions are satisfied (left-hand statements), the rule is executed and may change the state of a fluent. At this point, we accept statements that the predicate calculus has proved. According to these statements, we model rules that implement management logic. An event can model a fire alarm. The event triggers a set of rules which check if any actions must be taken (functional and non-functional properties of reaction strategy). When the counteractions are completed, another set of rules can be triggered to determine which security properties and safety properties are satisfied. The events and the changes they cause produce a trace in time. The final state of the trace determines the final outcome.

Examples of event calculus theoretical problems found in Sergot [1] and Mueller [2] show the logical representation of the aforementioned simplified basis in a real-world problem. Additionally, Liu [3] focusing on the Self-adaptation of multi-agent systems expands the requirements for self-adaptation; they facilitate monitoring and reasoning about the actions of agents, achieving requirements-driven planning at runtime.

4.1.1.2 Evidence auditing mechanism

The reasoning mechanism of the auditing module is modelled in EvC. Its basic implementation is based on Drools logical language, where several of its functionalities are implemented in Java. EvC is a first-order temporal logic that can both represent and reason actions and their effects over time. By abstracting the above concepts, EvC basic elements are comprised of events and fluents. An auditing mechanism is a monitoring tool that performs continuous assessments and is based upon these core logic factors of EvC that we mentioned in terms of comprising the rules continuously checked in a system.

An event in Event Calculus is specified as something that occurs at a specific time instance and is of instantaneous duration. Furthermore, it may cause some change in the state of the reality that is being

Document name:	Security and Certification Manager components design and implementation (IT-2)				Page:	21 of 48	
Reference:	D4.3	Dissemination:	PU	Version:	1.0	Status:	Final

modelled while fluents represent this state. The logic is implemented in the form of reasoning rules. Based on the ongoing events and the status of the related fluents, when the preconditions of a rule are satisfied (left side of the rule), the rule fires and performs some actions (right side of the rule). In our case, a set of rules define monitoring criteria for the CIA or other security/privacy aspects of the examined assets.

As a result, the auditing mechanism core functionality introduces event calculus and its main elements as a basis of its logic. Thereupon, the auditing mechanism reasons (using the rule sets/axioms), maintains the status of the monitored assets (using fluents), and exchanges information with the other components of the SACM, such as the evidence collection engine based on messages, which resemble events in the event calculus.

4.1.1.3 Drools

Drools³ is a group of tools that provide us the ability to distinguish between and make sense of the logic and data present in a business processes. Focusing on knowledge representation to express propositional and first-order logic in a clear, unambiguous, and declarative manner, Drools is a Turing-complete Production Rule System [4] where an inference engine, that scales many rules and facts, operates as its core system. This inference engine makes conclusions that result in actions when facts and data are examined against production rules, commonly referred to as productions or just rules.

The Drools engine, which serves as the brains behind the Drools tools, saves, manipulates, and assesses data in order to carry out user-defined business rules or decision models. The Drools engine works to compare incoming data, or facts, to the conditions of rules in order to determine if and how to execute the rules.

As shown in Figure 6, the Drools engine consists of the following essential parts:

- Rules: Decision Model and Notation (DMN) or business rules
- Facts: Domain model objects for evaluating situations and carry out actions.
- Production memory: the place in the Drools engine where rules are kept.
- Working memory: a stateful object that allows the rules to affect the objects.
- Agenda: site where registered, organized, and ready for execution activated rules are kept.

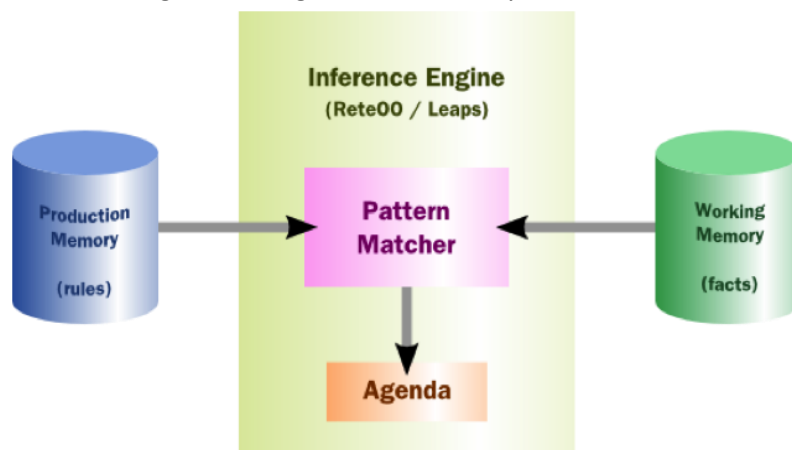


Figure 6: High-Level Drools Functionality

³ <https://www.drools.org/>

Document name:	Security and Certification Manager components design and implementation (IT-2)				Page:	22 of 48
Reference:	D4.3	Dissemination:	PU	Version:	1.0	Status: Final

The Rete algorithm⁴ is implemented and enhanced by Drools while ReteOO⁵ is the name of the Drools Rete implementation, which denotes that the Rete algorithm has been improved and optimized for use in object-oriented systems [5]. RetePlus [6] and Rete III [7] are two examples of the marketing names used by other Rete-based engines for their in-house improvements to Rete. The Production Matching for Large Learning Systems article [8] discusses the most typical improvements. The Rete algorithm has been extended in a variety of previous works. For example, Rete has been extended to include the concepts of time-stamped events and temporal constraints between events, allowing applications to write rules that process both facts and events [9], as well as being extended in terms of rule decomposition in Alpha-Node-Hashing and Beta-Node-Indexing [10].

Rule compilation and runtime execution are the two components of the Rete algorithm. The compilation algorithm outlines the steps taken to create an effective discriminating network from the rules stored in the Production Memory. A discrimination network, to put it simply, filters data as it travels through the network. Any Rete-based expert system creates a network of nodes, each of which corresponds to a pattern occurring in the rule's left-hand side (apart from the root) (the condition part). A complete rule is defined on the left-hand side by the route taken from the root node to a leaf node. Every node keeps a recollection of the information that fits that pattern. As new facts are stated or changed, they spread throughout the network and annotate nodes when they meet the pattern. A leaf node is reached and the accompanying rule is activated when one or more facts cause all of the patterns for a given rule to be satisfied.

An effective pattern-matching method for developing production rule systems is the Rete algorithm. In order to boost performance, the Rete algorithm is designed to sacrifice memory. The Rete algorithm demonstrates the following crucial traits:

- Through the use of node sharing, certain types of redundancy are reduced or eliminated.
- When joining data from several fact kinds, it stores partial matches.
- When facts are retracted from working memory, it enables the effective removal of memory components.

An inference engine is a type of computer program that attempts to find solutions from a knowledge base stored in Working Memory or Production Memory (rules and facts). In our case, the latter is the central component of the auditing mechanism we are developing. It starts by compiling a pattern matcher using the Event Calculus rules and then waits for events represented as facts in the figure to extract the results represented as Satisfaction Violations of these rules.

Conflict Resolution is a supplementary Drools addition to the classic Rete algorithm that is required when there are multiple rules on the agenda. Because firing a rule may have side effects on working memory, the rule engine must know the order in which the rules should be fired (for example, firing rule A may cause rule B to be removed from the agenda) or, in the case of the Monitoring Assessment tool, rule A may produce facts that feed rule B, which produces an assessment result. Drools' default conflict resolution strategies are Saliency⁶ and LIFO (Last In, First Out).

A rule system can be executed in two ways: forward chaining or backward chaining; systems that implement both are known as hybrid rule systems. Drools is a chaining engine that moves forward. Forward chaining is "data-driven" and thus reactionary, with facts asserted into working memory, resulting in one or more rules being true at the same time and scheduled for execution by the Agenda. In short, we begin with a fact, which then spreads, and we end with a conclusion.

⁴<http://citeseer.ist.psu.edu/context/505087/0>

⁵<https://docs.jboss.org/drools/release/5.3.0.Final/drools-expert-docs/html/ch03.html>

⁶ <https://docs.jboss.org/drools/release/5.2.0.Final/drools-expert-docs/html/ch05.html>

Document name:	Security and Certification Manager components design and implementation (IT-2)				Page:	23 of 48
Reference:	D4.3	Dissemination:	PU	Version:	1.0	Status: Final

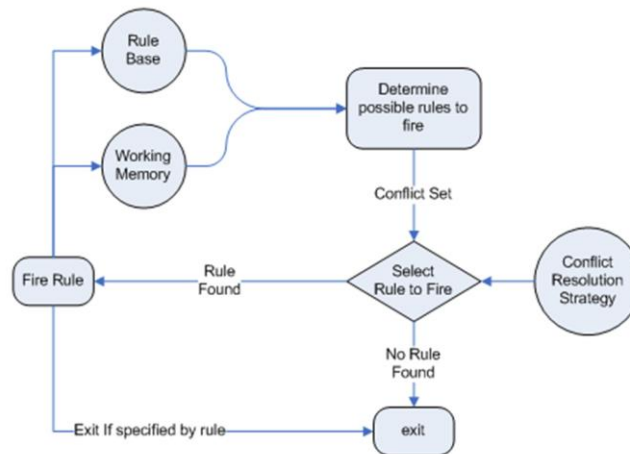


Figure 7: 4Drools Semantics

Rules are bits of information that are frequently written as "when some conditions occur, then take some actions," as we described when creating the Drools semantics in the abstract Drools syntax. The Left Hand Side (LHS) and the Right Hand Side (RHS) are used to illustrate this logic in the following way (RHS).

When (LHS)
<conditions are true>
Then (RHS)
<Do specified actions>
End

Figure 8: Drools Logic

The most important part of a Rule is its 'when' part. If the when part is satisfied, the 'then' part is triggered.

Rule <rule_name>
<attribute> <value>
When
<conditions>
Then
<actions>
End

Figure 9: Basic Rule Syntax in Drools

Additionally, Drools introduces global variables that can be used by all rules. In a typical monitoring assessment, the "when" part is where the event calculus logic is introduced, and the "then" part is where the storage of the assessment results, execution events, and memory object retractions are added.

Drools share nodes as well. Node sharing enables us to collapse several rules that repeat the same patterns so they don't need to be evaluated for every instance. We follow the same reasoning in our situation to yield conflicting outcomes (violation/satisfaction).

Document name:	Security and Certification Manager components design and implementation (IT-2)				Page:	24 of 48	
Reference:	D4.3	Dissemination:	PU	Version:	1.0	Status:	Final

4.1.1.4 Event Calculus in Drools Syntax

In order to encapsulate the concepts of event calculus already described into the Drools logic engine, we needed somehow to port the axioms of event calculus into the Drools specified logic. The first step for this integration was to define a very well-defined object-oriented structure that holds the basic principles of event calculus in Drools syntax. The challenge to this point is to syntax the drools itself in a manner that explains key concepts of event calculus such as the predicates (Happens, HoldsAt) and additionally define the event calculus Axioms.

Moreover, from a technical perspective, we had to keep in mind memory safety while defining this common schema and provide solutions in axioms to balance the code coherence and the theoretical/logical concepts of the event calculus in Drools syntax. Similar implementations are referred to as the Cerbere, a Jess tool production system designed to perform online causal, temporal, and epistemic reasoning based on the Event Calculus [11]. A Drools rule syntax example is presented in Annex B, where those terms are represented in the following contextual form and mapped with the Axioms of event calculus described in the previous section.

