D6.1
Integration and validation design and set-up

Project number	611458
Project acronym	SECURED
Project title	SECURity at the network EDge
Project duration	36 months (1/10/2013–30/9/2016)
Programme	FP7 (Collaborative Project)

Deliverable type	**R** - Report
Deliverable number	D6.1
Version (date)	v1.3 (10/02/2015)
Work package(s)	WP6
Due date	M12

Responsible organisation	VTT
Editor	Jarkko Kuusijärvi
Dissemination level	**PU** - Public

Abstract	This deliverable describes the design of the SECURED integration lab and the system validation test-beds for the two rounds (basic and full architecture) of the SECURED development process. It includes description of the respective procedures and the IT support infrastructure that enable conducting integration and validation tasks efficiently, systematically and transparently.
Keywords	Integration and validation, test cases, test-beds.

(This page is left blank intentionally.)

Editors

Jarkko Kuusijärvi (VTT)

Jouni Hiltunen (VTT)

Reviewers

Adrian L. Shaw (HPLB)

Marcelo Yannuzzi (UPC)

Diego Montero (UPC)

Antonio Lioy (POLITO) – quality control

Contributors

Michael Georgiades (PTL)

Savvas Charalambides (PTL)

Antonio Pastor (TID)

Fulvio Risso (POLITO)

Roberto Bonafiglia (POLITO)

Acknowledgement

This work was partially supported by the European Commission (EC) through the FP7-ICT programme under project SECURED (grant agreement no. 611458).

Change Log

Version	Date	Note	Author
v1.0	30.09.2014	Base version.	J. Kuusijärvi, J. Hiltunen (VTT)
v1.1	08.10.2014	Quality control	Antonio Lioy (POLITO)
v1.2	30.01.2015	Changed according to review recommendations	J. Kuusijärvi (VTT)
v1.3	10.02.2015	Quality control	Antonio Lioy (POLITO)

Executive summary

This document describes the Integration and Validation procedures and the related IT infrastructure for the SECURED project. General software testing process and related verification and validation procedures are explained and exemplified in the context of this project.

The virtual integration labs are designed at a high-level to be able to easily integrate different software components in various partner sites. For the validation activities we also describe the two test-beds, namely, the test-bed for corporate users hosted by VTT and a test-bed for domestic and mobile users hosted by PrimeTel.

The integration and validation work is designed according to the two integration and validation rounds for both basic and full architecture. The use cases described in this document are derived from the high-level use cases in the Description of Work. The most relevant components and the system integration tests cases for selected components are described according to the system integration test case template. Additionally, this deliverable presents initial templates for integration test cases, system tests, and validation use cases.

Finally, this document presents the currently identified system integration test cases based on the available information at the time of writing. While this document presents the high-level processes/workflows, more detailed information on specific test support tools (e.g. Source Code Management, test automation) is provided in a private project web page. In addition, the detailed test plans and test cases for integration tests of the alpha (basic) and full architectures will also be maintained in a private project web page to support the iterative integration process.

Contents

1 Introduction

This document describes the Integration and Validation (I&V) procedures of the SECURED project and the ICT infrastructure supporting them. Integration will be carried out in virtual integration labs that will be hosted by various project partners. This document also describes the high-level process of how components are delivered to the integration work and how feedback will be given on the testing activities.

During the project, we will have two I&V rounds, specifically one for the basic (alpha) architecture and one for the full architecture. All the high-level test scenarios and use cases for these activities are described in this document according to the current knowledge of the SECURED project and architecture specifications. The more detailed use cases and test scenarios will be defined later in a private live web version for the project.

The aim of the I&V activities is to execute validation of the technical specifications and components and answer the question "does this product work as expected" and to provide feedback for the project assessment and the technical work packages implementing the various components.

This document constitutes guidelines for I&V procedures followed in the project. Compared to relevant standards such as IEEE829, Standard for Software and System Test Documentation [1], the presented guidelines concentrate on the essential procedures to enable conducting efficient work in the scope of the project.

The document is structured as follows. Section 2 describes the general integration testing and system validation process for the SECURED project. In particular, it describes the general software verifications strategies, timeline and milestones for the integration work and the high-level process for both integration testing and system validation.

Section 3 presents the verification and validation infrastructure designed for SECURED, describing the virtual integration labs (hosted at various partner sites) and the end-user validation test-beds used for validating the whole system.

Section 4 describes I&V work at a more practical level and defines example templates for integration test scenarios, integration test cases and validation use cases, and presents the current high-level system integration test cases designed. All the activities planned for integration and validation throughout the project are designed separately for the basic (alpha) and full architectures since the requirements, specifications and features for them vary.

Finally, Section 5 summarises the contents of the whole deliverable, while Appendix A contains the current set of system integration test cases.

2 Integration testing and system validation processes

This section describes the general testing process that will be carried out in the technical work packages (WP3, WP4, and WP5) and in the integration work package WP6. Initial schedules for component integration and validation are presented in the following sub-section. The integration and validation process is described, encompassing what kind of tests are executed at a given phase, how the components will be delivered to testing and how feedback will be given back to the technical implementation work packages. Once a stable version of the SECURED architecture is ready, we will perform a validation on small scale test-beds targeted at the end-users in both corporate setup and in the fixed domestic and mobile end-user setup.

2.1 Software verification strategies

Today in complex software development projects it is widely known that merely concentrating on software testing and system validation in the end of the software development process is not enough to ensure high quality of the final product. For instance, while selecting the implemented functionalities during the requirement engineering process, one must take account resource and time constraints to enable high quality work. In the case of SECURED (which is a multi-actor research project rather than a single-actor industrial software development one) we must concentrate on defining what is essential to produce for a working software prototype, i.e. integration testing and system validation processes.

Figure 1 depicts the main applied verification and validation mechanisms. Integration testing is part of verification. Verification includes all means for checking consistency between entities such as software implementation and specifications. The means may include code inspection or software testing, as in the case of integration testing. A software test may be performed by creating test case, running the test case and checking the result of the run. A test case defines a series of events that enable a specific verification to take place. Specifically, the integration testing is conducted between subsystems, which include two or more software components, and architecture specifications, which define how the subsystem should behave. Integration testing is finished, and system validation started, after all test cases are passed successfully, i.e. no inconsistencies are found. Modification to any entities under verification are required if inconsistencies are observed at any point.

The basic integration testing strategies are incremental testing and big bang testing [2]. In big bang testing the verification of the developed components is done only with the complete system. In contrast, the incremental testing may be started earlier in the development cycle as it proceeds from smaller subsystem to larger ones. However, usually there are unavailable components that are required for the incremental testing to be executed. These unavailable components must be implemented with stubs that simulate lower level components or drivers that simulate higher level components required for each integration test. The advantage of the big bang testing is the lack of stubs and drivers but it has the disadvantage of providing less tractable feedback on underlying software inconsistencies than in the incremental approach [2].

Incremental testing can be further categorized into various strategies such as structural and feature oriented approaches. In the structural approach, the components are sorted to a hierarchy based on the use/include relationships. In a top-down strategy, testing begins from the top of the hierarchy and in the bottom-up strategy from the bottom of the hierarchy. Top-down strategy provides the advantage of early end-user feedback but it is vulnerable to ignore system constraints that are bound to arise from low level components, such as hardware limitations, that can be effectively handled with bottom-up strategy. Using these two strategies simultaneously is often called as backbone testing strategy. It can provide the advantages of both strategies but it requires more effort in planning and monitoring compared to single strategy usage.

In contrast to the structural approach, the feature-oriented approach does not follow static hierarchical order of integration test execution. Instead, the order could be defined in various ways such as identifying threads of execution that correspond to system features or identifying risk factors

of each component. The thread-based approach emphasises module collaboration related to a specific functionality which enables better tracking of the readiness level in context of externally visible functionalities and even enables early releases of non-complete system. Instead, the risk-based approach may be appropriate when there is some uncertainty regarding the system constraints and requirements. It may enable early modifications to the specifications, reducing the number of iteration in the development process. The drawback of the feature-oriented approach is the increased complexity in planning and management between system design and integration testing.

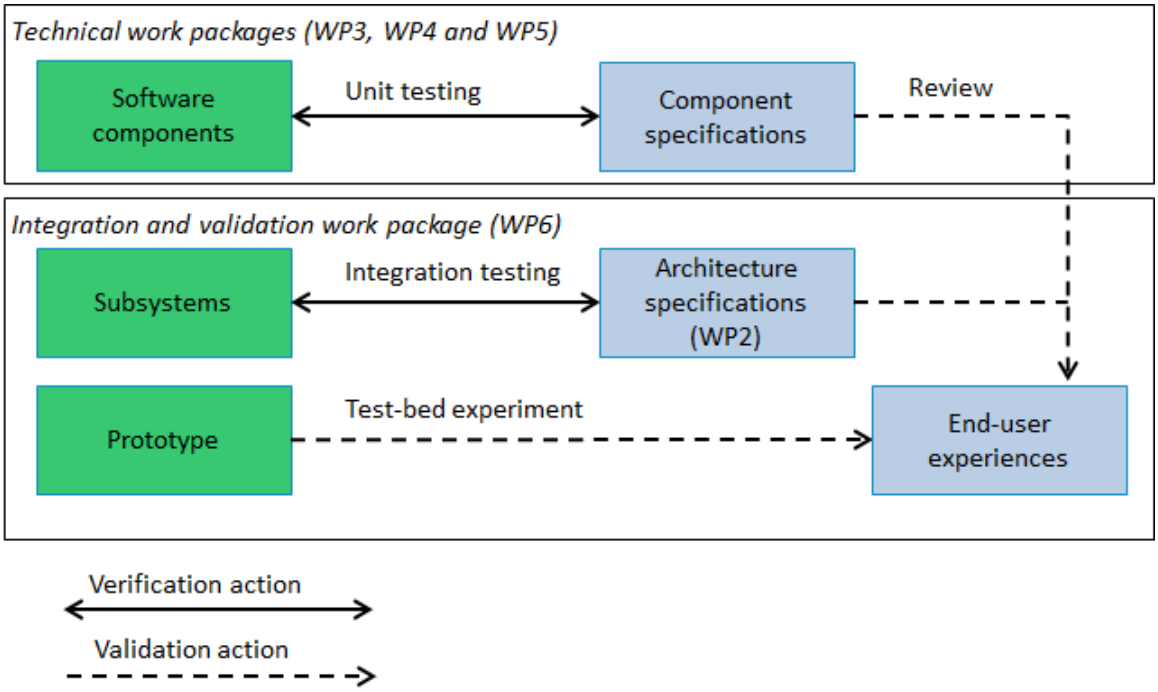

Figure 1: Overview of the verification and validation mechanisms

System testing and regression testing are particular parts of integration testing. System testing is the final part of integration testing which is executed on the complete system. Incremental testing allows focussing the system testing on system-wide properties, such as performance issues. What distinguishes system testing from other integration testing is that the system wide properties are often not derived directly from component design and specifications. The incremental testing strategy stipulates some of the integration tests to be re-executed after moving to larger subcomponents or when previously verified component is revised. This kind of testing is called regression testing. It may be more efficient to create special test cases for regression testing, thus decreasing the number of test cases to be executed, but the potential efficiency gain depends on the test automation possibilities and scope of the existing test cases.

