Secure Provisioning of Cloud Services
based on SLA Management

SPECS Project - Deliverable 4.5.3

Testing and validation - Finalized

Version no. 1.1
19 July 2016

The activities reported in this deliverable are partially supported
by the European Community's Seventh Framework Programme under grant agreement no. 610795.

Deliverable information

Deliverable no.:	D4.5.3
Deliverable title:	Testing and validation – Finalized
Deliverable nature:	Report
Dissemination level:	Public
Contractual delivery:	19 July 2016
Actual delivery date:	19 July 2016
Author(s):	Jolanda Modic (XLAB), Miha Stopar (XLAB)
Contributors:	Damjan Murn (XLAB), Aljaž Košir (XLAB), Silviu Panica (IeAT), Massimiliano Rak (CeRICT), Alessandra De Benedictis (CeRICT), Giancarlo Capone (CeRICT)
Reviewers:	Stefano Marrone (CeRICT), Andrew Byrne (EMC), Rubén Trapero (TUDA)
Task contributing to the deliverable:	T4.5
Total number of pages:	120

Executive summary

This deliverable is the final document presenting the validation and testing of the Enforcement module. Its main focus is to summarize the adopted testing and validation techniques, validate all components of the Enforcement module, and present tests for the associated prototypes.

In particular, this document presents:

- Validation and testing techniques: We summarize the validation and testing approach defined in the previous iterations of this deliverable (namely D4.5.1 and D4.5.2).
- Validation of the Enforcement module: We present the results of the Enforcement validation process. We analyse validation scenarios and requirements associated with the Enforcement module and discuss their coverage.
- Functional testing of the Enforcement module: We present and discuss all unit tests executed for the verification of the implementation of the core Enforcement components as well as the security mechanisms developed in task T4.3. Additionally, for each component we present a code quality report.
- Performance and scalability analysis: We present and discuss all tests executed for the analysis of the performance and scalability aspects of the developed Enforcement components.

All aspects associated with integration of the Enforcement components with the SPECS framework are discussed in deliverables D1.5.1 and D1.5.2. A security review was conducted on the framework and at the application level which is available in D1.5.2.

Table of contents

Index of figures

Index of tables

1. Introduction

The Enforcement module plays a very important role in the SPECS framework. Not only that it orchestrates many crucial steps in the SLA life-cycle, namely the SLA implementation and the SLA remediation phase, but it also comprises a set of security mechanisms that enhance the security level of the negotiated cloud services.

As defined in deliverable D4.2.2 and depicted in Figure 1, the Enforcement module comprises the following set of core components:

- **Planning:** Builds supply chains according to security requirements provided by the End-user and resource properties provided by SPECS and Cloud Service Providers (CSPs), and generates implementation plans according to the signed SLAs.
- **Implementation:** Acquires and configures resources according to implementation plans, and reconfigures resources according to remediation plans.
- **Diagnosis:** Analyses notified monitoring events that potentially represent SLA violations.
- **Remediation Decision System (RDS):** Prepares remediation plans, i.e., identifies countermeasures needed to be taken to mitigate the risk of having SLA violations or to recover from them.

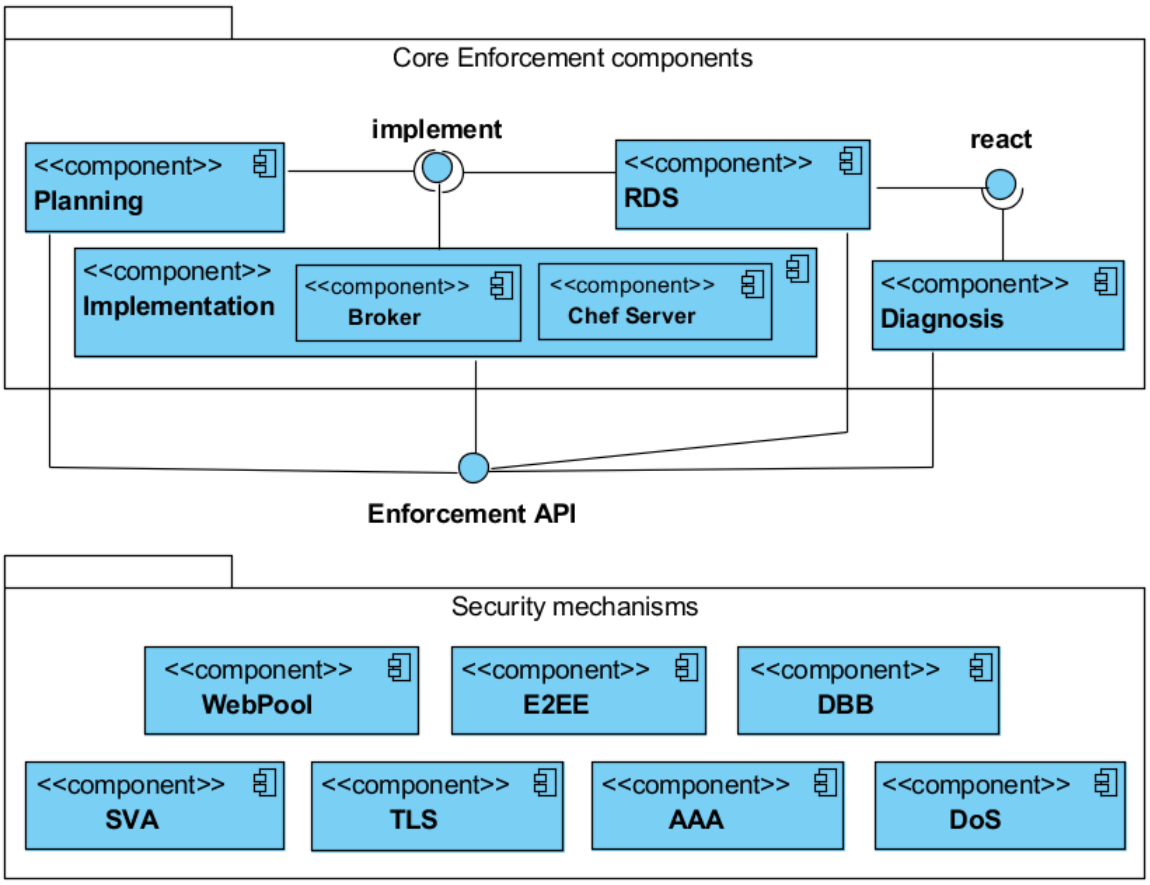


Figure 1. Enforcement module design

Additionally, as discussed in deliverable D4.2.2 and shown in Figure 1, the Enforcement module comprises the following security mechanisms:

- **WebPool:** Provides pools of web servers and assures resilience to security incidents through redundancy and diversity.
- **TLS:** Ensures communication privacy with a set of possible configurations for the TLS protocol (such as cryptographic strength, certificate pinning, HTTP to HTTPS redirection, etc.).
- **SVA:** Offers evaluation of the security level of the system achieved through periodic vulnerability scans and reports about available updates and upgrades of vulnerable libraries on the system.
- **DBB:** Provides storage and assures business continuity through backup. Moreover, it enforces and monitors write-serializability and read-freshness.
- **E2EE:** Offers end-2-end encryption to guarantee security and integrity of the stored data within the secure storage service.
- **DoS:** Provides functionalities for the detection and mitigation of Denial of Service attacks.
- **AAA:** Provides federated identity and access management features over resources and services of different applications.

In the previous iterations of this deliverable, we reported the results of the intermediate verification of all components and mechanisms of the Enforcement module, and presented some initial unit tests and code quality reports. The final validation, code quality analysis, and the entire set of tests executed for the Enforcement module are presented in this document.

Note that the defined approach to the testing is also adopted in other workpackages, i.e., for other modules.

The document is structured as follows. In Section 2, relationships between this document and other deliverables of the SPECS project are discussed. Section 3 briefly summarizes the SPECS validation and testing approach. The final validation of the Enforcement module is presented in Section 4, and functional tests and code quality reports associated to its components are reported in Section 5. The performance and scalability analysis is presented in Section 6. The document concludes with a brief summary in Section 7. For the purpose of providing the reader with the supporting material for the validation results, we report the list of requirements and validation scenarios associated to the Enforcement module in Appendix 1 and Appendix 2, respectively.

2. Relationship with other deliverables

Validation and testing activities presented in this document are based on three sets of input. First, the final design and prototypes of the Enforcement module described in deliverables D4.3.2 and D4.3.3 are considered along with the APIs defined in deliverable D1.3. Then, the requirements and validation scenarios presented in D4.1.2 and D5.1.2, respectively, are considered. Finally, the validation and testing approach defined in deliverable D4.5.2 is taken into account.

The results discussed in this document serve as an input for the implementation activities in task T4.3 (i.e., deliverable D4.3.3) and integration activities in task T1.5 (i.e., deliverables D1.5.1 and D1.5.2).

The discussed relationships are depicted in Figure 2.

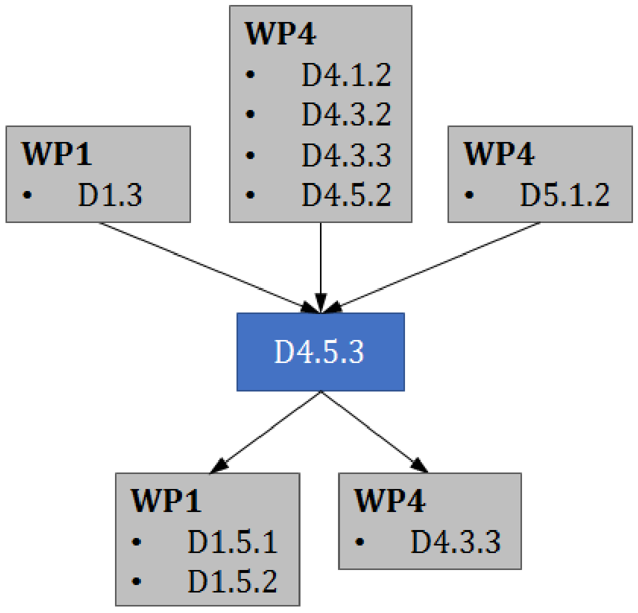


Figure 2. Relationship with other deliverables

3. Validation and testing methodologies

In the previous iterations of this deliverable, we defined the validation and testing methodology to be adopted in SPECS. Namely, in D4.5.1 we presented the initial methodology for validation and explored testing techniques and tools, whereas in D4.5.2 we finalized the validation technique with more details and determined the final testing approach.

The development process adopted in SPECS is depicted in Figure 3. The first step in the process (*Specification*) encompassed the definition of validation scenarios and the specification of functional and non-functional requirements. Afterwards, in the *Design* step, the initial architecture of the entire framework was set, along with all interactions and interfaces. After the initial implementation and testing of the prototypes, the first cycle of verification was performed in the *Coding* step. In year 2 of the project, the design was refined according to the implementation, testing, and verification feedback (collected in the *Verification* step). Refinements were implemented and verified. During all these phases, we used versioning software systems and tracked issues as part of the *Operation & Maintenance* step.

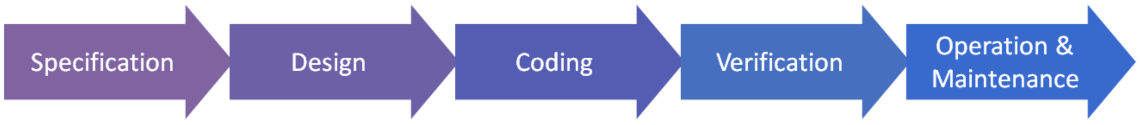


Figure 3. SPECS development process

When it comes to defining the set of validation and testing techniques, we followed the process depicted in Figure 4.

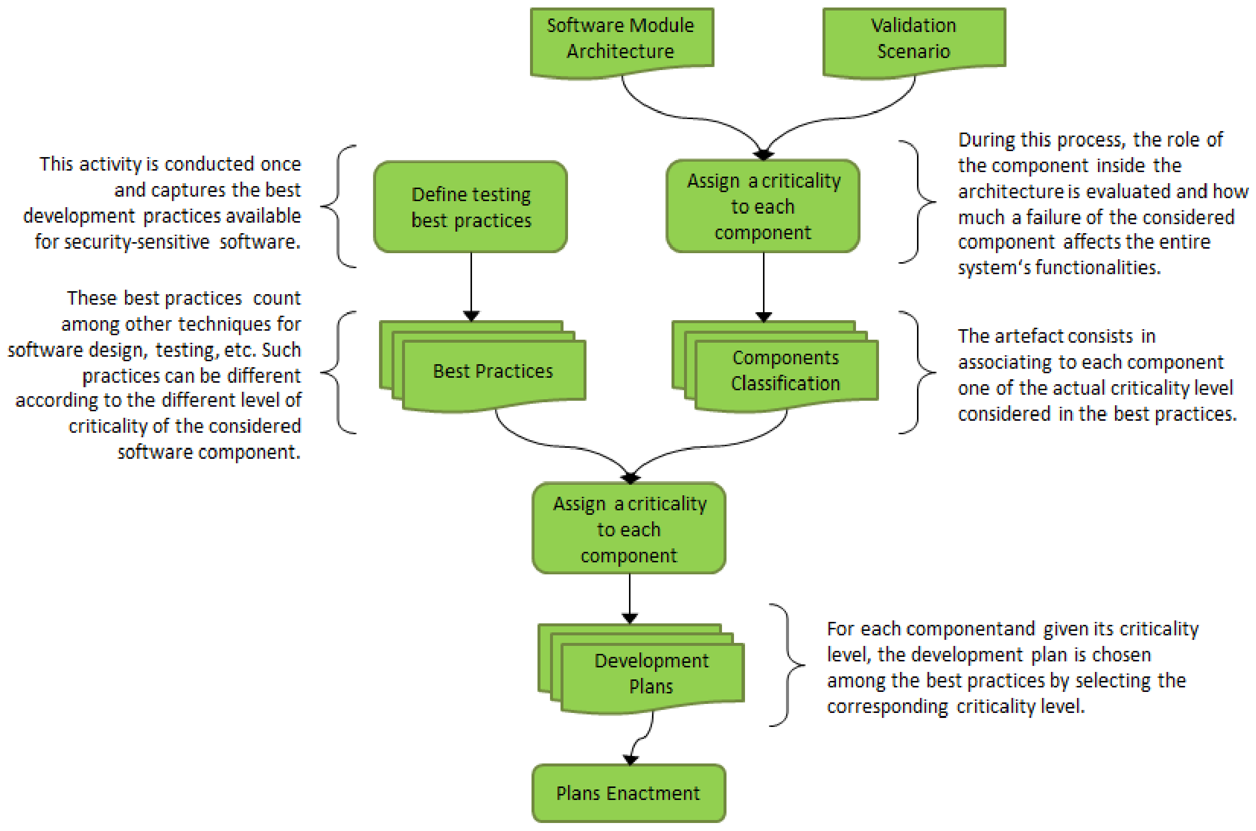

Software Module Architecture
Validation Scenario
This activity is conducted once and captures the best development practices available for security-sensitive software.
Define testing best practices
Assign a criticality to each component
During this process, the role of the component inside the architecture is evaluated and how much a failure of the considered component affects the entire system's functionalities.
These best practices count among other techniques for software design, testing, etc. Such practices can be different according to the different level of criticality of the considered software component.
Best Practices
Components Classification
The artefact consists in associating to each component one of the actual criticality level considered in the best practices.
Assign a criticality to each component
Development Plans
For each componentand given its criticality level, the development plan is chosen among the best practices by selecting the corresponding criticality level.
Plans Enactment

Figure 4. SPECS validation and testing process

As depicted in Figure 4, during the *Specification* and *Design* steps of the SPECS development process, each SPECS artefact is assigned a criticality level. The inputs for the assignment phase are requirements and validation scenarios (for the Enforcement module see Appendix 1 and Appendix 2, respectively) on one side, and architecture and interactions (for the Enforcement module see D4.3.2 and D4.3.3) on the other. Based on this assignment, the techniques reported in Table 1 (initially introduced in D4.5.2) were applied. For details on each of the introduced techniques see D4.5.2.

	High	**Medium**	**Low**
Specification	• Traceability • Peer-review inspection	• Traceability	• Traceability
Design	• Interfaces and behavioural UML modelling	• Interfaces and behavioural UML modelling	• Interfaces and behavioural UML modelling
Coding	• Secure programming • Coding standard	• Coding standard	• Structured Programming
Verification	• Code quality analysis • Black box functional testing • Branch coverage white box testing • Security testing • Security review • Interoperability testing • Dependability and robustness testing	• Black box functional testing • Unit testing • Statement coverage white box testing	• Black box functional testing • Unit testing
Operation & Maintenance	• Use of versioning software systems • Use of an open problems log	• Use of versioning software systems	• Use of versioning software systems

Table 1. Techniques/criticality matrix

In Table 2 we report criticality levels assigned to components and mechanisms of the Enforcement module. All core components are crucial for the execution of the entire SPECS flow, thus their criticality level is high (H). The same goes for the WebPool and DBB, which are the mandatory security mechanisms for the Secure Web Container and Secure Storage services, respectively. The TLS, E2EE, and AAA are involved in many validation scenarios and are therefore assigned a medium (M) criticality level. The SVA and the DoS security mechanisms are only involved in a few validation scenarios and they are not mandatory for the provisioning of SPECS services, thus their level of criticality is low (L).

Artefact of the Enforcement module		**Criticality level**
Core component	Planning	H
	Implementation with Broker	H
	Diagnosis	H
	RDS	H
Security Mechanism	WebPool	H
	TLS	M
	SVA	L
	DBB	H
	E2EE	M
	DoS	L
	AAA	M

Table 2. Criticality levels for the Enforcement module

As for the testing, we analysed the entire framework, and decided to execute the following types of tests:

- **Unit tests:** To verify correctness of the implementation on the component level.
- **Performance tests:** For some artefacts where performance is of high importance, e.g., the Monitoring module, the E2EE mechanism.
- **Interoperability tests:** Due to the complex architecture and the variety of programming languages used.
- **Dependability and robustness tests:** In terms of stress testing and perturbation analysis.
- **Security tests:** To identify possible vulnerabilities in terms of security.

The results of the validation of the Enforcement module (in terms of coverage of validation scenarios and requirements with developed prototypes) are discussed in Section 4. Unit tests for the Enforcement module are reported in Section 5 and performance and scalability of the Enforcement module is discussed in Section 6. For results of security testing performed on the system level see deliverable D1.5.2.

4. Validation of the Enforcement module

In deliverable D4.5.2 we presented the intermediate verification of Enforcement components that were developed up to the M24 and discussed in deliverables D4.3.2 and D4.4.2. However, this section presents the final validation of the entire Enforcement module.

First, the coverage of the validation scenarios is discussed. Scenarios were defined in D5.1.1 and refined in deliverable D5.1.2. During the last six months of the project, the scenarios have been further refined according to the feedback from the development and integration activities, and the final version of validation scenarios is reported in Appendix 2.

In the last subsection (namely, in Section 4.2) we discuss the coverage of requirements associated with the Enforcement module (listed in Appendix 1) and report about how each requirement is covered by the developed prototypes.

For details about the Enforcement prototypes see D4.3.2 and D4.3.3, for the APIs see D1.3.

4.1. Coverage of validation scenarios

In the following tables, we report validation scenarios that involve Enforcement module, and analyse them in terms of components, mechanisms, and requirements that they cover.

Although the initial analysis has already been performed and its results had been reported in deliverable D4.5.2, we analyse and report the complete list of validation scenarios for the sake of completeness. We outline any changes to the contents that have occurred after the initial analysis in the *Comment* part of each validation scenario table.

Due to the latest updates in the implementation of the AAA mechanism, some functionalities that were used in the original validation scenarios are no longer relevant. For further details on the validation scenarios associated with the AAA mechanism please refer to D5.4.

Scenario ID	*SST.1 Secure_Storage_Selection*	
Scenario description	The End-user aims at acquiring a secure storage service from a cloud provider, which fulfils specific security-related requirements. To achieve this, the End-user negotiates the desired features with SPECS. In this validation scenario, the desired features are entirely implemented by an external CSP, while SPECS only provides to the End-user the functionalities to search, rank and select a service, which are compliant with her/his requirements. Moreover, in this scenario, the End-user signs an SLA with the selected provider.	
Involved Enforcement components/mechanisms		**Related requirements**
• Planning component (builds valid supply chains)		• *ENF_PLAN_R1-R4* • *ENF_PLAN_R10-R12*
Comment	*No changes with respect to M24 (D4.5.2).*	

Scenario ID	*SST.2 Secure_Storage_Brokering_with_Client_Crypto*
Scenario description	The End-user aims at acquiring a secure storage service from a remote cloud provider, which fulfils specific security-related requirements. Specifically, the End-user needs two capabilities, Database-as-a-Service and End-to-End Encryption, in order to detect and prove security-related violations, and to locally

	encrypt her/his data. To enable this service, the End-user negotiates the desired features with SPECS and signs an SLA including all service terms and guarantees. SPECS acquires the Database-as-a-Service on behalf of the End-user (registered on SPECS) and provides her/him with the End-to-End Encryption security mechanism. In this scenario, SPECS also provides monitoring functionalities.
Involved Enforcement components/mechanisms	**Related requirements**
• Planning component (builds valid supply chains, builds implementation plan, updates MoniPoli) • Implementation component (configures resources and SPECS components) • Broker mechanism (acquires resources) • DBB mechanism (offers secure storage with backup) • E2EE mechanism (provides client-side encryption)	• *ENF_PLAN_R1-R7* • *ENF_PLAN_R10-R12* • *ENF_IMPL_R1-R8* • *ENF_IMPL_R10* • *ENF_BROKER_R1-R5* • *ENF_CRYPTO_R1-R4* • *ENF_DBB_R1-R2*
Comment	*No changes with respect to M24 (D4.5.2).*

Scenario ID	*SST.3 Secure_Storage_with_Defined_CSP*
Scenario description	The End-user aims at storing encrypted data on a known remote cloud provider which offers a Database-as-a-service capability. The End-user asks SPECS for End-to-End Encryption capability, needed to locally encrypt her/his data. To enable this service, the End-user also gives SPECS her/his credentials on the chosen provider; SPECS securely manages these credentials and uses them to log into the chosen provider and store the End-user's data. In this scenario, SPECS also provides monitoring functionalities.
Involved Enforcement components/mechanisms	**Related requirements**
• Planning component (builds valid supply chains, builds implementation plan, updates MoniPoli) • Implementation component (configures resources and SPECS components) • Broker mechanism (acquires resources) • DBB mechanism (offers secure storage with backup) • E2EE mechanism (provides client-side encryption)	• *ENF_PLAN_R1-R7* • *ENF_PLAN_R10-R12* • *ENF_IMPL_R1-R8* • *ENF_IMPL_R10* • *ENF_BROKER_R1-R5* • *ENF_CRYPTO_R1-R4* • *ENF_DBB_R1-R2*
Comment	*No changes with respect to M24 (D4.5.2).*

Scenario ID	*SST.4 Secure_Storage_Brokering_with_Client_Crypto_Alert*
Scenario description	The End-user aims at acquiring a secure storage service from a remote cloud provider, which fulfils specific security-related requirements. Specifically, the End-user needs two capabilities, Database-as-a-Service and End-to-End Encryption, in order to detect and prove security-related violations, and to locally encrypt her/his data. To enable this service, the End-user negotiates the desired features with SPECS and signs an SLA including all service terms and guarantees. SPECS acquires the Database-as-a-Service on behalf of the End-user (registered on SPECS) and provides her/him with the End-to-End Encryption security mechanism. In this scenario, SPECS also provides monitoring functionalities. In this scenario, an alert is raised since the Encryption Server component is

	detected to be down and, since no data is sent from the End-user during the down time, no violation occurs.
Involved Enforcement components/mechanisms	**Related requirements**
• Planning component (builds valid supply chains, builds implementation plan, updates MoniPoli) • Implementation component (configures resources and SPECS components, executes remediation plan) • Diagnosis component (analyses and classifies the monitoring event) • RDS component (builds remediation plan) • Broker mechanism (acquires resources) • DBB mechanism (offers secure storage with backup) • E2EE mechanism (provides client-side encryption)	• *ENF_PLAN_R1-R8* • *ENF_PLAN_R10-R12* • *ENF_IMPL_R1-R8* • *ENF_IMPL_R10* • *ENF_DIAG_R1-R18* • *ENF_REM_R2-R9* • *ENF_BROKER_R1-R5* • *ENF_CRYPTO_R1-R4* • *ENF_DBB_R1-R2* • *SLANEG_R31*
Comment	*No changes with respect to M24 (D4.5.2).*

Scenario ID	*SST.5 Secure_Storage_Brokering_with_Client_Crypto_Violation*
Scenario description	The End-user aims at acquiring a secure storage service from a remote cloud provider, which fulfils specific security-related requirements. Specifically, the End-user needs two capabilities, Database-as-a-Service and End-to-End Encryption, in order to detect and prove security-related violations, and to locally encrypt her/his data. To achieve this service, the End-user negotiates the desired features with SPECS and signs an SLA including all service terms and guarantees. SPECS acquires the Database-as-a-Service on behalf of the End-user (registered on SPECS) and provides her/him with the End-to-End Encryption security mechanism. In this scenario, SPECS also provides monitoring functionalities. In this scenario, a violation is detected since the Encryption Server component is detected to be down.
Involved Enforcement components/mechanisms	**Related requirements**
• Planning component (builds valid supply chains, builds implementation plan, updates MoniPoli) • Implementation component (configures resources and SPECS components, executes remediation plan) • Diagnosis component (analyses and classifies the monitoring event) • RDS component (builds remediation plan) • Broker mechanism (acquires resources) • DBB mechanism (offers secure storage with backup) • E2EE mechanism (provides client-side encryption)	• *ENF_PLAN_R1-R8* • *ENF_PLAN_R10-R12* • *ENF_IMPL_R1-R8* • *ENF_IMPL_R10* • *ENF_DIAG_R1- R18* • *ENF_REM_R2-R9* • *ENF_BROKER_R1-R5* • *ENF_CRYPTO_R1-R4* • *ENF_DBB_R1-R2* • *SLANEG_R31*
Comment	*No changes with respect to M24 (D4.5.2).*

Scenario ID	*SWC.1 Secure_Web_Container_Selection*
Scenario description	The End-user aims at acquiring a web container service fulfilling specific security requirements (e.g., availability, resilience to attacks). To enable this service, the End-user negotiates the desired features with SPECS. In this validation scenario, the desired features are already provided by a CSP, and SPECS only returns to the End-user the reference to such provider.
Involved Enforcement components/mechanisms	**Related requirements**

• Planning component (builds valid supply chains)		• *ENF_PLAN_R1-R4* • *ENF_PLAN_R10-R12*
Comment	*No changes with respect to M24 (D4.5.2).*	

Scenario ID	*SWC.2 Secure_Web_Container_Brokering*	
Scenario description	The End-user aims at acquiring a web container service fulfilling specific security requirements (e.g., availability, resilience to attacks). To enable this service, the End-user negotiates the desired security features with SPECS. In this validation scenario, the desired features are already provided by a CSP, but SPECS acts as a broker by acquiring the resources on behalf of the End-user (registered on SPECS) and by setting up some monitoring functionalities in order to monitor the fulfilment of the SLA.	
Involved Enforcement components/mechanisms		**Related requirements**
• Planning component (builds valid supply chains, builds implementation plan, updates MoniPoli) • Implementation component (configures resources and SPECS components) • Broker mechanism (acquires resources) • WebPool mechanism (offers a secure web container)		• *ENF_PLAN_R1-R7* • *ENF_PLAN_R10-R12* • *ENF_IMPL_R1-R8* • *ENF_IMPL_R10* • *ENF_BROKER_R1-R5* • *ENF_POOL_R1-R5*
Comment	*No changes with respect to M24 (D4.5.2).*	

Scenario ID	*SWC.3 Secure_Web_Container_TLS_Enhanced*	
Scenario description	The End-user aims at acquiring a web container service fulfilling specific security requirements. In particular, the End-user requires the adoption of the TLS protocol to protect the network communications, and of the DoS detection and mitigation features. To enable this service, the End-user negotiates the desired features with SPECS. In this validation scenario, a bare web container is offered by a CSP, while the TLS protocol and the DoS detection and mitigation features are provided by SPECS through the activation of proper mechanisms. SPECS acquires the resources on behalf of the End-user (registered on SPECS), deploys and activates the TLS and DoS related mechanisms, and sets up elated monitoring functionalities. In this scenario, an alert regarding a DoS attack is generated, and SPECS reacts by activating proper mitigation strategies. The scenario ends without any other alert.	
Involved Enforcement components/mechanisms		**Related requirements**
• Planning component (builds valid supply chains, builds implementation plan, updates MoniPoli) • Implementation component (configures resources and SPECS components, executes remediation plan) • Diagnosis component (analyses and classifies the monitoring event) • RDS component (builds remediation plan)		• *ENF_PLAN_R1-R8* • *ENF_PLAN_R10-R12* • *ENF_IMPL_R1-R8* • *ENF_IMPL_R10* • *ENF_DIAG_R1- R18* • *ENF_REM_R2-R9*

• Broker mechanism (acquires resources) • WebPool mechanism (offers a secure web container) • TLS mechanism (provides TLS protocol) • DoS mechanism (provides DoS detection and mitigation functionalities)	• *ENF_BROKER_R1-R5* • *ENF_POOL_R1-R5* • *ENF_TLS_R1-R5* • *ENF_DOS_R1-R3* • *SLANEG_R31*
Comment	*With respect to the coverage reported in D4.5.2, the refined validation scenario now involves the DoS mechanism.*

Scenario ID	*SWC.4 Secure_Web_Container_SVA_Enhanced_Alert*
Scenario description	The End-user aims at acquiring a web container service fulfilling specific security requirements. In particular, the End-user requires the adoption of a Software Vulnerability Assessment (SVA) tool to protect the web container environment. To enable this service, the End-User negotiates the desired features with SPECS. In this validation scenario, the bare web container is offered by a CSP, while the SVA tools are provided by SPECS. SPECS acquires the resources on behalf of the End-user (registered on SPECS), deploys and activates the needed SVA agents and sets-up related monitoring functionalities. In this scenario, an alert is generated due to the existence of some critical vulnerability in the installed software. SPECS reacts by updating the software version to remove the vulnerability. The scenario ends without any other alert.
Involved Enforcement components/mechanisms	**Related requirements**
• Planning component (builds valid supply chains, builds implementation plan, updates MoniPoli) • Implementation component (configures resources and SPECS components, executes remediation plan) • Diagnosis component (analyses and classifies the monitoring event) • RDS component (builds remediation plan) • Broker mechanism (acquires resources) • WebPool mechanism (offers a secure web container) • SVA mechanism (provides SVA security services)	• *ENF_PLAN_R1-R8* • *ENF_PLAN_R10-R12* • *ENF_IMPL_R1-R8* • *ENF_IMPL_R10* • *ENF_DIAG_R1- R18* • *ENF_REM_R2-R9* • *ENF_BROKER_R1-R5* • *ENF_POOL_R1-R5* • *ENF_SVA_R1-R4* • *SLANEG_R31*
Comment	*No changes with respect to M24 (D4.5.2).*

Scenario ID	*SWC.5 Secure_Web_Container_TLS_SVA_Enhanced_Violation*
Scenario description	The End-user aims at acquiring a web container service fulfilling specific security requirements. In particular, the End-user requires the adoption of Software Vulnerability Assessment (SVA) tools to protect the Web Server environment. To enable this service, the End-user negotiates the desired features with the SPECS. In this validation scenario, the VM (without SVA) is provided by a CSP, while the SVA agents are installed by SPECS. SPECS acquires the resources on behalf of the End-user (registered on SPECS), it adds the SVA agents, and sets up the associated monitoring functionalities in order to detect the presence of SLA violations.