4.1.2 Evidence Collection Engine-IT2

The evidence collection engine is a tool that, based on the collected data and triggering events, formulates a rule or a set of rules and pushes the latter toward the monitor module for evaluation. Data and events, such as processes execution information, login attempt's information, etc. are mostly collected through Elasticsearch⁷ based on lightweight shippers (namely Beats) such as Filebeat⁸, MetricBeat⁹, PacketBeat¹⁰, which forward and centralize log data. Data can also be collected through Logstash¹¹, an open server-side data processing pipeline that ingests data from many sources transforms it, and then sends it to ElasticSearch. The evidence collection engine is initiated through respectively REST calls from the monitor module. This tool is essential for the evidence auditing mechanism functionality since it collects and feeds the essential information from the assets as events.

4.1.3 Asset Loader

Asset Loader Module is essential for the functionality of the SACM. It contains the definitions of the organizations, assets, projects inside the organizations, assessment criteria (rules and assumptions used), and assessment profiles. This component is responsible for receiving the security assurance model for the target organization. This model includes the organization's assets, the security properties for these assets, the threats that may violate these properties, and the security controls that protect the assets. The Asset Loader component is based on the custom-created SACM Assurance Model and it is introduced into FISHY in two different ways: either by inserting all the model data manually using respectively screens from the GUI of the SACM platform; or automatically, by filling up an excel file that includes all necessary information of the model that in a following step can be parsed using the GUI of the SACM.

⁷ <https://www.elastic.co/elasticsearch>

⁸ <https://www.elastic.co/beats/filebeat>

⁹ <https://www.elastic.co/beats/metricbeat>

¹⁰ <https://www.elastic.co/beats/packetbeat>

¹¹ <https://www.elastic.co/logstash/>

Document name:	Security and Certification Manager components design and implementation (IT-2)				Page:	25 of 48
Reference:	D4.3	Dissemination:	PU	Version:	1.0	Status: Final

4.2 IT-2 version of the SACM

4.2.1 Planning of the changes until the end of IT-2

Integration of the SACM with the FISHY dashboard has been completed. However, changes to the newest GUI based on Angular are expected before M30. Such changes will not affect the overall architecture of FISHY or the main components that SACM includes.

Asset Loader component will be configured for the rest of the use cases of FISHY/pilots. The already included-delivered security metrics that the SACM tool includes will be used across the rest of the supply chain scenarios. Furthermore, a new security metrics/rule regarding smart Contracts for supply chains has already started in term of developing focusing on F2F use case.

Integration of SACM with the Central Repository of FISHY has started and will be concluded before M26.

4.3 STS Security Assurance solution –IT2

4.3.1 Security Assurance platform-IT2

While the GUI of the Security Assurance Solution is fully integrated within the FISHY dashboard, only the auditing mechanism submodule of SACM will be integrated with the FISHY. The auditing mechanism submodule will transmit the results of the auditing procedure to the central repository of FISHY. However, the initiation of an assessment, the real time observation of the auditing procedure, along with the asset model creation are performed by the respectively described components of SACM via its GUI.

The respectively screens of the GUI of the SACM that has been integrated to the FISHY dashboard are presented in Annex A.

4.4 Certification

Individual businesses and Supply Chains are recognized internationally and particularly by the European Union as key enablers for economic growth; thus, a managerial capability is directly linked with the level of efficiency and effectiveness. Many businesses, mainly part of a Supply Chain, outsource various processes, critical information, and Information Communication Technologies services to third parties and highly interdependent dispersed nodes of heterogeneous cyber-physical infrastructures. An analysis of the extended concepts that revolve around certification will not be done since the legislative ecosystem is quite broad, and legal analysis is out of the scope of this deliverable.

In this section, we will provide an encapsulation of the basic concepts of the certification process to pinpoint the role of the EU interest in regulating the spectrum of privacy and security measures that need to be imposed by an organization or by a group of organizations, by building an information safe, security resilient and trustworthy environment for businesses and service providers. As a result of this process, we also highlight the necessity of the tools that have a crucial role in providing a complete cybersecurity posture for these organizations. This environment provides assurance not only as a regulatory measure for businesses; from the client's perspective, it supports the development of their confidence and trust in the services provided by an EU certified organization.

The NIS Directive 2.0 [12] contains a series of measures for improving cybersecurity infrastructure and particularly the resilience and incident response capabilities of public and private competent authorities. It provides a wide range of indications that encapsulate the general cybersecurity posture

Document name:	Security and Certification Manager components design and implementation (IT-2)				Page:	26 of 48
Reference:	D4.3	Dissemination:	PU	Version:	1.0	Status: Final

of an organization. Also, one of the key elements of the Commission's proposal is to address the security of supply chains and supplier relationships by requiring individual companies to address cybersecurity risks in supply chains and supplier relationships. Cybersecurity certification of the supply chains can be considered a mitigation action against cybersecurity supply chain risks.

As the threat landscape is enormously evolving, the Regulation (EU) 2019/881 of the European Parliament and the Council, known as the EU Cybersecurity Act (EUCSA) [13], promotes the cybersecurity certification for ICT products (software, hardware, processes, services) and it will scale up the response to cyber-attacks, fostering cyber resilience and trust for consumers within the EU. The EUCSA puts the basis for the creation of the EU certification framework for ICT products.

The EU cybersecurity certification aims at establishing a comprehensive set of rules, technical requirements, standards, and procedures that apply to the certification or Conformity Assessment (CA) of specific ICT products (software, hardware, systems, services). Each certification scheme specifies the categories of products and services covered; the cybersecurity requirements that need to be met -such as standards or technical specifications-, the type of evaluation that is planned to be done - such as a self-assessment or a third-party assessment - and the intended level of assurance that is going to be achieved. The certificates will be valid across all Member States (MSs).

These certification procedures, as well as the general guidelines, protect EU citizens from malicious activities and leakage of personally identifiable information that are used mainly for extracting advertisement profits. These principles are imposed on every organization active on EU soil or have expanded its services to the EU and are expressed in the GDPR regulation.

A certification scheme is much more than assuring that a particular technology is performing according to some security requirements since it demands risk analysis, definition of security controls to mitigate the identified risks, and definition of procedures to monitor some of those controls. SACM with its basic modules, the auditing mechanism and the evidence collection engine, via its reasoning capabilities as described in 4.5 may produce all the necessary evidence that can be used within a certification scheme. Furthermore, the capability that SACM includes, referring that it can support the evaluation of tailored-custom based rules of an Access Control System and report the respectively security events of it, extends and attach importance to the role of the SACM towards a certification procedure.

4.5 Backend – SACM reasoning capabilities

As mentioned before, the Assurance Platform will perform the main automated detection and analysis processes in the backend (supervisory agent). Data gathering is accomplished via Beats and customized Event Captors. Then, the Monitor processes these pieces of information and reasons about the system's current status. This includes the assessment of criteria or policies for security, privacy, or other properties, compliance with Service Level Agreements (SLAs), computation of metrics (e.g., service up-time, mean time to response (MTTResp), mean time to restore MTTRest, etc.), as well as security events.

Indicative use case scenarios are described below. All these access control system rules are assessed simultaneously and continuously while there can be customized or combined in order to tackle more use cases if required through the course of the project. Reports of violation or satisfaction of these rules will produce security events that will inform FISHY project, via its Central Repository, regarding the nature and the status of this specific event.

Document name:	Security and Certification Manager components design and implementation (IT-2)				Page:	27 of 48
Reference:	D4.3	Dissemination:	PU	Version:	1.0	Status: Final

4.5.1 Confidentiality Property Criterion – The users access a service from a set of white-listed IPs

A Filebeat ¹² or Auditbeat ¹³ captures the user interaction with a service (or other resources). If user access is recorded from a different IP, it can be due to some attack that manages to overcome the deployed defenses (e.g., firewall) and infiltrate the system. The EvC theory for this assessment consists of 2 rules:

- **Rule 1:** if there is a call request of the service ($_serviceName$) at some timepoint ($_t$) from a user ($_userName$) who accesses the system from an IP ($_IP$), and this IP is also denoted in the white-list with the legitimate IPs (defined as a fluent), then record a success.

```
Happens(Event(_e, call (_serviceName, _userName, _IP), _t, [_t, _t]) ^ HoldsAt( Fluent(_f, IPsWL
(_serviceName, IPsList), _t, [_t, _t]) ^ Contains(_IPsList, _IP)
=> Initiates(Event(_e), Fluent(SuccessfullUse), _t)
```

- **Rule 2:** if there is a call request of the service ($_serviceName$) at some timepoint ($_t$) from a user ($_userName$) who accesses the system from an IP ($_IP$), and this IP has not been denoted in the white-list with the legitimate IPs (defined as a fluent), then record a violation.

```
Happens(Event(_e, call (_serviceName, _userName, _IP), _t, [_t, _t]) ^ HoldsAt( Fluent(_f, IPsWL
(_serviceName, IPsList), _t, [_t, _t]) ^ ~Contains(_IPsList, _IP)
=> Initiates(Event(_e), Fluent(ViolatedUse), _t)
```

4.5.2 Confidentiality Property & Privacy Criteria – A system resource (e.g., file or service call) is accessed only by a list of authorized users

A Filebeat or Auditbeat captures the resource interaction with the system's users. If an unauthorized user accesses the resource, it can be due to some attack (e.g., privilege escalation) that manages to overcome the deployed authorization techniques. The EC theory for this assessment consists of 2 rules:

- **Rule 1:** if there is access to a resource ($_resourceName$) at some timepoint ($_t$) from a user ($_userName$) and this user also has the access privileges to do so (defined as a fluent), then record a success.

```
Happens(Event(_e, access (_resourceName, _userName), _t, [_t, _t]) ^ HoldsAt( Fluent(_f,
authorizedUsers (_resourceName, _usersList), _t, [_t, _t]) ^ Contains(_usersList, _userName)
=> Initiates(Event(_e), Fluent(SuccessfullUse), _t)
```

- **Rule 2:** if there is access to a resource ($_resourceName$) at some timepoint ($_t$) from a user ($_userName$) and this user has not to have the access privileges to do so (defined as fluent), then record a violation.

```
Happens(Event(_e, access (_resourceName, _userName), _t, [_t, _t]) ^ HoldsAt( Fluent(_f,
authorizedUsers (_resourceName, _usersList), _t, [_t, _t]) ^ ~Contains(_usersList, _userName)
=> Initiates(Event(_e), Fluent(SuccessfullUse), _t)
```

¹² <https://www.elastic.co/beats/filebeat>

¹³ <https://www.elastic.co/beats/auditbeat>

Document name:	Security and Certification Manager components design and implementation (IT-2)				Page:	28 of 48
Reference:	D4.3	Dissemination:	PU	Version:	1.0	Status: Final

4.5.3 Integrity Property Criterion – For every request on a specified service S2, there must have been called the service S1 first

A Filebeat reads the log file of two monitored services, S2 and S1. There is a criterion that the S1 must always be called before S2. If S2 has been called without the prior execution of S1, it can be due to an attack that bypasses the defined workflow or sequence (e.g., an SQL injection that requests data for users who are not logged in the system). The EC theory for this assessment consists of 2 rules:

- **Rule 1:** if there is a call request of the service S2 at some timepoint (t_2) and there is also a relevant call on S1 (this is checked via the other event arguments, which are the same for both events) at a past timepoint ($t_1 + \text{SLA_Threshold}$), then record a success.

```
Happens(Event(_e2, call(_S2, _opInst, _arg1, _arg2), _t2, [_t2, _t2]) ^ Happens( Event(_e1, call(_S1,
_opInst, _arg1, _arg3), _t1, [_0, _t2])
=> Initiates(Event(_e2), Fluent(SuccessfullCall), _t2)
```