In contrast to verification, validation includes comparing entities against their intended purposes. Specifically, system validation ensures that the prototype, which consists of all components included in the basic or full architecture, works as intended in predefined scenarios, which are implemented through test-bed experiments with real end-users. The test-bed experiments may reveal unexpected behaviour of the subsystems which may cause modification of the software component that in turn requires running of relevant integration tests again. Such iterations are expected to be frequent, which suggests automating the integration testing. In addition, even if the prototype behaves according to the specification, the end-user experience might not be satisfactory which requires additional validation action in which the specifications are compared to end-user experience and the specifications are modified accordingly causing another iterative cycle in the development process.

In the context of software development, the integration testing and system validation are performed in the later stages of the software development iteration cycle. Before they can be started, other

activities must have been completed: planning, specification, design, and software implementation. An important activity related to verification and validation mechanisms is unit testing between software components and components specifications. Thorough unit testing is a prerequisite for conducting integration tests successfully within the project schedule without an unfeasible number of iterations in integration testing. There exist numerous testing strategies specifically suitable for unit testing that are not discussed here.

This document constitutes the planning phase of the system development: architecture specification must be ready, software components must be implemented, necessary infrastructure must be built and test cases must be created before the processes defined here can be applied.

2.2 Integration and validation timeline and milestones

The schedule for I&V activities during the project is presented in the following sub-sections for both basic and full architecture. During the course of integrating the two architecture versions the internal deadlines will be coordinated with the relevant technical WPs that produce components to be integrated. For example, PSAs required to run the validation test-beds will be discussed with WP5 in order for them to be able to produce the required PSAs in time for validation. The milestones and schedules for integration and validation will be published and updated on the private internal web site.

Integration and validation of the basic architecture

The purpose of the basic (alpha) architecture integration and validation round is to get feedback from the actual end-users about the SECURED architecture, its benefits and its deficiencies so that we can correct them for the full architecture. The integration of the basic architecture runs between months M13-M24, while being in support mode for validation in the period M21-M24. The validation round for basic architecture runs between M21-M24, that is, June 2015 to September 2015. The actual runtime for the test-beds will be decided later, but we expect to run the test-beds for 2-4 weeks in order to get enough usage data. Figure 2 depicts the timeline for basic architecture integration and validation. M18 is the milestone for having individual components ready for basic architecture, so that we can integrate them before the validation run starting in M21 with the basic architecture prototype release. In M24 we will publish D6.2 “Basic architecture: integration and validation report”.

Figure 2: Basic architecture integration and validation schedule

Integration and validation of the full architecture

The purpose of the full architecture integration and validation round is to get feedback from the actual end-users about the final SECURED architecture, its benefits and its deficiencies. The integration of the full architecture runs between M25-M33, while being in support mode for validation in M33 (to M36). The validation rounds for full architecture runs between M33-M36. As in the basic architecture test-bed, the actual runtime for the test-beds will be decided later, but we expect to run the test-beds for 2-4 weeks in order to get enough usage data. Figure 3 depicts the timeline for full architecture integration and validation. M30 is the milestone for having individual components ready for the full architecture, so that we can integrate them before the validation run

starting in M33 with the full architecture prototype release. In M36 we will publish D6.3 "Full feature integration and validation report".

Figure 3: Full architecture integration and validation schedule

2.3 Integration testing process

Integration testing is a verification mechanism aimed to ensure compatibility between software components, thus focusing mostly on interfaces between components. In this project, the incremental approach is selected, which enables integration testing on parts of the system that are used for demonstrations during the project. In a multi-actor research project, such as SECURED, achieving light test management overheads is necessary, which eliminates the use of feature-oriented testing approaches and backbone structural strategy. That leaves us with the choice between top-down and bottom-up strategies. However, neither one is able to provide a framework for delivering prototypes of the system within the project schedule. Therefore, integration testing proceeds in the order in which specific functionalities are selected for demonstration and the required components are implemented accordingly. The integration testing follows bottom-up approach since it is assumed that the demonstrated functionalities are implemented in bottom of the use/include hierarchy while the top of the hierarchy could be simulated for the demonstrations.

Figure 4 depicts the high-level integration testing flow and interaction with test support systems.

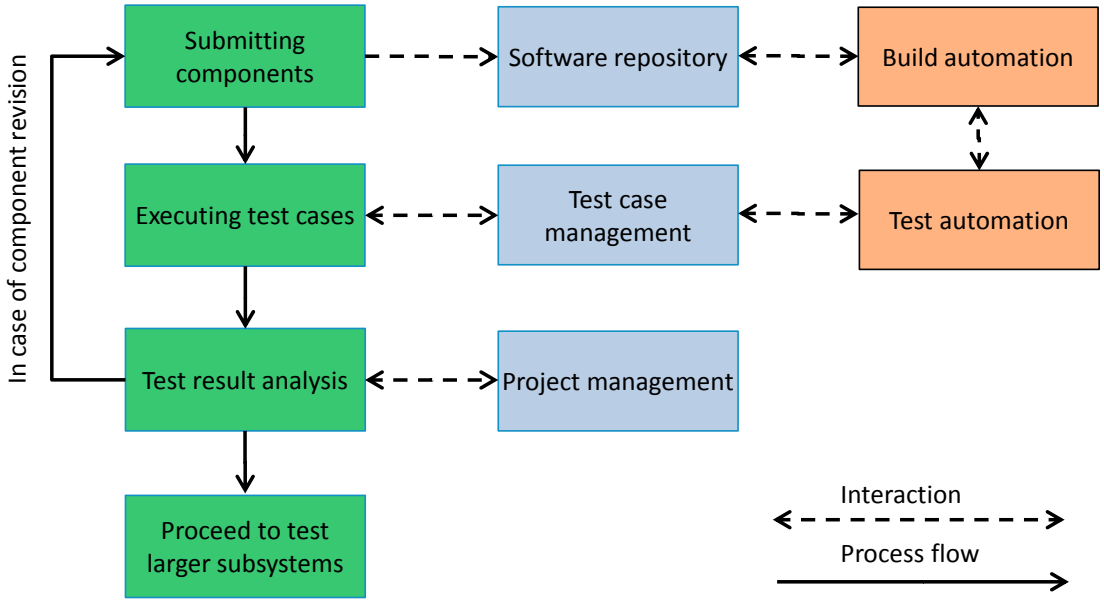

Figure 4: Integration testing process flow and interaction with test support systems

The following list describes the planned high-level process of submitting / updating components:

1. the developer of the technical WP runs all the defined unit tests in the local repository, either manually or by building the component via the build automation software;

2. build automation software builds the components and runs automated testing, if success uploads the build to I&V repository;
3. the developer in WP6 receives notification of new components and runs the defined test suites/test cases, which can be part automatic and part manual;
4. test case execution results are shown in the I&V wiki/web page/test tool, where technical WP participants can read the reports.

The exact integration process will be described in detail at the internal web site, once the planned testing support services are installed and configured. Regression testing, which is needed during the system evolution, is primarily executed through test automation. The preferred method is to run regression tests overnight when components are updated to the software repository.

2.4 System validation process

After the integration work is completed (i.e. we have the necessary components that have passed the test criteria in the integration testing), the components are frozen and delivered to validation activities, where they will be used to run both validation test-beds: corporate and fixed domestic and mobile end-user test-bed. The system validation will be executed according the process flow, of which the main interactions are presented in Figure 5. The validation starts with end-user selection and training. Careful selection of end-users is necessary to ensure the end-user actions will cover the required use cases. In addition, independence of the validation will be assured by selecting end-users that are not working for the SECURED project. Also, technical support staff is needed to ensure end-user hardware and software prerequisites are met and that the end-users have the basic knowledge to operate the hardware and software themselves.