	This scenario includes the raising of an alert regarding a vulnerability assessment report, which corresponds to a violation of the agreed SLA. SPECS reacts by renegotiating the SLA; the End-user asks for the adoption of the TLS protocol to protect the Web Server communications. The renegotiated SLA is hence signed and properly monitored by SPECS.
Involved Enforcement components/mechanisms	**Related requirements**
• Planning component (builds valid supply chains, builds implementation plan, builds reaction plan, updates MoniPoli) • Implementation component (configures resources and SPECS components, executes remediation plan, executes reaction plan) • Diagnosis component (analyses and classifies the monitoring event) • RDS component (builds remediation plan) • Broker mechanism (acquires resources) • WebPool mechanism (offers a secure web container) • SVA mechanism (provides SVA security services) • TLS mechanism (offers TLS protocol)	• *ENF_PLAN_R1-R12* • *ENF_IMPL_R1-R10* • *ENF_DIAG_R1- R18* • *ENF_REM_R1-R9* • *ENF_BROKER_R1-R5* • *ENF_POOL_R1-R5* • *ENF_TLS_R1-R5* • *ENF_SVA_R1-R4* • *SLANEG_R30-R31*
Comment	*No changes with respect to M24 (D4.5.2).*

Scenario ID	*SWC.6 Secure_Web_Container_TLS_Multitenancy*
Scenario description	Two End-users aim at acquiring different web container services fulfilling specific security requirements. In addition, both End-users require the adoption of the TLS protocol to protect the communications of Web Servers. To enable this service, the first End-user negotiates the desired features with SPECS. The VM (without TLS) is provided by a CSP, while the TLS protocol is added by SPECS by setting up the appropriate resources (e.g., reverse proxy). The second End-user negotiates the desired features with SPECS. A different VM (without TLS) is provided by a CSP (either the same or a different one) while the TLS protocol is added by SPECS reusing, for scalability purposes, the same resources configured for the first End-user. This validation scenario considers the multi-tenancy in the usage of shared resources between End-users.
Involved Enforcement components/mechanisms	**Related requirements**
• Planning component (builds valid supply chains, builds implementation plan, updates MoniPoli) • Implementation component (configures resources and SPECS components) • Broker mechanism (acquires resources) • WebPool mechanism (offers a secure web container) • TLS mechanism (offers TLS protocol) • DoS mechanism (provides DoS detection and mitigation functionalities)	• *ENF_PLAN_R1-R7* • *ENF_PLAN_R10-R12* • *ENF_IMPL_R1-R8* • *ENF_IMPL_R10* • *ENF_BROKER_R1-R5* • *ENF_POOL_R1-R5* • *ENF_TLS_R1-R5* • *ENF_DOS_R1-R3*
Comment	*With respect to the coverage reported in D4.5.2, the refined validation scenario now involves the DoS mechanism.*

Scenario ID	*SWC.7 Secure_Web_Container_Web_Pool_Replication_Enhanced_Alert*

Scenario description	The End-user aims at acquiring a web container service fulfilling specific security requirements. In particular, the End-user requires a specific level of redundancy and session persistence among web container replicas. To enable this service, the End-User negotiates the desired features with SPECS. In this validation scenario, the bare web containers are offered by a CSP, while the redundancy features with session persistence among replicas is provided by SPECS through the *WebPool* mechanism. SPECS acquires the resources on behalf of the End-user (registered on SPECS), adds the WebPool mechanism's components, and sets-up proper resources to handle HTTP requests through proxy functionalities, in order to forward the requests to one of the available web container replicas. In this scenario, the proxy functionality is added, by SPECS, on a dedicated VM. In this scenario, an alert is generated because one of the replicas slows down, thus risking compromising the desired level of redundancy. SPECS reacts by isolating the replica and by restarting it. The scenario ends without any other alert	
Involved Enforcement components/mechanisms		**Related requirements**
• Planning component (builds valid supply chains, builds implementation plan, updates MoniPoli) • Implementation component (configures resources and SPECS components, executes remediation plan) • Diagnosis component (analyses and classifies the monitoring event) • RDS component (builds remediation plan) • Broker mechanism (acquires resources) • WebPool mechanism (offers a secure web container)		• *ENF_PLAN_R1-R8* • *ENF_PLAN_R10-R12* • *ENF_IMPL_R1-R8* • *ENF_IMPL_R10* • *ENF_DIAG_R1- R18* • *ENF_REM_R2-R9* • *ENF_BROKER_R1-R5* • *ENF_POOL_R1-R5* • *SLANEG_R31*
Comment	*With respect to M24 (D4.5.2) the alert associated to the vulnerability threat has been replaced by an alert associated to the level of redundancy.*	

Scenario ID	*SWC.8 Secure_Web_Container_Web_pool_Replication_Enhanced_Violation*	
Scenario description	An End-user aims at acquiring a web container service fulfilling specific security requirements. In particular, the End-user requires a specific level of redundancy and session persistence among web container replicas. To achieve this service, the End-User negotiates the desired features with SPECS. In this validation scenario, the bare web containers are offered by a CSP, while the redundancy features with session persistence among replicas is provided by SPECS through the *WebPool* mechanism. SPECS acquires the resources on behalf of the End-user (registered on SPECS), adds the WebPool mechanism's components, and sets-up proper resources to handle HTTP requests through proxy functionalities, in order to forward the requests to one of the available web container replicas. In this scenario, the proxy functionality is added, by SPECS, on a dedicated VM. In this scenario, an alert is generated because one of the replicas goes down, thus compromising the desired level of redundancy. SPECS reacts by isolating the replica and by removing it from the pool of replicas. The SLA is violated since the level of redundancy is not preserved.	
Involved Enforcement components/mechanisms		**Related requirements**
• Planning component (builds valid supply chains, builds implementation plan, updates MoniPoli) • Implementation component (configures resources and SPECS components, executes remediation plan)		• *ENF_PLAN_R1-R8* • *ENF_PLAN_R10-R12* • *ENF_IMPL_R1-R8*

• Diagnosis component (analyses and classifies the monitoring event) • RDS component (builds remediation plan) • Broker mechanism (acquires resources) • WebPool mechanism (offers a secure web container)		• *ENF_IMPL_R10* • *ENF_DIAG_R1- R18* • *ENF_REM_R2-R9* • *ENF_BROKER_R1-R5* • *ENF_POOL_R1-R5* • *SLANEG_R31*
Comment	*No changes with respect to M24 (D4.5.2).*	

Scenario ID	*NGDC.1 Data_Center_Bursting_for_Storage_Resources*	
Scenario description	A CSP hosting its own next generation Data Center (ngDC), acting within a Cloud Service Customer (CSC) role, aims at using the SPECS framework to perform Cloud bursting in order to extend its Secure Storage as a Service (SStaaS) capabilities. This occurs during a period of increased storage demand, which exceeds the CSP's own ngDC storage capabilities. The CPS considers its storage as *first class* storage due to the fine grained control it has over all the security parameters. The CSP will allocate the first class storage to End-users that do not require high-security capabilities enabled. Otherwise, it will allocate storage acquired from an external provider through SPECS. The entire process is transparent to the End-user. Note, while a CSP acquiring storage resources from an external 3rd party CSP is typically defined as an End-user, it is not in the context of a SPECS defined in this way. That is, the CSP intends to resell its acquired external storage resources and so it is considered a CSC (in the context of SPECS). For ease of exposition 'customer' is used as a common reference to either a CSC or End-user of the CSP hosting the ngDC.	
Involved Enforcement components/mechanisms		**Related requirements**
• Planning component (builds valid supply chains)		• *ENF_PLAN_R1-R4* • *ENF_PLAN_R10-R12*
Comment	*No changes with respect to M24 (D4.5.2).*	

Scenario ID	*NGDC.3 Data_Center_Storage_Selection*	
Scenario description	A CSP owning SPECS and hosting its own ngDC, acting within a CSC role, aims at using the SPECS framework to perform Cloud bursting in order to extend its SStaaS capabilities. This occurs during a period of increased storage demand, which exceeds the CSP's own ngDC storage capabilities. In this validation scenario, the desired features are entirely implemented by an external CSP, while SPECS only offers the End-user the ability to search, rank and select a service, which is compliant to her/his requirements. Moreover, in this scenario, SPECS supports the End-user in signing an SLA with the selected provider.	
Involved Enforcement components/mechanisms		**Related requirements**
• Planning component (builds valid supply chains)		• *ENF_PLAN_R1-R4* • *ENF_PLAN_R10-R12*
Comment	*The analysis of this validation scenario is new in this document, as it has not yet been reported in D4.5.2.*	

Scenario ID	*CRO.3 Security_Tokens_Revocation*
Scenario	In this validation scenario, the revocation of a security token is shown.

description		
Involved Enforcement components/mechanisms		**Related requirements**
• Implementation component (revokes a security token)		• *ENF_TOK_R5*
Comment	*The analysis of this validation scenario is new in this document, as it has not yet been reported in D4.5.2.*	

Scenario ID	*CRO.10 SPECS_Application_Development*	
Scenario description	A SPECS developer aims at developing a new SPECS application. In this scenario, the development of a new SPECS application, using the default SPECS application as a template, is shown.	
Involved Enforcement components/mechanisms		**Related requirements**
• Planning component (builds valid supply chains, builds implementation plan, updates MoniPoli) • Implementation component (configures resources and SPECS components, executes remediation plan) • Diagnosis component (analyses and classifies the monitoring event) • RDS component (builds remediation plan) • Broker mechanism (acquires resources) • WebPool mechanism (offers a secure web container) • TLS mechanism (offers TLS protocol) • SVA mechanism (provides SVA security services) • DoS mechanism (provides DoS detection and mitigation functionalities) • DBB mechanism (offers secure storage with backup) • E2EE mechanism (provides client-side encryption) • AAA (provides federated identity and access management functionalities)		• *ENF_PLAN_R1-R12* • *ENF_IMPL_R1-R10* • *ENF_DIAG_R1-R18* • *ENF_REM_R1-R9* • *ENF_BROKER_R1-R5* • *ENF_POOL_R1-R5* • *ENF_TLS_R1-R5* • *ENF_SVA_R1-R4* • *ENF_CRYPTO_R1-R4* • *ENF_AAA_R1-R9* • *ENF_DOS_R1-R3* • *ENF_DBB_R1-R2* • *SLANEG_R30-R31*
Comment	*The analysis of this validation scenario is new in this document, as it has not yet been reported in D4.5.2.*	

Scenario ID	*AAA.1 Identity_Management_Set-up*	
Scenario description	In this scenario, a customer acquires the enhanced secure storage service from the SPECS Owner, configures the service and sets the access control policies for its End-users by using the identity management features offered by the service. Moreover, the provider configures the Identity Federation by identifying the supported identity providers.	
Involved Enforcement components/mechanisms		**Related requirements**
• AAA (provides federated identity and access management functionalities)		• *ENF_AAA_R6* • *ENF_AAA_R8* • *ENF_AAA_R9*
Comment	*The analysis of this validation scenario is new in this document, as it has not yet been reported in D4.5.2.*	

Scenario ID	*AAA.2 User_Registration*
Scenario description	In this scenario, an End-user of the enhanced secure storage service performs a registration by providing her/his data.

Involved Enforcement components/mechanisms		**Related requirements**
• AAA (provides federated identity and access management functionalities)		• *ENF_AAA_R4*
Comment	*The analysis of this validation scenario is new in this document, as it has not yet been reported in D4.5.2.*	

Scenario ID	*AAA.3 User_Access_Internal_Account*	
Scenario description	In this scenario, an End-user requests the access to the storage system by using the account created when registering with the service;	
Involved Enforcement components/mechanisms		**Related requirements**
• AAA (provides federated identity and access management functionalities)		• *ENF_AAA_R4-R9*
Comment	*The analysis of this validation scenario is new in this document, as it has not yet been reported in D4.5.2.*	

Scenario ID	*AAA.4 User_Access_External_Account*	
Scenario description	In this scenario, an End-user requests access to the storage system by using an external account belonging to a supported Identity Provider. When the user chooses to authenticate through an external source, the application checks that the external account is associated with a supported identity provider. In this case, the user is authenticated. Otherwise the application asks if the End-user wants to associate the external account to her/his existing internal account. In this latter case, the End-user must first be authenticated on the application in order to prove the ownership of the internal account.	
Involved Enforcement components/mechanisms		**Related requirements**
• AAA (provides federated identity and access management functionalities)		• *ENF_AAA_R4-R9*
Comment	*The analysis of this validation scenario is new in this document, as it has not yet been reported in D4.5.2.*	

Table 3 presents the traceability matrix summarizing the correlation between validation scenarios and the Enforcement module.

Enforcement component / mechanism	SST.1	SST.2	SST.3	SST.4	SST.5	SWC.1	SWC.2	SWC.3	SWC.4	SWC.5	SWC.6	SWC.7	SWC.8	NGDC.1	NGDC.3	CRO.3	CRO.10	AAA.1	AAA.2	AAA.3	AAA.4
Planning	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x		x				
Implementation with Broker		x	x	x	x		x	x	x	x	x	x	x			x	x				
Diagnosis				x	x			x	x	x		x	x				x				
RDS				x	x			x	x	x		x	x				x				
WebPool							x	x	x	x	x	x	x				x				
TLS								x		x	x						x				
SVA									x	x							x				
DBB		x	x	x	x												x				
E2EE		x	x	x	x												x				

DoS								x			x						x				
AAA																	x	x	x	x	x

Table 3. Coverage of validation scenarios by Enforcement components/mechanisms

4.2. Coverage of requirements

Previous iterations of this deliverable reported not only the coverage of requirements with validation scenarios and components, but also elaborated on how requirements have been covered by the Enforcement prototypes developed up to M24. In Table 4 we report the final analysis of the requirements with respect to their coverage with Enforcement components and mechanisms.

For the sake of completeness we report the entire set of requirements, but we highlight those that have only been covered by the final prototypes.

REQ_ID	Requirement	Comment
ENF_PLAN_R1	***Get SLA to enforce***	The Planning component parses the SLA to build supply chains and to prepare implementation plans.
ENF_PLAN_R2	***Define security mechanisms related to SLOs***	The Planning component considers the set of SLOs in the SLA (while building supply chains) and determines which kind of security mechanisms are to be applied.
ENF_PLAN_R3	***Get security components***	The Planning component (while building supply chains) retrieves all security mechanisms able to implement negotiated SLA.
ENF_PLAN_R4	***Select best security component***	The Planning component (while building supply chains) selects the best available security mechanisms able to implement negotiated SLA.
ENF_PLAN_R5	***Activate implementation***	The Planning component triggers execution of an implementation plan by invoking the Implementation component.
ENF_PLAN_R6	***Log component activation and deactivation***	The Planning component logs its activation and deactivation.
ENF_PLAN_R7	***Build an implementation plan***	The Planning component prepares an implementation plan based on the signed SLA and associated supply chain. Building implementation plan includes deducing alert thresholds.
ENF_PLAN_R8	***Build a reaction plan***	The RDS component is able to build a reaction plan after an alert or a violation *(covered at M24)*. The Planning component is able to build a reaction plan after renegotiation *(covered at M30)*.
ENF_PLAN_R9	***Build a migration plan***	This requirement remains uncovered (we do not support migration of data from one CSP to another).
ENF_PLAN_R10	***Get monitoring systems***	The Planning component (while building supply chains) retrieves all monitoring systems able to monitor negotiated SLA.
ENF_PLAN_R11	***Select best monitoring systems***	The Planning component (while building supply chains) selects the best available monitoring systems able to monitor negotiated SLA.
ENF_PLAN_R12	***Validate an SLA***	The Planning component builds only valid supply chains. Consequently (in the SLA negotiation

		process) only valid SLAs are built.
ENF_IMPL_R1	***Implement Plan***	The Implementation component executes implementation plan by orchestrating the acquisition of the needed resources, their configuration, and the activation of involved services.
ENF_IMPL_R2	***Acquire resources***	The Implementation component (the Broker) acquires all resources needed to realize an implementation plan.
ENF_IMPL_R3	***Deploy and configure***	The Implementation component deploys and configures all acquired resources according to the implementation plan.
ENF_IMPL_R4	***Start services***	The Implementation component activates all services deployed and configured on top of acquired resources.
ENF_IMPL_R5	***Trigger monitoring agent activation or deactivation***	The Implementation component activates/deactivates all monitoring agents deployed and configured on top of acquired resources.
ENF_IMPL_R6	***Log service activation***	The Implementation component logs a successful activation of each security service related to the implemented SLA.
ENF_IMPL_R7	***Update SLA state***	The Implementation component updates the state of an SLA after its successful implementation.
ENF_IMPL_R8	***Log component activation or deactivation***	The Implementation component logs its activation or deactivation.
ENF_IMPL_R9	***Implement reaction plan***	The Implementation component implements a reaction plan built after renegotiation *(covered at M30)*.
ENF_IMPL_R10	***Update monitoring policy***	The Planning component updates the monitoring policy (with violation and alert thresholds) according to a signed SLA.
ENF_DIAG_R1	***Get monitoring event notification***	The Diagnosis component receives notifications of monitoring events from the Monitoring module.
ENF_DIAG_R2	***Get monitoring event information***	The Diagnosis component is able to retrieve all information related to a monitoring event by accessing the Auditing component.
ENF_DIAG_R3	***Identify SLOs affected by a monitoring event***	The Diagnosis component analyses notified monitoring events and identify the SLOs at risk or violated.
ENF_DIAG_R4	***Update SLA state***	The Diagnosis component updates the state of an SLA (to *Alerted* or *Violated*) depending on the classification of the notified monitoring event.
ENF_DIAG_R5	***Get SLAs affected by a monitoring event***	The Diagnosis component identifies and retrieves all SLAs affected by a notified monitoring event.
ENF_DIAG_R6	***Activate reaction***	The Diagnosis component activates the RDS component to react to an alert or a violation.
ENF_DIAG_R7	***Express SLA violation in terms of KPI***	The Diagnosis component expresses SLA violations in terms of the affected SLOs.
ENF_DIAG_R8	***Query metric***	The Diagnosis component is able to query the metric data stored inside the Event Archiver (Monitoring module) when evaluating the status of

		a notified monitoring event.
ENF_DIAG_R9	***Log component activation or deactivation***	The Diagnosis component logs its activation and deactivation.
ENF_DIAG_R10	***Determine effect on an SLA***	For each SLA affected by a monitoring event, the Diagnosis component determines the effect the monitoring event has on the SLA (i.e., is it alerted or violated).
ENF_DIAG_R11	***Log SLA impact***	The Diagnosis component logs all event related information.
ENF_DIAG_R12	***Classify event***	The Diagnosis component classifies all notified monitoring events.
ENF_DIAG_R13	***Identify root cause***	The Diagnosis component performs a root cause analysis of each monitoring event.
ENF_DIAG_R14	***Log root cause***	The Diagnosis component logs all event related information.
ENF_DIAG_R15	***Analyse monitoring event***	The Diagnosis component analyses each notified monitoring event.
ENF_DIAG_R16	***Prioritize events***	The Diagnosis component prioritizes monitoring events (actually, SLAs affected by notified events) according to their risk/severity levels.
ENF_DIAG_R17	***Log priority queue***	The Diagnosis component logs all event related information.
ENF_DIAG_R18	***Log queue change***	The Diagnosis component must be able to compare the current metric/SLO data with the alert/violation thresholds specified for an alerted/violated SLA to verify if the severity of the alert/violation has changed.
ENF_REM_R1	***Trigger renegotiation***	The RDS component triggers renegotiation when available remediation activities are unable to resolve SLA violations.
ENF_REM_R2	***Log component activation or deactivation***	The RDS logs its activation and deactivation.
ENF_REM_R3	***Get SLA state***	The RDS checks the state of an SLA in order to identify proper remediation actions.
ENF_REM_R4	***Update SLA state***	The RDS component updates SLA's state (to *Proactive Redressing* or *Remediating*) depending to the type of the notified event (alert or violation).
ENF_REM_R5	***Get SLA***	The RDS component retrieves an alerted/violated SLA in order to identify required remediation actions.
ENF_REM_R6	***Get SLA impact***	The RDS component is able to retrieve all information related to an alert/violation.
ENF_REM_R7	***Get security components***	The RDS component retrieves all event related security components.
ENF_REM_R8	***Search for redressing techniques***	The RDS identifies remediation actions based on the event information and affected SLAs.
ENF_REM_R9	***Notify End-user***	When End-user's decision is needed in the process of managing an alert or a violation, the RDS component communicates the issue with the End-user through the SPECS Application.
ENF_BROKER_R1	***Enable CSP***	The SPECS Administrator is able to configure and

		enable the Broker to access and use an external CSP.
ENF_BROKER_R2	***Acquire cluster***	The Broker component is able to acquire a cluster of VMs on one of the enabled CSPs.
ENF_BROKER_R3	***Delete cluster***	The Broker component is able to delete a cluster of VMs previously acquired.
ENF_BROKER_R4	***Add user***	The Broker component is able to add a new user to the available cluster of VMs.
ENF_BROKER_R5	***Execute script on node***	The Broker component is able to execute on or more scripts on a cluster of VMs.
ENF_POOL_R1	***Diversity***	This requirement is satisfied by acquiring many VMs and configuring a different web server engines on them.
ENF_POOL_R2	***Load balancing***	This requirement is satisfied by configuring a proxy that is able to forward all incoming requests to one of the VMs hosting the web server engines. The scheduling policy can be configured.
ENF_POOL_R3	***Survivability***	This requirement is satisfied by acquiring and configuring the same web server engine on more than one VM.
ENF_POOL_R4	***Session sharing***	This requirement is satisfied by configuring each web server engine with a cache accessible to all web server engines,
ENF_POOL_R5	***Incident management***	*This requirement has been covered by devising proper remediation actions, which are to be executed when the WebPool Agent component (the HAProxy) detects that a VM is down and notifies the Monitoring module.*
ENF_TLS_R1	***Translate TLS constraints***	TLS Reasoner translates security high level constraints and requirements in configuration templates that are used by both TLS Terminator, to enforce, and TLS Prober, to monitor and generate events.
ENF_TLS_R2	***Verify TLS constraints***	TLS Configurator verifies if the configuration templates do not overlap or generate misconfigurations by adding contradictory features or configurations.
ENF_TLS_R3	***Instantiate TLS configuration***	TLS Terminator Configurator will add to the TLS Terminator the right configuration templates that meet the negotiated requirements.
ENF_TLS_R4	***Deploy TLS configuration***	TLS Terminator Controller will deploy the configuration instantiated by the TLS Terminator Configurator.
ENF_TLS_R5	***Probe TLS endpoint configuration***	TLS Prober will periodically check if the instantiated and deployed configuration template is not altered during the lifecycle of the component.
ENF_SVA_R1	***Detect vulnerabilities and misconfigurations***	The SVA mechanism is able to detect software vulnerabilities.
ENF_SVA_R2	***Report vulnerabilities and misconfigurations***	The SVA mechanism reports about the detected software vulnerabilities.
ENF_SVA_R3	***Upgrade libraries and fix***	Due to complexity of the automatically upgrading libraries and fixing misconfigurations, this

	misconfigurations	requirement remains uncovered.
ENF_SVA_R4	***Visualize detected vulnerabilities and misconfigurations***	The SVA Dashboard presents all software vulnerability reports, list of published vulnerabilities, and status of scans and measurements taken under the umbrella of the SVA mechanism.
ENF_CRYPTO_R1	***Provide client-side encryption tool as a plugin/extension***	The E2EE mechanism provides client-side encryption with the E2EE Client component.
ENF_CRYPTO_R2	***Configure and deploy encryption tools***	Encryption tools (components of the E2EE mechanism) are configurable.
ENF_CRYPTO_R3	***Encrypt data***	The E2EE mechanism enables local encryption of files.
ENF_CRYPTO_R4	***Decrypt data***	The E2EE mechanism enables local decryption of encrypted files.
ENF_AAA_R1	***Support different authentication sources***	The AAA mechanism supports an internal authentication source, represented by an LDAP directory service. Support to other external authentication sources must be added by properly configuring a client application to connect to other OAuth servers (e.g., LinkedIn, Facebook, etc.). Since it is delegated to the client, which is not part of the AAA package, this requirement is deprecated.
ENF_AAA_R2	***Manage different accounts for a user***	We adopted the OAUTH protocol to authorize access to services offered by SPECS Platform. The authentication process is thus delegated to the OAUTH server. According to such an approach, we do not need to explicitly support multiple accounts, but simply manage multiple accounts, referring to different OAUTH servers.
ENF_AAA_R3	***Link different identities to a single account***	We adopted the OAUTH protocol to authorize access to services offered by the SPECS Platform. The authentication process is delegated to the OAUTH server. Our OAUTH server does not support multiple identities explicitly, because different identities will be usually managed referring to different OAUTH servers.
ENF_AAA_R4	***Login***	The AAA mechanism allows the users to login in the application via the OAuth Server by using an internal account.
ENF_AAA_R5	***Authenticate***	The AAA mechanism enforces access control policies via the OAuth Server for "basic" resources. Moreover, it is able to manage the authorization of more complex access requests via the Authorization Service, based on XACML.
ENF_AAA_R6	***Dynamically manage access control policies***	The AAA mechanism allows an administrator to update the access control policies by simply changing the XACML policy stored in the AAA package.
ENF_AAA_R7	***Logout***	The AAA mechanism supports the logout operations, including the token revocation.
ENF_AAA_R8	***Authentication and authorization***	The AAA mechanism included two separate modules for authentication and authorization,

	independency	respectively the OAuth Server, which makes use of the Authentication Backend, and the Authorization Service, which makes use of the XACML PEP and PDP modules.
ENF_AAA_R9	***Confidentiality and integrity***	In the AAA mechanism, the communications among the OAuth Client and the OAuth Server are carried out over a HTTPS connection.
ENF_DOS_R1	***Detect DoS attack***	The DoS mechanism is able to detect several kinds of attacks, including DoS attacks, thanks to the integration of the OSSEC monitoring tool.
ENF_DOS_R2	***Classify detected DoS attacks***	The DoS mechanism, by relying upon the OSSEC tool, is able to detect and classify DoS attacks based on their features and impact.
ENF_DOS_R3	***Mitigate DoS attacks***	DoS attacks are mitigated by the DoS mechanism thanks to the rules included in OSSEC.
ENF_DBB_R1	***Offer secure storage***	The DBB mechanism automatically offers secure storage in the cloud.
ENF_DBB_R2	***Assure business continuity with backup***	The DBB mechanism comprises components orchestrating backup services.
SLANEG_R30	***Remediation through SLA renegotiation***	The RDS component considers renegotiation of an existing signed SLA as a potential remedy to apply in case of alerts and violations.
SLANEG_R31	***Alerts/violations affecting multiple elements of the secure SLA hierarchy***	The RDS component considers interrelationships among SLOs when choose the optimal redressing technique in case of SLA alerts and violations.

Table 4. Coverage of Enforcement requirements

Table 5 summarizes the coverage of the requirements by the components and security mechanisms of the Enforcement module that was initially presented in deliverable D4.5.2.

Requirement ID	***Planning***	***Implementation with Broker***	***Diagnosis***	***RDS***	***WebPool***	***DBB***	***E2EE***	***SVA***	***TLS***	***AAA***	***DoS***
ENF_PLAN_R1-R12	x										
ENF_PLAN_R8-R9	x			x							
ENF_IMPL_R1-R9		x									
ENF_IMPL_R10	x										
ENF_DIAG_R1-R18			x								
ENF_REM_R1-R11				x							
SLA_NEG_R30-R31				x							
ENF_BROKER_R1-R5		x									
ENF_POOL_R1-R5					x						
ENF_TLS_R1-R5									x		
ENF_SVA_R1-R4								x			

ENF_CRYPTO_R1-R4							x				
ENF_DBB_R1-R2						x					
ENF_AAA_R1-R9										x	
ENF_DOS_R1-R3											x

Table 5. Coverage of requirements by Enforcement components and mechanisms

5. Functional testing of the Enforcement module

The initial survey of the testing methodologies and tools was reported in D4.5.1. The second iteration of the document (namely, D4.5.2) reported the actual tests to be performed and the tools to be used. In this section we provide results of the defined testing activities.

For each core Enforcement component (in Section 5.1) and each security mechanism (in Section 5.2) we list and discuss the entire set of executed unit tests. Of course, it is impossible to test all possible inputs and preconditions, or even to test all interactions. Therefore, the agreement made among the developers was, to ensure at least 50% code coverage with unit tests (for other collaborative development rules see D4.5.2). All unit tests are available on the project's Bitbucket web site [1] along with the prototypes.

In each subsection for core components (in Section 5.1) we also include the code quality report that confirms this "*minimal code coverage*" agreement is respected and provides a brief analysis of the code. Only core components are evaluated in terms of code quality, since they were developed from scratch. Security mechanisms, on the other hand, are mainly based on existing open source solutions. The code quality has been analysed with SonarQube [2] and all reports are available online [3]. Note that the code quality monitoring provided feedback during the development process. Any issues that were revealed by the SonarQube were addressed by developers.

The testing and the code quality data is summarized and analysed in Section 5.3. Note that performance and scalability is analysed in Section 6, whereas the security analysis is discussed in deliverable D1.5.2.

5.1. Core Enforcement components

In the following subsections we report unit tests and code quality reports for core components of the Enforcement module.

5.1.1. Planning

The Planning component supports the negotiation phase by building valid supply chains. After the SLA signature and after each remediation and renegotiation, the Planning component prepares implementation plans to set up or reconfigure the provisioned services.

Although some initial unit tests have already been reported in deliverable D4.5.2, for the sake of completeness, we report the entire list of tests executed for the verification of the Planning implementation. Tests already reported at previous milestones are reported in grey tables and have a label *old* with the Test ID.

Test ID	test_supply_chain_activity_repository (*old*)
Test objective	Test Supply Chain Activity repository operations.
Verified requirements	*ENF_PLAN_R1*, *ENF_PLAN_R2*, *ENF_PLAN_R3*, *ENF_PLAN_R4*, *ENF_PLAN_R10*, *ENF_PLAN_R11*, *ENF_PLAN_R12*
Inputs	A test Supply Chain Activity object.
Expected results	All operations execute successfully.
Outputs	None.
Comments	All operations executed successfully.

Test ID	test_supply_chain_repository (*old*)
Test objective	Test Supply Chain repository operations.
Verified requirements	*ENF_PLAN_R1*, *ENF_PLAN_R2*, *ENF_PLAN_R3*, *ENF_PLAN_R4*, *ENF_PLAN_R10*, *ENF_PLAN_R11*, *ENF_PLAN_R12*
Inputs	A test Supply Chain object.
Expected results	All operations execute successfully.
Outputs	None.
Comments	All operations executed successfully.

Test ID	test_planning_activity_repository (*old*)
Test objective	Test Planning Activity repository operations.
Verified requirements	*ENF_PLAN_R1*, *ENF_PLAN_R7*
Inputs	A test Planning Activity object.
Expected results	All operations execute successfully.
Outputs	None.
Comments	All operations executed successfully.

Test ID	test_supply_chain_activity_service (*old*)
Test objective	Test service class that provides operations for dealing with Supply Chain Activity objects.
Verified requirements	*ENF_PLAN_R1*, *ENF_PLAN_R2*, *ENF_PLAN_R3*, *ENF_PLAN_R4*, *ENF_PLAN_R10*, *ENF_PLAN_R11*, *ENF_PLAN_R12*
Inputs	• A test SLA Template document in the XML format. • A test Supply Chain Activity input data (as sent by the Supply Chain Manager component).
Expected results	• A valid Supply Chain Activity is created. • A valid Supply Chain is created. • All operations execute successfully.
Outputs	None.
Comments	• All operations executed successfully. • Uses SLA Platform's Service Manager mock service.

Test ID	test_supply_chain_activity_service_error_behaviour (*old*)
Test objective	Test the *build supply chains* method's behaviour when an invalid input data is given.
Verified requirements	*ENF_PLAN_R1*, *ENF_PLAN_R2*, *ENF_PLAN_R3*, *ENF_PLAN_R4*, *ENF_PLAN_R10*, *ENF_PLAN_R11*, *ENF_PLAN_R12*
Inputs	• A test SLA Template document in the XML format. • An invalid test Supply Chain Activity input data.
Expected results	• The status of the created Supply Chain Activity is ERROR. • The annotation contains the error description and the stack trace.
Outputs	None.
Comments	All operations executed as expected.

Test ID	test_supply_chain_service (*old*)
Test objective	Test service class that provides operations for dealing with Supply Chain objects.
Verified requirements	*ENF_PLAN_R1*, *ENF_PLAN_R2*, *ENF_PLAN_R3*, *ENF_PLAN_R4*, *ENF_PLAN_R10*, *ENF_PLAN_R11*, *ENF_PLAN_R12*
Inputs	A test Supply Chain object.
Expected results	All operations execute successfully.
Outputs	None.
Comments	All operations executed successfully.

Test ID	test_planning_activity_service (*old*)
Test objective	Test service class that provides operations for dealing with Planning Activity objects.
Verified requirements	*ENF_PLAN_R1*, *ENF_PLAN_R7*
Inputs	A test Supply Chain object.
Expected results	• A valid Implementation Plan object is created. • All operations execute successfully.
Outputs	None.
Comments	Uses Enforcement Implementation mock service and Monitoring module mock service.

Test ID	test_planning_activity_service_error_behaviour (*old*)
Test objective	Test the *create planning activity* method's behaviour when an invalid input data is given.
Verified requirements	*ENF_PLAN_R1*, *ENF_PLAN_R7*
Inputs	A test Supply Chain object with some invalid data.
Expected results	• The status of the created Planning Activity is ERROR. • The annotation contains the error description and the stack trace.
Outputs	None.
Comments	All operations executed as expected.

Test ID	test_solver (*old*)
Test objective	Test the Solver functionality (building supply chains).
Verified requirements	*ENF_PLAN_R1*, *ENF_PLAN_R2*, *ENF_PLAN_R3*, *ENF_PLAN_R4*, *ENF_PLAN_R10*, *ENF_PLAN_R11*, *ENF_PLAN_R12*
Inputs	A test Supply Chain Activity object.
Expected results	Valid Supply Chain objects are created corresponding to the Supply Chain Activity data.
Outputs	None.
Comments	All operations executed as expected.

Test ID	test_dump (*old*)
Test objective	Test dump method of the JsonDumper class.

Verified requirements	/
Inputs	A test Java object.
Expected results	Java object is serialized to a valid JSON string.
Outputs	None.
Comments	All operations executed as expected.

Test ID	test_app_config (*old*)
Test objective	Test application configuration loading from a file.
Verified requirements	/
Inputs	A test application configuration file.
Expected results	Application configuration properties are set correctly.
Outputs	None.
Comments	All operations executed as expected.