- **Rule 2:** if there is a call request of the service ($serviceName$) at some timepoint (t_1) and there is not a relevant response (this is checked via the other event arguments, which are the same for both events) within the acceptable time window ($t_1 + \text{SLA_Threshold}$), then record a violation.

```
Happens(Event(_e2, call(_S2, _opInst, _arg1, _arg2), _t2, [_t2, _t2]) ^ ¬Happens( Event(_e1, call(_S1,
_opInst, _arg1, _arg3), _t1, [_0, _t2])
=> Initiates(Event(_e2), Fluent(ViolationCall), _t2)
```

4.5.4 Integrity Property Criterion – There is only one active login session for each user on a service

A Filebeat reads the log file of a monitored service. There is a criterion that each user can have only one active login. If a user has more than one active session, this could be due to an attacker that has gained access to the system and is currently online. The system operator must further check the sessions. The EC theory for this assessment consists of 2 rules:

- **Rule 1:** if there is a new login in the service ($serviceName$) for a specific user ($userName$), there must have been a recorded logout event for every previous successful login.

```
Happens(Event(_e1, login (_serviceName, _userName, _session1), _t1, [_t1, _t1]) ^ Happens(
Event(_e2, login (_serviceName, _userName, _session2), _t2, [_t1, _t2]) ^ Happens( Event(_e3, logout
(_serviceName, _userName, _session1), _t3, [_t1, _t2])
=> Initiates(Event(_e2), Fluent(SuccessfullCall), _t2)
```

- **Rule 2:** if there is a new login in the service ($serviceName$) for a specific user ($userName$) and a past logged-in session has not been ended yet, then record a violation.

```
Happens(Event(_e1, login (_serviceName, _userName, _session1), _t1, [_t1, _t1]) ^ Happens(
Event(_e2, login (_serviceName, _userName, _session2), _t2, [_t1, _t2]) ^ ¬Happens( Event(_e3, logout
(_serviceName, _userName, _session1), _t3, [_t1, _t2])
=> Initiates(Event(_e2), Fluent(ViolationCall), _t1)
```

Document name:	Security and Certification Manager components design and implementation (IT-2)				Page:	29 of 48
Reference:	D4.3	Dissemination:	PU	Version:	1.0	Status: Final

4.5.5 Availability property SLA – For every request on a specified service, there is a response within a specified time window

A Filebeat reads the log file of a monitored service. There is a Service Level Agreement (SLA) that the service must respond to each request within a specified period. Failure to deliver the service on time could be to congestion, system failure or breakdown, and/or malicious disruption (e.g., flooding attacks). The EC theory for this assessment consists of 2 rules:

- **Rule 1:** if there is a call request of the service (*_serviceName*) at some timepoint (*_t1*) and there is also a relevant response (this is checked via the other event arguments, which are the same for both events) within the acceptable time window (*_t1+SLA_Threshold*), then record a success.

```
Happens(Event(_e1, call (_serviceName, _serviceInst, _arg1, _arg2), _t1, [_t1, _t1]) ^ Happens(
Event(_e2, res (_serviceName, _serviceInst, _arg1, _arg2), _t2, [_t1, _t1+SLA_Threshold]))
=> Initiates(Event(_e2), Fluent(SuccessfullResponse), _t2)
```

- **Rule 2:** if there is a call request of the service (*_serviceName*) at some timepoint (*_t1*) and there is not a relevant response (this is checked via the other event arguments, which are the same for both events) within the acceptable time window (*_t1+SLA_Threshold*), then record a violation.

```
Happens(Event(_e1, call (_serviceName, _serviceInst, _arg1, _arg2), _t1, [_t1, _t1]) ^ ¬Happens(
Event(_e2, res (_serviceName, _serviceInst, _arg1, _arg2), _t2, [_t1, _t1+SLA_Threshold]))
=> Initiates(Event(_e2), Fluent(ViolatedResponse), _t2)
```

4.5.6 Availability property SLA – A service must be available and must not be down for more than a predefined threshold

A Heartbeat or customized Event Captor (i.e., get the HTTP status) that checks the status of a service. There is a Service Level Agreement (SLA) that the service must be up and running, and in case of unavailability, the service administrator/operator has a maximum predefined time window (e.g., 1 hour) to fix the problem and restore the proper operation. Service unavailability could be to congestion, system failure or breakdown, and/or malicious disruption (e.g., Denial of Service (DoS) attack). The EC theory for this assessment consists of 4 rules:

- **Rule 1:** if the status check for a service (*_serviceName*) at some timepoint (*_t1*) is normal, then record a success.

```
Happens(Event(_e, serviceStatus (_serviceName, "Available"), _t, [_t, _t]))
=> Initiates(Event(_e), Fluent(AvailableService, _serviceName), _t)
```

- **Rule 2:** if the status check for a service (*_serviceName*) at some timepoint (*_t1*) is unavailable, then record a violation and start checking against the SLA threshold (Rules 3 and 4).

```
Happens(Event(_e, serviceStatus (_serviceName, "Unavailable"), _t, [_t, _t]) ^ ¬HoldsAt( Fluent(_f,
UnavailableService (_serviceName), _t, [_t, _t]))
=> Initiates(Event(_e), Fluent(UnavailableService, _serviceName), _t)
```

Document name:	Security and Certification Manager components design and implementation (IT-2)					Page:	30 of 48
Reference:	D4.3	Dissemination:	PU	Version:	1.0	Status:	Final

- **Rule 3:** if the service (*_serviceName*) was unavailable and the operation was restored within the predefined time window (*_PreDefinedThreshold*) of the SLA, then record a success.

```
Happens(Event(_e, serviceStatus (_serviceName, "Available"), _t2, [_t2, _t2]) ^ HoldsAt( Fluent(_f, UnavailableService (_serviceName), _t1, [_t1, _t2]) ^ eval(_t2-_t1<=_PreDefinedThreshold)
=> Terminates(Event(_e), Fluent(_f), _t2) ^ Initiates(Event(_e), Fluent(SuccessfulServiceRestore, _serviceName), _t2)
```

- **Rule 4:** if the service (*_serviceName*) was unavailable and the operation was not restored within the predefined time window (*_PreDefinedThreshold*) of the SLA, then record a violation.

```
Happens(Event(_e, serviceStatus (_serviceName, "Available"), _t2, [_t2, _t2]) ^ HoldsAt( Fluent(_f, UnavailableService (_serviceName), _t1, [_t1, _t2]) ^ eval(_t2-_t1>_PreDefinedThreshold)
=> Terminates(Event(_e), Fluent(_f), _t2) ^ Initiates(Event(_e), Fluent(ViolatedServiceRestore, _serviceName), _t2)
```

4.5.7 Integrity and Availability property – Observe potential ransomware activity on system resources

When ransomware is activated, it will start to read and encrypt high volumes of data recursively. A customized Event Captor periodically observes (e.g., 1 minute) the volume of files in a folder accessed within the current time window. If this volume exceeds a predefined threshold (e.g., 100 accesses), notify for potential ransomware activity. The EC theory for this assessment consists of 1 rule:

- **Rule 1:** if the file access volume check for a folder with valuable data (*_folderName*) at some timepoint (*_t*) is beyond a predefined threshold (*_PreDefinedThreshold*), then record a violation.

```
Happens(Event(_e, accessVolume (_folderName, _measuredVolume), _t, [_t, _t]) ^ eval(_measuredVolume > _PreDefinedThreshold)
=> Initiates(Event(_e), Fluent(SuspiciousRansomwareActions, _folderName), _t)
```

4.5.8 Metrics – Compute usage metric

The previous rules can be further extended (mostly the Availability criteria) in order to estimate measurable variables for system or service usage. Indicative examples include i) the total up-time for a monitored period, ii) the mean time to respond, and iii) the mean time to restore. The EC theory for these assessments consists of 6 rules, respectively.

Total up-time for a service:

- **Rule 1:** if there is a new monitoring period (e.g., every year or month), then initiate the total up-time (maintained as the fluent *totalUpTime*) for the service (*_serviceName*).

```
Happens(Event(_e, newPeriod (_serviceName), _t, [_t, _t]) ^ HoldsAt(Fluent(_f, totalUpTime, _serviceName, _value), _t)
=> Terminates(Event(_e), Fluent(_f), _t) ^ Initiates(Event(_e), Fluent(totalUpTime, _serviceName, 0), _t)
```

- **Rule 2:** if an examined service (*_serviceName*) is up and running at some timepoint (*_t*) within the current monitoring period (e.g., running year), then update the total up-time metric accordingly (maintained as the fluent *totalUpTime*).

Document name:	Security and Certification Manager components design and implementation (IT-2)				Page:	31 of 48	
Reference:	D4.3	Dissemination:	PU	Version:	1.0	Status:	Final

```
Happens(Event(_e, serviceStatus (_serviceName, "Available"), _t2, [_t2, _t2]) ^ HoldsAt(Fluent(_f,
totalUpTime, _serviceName, _value), _t1)
=> Terminates(Event(_e), Fluent(_f), _t) ^ Initiates(Event(_e), Fluent(totalUpTime, _serviceName, (_t2
- _t1 + _value), _t)
```

Mean Time To Respond (MTTResp):

- **Rule 1:** if there is a new monitoring period (e.g., every year or month), then initiate the MTTResp (maintained as the fluent *MTTResp*) for the service (*_serviceName*).

```
Happens(Event(_e, newPeriod (_serviceName), _t, [_t, _t]) ^ HoldsAt(Fluent(_f, MTTResp,
_serviceName, _value), _t)
=> Terminates(Event(_e), Fluent(_f), _t) ^ Initiates(Event(_e), Fluent(MTTRep, _serviceName,
func_InitMTTResp(_serviceName)), _t)
```

#func_InitMTTResp is an internal method of the tool in Java that initializes an object that maintains an internal data structure for the response times of the service (*_serviceName*) and calculates the MTTResp. Initially, it is 0.

- **Rule 2:** if there is a new response (*_res*) on a previous call (*_call*) for a service (*_serviceName*), then update the MTTResp accordingly (maintained as the fluent *MTTResp*).

```
Happens(Event(_e2, res (_serviceName, sessionID), _t2, [_t2, _t2]) ^ Happens(Event(_e1, call
(_serviceName, sessionID), _t1, [_t1, _t2]) ^ HoldsAt(Fluent(_f, MTTR, _serviceName, _value), _t1)
=> Terminates(Event(_e2), Fluent(_f), _t2) ^ Initiates(Event(_e2), Fluent(MTTResp, _serviceName, (
func_UpdateMTTResp(_serviceName, _t2 - _t1)), _t2)
```

#func_UpdateMTTResp is an internal method of the tool in Java for the object that maintains the response times of the service (*_serviceName*) in an internal data structure, adds the new response time (*_t2 - _t1*), and calculates the MTTResp.

Mean Time To Restore (MTTRest) – In combination with Availability criteria of subsection 4.5.6:

- **Rule 1:** if there is a new unavailability period for a service (*_serviceName*), then initiate the MTTRest (maintained as the fluent *MTTRest*) metric.

```
Happens(Event(_e, serviceStatus (_serviceName, "Unavailable"), _t, [_t, _t]) ^ ~HoldsAt( Fluent(_f1,
UnavailableService (_serviceName), _t, [_t, _t]) ^ HoldsAt(Fluent(_f2, MTTRest, _serviceName,
_value), _t)
=> Terminates(Event(_e), Fluent(_f2), _t) ^ Initiates(Event(_e), Fluent(MTTRest, _serviceName,
func_InitMTTRest(_serviceName, _t)), _t)
```

#func_InitMTTRest is an internal method of the tool in Java that initializes an object that maintains an internal data structure for the unavailable periods of the service (*_serviceName*). The metric starts to count when the unavailable period is observed (*_t*) and will calculate the MTTRest when the service is later restored.