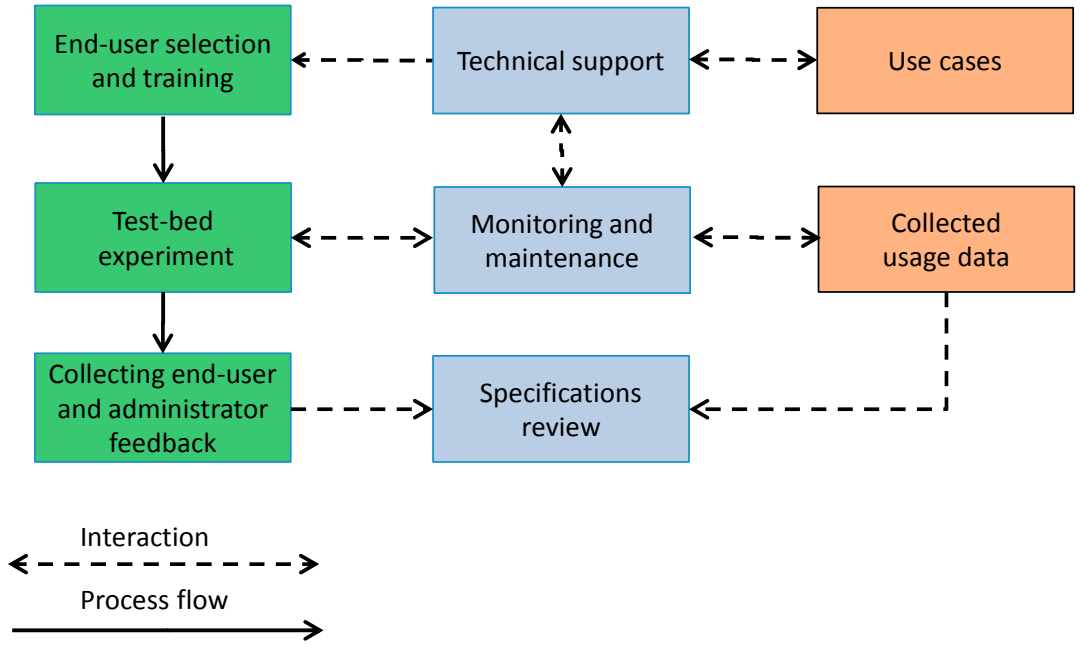

Figure 5: System validation process flow and main interaction activities

Before the test-bed experiment can be started, the operation of the test-bed is tested by the technical support staff with basic use cases. During the experiment, the monitoring and maintenance of the test-bed is important. It is performed automatically, such as in the case of usage data collection, or by technical support staff that will ensure availability of the test-bed during the test-bed execution. Careful monitoring of the test-bed operation is also important for discovering failures that might cause harm to end-users or test-bed operators and consequently stop the experiment prematurely. Such cases could occur during frequent unavailability of network connection which hinders the work of the end-user in corporate use case or if a serious security vulnerability is detected.

End-user and administrator feedbacks are collected during and/or after the experiment. They enable, together with the collected usage data, evaluation of the system under validation through specifications review. The feedback is delivered through questionnaires that will focus on the quality of experience aspects as well as problems encountered and ideas for improving the system. Any problem situations can also be notified during the trial period via a web page. The end-users will have to agree to our privacy policy which allows gathering of anonymous usage data during the experiment. Interesting data to be monitored includes amount of data and control network traffic, security policy usage, PSA operations performed, NED resource utilization, and types of the client terminals used. The usage data will be collected by the parties running the validation test-beds and delivered after the validation period to the technical WPs and project assessment.

Because the two validation runs have different features in terms of platform functionality and more importantly the available PSAs, the users are guided on what the SECURED platform can and cannot do. This is specifically important in the validation of the basic architecture, since some security features might not be implemented that will be available in the final validation round. Additionally, feedback received during the first validation of the basic architecture allows us to focus on problems and features that the end-users have reported and/or suggested.

3 Verification and validation infrastructure

The infrastructure facilitating the verification and validation work is presented in Figure 6. It consists of integration labs, physical test-beds and testing support services. The infrastructure is physically distributed among different partner's premises but any partner can access it as needed. This enables efficient use of resources without unnecessarily duplicating the infrastructure.

The integration labs are distributed (virtual) testing environments that each contributes to the integration testing with some specific focus in terms of components, scenarios and external entities. Interconnection between the labs enables comprehensive support for the full range of integration tests and project assessment tasks. However, some of the testing must be conducted locally, such as that related to security testing and mobile access,.

The testing support services are hosted at POLITO premises and contain services to support the integration and validation processes. The services included in the support services are the open-source Source Code Management (SCM) GitLab [3], which also provides issue tracking for bugs and features, wiki for the software development and management, and the open-source Jenkins CI [4], which provides continuous integration (CI) services for software development and automating unit and integration test case execution. Additionally, the supporting services will provide some service for test case management, test automation, and for reviewing code commits.

The physical test-beds will be hosted at VTT and PrimeTel premises and will be partially connected to the testing support services to provide feedback about the actual validation runs. The test-beds will not be connected to each other or the virtual integration labs to maintain an isolation between the end-user tests (i.e. field-trials) and the integration labs, which will run the integration tests until the test criteria have been met and the validation round can start.

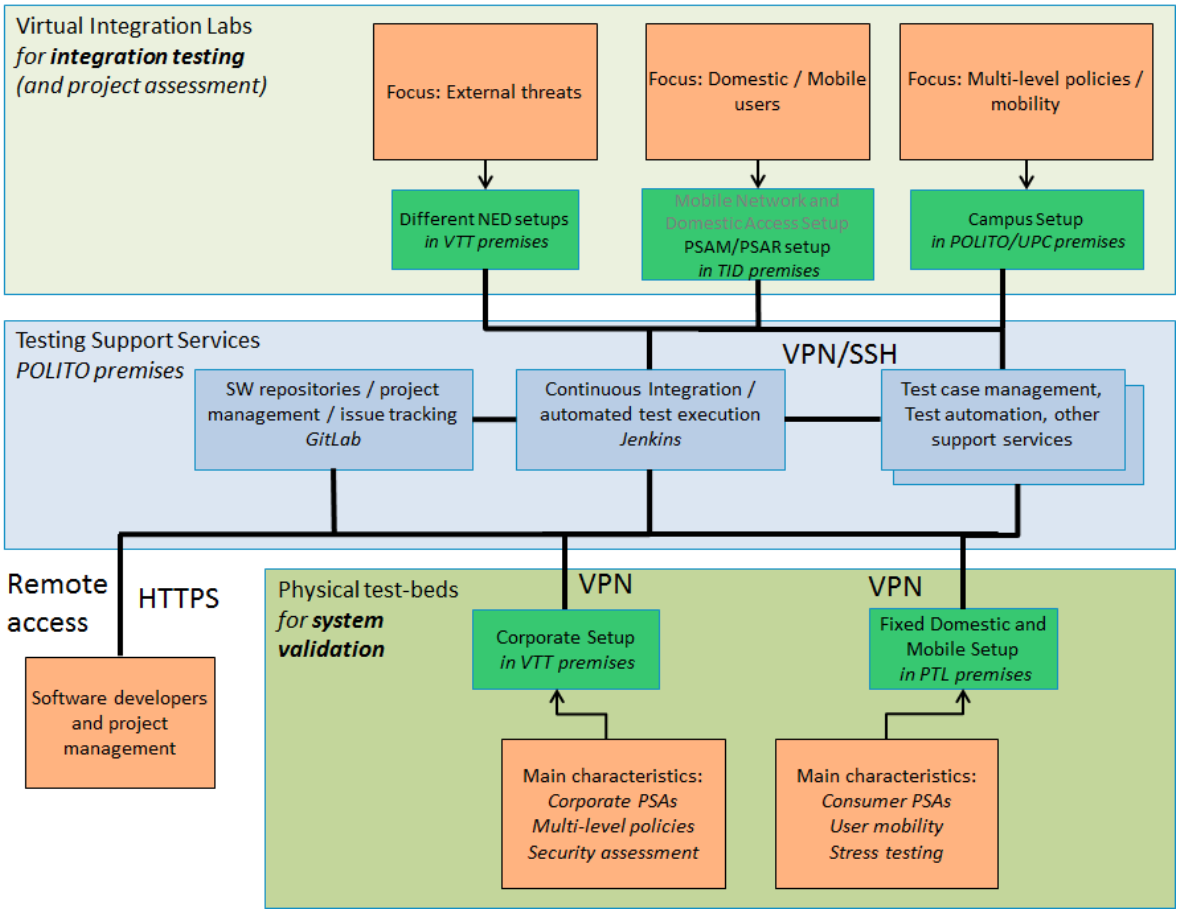

Figure 6: Integration labs and validation test-beds overview

The design of the virtual integration labs presented above only addresses the use case of various SECURED partners using the integration lab. Later in the project we might open parts of the integration lab also to external developers to get their feedback about the architecture and encourage the development of applications for SECURED.

3.1 Integration labs

Basic NED deployments and external threats

VTT will host an integration lab, depicted in Figure 7, that provides the bedrock for integration tests by making sure each selected NED deployment option (e.g. monolithic and virtual) is available and selected PSA deployment options are covered (e.g. virtual machine (VM) and container PSA models). Thus it can facilitate most of the integration tests.

The lab is connected to external testing support services, such as build and test automation, and to project management systems through a trusted channel (e.g. VPN and HTTPS). The NED is also connected to the Internet and user terminals, to enable system level integration testing where it is necessary to test the system in realistic end-user scenarios.

The special focus of this integration lab is the use of security testing tools to facilitate project assessment regarding the security of the developed system itself. Specifically stress testing and fuzzing capabilities are provided through open-source tools, such as those included in Kali Linux [5] distribution and Codenomicon Defensics fuzzing software [6].