Test ID	test_api_supply_chains (*old*)
Test objective	Perform full test of the functionalities of the Planning component related to supply chains through the REST API provided by the planning-api (part of the Planning component).
Verified requirements	*ENF_PLAN_R1*, *ENF_PLAN_R2*, *ENF_PLAN_R3*, *ENF_PLAN_R4*, *ENF_PLAN_R10*, *ENF_PLAN_R11*, *ENF_PLAN_R12*
Inputs	• A test SLA Template document in the XML format. • A test supply chain activity input data (as sent by the Supply Chain Manager).
Expected results	• A valid Supply Chain Activity object is created. • A valid Supply Chain objects are created according to the input SLA Template, Supply Chain Activity object input data, and security mechanisms. • All operations execute successfully.
Outputs	None.
Comments	• Uses SLA Platform's Service Manager mock service. • All operations executed as expected.

Test ID	test_api_planning_activity (*old*)
Test objective	Perform full test of the functionalities of the Planning component related to planning activities through the REST API provided by the planning-api (part of the Planning component).
Verified requirements	*ENF_PLAN_R1*, *ENF_PLAN_R7*
Inputs	A test Supply Chain object.
Expected results	• A valid Planning Activity object is created. • A valid Implementation Plan object is created according to the input Supply Chain, and security mechanisms. • All operations execute successfully.

Outputs	None.
Comments	• Uses Enforcement's Implementation mock service and Monitoring module mock service. • All operations executed as expected.

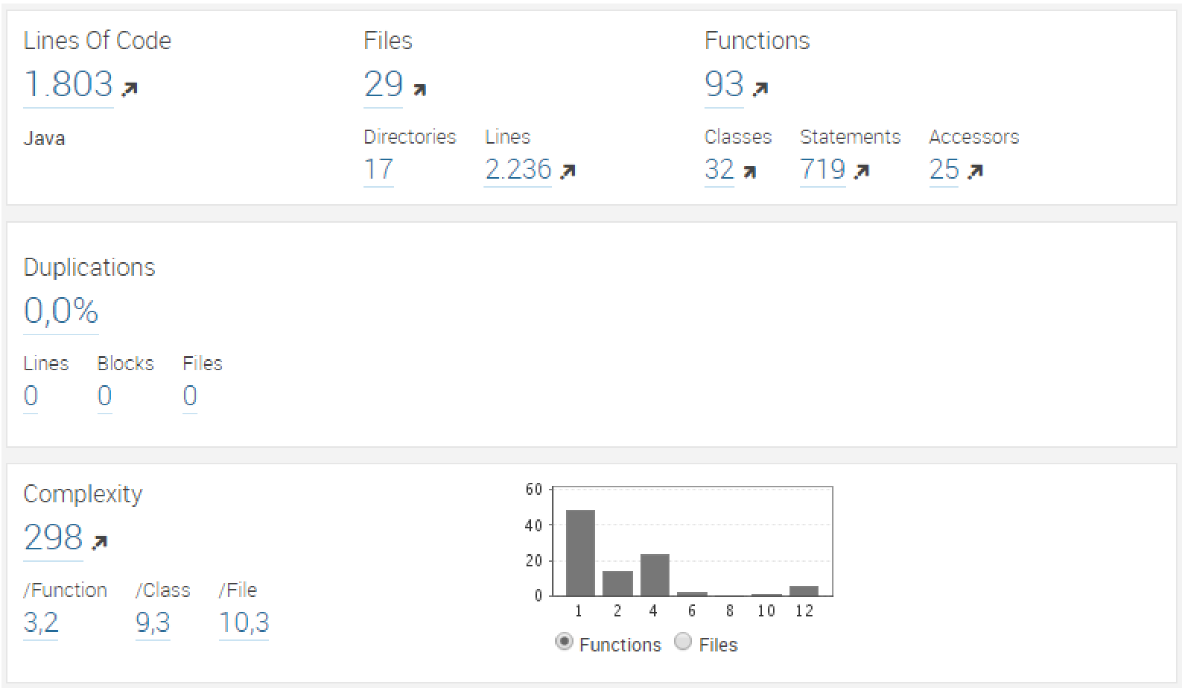
Test ID	test_sla_termination_java
Test objective	Tests planning after SLA termination functionality using the reconfiguration service (ReconfigService) Java API.
Verified requirements	*ENF_PLAN_R5, ENF_PLAN_R6, ENF_PLAN_R8, ENF_IMPL_R10*
Inputs	• A test Planning Activity object in json format.
Expected results	• All operations execute successfully.
Outputs	None.
Comments	• All operations executed successfully. • Uses Implementation, SLA Manager, MoniPoli and Auditing mock objects.

Test ID	test_sla_termination_rest
Test objective	Tests planning after SLA termination functionality using the reconfiguration REST API service (ReconfigController).
Verified requirements	*ENF_PLAN_R5, ENF_PLAN_R6, ENF_PLAN_R8, ENF_IMPL_R10*
Inputs	• A test Planning Activity object in json format.
Expected results	• SLA termination executes successfully.
Outputs	None.
Comments	• All operations executed successfully. • Uses Implementation, SLA Manager, MoniPoli and Auditing mock objects.

Test ID	test_sla_reconfiguration_java
Test objective	Tests planning after SLA renegotiation functionality using the reconfiguration service (ReconfigService) Java API.
Verified requirements	*ENF_PLAN_R5, ENF_PLAN_R6, ENF_PLAN_R8, ENF_IMPL_R10*
Inputs	• A test Planning Activity object in json format. • A test Supply Chain object in json format.
Expected results	• All operations execute successfully, reaction plan is built correctly.
Outputs	None.
Comments	• All operations executed successfully. • Uses Implementation, SLA Manager, MoniPoli and Auditing mock objects.

Test ID	test_sla_reconfiguration_rest
Test objective	Tests planning after SLA renegotiation functionality using the reconfiguration REST API service (ReconfigController).
Verified requirements	*ENF_PLAN_R5, ENF_PLAN_R6, ENF_PLAN_R8, ENF_IMPL_R10*
Inputs	• A test Planning Activity object in json format. • A test Supply Chain object in json format.
Expected results	• All operations execute successfully, reaction plan is built correctly.
Outputs	None.
Comments	• All operations executed successfully. • Uses Implementation, SLA Manager, MoniPoli and Auditing mock objects.

As seen from the code quality report shown in Figure 5 and Figure 6, the tests outlined above cover 74.9% of all code.


Lines Of Code 1.803 ↗ Java

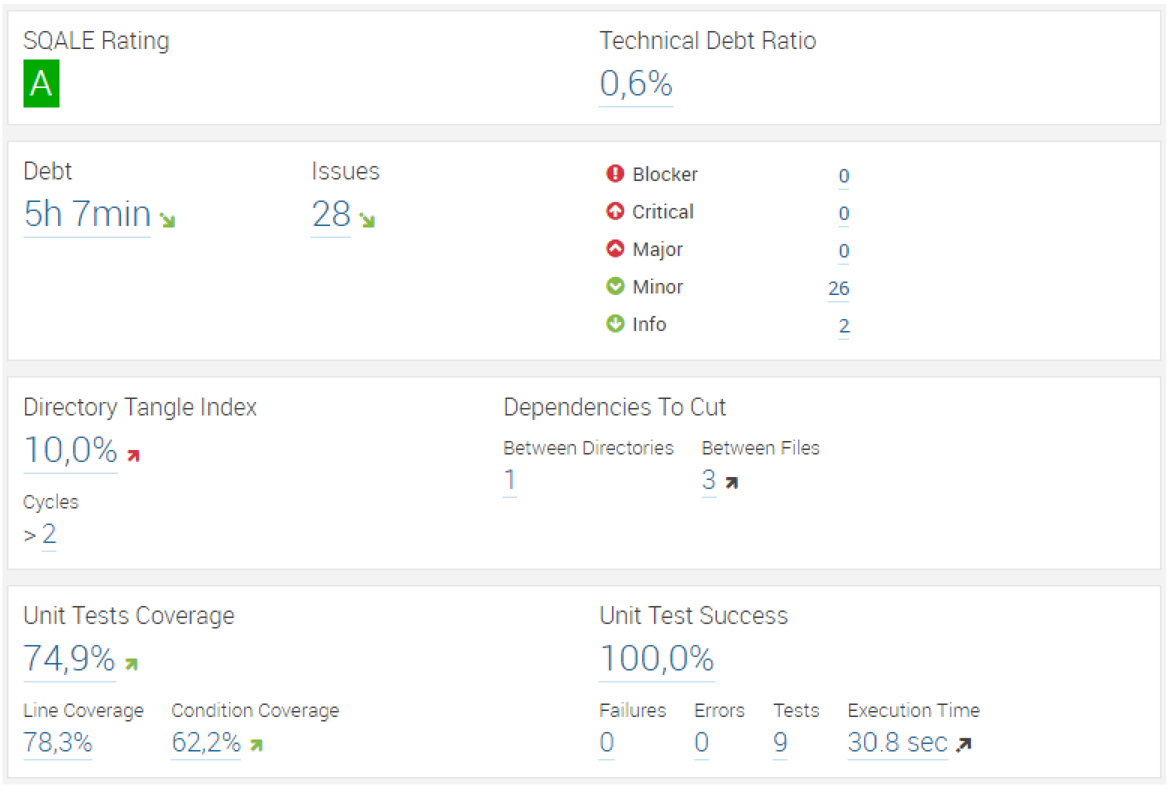
Files 29 ↗ Directories 17 Lines 2.236 ↗

Functions 93 ↗ Classes 32 ↗ Statements 719 ↗ Accessors 25 ↗

Duplications 0,0% Lines 0 Blocks 0 Files 0

Complexity 298 ↗ /Function 3,2 /Class 9,3 /File 10,3

Figure 5. Code quality report for Planning - 1/2


SQALE Rating
A

Technical Debt Ratio
0,6%

Debt
5h 7min

Issues
28

Blocker	0
Critical	0
Major	0
Minor	26
Info	2

Directory Tangle Index
10,0%

Cycles
> 2

Dependencies To Cut

Between Directories	Between Files
1	3

Unit Tests Coverage
74,9%

Line Coverage	Condition Coverage
78,3%	62,2%

Unit Test Success
100,0%

Failures	Errors	Tests	Execution Time
0	0	9	30.8 sec

Figure 6. Code quality report for Planning - 2/2

5.1.2. Implementation and Broker

During the SLA implementation phase, the Implementation component oversees the execution of the implementation plan built by the Planning component. In particular, the Implementation component triggers the Broker to acquire resources specified in the plan and then deploys and configures them through Chef (in SPECS, all configuration actions are managed with Chef [4]).

In the following tables we report the set of tests executed for the Implementation component.

Test ID	test_implementation_plan_service_java
Test objective	Tests implementation plan service functionality using the Java API (ImplPlanService): storing and retrieving implementation plans.
Verified requirements	*ENF_IMPL_R1, ENF_IMPL_R7*
Inputs	A test implementation plan in json format.
Expected results	All operations execute successfully.
Outputs	None.
Comments	All operations executed successfully.

Test ID	test_implementation_plan_service_rest
Test objective	Tests implementation plan service functionality using the REST API (ImplPlanController): storing and retrieving implementation plans.
Verified requirements	*ENF_IMPL_R1, ENF_IMPL_R7*
Inputs	A test implementation plan in json format.

Expected results	All operations execute successfully.
Outputs	None.
Comments	All operations executed successfully.

Test ID	test_remediation_plan_service_java
Test objective	Tests remediation plan service functionality using the Java API (RemPlanService): implementing remediation plan, executing remediation actions, retrieving measurements, evaluating monitoring event conditions, etc.
Verified requirements	*ENF_IMPL_R7, ENF_IMPL_R8, ENF_IMPL_R9*
Inputs	• A test remediation plan in JSON format. • A test implementation plan in JSON format. • A test monitoring event in JSON format.
Expected results	All operations execute successfully, remediation actions execute correctly.
Outputs	None.
Comments	• All operations executed successfully. • Uses Broker and Event Archiver mock objects.

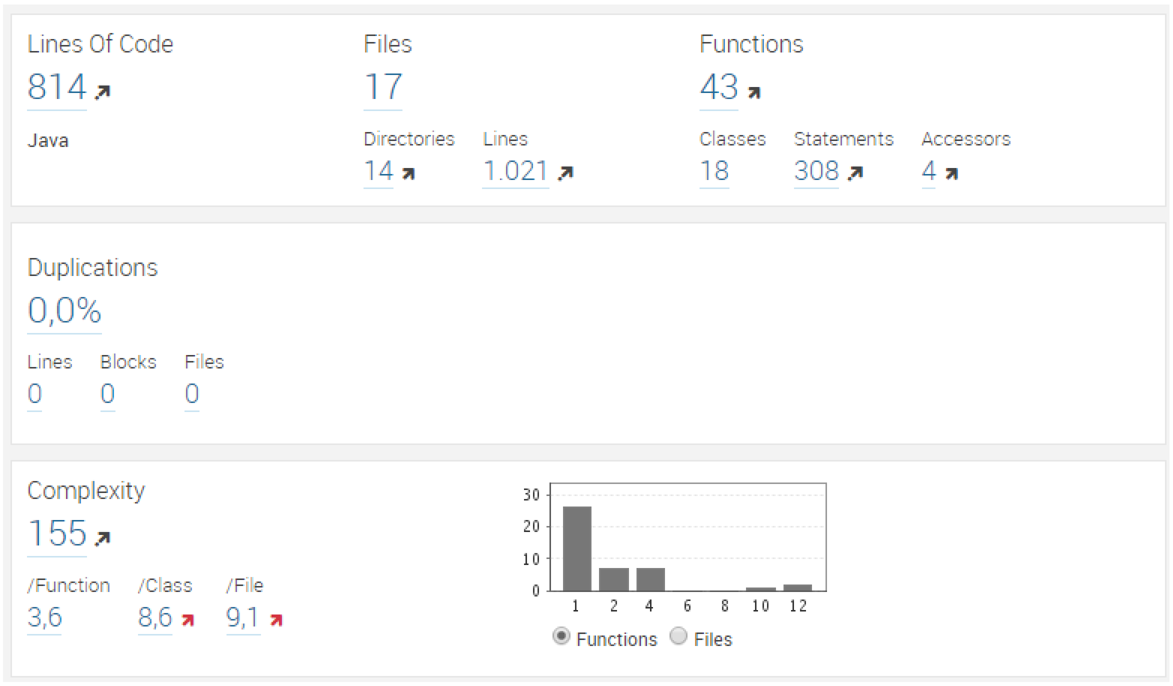
Test ID	test_remediation_plan_service_rest
Test objective	Tests remediation plan service functionality using the REST API (RemPlanController): implementing remediation plan, executing remediation actions, retrieving measurements, evaluating monitoring event conditions, etc.
Verified requirements	*ENF_IMPL_R7, ENF_IMPL_R8, ENF_IMPL_R9*
Inputs	• A test remediation plan in JSON format. • A test implementation plan in JSON format. • A test monitoring event in JSON format.
Expected results	All operations execute successfully, remediation actions execute correctly.
Outputs	None.
Comments	• All operations executed successfully. • Uses Broker and Event Archiver mock objects.

Test ID	test_implementation_activity_service_java
Test objective	Tests implementation activity service functionality using the Java API (ImplActivityService): creating implementation activity, implementing implementation plan, retrieving and deleting implementation activity.
Verified requirements	*ENF_IMPL_R1, ENF_IMPL_R6, ENF_IMPL_R7, ENF_IMPL_R8*
Inputs	A test implementation plan in JSON format.
Expected results	All operations execute successfully, implementation activity is

	created correctly, the activity status is set correctly (passes from CREATED to ACTIVE).
Outputs	None.
Comments	• All operations executed successfully. • Uses Broker mock object.

Test ID	test_implementation_activity_service_rest
Test objective	Tests implementation activity service functionality using the REST API (ImplActivityController): creating implementation activity, implementing implementation plan, retrieving and deleting implementation activity.
Verified requirements	*ENF_IMPL_R1, ENF_IMPL_R6, ENF_IMPL_R7, ENF_IMPL_R8*
Inputs	A test implementation plan in JSON format.
Expected results	All operations execute successfully, implementation activity is created correctly, the activity status is set correctly (passes from CREATED to ACTIVE).
Outputs	None.
Comments	• All operations executed successfully. • Uses Broker mock object.

The code quality reports in the figures below show that the tests above cover 61.5% of the codebase for the Implementation component.



Lines Of Code
814
Java

Files
17

Directories	Lines
14	1.021

Functions
43

Classes	Statements	Accessors
18	308	4

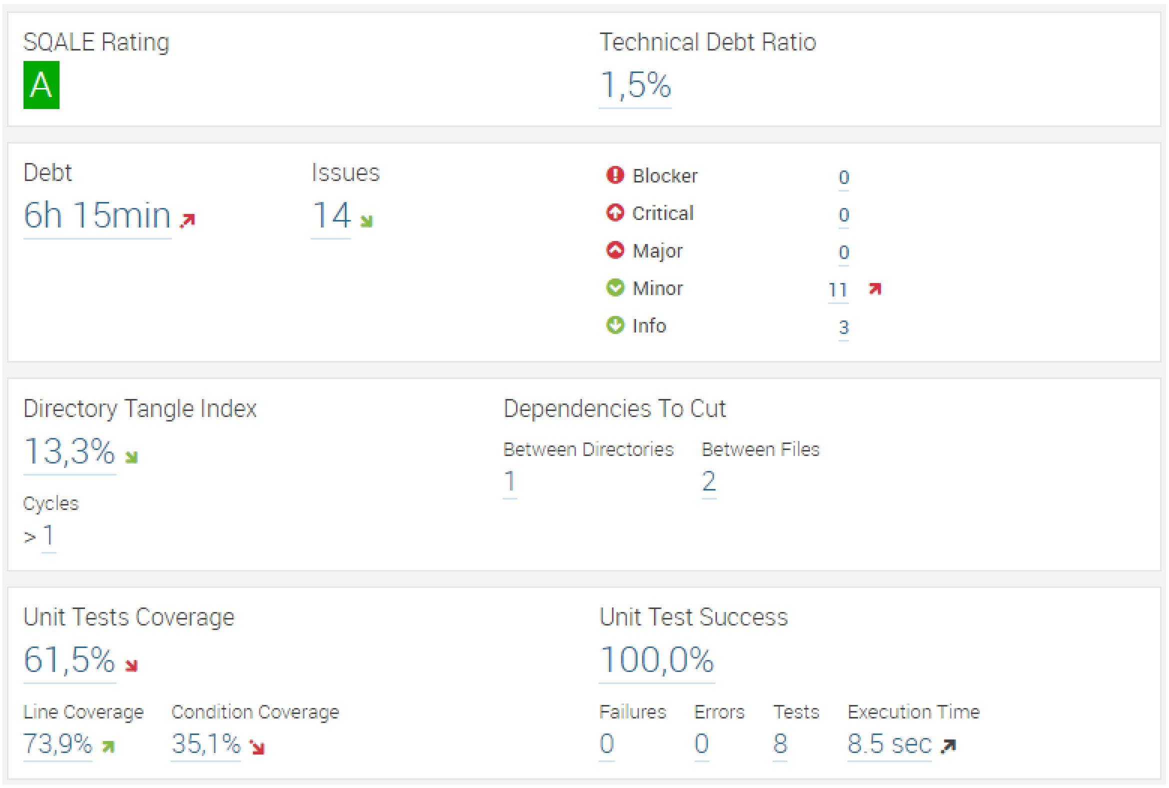
Duplications
0,0%

Lines	Blocks	Files
0	0	0

Complexity
155

/Function	/Class	/File
3,6	8,6	9,1

Figure 7. Code quality report for Implementation - 1/2


SQALE Rating
A
Technical Debt Ratio
1,5%

Debt
6h 15min
Issues
14

Blocker	0
Critical	0
Major	0
Minor	11
Info	3

Directory Tangle Index
13,3%
Cycles
> 1
Dependencies To Cut
Between Directories 1
Between Files 2

Unit Tests Coverage
61,5%
Line Coverage 73,9%
Condition Coverage 35,1%

Unit Test Success
100,0%

Failures	Errors	Tests	Execution Time
0	0	8	8.5 sec

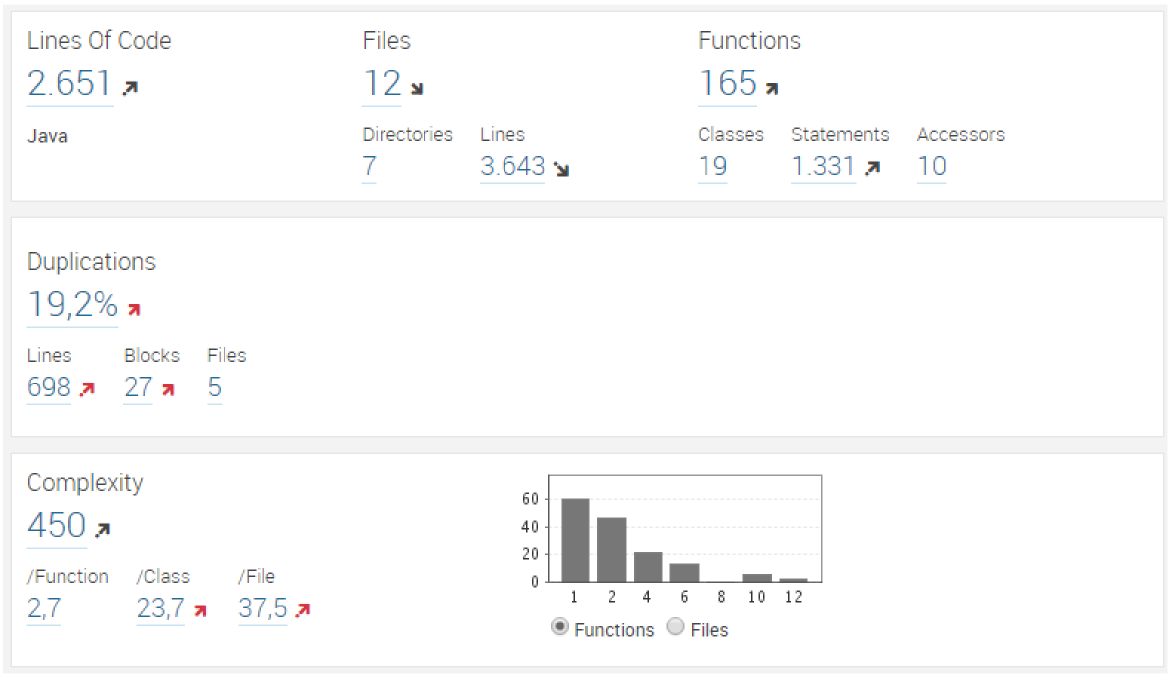
Figure 8. Code quality report for Implementation - 2/2

In the following tables we report the set of tests executed for the Broker component.

Test ID	test_upload_databag_item
Test objective	Tests the upload of an implementation plan on the chef server.
Verified requirements	/
Inputs	A test implementation plan in JSON format.
Expected results	The test implementation plan is stored properly on Chef Server.
Outputs	None.
Comments	All operations executed successfully.

Test ID	test_acquisition_Vm_on_Eucalyptus
Test objective	Tests the acquisition of VMs according to the implementation plan.
Verified requirements	*ENF_IMPL_R2, ENF_IMPL_R3, ENF_IMPL_R4, ENF_IMPL_R5, ENF_BROKER_R1, ENF_BROKER_R2, ENF_BROKER_R3, ENF_BROKER_R4, ENF_BROKER_R5*
Inputs	A test implementation plan in JSON format.
Expected results	The acquired resources and their configurations have to be compliant with what is defined in the implementation plan.
Outputs	None.
Comments	All operations executed successfully.

The code quality reports in the figures below show that all presented tests cover 23.0% of the codebase for the Broker component. Although one of the development rules was to cover at least 50% of all code, it is impossible to test all functionalities associated to the acquisition and configuration of cloud resources (VMs) in a real-world environment. A significant contributing factor to this is that the acquisition and management of resources requires payment, restricting the capability of the automated tests.



Lines Of Code
2.651 ↗
Java

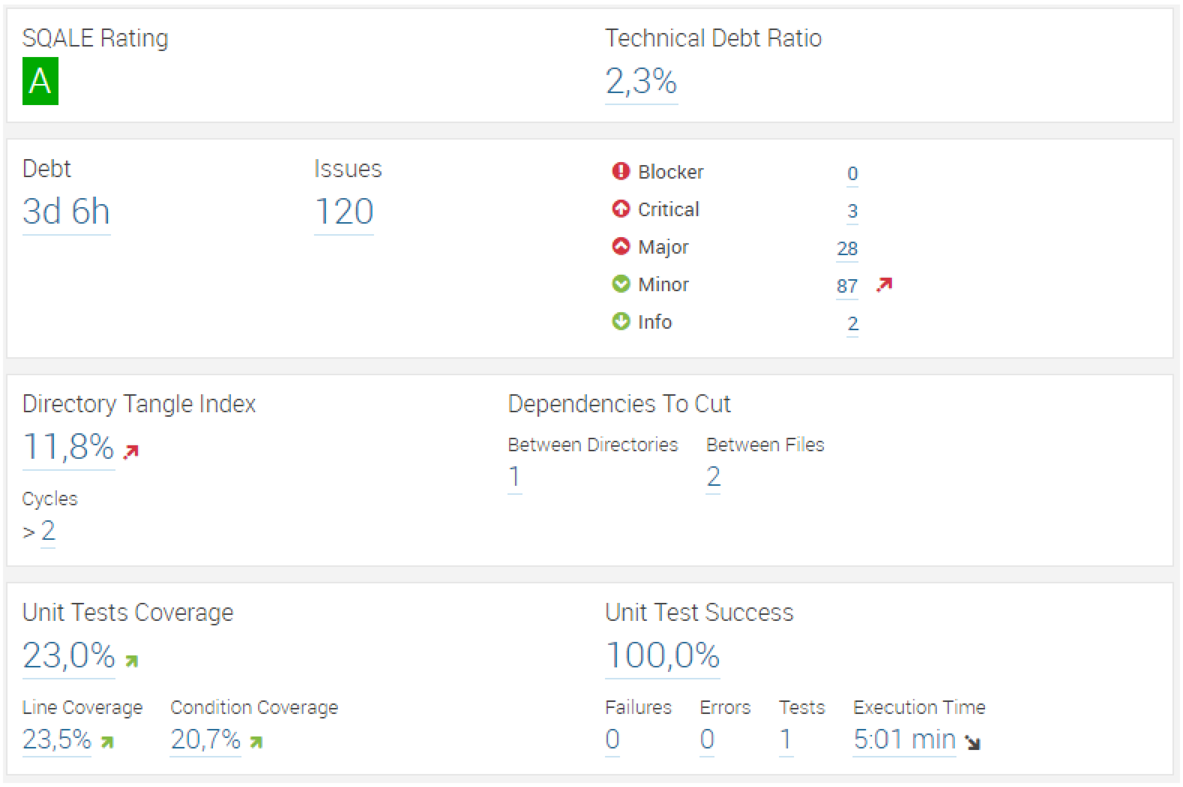
Files
12 ↘
Directories 7
Lines 3.643 ↘

Functions
165 ↗
Classes 19
Statements 1.331 ↗
Accessors 10

Duplications
19,2% ↗
Lines 698 ↗
Blocks 27 ↗
Files 5

Complexity
450 ↗
/Function 2,7
/Class 23,7 ↗
/File 37,5 ↗

Figure 9. Code quality report for Broker - 1/2



SQALE Rating
A

Technical Debt Ratio
2,3%

Debt
3d 6h

Issues
120

Blocker	0
Critical	3
Major	28
Minor	87 ↗
Info	2

Directory Tangle Index
11,8% ↗
Cycles
> 2

Dependencies To Cut
Between Directories 1
Between Files 2

Unit Tests Coverage
23,0% ↗
Line Coverage 23,5% ↗
Condition Coverage 20,7% ↗

Unit Test Success
100,0%
Failures 0
Errors 0
Tests 1
Execution Time 5:01 min ↘

Figure 10. Code quality report for Broker - 2/2

The code quality report for the Broker component reveals a few critical and major issues. These issues are mainly related to logging exceptions and printing format, which we intend to fix during the exploitation activities after the end of the project.

5.1.3. Diagnosis

Whenever the Monitoring module detects a possible SLA alert or an SLA violation, the Diagnosis component analyses the event. In the following tables we report the set of tests executed for the Diagnosis component.

Test ID	test_diagnosis_full_java
Test objective	Tests the diagnosis full flow using the Java API (DiagnosisActivityService and NotificationService): creating diagnosis activity for the given notification, classifying monitoring event, processing diagnosis activity, calling remediation, retrieving diagnosis activity, retrieving notification, etc.
Verified requirements	*ENF_DIAG_R1- R18*
Inputs	• A test notification in JSON format. • A test implementation plan in JSON format. • A test planning activity in JSON format.
Expected results	All operations execute successfully, diagnosis activity status is set correctly (passes from RECEIVED to SOLVED)
Outputs	None.
Comments	• All operations executed successfully. • Uses SLA Manager, Planning, Implementation, RDS and Event Archiver mock objects.

Test ID	test_diagnosis_full_rest
Test objective	Tests the diagnosis full flow using the REST API (DiagnosisActivityController and NotificationController): creating diagnosis activity for the given notification, classifying monitoring event, processing diagnosis activity, calling remediation, retrieving diagnosis activity, retrieving notification, etc.
Verified requirements	*ENF_DIAG_R1- R18*
Inputs	• A test notification in JSON format. • A test implementation plan in JSON format. • A test planning activity in JSON format.
Expected results	All operations execute successfully, diagnosis activity status is set correctly (passes from RECEIVED to SOLVED).
Outputs	None.
Comments	• All operations executed successfully. • Uses SLA Manager, Planning, Implementation, RDS and Event Archiver mock objects.

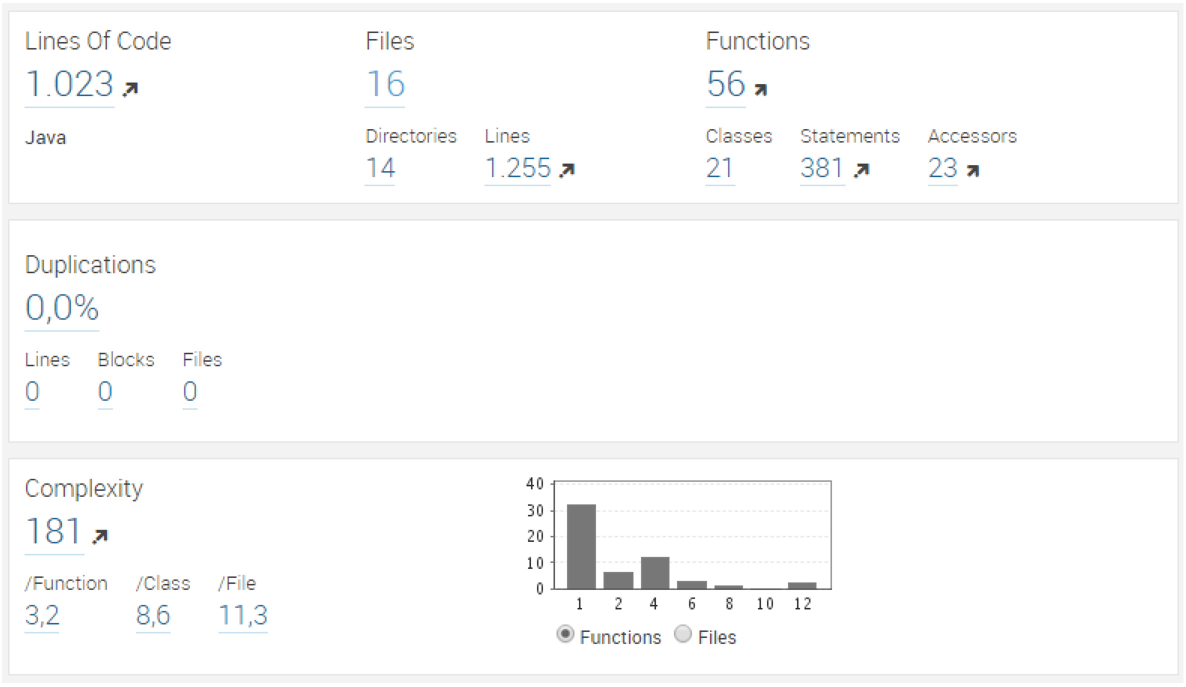
Test ID	test_diagnosis_activity_repository
Test objective	Tests the DiagnosisActivityRepository functionality (storing, retrieving and deleting diagnosis activities, retrieving diagnosis activities by given filter).
Verified requirements	/
Inputs	A test DiagnosisActivity Java object.

Expected results	All operations execute successfully.
Outputs	None.
Comments	All operations executed successfully.

Test ID	test_notification_repository
Test objective	Tests the NotificationRepository functionality (storing, retrieving and deleting notifications, retrieving notifications by given filter).
Verified requirements	/
Inputs	A test Notification Java object.
Expected results	All operations execute successfully.
Outputs	None.
Comments	All operations executed successfully.

Test ID	test_evaluate_condition
Test objective	Tests the evaluateCondition() method (evaluates measurement value against the corresponding monitoring event condition) for different operators and data types.
Verified requirements	*ENF_DIAG_R18*
Inputs	A test Measurement object and measurement value.
Expected results	Conditions are evaluated correctly.
Outputs	None.
Comments	All operations executed successfully.