- **Rule 2:** if a previously unavailable service (*_serviceName*) is now restored, then update the MTTResp accordingly (maintained as the fluent *MTTResp*).

```
Happens(Event(_e, serviceStatus (_serviceName, "Available"), _t2, [_t2, _t2]) ^ HoldsAt( Fluent(_f,
UnavailableService (_serviceName), _t1, [_t1, _t2]) ^ HoldsAt(Fluent(_f, MTTR, _serviceName, _value),
_t1)
```

Document name:	Security and Certification Manager components design and implementation (IT-2)				Page:	32 of 48
Reference:	D4.3	Dissemination:	PU	Version:	1.0	Status: Final

```
=> Terminates(Event(_e2), Fluent(_f), _t2) ^ Initiates(Event(_e2), Fluent(MTTResp, _serviceName, (
func_UpdateMTTRest(_serviceName, _t2 - _t1)), _t2)
```

#func_UpdateMTTRest is an internal method of the tool in Java for the object that maintains the unavailable times of the service (*_serviceName*) in an internal data structure, adds the new restore time (*_t2 - _t1*), and calculates the MTTRest.

Document name:	Security and Certification Manager components design and implementation (IT-2)				Page:	33 of 48
Reference:	D4.3	Dissemination:	PU	Version:	1.0	Status: Final

5 Conclusions

This deliverable presents the work done during the second iteration (IT-2) of the design and implementation of the SCM components, related to tasks T4.1 and T4.2:

- The final design, with architecture updates and final implementations of EDC and SACM components
- Supply chain use cases helped identify new requirements, with no impact to EDC and strong impact to SACM
- The EDC and SACM integration with SCM are described
- The use of multitenancy is evaluated based on the supply chain use case requirements and based on the architectural and development particularities of SACM and EDC
- SACM properties and criteria related to the CIA triad and SLAs are carefully explained with rule code examples

With these updates, SCM components are ready to finalize the integration with WP5.

This is the final deliverable for this task, and the coming efforts related to EDC and SACM, that is, finalize integration, test final version of use case implementation and validation, will be registered in other WPs (WP5 and WP6) deliverables.

This deliverable verifies the completion of milestone:

- MS18: FISHY Sec.&Cert. block components ready for integration (IT-2)

Document name:	Security and Certification Manager components design and implementation (IT-2)				Page:	34 of 48
Reference:	D4.3	Dissemination:	PU	Version:	1.0	Status: Final

6 References

- [1] Sergot, M. (2006). *Knowledge Representation for Event Calculus*. London: Department of Computing Imperial College.
- [2] Mueller, E. T. (2004). Event Calculus Reasoning Through Satisfiability. *Journal of Logic and Computation*, 27.
- [3] Liu, W. a. (2015). Requirements Planning with Event Calculus for Runtime Self-Adaptive System. *IEEE 39th Annual Computer Software and Applications Conference* (pp. 77-82). Institute of Electrical and Electronics Engineers.
- [4] L. Liberti, F. M. (2014). Mathematical programming: Turing completeness and. *J Comb Optim* (pp. 82–104). New York: Springer Science+Business Media .
- [5] Salatino, M. (2006, June 11). *Opensource Knowledge Salaboy*. Retrieved from salaboy web site: <https://salaboy.com/2011/06/06/drools-reteoo-for-dummies-1-intro/>
- [6] IBM. (n.d.). *ibm. Retrieved from ibm Web Site: https://www.ibm.com/docs/en/odmoc?topic=SS7J8H/com.ibm.odm.dserver.rules.designer.run/optimizing_topics/tpc_opt_reteplusalgo.html*
- [7] FICO. (2005, September 26). *Fair Isaac COporation (FICO)*. Retrieved from fico Web site: <https://www.fico.com/blogs/what-rete-iii>
- [8] Doorenbos, R. B. (1995). Production Matching for Large Learning Systems. Defense Technical Information Center
- [9] Berstel, B. (2002). Proceedings Ninth International Symposium on Temporal Representation and Reasoning. *Proceedings Ninth International Symposium on Temporal Representation and Reasoning* (pp. 49-51). Manchester, UK: IEEE.
- [10]Xue, D. L.-P. (2010). Rule Engine based on improvement Rete algorithm. *The 2010 International Conference on Apperceiving Computing and Intelligence Analysis Proceeding* (pp. 346-349). IEEE.
- [11]PATKOS, T. (2016). An event calculus production rule system for reasoning in dynamic and uncertain domains. *Special Issue on the International Web Rule Symposia 2012-2014* (pp. 325-352). Cambridge University Press.
- [12]The Directive on security of network and information systems (NIS Directive). (2020, December 16). An official website of the European Union. Retrieved from Europa: <https://ec.europa.eu/digital-single-market/en/directive-security-network-and-information-systems-nis-directive>
- [13]European Parliament and Council. (2019). Regulation (EU)2019/881 on ENISA and on information and communications technology cybersecurity certification and repealing Regulation (EU) No 526/2013 (Cybersecurity Act).
- [14]Shanahan, M. (1999). The Event Calculus Explained. *Artificial Intelligence Today. Lecture Notes in Computer Science (Lecture Notes in Artificial Intelligence)*. Wooldridge M.J., Veloso M.: Springer, Berlin, Heidelberg.
- [15]Mueller, E. T. (2015). *Commonsense Reasoning - An Event Calculus-Based Approach*. Morgan Kaufmann

Document name:	Security and Certification Manager components design and implementation (IT-2)				Page:	35 of 48
Reference:	D4.3	Dissemination:	PU	Version:	1.0	Status: Final

Annex A. The GUI of the SACM

This subsection presents the integrated GUI of the Security Assurance Solution - Platform to the FiSHY project. Specifically, it presents how a user, after successfully logging into the dedicated GUI of the SACM via the FiSHY dashboard, can start an assessment based on security or privacy criteria for a specific asset and review the results during runtime. Registration of assets can be performed either manually through the GUI (as illustrated in the following figures) or by the Asset Loader Module. The latter is essential for the functionality of the security assessment platform since it contains the definitions of the organizations, assets, and projects inside the organizations, as well as assessment criteria (rules and assumptions used) and assessment profiles. A more detailed description of the Asset Loader was presented in 4.1.3.

Here, it is presented the scenario of checking the availability of a service (i.e., a database). In order to support that, an evidence-collection engine must be deployed when the assessment begins, which periodically examines the status of the latter service. For this scenario, two events are depicted. In the first case, the service is up and running (Criterion Satisfaction). In the second one, the service is down (Criterion Violation).

Initially, in Figure 10, a new project must be created for the assessments of a new organization in the platform.

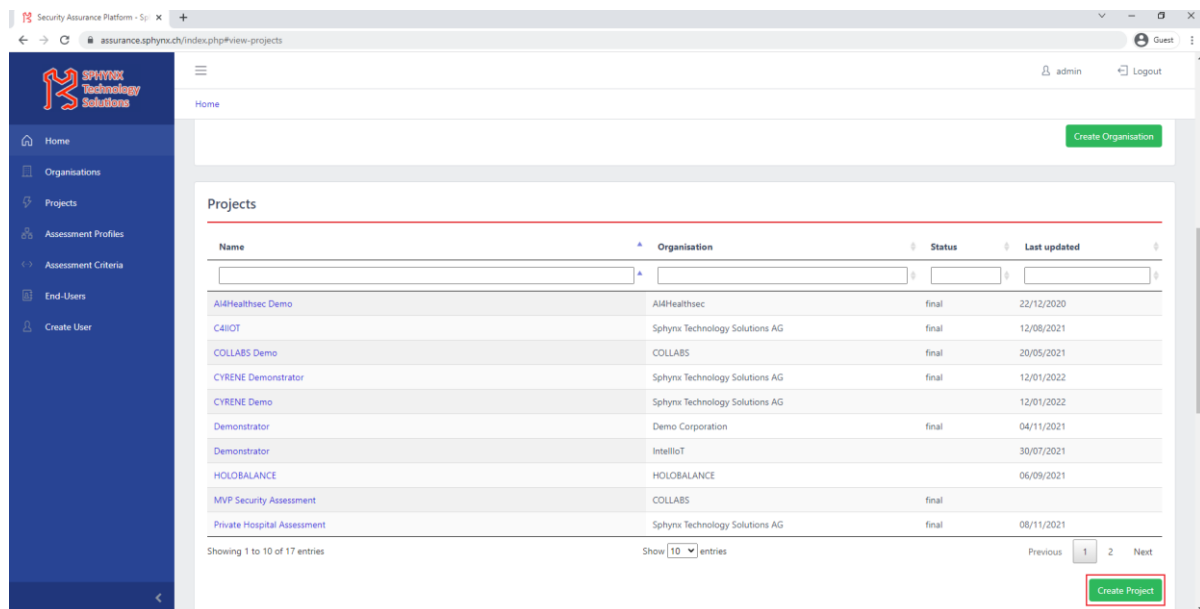
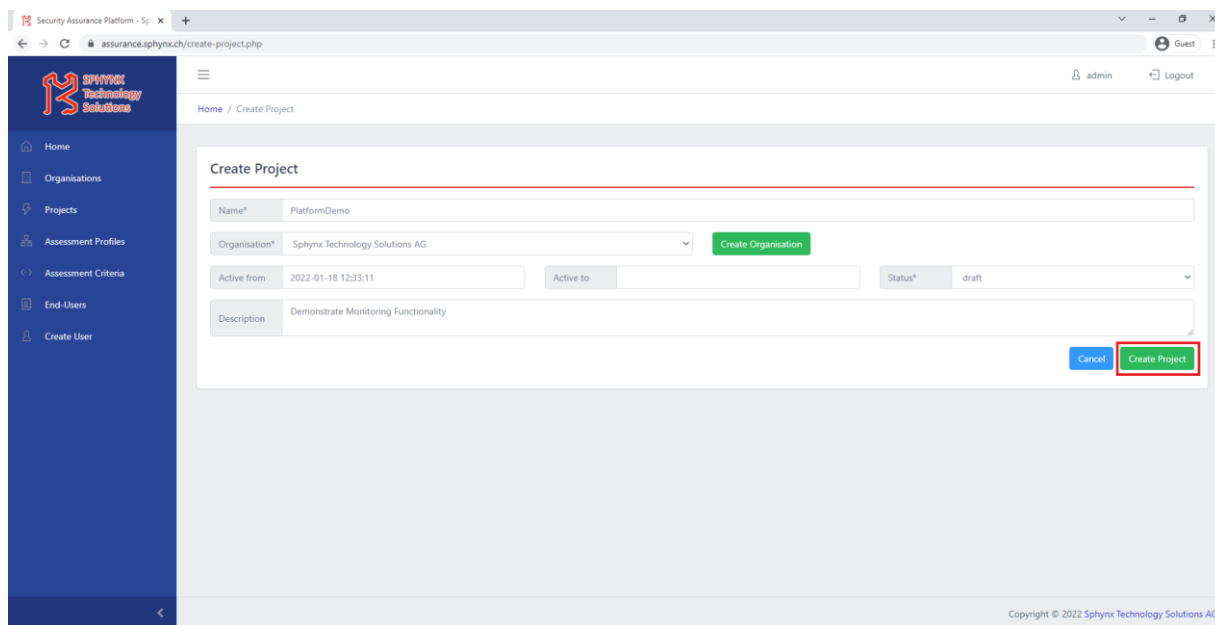


Figure 10: The SACM GUI – Assurance Platform – Start the creation of a New Project

Document name:	Security and Certification Manager components design and implementation (IT-2)					Page:	36 of 48
Reference:	D4.3	Dissemination:	PU	Version:	1.0	Status:	Final

The project's and organization's details are provided, and the project is created and stored (see Figure 11).