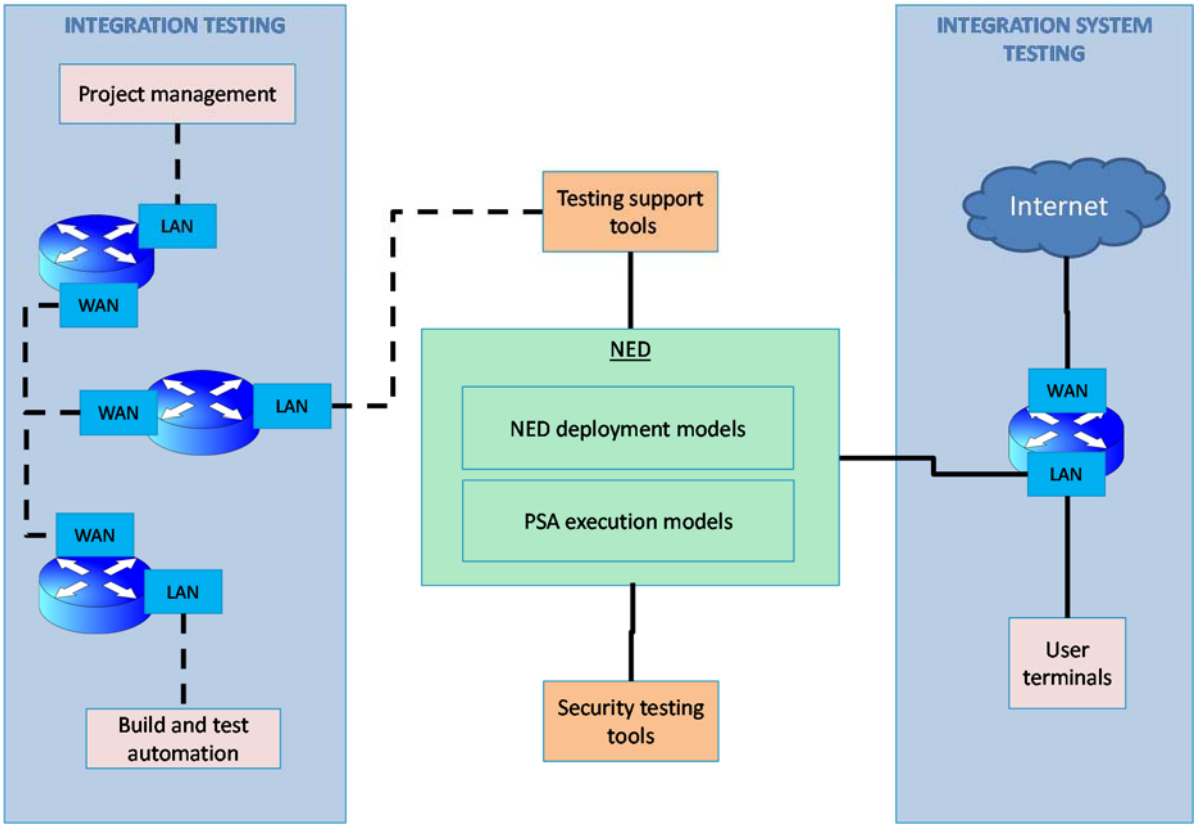

Figure 7: VTT integration lab overview

Domestic and Mobile Access

In order to execute integration test of the different access and SECURED applications from the point of view of an end-user, some equipment are located in TID premises. This is the available list:

- xDSL lines – Broadband Access lines based on xDSL technology.
- mobile access – real 3G/4G access will be not available in the integration test, but simulated mobile access based on WiFi. For an integration lab it is not necessary to have real radio access, so the use of Wi-Fi access points that simulate the low radio transport will be available.
- User terminals – Mobiles and laptops equipment are available, with different operating systems (e.g. Windows, Linux) to allow testing of the SECURED applications. Optionally, additional IoT devices like SmartTV can be added.

Figure 8 describes the designed integration lab. There is a Test Tool & Terminals where different end-user devices are hosted for mobile and domestic landline access. This equipment can be connected directly to external premises to offer IP connectivity to a NED outside of TID premises or can optionally be deployed in access network that connects the terminals to the SECURED. Also general-purpose servers will be available to instantiate a local NED as an option if the integration testing demands it. Finally the Gateway zone offers external connectivity (via Internet) to different parts of the integration network.

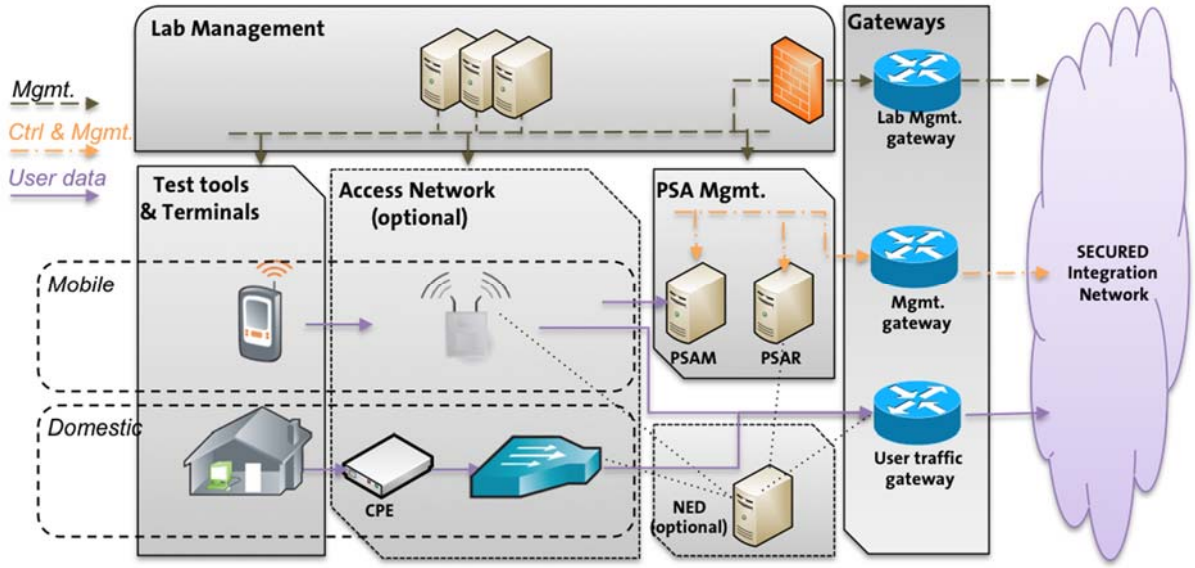

Figure 8: TID integration lab overview

PSA Management integration environment

As part of the TID integration lab, physical servers will be available available to host some PSA management tasks: specifically all integration tests that include the participation of the PSA Manager (PSAM) and PSA Repository (PSAR) described in D5.2 [7]. Capacity for different implementations of these elements is supported. For example, it is possible to implement a PSAM and PSAR for closed ecosystem and therefore will be only accessible through the integration Network. If an open ecosystem is tested, a controlled access via Internet will be opened using the Gateways. Figure 8 also depicts these components. PSAM and PSAR have access to all networks: user data to offer User Portal, Control & Management for downloading PSA to the NED, and the Lab management of the servers.

Campus setup

The integration lab setup hosted by POLITO is shown in Figure 9 and aims at validating the prototype during the various phases of the development, starting with the monolithic NED planned for the first release but with the support for more advanced configuration in the following phases of the project. The validation test-bed is installed in the "Networking and Multimedia" laboratory,

which allows about 15 students to simultaneously connect to the Internet through the SECURED service.

Particularly, users are connected to the switched LAN through wired or wireless connections. All the users' traffic towards Internet will be processed by the NED. The additional servers shown in the Figure 9 will be used to host services needed by SECURED, such as the PSA repository, the authentication service, and more. In the next phases of the project, a portion of those servers can be configured to act as a NED in order to validate the distributed case, or to host additional services (e.g. OpenStack controllers) for a possible prototype running in a cloud-based environment.

The network equipment available in the laboratory is rather standard and includes a Cisco 2950 Layer-2 switch for fixed access, a Linksys WRT54G configured as WiFi access point, and a Cisco 2911 router to connect to the University network. The network towards the user features 100 Mbps connectivity for the fixed part and 54 Mbps for the wireless. Instead, the NED and the additional servers are connected through a gigabit Ethernet internal network. The lab is connected towards the University via a gigabit link, while the entire University connects to the Internet through a 10 Gbps link.

Finally, all the servers are powerful workstations: an HP EliteDesk 8300, with a single CPU i7-4770 clocked at 3.4 GHz (4 physical cores with hyperthreading), 32 GB RAM DDR3-1600, and 500 GB HD (7200 RPM).

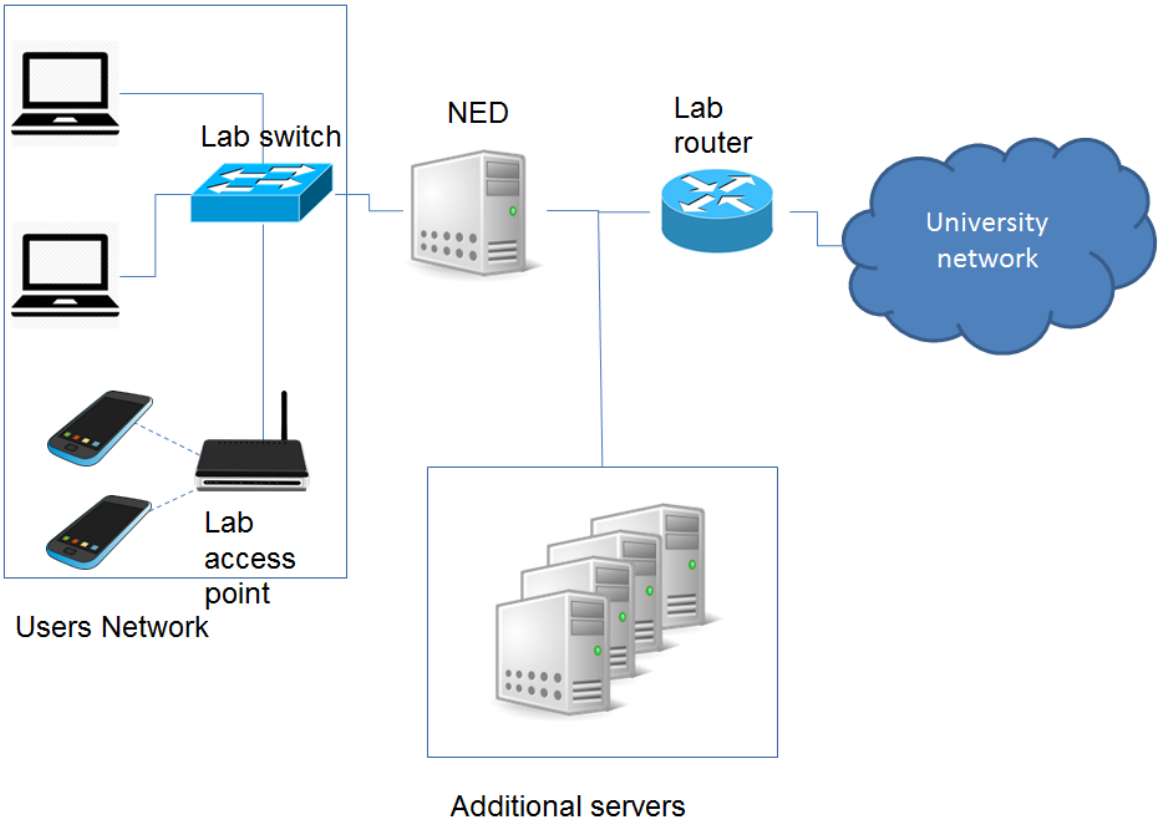

Figure 9: POLITO integration lab overview

3.2 Validation test-beds

The hardware used in the validation test-beds will mostly comprise of commodity desktop/server hardware, routers and switches. The test-beds and other hardware will be described in more detail in the following sub-sections. The hardware to be used in the full architecture test-beds will be decided later depending on the availability of hardware prototypes from WP3.