The code quality reports in the figures below show that all tests above cover 65.9% of the entire code for the Diagnosis component.


Lines Of Code
1.023
Java

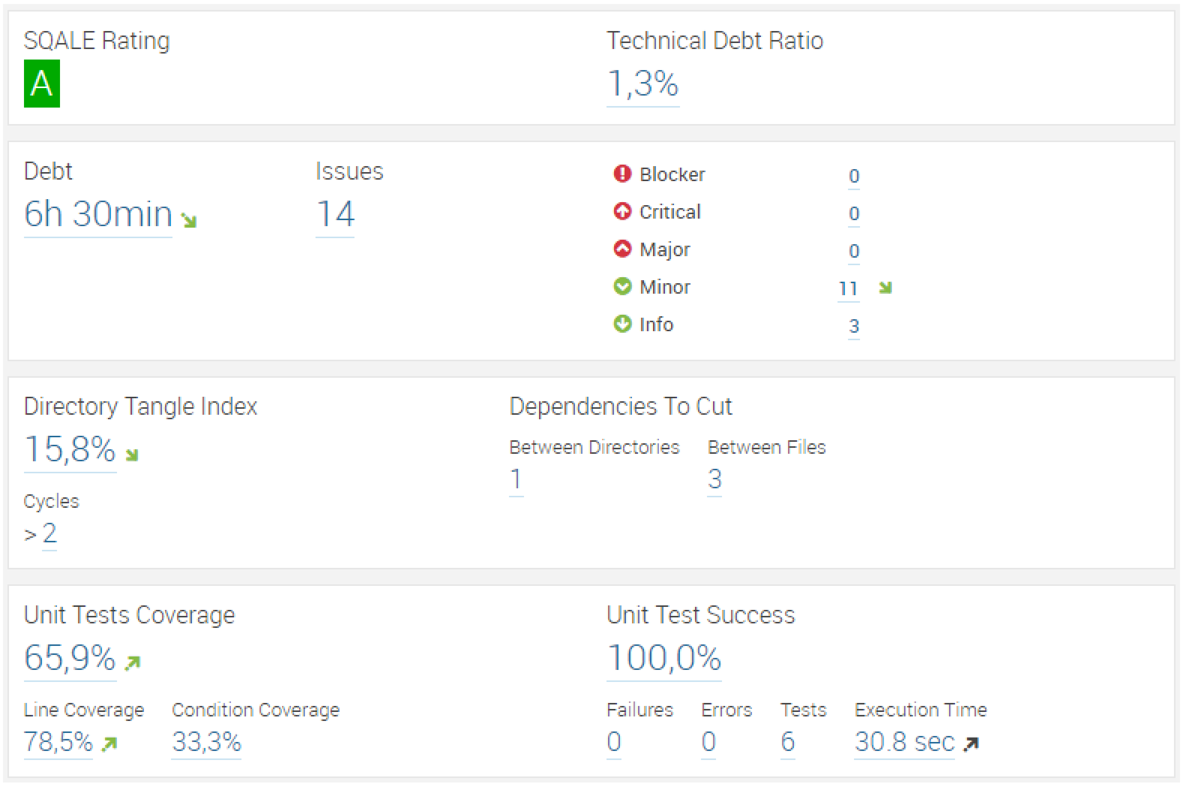
Files
16
Directories 14
Lines 1.255

Functions
56
Classes 21
Statements 381
Accessors 23

Duplications
0,0%
Lines 0
Blocks 0
Files 0

Complexity
181
/Function 3,2
/Class 8,6
/File 11,3

Figure 11. Code quality report for Diagnosis - 1/2



SQALE Rating
A

Technical Debt Ratio
1,3%

Debt	Issues		
6h 30min ↘	14	Blocker	0
		Critical	0
		Major	0
		Minor	11 ↘
		Info	3

Directory Tangle Index
15,8% ↘

Cycles
> 2

Dependencies To Cut

Between Directories	Between Files
1	3

Unit Tests Coverage
65,9% ↗

Line Coverage	Condition Coverage
78,5% ↗	33,3%

Unit Test Success
100,0%

Failures	Errors	Tests	Execution Time
0	0	6	30.8 sec ↗

Figure 12. Code quality report for Diagnosis - 2/2

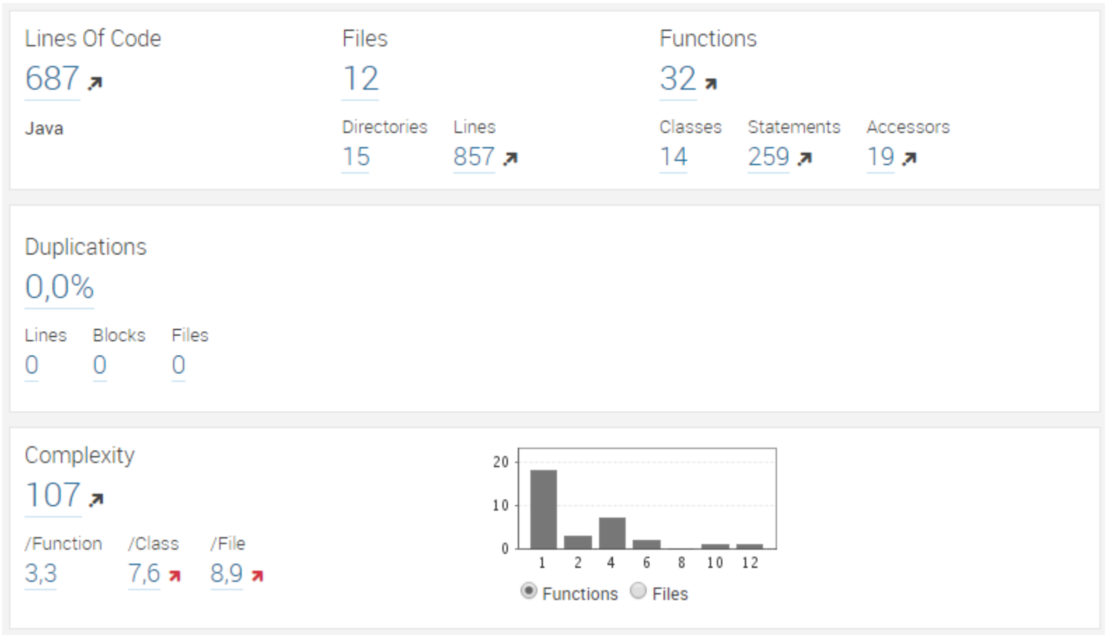
5.1.4. RDS

During the SLA remediation phase, the RDS component identifies countermeasures to be taken to either mitigate the risk of having an SLA violation or recover from it. In the tables below we report all tests performed for the RDS component.

Test ID	test_rds_full_java
Test objective	Tests the remediation full flow using the Java API (RemActivityService): creating remediating activity for the given input data, processing remediating activity, generating remediation plan, implementing remediation plan, etc.
Verified requirements	*ENF_REM_R1-R11, ENF_PLAN_R8-R9, SLANEG_R30-R31*
Inputs	• A test remediation activity input data (as created by the Diagnosis component). • A test implementation plan in JSON format. • A test security mechanism in JSON format corresponding to implementation plan.
Expected results	All operations execute successfully, remediation activity status is set correctly (passes from CREATED to SOLVED), remediation plan is generated correctly.
Outputs	None.
Comments	• All operations executed successfully. • Uses SLA Manager, Service Manager and Implementation mock objects.

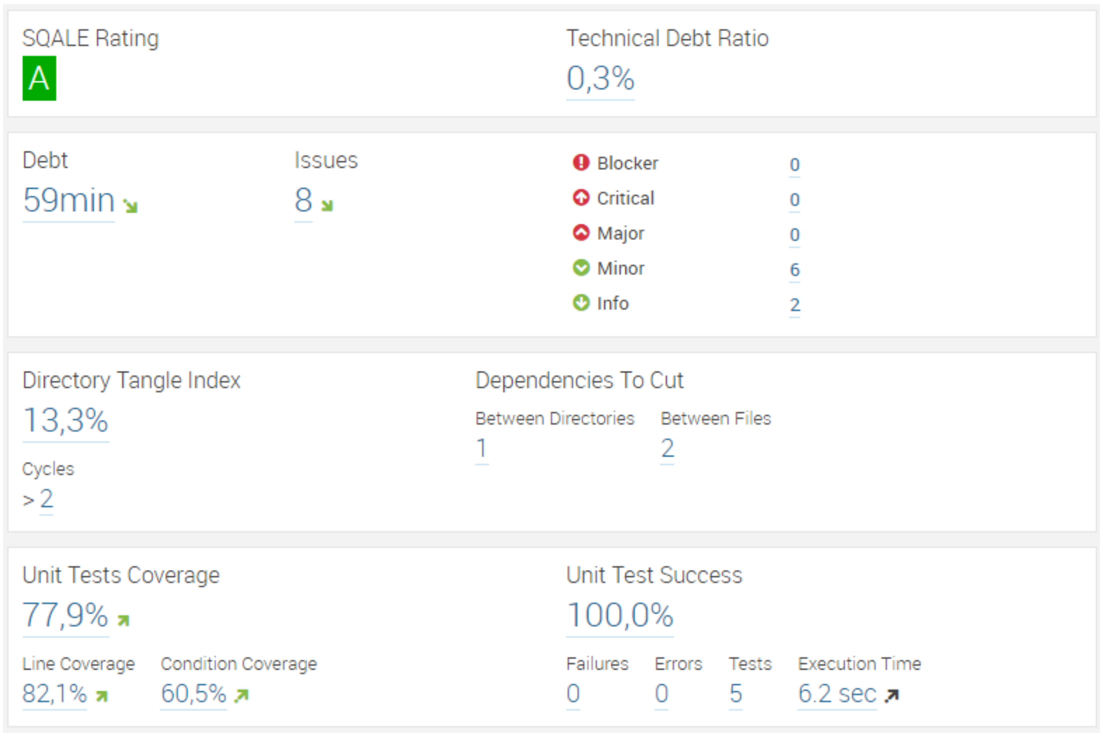
Test ID	test_rds_full_rest
Test objective	Tests the remediation full flow using the REST API (RemActivityController): creating remediating activity for the given input data, processing remediating activity, generating remediation plan, implementing remediation plan, etc.
Verified requirements	*ENF_REM_R1-R11, ENF_PLAN_R8-R9, SLANEG_R30-R31*
Inputs	• A test remediation activity input data (as created by the Diagnosis component). • A test implementation plan in JSON format. • A test security mechanism in JSON format corresponding to implementation plan.
Expected results	All operations execute successfully, remediation activity status is set correctly (passes from CREATED to SOLVED), remediation plan is generated correctly.
Outputs	None.
Comments	• All operations executed successfully. • Uses SLA Manager, Service Manager and Implementation mock objects.

The code quality reports in the figures below show that both tests above cover 77.9% of the entire code for the Diagnosis component.



Lines Of Code	687	Java		
Files	12	Directories 15	Lines 857	
Functions	32	Classes 14	Statements 259	Accessors 19
Duplications	0,0%	Lines 0	Blocks 0	Files 0
Complexity	107	/Function 3,3	/Class 7,6	/File 8,9

Figure 13. Code quality report for RDS - 1/2



SQALE Rating	A
Technical Debt Ratio	0,3%
Debt	59min
Issues	8
Blocker	0
Critical	0
Major	0
Minor	6
Info	2
Directory Tangle Index	13,3%
Cycles	> 2
Dependencies To Cut – Between Directories	1
Dependencies To Cut – Between Files	2
Unit Tests Coverage	77,9%
Line Coverage	82,1%
Condition Coverage	60,5%
Unit Test Success	100,0%
Failures	0
Errors	0
Tests	5
Execution Time	6.2 sec

Figure 14. Code quality report for RDS - 2/2

5.2. Security mechanisms

In the following subsections we report unit tests for security mechanisms developed under the umbrella of the Enforcement module.

5.2.1. WebPool

The WebPool mechanism is the mandatory security mechanism for the secure web server service. It provides a pool of web containers for hosting web applications, and is able to ensure redundancy and diversity, while also providing load balancing and session sharing features.

In the following tables we present a set of unit tests implemented for the WebPool mechanism.

Test ID	test_load_balancer_running
Test objective	Test if the configuration of the load balancer is correct and it works properly.
Verified requirements	*ENF_POOL_R2*
Inputs	IP of the load balancer.
Expected results	If more than one replica is active, the load balancer forwards the requests to the different replicas based on a round robin policy.
Outputs	None.
Comments	All operations executed successfully.

Test ID	test_diversity
Test objective	Test if the mechanism correctly instantiates different web container replicas based on user requirements.
Verified	*ENF_POOL_R1*

requirements	
Inputs	IP of the application (load balancer).
Expected results	The request is served by the different replicas according to the established load balancing policy .
Outputs	None.
Comments	All operations executed successfully.

Test ID	test_survivability
Test objective	Test if, in case of an incident occurring in one of the active replicas, the application running on them is still accessible.
Verified requirements	*ENF_POOL_R3*
Inputs	IP of the application (load balancer).
Expected results	The application is still accessible.
Outputs	None.
Comments	All operations executed successfully.

Test ID	test_incident_management
Test objective	Test if, in case of a security incident occurring in one of the active replicas, the application running on them is still accessible, and the level of redundancy and diversity requested by the End-user are restored.
Verified requirements	*ENF_POOL_R5*
Inputs	IP of the application (load balancer).
Expected results	The HAProxy detects that a web container replica is down and notifies the Monitoring module. The Monitoring module generates a monitoring event for the Enforcement module, which detects a violation. A remediation action is triggered, consisting in isolating the affected VM and acquiring and configuring a new VM.
Outputs	None.
Comments	All operations executed successfully.

Test ID	test_session_sharing
Test objective	Test if the configuration of the caching system is correct and it works properly.
Verified requirements	*ENF_POOL_R4*
Inputs	IP of the load balancer.
Expected results	The session information associated with a user is maintained on all active replicas.
Outputs	None.
Comments	All operations executed successfully.

Test ID	test_webcontainer_running
Test objective	Test if the configuration of the web server is correct and it works properly.

Verified requirements	*ENF_POOL_R1, ENF_POOL_R2, ENF_POOL_R3, ENF_POOL_R4, ENF_POOL_R5*
Inputs	IP of the web server to check.
Expected results	The web server is up and accessible.
Outputs	None.
Comments	All operations executed successfully.

5.2.2. TLS

The TLS security mechanism enforces a secure communication between the services hosted by the SPECS security mechanisms and external entities. TLS security mechanism translates a list of high level requirements (metrics, in SPECS terminology) into specific templates that are applied as configuration snippets to the HAProxy component (TLS Terminator). In this way, the TLS layer is enforced to be used as a communication layer for securing both data integrity and privacy between two or more entities.

The following tests are conducted in order to prove the correctness of the implementation of the component.

Test ID	test_tls_constraints_read
Test objective	Test if the TLS constraints are read correctly.
Verified requirements	*ENF_TLS_R1*
Inputs	LIST of TLS constraints.
Expected results	All operations execute successfully.
Outputs	None.
Comments	All operations executed as expected.

Test ID	validate_tls_constraints
Test objective	Test if the TLS constraints are supported and correct defined.
Verified requirements	*ENF_TLS_R1*
Inputs	LIST of TLS constraints.
Expected results	All operations execute successfully.
Outputs	None.
Comments	All operations executed as expected.

Test ID	test_tls_constraints
Test objective	Test if the TLS Endpoint is alive and responsive.
Verified requirements	*ENF_TLS_R1*
Inputs	TLS Endpoint address.
Expected results	TLS Endpoint is responsive.
Outputs	In case of errors: notify the Enforcement component about the error (HTTP call).
Comments	All operations executed as expected.

Test ID	test_if_tls_constraints_are_translated_into_templates
Test objective	Test if the TLS constraints correctly translated into configuration templates without overlaps.
Verified requirements	*ENF_TLS_R2*
Inputs	LIST of TLS constraints.
Expected results	The output template is correctly generated.
Outputs	low level configuration template.
Comments	All operations executed as.

Test ID	validate_tls_configuration.
Test objective	Test if the TLS configuration template is valid and TLS Terminator is able to instantiate it.
Verified requirements	*ENF_TLS_R3*
Inputs	LIST of TLS constraints.
Expected results	All operations execute successfully and TLS Terminator is able to check and validate the generated configuration.
Outputs	None.
Comments	All operations executed as expected.

Test ID	validate_tls_configuration_template_with_a_checksum
Test objective	Generate a checksum of the valid configuration template.
Verified requirements	*ENF_TLS_R3*
Inputs	LIST of TLS constraints.
Expected results	A checksum string.
Outputs	None.
Comments	All operations executed as expected.

Test ID	deploy_tls_configuration_template
Test objective	Test if the TLS configuration template is correctly deployed and default TLS Terminator instance is able to read the configuration file.
Verified requirements	*ENF_TLS_R4*
Inputs	Validated TLS configuration template.
Expected results	Default TLS Terminator instance starts without errors.
Outputs	None.
Comments	All operations executed as expected.

Test ID	test_tls_endpoint_health
Test objective	Test, periodically, if the TLS Endpoint is alive and responsive.
Verified requirements	*ENF_TLS_R5*
Inputs	TLS Endpoint address (extracted from the validated TLS

	Configuration template).
Expected results	TLS Endpoint is responsive.
Outputs	In case of errors: notify Enforcement component. (HTTP call).
Comments	All operations executed as expected.

5.2.3. SVA

The SVA security mechanism enhances cloud services with periodic vulnerability scans, updates of the list of published vulnerabilities, and checks for available updates and upgrades.

Although some tests have already been reported in deliverable D4.5.2, for the sake of completeness, we list the entire set of tests defined and executed for the mechanism. Tests already reported at previous milestones are reported in grey tables and have a label *old* with the Test ID.

Test ID	test_download_ovals (*old*)
Test objective	Test if oval (list of published vulnerabilities) is downloaded successfully.
Verified requirements	*ENF_SVA_R1*
Inputs	URL to oval repository.
Expected results	All operations execute successfully.
Outputs	None.
Comments	All operations executed as expected.

Test ID	test_reconfigure_repository (*old*)
Test objective	Test if repository is successfully changed.
Verified requirements	*ENF_SVA_R1*
Inputs	None.
Expected results	All operations execute successfully.
Outputs	None.
Comments	All operations executed as expected.

Test ID	test_run_scanner (*old*)
Test objective	Test if the scanner generates the scanning report.
Verified requirements	*ENF_SVA_R1*
Inputs	A test vulnerability list.
Expected results	All operations execute successfully.
Outputs	None.
Comments	All operations executed as expected.

Test ID	test_send_scanning_report (*old*)
Test objective	Test if the scanning report is successfully sent to the Django server.
Verified requirements	*ENF_SVA_R2*
Inputs	Django server IP.

Expected results	All operations execute successfully.
Outputs	None.
Comments	• The Django server is the server used by the SVA Dashboard. • All operations executed as expected.

Test ID	test_generate_up_report (*old*)
Test objective	Test if the update/upgrade report is successfully generated.
Verified requirements	*ENF_SVA_R1*
Inputs	None.
Expected results	All operations execute successfully.
Outputs	None.
Comments	All operations executed as expected.

Test ID	test_send_up_report (*old*)
Test objective	Test if the update/upgrade report is successfully sent to the Django server.
Verified requirements	*ENF_SVA_R2*
Inputs	None.
Expected results	All operations execute successfully.
Outputs	None.
Comments	• The Django server is the server used by the SVA Dashboard. • All operations executed as expected.

Test ID	test_vulnerability_list_command (*old*)
Test objective	Test if the vulnerability list command is executed without errors.
Verified requirements	*ENF_SVA_R1*
Inputs	None.
Expected results	All operations execute successfully.
Outputs	None.
Comments	All operations executed as expected.

Test ID	test_vulnerability_scan_command (*old*)
Test objective	Test if vulnerability scan command is executed without errors.
Verified requirements	*ENF_SVA_R1*
Inputs	A test vulnerability list.
Expected results	All operations execute successfully.
Outputs	None.
Comments	All operations executed as expected.

Test ID	test_up_report_command (*old*)
Test objective	Test if update/upgrade report command is executed without

	errors.
Verified requirements	*ENF_SVA_R1*
Inputs	None.
Expected results	All operations execute successfully.
Outputs	None.
Comments	All operations executed as expected.

Test ID	test_reconfigure_repository_command (*old*)
Test objective	Test if reconfigure repository command is executed without errors.
Verified requirements	*ENF_SVA_R1*
Inputs	None.
Expected results	All operations execute successfully.
Outputs	None.
Comments	All operations executed as expected.

Test ID	test_get_list_age_if_list_update_frequency_not_selected (*old*)
Test objective	Tests if the *vulnerability list age* measurement is taken in case when the *list update frequency* metric is not selected.
Verified requirements	*ENF_SVA_R1*
Inputs	None.
Expected results	All operations execute successfully (measurement is not taken).
Outputs	None.
Comments	• When the *list update frequency* metric is not selected, the SVA Monitoring component should not be taking *vulnerability list age* measurements. • All operations executed as expected.

Test ID	test_send_list_age (*old*)
Test objective	Test if the *vulnerability list age* measurement is sent to the Monitoring module (to the Event Hub) and the SVA Dashboard.
Verified requirements	*ENF_SVA_R2*
Inputs	None.
Expected results	All operations execute successfully.
Outputs	None.
Comments	• Both the SVA Dashboard and the Event Hub should be running or else test fails. • All operations executed as expected.

Test ID	test_send_repository_availability (*old*)
Test objective	Test if the *repository availability* measurement is sent to the Monitoring module (to the Event Hub).

Verified requirements	*ENF_SVA_R2*
Inputs	None.
Expected results	All operations execute successfully.
Outputs	None.
Comments	• The Event Hub must be running or else test fails. • All operations executed as expected.

Test ID	test_get_basic_report_age_if_basic_scan_frequency_not_selected (*old*)
Test objective	Test if the *basic scan report age* measurement is taken in case when the *scanning frequency – basic scan* metric is not selected.
Verified requirements	*ENF_SVA_R1*
Inputs	None.
Expected results	All operations execute successfully (measurement is not taken).
Outputs	None.
Comments	• When the *scanning frequency – basic scan* metric is not selected, the SVA Monitoring component should not be taking *basic scan report age* measurements. • All operations executed as expected.

Test ID	test_send_basic_report_age (*old*)
Test objective	Test if the *basic scan report age* measurement is sent to the Monitoring module (to the Event Hub) and the SVA Dashboard.
Verified requirements	*ENF_SVA_R2*
Inputs	None.
Expected results	All operations execute successfully.
Outputs	None.
Comments	• Both the SVA Dashboard and the Event Hub should be running or else test fails. • All operations executed as expected.

Test ID	test_send_list_availability (*old*)
Test objective	Test if the *list availability* measurement is sent to the Monitoring module (to the Event Hub).
Verified requirements	*ENF_SVA_R2*
Inputs	None.
Expected results	All operations execute successfully.
Outputs	None.
Comments	• The Event Hub must be running or else test fails. • All operations executed as expected.

Test ID	test_send_scanner_availability (*old*)
Test objective	Test if the *scanner availability* measurement is sent to the Monitoring module (to the Event Hub).
Verified requirements	*ENF_SVA_R2*
Inputs	None.
Expected results	All operations execute successfully.
Outputs	None.
Comments	• The Event Hub must be running or else test fails. • All operations executed as expected.

Test ID	test_get_up_report_age_if_up_report_frequency_not_selected (*old*)
Test objective	Test if the *update/upgrade report age* measurement is taken in case when the *up report frequency* metric is not selected.
Verified requirements	*ENF_SVA_R1*
Inputs	None.
Expected results	All operations execute successfully (measurement is not taken).
Outputs	None.
Comments	• When the *up report frequency* metric is not selected, the SVA Monitoring component should not be taking *update/upgrade report age* measurements. • All operations executed as expected.

Test ID	test_send_up_report_age (*old*)
Test objective	Test if the *update/upgrade report age* measurement is sent to the Monitoring module (to the Event Hub) and the SVA Dashboard.
Verified requirements	*ENF_SVA_R2*
Inputs	None.
Expected results	All operations execute successfully.
Outputs	None.
Comments	• Both the SVA Dashboard and the Event Hub should be running or else test fails. • All operations executed as expected.

Test ID	test_send_scan_report_availability (*old*)
Test objective	Test if the *scan report availability* measurement is sent to the Monitoring module (to the Event Hub).
Verified requirements	*ENF_SVA_R2*
Inputs	None.
Expected results	All operations execute successfully.
Outputs	None.
Comments	• The Event Hub must be running or else test fails. • All operations executed as expected.

Test ID	test_send_up_report_availability (*old*)
Test objective	Test if the *up report availability* measurement is sent to the Monitoring module (to the Event Hub).
Verified requirements	*ENF_SVA_R2*
Inputs	None.
Expected results	All operations execute successfully.
Outputs	None.
Comments	• The Event Hub must be running or else test fails. • All operations executed as expected.

Test ID	test_invoke_msr_repository_availability (*old*)
Test objective	Test if the command is executed without errors.
Verified requirements	*ENF_SVA_R1*
Inputs	None.
Expected results	All operations execute successfully.
Outputs	None.
Comments	All operations executed as expected.

Test ID	test_invoke_msr_list_availability (*old*)
Test objective	Test if the command is executed without errors.
Verified requirements	*ENF_SVA_R1*
Inputs	None.
Expected results	All operations execute successfully.
Outputs	None.
Comments	All operations executed as expected.

Test ID	test_invoke_msr_scanners_availability (*old*)
Test objective	Test if the command is executed without errors.
Verified requirements	*ENF_SVA_R1*
Inputs	None.
Expected results	All operations execute successfully.
Outputs	None.
Comments	All operations executed as expected.

Test ID	test_invoke_msr_scan_report_availability (*old*)
Test objective	Test if the command is executed without errors.
Verified requirements	*ENF_SVA_R1*
Inputs	None.
Expected results	All operations execute successfully.
Outputs	None.
Comments	All operations executed as expected.

Test ID	test_invoke_msr_up_report_availability (*old*)
Test objective	Test if the command is executed without errors.
Verified requirements	*ENF_SVA_R1*
Inputs	None.
Expected results	All operations execute successfully.
Outputs	None.
Comments	All operations executed as expected.

Test ID	test_scan_report_post (*old*)
Test objective	Test if the scan report is uploaded successfully.
Verified requirements	*ENF_SVA_R4*
Inputs	A test scan report.
Expected results	All operations execute successfully.
Outputs	None.
Comments	All operations executed as expected.

Test ID	test_oval_report_post (*old*)
Test objective	Test if the vulnerability list is uploaded successfully.
Verified requirements	*ENF_SVA_R4*
Inputs	A test vulnerability list.
Expected results	All operations execute successfully.
Outputs	None.
Comments	All operations executed as expected.

Test ID	test_up_report_post (*old*)
Test objective	Test if the update/upgrade report is uploaded successfully.
Verified requirements	*ENF_SVA_R4*
Inputs	A test up report.
Expected results	All operations execute successfully.
Outputs	None.
Comments	All operations executed as expected.

Test ID	test_scan_report_available (*old*)
Test objective	Test if the scan report is available in the database after the upload.
Verified requirements	*ENF_SVA_R4*
Inputs	A test scan report.
Expected results	All operations execute successfully.
Outputs	None.
Comments	All operations executed as expected.

Test ID	test_vulnerability_list_available (*old*)
Test objective	Test if the vulnerability list is available in the database after the upload.
Verified requirements	*ENF_SVA_R4*
Inputs	A test vulnerability list.
Expected results	All operations execute successfully.
Outputs	None.
Comments	All operations executed as expected.

Test ID	test_up_report_available (*old*)
Test objective	Test if the update/upgrade report is available in the database after the upload.
Verified requirements	*ENF_SVA_R4*
Inputs	A test up report.
Expected results	All operations execute successfully.
Outputs	None.
Comments	All operations executed as expected.

Test ID	test_report_url (*old*)
Test objective	Test if URL for a VM is available.
Verified requirements	*ENF_SVA_R4*
Inputs	A test URL for a VM.
Expected results	All operations execute successfully.
Outputs	None.
Comments	All operations executed as expected.

Test ID	test_wrong_vm (*old*)
Test objective	Test if the client is redirected to the index page after accessing a virtual machine, which is not in the database.
Verified requirements	*ENF_SVA_R4*
Inputs	A test URL for a VM not in the database.
Expected results	Redirected to index page.
Outputs	None.
Comments	All operations executed as expected.

Test ID	test_oval_file_url (*old*)
Test objective	Test if the client is able to view the vulnerability list.
Verified requirements	*ENF_SVA_R4*
Inputs	A test vulnerability list.
Expected results	All operations execute successfully.
Outputs	None.

Comments	All operations executed as expected.
Test ID	test_scanning_report_file_url (*old*)
Test objective	Test if the client is able to view the scanning report.
Verified requirements	*ENF_SVA_R4*
Inputs	A test scan report.
Expected results	All operations execute successfully.
Outputs	None.
Comments	All operations executed as expected.

Test ID	test_up_report_file_url (*old*)
Test objective	Test if the client is able to view the update/upgrade report.
Verified requirements	*ENF_SVA_R4*
Inputs	A test update/upgrade report.
Expected results	All operations execute successfully.
Outputs	None.
Comments	All operations executed as expected.

Test ID	test_openvas_scanning_execution
Test objective	Test if the openvas tool is able to execute a vulnerability scanning on a target virtual machine.
Verified requirements	*ENF_SVA_R1*
Inputs	IP address of the target machine.
Expected results	The vulnerability scanning is correctly done.
Outputs	None.
Comments	All operations executed as expected.

Test ID	test_openvas_scanning_report
Test objective	Test if the openvas tool is able to report the results of a vulnerability scanning.
Verified requirements	*ENF_SVA_R2*
Inputs	IP address of the target machine.
Expected results	The vulnerability scanning report is correctly returned.
Outputs	None.
Comments	All operations executed as expected.

5.2.4. DBB and E2EE

The DBB is the mandatory security mechanism for the secure storage service. It manages storage with backup and monitors integrity, write-serializability, and read-freshness. In the following tables we report the entire set of unit tests executed for the mechanism.

Test ID	test_account_not_exists
Test objective	Test that an account is not found when a token without an existing corresponding account is used for authentication.
Verified	*ENF_DBB_R1*

requirements	
Inputs	Token.
Expected results	Returns false.
Outputs	None.
Comments	All operations executed as expected.

Test ID	test_account_exists
Test objective	Test that an account is found when a token with an existing corresponding account is used for authentication.
Verified requirements	*ENF_DBB_R1*
Inputs	Token.
Expected results	Returns true.
Outputs	None.
Comments	All operations executed as expected.

Test ID	test_get_account
Test objective	Test that an account is properly returned.
Verified requirements	ENF_DBB_R1
Inputs	Token.
Expected results	Returns account.
Outputs	None.
Comments	All operations executed as expected.

Test ID	test_container_create
Test objective	Test that a container is properly created.
Verified requirements	*ENF_DBB_R1*
Inputs	Token.
Expected results	Returns true.
Outputs	None.
Comments	All operations executed as expected.

Test ID	test_container_record_create
Test objective	Test that a container record is properly created.
Verified requirements	*ENF_DBB_R1*
Inputs	Token.
Expected results	Returns true.
Outputs	None.
Comments	All operations executed as expected.

Test ID	test_container_get
Test objective	Test that a container record is properly retrieved.
Verified	*ENF_DBB_R1*

requirements	
Inputs	Token.
Expected results	Returns container with all its records.
Outputs	None.
Comments	All operations executed as expected.

Test ID	test_container_share
Test objective	Test that a container is properly shared.
Verified requirements	*ENF_DBB_R1*
Inputs	Token.
Expected results	Returns true.
Outputs	None.
Comments	All operations executed as expected.

Test ID	test_get_messages
Test objective	Test that messages from other users are properly retrieved.
Verified requirements	*ENF_DBB_R1*
Inputs	Token.
Expected results	Returns list of messages.
Outputs	None.
Comments	All operations executed as expected.

Test ID	test_container_unshare
Test objective	Test that container is properly unshared.
Verified requirements	*ENF_DBB_R1*
Inputs	Token.
Expected results	Returns true.
Outputs	None.
Comments	All operations executed as expected.

Test ID	test_container_delete
Test objective	Test that container is properly deleted.
Verified requirements	*ENF_DBB_R1*
Inputs	Token.
Expected results	Returns true.
Outputs	None.
Comments	All operations executed as expected.

Test ID	test_require_token_authentication
Test objective	Test that the authentication with a valid token is successful.
Verified requirements	*ENF_DBB_R1*

Inputs	Token.
Expected results	Returns http.StatusOK
Outputs	None.
Comments	All operations executed as expected.

Test ID	test_require_token_authentication_invalid_token
Test objective	Test that the authentication with an invalid token is not successful.
Verified requirements	*ENF_DBB_R1*
Inputs	Token.
Expected results	Returns http.StatusUnauthorized
Outputs	None.
Comments	All operations executed as expected.

Test ID	test_require_token_authentication_wihtout_token
Test objective	Test that the authentication without a token is not successful.
Verified requirements	*ENF_DBB_R1*
Inputs	Token.
Expected results	Returns http.StatusUnauthorized.
Outputs	None.
Comments	All operations executed as expected.