Security Assurance Platform - Sphymx Technology Solutions AG

assurance.sphymx.ch/create-project.php

Home / Create Project

Create Project

Name* PlatformDemo

Organisation* Sphymx Technology Solutions AG Create Organisation

Active from 2022-01-18 12:33:11 Active to Status* draft

Description Demonstrate Monitoring Functionality

Cancel Create Project

Copyright © 2022 Sphymx Technology Solutions AG

Figure 11: 0The SACM GUI – Assurance Platform – Create and store Project

From the project's view, the user can monitor the current status of the current assessments for the included assets. Initially, the project is empty (Figure 12).

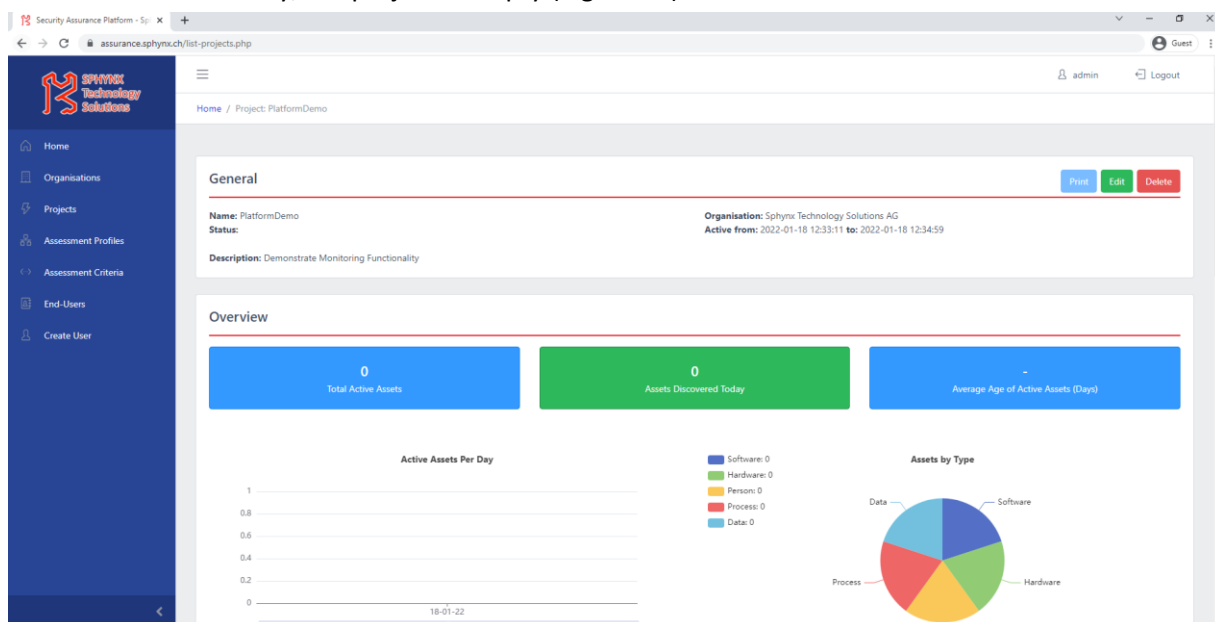


Figure 12: The SACM GUI – Assurance Platform – Project's empty view

Document name:	Security and Certification Manager components design and implementation (IT-2)				Page:	37 of 48	
Reference:	D4.3	Dissemination:	PU	Version:	1.0	Status:	Final

Then, the user can add the organization's assets. With the manual option (see Figure 13), the user chooses to create a new asset and provide the requested details.

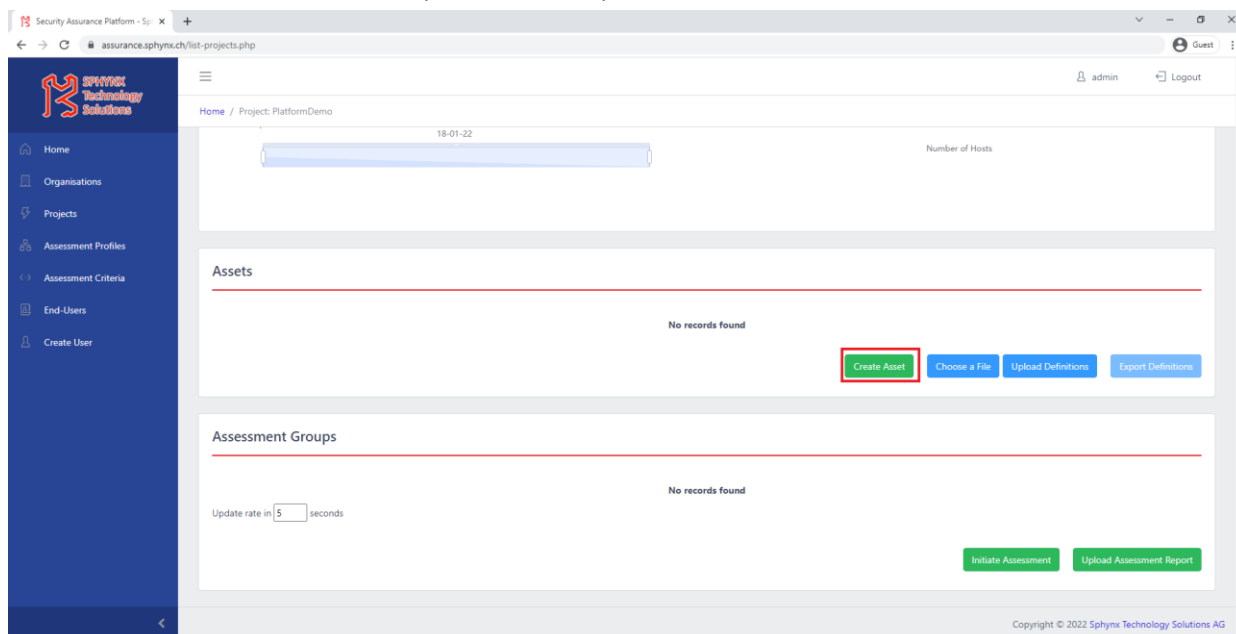


Figure 13: The SACM GUI – Assurance Platform – Start the creation of a New Asset

In Figure 14, the user chooses to create a new software asset (i.e., a database).

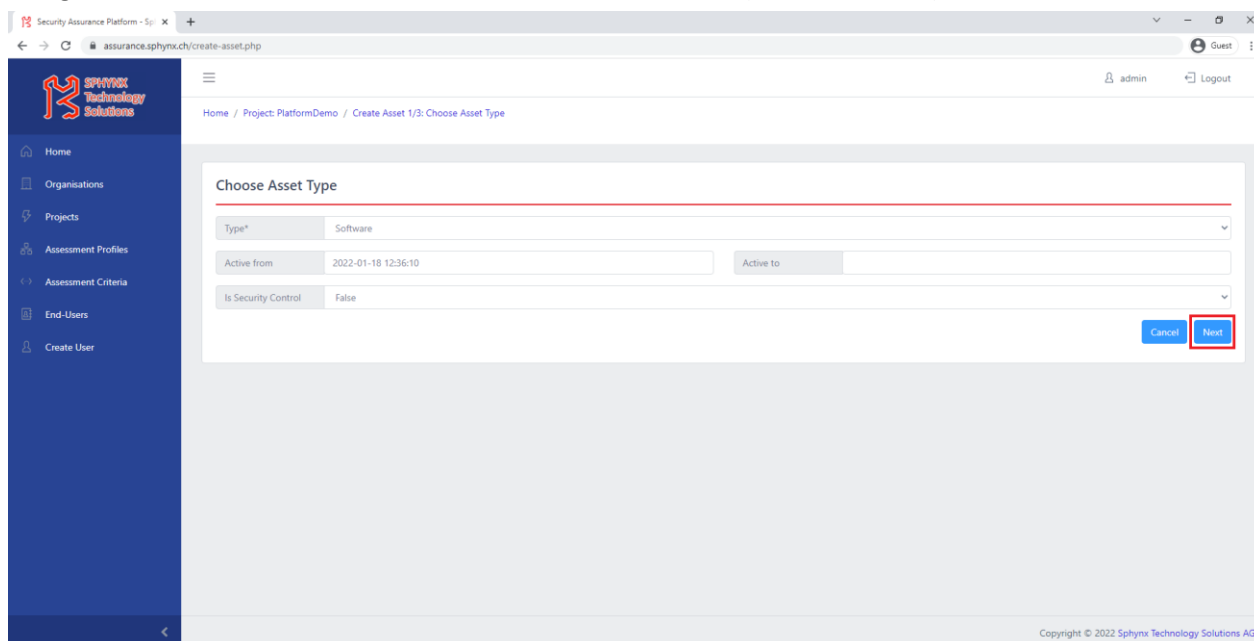
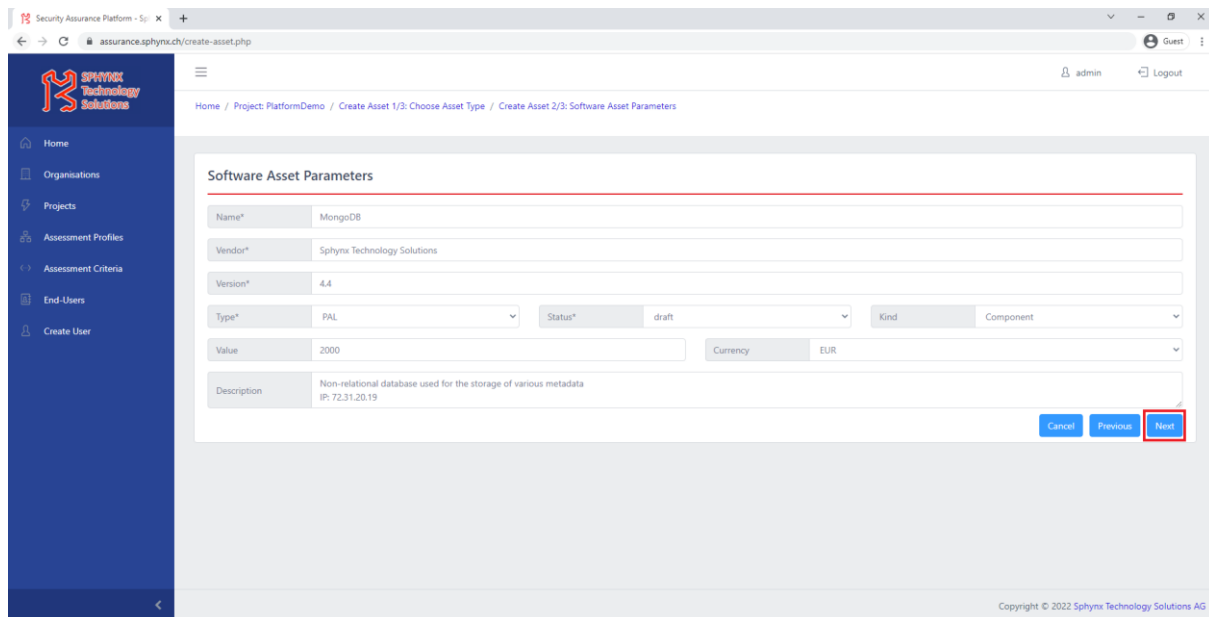


Figure 14: The SACM GUI – Assurance Platform – Provide the type of the asset

Document name:	Security and Certification Manager components design and implementation (IT-2)				Page:	38 of 48	
Reference:	D4.3	Dissemination:	PU	Version:	1.0	Status:	Final

Then, the user is requested to provide the details for the specific type of asset. In Figure 15, the user gives the descriptive information for the database to be monitored, including the IP address of the web interface to be checked.