3.2.1 Corporate setup

The first test-bed is located at VTT premises and concentrates on corporate set-up. The purpose of the test-bed is to provide credible validation environment with real corporate end-users in a corporate network (in this case VTT researchers). Because we are validating an experimental security solution and because usually corporate networks have already high-end security measures as well as strict security policies in place, there has to be made compromises in order to utilize real corporate end-users conducting their day-to-day work. However, the possible test-bed architecture for corporate users, shown in Figure 10, is able to capture the most relevant aspects of the validation procedures efficiently.

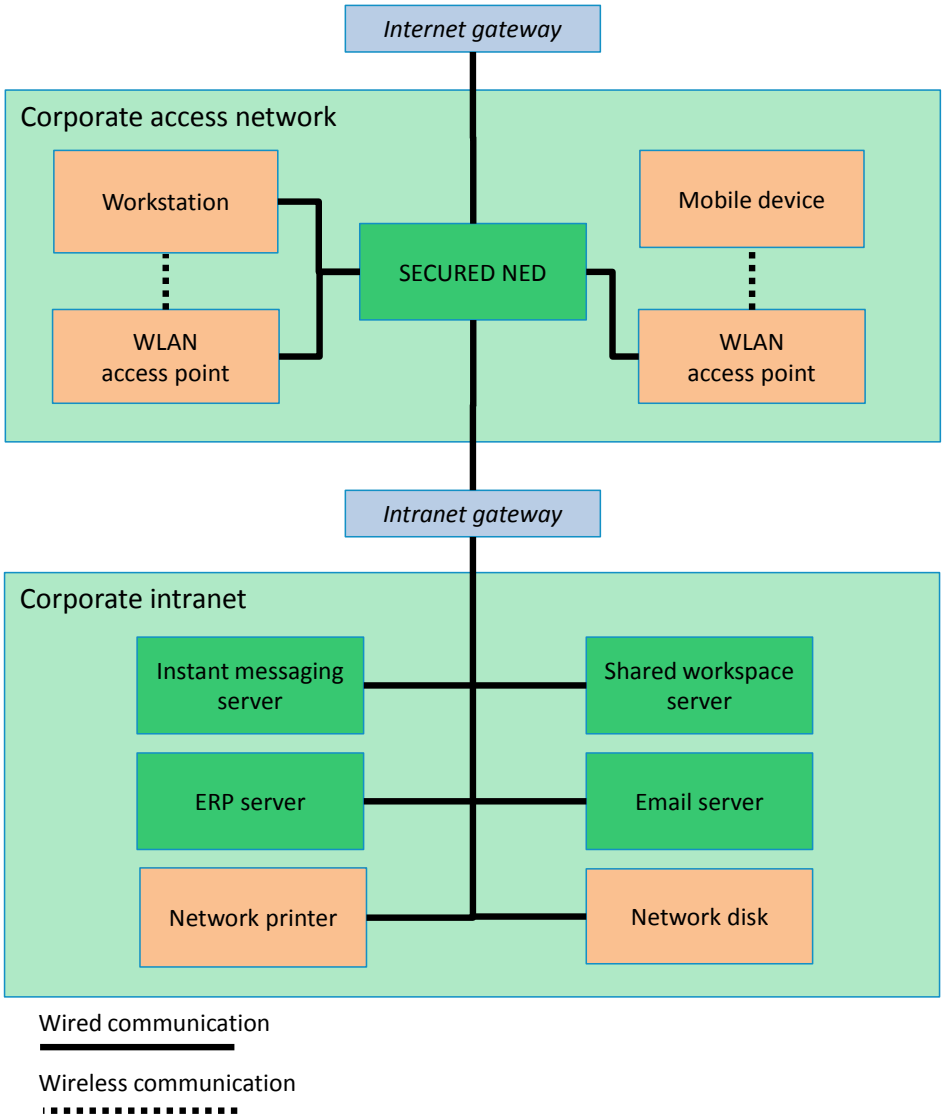

Figure 10: Validation test-bed architecture with corporate users

The day-to-day work done by the end-users during the validation are typical to corporate end-users. It includes usage of workstation in their office and mobile device in meeting rooms leading to both wired and wireless communication scenarios. The used corporate services include instant messaging (with potentially audio and video support) and shared workspaces for online collaboration between end-users, email and ERP (Enterprise Resource Planning) for interacting with production databases. In addition, the end-users utilize hardware network resources such as printers and external storages during the working day.

The most important NED architectural aspects covered by the test-bed are:

- all end-user traffic flows through the NED;

- secure attestation to the NED is possible through close localization of the NED and the user terminals;
- each end-user is able to use two different types of terminals (workstation and mobile device) and two different means of access (wired and wireless) to the NED, which enables validation of multi-device security policies;
- monolithic NED deployment.

There are multiple NED deployment options available described in D3.1.1 [8], but corporate security policies force us to deploy the NED outside the corporate intranet but inside corporate access network. This implies a limited coverage in terms of end-user using the test-bed but nevertheless the number of end-user must be low enough to enable efficient monitoring of the test-bed experiment execution under strict security requirements. An estimate of 10-20 end-users is assumed to be feasible.

A perimeter-based approach is governing the corporate network security domain. The corporate perimeter consists of security measures, such as firewalls, that protect the valuable assets, such as product design documents, located inside the perimeter. In our case the perimeter is two-level; the most valuable assets are located inside the intranet perimeters, whereas NED is located inside less strict perimeter, i.e. corporate access perimeter. The modern threat model in corporate networks is focusing on penetrating the perimeters starting from the weakest. Nowadays the mobile devices, that are often carried in and out the perimeter, offer suitable attack surface for the offenders. Therefore, PSAs would be needed to mitigate the threats caused by compromised end-user devices towards end-users and corporate network. Since all the client terminal traffic flows (when inside the perimeter) through the NED, implementing such PSAs is possible. One such potential PSA could implement access control based on device type, mobility and level of attestation.

3.2.2 Fixed domestic and mobile end-user

The second test-bed is located at PrimeTel premises and concentrates on fixed domestic and mobile users. Figure 11 illustrates PrimeTel's Triple play platform, called Twister, which is a converged end-to-end telecom platform, capable of supporting an integrated multi-play network for various media, services and devices. The platform encompasses all elements of voice, video and data in a highly customisable and upgradeable package. Together with VTT we estimate that around 50-100 end-users will be used within the two test-beds for the validation. For initial tests at Primetel we will utilise beta-testers, i.e. Primetel R&D staff member to test the functionality of the system. Once this is successful in the second phase we will ask additional employees who are also Primetel subscribers (the majority are) to support validation from other departments but also actual users if necessary to further test scalability, interoperability and other requirements. The PSAs to be tested will be selected at the beginning of the validation process. We will also provide an online service in the form of a questionnaire to obtain sufficient feedback on the user experience (QoE).

The IPTV streamers receive content from satellite, off-air terrestrial and studios and convert it to MPEG-2/MPEG-4 over UDP multicast, while Video on demand services are delivered over UDP unicast. Twister telephony platform uses Voice over IP (VoIP) technology. The solution is based on open SIP protocol and provides all essential features you expect from Class 5 IP Centrex softswitches. Media Gateways are used for protocol conversion between VoIP and traditional SS7/ISDN telephone networks. IP interconnections with international carriers are provided through international POPs. It also includes components that provide centralised and distributed traffic policy enforcement, monitoring and analytics in an integrated management system. Twister Converged Billing System provides mediation, rating and bill generation for multiple services. It maintains also a profile for each subscriber; Customer premises. The customer premises equipment (CPE) provides to customers Internet, telephony and IPTV connection. It behaves as an integrated ADSL modem, IP router, Ethernet switch and VoIP media gateway. STB receives multicast/unicast MPEG-2/MPEG-4 UDP streams and shows them on TV. Through a sonus interface and IP Connectivity the platform is linked to partner's 3G Mobile Network for offering IP services provisioning to mobile customers.