Test ID	test_require_token_authentication_after_logout
Test objective	Test that the authentication after a logout is not successful.
Verified requirements	*ENF_DBB_R1*
Inputs	Token.
Expected results	Returns http.StatusUnauthorized.
Outputs	None.
Comments	All operations executed as expected.

Test ID	test_authentication_initialization
Test objective	Test that the authentication backend is properly initialized.
Verified requirements	*ENF_DBB_R1*
Inputs	None.
Expected results	Returns true
Outputs	None.
Comments	All operations executed as expected.

Test ID	test_generate_token
Test objective	Test that the token is properly generated
Verified requirements	*ENF_DBB_R1*
Inputs	None.

Expected results	Returns token.
Outputs	None.
Comments	All operations executed as expected.

Test ID	test_authenticate
Test objective	Test that the authentication with a valid user credentials is successful.
Verified requirements	*ENF_DBB_R1*
Inputs	None.
Expected results	Returns true.
Outputs	None.
Comments	All operations executed as expected.

Test ID	test_authenticate_incorrect_password
Test objective	Test that the authentication with an incorrect password is not successful.
Verified requirements	*ENF_DBB_R1*
Inputs	None.
Expected results	Returns true.
Outputs	None.
Comments	All operations executed as expected.

Test ID	test_authenticate_incorrect_username
Test objective	Test that the authentication with an incorrect username is not successful.
Verified requirements	*ENF_DBB_R1*
Inputs	None.
Expected results	Returns true.
Outputs	None.
Comments	All operations executed as expected.

Test ID	test_logout
Test objective	Test that the logout works correctly.
Verified requirements	*ENF_DBB_R1*
Inputs	None.
Expected results	Returns true.
Outputs	None.
Comments	All operations executed as expected.

Test ID	test_is_in_blacklist
Test objective	Test that the revoked token is in a blacklist.
Verified	*ENF_DBB_R1*

requirements	
Inputs	None.
Expected results	Returns true.
Outputs	None.
Comments	All operations executed as expected.

Test ID	test_is_not_in_blacklist
Test objective	Test that a token is not in a blacklist.
Verified requirements	*ENF_DBB_R1*
Inputs	None.
Expected results	Returns true.
Outputs	None.
Comments	All operations executed as expected.

Test ID	test_refresh_token
Test objective	Test that a token is properly refreshed.
Verified requirements	*ENF_DBB_R1*
Inputs	None.
Expected results	Returns true.
Outputs	None.
Comments	All operations executed as expected.

Test ID	test_backup_availability
Test objective	Test that database is properly backed up.
Verified requirements	*ENF_DBB_R2*
Inputs	None.
Expected results	Returns true.
Outputs	None.
Comments	All operations executed as expected.

The tests above only cover the DBB code. The tests for the E2EE mechanism are the same as provided by Crypton and are available at [5].

5.2.5. DoS

The DoS Detection and Mitigation mechanism provides attack detection features through the integration of the OSSEC monitoring tool. The activation of the mechanism triggers the installation of an OSSEC Server and of one or more OSSEC Agents on each machine to be monitored. The unit tests for the DoS mechanism are reported in the following tables.

Test ID	test_ossec_running
Test objective	Test if all processes related to OSSEC are running after the activation of the mechanism.
Verified requirements	*ENF_DOS_R1, ENF_DOS_R2, ENF_DOS_R3*
Inputs	None.

Expected results	List of all OSSEC processes labelled as running.
Outputs	None.
Comments	The test was executed by running the script *ossec-control status*. To issue the command connect using ssh to the machine hosting the server and navigate to the folder /opt/ossec/bin. All operations executed as expected.

Test ID	test_agents_associated
Test objective	Test if all the agents have been properly set-up and associated with the server.
Verified requirements	*ENF_DOS_R1, ENF_DOS_R2, ENF_DOS_R3*
Inputs	None .
Expected results	List of all agents reported as connected.
Outputs	None.
Comments	The test was executed by running the script *manage_agent* with option *-c*. To issue the command connect using ssh to the machine hosting the server and navigate to the folder /opt/ossec/bin. All operations executed as expected.

Test ID	Test_sql_injection_detection_mitigation
Test objective	Test if an SQL injection attack is correctly detected and mitigated.
Verified requirements	*ENF_DOS_R1, ENF_DOS_R2, ENF_DOS_R3*
Inputs	An HTTP request with an URL containing the string "select%20" issued to a node on which an OSSEC agent is running.
Expected results	The agent retrieves the information contained in the log and sends it to the server. The matching of the OSSEC rule 31103 causes the detection of the attack at the server, which classifies it as a SQL injection attack. The server generates an alert that results in the banning of the IP that generated the request. The alert is added to the server log.
Outputs	None
Comments	All operations executed as expected.

Test ID	Test_DoS_detection_mitigation
Test objective	Test if a DoS attack is correctly detected and mitigated.
Verified requirements	*ENF_DOS_R1, ENF_DOS_R2, ENF_DOS_R3*
Inputs	A high number of bogus HTTP requests are issued, in order to generate responses with 404 error code, to the nodes hosting DoS agents.
Expected results	The agent retrieves the information contained in the log and sends it to the server. The matching of the OSSEC rule 31151 causes the detection of the attack at the server, which classifies it as a DoS attack. The server generates an alert that results in the banning of the IP that generated the request. The alert is added to the server log.
Outputs	None.

Comments	All operations executed as expected.
Test ID	Test_XSS_attacks
Test objective	Test if a (Cross-site scripting) XSS attack is correctly detected and mitigated.
Verified requirements	*ENF_DOS_R1, ENF_DOS_R2, ENF_DOS_R3*
Inputs	An HTTP request whose URL contains the string: "script".
Expected results	The agent retrieves the information contained in the log and sends it to the server. The matching of the OSSEC rule 31105 causes the detection of the attack at the server, which classifies it as a XSS attack. The server generates an alert that results in the banning of the IP that generated the request. The alert is added to the server log.
Outputs	None.
Comments	All operations executed as expected.

5.2.6. AAA

The AAA mechanism offers identity management and access control functionalities through the adoption of an OAuth Server, which is the endpoint for the authentication and authorization operations carried out by users by means of registered clients. The mechanism uses an LDAP directory service to authenticate internal users, and an XACML-base authorization mechanism to apply complex authorization policies. In the following tables we present the unit tests associated to the AAA security mechanism.

Test ID	test_login_redirect
Test objective	Test if the OAuth server redirects the user to the login page when requested.
Verified requirements	*ENF_AAA_R4*
Inputs	Client ID, Redirect URI and Response Type values.
Expected results	The login page.
Outputs	None.
Comments	All operations executed as expected.

Test ID	test_login_user_ok
Test objective	Test if the login works fine.
Verified requirements	*ENF_AAA_R4*
Inputs	Username and Password of a registered account.
Expected results	The user is correctly recognized from the OAuth server through the openLDAP server, the Redirect URI is correctly called and an OAuth Code is sent to it.
Outputs	None.
Comments	All operations executed as expected.

Test ID	test_login_user_error
Test objective	Test if the login works fine.

Verified requirements	*ENF_AAA_R4*
Inputs	Username and Password of a not registered account.
Expected results	The user is correctly recognized as not registered from the OAuth server through the openLDAP server, the Redirect URI is correctly called and the error message is sent to it.
Outputs	None.
Comments	All operations executed as expected.

Test ID	test_oauth_code_exchange
Test objective	Test if the code exchange to receive an OAuth token at the OAuth server works fine.
Verified requirements	*ENF_AAA_R4*
Inputs	Client ID, Client Secret, Valid OAuth Code.
Expected results	The client is correctly recognized from the OAuth server through the DBMS query, the OAuth Code is recognized, the Redirect URI is correctly called and an OAuth Token is sent to it.
Outputs	None.
Comments	All operations executed as expected.

Test ID	test_access_profile_resource_ok
Test objective	Test if the profile resource is correctly accessed when using a valid OAuth token.
Verified requirements	*ENF_AAA_R5*
Inputs	Valid OAuth Token, Profile Resource Path.
Expected results	The OAuth token is correctly validated from the OAuth server, the associated policies are correctly evaluated and the user profile associated to it is correctly returned.
Outputs	None.
Comments	All operations executed as expected.

Test ID	test_access_profile_resource_error
Test objective	Test if the profile resource is correctly protected when using a revoked OAuth token.
Verified requirements	*ENF_AAA_R5*
Inputs	Revoked OAuth Token, Profile Resource Path.
Expected results	The OAuth token is correctly not recognized from the OAuth server and an authorization error is returned.
Outputs	None.
Comments	All operations executed as expected.

Test ID	test_access_xacml_resource
Test objective	Test if a resource configured into the OAuth server policies is

	available through the XACML Authorization Service.
Verified requirements	*ENF_AAA_R5, ENF_AAA_R6*
Inputs	Valid OAuth Token, XACML Resource Path.
Expected results	The OAuth token is correctly validated from the OAuth server, the associated XACML policies are correctly evaluated and the requested resource is correctly returned.
Outputs	None.
Comments	All operations executed as expected.

Test ID	test_logout
Test objective	Test if the logout works fine.
Verified requirements	*ENF_AAA_R7*
Inputs	Valid OAuth Token.
Expected results	The OAuth token is correctly revoked and the user is correctly logged out.
Outputs	None.
Comments	All operations executed as expected.

5.3. Summary of testing results

To provide the reader with an overview of the quality of the code for the core Enforcement components, we summarize results of the code quality analysis in Table 6.

We report data for the following SonarQube metrics (further details about the SonarQube metrics are presented in deliverable D4.5.2):

- **Lines of code**: Number of physical lines that contain at least one character which is neither a whitespace or a tabulation or part of a comment.
- **Duplications**: The density (in %) of duplicated lines.
- **Complexity**: Whenever the control flow of a function splits, the complexity counter gets incremented by one. In Java, for example, keywords incrementing the complexity, are: `if`, `for`, `while`, `case`, `catch`, `throw`, etc.
- **Technical debt ratio (TDR)**: The ratio (in %) between estimations of the effort needed to fix detected issues and the effort needed to develop the code from scratch.
- **SQUALE rating**: The SQUALE rating score depends on the TDR (equals A if TDR≤10%, B if TDR≤20%, C if TDR≤50%, D if TDR≤100%, and E if TDR>100%).
- **Issues**: Number of detected issues.
    - **Blocker**: Number of detected issues that might make the whole code unstable in production.
    - **Critical**: Number of detected issues that might lead to an unexpected behaviour in production without impacting the integrity of the whole application
    - **Major**: Number of detected issues that might have a substantial impact on productivity
    - **Minor**: Number of detected issues that might have a potential and minor impact on productivity.
- **Directory tangle index**: Level of directory interdependency (in %). Best value of 0% means that there is no cycle and worst value of 100% means that directories are really tangled.

- **Unit test coverage**: The density (in %) of unit test coverage in terms of how much of the source code has been covered by the unit tests.

SonarQube metric	Planning	Implementation	Broker	Diagnosis	RDS
Lines of code	1803	814	2651	1023	687
Duplications	0.0%	0.0%	19.2%	0.0%	0.0%
Complexity	298	155	450	181	107
Technical debt ratio	0.6%	1.5%	2.3%	1.3%	0.3%
SQUALE rating	A	A	A	A	A
Issues	28	14	120	14	8
Blocker issues	0	0	0	0	0
Critical issues	0	0	3	0	0
Major issues	0	0	28	0	0
Minor issues	26	11	87	11	6
Directory tangle index	10.0%	13.3%	11.8%	15.8%	13.3%
Unit tests coverage	74.9%	61.5%	23.0%	65.9%	77.9%

Table 6. Code quality analysis for the core Enforcement components

As seen from the table above, the code for the core Enforcement components (except the Broker component) is a high quality code with no duplications, low complexity, low directory tangle index, low technical debt ratio, only some minor issues, and very high unit test coverage. The code for the Broker component requires some effort for the overall improvement; we intend to improve the code during our exploitation activities after the finalisation of the project.

In the next two tables we present the number of requirements (#R) associated to each core Enforcement component and each security mechanisms (according to Table 5), and report about the number of requirements that are covered with unit tests (#CR) introduced in this deliverable (in Section 5).

Core Enforcement component	#R	#CR	Comment
Planning	16	15	*ENF_PLAN_R9* is not implemented.
Implementation with Broker	16	16	
Diagnosis	18	18	
RDS	15	15	

Table 7. Unit test coverage of requirements – core Enforcement components

Security mechanism	#R	#CR	Comment
WebPool	5	5	
DBB and E2EE	6	6	
SVA	4	3	*ENF_SVA_R3* is not implemented.
TLS	5	5	
AAA	9	4	*ENF_AAA_R1* has been deprecated, *ENF_AAA_R2-R3* are not implemented, *ENF_AAA_R8* is related to design, *ENF_AAA_R9* is related to configuration.
DoS	3	3	

Table 8. Unit test coverage of requirements – security mechanisms

The unit test coverage tables above show that the Enforcement prototypes implement almost all associated requirements. The final prototypes implement 90 out of 97 requirements, which is 92.8% of all elicited requirements. In particular, core Enforcement components implement 64 out of 65 requirements (~98.5%), and security mechanisms implement 26 out of 32 (~ 81.3%) associated requirements.

6. Performance and scalability analysis

In deliverable D1.5.2 we introduced the approach and the infrastructure for the performance and scalability analysis in SPECS to evaluate the capability of the system. In this section we present the process and the results of the performance and scalability evaluation of the core components of the Enforcement module.

As discussed in deliverable D1.5.2, the approach to performance testing starts with setting performance goals. We defined two performance indices, namely the *response time* (the time elapsed between the request of a service up to the production of the result) and the *throughput* (the number of requests executed per second). The process continues with definition of workloads (a set of standard user profiles modelled according to the REST API usage patterns and according to the SPECS flow; see deliverable D1.3) and the preparation of the testing environment (described in detail in deliverable D1.5.2). The last step focuses on collection of measurements (that were made with Gatling [6]) and analysis of the results.

In the remainder of this section, we present the user profiles defined for the Enforcement module and the analysis of the obtained results.

6.1. User profiles

We prepared *user profiles* for the following core components of the Enforcement module:

- Planning
- Implementation
- Diagnosis
- Remediation Decision System (RDS)

Note that not all functionalities orchestrated by the Enforcement components can be tested in terms of performance. Performance of functionalities associated to the acquisition and configuration of cloud resources (VMs) in a real-world environment depend on the performance of an external cloud provider (moreover, the acquisition and management of resources requires payment). Thus the performance of the Broker component has not been evaluated.

The four tables below summarize the user profiles used to stress test each of these components. For the resources maintained by the components and the API calls see deliverable D1.3 and for the complete behaviour of the Enforcement module see deliverable D4.3.3.

User profile	Description	Scripts
Create Supply Chain Activity	Create an activity for building supply chains.	`CreateSupplyChainActivityRamp.scala`
Create Planning Activity	Create an activity for building implementation plans.	`CreatePlanningActivityRamp.scala`

Table 9. Planning component user profiles for performance tests

User profile	Description	Scripts
Create Implementation Activity	Create an activity for executing implementation plans	`CreateImplementationActivityRamp.scala`

Table 10. Implementation component user profiles for performance tests

User profile	Description	Scripts
Create Diagnosis Activity	Create an activity for analysing notified monitoring events	`CreateDiagnosisNotificationRamp.scala`

Table 11. Diagnosis component user profiles for performance tests

User profile	Description	Scripts
Create Remediation Activity	Create an activity for building remediation plans	`CreateRemediationActivityRamp.scala`

Table 12. RDS component user profiles for performance tests

In the following we present results of performance tests on core Enforcement components according to the user profiles defined in the tables above.

6.2. Planning component

Figure 15illustrates the throughput (and the associated response time) granted by the Planning component when creating the Planning Activity object (introduced in deliverable D1.3) during the SLA negotiation phase. The performance of the Planning component while building supply chains is good considering that (i) this activity involves solving an optimization problem and (ii) that it can handle up to 45 requests per second.

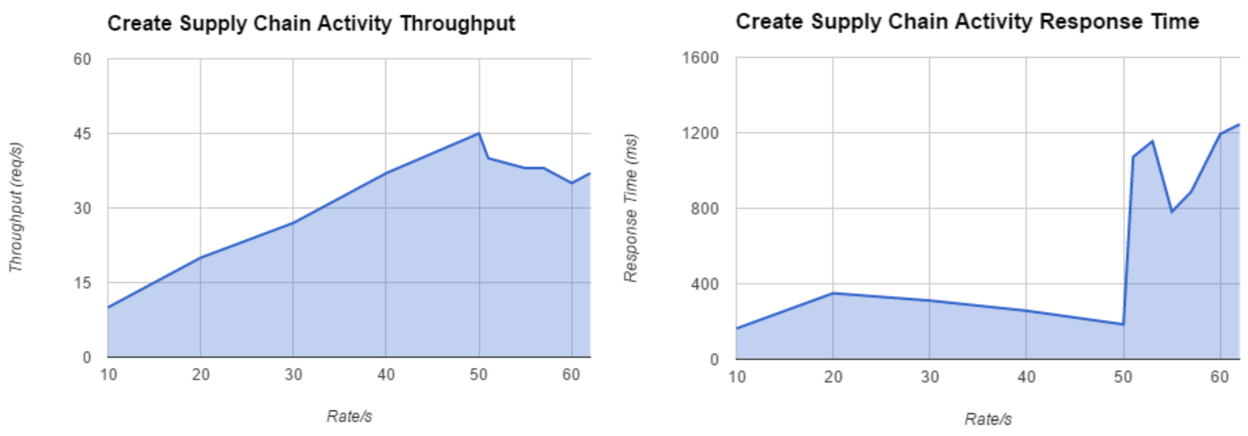


Figure 15. Throughput and response time for the Planning component – Supply chains

Figure 16 illustrates the throughput (and the associated response time) granted by the Planning component when creating the Planning Activity object (introduced in deliverable D1.3) during the SLA implementation phase. The performance of the Planning component is good considering that it can handle up to 330 requests per second. Note that the creation of the Planning Activity involves generation of an implementation plan according to the signed SLA, associated supply chain, and configuration details of the involved security mechanisms.

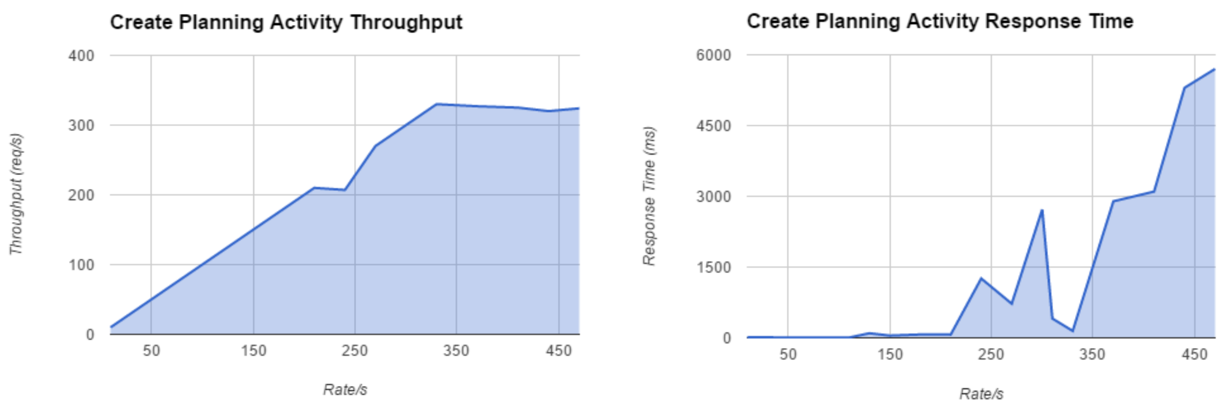


Figure 16. Throughput and response time for the Planning component - Implementation

6.3. *Implementation component*

Figure 17 illustrates the throughput (and the associated response time) granted by the Implementation component when creating the Implementation Activity object (introduced in deliverable D1.3) during the SLA implementation phase. The performance of the Implementation component while executing implementation plans is good considering that it can handle up to 130 requests per second.

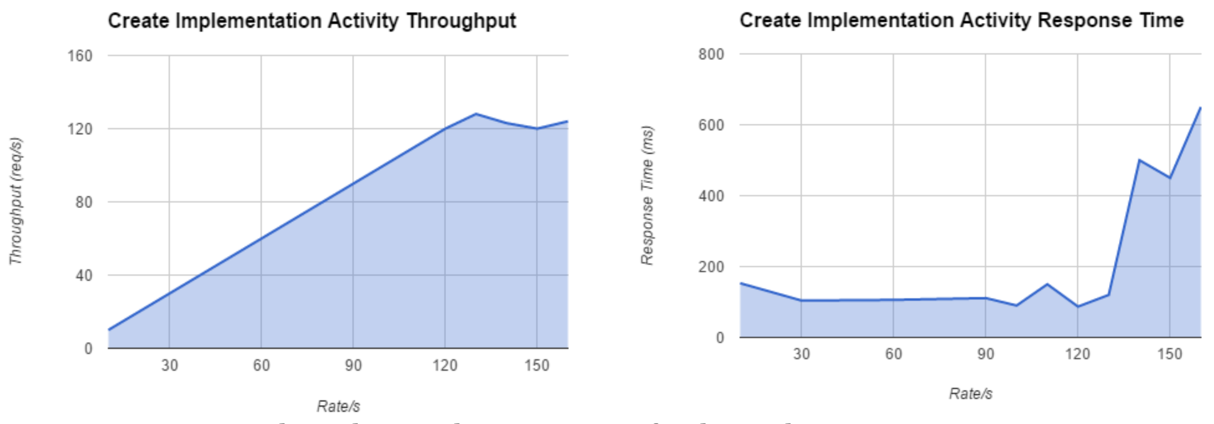


Figure 17. Throughput and response time for the Implementation component

6.4. *Diagnosis component*

Figure 18 illustrates the throughput (and the associated response time) granted by the Diagnosis component when creating the Diagnosis Notification object (introduced in deliverable D1.3) during the SLA remediation phase. The performance of the Diagnosis component while analysing detected and notified SLA alerts/violations is good considering that it can handle up to 450 requests per second.

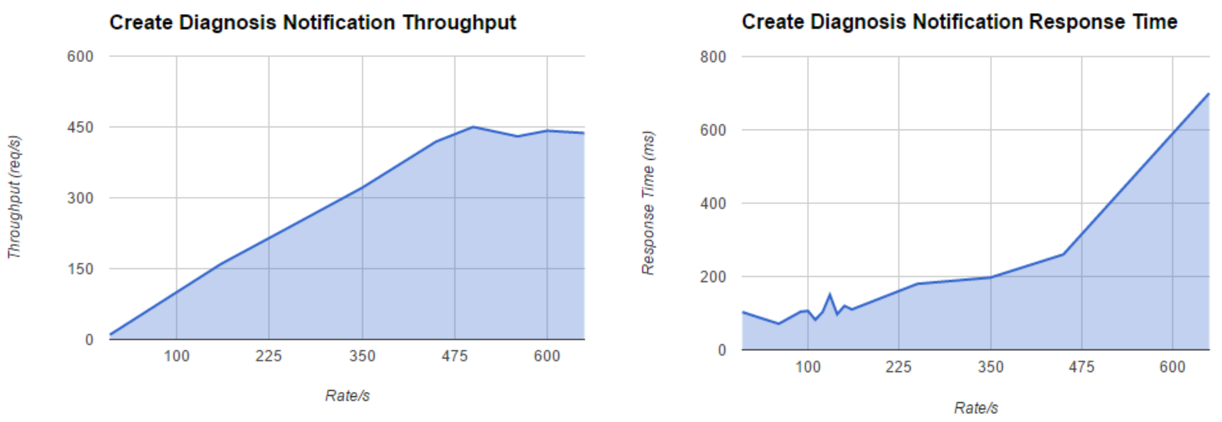


Figure 18. Throughput and response time for the Diagnosis component

6.5. RDS component

Figure 19 illustrates the throughput (and the associated response time) granted by the RDS component when creating the Remediation Activity object (introduced in deliverable D1.3) during the SLA remediation phase. The performance of the RDS component while generating remediation plans is good considering that it can handle up to 120 requests per second.

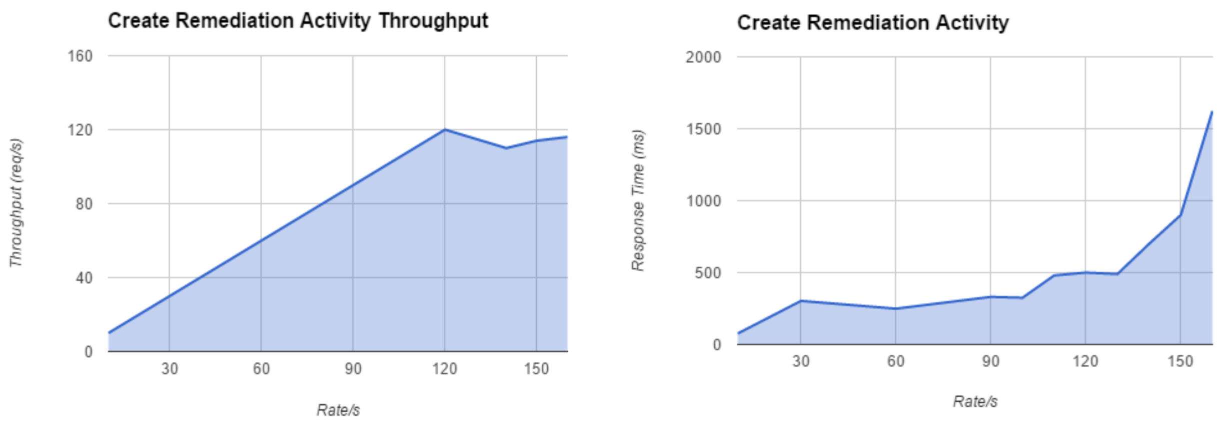


Figure 19. Throughput and response time for the RDS component

7. Conclusions

Task T4.5 focuses on the validation and testing of the Enforcement module. In particular, this document first summarizes the validation and testing techniques, and presents results of the final validation of the Enforcement module in terms of coverage of validation scenarios and requirements with the Enforcement design and developed prototypes. Then the testing activities on the Enforcement module level are presented in terms of (i) the unit tests executed for core components and security mechanisms of the Enforcement module, and (ii) a synthesis of the testing activities. Additionally, a performance and scalability of the final prototypes is analysed.

Results of the validation activities show that the Enforcement design covers the entire set of validation scenarios and all elicited requirements.

Results of the testing activities show that the final prototypes of the Enforcement module implement 92.8% (90 out of 97) of all associated requirements, and that unit tests cover in average 60.6% of the code for the core Enforcement components. Moreover, the code quality analysis reveals that there are no critical or major issues present in the code, which is evaluated with SQUALE rating score A (which implies a low technical debt). From here we can conclude that the developed software is of high quality and that it complies with the elicited requirements and design specifications.

The performance and scalability analysis confirms the statements above about the quality of the code, since all analysed components show that they can handle large amount of requests in a short amount of time. Given that the number of requests per second which can be handled by the core components is greater than 100, the solution should not introduce issues when integrated in a medium size CSP. Planning and Implementation components which are involved in the negotiation and implementation of SLAs can handle more than 100 users per second – it is highly unlikely that a medium CSP experiences such a rate of new users. While Diagnosis can be under heavier load because it has to analyse the existing SLAs, it has been shown that it can handle 450 requests per second – again even if the number of SLAs is much larger than this number, it is unlikely that a few hundred SLAs will be in an alerted or violated state in the same point in time, because one physical machine will usually serve one to five SLAs and it is unlikely that a few hundred physical machines will be attacked or broken at the same time.

Since validation scenarios were translated into integration scenarios, further details about how they were implemented are reported in both deliverables of the task T1.5 (namely D1.5.1 for the definition of integration scenarios and D1.5.2 for the executed integration tests).

8. Bibliography

[1] SPECS, "*SPECS Bitbucket account*", 2015. [Online]. Available online, https://bitbucket.org/specs-team/, last accessed in April 2016.

[2] SonarSource, "*SonarQube*", 2015. [Online]. Available online, http://www.sonarqube.org/, last accessed in April 2016.

[3] SPECS, "*Sonar Dashboard*", 2015. [Online]. Available online, https://sonar.services.ieat.ro/dashboard, last accessed in April 2016.

[4] Chef Software, "*Chef*", 2015. [Online]. Available online, https://www.chef.io/, last accessed in April 2016.

[5] SpiderOak, "*Crypton tests*", 2015. Available online, https://github.com/SpiderOak/crypton/tree/master/client/test , last accessed in April 2016.

[6] Gatling Corp, "Gatling", 2015. Available online, http://gatling.io/#/, last accessed in April 2016.

Appendix 1. Requirements associated to the Enforcement module

The following table presents a list of all requirements associated to the Enforcement module.

REQ_ID	**Requirement**	**Description**
ENF_PLAN_R1	***Get SLA to enforce***	The Planning component must be able to retrieve and parse the SLA to implement, by invoking proper functionalities offered by the Platform.
ENF_PLAN_R2	***Define security mechanisms related to SLOs***	The Planning component must be able to determine which kind of security mechanisms are to be applied, given a set of high-level SLOs contained in the SLA to implement.
ENF_PLAN_R3	***Get security components***	The Planning component must be able to retrieve the available Enforcement security components that implement the security mechanisms related to the fulfilment of the SLOs defined in the SLA to implement.
ENF_PLAN_R4	***Select best security components***	Based on the selected target service and on the negotiated SLA, the Planning component must be able to select the best available Enforcement components to invoke, among different technology stacks, in order to meet the SLOs defined in the SLA.
ENF_PLAN_R5	***Activate implementation***	The Planning component must be able to activate the selected plan, by properly invoking the Implementation component.
ENF_PLAN_R6	***Log component activation and deactivation***	The Planning component must be able to report about its activation or deactivation for accountability purposes.
ENF_PLAN_R7	***Build an implementation plan***	After the set of high-level SLOs specified in an SLA have been correlated to the appropriate security mechanisms and the best associated security components have been retrieved, the Planning component must be able to prepare an implementation plan. Building implementation plan includes deducing alert thresholds.
ENF_PLAN_R8	***Build a reaction plan***	The Planning component must be able to plan the actual activation of the redressing technique selected by the Remediation Decision System component. This may include different strategies (e.g., the definition of a chain of service invocations or the activation of a new configuration of a running service).
ENF_PLAN_R9	***Build a migration plan***	The Planning component must be able to plan the strategy to migrate from the target service currently being delivered to the new version of it, if this is a part of a redressing technique chosen by the Remediation Decision System component.
ENF_PLAN_R10	***Get monitoring systems***	The Planning component must be able to retrieve a list of available monitoring systems/agents, associated to security components that fulfil the requirements of the SLA.
ENF_PLAN_R11	***Select best monitoring systems***	The Planning component must be able to select the appropriate monitoring systems/agents that will monitor metrics/SLOs specified in the SLA.

ENF_PLAN_R12	***Validate an SLA***	The Planning component has to be able to validate an SLA by verifying that it can be enforced (*ENF_PLAN_R1*).
ENF_IMPL_R1	***Implement Plan***	The Implementation component must be able to actually realize the plan built by the Planning component, by orchestrating the acquisition of the needed resources, their configuration, and the activation of involved services.
ENF_IMPL_R2	***Acquire resources***	The Implementation component must be able to acquire all the resources needed, based on the plan built by the Planning component.
ENF_IMPL_R3	***Deploy and configure***	The Implementation component must be able to deploy and configure all the resources, based on the plan built by the Planning component.
ENF_IMPL_R4	***Start services***	The Implementation component must be able to properly start the needed services on top of the acquired resources, in order to build the plan.
ENF_IMPL_R5	***Trigger monitoring system agent activation or deactivation***	The Implementation component must be able to trigger activation/deactivation or reconfiguration of the appropriate monitoring agents by accessing the functionalities offered by the Platform.
ENF_IMPL_R6	***Log service activation***	The Implementation component must be able to log a successful activation of each security service related to a certain SLO in an SLA.
ENF_IMPL_R7	***Update SLA state***	The Implementation component must be able to update the state of an SLA after its successful implementation.
ENF_IMPL_R8	***Log component activation or deactivation***	The Implementation component must be able to report about its activation or deactivation for accountability purposes.
ENF_IMPL_R9	***Implement reaction plan***	The Implementation component must be able to apply the reaction and migration plans previously defined in the reaction plan.
ENF_IMPL_R10	***Update monitoring policy***	The Implementation component must be able to update the monitoring policy according to each signed SLA.
ENF_DIAG_R1	***Get monitoring event notification***	The Diagnosis component must be able to receive notifications from the Platform about monitoring events captured by the Monitoring module.
ENF_DIAG_R2	***Get monitoring event information***	The Diagnosis component must be able to retrieve all information, related to a monitoring event notified through the Platform, by accessing the Auditing component.
ENF_DIAG_R3	***Identify SLOs affected by a monitoring event***	The Diagnosis component must be able to identify the SLOs at risk or violated by processing a monitoring event that has been notified by the Platform.
ENF_DIAG_R4	***Update SLA state***	The Diagnosis component must be able to update the state of an SLA by accessing the proper functionalities offered by the Platform.
ENF_DIAG_R5	***Get SLAs affected by a monitoring event***	Given a monitoring event which has been notified by the Platform, the Diagnosis component must be able to retrieve all SLAs affected by such an event.
ENF_DIAG_R6	***Activate reaction***	The Diagnosis component must be able to activate the Remediation System component to react to an alert or a violation and find the best redressing techniques or remediation actions, respectively.