Security Assurance Platform - Sphynx Technology Solutions

Home / Project: PlatformDemo / Create Asset 1/3: Choose Asset Type / Create Asset 2/3: Software Asset Parameters

Software Asset Parameters

Name* MongoDB

Vendor* Sphynx Technology Solutions

Version* 4.4

Type* PAL Status* draft Kind* Component

Value 2000 Currency EUR

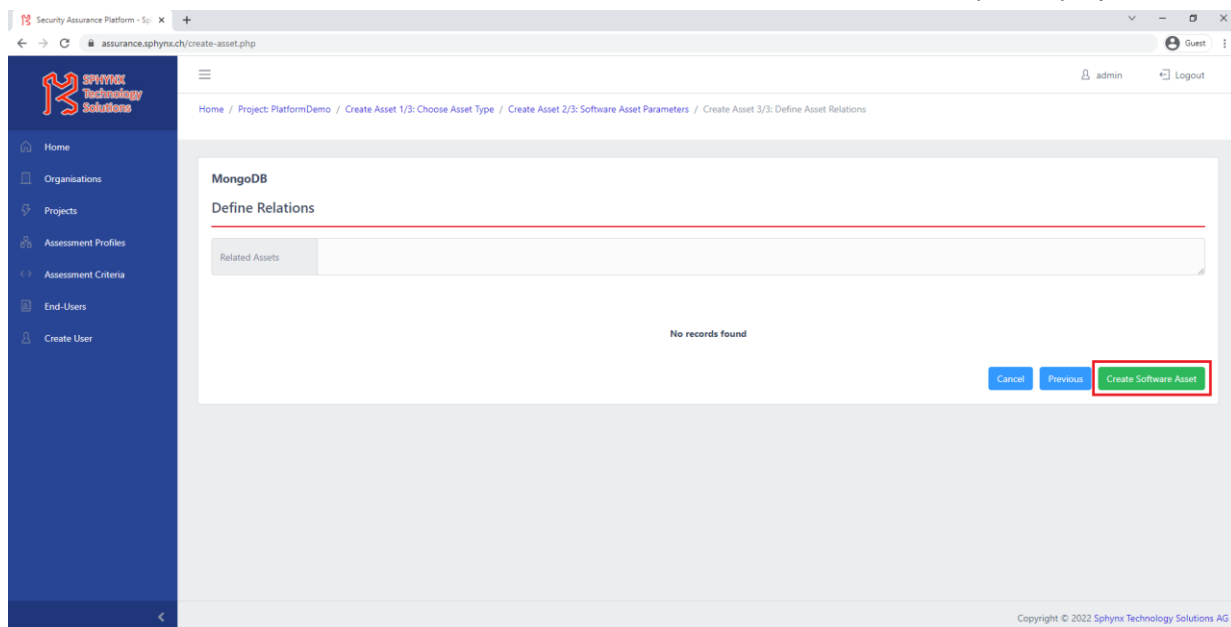
Description Non-relational database used for the storage of various metadata IP: 72.31.20.19

Cancel Previous **Next**

Copyright © 2022 Sphynx Technology Solutions AG

Figure 15: The SACM GUI – Assurance Platform – Provide details for a software asset

The user can also declare potential relations/dependencies with other existing assets. Then, the asset is created and stored in the internal database of the Assurance Platform for this specific project.



Security Assurance Platform - Sphynx Technology Solutions

Home / Project: PlatformDemo / Create Asset 1/3: Choose Asset Type / Create Asset 2/3: Software Asset Parameters / Create Asset 3/3: Define Asset Relations

MongoDB Define Relations

Related Assets

No records found

Cancel Previous **Create Software Asset**

Copyright © 2022 Sphynx Technology Solutions AG

Figure 16: The SACM GUI – Assurance Platform – Create and store asset

Document name:	Security and Certification Manager components design and implementation (IT-2)				Page:	39 of 48	
Reference:	D4.3	Dissemination:	PU	Version:	1.0	Status:	Final

Back in the project's view, the user can view the included assets for this assessment project (Figure 17).

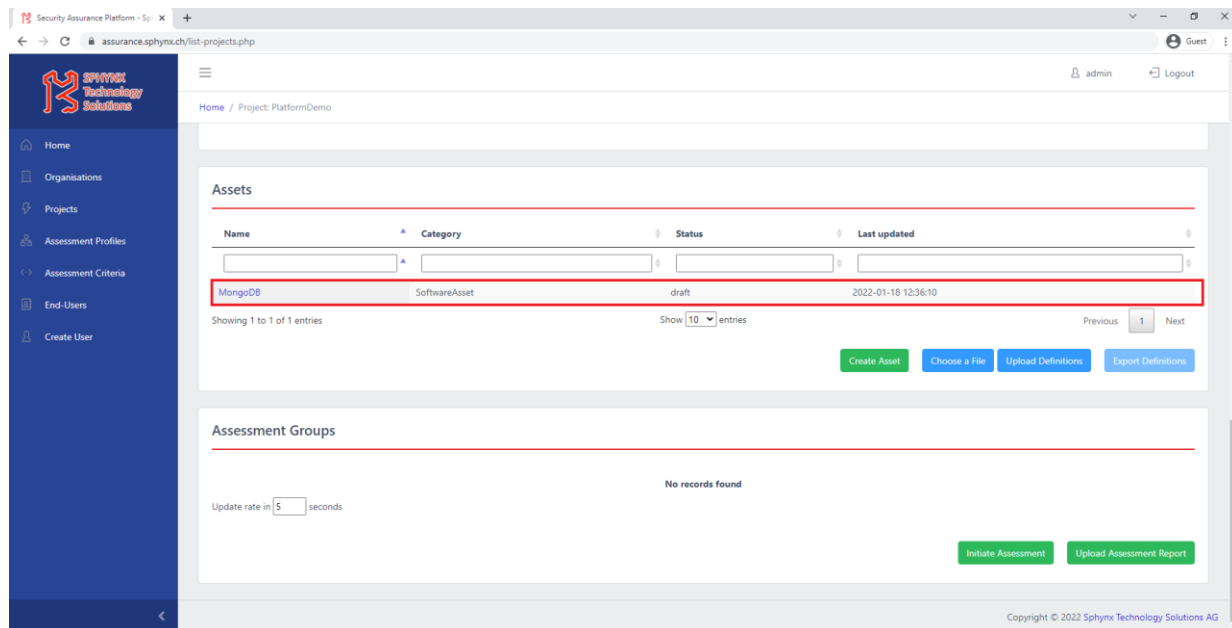


Figure 17: The SACM GUI – Assurance Platform – Created assets in the project's view

Now, the user can initiate a new monitoring assessment (Figure 18).

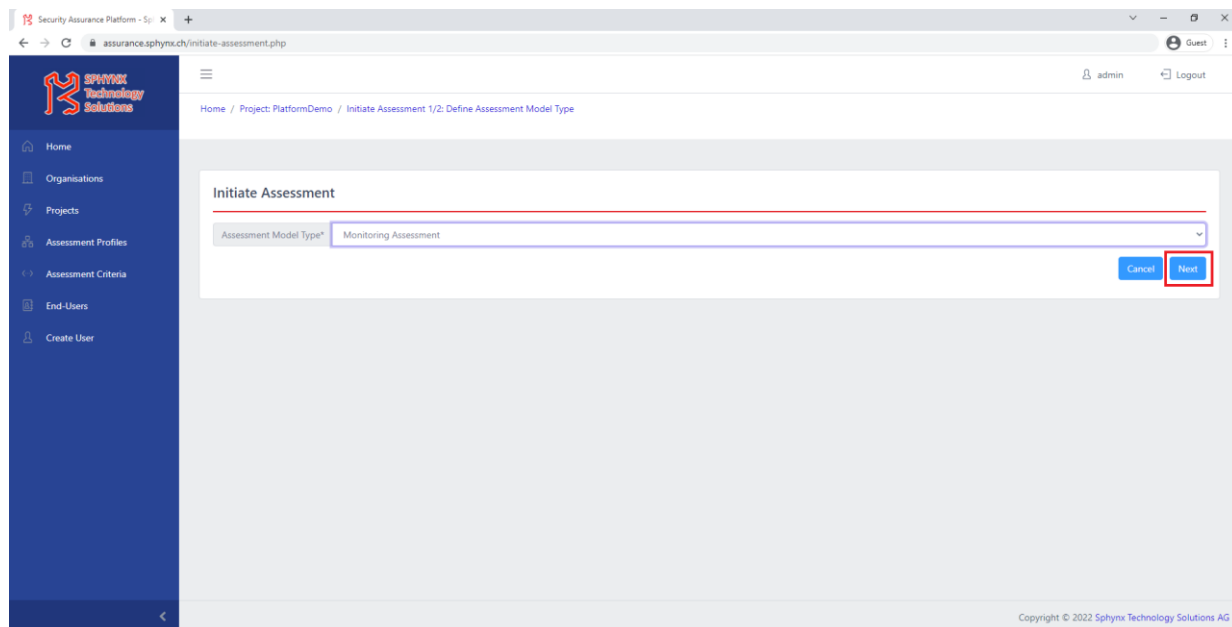


Figure 18: The SACM GUI – Assurance Platform – Initiate a monitoring assessment

Document name:	Security and Certification Manager components design and implementation (IT-2)				Page:	40 of 48	
Reference:	D4.3	Dissemination:	PU	Version:	1.0	Status:	Final

At first, the user selects the assessment profile that will be applied. The profile is formed on a set of rules that implement the logic of the assessment. In Figure 19, an availability profile is selected, which will be continuously recorded wherever the database's web interface is available or not (see Figure 20).