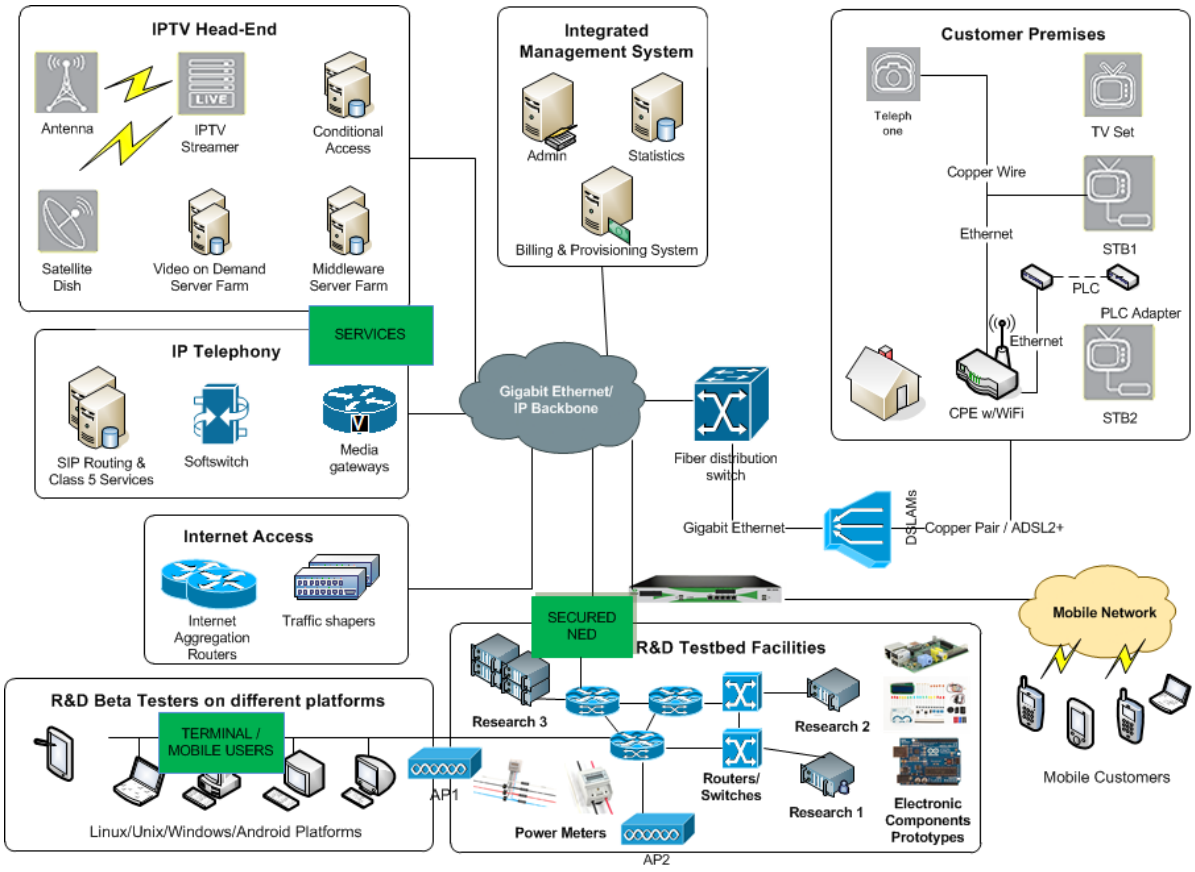

Figure 11: Validation test-bed architecture with fixed domestic and mobile end-users

PrimeTel's R&D test-bed facilities can connect to the company's network backbone and utilise the network accordingly. Through the R&D test-bed research engineers can connect to parts of interest on the real network. In collaboration with the Networks Department, R&D can conduct network analysis, traffic monitoring, power measurements etc. and also allow for testing and validation of newly introduced components as part of the its research projects and activities. A number of beta testers will be connected to the test-bed for supporting validation acting as real users and providing the necessary feedback of any proposed system, component or application developed.

Thus, deploying and validating the NED via PrimeTel's R&D test-bed will allow for the use of end user data traffic. Moreover, since a NED is a security product at a research level, using it in the frame of the R&D test-bed will not raise any concerns or legal and ethical issues which could be faced if it was deployed at PrimeTel's core network. The R&D test-bed will allow the assessment of a NED's resilience since it will handle the traffic of multiple users that will engage the NED to access the Internet for different purposes; providing this way a variety of traffic patterns. Moreover, end-users will be mobile as well as domestic thus the NED deployed in the R&D test-bed will be covering different types of devices. This will enable the deployment of PSAs that will be developed inside the frames of SECURED and will address the needs of single clients contrary to the corporate validation. For example, PSAs that fall under user security or privacy and network access control will be deployed.

4 Test cases, test scenarios and use cases

Integration tests are carried out according to the defined test cases. Test cases are created in the basis of the test scenarios that provide high-level description of conditions and interactions that are foreseen based on the system requirements. In turn, validation is carried out by end-users whose actions cannot be fully controlled. However, test support staff can manipulate the end-user behaviour, e.g. by limiting the use of client devices and spatial distribution of the network access points to make sure that essential use cases will be covered during the validation phase. Therefore, use cases provide high level description of conditions and interactions that guide validation execution planning.

The integration tests are executed according the integration test cases while the execution of system validation is based on the defined use cases. This section describes the initial test suite and/or test cases for both the integration and validation phases.

4.1 Integration test scenarios

The purpose of the integration test scenarios is to break down high-level use cases into more detailed scenarios. The integration scenarios take also into account D2.1 "User requirements" [11]. Table 1 provides the template for integration test scenarios. The integration test scenarios will be described in more detail in the internal web site. The following sub-sections describe the most relevant components for the basic and full architecture integration scenarios.

Table 1: Integration test scenario template

ID	**T_SC_1**
Name	Descriptive name of the scenario
Involved system components	1. Name of the component 2. …
Operating environments	1. NED types: (all, monolithic, …) 2. Execution environments: (all, virtual machine, …) 3. Network setup: (all, mobile, …) 4. …
Scenario description	Describe what kinds of interaction, events, inputs, outputs etc. are expected and some explanation of their purposes.
Reference to requirements	Describe which requirement this scenario originates from.

4.1.1 Basic architecture

The basic architecture description is given in D2.3.1 [9] and the NED alpha specifications in D3.1.1 [8]. A selection of system components to be tested is presented in Table 2.

Table 2: Basic architecture components for test scenarios

Component	**Test to be executed**	**Comments**
PSA	CTRL+MGMT API of the PSA	Test the interactions between commands given by the PSC to the CTRL+MGMT part of the PSA.
PSA	MSPL policy to specific configuration	Test the M2L policy translation and loading into specific PSA configuration.

PSC	Monitoring report	Test the health status and monitoring processes.
SECURED APP	End-user App processes for authN, authZ and remote attestation	Test the SECURED application in the user access to the system.
PSAM	PSA user portal	Test the basic functionality: authN, search and select by users.
SPM	Policy user GUI	Test the basic functionality: access, select policies and apply.
PSAR	PSA deployment process	NED requirements of info and images to download from PSAR.
PSAM	On-boarding process API in PSAM	Test the API to interact between a developer and the PSAM to upload new PSA in the platform.
PSAR	PSAM and PSAR interaction	Test the backend API of PSAR in the interactions with PSAM.
PSAR	SPM access to PSAR	Test the SPM modules that query information from PSA.

4.1.2 Full architecture

The full architecture beta description to be used for integration specifications will be finalized in M24 in D2.3.2. In addition to the components described in the previous section, we identify some specific scenarios to be considered for the full architecture. A selection of system components to be tested is presented in Table 3.

Table 3: Full architecture components for test scenarios

Component	Test to be executed	Comments
PSA	CTRL+MGMT	Test the CTRL+MGMT interface with PSC.
NED	Access clientless process	Test the access to the NED from a device without SECURED application in the end-user terminal.
User Profile Repository	User profile download	NED retrieves of the User Profile that includes, PSA, policies and service graph.
NED	AuthN, authZ	AuthN, authZ processes in distributed NED models.
PSAR	PSA deployment process in distributed repositories	Test the process of retrieving a PSA from a local PSAR.
NED	Deploy a PSA in distributed NED	Include two computer nodes to deploy different components of the NED, as orchestrator, PSC and TVD.

4.2 Integration test cases

Integration test cases are created according to the template in Table 4. The purpose and usage of each field in the template is described within. The template is designed to effectively reduce the

number of test cases through aggregation of alternative course of events thus reducing the redundancy (preconditions etc.) between test cases.

The test cases will be updated during the run of the project at the internal test management web site. Integration test cases formulated so far consider system level testing and the test cases are included in the Annex A. The defined tests are based on validation use cases that are presented in the next section. Therefore, top-down approach is used for defining the initial test cases based on the current knowledge from other deliverables. Later, we will specify the test cases based on the test scenarios, which can be formulated after we have the full specifications for each of the SECURED architectures.

Table 4: Integration test case template

ID	**I_TC_X.**
Name	Descriptive name of the test case
Reference to scenario	List the related scenarios
Reference to non-functional requirements	List the related non-functional requirements
Priority	(low, medium, high, optional)
Involved components	1. Software component 1 2. Software component 2 3. …
Preconditions	1. Precondition 1 (in order this test case to be performed, i.e. the state of the system before the first action in this test case is performed) 2. Precondition 2 3. …
Main postconditions	1. Postcondition 1 (state of the system after main course of events) 2. Postcondition 2 3. …
Main course of events	1. Event 1 (events such as user interaction, network events, inter-component messaging, etc. that form an ideal scenario) 2. Event 2 3. …
Alternative course of events	2a. Exception related to Event 2. 2a1. Alternative event taking place 2a2. e.g. Continue to Event 3 2b. Exception related to Event 2. 2b1. e.g. error message displayed (this would be an alternative postcondition; note that all postconditions are acceptable, i.e. system failures are not included in the test cases and all actions and events that are not defined in the test case are considered as system failures. System failures will cause test case to fail during test execution) 2b2. e.g. end test case 4a. Exception related to Event 4. 4a1. Alternative event taking place 4a2. …

The integration test cases include course of events that will cover all the specifications defined for each component's interfaces, e.g. D5.1 (PSA) [10] , D5.2 (PSAM and PSAR) [7], and SECURED architecture version as a whole, e.g. D2.3.1 Basic architecture specification [9], D2.3.2 Full architecture specification. Therefore, the test cases make sure that the interfaces are completely

implemented and they accomplish the task they were designed for. However, inconsistencies of interface implementations may be caused by multiple factors which stipulate intelligent design of test cases. The typical inconsistencies to be considered are:

- emergent performance and other non-functional issues caused by interaction between components (especially important when requirements such as component response times are not specified explicitly);
- side effects of components, such as those caused by resource usage that is not explicitly mentioned in the component specifications (e.g. conflicting temporary file names);
- parameter unit mismatches that arise from different interpretations of parameter specifications (e.g. confusing byte and bit quantities);
- violations of size limits (e.g. implicitly assuming range of values may cause buffer overflows).

For regression testing, the existing integration test cases will be primarily used. Test automation compatibility is one of the important factors to determine whether a test case is suitable for regression testing. However, additional regression test cases may have to be created and test engineer supervision needed to enable extensive coverage of the regression tests.