ENF_DIAG_R7	***Express SLA violation in terms of KPI***	The Diagnosis component must express the SLA violation detection in terms of KPI rules.
ENF_DIAG_R8	***Query metric***	The Diagnosis component can query the metric data stored inside the monitoring results repository in the Platform.
ENF_DIAG_R9	***Log component activation or deactivation***	The Diagnosis component must be able to log its activation or deactivation for accountability purposes.
ENF_DIAG_R10	***Determine effect on an SLA***	For each SLA affected by a monitoring event, the Diagnosis component must be able to determine the effect the monitoring event has on the SLA (i.e., is it alerted or violated).
ENF_DIAG_R11	***Log SLA impact***	When all SLOs affected by a monitoring event are identified, and the severity of the impact of the monitoring event has been determined, the Diagnosis component must be able to log this information.
ENF_DIAG_R12	***Classify event***	The Diagnosis component must be able to classify a monitoring event with regard to each affected SLA, based on the information provided by the Monitoring component and the affected SLOs and SLAs.
ENF_DIAG_R13	***Identify root cause***	The Diagnosis component must be able to perform a root cause analysis of each monitoring event that causes alerts or violations of one or more monitored SLAs.
ENF_DIAG_R14	***Log root cause***	The Diagnosis component must be able to log the information about the root cause of a monitoring event.
ENF_DIAG_R15	***Analyse monitoring event***	The Diagnosis component must be able to analyse each monitoring event related to an alert or a violation of one or more monitored SLAs.
ENF_DIAG_R16	***Prioritize events***	After the impact of a monitoring event on each of the affected SLAs is known and the root cause of the monitoring event is identified, the Diagnosis component must be able to create a priority queue.
ENF_DIAG_R17	***Log priority queue***	The Diagnosis component must be able to log the information about the priority queue.
ENF_DIAG_R18	***Verify SLA state***	The Diagnosis component must be able to compare the current metric/SLO data with the alert/violation thresholds specified for an alerted/violated SLA to verify if the severity of the alert/violation has changed.
ENF_REM_R1	***Trigger renegotiation***	The Remediation Decision System component will provide a mechanism to trigger renegotiation activities, by accessing the proper Platform functionalities.
ENF_REM_R2	***Log component activation or deactivation***	The Remediation Decision System component must be able to log its activation or deactivation.
ENF_REM_R3	***Get SLA state***	The Remediation Decision System component must be able to check the state of an SLA in order to react either to an alert or a violation.
ENF_REM_R4	***Update SLA state***	In the process of reacting to an event, the Remediation Decision System component must be able to update SLA's state.
ENF_REM_R5	***Get SLA***	The Remediation System Component must be able to retrieve an SLA.

ENF_REM_R6	***Get SLA impact***	The Remediation Decision System component must be able to retrieve information about the impact of a monitoring event to an affected SLA, provided by the Diagnosis component through the Auditing component.
ENF_REM_R7	***Get security components***	In the process of searching for the best actions to apply in order to mitigate the risk of having a violation or to recover from a violation, the Remediation Decision System component must be able to retrieve all relevant security components.
ENF_REM_R8	***Search for redressing techniques***	Based on the event information, associated SLAs and security mechanisms available, the Remediation Decision System component must be able to find redressing techniques to invoke in case of an alert or a violation.
ENF_REM_R9	***Notify End-user***	When End-user's decision is needed in the process of managing an alert or a violation, the Remediation Decision System component must be able to communicate the issue with the End-user through the SPECS Application.
ENF_BROKER_R1	***Enable CSP***	The SPECS Administrator must be able to configure and enable the Broker to access and use an external CSP.
ENF_BROKER_R2	***Acquire cluster***	The Broker component must be able to acquire a cluster of VMs on one of the enabled CSPs.
ENF_BROKER_R3	***Delete cluster***	The Broker component must be able to delete a cluster of VMs.
ENF_BROKER_R4	***Add user***	The Broker component must be able to add a new user to the available cluster of VMs.
ENF_BROKER_R5	***Execute script on node***	The Broker component must enable the execution of scripts on a cluster of VMs.
ENF_POOL_R1	***Diversity***	A minimum (with respect to End-user's requirements and technological constraints) *Level of Diversity* must be ensured, through the availability of a *pool* of different web server engines for hosting End-user's applications.
ENF_POOL_R2	***Load balancing***	Load balancing features should be provided, to enable the distribution of the workload generated by the End-user's web applications across multiple servers.
ENF_POOL_R3	***Survivability***	A minimum (with respect to End-user's requirements and technological constraints) *Level of Redundancy* must be ensured: in case some web containers become unavailable, the End-user's web application shall still run on the other web containers belonging to the *pool*. If all web containers in a *pool* fail, the End-user's web application will become unavailable until at least one of those web containers become healthy again.
ENF_POOL_R4	***Session sharing***	All web containers belonging to a *pool* must be able to access the shared session variables saved into a distributed caching system. This ensures session data sharing among different web servers. Also this system part exploits the advantages of replication.
ENF_POOL_R5	***Incident management***	Incident management features must be provided, enabled by the interaction with the SPECS Monitoring module and the Enforcement components, and consisting in isolating the VMs affected/targeted by some incident while ensuring business continuity to the End-user.

ENF_TLS_R1	***Translate TLS constraints***	Based on high-level constraints and requirements, the TLS component must be able to generate technology independent configuration parameters.
ENF_TLS_R2	***Verify TLS constraints***	The TLS component must be able to verify that the high-level constraints and requirements are valid and not contradictory.
ENF_TLS_R3	***Instantiate TLS configuration***	Based on technology independent configuration parameters, the TLS component must be able to generate technology dependent parameters, ready for deployment.
ENF_TLS_R4	***Deploy TLS configuration***	Taking as input the technology dependent configuration parameters, the TLS component must be able to configure a target server.
ENF_TLS_R5	***Probe TLS endpoint configuration***	The TLS component must be able to periodically check the actual exposed parameters by a TLS endpoint.
ENF_SVA_R1	***Detect vulnerabilities and misconfigurations***	Software modules/libraries that should be upgraded to resolve known issues in older versions of the monitored software, as well as misconfigurations enabling known vector attacks, must be detected.
ENF_SVA_R2	***Report vulnerabilities and misconfigurations***	Software modules/libraries that need an upgrade and any detected misconfigurations must be reported to the Platform.
ENF_SVA_R3	***Upgrade libraries and fix misconfigurations***	Upgrade or reconfiguration of the vulnerable libraries must be supported.
ENF_SVA_R4	***Visualize detected vulnerabilities and misconfigurations***	A dashboard for the visualization of detected vulnerabilities and misconfigurations as well as of the policies and rules defined by the Enforcement module must be provided.
ENF_CRYPTO_R1	***Provide client-side encryption tool as a plugin/extension***	The mechanism must provide client-side encryption in the form of a plugin/extension to download and add to the browser, in order to avoid MITM attacks. It needs to be provided as a plugin or extension (Chrome) to avoid modifications of the tool when it is being transferred to the user's machine.
ENF_CRYPTO_R2	***Configure and deploy encryption tools***	Encryption tools must be configurable. They should support asymmetric/symmetric encryption, different key management techniques, file sharing etc.
ENF_CRYPTO_R3	***Encrypt data***	The mechanism should enable encryption of files – either locally (end-to-end) or on server (depending on the security requirements).
ENF_CRYPTO_R4	***Decrypt data***	The mechanism should enable decryption of files.
ENF_AAA_R1	***Support different authentication sources***	The AAA mechanism should support different authentication sources, i.e., internal/external software components providing authentication services (e.g., LDAP server, DB, social networks).
ENF_AAA_R2	***Manage different accounts for a user***	In case of multiple supported authentication sources, the AAA mechanism must properly manage the different accounts associated to an End-user (for example, via a federation identity).
ENF_AAA_R3	***Link different identities to a single account***	The AAA mechanism must allow an End-user to create a personal account on the target system, and to associate one or more external identities to this account.

ENF_AAA_R4	***Login***	The AAA mechanism must allow End-users owning a valid account to login with such account or with any of the other identities associated with it.
ENF_AAA_R5	***Authenticate***	The AAA mechanism must apply access control policies to an End-user, whenever it invokes a service provided by the target system.
ENF_AAA_R6	***Dynamically manage access control policies***	The AAA mechanism must envision a dynamic management of access control policies carried out by an administrator.
ENF_AAA_R7	***Logout***	The AAA mechanism must provide a user with the capability of logging out of a target system.
ENF_AAA_R8	***Authentication and authorization independency***	The AAA mechanism must include authentication and authorization modules which are independent one from the other and can be configured dynamically.
ENF_AAA_R9	***Confidentiality and integrity***	The AAA mechanism itself must be protected from external compromise.
ENF_DOS_R1	***Detect DoS attack***	DoS attack detection features must be provided.
ENF_DOS_R2	***Classify detected DoS attacks***	Detected DoS attacks must be correctly classified: there are numerous DoS attack types based on consumption of computational resources, disruption of configuration, obstructing the communication media, etc.
ENF_DOS_R3	***Mitigate DoS attacks***	Mitigation functionalities must be provided. Note that mitigation depends on type of attack (e.g., filters may be used to block illegitimate traffic, using reverse proxies).
ENF_DBB_R1	***Offer secure storage***	The mechanism must be able to automatically offer secure storage in the cloud.
ENF_DBB_R2	***Assure business continuity with backup***	The mechanism must be able to guarantee business continuity with backup.
SLANEG_R30	***Remediation through SLA renegotiation***	Enforcement should consider the renegotiation of an existing SLA as a potential remedy to apply in case of alerts and violations.
SLANEG_R31	***Alerts/violations affecting multiple elements of the secure SLA hierarchy***	A detected alert/violation might affect more than one element of the SPECS security SLA hierarchy. Enforcement should consider interrelationships along SLA elements to choose the optimal redressing technique (e.g., renegotiation might help to manage multiple alerts/violations).

Appendix 2. Validation scenarios associated to the Enforcement module

This section presents the final version of all validation scenarios defined in the project (in tasks T5.1, T4.2, and T5.4). We group scenarios according to the user stories.

SST.1 Secure_Storage_Selection

General Information		
ID	*SST.1 - Secure_Storage_Selection*	
Version	*2.0*	
User Story	*STO*	*Secure Storage*
Invocation Chain	*IM1-P, IM3*	*Interaction Model 1- SPECS acting the role of Partner*
Scenario Steps		
General Description	*The End-user aims at acquiring a secure storage service from a cloud provider, which fulfils specific security-related requirements. To achieve this, the End-user negotiates the desired features with SPECS. In this validation scenario, the desired features are entirely implemented by an external CSP, while SPECS only provides to the End-user the functionalities to search, rank and select a service, which are compliant with her/his requirements. Moreover, in this scenario, the End-user signs an SLA with the selected provider.*	

Steps		
1	Phase	*SLA Negotiation*
	Actor	*End-user, SPECS application, SPECS Negotiation module*
	Preconditions	*The End-user has very basic security knowledge; she/he is able to express qualitatively requirements at a high-level of abstraction.*
	Trigger	
	Actions	*The End-user accesses the SPECS application interface. The negotiation request is forwarded to the SPECS Negotiation module, which retrieves the list of available SLA templates representing the available security services and the related security capabilities, controls and metrics. The services are returned to the End-user.*
	Postconditions	
2	Phase	*SLA Negotiation*
	Actor	*End-user, SPECS application*
	Preconditions	
	Trigger	
	Actions	*The End-user selects, among the available service offers, the desired one, i.e. the Database and Backup. The End-user specifies the desired security features by selecting the capabilities she/he is interested in and specifying the related security controls. She/he also specifies the desired metrics and sets the related SLOs.*
	Postconditions	*A supply chain compliant to the End-user requirements is built.*
3	Phase	*SLA Negotiation*
	Actor	*SPECS application, SPECS Negotiation module, SPECS Enforcement module*
	Preconditions	*A secure storage service that fulfils the specific security requirements is known to SPECS.*
	Trigger	

	Actions	*The End-user's choices are forwarded by the SPECS application to the SPECS Negotiation module, which searches for valid supply chains. In particular, the list of supply chains is built with the help of the SPECS Enforcement module. For each valid supply chain, an SLA Offer is created. The set of SLA Offers are hence ranked and returned to the SPECS application. The CSPs also add the cost of each service offer.*
	Postconditions	
4	Phase	*SLA Negotiation*
	Actor	*End-user, SPECS application, SLA Platform*
	Preconditions	*The End-user shall be logged on SPECS.*
	Trigger	
	Actions	*The SPECS application validates the SLA Offers, which are then presented to the End-user. The service offer is associated with an SLA published by an external CSP. The End-user either: 1. Accepts and signs the SLA offered by the external CSP; 2. Does not select any SLA Offer from the list and repeats the whole process from step 1 (possibly specifying a different set of requirements); 3. Does not select any SLA Offer from the list and exits the application.*
	Postconditions	*In case 1 - the signed SLA is stored by SPECS. The End-user is enabled to invoke the desired service on the external CSP with the configuration information included in the SLA.*
Graphical Model		*Not reported to avoid replication of information. See D1.3 for detailed interactions between SPECS modules.*
Coverage Information		
Users	*U_1 (CSC:User)*	
Target services	*TS_3 (Data Storage as a Service)*	
SPECS services	*See Appendix B of D5.1.2*	
SLA	*SLA_1, SLA_3, SLA_4, SLA_5*	

SST.2 Secure_Storage_Brokering_with_Client_Crypto

General Information		
ID	*SST.2 - Secure_Storage_Brokering_with_Client_Crypto*	
Version	*2.0*	
User Story	*STO*	*Secure Storage*
Invocation Chain	*IM1-CSP, IM3*	*Interaction Model 1- SPECS acting the role of CSP*
Scenario Steps		
General Description	*The End-user aims at acquiring a secure storage service from a remote cloud provider, which fulfils specific security-related requirements. Specifically, the End-user needs two capabilities, Database-as-a-Service and End-to-End Encryption, in order to detect and prove security-related violations, and to locally encrypt her/his data. To enable this service, the End-user negotiates the desired features with SPECS and signs an SLA including all service terms and guarantees. SPECS acquires the Database-as-a-Service on behalf of the End-user (registered on SPECS) and provides her/him with the End-to-End encryption security mechanism. In this scenario, SPECS also provides monitoring functionalities.*	

Steps		
1	Phase	*SLA Negotiation*
	Actor	*End-user, SPECS application, SPECS Negotiation module*
	Preconditions	*The End-user has very basic security knowledge; she/he is able to express qualitatively requirements at a high-level of abstraction.*

	Trigger	
	Actions	*The End-user accesses the SPECS application interface. The negotiation request is forwarded to the SPECS Negotiation module, which retrieves the list of available SLA templates representing the available security services and the related security capabilities, controls and metrics. The services are returned to the End-user.*
	Postconditions	
2	Phase	*SLA Negotiation*
	Actor	*End-user, SPECS application*
	Preconditions	
	Trigger	
	Actions	*The End-user selects, among the available service offers, the desired one, i.e. the Database and Backup with End-2-End Encryption. The End-user specifies the desired security features by selecting the capabilities she/he is interested in and by specifying the related security controls. She/he also specifies the desired metrics and sets the related SLOs. Precisely, the End-user specifies, between others, the need of having a client-side encryption mechanism.*
	Postconditions	*A supply chain compliant to the End-user requirements is built.*
3	Phase	*SLA Negotiation*
	Actor	*SPECS application, SPECS Negotiation module, SPECS Enforcement module*
	Preconditions	*A secure storage service, which fulfils the specific security requirements is not known to SPECS.* *An external CSP offering the Database-as-a-Service compliant with the related End-user's requirements is known to SPECS, and the end-2-end encryption is offered as SPECS security mechanism.*
	Trigger	
	Actions	*The End-user's choices are forwarded by the SPECS application to the SPECS Negotiation module, which searches for valid supply chains. In particular, the list of supply chains is built with the help of the SPECS Enforcement module. In this step, an external CSP offering the Database-as-a-Service is identified while the Encryption Package, able to support the client-side encryption, is added as a SPECS Enforcement service.* *For each valid supply chain, an SLA Offer is created. The set of SLA Offers are hence ranked and returned to the SPECS application.*
	Postconditions	
4	Phase	*SLA Negotiation*
	Actor	*End-user, SPECS application, SLA Platform*
	Preconditions	*The End-user shall be logged on SPECS.*
	Trigger	
	Actions	*The SPECS application validates the SLA Offers which are then presented to the End-user. The End-user selects the SLA Offer in which the Database-as-a-Service is offered by an external CSP while the client-side encryption is offered as a SPECS security mechanism. The selected SLA Offer is used to update and sign the SLA in the SLA Platform.*
	Postconditions	*The SLA, containing all information needed for SLA implementation, has been signed.*
5	Phase	*SLA Implementation*
	Actor	*SPECS application, SPECS Enforcement module, SLA Platform*
	Preconditions	*A valid signed SLA containing all service terms and service guarantees is available in the SLA Platform.*
	Trigger	

	Actions	*The SPECS application invokes the SPECS Enforcement module which retrieves the SLA to implement from the SLA Platform and prepares a plan to implement the signed SLA: it analyses the SLA, deduces alert thresholds, chooses the security and monitoring mechanisms to activate and determines all related software to install, as well as their configurations.*
	Postconditions	
6	Phase	*SLA Implementation*
	Actor	*SPECS Enforcement module*
	Preconditions	*A plan has been built to implement a signed SLA.*
	Trigger	
	Actions	*The SPECS Enforcement module implements the plan, by configuring and deploying all the components in order to respect the features granted in the SLA. The SPECS Enforcement module deploys and configures monitoring agents and activates all the components and services.*
	Postconditions	
7	Phase	*SLA Implementation*
	Actor	*SPECS Enforcement module, SPECS Monitoring Module*
	Preconditions	*All components and services needed for SLA implementation have been correctly configured and activated.*
	Trigger	
	Actions	*The SPECS Enforcement module configures the Monitoring module with a monitoring policy by setting proper alert/violation thresholds for specific metrics.*
	Postconditions	
8	Phase	*SLA Monitoring*
	Actor	*SPECS Monitoring module*
	Preconditions	
	Trigger	
	Actions	*SPECS keeps collecting information about the provided service and evaluates them against the current monitoring policy.*
	Postconditions	
Graphical Model		*Not reported to avoid replication of information. See D1.3 for detailed interactions between SPECS modules.*

Coverage Information	
Users	*U_1 (CSC:User)*
Target services	*TS_3 (Data Storage as a Service), TS_7 (Software as a Service)*
SPECS services	*See Appendix B of D5.1.2*
SLA	*SLA_1, SLA_3, SLA_6, SLA_7*

SST.3 Secure_Storage_with_Defined_CSP

General Information		
ID	*SST.3 - Secure_Storage_with_Defined_CSP*	
Version	*2.0*	
User Story	*STO*	*Secure Storage*
Invocation Chain	*IM1-CSP, IM3*	*Interaction Model 1- SPECS acting the role of CSP*
Scenario Steps		

General Description		*The End-user aims at storing encrypted data on a known remote cloud provider which offers a Database-as-a-service capability. The End-user asks SPECS for End-to-End Encryption capability, needed to locally encrypt her/his data. To enable this service, the End-user also gives SPECS her/his credentials on the chosen provider; SPECS securely manages these credentials and uses them to log into the chosen provider and store the End-user's data. In this scenario, SPECS also provides monitoring functionalities.*
Steps		
1	Phase	*SLA Negotiation*
	Actor	*End-user, SPECS application, SPECS Negotiation module*
	Preconditions	*The End-user has very basic security knowledge; she/he is able to express qualitatively requirements at a high-level of abstraction.*
	Trigger	
	Actions	*The End-user accesses the SPECS application interface. The negotiation request is forwarded to the SPECS Negotiation module, which retrieves the list of available SLA templates representing the available security services and the related security capabilities, controls and metrics. The services are returned to the End-user.*
	Postconditions	
2	Phase	*SLA Negotiation*
	Actor	*End-user, SPECS application*
	Preconditions	*The external CSP offering the Database-as-a-Service chosen by the End-user is known to SPECS, and the end-2-end encryption is offered as SPECS security mechanism.*
	Trigger	
	Actions	*The End-user selects, among the available service offers, the desired one, i.e. the Database and Backup with End-2-End Encryption. The End-user specifies the desired security features by selecting the capabilities she/he is interested in and by specifying the related security controls. She/he also specifies the desired metrics and sets the related SLOs. Precisely, the End-user specifies, between others, the needs of using a specific CSP as Database-as-a-Service provider and having a client-side encryption mechanism.*
	Postconditions	*A supply chain compliant to the End-user requirements is built.*
3	Phase	*SLA Negotiation*
	Actor	*SPECS application, SPECS Negotiation module, SPECS Enforcement module*
	Preconditions	
	Trigger	
	Actions	*The End-user's choices are forwarded by the SPECS application to the SPECS Negotiation module, which searches for valid supply chains. In particular, the list of supply chains is built with the help of the SPECS Enforcement module. In this step, the specific CSP defined by the End-user is identified while the Encryption Package, able to support the client-side encryption, is added as a SPECS Enforcement service. For each valid supply chain, an SLA Offer is created. The set of SLA Offers are hence ranked and returned to the SPECS application.*
	Postconditions	
4	Phase	*SLA Negotiation*
	Actor	*End-user, SPECS application, SLA Platform*
	Preconditions	*The End-user shall be logged on SPECS.*
	Trigger	

	Actions	*The SPECS application validates the SLA Offers, which are then presented to the End-user. The End-user selects the SLA Offer in which the Database-as-a-Service is offered by an external CSP while the client-side encryption is offered as a SPECS security mechanism. The selected SLA Offer is used to update and sign the SLA in the SLA Platform.*
	Postconditions	*The SLA, containing all information needed for SLA implementation, has been signed.*
5	Phase	*SLA Implementation*
	Actor	*SPECS application, SPECS Enforcement module, SLA Platform*
	Preconditions	*A valid signed SLA containing all service terms and service guarantees is available in the SLA Platform.*
	Trigger	
	Actions	*The SPECS application invokes the SPECS Enforcement module which retrieves the SLA to implement from the SLA Platform and prepares a plan to implement the signed SLA: it analyses the SLA, deduces alert thresholds, chooses the security and monitoring mechanisms to activate and determines all related software to install as well as their configurations.*
6	Phase	*SLA Implementation*
	Actor	*SPECS Enforcement module*
	Preconditions	*A plan has been built to implement a signed SLA.* *The credentials of the End-user on the external CSP have been acquired.*
	Trigger	
	Actions	*The SPECS Enforcement module implements the plan, by configuring and deploying all the components in order to respect the features granted in the SLA. The SPECS Enforcement module acquires the storage service with the credentials of the End-user on the external CSP and deploys and configures monitoring agents. The SPECS Enforcement module activates all the components and services.*
	Postconditions	
7	Phase	*SLA Implementation*
	Actor	*SPECS Enforcement module, SPECS Monitoring Module*
	Preconditions	*All components and services needed for SLA implementation have been correctly configured and activated.*
	Trigger	
	Actions	*The SPECS Enforcement module configures the Monitoring module with a monitoring policy by setting proper alert/violation thresholds for specific metrics.*
	Postconditions	
8	Phase	*SLA Monitoring*
	Actor	*SPECS Monitoring module*
	Preconditions	
	Trigger	
	Actions	*SPECS keeps collecting information about the provided service and evaluates them against the current monitoring policy.*
	Postconditions	
Graphical Model		*Not reported to avoid replication of information. See D1.3 for detailed interactions between SPECS modules.*

Coverage Information	
Users	*U_1 (CSC:User)*
Target services	*TS_3 (Data Storage as a Service), TS_7 (Software as a Service)*
SPECS services	*See Appendix B of D5.1.2*
SLA	*SLA_1, SLA_3, SLA_6, SLA_7*

SST.4 Secure_Storage_Brokering_with_Client_Crypto_Alert

General Information		
ID	*SST.4 - Secure_Storage_Brokering_with_Client_Crypto_alert*	
Version	*2.0*	
User Story	*STO*	*Secure Storage*
Invocation Chain	*IM1-CSP, IM3*	*Interaction Model 1- SPECS acting the role of CSP*
Scenario Steps		
General Description	*The End-user aims at acquiring a secure storage service from a remote cloud provider, which fulfils specific security-related requirements. Specifically, the End-user needs two capabilities, Database-as-a-Service and End-to-End Encryption, in order to detect and prove security-related violations, and to locally encrypt her/his data.* *To enable this service, the End-user negotiates the desired features with SPECS and signs an SLA including all service terms and guarantees.* *SPECS acquires the Database-as-a-Service on behalf of the End-user (registered on SPECS) and provides her/him with the End-to-End Encryption security mechanism. In this scenario, SPECS also provides monitoring functionalities.* *In this scenario, an alert is raised since the Encryption Server component is detected to be down and, since no data is sent from the End-user during the down time, no violation occurs.*	

Steps		
1	Phase	*SLA Negotiation*
	Actor	*End-user, SPECS application, SPECS Negotiation module*
	Preconditions	*The End-user has very basic security knowledge; she/he is able to express qualitatively requirements at a high-level of abstraction.*
	Trigger	
	Actions	*The End-user accesses the SPECS application interface. The negotiation request is forwarded to the SPECS Negotiation module, which retrieves the list of available SLA templates representing the available security services and the related security capabilities, controls and metrics. The services are returned to the End-user.*
	Postconditions	
2	Phase	*SLA Negotiation*
	Actor	*End-user, SPECS application*
	Preconditions	
	Trigger	
	Actions	*The End-user selects, among the available service offers, the desired one, i.e. the Database and Backup with End-2-End Encryption. The End-user specifies the desired security features by selecting the capabilities she/he is interested in and by specifying the related security controls. She/he also specifies the desired metrics and sets the related SLOs. Precisely, the End-user specifies, between others, the need of having a client-side encryption mechanism.*
	Postconditions	*A supply chain compliant to the End-user requirements is built.*
3	Phase	*SLA Negotiation*
	Actor	*SPECS application, SPECS Negotiation module, SPECS Enforcement module*
	Preconditions	*A secure storage service that fulfils the specific security requirements is not known to SPECS.* *An external CSP offering the Database-as-a-Service compliant with the related End-user's requirements is known to SPECS, and the end-2-end encryption is offered as SPECS security mechanism.*
	Trigger	

	Actions	*The End-user's choices are forwarded by the SPECS application to the SPECS Negotiation module, which searches for valid supply chains. In particular, the list of supply chains is built with the help of the SPECS Enforcement module. In this step, an external CSP offering the Database-as-a-Service is identified while the Encryption Package, able to support the client-side encryption, is added as a SPECS Enforcement service.* *For each valid supply chain, an SLA Offer is created. The set of SLA Offers are hence ranked and returned to the SPECS application.*
	Postconditions	
4	Phase	*SLA Negotiation*
	Actor	*End-user, SPECS application, SLA Platform*
	Preconditions	*The End-user shall be logged on SPECS.*
	Trigger	
	Actions	*The SPECS application validates the SLA Offers, which are then presented to the End-user. The End-user selects the SLA Offer in which the Database-as-a-Service is offered by an external CSP while the client-side encryption is offered as a SPECS security mechanism. The selected SLA Offer is used to update and sign the SLA in the SLA Platform.*
	Postconditions	*The SLA, containing all information needed for SLA implementation, has been signed.*
5	Phase	*SLA Implementation*
	Actor	*SPECS application, SPECS Enforcement module, SLA Platform*
	Preconditions	*A valid signed SLA containing all service terms and service guarantees is available in the SLA Platform*
	Trigger	
	Actions	*The SPECS application invokes the SPECS Enforcement module which retrieves the SLA to implement from the SLA Platform and prepares a plan to implement the signed SLA: it analyses the SLA, deduces alert thresholds, chooses the security and monitoring mechanisms to activate and determines all related software to install along with their configurations.*
	Postconditions	
6	Phase	*SLA Implementation*
	Actor	*SPECS Enforcement module*
	Preconditions	*A plan has been built to implement a signed SLA.*
	Trigger	
	Actions	*The SPECS Enforcement module implements the plan, by configuring and deploying all the components in order to respect the features granted in the SLA. The SPECS Enforcement module deploys and configures monitoring agents and activates all the components and services.*
	Postconditions	
7	Phase	*SLA Implementation*
	Actor	*SPECS Enforcement module, SPECS Monitoring Module*
	Preconditions	*All components and services needed for SLA implementation have been correctly configured and activated.*
	Trigger	
	Actions	*The SPECS Enforcement module configures the Monitoring module with a monitoring policy by setting proper alert/violation thresholds for specific metrics.*
	Postconditions	
8	Phase	*SLA Monitoring*
	Actor	*SPECS Monitoring module*
	Preconditions	

	Trigger	
	Actions	*SPECS keeps collecting information about the provided service and evaluates them against the current monitoring policy.*
	Postconditions	
9	Phase	*SLA Remediation*
	Actor	*SPECS Monitoring module, SPECS Enforcement module*
	Preconditions	
	Trigger	*The SPECS Monitoring module generates monitoring events due to the deviation of some metrics from set thresholds (since the the Encryption Server component is down).*
	Actions	*The SPECS Enforcement module analyses monitoring events and classifies it as an alert. The root cause of the monitoring event is determined (the Encryption server component is detected to be down, but no data has been sent from the End-user during the down time; thus no violation occurs).*
	Postconditions	*A report on the alert and on the root cause of the monitoring event is created.*
10	Phase	*SLA Remediation*
	Actor	*SPECS Enforcement module*
	Preconditions	
	Trigger	
	Actions	*The SPECS Enforcement module reacts by restarting the component before any encrypted files are sent to the server.*
	Postconditions	*The alert is solved.*
Graphical Model		*Not reported to avoid replication of information. See D1.3 for detailed interactions between SPECS modules.*

Coverage Information	
Users	*U_1 (CSC:User)*
Target services	*TS_3 (Data Storage as a Service), TS_7 (Software as a Service)*
SPECS services	*See Appendix B of D5.1.2*
SLA	*SLA_1, SLA_3, SLA_6, SLA_7, SLA_9, SLA_10, SLA_11*

SST.5 Secure_Storage_Brokering_with_Client_Crypto_Violation

General Information		
ID	*SST.5 - Secure_Storage_Brokering_with_Client_Crypto_violation*	
Version	*2.0*	
User Story	*STO*	*Secure Storage*
Invocation Chain	*IM1-CSP, IM3*	*Interaction Model 1- SPECS acting the role of CSP*
Scenario Steps		
General Description	*The End-user aims at acquiring a secure storage service from a remote cloud provider, which fulfils specific security-related requirements. Specifically, the End-user needs two capabilities, Database-as-a-Service and End-to-End Encryption, in order to detect and prove security-related violations, and to locally encrypt her/his data.* *To achieve this service, the End-user negotiates the desired features with SPECS and signs an SLA including all service terms and guarantees.* *SPECS acquires the Database-as-a-Service on behalf of the End-user (registered on SPECS) and provides her/him with the End-to-End Encryption security mechanism.* *In this scenario, SPECS also provides monitoring functionalities.* *In this scenario, a violation is detected since the Encryption Server component is detected to be down.*	

Steps		
1	Phase	*SLA Negotiation*
	Actor	*End-user, SPECS application, SPECS Negotiation module*
	Preconditions	*The End-user has very basic security knowledge; she/he is able to express qualitatively requirements at a high-level of abstraction.*
	Trigger	
	Actions	*The End-user accesses the SPECS application interface. The negotiation request is forwarded to the SPECS Negotiation module, which retrieves the list of available SLA templates representing the available security services and the related security capabilities, controls and metrics. The services are returned to the End-user.*
	Postconditions	
2	Phase	*SLA Negotiation*
	Actor	*End-user, SPECS application*
	Preconditions	
	Trigger	
	Actions	*The End-user selects, among the available service offers, the desired one, i.e. the Database and Backup with End-2-End Encryption. The End-user specifies the desired security features by selecting the capabilities she/he is interested in and by specifying the related security controls. She/he also specifies the desired metrics and sets the related SLOs. Precisely, the End-user specifies, between others, the need of having a client-side encryption mechanism.*
	Postconditions	*A supply chain compliant to the End-user requirements is built.*
3	Phase	*SLA Negotiation*
	Actor	*SPECS application, SPECS Negotiation module, SPECS Enforcement module*
	Preconditions	*A secure storage service that fulfils the specific security requirements is not known to SPECS.* *An external CSP offering the Database-as-a-Service compliant with the related End-user's requirements is known to SPECS, and the end-2-end encryption is offered as SPECS security mechanism.*
	Trigger	
	Actions	*The End-user's choices are forwarded by the SPECS application to the SPECS Negotiation module, which searches for valid supply chains. In particular, the list of supply chains is built with the help of the SPECS Enforcement module. In this step, an external CSP offering the Database-as-a-Service is identified while the Encryption Package, able to support the client-side encryption, is added as a SPECS Enforcement service.* *For each valid supply chain, an SLA Offer is created. The set of SLA Offers are hence ranked and returned to the SPECS application.*
	Postconditions	
4	Phase	*SLA Negotiation*
	Actor	*End-user, SPECS application, SLA Platform*
	Preconditions	*The End-user shall be logged on SPECS.*
	Trigger	
	Actions	*The SPECS application validates the SLA Offers which are then presented to the End-user. The End-user selects the SLA Offer in which the Database-as-a-Service is offered by an external CSP while the client-side encryption is offered as a SPECS security mechanism. The selected SLA Offer is used to update and sign the SLA in the SLA Platform.*
	Postconditions	*The SLA, containing all information needed for SLA implementation, has been signed.*
5	Phase	*SLA Implementation*
	Actor	*SPECS application, SPECS Enforcement module, SLA Platform*