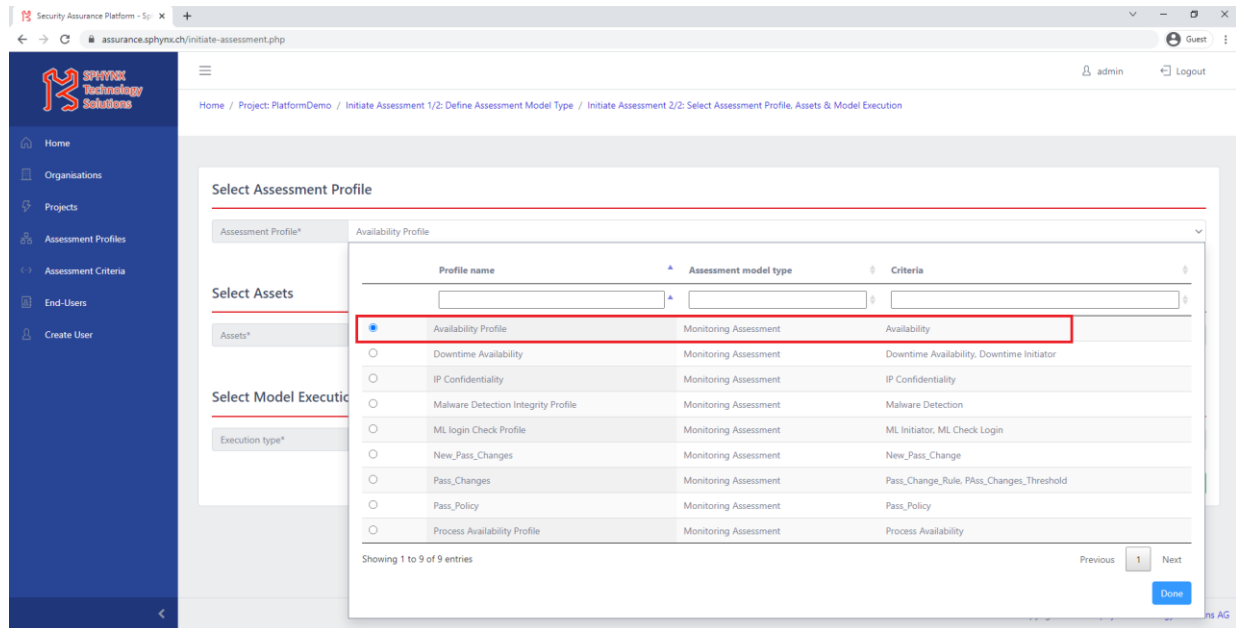


Figure 19: The SACM GUI – Assurance Platform – Select an availability profile

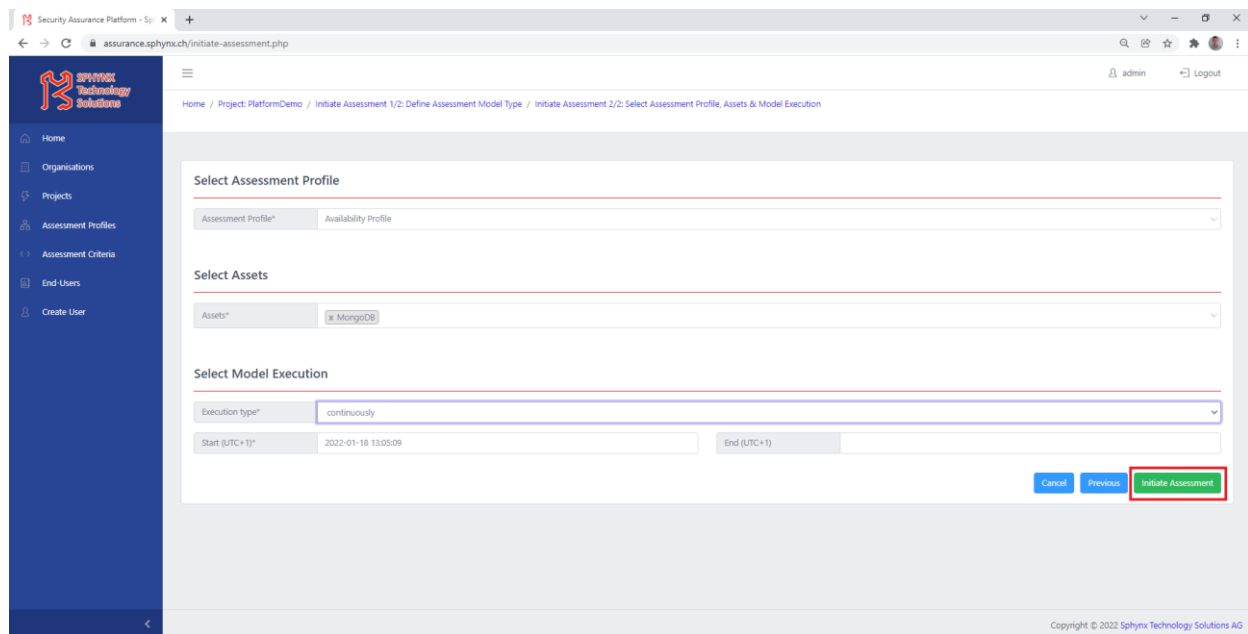


Figure 20: The SACM GUI – Assurance Platform – Select the asset and the model execution type of the assessment

Document name:	Security and Certification Manager components design and implementation (IT-2)				Page:	41 of 48	
Reference:	D4.3	Dissemination:	PU	Version:	1.0	Status:	Final

Back in the project's view, the user can monitor the running assessment profiles of the project (Figure 21).

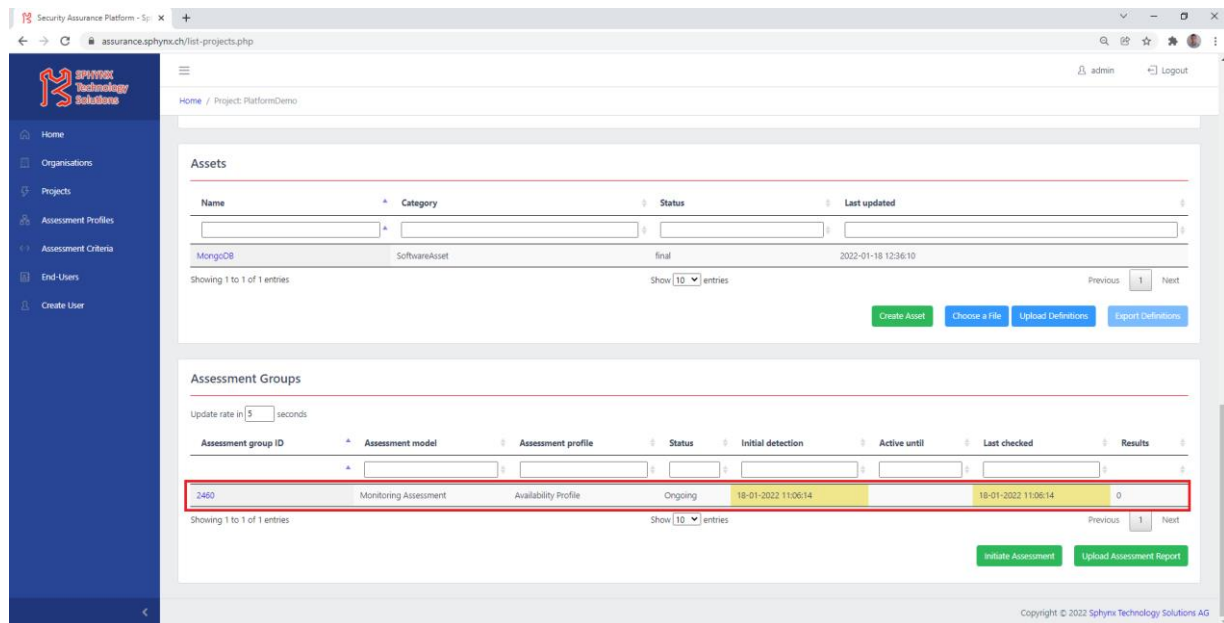


Figure 21: The SACM GUI – Assurance Platform – Created assessment profiles in the project's view

Thereupon, an Event Captor has been deployed, which is periodically checking (e.g., every second) the HTTP status of the database's web interface, and sends relevant EC events to the Monitor. The rule set of the availability assessment profile is triggered and records the result. Figure 22 depicts the reasoning for a successful event. The Captor checked the status of the web interface, and it was up and running.

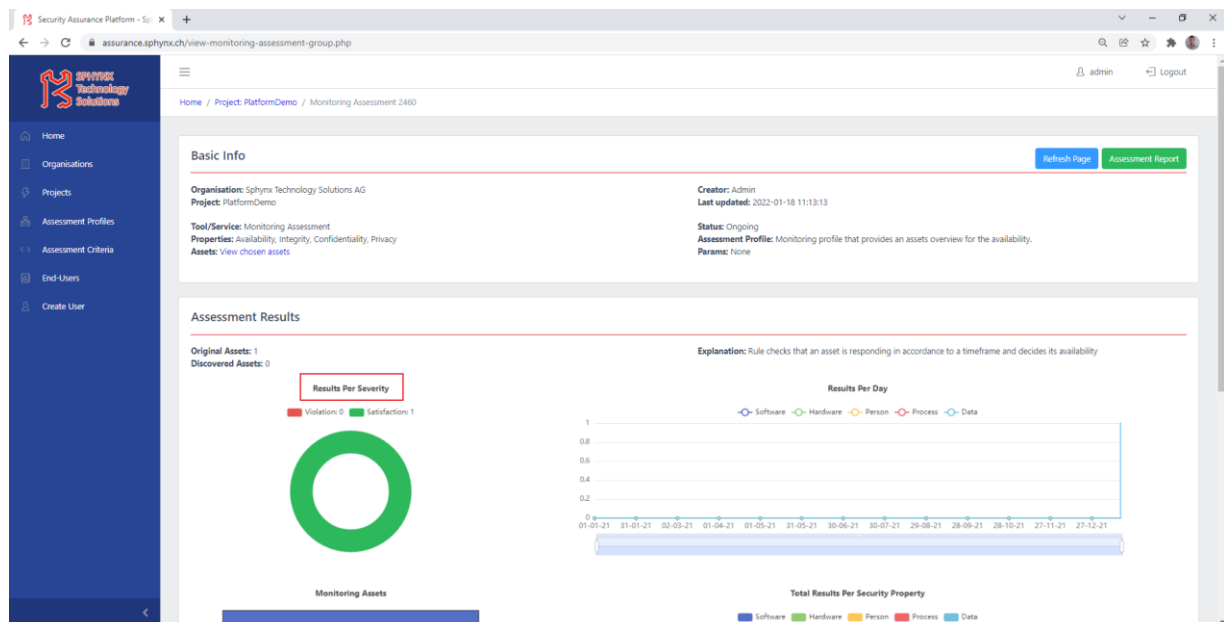


Figure 22: The SACM GUI – Assurance Platform – Assessment Result overview

Document name:	Security and Certification Manager components design and implementation (IT-2)				Page:	42 of 48	
Reference:	D4.3	Dissemination:	PU	Version:	1.0	Status:	Final

The user can also review the evidence and the detailed events for this result (see Figure 23).

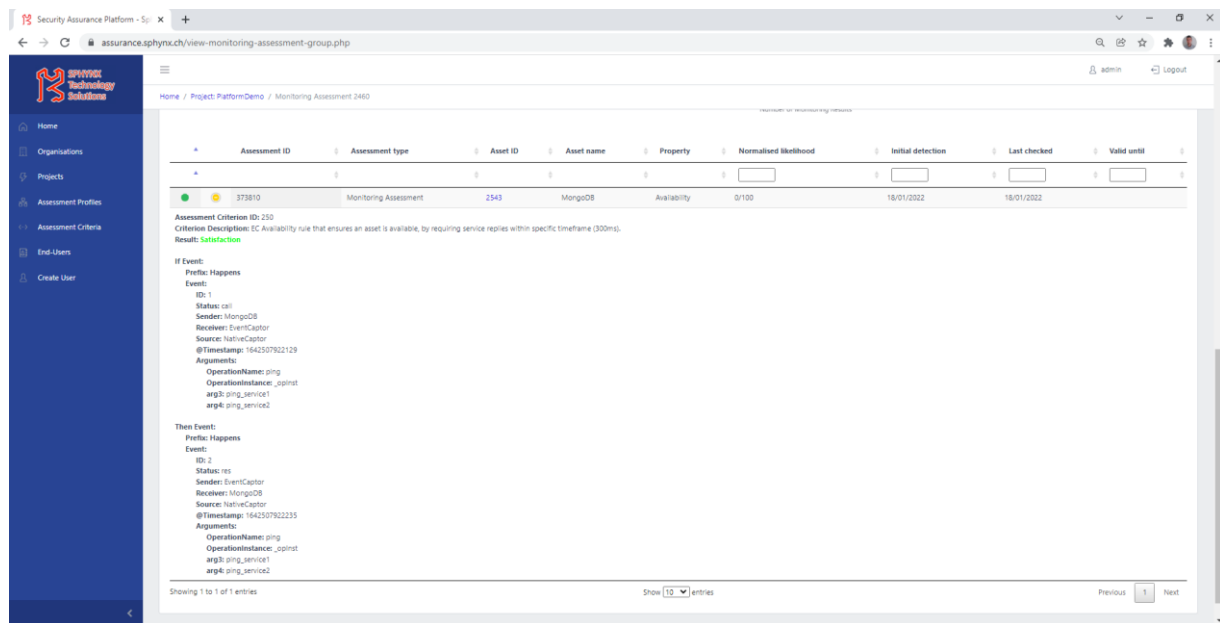


Figure 23: The SACM GUI – Assurance Platform – Availability of successful event details

Similarly, the Captor checks the status for the second time when the service is down. The violation event for the availability criterion is also depicted in Figure 24.