4.3 System validation use cases

As described in the DoW, testing/validation will evolve around three main use case scenarios, namely: fixed domestic scenario, mobile user scenario and corporate business client. The basic architecture description also defines a set of use cases in D2.3.1 [9], which can be later used to define more detailed use cases. The following subsections describe the currently defined use cases for both basic and full architecture that may be modified and possibly divided into more detailed use cases at the internal integration and validation web page. These use case scenarios aim to capture all addressed test cases described in more detail in Section 4.2 / Appendix A. Table 21 in Appendix A shows the coverage of user requirements (from D2.1 [11]) by the planned system level integration test cases. The entire list of user requirements is not covered by the currently planned system level integration test cases or they cannot be covered (e.g. USR.3: Cost). The use cases are described at a high-level, i.e. from the post-condition point of view.

Table 5 presents the current use case template, which will be used to describe different use cases in this deliverable. In the use cases it is assumed that all system components are available for the specific architecture version and the use case can be run on any NED, execution environments, network setups etc. (unless stated otherwise).

Table 5: Use case template

ID	**V_UC_X**
Name	Descriptive name of the use case
Use case description	Describe what kinds of interaction, events, inputs, outputs etc. are expected and some explanation of their purposes.
Source of the use case	Describe where this use case originates from (DoW, another deliverable, requirement, meeting, internal document, EC review recommendation, some standard, etc.)

4.3.1 Basic architecture

This section presents the current high-level use cases defined for the basic architecture in Table 6-Table 8. These use cases may be updated or divided into more detailed use cases.

Table 6: Use case 1 - Fixed Domestic Basic Scenario

ID	V_UC_1
Name	Fixed Domestic Basic Scenario
Use case description	This will be the most basic of the three scenarios where a user attempts to establish a secure connection at home. A home user will attempt to initiate a home connection for the first time and connect to his/her Internet service provider's (ISP) network. Initially, the locally placed NED, e.g. home NED installed and configured by the ISP, needs to be attested by means of a third party verifier. The user's device will need to be verified by the ISP's verifier which is installed remotely at a company's cloud of services or somewhere at the edge of the network handling a substantially large number of home users. The home user has requested for parental control to be activated as part of its security service provisioning by the ISP. A home user's child attempts to access undesired sites, e.g. a casino site but is blocked by the NED. The child then attempts to access his/her school's email account and is allowed to continue to the login page. This basic scenario will aim to test the operation of at least 1-2 basic PSAs. The case with multiple devices and a variety of operating systems is tested in the advanced Use Case 4.
Source of the use case	This use case is proposed in the DoW. It aims to cover the following Test Cases (partially derived from D2.2 [12]): I_TC_1, I_TC_3, I_TC_6.

Table 7: Use case 2 - Corporate Business Client Scenario (basic)

ID	V_UC_2
Name	Corporate Business Client Scenario (Basic)
Use case description	This use case introduces use of corporate security policies that are usually stricter than in other use cases. The focus is also in securing the corporate network, e.g. with access control, in addition to securing corporate end-users from malicious actions. The corporate use case also offers possibility for multiple actors, e.g. ICT managers, security managers and end-users, to define security policies that are applied at the same time. In addition, the end-users will conduct day-to-day work utilizing the corporate network resources such as ERP services and shared network resources such as network storages.
Source of the use case	This use case is proposed in the DoW. It aims to cover the following Test Cases (partially derived from D2.2 [12]): I_TC_1, I_TC_2, I_TC_3, I_TC_6.

Table 8: Use Case 3 – Roaming Mobile User Scenario (basic)

ID	V_UC_3
Name	Roaming Mobile User Scenario (Basic)
Use case description	This use case scenario considers a mobile user able to move between different wireless networks. The idea is to demonstrate that once the user moves and switches the NED (point of connection), his/hers security is migrated along with him/her. Thus, we foresee the case where different access points with SECURED capabilities are configured and an end-user mobile device, equipped with the SECURED app, moves around the campus while his/her security is maintained close to him/her. Furthermore, we can emulate the case that each access point is a different domain, which may result in the enforcement of different policies, i.e. at destination the policy stack for the user is different than the one at origin, thus policy reconciliation is required.

	In this use case scenario a user will first establish a connection through his/her terminal (e.g. a laptop) at home and then with the same device move to work and continue his/her work activities at his/her office at the company's premises. This Use Case will incorporate aspects from the previous two use cases. The user will hence change NEDs while moving from home to the company environment and hence will be constrained under a different policy set. This case aims to validate situations where a roaming user could change environment that maintain different policies and policy conflicts may occur. In this case we have to redo the attestation process.
Source of the use case	This use case is proposed in the DoW. It aims to cover the following Test Cases (partially derived from D2.2 [12]): I_TC_1, I_TC_2, I_TC_3, I_TC_4, I_TC_6.

Table 9 presents a comparison of which test case covers which use case in the basic (alpha) architecture.

Table 9: Test Cases covered by which Use Case

	V_UC_1	V_UC_2	V_UC_3
I_TC_1	●	●	●
I_TC_2		●	●
I_TC_3	●	●	●
I_TC_4			●
I_TC_5			
I_TC_6	●	●	●
I_TC_7			

4.3.2 Full architecture

This section describes the uses cases for validation of the the full architecture (Table 10-Table 12). It should be noted that these use cases may evolve over time to reflect the specifications of the full architecture.

Table 10: Use Case 4 – Fixed Domestic Customer Scenario (advanced)

ID	**V_UC_4**
Name	Fixed Domestic Advanced Scenario
Use case description	This use case will build on Use Case 1 by testing for multi-tenancy inside a NED and when multiple users share a PSA. Perform tests with different number of users and PSAs. For different applications, performance and Quality of Service (QoS) will be checked. Moreover, a user using a variety of devices, platforms and operating systems will be used to validate interoperability.
Source of the use case	This use case is proposed in the DoW. It aims to cover the following Test Cases (partially derived from D2.2 [12]): I_TC_1, I_TC_3, I_TC_6.

Table 11: Corporate Business Client Scenario (advanced)

ID	**V_UC_5**
Name	Corporate Advanced Scenario

Use case description	This case will extend Use Case 2 to include further tests when multiple users are connected, using various platforms, sharing specific PSAs, etc. In this use case a company employee will attempt to securely connect for the first time at the company's network.
Source of the use case	This use case is proposed in the DoW. It aims to cover the following Test Cases (partially derived from D2.2 [12]): I_TC_1, I_TC_2, I_TC_3, I_TC_4, I_TC_6

Table 12: Use Case 6 – Roaming Mobile User Scenario (advanced)

ID	**V_UC_6**
Name	Roaming Mobile User Scenario (Advanced)
Use case description	This use case scenario will build on Use Case 3 by testing for multi-tenancy inside a NED and when multiple users share a PSA. Perform tests with different number of roaming users using specific PSAs. For different applications performance and QoS will be checked. Moreover, users using a variety of devices, platforms and operating systems will be used to validate interoperability during mobility as well. As with Use Case 3, in this case we have to redo the attestation process.
Source of the use case	This use case is proposed in the DoW. It aims to cover the following Test Cases (partially derived from D2.2 [12]): I_TC_1, I_TC_2, I_TC_3, I_TC_4, I_TC_6

Table 13 provides an overview about which test case is covered by which use case in the full architecture.

Table 13: Test Cases covered by which Use Case

	V_UC_4	**V_UC_5**	**V_UC_6**
I_TC_1	●	●	●
I_TC_2	●	●	●
I_TC_3	●	●	●
I_TC_4	●	●	●
I_TC_5			●
I_TC_6	●	●	●
I_TC_7			●

5 Summary and conclusions

This document presented the overall design and setup for the integration and validation procedures for the SECURED project. The virtual integration lab design describes in a high-level how the different partner sites will be connected together to allow each partner to run tests in different labs to verify specific features, if necessary. Additionally, some parts of the platform, e.g. PSAR/PSAM, can be provided by a single partner and allow other partner sites to focus on the other parts of the SECURED platform, e.g. NED or other components. This allows us to flexibly run integration tests at different levels from integration test of individual components interfaces into system level integration tests, where we test most/full architecture of the SECURED platform as a whole before the actual end-user validation. The system integration tests will be run before the architecture version will be approved for validation round for real end-users. The system integration tests will run the whole SECURED platform and test most components/the whole system in scenarios to fulfil the use cases defined for the specific validation run. After the defined system integration test scenarios are passed, the architecture can be frozen and passed to the system validation, which will be run in the two validation test-beds.

The high-level use cases based on the main use cases defined in the SECURED DoW have been presented as the validation use cases for the actual end-user validation rounds. While defining the system integration test cases, we aligned the test cases with the ones defined in D2.2 - Project assessment to assist the project assessment with technical test cases when applicable.

The detailed descriptions about the workflow/process of submitting components/source code into review are presented in the live web version at the internal project site. Additionally, the detailed test plans and test scenarios for the basic and full architecture integration and validation will be defined in the private live web version mentioned above, because the specifications for the basic (alpha) architecture will be finalized alongside this deliverable, and we do not have the finalized specifications before that. Also, defining the more detailed plans for integration and validation at a private live web version allows us to actually update our plans in an agile manner in the runtime of the project to reflect to possible updated specifications.