	Preconditions	*A valid signed SLA containing all service terms and service guarantees is available in the SLA Platform.*
	Trigger	
	Actions	*The SPECS application invokes the SPECS Enforcement module which retrieves the SLA to implement from the SLA Platform and prepares a plan to implement the signed SLA: it analyses the SLA, deduces alert thresholds, chooses the security and monitoring mechanisms to activate and determines all related software to install along with their configurations.*
	Postconditions	
6	Phase	*SLA Implementation*
	Actor	*SPECS Enforcement module*
	Preconditions	*A plan has been built to implement a signed SLA.*
	Trigger	
	Actions	*The SPECS Enforcement module implements the plan, by configuring and deploying all the components in order to respect the features granted in the SLA. The SPECS Enforcement module deploys and configures monitoring agents and activates all the components and services.*
	Postconditions	
7	Phase	*SLA Implementation*
	Actor	*SPECS Enforcement module, SPECS Monitoring Module*
	Preconditions	*All components and services needed for SLA implementation have been correctly configured and activated.*
	Trigger	
	Actions	*The SPECS Enforcement module configures the Monitoring module with a monitoring policy by setting proper alert/violation thresholds for specific metrics.*
	Postconditions	
8	Phase	*SLA Monitoring*
	Actor	*SPECS Monitoring module*
	Preconditions	
	Trigger	
	Actions	*SPECS keeps collecting information about the provided service and evaluates them against the current monitoring policy.*
	Postconditions	
9	Phase	*SLA Remediation*
	Actor	*End-user, SPECS Monitoring module, SPECS Enforcement module*
	Preconditions	*The End-user has sent files to encrypt to the server while it is down*
	Trigger	*The SPECS Monitoring module generates monitoring events due to the deviation of some metrics from set thresholds (since the Encryption Server component is down).*
	Actions	*The SPECS Enforcement module analyses monitoring events and detects a violation. The root cause analysis of the monitoring event is determined (the Enforcement module determines that the SLA violation occurred due to the Encryption Server component being down).*
	Postconditions	*A report on the violation and on the root cause of the monitoring event is created.*
10	Phase	*SLA Remediation*
	Actor	*SPECS Enforcement module*
	Preconditions	
	Trigger	
	Actions	*SPECS notifies the violation to the End-User through the SPECS Application. The SPECS Enforcement module searches for alternatives for the End-user by building new services.*

	Postconditions	*The SLA is no more fulfilled.*
Graphical Model		*Not reported to avoid replication of information. See D1.3 for detailed interactions between SPECS modules.*
Coverage Information		
Users	*U_1 (CSC:User)*	
Target services	*TS_3 (Data Storage as a Service), TS_7 (Software as a Service)*	
SPECS services	*See Appendix B of D5.1.2*	
SLA	*SLA_1, SLA_3, SLA_6, SLA_7, SLA_9, SLA_12*	

SWC.1 Secure_Web_Container_Selection

General Information			
ID		*SWC.1 - Secure_Web_Container_Selection*	
Version		*2.0*	
User Story		*WEB*	*Secure Web Container*
Invocation Chain		*IM1-P*	*Interaction Model 1- SPECS acting the role of Partner*
Scenario Steps			
General Description		*The End-user aims at acquiring a web container service fulfilling specific security requirements (e.g., availability, resilience to attacks). To enable this service, the End-user negotiates the desired features with SPECS.* *In this validation scenario, the desired features are already provided by a CSP, and SPECS only returns to the End-user the reference to such provider.*	
Steps			
1	Phase	*SLA Negotiation*	
	Actor	*End-user, SPECS application, SPECS Negotiation module*	
	Preconditions	*The End-user is an expert customer since she/he is able to evaluate each individual metric with respect to her/him own security requirements.*	
	Trigger		
	Actions	*The End-user accesses the SPECS application interface using the expert interface, in order to enter/specify in a specific way her/his security requirements. The negotiation request is forwarded to the SPECS Negotiation module, which retrieves the list of available SLA templates representing the available security services and the related security capabilities, controls and metrics. The services are returned to the End-user.*	
	Postconditions		
2	Phase	*SLA Negotiation*	
	Actor	*End-user, SPECS application*	
	Preconditions		
	Trigger		
	Actions	*The End-user selects, among the available service offers, the desired one, i.e. the Secure Web Container. The End-user specifies the desired security features by selecting the capabilities she/he is interested in and by specifying the related security controls. She/he also specifies the desired metrics and sets the related SLOs.*	
	Postconditions	*A supply chain compliant to the End-user requirements is built.*	
3	Phase	*SLA Negotiation*	
	Actor	*SPECS application, SPECS Negotiation module, SPECS Enforcement module*	
	Preconditions	*A web container, which fulfils the specific security requirements, is offered by at least one external CSP, known to SPECS.*	
	Trigger		

	Actions	*The End-user's choices are forwarded by the SPECS application to the SPECS Negotiation module, which searches for valid supply chains. In particular, the list of supply chains is built with the help of the SPECS Enforcement module. In this step, an external CSP offering the Secure Web Container is identified. For each valid supply chain, an SLA Offer is created. The set of SLA Offers are hence ranked and returned to the SPECS application. The CSPs also add the cost of each service offer.*
	Postconditions	
4	Phase	*SLA Negotiation*
	Actor	*End-user, SPECS application, SLA Platform*
	Preconditions	*The End-user shall be logged on SPECS.*
	Trigger	
	Actions	*The SPECS application validates the SLA Offers which are then presented to the End-user. The service offer is associated with an SLA published by an external CSP. The End-user either: 1. Accepts and signs the SLA offered by the external CSP; 2. Does not select any SLA Offer from the list and repeats the whole process from step 1 (possibly specifying a different set of requirements); 3. Does not select any SLA Offer from the list and exits the application.*
	Postconditions	*In case 1 - the signed SLA is stored by SPECS. The End-user is enabled to invoke the desired service on the external CSP with the configuration information included in the SLA.*
Graphical Model		*Not reported to avoid replication of information. See D1.3 for detailed interactions between SPECS modules.*
Coverage Information		
Users		*U_1 (CSC:User)*
Target services		*TS_4 (Infrastructure as a Service)*
SPECS services		*See Appendix B of D5.1.2*
SLA		*SLA_1, SLA_3. SLA_4, SLA_5*

SWC.2 Secure_Web_Container_Brokering

General Information			
ID		*SWC.2 - Secure_Web_Container_Brokering*	
Version		*2.0*	
User Story		*WEB*	*Secure Web Container*
Invocation Chain		*IM1-CSP*	*Interaction Model 1- SPECS acting the role of CSP*
Scenario Steps			
General Description		*The End-user aims at acquiring a web container service fulfilling specific security requirements (e.g., availability, resilience to attacks). To enable this service, the End-user negotiates the desired security features with SPECS. In this validation scenario, the desired features are already provided by a CSP, but SPECS acts as a broker by acquiring the resources on behalf of the End-user (registered on SPECS) and by setting up some monitoring functionalities in order to monitor the fulfilment of the SLA.*	
Steps			
1	Phase	*SLA Negotiation*	
	Actor	*End-user, SPECS application, SPECS Negotiation module*	
	Preconditions	*The End-user has very basic security knowledge; she/he is able to express qualitatively requirements at a high-level of abstraction.*	
	Trigger		

	Actions	*The End-user accesses the SPECS application interface. The negotiation request is forwarded to the SPECS Negotiation module, which retrieves the list of available SLA templates representing the available security services and the related security capabilities, controls and metrics. The services are returned to the End-user.*
	Postconditions	
2	Phase	*SLA Negotiation*
	Actor	*End-user, SPECS application*
	Preconditions	
	Trigger	
	Actions	*The End-user selects, among the available service offers, the desired one, i.e. the Secure Web Container. The End-user specifies the desired security features by selecting the capabilities she/he is interested in and by specifying the related security controls- She/he also specifies the desired metrics and sets the related SLOs.* *The End-user accesses the Security Metric Catalogue in order to have additional and detailed information about the specific chosen metrics.*
	Postconditions	*A supply chain compliant to the End-user requirements is built.*
3	Phase	*SLA Negotiation*
	Actor	*SPECS application, SPECS Negotiation module, SPECS Enforcement module*
	Preconditions	*A web container, which fulfils the specific security requirements, is offered by at least one external CSP, known to SPECS.*
	Trigger	
	Actions	*The End-user's choices are forwarded by the SPECS application to the SPECS Negotiation module, which searches for valid supply chains. In particular, the list of supply chains is built with the help of the SPECS Enforcement module. In this step, an external CSP offering the Secure Web Container is identified.* *For each valid supply chain, an SLA Offer is created. The set of SLA Offers are hence ranked and returned to the SPECS application.*
	Postconditions	
4	Phase	*SLA Negotiation*
	Actor	*End-user, SPECS application, SLA Platform*
	Preconditions	*The End-user shall be logged on SPECS.*
	Trigger	
	Actions	*The SPECS application validates the SLA Offers, which are then presented to the End-user. The End-user selects the SLA Offer in which the Secure Web Container is offered by an external CSP. The selected SLA Offer is used to update and sign the SLA in the SLA Platform.*
	Postconditions	*The SLA, containing all information needed for SLA implementation, has been signed.*
5	Phase	*SLA Implementation*
	Actor	*SPECS application, SPECS Enforcement module, SLA Platform*
	Preconditions	*A valid signed SLA containing all service terms and service guarantees is available in the SLA Platform.*
	Trigger	
	Actions	*The SPECS application invokes the SPECS Enforcement module which retrieves the SLA to implement from the SLA Platform and prepares a plan to implement the signed SLA: it analyses the SLA, deduces alert thresholds, chooses the security and monitoring mechanisms to activate and determines all related software to install along with their configurations.*
	Postconditions	
6	Phase	*SLA Implementation*
	Actor	*SPECS Enforcement module*

	Preconditions	*A plan has been built to implement a signed SLA.*
	Trigger	
	Actions	*The SPECS Enforcement module implements the plan, by configuring and deploying all the components in order to respect the features granted in the SLA. The SPECS Enforcement module deploys and configures monitoring agents and activates all the components and services.*
	Postconditions	
7	Phase	*SLA Implementation*
	Actor	*SPECS Enforcement module, SPECS Monitoring Module*
	Preconditions	*All components and services needed for SLA implementation have been correctly configured and activated.*
	Trigger	
	Actions	*The SPECS Enforcement module configures the Monitoring module with a monitoring policy by setting proper alert/violation thresholds for specific metrics.*
	Postconditions	
8	Phase	*SLA Monitoring*
	Actor	*SPECS Monitoring module*
	Preconditions	
	Trigger	
	Actions	*SPECS keeps collecting information about the provided service and evaluates them against the current monitoring policy.*
	Postconditions	
Graphical Model		*Not reported to avoid replication of information. See D1.3 for detailed interactions between SPECS modules.*
Coverage Information		
Users	*U_1 (CSC:User)*	
Target services	*TS_4 (Infrastructure as a Service)*	
SPECS services	*See Appendix B of D5.1.2*	
SLA	*SLA_1, SLA_3, SLA_6, SLA_7*	

SWC.3 Secure_Web_Container_TLS_Enhanced

General Information		
ID	*SWC.3 - Secure_Web_Container_TLS_enhanced*	
Version	*2.0*	
User Story	*WEB*	*Secure Web Container*
Invocation Chain	*IM1-CSP*	*Interaction Model 1- SPECS acting the role of CSP*
Scenario Steps		
General Description	*The End-user aims at acquiring a web container service fulfilling specific security requirements. In particular, the End-user requires the adoption of the TLS protocol to protect the network communications, and of the DoS detection and mitigation features. To enable this service, the End-user negotiates the desired features with SPECS. In this validation scenario, a bare web container is offered by a CSP, while the TLS protocol and the DoS detection and mitigation features are provided by SPECS through the activation of proper mechanisms. SPECS acquires the resources on behalf of the End-user (registered on SPECS), deploys and activates the TLS and DoS related mechanisms, and sets up elated monitoring functionalities. In this scenario, an alert regarding a DoS attack is generated, and SPECS reacts by activating proper mitigation strategies. The scenario ends without any other alert.*	
Steps		

1	Phase	*SLA Negotiation*
	Actor	*End-user, SPECS application, SPECS Negotiation module*
	Preconditions	*The End-user has very basic security knowledge; she/he is able to express qualitatively requirements at a high-level of abstraction.*
	Trigger	
	Actions	*The End-user accesses the SPECS application interface. The negotiation request is forwarded to the SPECS Negotiation module, which retrieves the list of available SLA templates representing the available security services and the related security capabilities, controls and metrics. The services are returned to the End-user.*
	Postconditions	
2	Phase	*SLA Negotiation*
	Actor	*End-user, SPECS application*
	Preconditions	
	Trigger	
	Actions	*The End-user selects, among the available service offers, the desired one, i.e. the Secure Web Container. The End-user specifies the desired security features by selecting the capabilities she/he is interested in and by specifying the related security controls. She/he also specifies the desired metrics and sets the related SLOs.*
	Postconditions	*A supply chain compliant to the End-user requirements is built.*
3	Phase	*SLA Negotiation*
	Actor	*SPECS application, SPECS Negotiation module, SPECS Enforcement module*
	Preconditions	*A web container, which fulfils the specific security requirements, is not known to SPECS.* *An Infrastructure-as-a-Service provider that offers plain VMs is known to SPECS, and the TLS and DoS detection and mitigation tools are offered as SPECS security mechanisms.*
	Trigger	
	Actions	*The End-user's choices are forwarded by the SPECS application to the SPECS Negotiation module, which searches for valid supply chains. In particular, the list of supply chains is built with the help of the SPECS Enforcement module. In this step, an external CSP offering the Secure Web Container is identified. TLS, DoS detection and DoS mitigation components are identified among SPECS Enforcement security components.* *For each valid supply chain, an SLA Offer is created. The set of SLA Offers are hence ranked and returned to the SPECS application.*
	Postconditions	
4	Phase	*SLA Negotiation*
	Actor	*End-user, SPECS application, SLA Platform*
	Preconditions	*The End-user shall be logged on SPECS.*
	Trigger	
	Actions	*The SPECS application validates the SLA Offers which are then presented to the End-user. The End-user selects the SLA Offer in which the Secure Web Container is offered by an external CSP, while the TLS, DoS detection and DoS mitigation are offered as SPECS security mechanisms. The selected SLA Offer is used to update and sign the SLA in the SLA Platform.*
	Postconditions	*The SLA, containing all information needed for SLA implementation, has been signed.*
5	Phase	*SLA Implementation*
	Actor	*SPECS application, SPECS Enforcement module, SLA Platform*
	Preconditions	*A valid signed SLA containing all service terms and service guarantees is available in the SLA Platform.*
	Trigger	

	Actions	*The SPECS application invokes the SPECS Enforcement module which retrieves the SLA to implement from the SLA Platform and prepares a plan to implement the signed SLA: it analyses the SLA, deduces alert thresholds, chooses the security and monitoring mechanisms to activate and determines all related software to install along with their configurations.*
	Postconditions	
6	Phase	*SLA Implementation*
	Actor	*SPECS Enforcement module*
	Preconditions	*A plan has been built to implement a signed SLA.*
	Trigger	
	Actions	*The SPECS Enforcement module implements the plan, by configuring and deploying all the components in order to respect the features granted in the SLA. The SPECS Enforcement module deploys and configures monitoring agents and activates all the components and services.*
	Postconditions	
7	Phase	*SLA Implementation*
	Actor	*SPECS Enforcement module, SPECS Monitoring Module*
	Preconditions	*All components and services needed for SLA implementation have been correctly configured and activated.*
	Trigger	
	Actions	*The SPECS Enforcement module configures the Monitoring module with a monitoring policy by setting proper alert/violation thresholds for specific metrics.*
	Postconditions	
8	Phase	*SLA Monitoring*
	Actor	*SPECS Monitoring module*
	Preconditions	
	Trigger	
	Actions	*SPECS keeps collecting information about the provided service and evaluates them against the current monitoring policy.*
	Postconditions	
9	Phase	*SLA Remediation*
	Actor	*SPECS Monitoring module, SPECS Enforcement module*
	Preconditions	
	Trigger	*The SPECS Monitoring module generates monitoring events related to detection of DoS attack by the DoS Monitoring component.*
	Actions	*The SPECS Enforcement module analyses monitoring events and, relying upon the attack classification functionalities provided by the SPECS DoS Mitigation component, classifies it as an alert.*
	Postconditions	
10	Phase	*SLA Remediation*
	Actor	*SPECS Enforcement module*
	Preconditions	*Some mitigation strategies are available.*
	Trigger	*An alert has been detected.*
	Actions	*The SPECS Enforcement module reacts by activating proper mitigation strategies, defined by the SPECS DoS Mitigation component.*
	Postconditions	*The alert is solved and the SLA is completed since neither alerts nor violations occur.*
Graphical Model		*Not reported to avoid replication of information. See D1.3 for detailed interactions between SPECS modules.*
<u>Coverage Information</u>		

Users	*U_1 (CSC:User)*
Target services	*TS_4 (Infrastructure as a Service)*
SPECS services	*See Appendix B of D5.1.2*
SLA	*SLA_1, SLA_3, SLA_6, SLA_7, SLA_9, SLA_10, SLA_11, SLA_8*

SWC.4 Secure_Web_Container_TLS_Enhanced_Alert

General Information			
ID		*SWC.4 - Secure_Web_Container_SVA_enhanced_alert*	
Version		*2.0*	
User Story		*WEB*	*Secure Web Container*
Invocation Chain		*IM1-CSP*	*Interaction Model 1- SPECS acting the role of CSP*
Scenario Steps			
General Description		*The End-user aims at acquiring a web container service fulfilling specific security requirements. In particular, the End-user requires the adoption of a Software Vulnerability Assessment (SVA) tool to protect the web container environment. To enable this service, the End-User negotiates the desired features with SPECS. In this validation scenario, the bare web container is offered by a CSP, while the SVA tools are provided by SPECS. SPECS acquires the resources on behalf of the End-user (registered on SPECS), deploys and activates the needed SVA agents and sets-up related monitoring functionalities. In this scenario, an alert is generated due to the existence of some critical vulnerability in the installed software. SPECS reacts by updating the software version to remove the vulnerability. The scenario ends without any other alert.*	
Steps			
1	Phase	*SLA Negotiation*	
	Actor	*End-user, SPECS application, SPECS Negotiation module*	
	Preconditions	*The End-user has very basic security knowledge; she/he is able to express qualitatively requirements at a high-level of abstraction.*	
	Trigger		
	Actions	*The End-user accesses the SPECS application interface. The negotiation request is forwarded to the SPECS Negotiation module, which retrieves the list of available SLA templates representing the available security services and the related security capabilities, controls and metrics. The services are returned to the End-user.*	
	Postconditions		
2	Phase	*SLA Negotiation*	
	Actor	*End-user, SPECS application*	
	Preconditions		
	Trigger		
	Actions	*The End-user selects, among the available service offers, the desired one, i.e. the Secure Web Container. The End-user specifies the desired security features by selecting the capabilities she/he is interested in and by specifying the related security controls. She/he also specifies the desired metrics and sets the related SLOs.*	
	Postconditions	*A supply chain compliant to the End-user requirements is built.*	
3	Phase	*SLA Negotiation*	
	Actor	*SPECS application, SPECS Negotiation module, SPECS Enforcement module*	
	Preconditions	*A web container, which fulfils the specific security requirements, is not known to SPECS. An Infrastructure-as-a-Service provider that offers plain VMs is known to SPECS, and SVA agents are offered as SPECS security mechanisms.*	
	Trigger		

	Actions	*The End-user's choices are forwarded by the SPECS application to the SPECS Negotiation module, which searches for valid supply chains. In particular, the list of supply chains is built with the help of the SPECS Enforcement module. In this step, an external CSP offering the Secure Web Container is identified. SVA agents are identified among SPECS Enforcement security components. For each valid supply chain, an SLA Offer is created. The set of SLA Offers are hence ranked and returned to the SPECS application.*
	Postconditions	
4	Phase	*SLA Negotiation*
	Actor	*End-user, SPECS application, SLA Platform*
	Preconditions	*The End-user shall be logged on SPECS.*
	Trigger	
	Actions	*The SPECS application validates the SLA Offers, which are then presented to the End-user. The End-user selects the SLA Offer in which the Secure Web Container is offered by an external CSP, while the SVA agents are offered as SPECS security mechanisms. The selected SLA Offer is used to update and sign the SLA in the SLA Platform.*
	Postconditions	*The SLA, containing all information needed for SLA implementation, has been signed.*
5	Phase	*SLA Implementation*
	Actor	*SPECS application, SPECS Enforcement module, SLA Platform*
	Preconditions	*A valid signed SLA containing all service terms and service guarantees is available in the SLA Platform.*
	Trigger	
	Actions	*The SPECS application invokes the SPECS Enforcement module which retrieves the SLA to implement from the SLA Platform and prepares a plan to implement the signed SLA: it analyses the SLA, deduces alert thresholds, chooses the security and monitoring mechanisms to activate and determines all related software to install along with their configurations.*
	Postconditions	
6	Phase	*SLA Implementation*
	Actor	*SPECS Enforcement module*
	Preconditions	*A plan has been built to implement a signed SLA.*
	Trigger	
	Actions	*The SPECS Enforcement module implements the plan, by configuring and deploying all the components in order to respect the features granted in the SLA (including the installation of SVA agents on the plain VM). The SPECS Enforcement module deploys and configures monitoring agents and activates all the components and services.*
	Postconditions	
7	Phase	*SLA Implementation*
	Actor	*SPECS Enforcement module, SPECS Monitoring Module*
	Preconditions	*All components and services needed for SLA implementation have been correctly configured and activated.*
	Trigger	
	Actions	*The SPECS Enforcement module configures the Monitoring module with a monitoring policy by setting proper alert/violation thresholds for specific metrics.*
	Postconditions	
8	Phase	*SLA Monitoring*
	Actor	*SPECS Monitoring module*
	Preconditions	

	Trigger	
	Actions	*SPECS keeps collecting information about the provided service and evaluates them against the current monitoring policy.*
	Postconditions	
9	Phase	*SLA Remediation*
	Actor	*SPECS Monitoring module, SPECS Enforcement module*
	Preconditions	
	Trigger	*The SPECS Monitoring module generates monitoring events related to the deviation of some metrics from set thresholds (e.g., number of exposed vulnerabilities).*
	Actions	*The SPECS Enforcement module makes an analysis of monitoring events and classifies them as an alert.*
	Postconditions	
10	Phase	*SLA Remediation*
	Actor	*SPECS Enforcement module*
	Preconditions	*The new version of the vulnerable software is available.*
	Trigger	*An alert regarding a vulnerability threat has been detected*
	Actions	*The SPECS Enforcement module reacts by activating the available redressing technique (it checks the presence of new versions, and updates the vulnerable software).*
	Postconditions	*The alert is solved.*
Graphical Model		*Not reported to avoid replication of information. See D1.3 for detailed interactions between SPECS modules.*

Coverage Information	
Users	*U_1 (CSC:User)*
Target services	*TS_4 (Infrastructure as a Service)*
SPECS services	*See Appendix B of D5.1.2*
SLA	*SLA_1, SLA_3, SLA_6, SLA_7, SLA_9, SLA_10, SLA_11*

SWC.5 Secure_Web_Container_TLS_SVA_Enhanced_Violation

General Information		
ID	*SWC.5 - Secure_Web_Container_TLS_SVA_enhanced_violation*	
Version	*2.0*	
User Story	*WEB*	*Secure Web Container*
Invocation Chain	*IM1-CSP*	*Interaction Model 1- SPECS acting the role of CSP*
Scenario Steps		

General Description		*The End-user aims at acquiring a web container from an Infrastructure-as-a-Service CSP, represented by a VM hosting the web server, which fulfils specific security-related requirements. In particular, the End-user requires the adoption of Software Vulnerability Assessment (SVA) tools to protect the Web Server environment. To enable this service, the End-user negotiates the desired features with the SPECS.* *In this validation scenario, the VM (without SVA) is provided by an Infrastructure-as-a-Service CSP, while the SVA agents are installed by SPECS. SPECS acquires the resources on behalf of the End-user (registered on SPECS), it adds the SVA agents, and sets up some monitoring functionalities in order to detect the presence of exposed vulnerabilities.* *This scenario includes the raising of an alert regarding a vulnerability assessment report, which corresponds to a violation of the agreed SLA. SPECS reacts by renegotiating the SLA; the End-user asks for the adoption of the TLS protocol to protect the Web Server communications. The renegotiated SLA is hence signed and properly monitored by SPECS.*
Steps		
1	Phase	*SLA Negotiation*
	Actor	*End-user, SPECS application, SPECS Negotiation module*
	Preconditions	*The End-user has very basic security knowledge; she/he is able to express qualitatively requirements at a high-level of abstraction.*
	Trigger	
	Actions	*The End-user accesses the SPECS application interface. The negotiation request is forwarded to the SPECS Negotiation module, which retrieves the list of available SLA templates representing the available security services and the related security capabilities, controls and metrics. The services are returned to the End-user.*
	Postconditions	
2	Phase	*SLA Negotiation*
	Actor	*End-user, SPECS application*
	Preconditions	
	Trigger	
	Actions	*The End-user selects, among the available service offers, the desired one, i.e. the Secure Web Container. The End-user specifies the desired security features by selecting the capabilities she/he is interested in and by specifying the related security controls. She/he also specifies the desired metrics and sets the related SLOs.*
	Postconditions	*A supply chain compliant to the End-user requirements is built.*
3	Phase	*SLA Negotiation*
	Actor	*SPECS application, SPECS Negotiation module, SPECS Enforcement module*
	Preconditions	*A web container, which fulfils the specific security requirements, is not known to SPECS.* *An Infrastructure-as-a-Service provider that offers plain VMs is known to SPECS, and SVA agents are offered as SPECS security mechanisms.*
	Trigger	
	Actions	*The End-user's choices are forwarded by the SPECS application to the SPECS Negotiation module, which searches for valid supply chains. In particular, the list of supply chains is built with the help of the SPECS Enforcement module. In this step, an external CSP offering the Secure Web Container is identified. SVA agents are identified among SPECS Enforcement security components.* *For each valid supply chain, an SLA Offer is created. The set of SLA Offers are hence ranked and returned to the SPECS application.*
	Postconditions	
4	Phase	*SLA Negotiation*
	Actor	*End-user, SPECS application, SLA Platform*

	Preconditions	*The End-user shall be logged on SPECS.*
	Trigger	
	Actions	*The SPECS application validates the SLA Offer,s which are then presented to the End-user. The End-user selects the SLA Offer in which the Secure Web Container is offered by an external CSP while the SVA agents are offered as SPECS security mechanisms. The selected SLA Offer is used to update and sign the SLA in the SLA Platform.*
	Postconditions	*The SLA, containing all information needed for SLA implementation, has been signed.*
5	Phase	*SLA Implementation*
	Actor	*SPECS application, SPECS Enforcement module, SLA Platform*
	Preconditions	*A valid signed SLA containing all service terms and service guarantees is available in the SLA Platform.*
	Trigger	
	Actions	*The SPECS application invokes the SPECS Enforcement module which retrieves the SLA to implement from the SLA Platform and prepares a plan to implement the signed SLA: it analyses the SLA, deduces alert thresholds, chooses the security and monitoring mechanisms to activate and determines all related software to install along with their configurations.*
	Postconditions	
6	Phase	*SLA Implementation*
	Actor	*SPECS Enforcement module*
	Preconditions	*A plan has been built to implement a signed SLA.*
	Trigger	
	Actions	*The SPECS Enforcement module implements the plan, by configuring and deploying all the components in order to respect the features granted in the SLA (including the installation of SVA agents on the plain VM). The SPECS Enforcement module deploys and configures monitoring agents and activates all the components and services.*
	Postconditions	
7	Phase	*SLA Implementation*
	Actor	*SPECS Enforcement module, SPECS Monitoring Module*
	Preconditions	*All components and services needed for SLA implementation have been correctly configured and activated.*
	Trigger	
	Actions	*The SPECS Enforcement module configures the Monitoring module with a monitoring policy by setting proper alert/violation thresholds for specific metrics.*
	Postconditions	
8	Phase	*SLA Monitoring*
	Actor	*SPECS Monitoring module*
	Preconditions	
	Trigger	
	Actions	*SPECS keeps collecting information about the provided service and evaluates them against the current monitoring policy.*
	Postconditions	
9	Phase	*SLA Remediation*
	Actor	*SPECS Monitoring module, SPECS Enforcement module*
	Preconditions	
	Trigger	*The SPECS Monitoring module generates monitoring events related to the deviation of some metrics from set thresholds (e.g., number of exposed vulnerabilities).*

	Actions	*The SPECS Enforcement module makes an analysis of monitoring events and classifies them as a violation.*
	Postconditions	
10	Phase	*SLA Remediation*
	Actor	*SPECS Application, SPECS Enforcement module*
	Preconditions	*No remedies can be applied by SPECS; renegotiation is needed.*
	Trigger	*A violation of the signed SLA has been detected.*
	Actions	*SPECS notifies the violation to the End-User through the SPECS Application. The SPECS Enforcement module searches for alternatives for the End-user by building new services.*
	Postconditions	*The SLA is no more fulfilled*
11	Phase	*Renegotiation*
	Actor	*End-user, SPECS Application, SPECS Negotiation module*
	Preconditions	
	Trigger	
	Actions	*The End-user asks for the adoption of Transport Layer Security (TLS) protocol to protect the Web Server communications. The renegotiation follows the same activities described in steps 1 to 4.*
	Postconditions	*The renegotiated SLA is signed.*
12	Phase	*SLA Implementation*
	Actor	*SPECS Enforcement module, SPECS Monitoring Module*
	Preconditions	*A valid signed SLA containing all service terms and service guarantees is available.*
	Trigger	
	Actions	*The implementation of the SLA follows the same activities described in steps 5 to 7.*
	Postconditions	
13	Phase	*SLA Monitoring*
	Actor	*SPECS Monitoring module*
	Preconditions	*The monitoring policy has been updated to include thresholds related to the SLA.*
	Trigger	
	Actions	*SPECS keeps collecting information about the provided service and evaluates them against the current monitoring policy.*
	Postconditions	
Graphical Model		*Not reported to avoid replication of information. See D1.3 for detailed interactions between SPECS modules.*

Coverage Information	
Users	*U_1 (CSC:User)*
Target services	*TS_4 (Infrastructure as a Service)*
SPECS services	*See Appendix B of D5.1.2*
SLA	SLA_1, SLA_3, SLA_6, SLA_7, SLA_13, SLA_14, SLA_17, SLA_19

SWC.6 Secure_Web_Container_TLS_Multitenancy

General Information		
ID	*SWC.6 - Secure_Web_Container_TLS_multitenancy*	
Version	*2.0*	
User Story	*WEB*	*Secure Web Container*
Invocation Chain	*IM1-CSP*	*Interaction Model 1- SPECS acting the role of CSP*
Scenario Steps		

General Description		*Two End-users aim at acquiring different web container services fulfilling specific security requirements. In addition, both End-users require the adoption of the TLS protocol to protect the communications of Web Servers. To enable this service, the first End-user negotiates the desired features with SPECS. The VM (without TLS) is provided by a CSP, while the TLS protocol is added by SPECS by setting up the appropriate resources (e.g., reverse proxy). The second End-user negotiates the desired features with SPECS. A different VM (without TLS) is provided by a CSP (either the same or a different one) while the TLS protocol is added by SPECS reusing, for scalability purposes, the same resources configured for the first End-user. This validation scenario considers the multi-tenancy in the usage of shared resources between End-users.*
Steps		
1	Phase	*SLA Negotiation*
	Actor	*End-user (first), SPECS application, SPECS Negotiation module*
	Preconditions	*The End-user has very basic security knowledge; she/he is able to express qualitatively requirements at a high-level of abstraction.*
	Trigger	
	Actions	*The first End-user accesses the SPECS application interface. The negotiation request is forwarded to the SPECS Negotiation module, which retrieves the list of available SLA templates representing the available security services and the related security capabilities, controls and metrics. The services are returned to the End-user.*
	Postconditions	
2	Phase	*SLA Negotiation*
	Actor	*End-user (first), SPECS application*
	Preconditions	
	Trigger	
	Actions	*The first End-user selects, among the available service offers, the desired one, i.e. the Secure Web Container. The End-user specifies the desired security features by selecting the capabilities she/he is interested in and by specifying the related security controls. She/he also specifies the desired metrics and sets the related SLOs.*
	Postconditions	*A supply chain compliant to the End-user requirements is built.*
3	Phase	*SLA Negotiation*
	Actor	*SPECS application, SPECS Negotiation module, SPECS Enforcement module*
	Preconditions	*A web container, which fulfils the specific security requirements, is not known to SPECS. An Infrastructure-as-a-Service provider that offers plain VMs is known to SPECS, and the TLS and DoS detection and mitigation tools are offered as SPECS security mechanisms.*
	Trigger	
	Actions	*The End-user's choices are forwarded by the SPECS application to the SPECS Negotiation module, which searches for valid supply chains. In particular, the list of supply chains is built with the help of the SPECS Enforcement module. In this step, an external CSP offering the Secure Web Container is identified. TLS, DoS detection and DoS mitigation components are identified among SPECS Enforcement security components. For each valid supply chain, an SLA Offer is created. The set of SLA Offers are hence ranked and returned to the SPECS application.*
	Postconditions	
4	Phase	*SLA Negotiation*
	Actor	*End-user (first), SPECS application, SLA Platform*
	Preconditions	*The End-user shall be logged on SPECS.*