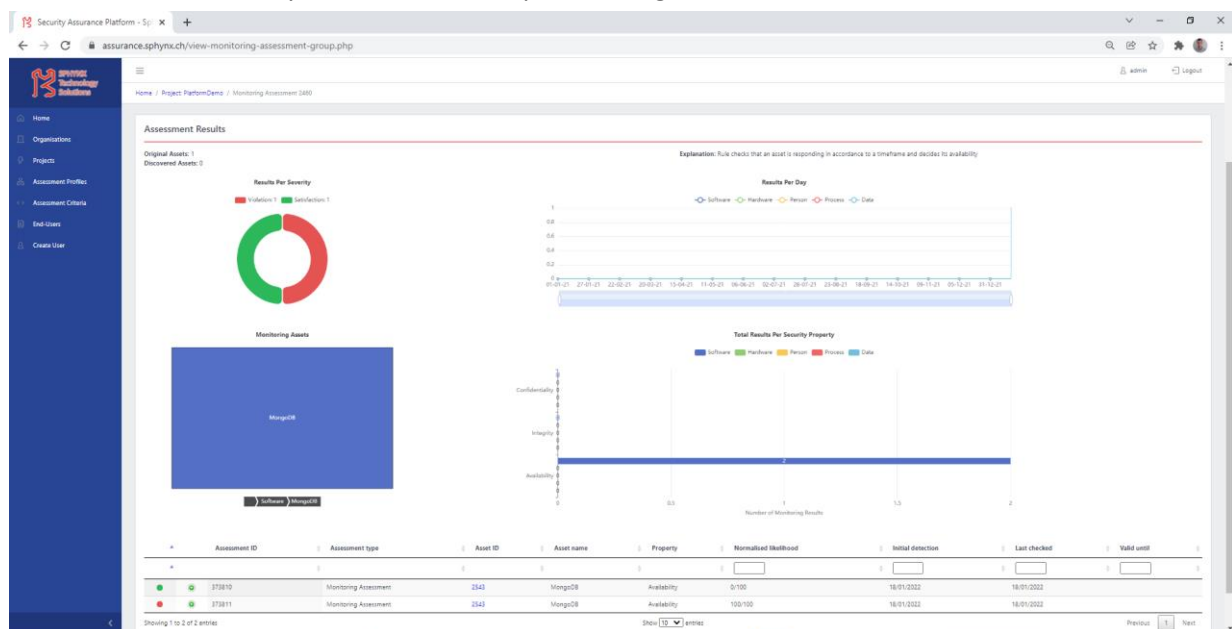


Figure 24: The SACM GUI – Assurance Platform – Assessment Results overview

Document name:	Security and Certification Manager components design and implementation (IT-2)				Page:	43 of 48	
Reference:	D4.3	Dissemination:	PU	Version:	1.0	Status:	Final

Figure 25 depicts the evidence and the detailed events for the violation result.

	Assessment ID	Assessment type	Asset ID	Asset name	Property	Normalised likelihood	Initial detection	Last checked	Valid until
●	373810	Monitoring Assessment	2543	MongoDB	Availability	0/100	18/01/2022	18/01/2022	
●	373811	Monitoring Assessment	2543	MongoDB	Availability	100/100	18/01/2022	18/01/2022	

Assessment Criterion ID: 250
 Criterion Description: EC Availability rule that ensures an asset is available, by requiring service replies within specific timeframe (300ms).
 Result: **Violation**

If Event:
 Prefix: Happens
 Event:
 ID: 3
 Status: call
 Sender: MongoDB
 Receiver: EventCaptor
 Sources: NativeCaptor
 @Timestamp: 164250792329
 Arguments:
 OperationName: ping
 OperationInstance: _opinst
 arg0: ping_service1
 arg1: ping_service2

Then Event: Expected predicates were not fulfilled.

Showing 1 to 2 of 2 entries

Show entries

Previous Next

Figure 25: The SACM GUI – Assurance Platform – Availability violation event details

Document name:	Security and Certification Manager components design and implementation (IT-2)				Page:	44 of 48	
Reference:	D4.3	Dissemination:	PU	Version:	1.0	Status:	Final

Annex B. Event Calculus Axioms

As for the Event Calculus Axioms, explained thoroughly in already published research [14], they are concluding the abovementioned logic. The following four axioms illustrate the basic structure of Event Calculus as it can be defined.

(EC1)	$\text{Clipped}(t1, f, t2) \Leftarrow (\exists e, t) \text{ Happens}(e, t, \mathfrak{R}(t1, t2))$ $\quad \quad \quad \wedge \text{Terminates}(e, f, t)$
(EC2)	$\text{HoldsAt}(f, t) \Leftarrow \text{Initially}(f) \wedge \neg \text{Clipped}(0, f, t)$
(EC3)	$\text{HoldsAt}(f, t) \Leftarrow (\exists e, t1) \text{ Happens}(e, t, \mathfrak{R}(t1, t))$ $\quad \quad \quad \wedge \text{Initiates}(e, f, t1)$ $\quad \quad \quad \wedge \neg \text{Clipped}(t1, f, t)$
(EC4)	$\text{Happens}(e, t, \mathfrak{R}(t1, t2)) \Rightarrow (t1 < t) \wedge (t \leq t2)$

Figure 26: The four axioms of the basic structure of Event Calculus

For the first Axiom (EC1), we define a Clipped predicate, which refers to a predicate that ‘locks’ the logical interpretation of a state, meaning that we cannot change the value of a fluent (EC2) when another value is meant to be Terminated by a specific event (Happens). If we observe the bigger picture for these axioms, we can conclude that they are describing the process of changing a HoldsAt value (fluent); in this, we insert the logic that “a fluent cannot be changed when the fluent is in the process of Termination or Initiation (EC3) by a specific event that demands this alternation”.

The EvC supports context-sensitive effects of events, indirect effects, action preconditions, and the common sense law of inertia [15]. Certain phenomena are addressed more naturally in the event calculus, including concurrent events, continuous time, continuous change, events with duration, non-deterministic effects, partially ordered events, and triggered events. Examples of such phenomena could be:

- The commonsense law of inertia: when moving a glass does not cause a glass in another room to move.
- Release from the commonsense law of inertia: if a person is holding a PDA (Personal Digital Assistant), then the location of the PDA is released from the commonsense law of inertia so that the location of the PDA is permitted to vary.
- Event ramifications or indirect effects of events: the PDA moves along with the person holding it (state constraint) or instantaneous propagation of interacting indirect effects, as in idealized electrical circuits (casual constraints).
- Conditional effects of events: the results of turning on a television depend on whether it is plugged in or not.
- Events with non-deterministic effects: flipping a coin results in the coin landing either heads or tails.
- Gradual change: the changing height of a falling object or volume of a balloon in the process of inflation.

The EC contains a set of fluents, a set of events, and a partially ordered set of time points. In the EC, the description of the worlds (possible scenarios) is based on the following axiom (assume e is an event, f is a fluent, and $t, t1$, and $t2$ are time points):

- $\text{Initiates}(e, f, t)$: f holds after event e at time t
- $\text{Terminates}(e, f, t)$: f does not hold after event e at time t
- $\text{Initially}P(f)$: f holds from time 0

Document name:	Security and Certification Manager components design and implementation (IT-2)					Page:	45 of 48
Reference:	D4.3	Dissemination:	PU	Version:	1.0	Status:	Final

- InitiallyN(f): f does not hold from time 0
- Happens(e, t1, t2): event e start at time t1 and ends at t2
- HoldsAt(f, t): f holds at time t
- Clipped(t1, f, t2): f is terminated between t1 and t2
- Declipped(t1, f, t2): f is initiated between t1 and t2

Document name:	Security and Certification Manager components design and implementation (IT-2)				Page:	46 of 48
Reference:	D4.3	Dissemination:	PU	Version:	1.0	Status: Final

Annex C. Drools Rule Syntax example

By using an example to describe the conflict resolution capabilities, we can ideate two Rules that are correlated and give priority to RuleA. So, in order to examine the salience property, we create a dummy example of a minified confidentiality property for the user's IP in Drools will look something like this:

```
Global StringArray white-list = ['8.8.8.8', '63.43.80.92', '195.32.45.12']
```

```
Rule RuleA salience 20
```

```
  when
```

```
    A($user, IP)
```

```
  then
```

```
    new B(loggedin=True, loggedip= IP)
```

```
end
```

```
Rule RuleB_Violation
```

```
  When
```

```
    A($user, IP)
```

```
    B(loggedin==True, loggedip == IP, whitelistip not contains loggedip)
```

```
  then
```

```
    new Alert('User '+$user+'logged in from anauthorised ip!')
```

```
    retract(A,B)
```

```
end
```

```
Rule RuleB_Satisfaction
```

```
  When
```

```
    A($user, IP)
```

```
    B(loggedin==True, loggedip == IP, whitelistip contains loggedip)
```

```
  then
```

```
    new Alert('User '+$user+' logged in from anauthorised ip!')
```

```
    retract(A,B)
```

```
end
```

To explain the context of this example, we define the white-list array as a global variable that can be accessed from all the presented rules, containing some predefined IPs as values. This variable represents all the eligible IPs that a user can log in from. Thus, if we now add an A('63.43.80.92') object into this logic session, the RuleA will first fire, evaluating that there is an A() Object that has the IP and the username field that is required, so it will evaluate the condition as true and will continue to the <action> part of the RuleA. The 'then' part of RuleA rule will create a new B() Object that contains the variables shown in the example (Boolean loggedin, String IP, and String username).

Now, there is a B() object with certain values in the logic session, thus, it will evaluate the second rule, RuleB_Satisfaction. We can see that the IP provided into the B() Object is not inquired into the global white-list array of IPs, so we trigger an alert. That will highlight in the monitoring assessment result that another user logged in did not comply with our confidentiality security policy. Another element

Document name:	Security and Certification Manager components design and implementation (IT-2)				Page:	47 of 48
Reference:	D4.3	Dissemination:	PU	Version:	1.0	Status: Final

that we can notice in the simple example presented is the retract command presented in both of the rules. This command is used to delete an object from the logic session.

We can observe that there is another rule in our example, RuleB_Violation. This rule is fired when an IP that is not contained in the white-list is inserted into the logic session. Consequently, after the initial run that provided us with a satisfaction alert as a monitoring result, it is inserted into the logic session in case A('23.12.31.02'). The first rule will be inserted into RuleA and will successfully continue to the 'then' part. In the second phase, though, the rule RuleB_Satisfaction <condition> part is violated, so it won't surpass the 'then' part of the rule, unlike the rule RuleB_Violation <condition> part, which is satisfied and produces the defined alert for the violation of the security policy.

Focusing more on the Rete algorithm, we can observe how it handles the facts and depict the example's outcome as logical steps. When defining the nodes, we can present Rete's algorithm terminology.

Document name:	Security and Certification Manager components design and implementation (IT-2)				Page:	48 of 48
Reference:	D4.3	Dissemination:	PU	Version:	1.0	Status: Final