6 References

[1] IEEE Standard for Software and System Test Documentation, IEEE Std 829-2008, July 2008

[2] M.Pezzè, M.Young, "Software testing and analysis: process, principles and techniques", John Wiley and Sons, 2008

[3] "GitLab" [online] https://about.gitlab.com/

[4] "Jenkins CI" [online] http://jenkins-ci.org/

[5] "Kali Linux" [online] http://www.kali.org/

[6] "Codenomicon Defensics Traffic Capture Fuzzer" [online] http://www.codenomicon.com/defensics/traffic-capture-fuzzer/

[7] SECURED consortium, "D5.2 – Specification of PSA management service and repository", September 2014

[8] SECURED consortium, "D3.1.1 – NED specification (alpha preview)", 2014

[9] SECURED consortium, "D2.3.1 – Specification of the SECURED architecture (Alpha version)", September 2014

[10] SECURED consortium, "D5.1 – Specification of PSA and associated data and interfaces", June 2014

[11] SECURED consortium, "D2.1 – User Requirements", January 2014

[12] SECURED consortium, "D2.2 – Project assessment metrics and procedures", June 2014

7 Abbreviations

CI	Continuous Integration
ERP	Enterprise Resource Planning
HSPL	High-Level Security Policy Language
IoT	Internet of Things
ISP	Internet Service Provider
I&V	Integration and Validation
NED	Network Edge Device
OS	Operating System
POP	Point of Presence
PSA	Personal Security Application
PSAM	Personal Security Application Manager
PSAR	Personal Security Application Repository
PSC	Personal Security Controller
SCM	Source Code Management
SECURED	Security at the Network Edge
SPM	Security Policy Manager
TVD	Trusted Virtual Domain
VoIP	Voice over IP
QoS	Quality of Service
VM	Virtual Machine

Appendix A. Initial system level integration test cases for basic and full architecture and a table of user requirement coverage by the defined test cases

Table 14: System level integration test case I_TC_1

ID	I_TC_1
Name	Network Edge Device (NED) Functional Operation
Reference to use case	Covered by V_UC_1, V_UC_2, V_UC_3, V_UC_4, V_UC_5, V_UC_6
Priority	High
Involved components	System level testing component setup
Preconditions	1. Secured App configured on User Terminal 2. Monolithic NED configured at an on-path edge device 3. PSC Manager (PSCM) is set 4. PSC at the NED for the user is configured 5. PSAs at the NED which the user will test are configured
Main postconditions	1. Once the user PSAs are setup and configured the user should be in position to establish corresponding secure sessions successfully with the appropriate security applications accordingly 2. After the user establishes connection with the NED and the selected PSAs, all associated traffic will pass through the NED.
Main course of events	1. User authentication and Attestation a. Test User Credentials b. Successful connection established 2. Setup of New User a. Handle New User b. Retrieve User Specific data, e.g. profile c. Setup New User Trusted Virtual Domain (TVD) , which includes his/hers PSC d. Request User Policies e. Setup and Configure User PSAs
Alternative course of events	1. Exception related to Event 1 in case of Authentication Failure (with non-legit credentials) returning Authentication Unsuccessful message. 2. Exception related to Event 2.b. when attempting to retrieve user profile. No profile found (when there is no such profile existing). 3. Exception related to Event 2.c. Error when setting up a new user's TVD and PSC (in case capacity of the NED is exceeded) 4. Exception related to Event 2.d. when requesting User Policies. No associated policies found. 5. Exception related to event 2.e. Error when configuring User PSA (when the PSA configuration is non-legit)

Table 15: System level integration test case I_TC_2

ID	I_TC_2
Name	Multi-tenancy inside a NED and Multiple users share a PSA
Reference to use case	Covered by use cases V_UC_2, V_UC_3, V_UC_5, V_UC_6

Priority	High
Involved components	System level testing component setup
Preconditions	1. PSC at the NED for the user is configured 2. PSAs at the NED are configured with shared PSAs
Main postconditions	1. Multiple users are connected to the NED. 2. Shared PSAs operate without disturbing performance costs.
Main course of events	1. User 1 connects to NED 2. User 2 connects to NED 3. User N sends a disconnection request to NED 4. Disconnection Successful reply
Alternative course of events	1. Exception related to Event 1 User 1 fails to connect … 2. Exception related to Event 2 User 2 fails to connect … 3. Exception related to Events 3 and 4 fails to disconnect … 4. In case of policies conflict users should be informed …

Table 16: System level integration test case I_TC_3

ID	**I_TC_3**
Name	Interoperability – Supporting various platforms
Priority	Medium
Involved components	System level testing component setup
Reference to use case	Covered by V_UC_1, V_UC_2, V_UC_3, V_UC_4, V_UC_5, V_UC_6
Preconditions	1. Setup a test-bed with users having different devices, etc. 2. Test-bed could include same devices with different OSs.
Main postconditions	1. Security connections established for all tested PSAs for different devices and different OSs.
Main course of events	1. Test interoperability when different user devices are used: • smart-phones • desktop computers • tablets • laptops • [OPTIONAL] IoT devices (e.g. SmartTV, wearable, sensors, …) 2. Interoperation of the NED when deployed in various networking environments … a. Can successfully be installed and configured in different technological environments
Alternative course of events	1. Exception related to Event 1. Device not interoperable. Some devices fail. a. Reconfiguration of NED. 2. Exception related to Event 2. OS not interoperable. Some OSs fail. a. Reconfiguration of NED.

Table 17: System level integration test case I_TC_4

ID	**I_TC_4**
Name	Policy conflict and resolution in the presence of multi-profile security configurations

Priority	Medium
Involved components	System level testing component setup
Reference to use case	Covered by V_UC_2, V_UC_3, V_UC_5, V_UC_6
Preconditions	1. Capability section of the PSA manifest is configured.
Main postconditions	1. New policies are added to the corresponding component for future use. 2. No conflict of policies status should be the state.
Main course of events	1. Application-driven approach where the end user directly specifies the security policies and selects the PSAs to enforce them. a. E.g. a user wants to enforce a kind of parental control: "do not connect to specific social networking sites" b. E.g. a manager wants to enforce a kind of access control on his employees: "do not connect to this specific intranet sites" 2. Policy-driven approach where the user solely specifies the security policies he wants to be enforced and SECURED identifies the most suitable PSAs. a. E.g. a user wants to enforce a more general kind of parental control: "do not connect to all social networking sites", "block all sites with adult content", block betting sites, block online casino sites, etc. 3. In case of change of High-Level Security Policy Language (HSPL), e.g. by a company. The expected results are that in all cases the policies are enforced accordingly and successfully.
Alternative course of events	1. More than one PSA match all the criteria specified. a. The User will have to define some criteria based on which the final PSA selection will be made by SECURED (e.g. maximise performance, use only open-source PSAs, use free PSAs)

Table 18: System level integration test case I_TC_5

ID	**I_TC_5**
Name	Mobile and nomadic users
Priority	Low
Involved components	System level testing component setup
Reference to use case	Covered by V_UC_3, V_UC_6
Preconditions	1. User security policies are set up that allow mobility between NEDs 2. Secured App configured on User Terminal 3. NEDs are setup to work in mobility support mode
Main postconditions	Handover between NEDs is performed (handover time depends on whether the PSAs used are migration-aware and what kind of computational models / execution environments are used for the PSAs, e.g. VM or container)
Main course of events	4. User authenticates at old NED 5. User sends a Handover Trigger 6. Point of Presence (POP) coordination initialisation send between NEDs 7. Attestation request from one NED to the next 8. Data forwarded from one NED to another as context

	9. Migration of App State Containers or VMs 10. New NED sends user policy request to PSAR/PSAM/SPM
Alternative course of events	N/A

Table 19: System level integration test case I_TC_6

ID	**I_TC_6**
Name	PSA performance and chaining
Reference to use case	Covered by V_UC_1, V_UC_2, V_UC_3, V_UC_4, V_UC_5, V_UC_6
Priority	High
Involved components	System level testing component setup
Preconditions	1. Several environment conditions must be applied, for example: a. Same application code. Use the same execution environment and versions. b. Similar computing capacity. Try to work with similar CPU, memory and network processors.
Main postconditions	1. PSAs will pass performance tests. 2. PSAs will pass setup tests with various configurations and PSA chaining.
Main course of events	1. Test the security application as part of the end-user device. 2. Test the security application adapted to be a PSA inside a SECURED environment. 3. Perform test with different number of users and PSAs a. Successfully handled 4. Try with different user applications and check QoS a. Successfully handled 5. Analysis of the NED's attack surface and comparison with current client/server solutions. 6. Test that security applications can be ported to SECURED as a PSA. 7. Test that security application can be configured and controlled in a user friendly manner. 8. Users can define their own rules/limits for different devices. 9. Test that a number of different PSAs can be created for a particular PSC.
Alternative course of events	N/A

Table 20: System level integration test case I_TC_7

ID	**I_TC_7**
Name	Overall Project KPIs Evaluation
Priority	Low
Involved components	System level testing component setup
Reference to use case	Covered by V_UC_3, V_UC_6

Preconditions	1. All available NED deployment options and software components are set up. 2. Users' security policies are set up.
Main postconditions	1. Indications of improved efficiency, robustness and performance compared to client-side security application deployments are observed. 2. Indications of improved resource consumption compared to client-side security application deployments are observed.
Main course of events	1. Users are using multiple PSAs on various NED deployments and PSA configurations. 2. Resource usage and performance characteristics of the PSAs are compared to client-side security application deployments.
Alternative course of events	N/A

Table 21: User requirements covered by the planned system level integration test cases

	I_TC_1	I_TC_2	I_TC_3	I_TC_4	I_TC_5	I_TC_6
USR.1					●	
USR.2	●	●	●	●	●	●
USR.3						
USR.4	●	●	●	●	●	●
USR.5	●	●	●	●	●	●
USR.6	●	●	●	●	●	●
USR.7						
ICT.1					●	
ICT.2	●	●	●	●	●	●
ICT.3	●	●	●	●	●	●
ICT.4	●	●	●	●	●	●
ICT.5						
ICT.6	●	●	●	●	●	●
ICT.7	●	●	●	●	●	●
ICT.8	●	●	●	●	●	●
DEV.1			●			
DEV.2			●			
DEV.3						
DEV.4	●	●	●	●	●	●
DEV.5		●	●			
DEV.6		●	●			●
DEV.7			●			●
DEV.8	●	●	●	●	●	●
ISP.1						
ISP.2	●	●	●	●	●	●
ISP.3		●				
ISP.4	●	●	●	●	●	●
ISP.5	●	●	●	●	●	●
ISP.6	●	●	●	●	●	●
ISP.7	●	●	●	●	●	●
SP.1		●	●			●
SP.2	●	●	●	●	●	●

SP.3					●	
SP.4						
SP.5	●	●	●	●	●	●
SP.6						
SP.7	●	●	●	●	●	●
MISC.1		●	●			
MISC.2						