	Trigger	
	Actions	*The SPECS application validates the SLA Offers, which are then presented to the End-user. The End-user selects the SLA Offer in which the Secure Web Container is offered by an external CSP while the TLS, DoS detection and DoS mitigation are offered as SPECS security mechanisms. The selected SLA Offer is used to update and sign the SLA in the SLA Platform.*
	Postconditions	*The SLA, containing all information needed for SLA implementation, has been signed.*
5	Phase	*SLA Implementation*
	Actor	*SPECS application, SPECS Enforcement module, SLA Platform*
	Preconditions	*A valid signed SLA containing all service terms and service guarantees is available in the SLA Platform.*
	Trigger	
	Actions	*The SPECS application invokes the SPECS Enforcement module which retrieves the SLA to implement from the SLA Platform and prepares a plan to implement the signed SLA: it analyses the SLA, deduces alert thresholds, chooses the security and monitoring mechanisms to activate, and determines all related software to install along with their configurations.*
	Postconditions	
6	Phase	*SLA Implementation*
	Actor	*SPECS Enforcement module*
	Preconditions	*A plan has been built to implement a signed SLA.*
	Trigger	
	Actions	*The SPECS Enforcement module implements the plan, by configuring and deploying all the components in order to respect the features granted in the SLA. The SPECS Enforcement module deploys and configures monitoring agents and activates all the components and services.*
	Postconditions	
7	Phase	*SLA Implementation*
	Actor	*SPECS Enforcement module, SPECS Monitoring Module*
	Preconditions	*All components and services needed for SLA implementation have been correctly configured and activated.*
	Trigger	
	Actions	*The SPECS Enforcement module configures the Monitoring module with a monitoring policy by setting proper alert/violation thresholds for specific metrics.*
	Postconditions	
8	Phase	*SLA Monitoring*
	Actor	*SPECS Monitoring module*
	Preconditions	
	Trigger	
	Actions	*SPECS keeps collecting information about the provided service and evaluates them against the current monitoring policy.*
	Postconditions	
9	Phase	*SLA Negotiation*
	Actor	*End-user (second), SPECS application, SPECS Negotiation module, SPECS Enforcement module, SLA Platform*
	Preconditions	*The End-user has very basic security knowledge; she/he is able to express qualitatively requirements at a high-level of abstraction.* *The End-user shall be logged on SPECS.*
	Trigger	

	Actions	*The second End-user accesses the SPECS application interface, asking for a secure web container which fulfils the specific security requirements. The negotiation follows the same activities described in steps 1 to 4.*
	Postconditions	*The SLA, containing all information needed for SLA implementation, has been signed.*
10	Phase	*SLA Implementation*
	Actor	*SPECS application, SPECS Enforcement module, SLA Platform, SPECS Monitoring Module*
	Preconditions	*A valid signed SLA containing all service terms and service guarantees is available in the SLA Platform*
	Trigger	
	Actions	*The SPECS application invokes the SPECS Enforcement module which prepares and implements the plan that implements the signed SLA. It configures the Monitoring module with a monitoring policy by setting proper alert/violation thresholds for specific metrics. The implementation of the SLA follows the same activities described in steps 5 to 7 but the TLS protocol is added by reusing, for scalability purposes, the same resources adopted for the first End-user.*
	Postconditions	
11	Phase	*SLA Monitoring*
	Actor	*SPECS Monitoring module*
	Preconditions	
	Trigger	
	Actions	*SPECS keeps collecting information about the provided service and evaluates the monitoring policies.*
	Postconditions	*The signed SLA is fulfilled since neither alerts nor violations occur.*
Graphical Model		*Not reported to avoid replication of information. See D1.3 for detailed interactions between SPECS modules.*

Coverage Information	
Users	*U_1 (CSC:User)*
Target services	*TS_4 (Infrastructure as a Service)*
SPECS services	*See Appendix B of D5.1.2*
SLA	SLA_1, SLA_3, SLA_6, SLA_7

SWC.7 Secure_Web_Container_Web_Pool_Replication_Enhanced_Alert

General Information		
ID	*SWC.7 - Secure_Web_Container_Web_Pool_Replication_enhanced_alert*	
Version	*2.0*	
User Story	*WEB*	*Secure Web Container*
Invocation Chain	*IM1-CSP*	*Interaction Model 1- SPECS acting the role of CSP*
Scenario Steps		

General Description		*The End-user aims at acquiring a web container service fulfilling specific security requirements. In particular, the End-user requires a specific level of redundancy and session persistence among web container replicas. To enable this service, the End-User negotiates the desired features with SPECS. In this validation scenario, the bare web containers are offered by a CSP, while the redundancy features with session persistence among replicas is provided by SPECS through the WebPool mechanism. SPECS acquires the resources on behalf of the End-user (registered on SPECS), adds the WebPool mechanism's components, and sets-up proper resources to handle HTTP requests through proxy functionalities, in order to forward the requests to one of the available web container replicas. In this scenario, the proxy functionality is added, by SPECS, on a dedicated VM. In this scenario, an alert is generated because one of the replicas slows down, thus risking compromising the desired level of redundancy. SPECS reacts by isolating the replica and by restarting it. The scenario ends without any other alert*
Steps		
1	Phase	*SLA Negotiation*
	Actor	*End-user, SPECS application, SPECS Negotiation module*
	Preconditions	*The End-user has very basic security knowledge; she/he is able to express qualitatively requirements at a high-level of abstraction.*
	Trigger	
	Actions	*The End-user accesses the SPECS application interface. The negotiation request is forwarded to the SPECS Negotiation module, which retrieves the list of available SLA templates representing the available security services and the related security capabilities, controls and metrics. The services are returned to the End-user.*
	Postconditions	
2	Phase	*SLA Negotiation*
	Actor	*End-user, SPECS application*
	Preconditions	
	Trigger	
	Actions	*The End-user selects, among the available service offers, the desired one, i.e. the Secure Web Container. The End-user specifies the desired security features by selecting the capabilities she/he is interested in and by specifying the related security controls. She/he also specifies the desired metrics and sets the related SLOs. In particular, the End-user requires the adoption of a web pool mechanism to ensure session persistence among web container replicas*
	Postconditions	*A supply chain compliant to the End-user requirements is built.*
3	Phase	*SLA Negotiation*
	Actor	*SPECS application, SPECS Negotiation module, SPECS Enforcement module*
	Preconditions	*An Infrastructure-as-a-Service provider that offers VMs that fulfil the specific requirements is known to SPECS. The web pool mechanism is offered as a SPECS security mechanism.*
	Trigger	
	Actions	*The End-user's choices are forwarded by the SPECS application to the SPECS Negotiation module, which searches for valid supply chains. In particular, the list of supply chains is built with the help of the SPECS Enforcement module. In this step, an external CSP offering the Secure Web Container is identified; the web pool mechanism is identified among SPECS security mechanisms. For each valid supply chain, an SLA Offer is created. The set of SLA Offers are hence ranked and returned to the SPECS application.*
	Postconditions	
4	Phase	*SLA Negotiation*
	Actor	*End-user, SPECS application, SLA Platform*
	Preconditions	*The End-user shall be logged on SPECS.*

	Trigger	
	Actions	*The SPECS application validates the SLA Offers, which are then presented to the End-user. The End-user selects the SLA Offer in which the Secure Web Container is offered by an external CSP while the web pool mechanism is offered as a SPECS security mechanism. The selected SLA Offer is used to update and sign the SLA in the SLA Platform.*
	Postconditions	*The SLA, containing all information needed for SLA implementation, has been signed.*
5	Phase	*SLA Implementation*
	Actor	*SPECS application, SPECS Enforcement module, SLA Platform, SPECS Monitoring Module*
	Preconditions	
	Trigger	
	Actions	*SPECS acquires the VMs on behalf of the End-user on the external CSP and adds the web pool components. SPECS also sets up proper resources to handle HTTP request through proxying functionality in order to forward the requests to one of the available web containers. SPECS launches the related monitoring services.*
	Postconditions	
6	Phase	*SLA Monitoring*
	Actor	*SPECS Monitoring module*
	Preconditions	
	Trigger	
	Actions	*SPECS keeps collecting information about the provided service and evaluates the monitoring policies.*
	Postconditions	
7	Phase	*SLA Remediation*
	Actor	*SPECS Monitoring module, SPECS Enforcement module*
	Preconditions	*A redressing technique can be adopted according to the signed SLA, and is available as SPECS security mechanisms.*
	Trigger	*An alert regarding the level of redundancy is raised by the enforcement diagnosis, after the notification of a monitoring event by the SPECS Monitoring module.*
	Actions	*SPECS updates the implemented forwarding policy (redressing technique) and removes the affected web container from the pool of available web containers*
	Postconditions	*The discovered alert is solved and no more alerts are generated.*
Graphical Model		*Not reported to avoid replication of information. See D1.3 for detailed interactions between SPECS modules.*

Coverage Information	
Users	*U_1 (CSC:User)*
Target services	*TS_4 (Infrastructure as a Service)*
SPECS services	*See Appendix B of D5.1.2*
SLA	*SLA_1, SLA_3, SLA_6, SLA_7, SLA_8, SLA_9, SLA_10, SLA_11*

SWC.8 Secure_Web_Container_Web_Pool_Replication_Enhanced_Violation

General Information		
ID	*SWC.8 - Secure_Web_Container_Web_Pool_Replication_enhanced_violation*	
Version	*2.0*	
User Story	*WEB*	*Secure Web Container*
Invocation Chain	*IM1-CSP*	*Interaction Model 1- SPECS acting the role of CSP*
Scenario Steps		

General Description		*An End-user aims at acquiring a web container service fulfilling specific security requirements. In particular, the End-user requires a specific level of redundancy and session persistence among web container replicas. To achieve this service, the End-User negotiates the desired features with SPECS. In this validation scenario, the bare web containers are offered by a CSP, while the redundancy features with session persistence among replicas is provided by SPECS through the WebPool mechanism. SPECS acquires the resources on behalf of the End-user (registered on SPECS), adds the WebPool mechanism's components, and sets-up proper resources to handle HTTP requests through proxy functionalities, in order to forward the requests to one of the available web container replicas. In this scenario, the proxy functionality is added, by SPECS, on a dedicated VM. In this scenario, an alert is generated because one of the replicas goes down, thus compromising the desired level of redundancy. SPECS reacts by isolating the replica and by removing it from the pool of replicas. The SLA is violated since the level of redundancy is not preserved.*
Steps		
1	Phase	*SLA Negotiation*
	Actor	*End-user, SPECS application, SPECS Negotiation module*
	Preconditions	*The End-user has very basic security knowledge; she/he is able to express qualitatively requirements at a high-level of abstraction.*
	Trigger	
	Actions	*The End-user accesses the SPECS application interface. The negotiation request is forwarded to the SPECS Negotiation module, which retrieves the list of available SLA templates representing the available security services and the related security capabilities, controls and metrics. The services are returned to the End-user.*
	Postconditions	
2	Phase	*SLA Negotiation*
	Actor	*End-user, SPECS application*
	Preconditions	
	Trigger	
	Actions	*The End-user selects, among the available service offers, the desired one, i.e. the Secure Web Container. The End-user specifies the desired security features (in particular, the End-user requires the adoption of a web pool mechanism to ensure session persistence among web container replicas) by selecting the capabilities she/he is interested in and by specifying the related security controls. The End-user also specifies the desired metrics and sets the related SLOs.*
	Postconditions	*A supply chain compliant to the End-user requirements is built.*
3	Phase	*SLA Negotiation*
	Actor	*SPECS application, SPECS Negotiation module, SPECS Enforcement module*
	Preconditions	*An Infrastructure-as-a-Service provider that offers VMs that fulfil the specific requirements, is known to SPECS. The web pool mechanism is offered as a SPECS security mechanism.*
	Trigger	
	Actions	*The End-user's choices are forwarded by the SPECS application to the SPECS Negotiation module, which searches for valid supply chains. In particular, the list of supply chains is built with the help of the SPECS Enforcement module. In this step, an external CSP offering the Secure Web Container is identified; the web pool mechanism is identified among SPECS security mechanisms. For each valid supply chain, an SLA Offer is created. The set of SLA Offers are hence ranked and returned to the SPECS application.*
	Postconditions	
4	Phase	*SLA Negotiation*
	Actor	*End-user, SPECS application, SLA Platform*

	Preconditions	*The End-user shall be logged on SPECS.*
	Trigger	
	Actions	*The SPECS application validates the SLA Offers, which are then presented to the End-user. The End-user selects the SLA Offer in which the Secure Web Container is offered by an external CSP while the web pool mechanism is offered as a SPECS security mechanism. The selected SLA Offer is used to update and sign the SLA in the SLA Platform.*
	Postconditions	*The SLA, containing all information needed for SLA implementation, has been signed.*
5	Phase	*SLA Implementation*
	Actor	*SPECS application, SPECS Enforcement module, SLA Platform, SPECS Monitoring Module*
	Preconditions	
	Trigger	
	Actions	*SPECS acquires the VMs on behalf of the End-user on the external CSP and adds the web pool components, and sets up proper resources to handle HTTP request through proxying functionality in order to forward the requests to one of the available web containers. SPECS launches the related monitoring services.*
	Postconditions	
6	Phase	*SLA Monitoring*
	Actor	*SPECS Monitoring module*
	Preconditions	
	Trigger	
	Actions	*SPECS keeps collecting information about the provided service and evaluates the monitoring policies.*
	Postconditions	
7	Phase	*SLA Remediation*
	Actor	*SPECS Monitoring module, SPECS Enforcement module*
	Preconditions	
	Trigger	*An alert regarding a vulnerability threat on a web container is raised by the enforcement diagnosis, after the notification of a monitoring event by the SPECS Monitoring module.*
	Actions	*SPECS removes the affected web container from the pool of available web containers.*
	Postconditions	
8	Phase	*SLA Remediation*
	Actor	*SPECS Enforcement module*
	Preconditions	
	Trigger	*A violation of the signed SLA is detected by the enforcement diagnosis.*
	Actions	*SPECS notifies the violation to the End-user.*
	Postconditions	*The SLA is no more fulfilled*
Graphical Model		*Not reported to avoid replication of information. See D1.3 for detailed interactions between SPECS modules.*

Coverage Information	
Users	*U_1 (CSC:User)*
Target services	*TS_4 (Infrastructure as a Service)*
SPECS services	*See Appendix B of D5.1.2*
SLA	*SLA_1, SLA_3, SLA_6, SLA_7, SLA_9, SLA_12*

NGDC.1 Data_Center_Bursting_for_Storage_Resources

General Information		
ID	*NGDC.1 - Data_Center_Bursting_for_Storage_Resources*	
Version	*1.1*	
User Story	*ngDC*	*Next Generation Data Center*
Invocation Chain	*IM2-CSP*	*Interaction Model 2- SPECS acting in the role of CSP*
Scenario Steps		
General Description	*A CSP hosting its own next generation Data Center (ngDC), acting within a Cloud Service Customer (CSC) role, aims at using the SPECS framework to perform Cloud bursting in order to extend its Secure Storage as a Service (SStaaS) capabilities. This occurs during a period of increased storage demand, which exceeds the CSP's own ngDC storage capabilities.* *The CPS considers its storage as first class storage due to the fine grained control it has over all the security parameters. The CSP will allocate the first class storage to End-users that do not require high-security capabilities enabled. Otherwise, it will allocate storage acquired from an external provider through SPECS. The entire process is transparent to the End-user.* *Note, while a CSP acquiring storage resources from an external 3rd party CSP is typically defined as an End-user, it is not in the context of a SPECS defined in this way. That is, the CSP intends to resell its acquired external storage resources and so it is considered a CSC (in the context of SPECS). For ease of exposition 'customer' is used as a common reference to either a CSC or End-user of the CSP hosting the ngDC.*	
Steps		
1	Phase	*Negotiation*
	Actor	*CSC (CSP is acting within a CSC role)*
	Preconditions	*The CSC monitors the current state of its ngDC in terms of its on-premise storage resources.*
	Trigger	*Capacity threshold reached.*
	Actions	*The CSC asks its locally hosted SPECS for an external CSP offering SStaaS, which fulfils its specific security requirements. These security requirements might be based on either or both the CSC's own security requirements or that of the CSC's own customers. Examples of security requirements are the data geo-location, the Drive type, RAID level, etc.*
	Postconditions	
2	Phase	*Negotiation*
	Actor	*SPECS Negotiation module*
	Preconditions	*An external CSP that fulfils the specific secure storage requirements must already be present within the locally hosted CSC's SPECS SLA Repository.*
	Trigger	
	Actions	*SPECS searches for possible supply chains compliant with the specified secure storage requirements, evaluates if the external CSP fulfils the End-User requirements SPECS will allocate directly the resource, otherwise it will allocate resource on the local storage platform.*
	Postconditions	
3	Phase	*Negotiation*
	Actor	*CSC*
	Preconditions	
	Trigger	

	Actions	*The CSC selects one supply chain from the retrieved list and signs the SLA with the external CSPs that form part of the SPECS supply chain.*
	Postconditions	
Graphical Model		
Coverage Information		
Users	*U_1 (CSC:user)*	
Target services	*TS_3 (Data Storage as a Service)*	
SPECS services	*See Appendix B of D5.1.2*	
SLA Lifecycle	*SLA_1, SLA_3, SLA_4, SLA_5, SLA_6*	

NGDC.3 Data_Center_Storage_Selection

General Information			
ID	*NGDC.3 – Data_Center_Storage_Selection*		
Version	*1.0*		
User Story	*NgDC*	*Next Generation Data Center*	
Invocation Chain	*IM2-CSP*	*Interaction Model 2- SPECS acting the role of CSP*	
Scenario Steps			
General Description	*A CSP owning SPECS and hosting its own ngDC, acting within a CSC role, aims at using the SPECS framework to perform Cloud bursting in order to extend its Secure SStaaS capabilities. This occurs during a period of increased storage demand, which exceeds the CSP's own ngDC storage capabilities. In this validation scenario, the desired features are entirely implemented by an external CSP, while SPECS only offers the End-user the ability to search, rank and select a service, which is compliant to her/his requirements. Moreover, in this scenario, SPECS supports the End-user in signing an SLA with the selected provider.*		
Steps			
1	Phase	*SLA Negotiation*	
	Actor	*CSC (CSP is acting within a CSC role)*	
	Preconditions	*The End-user has good security knowledge; she/he is able to express qualitatively requirements at a low-level of abstraction.*	
	Trigger		
	Actions	*The CSC asks its locally hosted SPECS for an external CSP offering SStaaS, which fulfils its specific security requirements. These security requirements might be based on either or both the CSC's own security requirements or that of the CSC's own customers. Examples of security requirements are the data geo-location, the Drive type, RAID level, etc. The End-user accesses the SPECS application interface. The negotiation request is forwarded to the SPECS Negotiation module, which retrieves the list of available SLA templates representing the available security services and the related security capabilities, controls and metrics. The services are returned to the End-user.*	
	Postconditions		
2	Phase	*SLA Negotiation*	
	Actor	*CSC, SPECS application*	
	Preconditions		
	Trigger		
	Actions	*The End-user selects, among the available service offers, the desired one, i.e. the Database and Backup. The End-user specifies the desired security features by selecting the capabilities she/he is interested in and by specifying the related security controls. The End-user also specifies the desired metrics and sets the related SLOs.*	

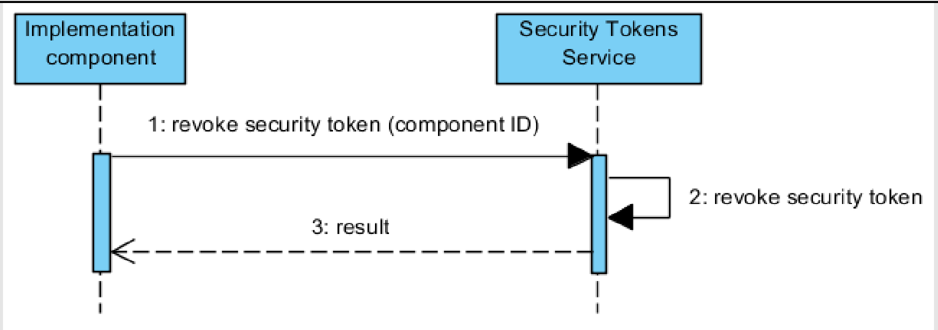
	Postconditions	*A supply chain compliant to the End-user requirements is built.*
3	Phase	*SLA Negotiation*
	Actor	*SPECS application, SPECS Negotiation module, SPECS Enforcement module*
	Preconditions	*A secure storage service that fulfils the specific security requirements is known to SPECS.*
	Trigger	
	Actions	*The End-user's choices are forwarded by the SPECS application to the SPECS Negotiation module, which searches for valid supply chains. In particular, the list of supply chains is built with the help of the SPECS Enforcement module.* *For each valid supply chain, an SLA Offer is created. The set of SLA Offers are hence ranked and returned to the SPECS application. The CSPs also add the cost of each service offer.*
	Postconditions	
4	Phase	*SLA Negotiation*
	Actor	*End-user, SPECS application, SLA Platform*
	Preconditions	*The End-user shall be logged on SPECS.*
	Trigger	
	Actions	*The SPECS application validates the SLA Offers, which are then presented to the End-user. The service offer is associated with an SLA published by an external CSP. The End-user either:* *1. Accepts and signs the SLA offered by the external CSP;* *2. Does not select any SLA Offer from the list and repeats the whole process from step 1 (possibly specifying a different set of requirements);* *3. Does not select any SLA Offer from the list and exits the application.*
	Postconditions	*In case 1 - the signed SLA is stored by SPECS. The End-user is enabled to invoke the desired service on the external CSP with the configuration information included in the SLA.*
Graphical Model		*Not reported to avoid replication of information. See D1.3 for detailed interactions between SPECS modules.*

Coverage Information	
Users	*U_1 (CSC:User)*
Target services	*TS_3 (Data Storage as a Service)*
SPECS services	*See Appendix B of D5.1.2*
SLA	*SLA_1, SLA_3. SLA_4, SLA_5*

CRO.3 Security_Tokens_Revocation

General Information		
ID	*CRO.3 - Security_Tokens_Revocation*	
Version	*2.0*	
User Story	*n.d.*	
Invocation Chain	*n.d.*	
Scenario Steps		
General Description	*In this validation scenario, the revocation of a security token is shown.*	
Steps		

1	Phase	*Token Revocation*
	Actor	*Implementation component*
	Preconditions	
	Trigger	*The SLA is terminated.*

	Actions	*The Implementation component sends request to the Security Tokens Service to revoke the security tokens issued to a specific SPECS component. The Implementation component is authenticated by its certificate.*
	Postconditions	
2	Phase	*Token Revocation*
	Actor	*Security Tokens Service*
	Preconditions	*The revoke request is authenticated and authorized.*
	Trigger	
	Actions	*The Security Tokens Service finds the tokens issued to the specified SPECS component, marks them as revoked and adds them to the token revocation list.*
	Postconditions	
3	Phase	*Token Revocation*
	Actor	*All SPECS components*
	Preconditions	
	Trigger	*Periodical update of the token revocation list*
	Actions	*SPECS components periodically pull delta token revocation list and update local token revocation list cache. The revoked tokens are propagated to the local token revocation list caches.*
	Postconditions	
4	Phase	*Token Revocation*
	Actor	*Blocked component*
	Preconditions	*The revoked tokens were propagated to local token revocation list caches.*
	Trigger	
	Actions	*The blocked component calls some other SPECS component with security token attached. The target component validates the token, finds out that the token is on the revocation list and denies the request.*
	Postconditions	
Graphical Model	 Implementation component Security Tokens Service 1: revoke security token (component ID) 2: revoke security token 3: result	

Coverage Information	
Users	*U_1 (CSC:User)*
Target services	*Not Applicable.*
SPECS services	*See Appendix B of D5.1.2*
SLA	*Not Applicable.*

CRO.10 SPECS_Application_Development

General Information		
ID	*CRO.10 - SPECS_Application_Development*	
Version	*1.0*	
User Story	*n.d.*	
Invocation Chain	*n.d.*	

Scenario Steps		
General Description		*A SPECS developer aims at developing a new SPECS application. In this scenario, the development of a new SPECS application, using the default SPECS application as a template, is shown.*
Steps		
1	Phase	*Cloud Service Definition*
	Actor	*SPECS developer*
	Preconditions	
	Trigger	
	Actions	*The SPECS developer defines the types of cloud services to deliver and prepares the related cookbooks. She/he needs to specify the mechanisms able to enforce specific security capabilities and/or monitor specific metrics, as well as she/he needs to provide proper mechanisms to automatically deploy and configure the target services themselves.*
	Postconditions	
2	Phase	*Prepare Security Mechanisms*
	Actor	*SPECS developer*
	Preconditions	
	Trigger	
	Actions	*The SPECS developer selects, among available security mechanisms, those needed to offer the cloud services.*
	Postconditions	
3	Phase	*Prepare SLA Template*
	Actor	*SPECS developer*
	Preconditions	
	Trigger	
	Actions	*The SPECS developer builds a WS-Agreement-compliant SLA template, which summarizes the security capabilities that can be offered and the related guarantees.*
	Postconditions	
4	Phase	*Deploy SLA Templates and Security Mechanisms*
	Actor	*SPECS developer, SLA Platform*
	Preconditions	
	Trigger	
	Actions	*The SPECS developer deploys the security mechanisms in order to make them available to the SPECS application. All the cookbooks must be registered with the Chef Server in order to enable the SPECS Enforcement module to implement the SLA, and the mechanisms' metadata must be registered in the SLA Platform in order to enable the SPECS application to retrieve the information and to implement the SLA.* *The SPECS developer tests the deployed SPECS application.*
	Postconditions	
Graphical Model		*Not reported to avoid replication of information. See D1.3 for detailed interactions between SPECS modules.*

Coverage Information	
Users	*U_4, U_5, U_6 (CSN:developer)*
Target services	*Not Applicable*
SPECS services	*See Appendix B of D5.1.2*
SLA	*Not Applicable*

AAA.1 Identity_Management_Set-up

General Information		
ID	*AAA.1 – Identity_Management_Set-up*	
Version	*1.0*	
User Story	*STOIM*	*Secure Storage with Identity Management*
Invocation Chain	*IM2-CSP*	*Interaction Model 2- SPECS acting the role of CSP*
Scenario Steps		
General Description	*In this scenario, a customer acquires the enhanced secure storage service from the SPECS Owner, configures the service and sets the access control policies for its End-users by using the identity management features offered by the service. Moreover, the provider configures the Identity Federation by identifying the supported identity providers.*	

Steps		
1	Phase	*Service acquisition*
	Actor	*Customer, SPECS Owner*
	Preconditions	
	Trigger	
	Actions	*The customer acquires the enhanced secure storage service from the SPECS Owner and is provided with access to an application for its configuration.*
	Postconditions	
2	Phase	*Identity Management Set-up*
	Actor	*Customer, AAA mechanisms component*
	Preconditions	
	Trigger	
	Actions	*The customer accesses the application and configures, via the AAA mechanisms offered, the storage service that will offer to End-users by identifying the access control policy. The customer sets different authorization roles for the users and configures the tools for authentication (e.g., LDAP, OAUTH) and authorization (e.g., XACML).*
	Postconditions	
3	Phase	*Identity Federation configuration*
	Actor	*Customer, AAA mechanisms component*
	Preconditions	
	Trigger	
	Actions	*The customer, via the AAA mechanisms offered with the service, identifies a set of external Identity Providers that he aims at supporting in an Identity Federation (e.g., Facebook)*
	Postconditions	
Graphical Model	*Not reported*	

Coverage Information	
Users	*U_1(CSC:User)*
Target services	*Not Applicable*
SPECS services	*AAA mechanisms*
SLA	*Not Applicable*

AAA.2 User_Registration

General Information			
ID		*AAA.2 – User_Registration*	
Version		*1.0*	
User Story		*STOIM*	*Secure Storage with Identity Management*
Invocation Chain		*IM2-CSP*	*Interaction Model 2- SPECS acting the role of CSP*
Scenario Steps			
General Description		*In this scenario, an End-user of the enhanced secure storage service performs a registration by providing her/his data.*	
Steps			
1	Phase	*Registration*	
	Actor	*End-user, AAA mechanisms component*	
	Preconditions	*The provider of the service (i.e. the customer that has acquired the service from the SPECS Owner in the User Story) has defined an access control policy.*	
	Trigger		
	Actions	*The End-user fills the registration form with her/his personal information and specifies the features of the storage service she/he is interested to use. The information is submitted to the AAA component.*	
	Postconditions		
2	Phase	*Registration*	
	Actor	*AAA mechanisms component, End-user*	
	Preconditions		
	Trigger		
	Actions	*A new account is created for the End-user and submitted information is associated with it. The End-user is provided the credentials to access the application.*	
	Postconditions		
Graphical Model		*Not reported.*	
Coverage Information			
Users		*U_1(CSC:User)*	
Target services		*Not Applicable*	
SPECS services		*AAA mechanisms*	
SLA		*Not Applicable*	

AAA.3 User_Access_Internal_Account

General Information			
ID		*AAA.3 - User_Access_Internal_Account*	
Version		*1.0*	
User Story		*STOIM*	*Secure Storage with Identity Management*
Invocation Chain		*IM2-CSP*	*IM2-CSP*
Scenario Steps			
General Description		*In this scenario, an End-user requests the access to the storage system by using the account created when registering with the service;*	
Steps			
1	Phase	*Authentication*	
	Actor	*End-user, AAA mechanisms component*	

	Preconditions	
	Trigger	
	Actions	*The End-user submits an authentication request (through, for example, an SAML request) to the SPECS AAA component by using the account previously created during registration.*
	Postconditions	
2	Phase	*Authentication*
	Actor	*End-user, AAA mechanisms component*
	Preconditions	*The End-user has a valid account on the application*
	Trigger	
	Actions	*The AAA component checks the account in the internal repository (e.g., LDAP server) and authenticates the End-user, by applying the access control policy related to her/his role.*
	Postconditions	
Graphical Model		*Not reported*
Coverage Information		
Users	*U_1(CSC:User)*	
Target services	*Not Applicable*	
SPECS services		
SLA	*Not Applicable*	

AAA.4 User_Access_External_Account

General Information		
ID	*AAA.4 - User_Access_External_Account*	
Version	*1.0*	
User Story	*STOIM*	*Secure Storage with Identity Management*
Invocation Chain	*IM2-CSP*	*IM2-CSP*
Scenario Steps		
General Description	*In this scenario, an End-user requests access to the storage system by using an external account belonging to a supported Identity Provider. When the user chooses to authenticate through an external source, the application checks that the external account is associated with a supported identity provider. In this case, the user is authenticated. Otherwise the application asks if the End-user wants to associate the external account to her/his existing internal account. In this latter case, the End-user must first be authenticated on the application in order to prove the ownership of the internal account.*	

Steps		
1	Phase	*Authentication*
	Actor	*End-user*
	Preconditions	*The End-user has a valid account on the selected external authentication source.*
	Trigger	
	Actions	*The End-user requests the access to the storage service by selecting an external authentication source and performs the login with the credentials of the external account, retrieving her/his personal information.*
	Postconditions	*The End-user is authenticated on the external authentication source.*
2.1	Phase	*Authentication*
	Actor	*AAA component, End-user*

	Preconditions	*An internal account exists for the End-user. The internal account is already linked to the external account.*
	Trigger	
	Actions	*The AAA component checks if the external account is associated with any valid internal account and authenticates the End-user.*
	Postconditions	*The End-user is authenticated on the application.*
2.2	Phase	*Authentication*
	Actor	*AAA component, End-user*
	Preconditions	*An internal account exists for the End-user. The internal account is not yet linked to the external account.*
	Trigger	
	Actions	*The AAA component checks if the external account is associated with any valid internal account and does not find any match. The AAA component asks the End-user to associate the external account to her/his existing internal account, if any exists.*
	Postconditions	*The internal account of the End-user is linked to this/he external account.*
3.2	Phase	*Authentication*
	Actor	*End-user, AAA component*
	Preconditions	
	Trigger	
	Actions	*The End-user logs into the application with the credentials of the internal account. The AAA component authenticates the End-user.*
	Postconditions	*The AAA component End-user is authenticated.*
4.2	Phase	*Account association*
	Actor	*SPECS AAA component*
	Preconditions	
	Trigger	
	Actions	*The link with the external account is created for the user entry by the AAA component.*
	Postconditions	*The link to the external account is stored in the AAA component repository*
Graphical Model		*Not reported*

Coverage Information	
Users	*U_1(CSC:User)*
Target services	*Not Applicable*
SPECS services	
SLA	*Not Applicable*